The Distributed Multiple Voting Problem

Florence Bénézit, Patrick Thiran, and Martin Vetterli

***Abstract*—A networked set of agents holding binary opinions does not seem to be able to compute its majority opinion by means of local binary interactions only. However, the majority problem can be solved using two or more bits, instead of one (F. Bénézit *et al.*, "Interval consensus: From quantized gossip to voting," Apr. 2009, pp. 3661–3664). Pairs of agents asynchronously exchange their states and update them according to a *voting automaton*. This paper presents binary voting automata as well as solutions to the multiple voting problem, where agents can vote for one candidate among $|\mathscr{C}| \geq 2$ candidates and need to determine the majority vote. The voting automata are derived from the pairwise gossip algorithm, which computes averages. In the binary case ($|\mathscr{C}| = 2$), we focus on averages in dimension 1, but in the multiple case ($|\mathscr{C}| \geq 2$) we quantize gossip in dimension $|\mathscr{C}| - 1$, which is larger than or equal to 1. We show in particular that a consensus on majority can be reached using 15 possible states (4 bits) for the ternary voting problem, and using 100 possible states (7 bits) for the quaternary voting problem.**

***Index Terms*—Density classification, distributed estimation, gossip algorithms, voting problem.**

I. Introduction

ALTHOUGH local interactions rule our physical world, they are believed to have strong limitations because a networked set of agents holding binary opinions does not seem to be able to compute its majority opinion by means of local binary interactions only. This is intriguing because majority is the simplest global piece of information that one can retrieve from a binary system: Are there more positive opinions than negative opinions? This question requires a simple yes-or-no answer, and yet, so far, no binary distributed scheme has managed to drive all the agents to reach a consensus on majority.

In this paper, we consider a finite set $\mathcal{V}$ of agents connected in an undirected graph $G = (\mathcal{V}, \mathcal{E})$, which is possibly time-varying. Each agent initially votes for a "color" among a finite unordered set $\mathscr{C}$ of colors. By means of local communications only, and using constant, identical, and simple updating rules, the agents want to distributively reach a state of consensus indicating the initial majority color. Agents do not know the total number of agents nor the network topology. So far, this problem has appeared in its binary version ($|\mathscr{C}| = 2$), under two names such as density classification task or voting problem. In this case, the two colors are often denoted by 0 and 1. We start by showing the close link between the voting problem and gossip algorithms.

A. From Gossip to Interval Consensus and Binary Voting

In the past years, algorithms called gossip were developed to compute in a distributed way the average of values disseminated in a network. *Pairwise gossip* [2], the most famous gossip algorithm, randomly and iteratively computes local pairwise averages until the estimates at each node reach some level of precision. Although gossip algorithms converge to the average, the states of the nodes will be equal to the true average only asymptotically. At finite times nodes hold only *approximate* values of the average, which differ in general for each node. As a consequence, an exact consensus cannot be reached in finite time.[1] Yet, in any real problem, the states of the nodes must be read after a finite time.

In some scenarios however, such as sounding an alarm as soon as the mean temperature exceeds some threshold, nodes must take a common decision knowing which *interval* contains the average, but do not require to know the *exact value* of the average. In these coordinated decision making scenarios, classical gossip algorithms are not an ideal solution, in particular when the average is close to the boundary between two consecutive intervals. In this case indeed, stopping the algorithm after a finite time will result in having some nodes holding estimates in one interval, whereas some other nodes hold estimates in the adjacent interval. This leads to rather inconvenient, contradictory, and possibly erroneous decisions.

A first solution to this problem is to enforce consensus at finite time at the price of some error on the output consensus value. Kar and Moura [3] and Aysal, Coates, and Rabbat [4] designed probabilistic algorithms that are able to reach a true consensus in finite time. In [4], it is shown that if nodes use probabilistic quantization at each iteration, then all the states converge to a common but random quantization level. These probabilistic algorithms compute an unbiased estimate of the average of the initial data, but with a nonzero variance error; hence, the correctness of the result is not guaranteed.

To solve this correctness issue, in this paper, we quantize gossip such that nodes can reach a consensus on the interval in which the average lies, rather than on the average itself [1]. Our quantization idea resembles the work of Kashyap, Başar, and Srikant in [5], where the authors suggested an average consensus algorithm over integers, which is a quantized version of pairwise gossip. In their setting, nodes initially measure

Manuscript received July 16, 2010; revised December 03, 2010; accepted January 25, 2011. Date of publication February 14, 2011; date of current version July 20, 2011. This work was supported in part by the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center supported by the Swiss National Science Foundation under Grant 5005-67322. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Anna Scaglione.

F. Bénézit is with the TREC, ENS-INRIA, Paris 75013, France.

P. Thiran and M. Vetterli are with the School of Informatique and Communications, EPFL, Lausanne CH-1015, Switzerland.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JSTSP.2011.2114326

[1]Except for some rare cases, e.g., the agents start from a consensus already, or the number of nodes n is a power of 2, etc.

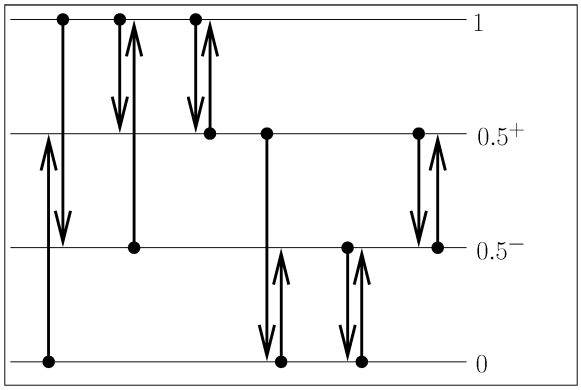


Fig. 1. Two-bit voting automaton.

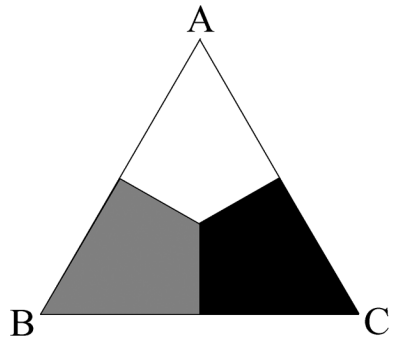


Fig. 2. Ternary voting problem: three majority zones.

some integer values, and the two integers framing the average of these values are denoted by L and $L+1$. When convergence is reached, some nodes have state L, the remaining nodes have state $L+1$, and the *overall average is preserved*. This quasi-consensus was called "quantized consensus." By changing the set of quantized states, we are able in this paper to transform "quantized consensus" into "interval consensus."

Our algorithm, called interval consensus gossip [1], can be directly applied to solve the binary voting problem. If votes are denoted by 0 and 1, and if the number n of nodes is odd, the voting question can be restated as: "Is the average of votes in the interval [0, 0.5] or in the interval [0.5, 1]?" Answering this question with a regular gossip algorithm would require to code the estimates on $\log n$ bits in order to deal with the case where there are $\lfloor n/2 \rfloor$ zeros and $\lceil n/2 \rceil$ ones. Happily, with interval consensus gossip, two coding bits are sufficient.

The two-bit voting algorithm works as follows. The four states are denoted by 0, 0.5^-, 0.5^+ and 1. States 0 and 0.5^- decide majority 0 and states 0.5^+ and 1 decide majority 1. Nodes wake up iteratively uniformly at random and call a random neighbor. At each iteration, the pair of activated neighbors exchange their states and update them according to the automaton of Fig. 1: A node with state 0 and a node with state 1 will update to, respectively, state 0.5^+ and state 0.5^-, a node with state 1 and a node with state 0.5^- will update to, respectively, state 0.5^+ and state 1, etc. The reader can check that the average value of states is conserved through iterations (states 0.5^- and 0.5^+ both have value 0.5). Nodes with consecutive states swap their states, which allows information flow, and finally updated states are either consecutive or equal, which enforces convergence. A simple formal proof of convergence of the algorithm can be found in [1]. In this paper, we give a general proof of convergence for the multiple voting automata, which includes the simple binary case.

In the next section, we show how an extension of the interval consensus problem allows us to solve the multiple voting problem.

B. From Interval Consensus to Volume Consensus and Multiple Voting

To solve the multiple voting problem, we first restate it as a geometry problem, then we show that extending the interval consensus idea to higher dimensions could solve the latter geometry problem. By extension to higher dimensions, we mean that intervals become convex areas in 2-D (two dimensions), convex volumes in 3-D, and convex hyper-volumes in greater dimensions.

While the binary voting problem can be restated as an interval consensus problem, the ternary voting problem can be restated as an area consensus problem in 2-D. Associate votes A, B and C to each of the three vertices of an equilateral triangle (Fig. 2). Then the majority vote is determined by the position of the barycenter of the votes in either the white zone, the gray zone, or the black zone in Fig. 2. This barycenter idea is generalizable to any dimension, as formally explained in the next paragraph.

Notations: Let $S_{|\mathscr{C}|}$ be the regular simplex with $|\mathscr{C}|$ vertices (denoted by $O_1, \ldots, O_{|\mathscr{C}|}$), and G be their center of mass (or barycenter). For every $c \in \mathscr{C}$, vertex O_c is called the vertex of color c. Simplex S_1 is a point, simplex S_2 is a line, simplex S_3 is an equilateral triangle, simplex S_4 is a regular tetrahedron, etc. We denote by $\vec{e_c}$ the vector $\vec{e_c} = \overrightarrow{GO_c}$. By definition $\sum_{c=1}^{|\mathscr{C}|} \vec{e_c} = 0$. For any color $c \in \mathscr{C}$, let n_c be the number of agents that voted for c. Let B be the barycenter of the system $\{(O_1, n_1), \ldots, (O_{|\mathscr{C}|}, n_{|\mathscr{C}|})\}$. By definition, B is such that

$$\overrightarrow{GB} = \frac{\sum_{c=1}^{|\mathscr{C}|} n_c \vec{e_c}}{\sum_{c=1}^{|\mathscr{C}|} n_c}.$$

We claim that the position of the barycenter B contains the majority vote information. Indeed, the following lemma shows that there is a bijective mapping between the barycenter points in $S_{|\mathscr{C}|}$ and the vote ratio configurations $(n_1/n, \ldots, n_{|\mathscr{C}|}/n)$, where $\sum_{c=1}^{|\mathscr{C}|} n_c = n$.

Lemma I.1 (Center of Mass Mapping or Caratheodory Theorem): For any point $B \in S_{|\mathscr{C}|}$, there is a unique sequence $(\lambda_1, \lambda_2, \ldots, \lambda_{|\mathscr{C}|}) \in [0,1]^{|\mathscr{C}|}$, such that $\sum_{c=1}^{|\mathscr{C}|} \lambda_c = 1$ and

$$\overrightarrow{GB} = \sum_{c=1}^{|\mathscr{C}|} \lambda_c \vec{e_c}.$$

Proof: Suppose that there were two such sequences $(\lambda_1, \lambda_2, \ldots, \lambda_{|\mathscr{C}|})$ and $(\mu_1, \mu_2, \ldots, \mu_{|\mathscr{C}|})$. Then,

$$0 = \sum_{c=1}^{|\mathscr{C}|} (\lambda_c - \mu_c)\vec{e_c} = \sum_{c=1}^{|\mathscr{C}|} (\lambda_c - \mu_c)(\vec{e_c} - \vec{e_1}) = \sum_{c=2}^{|\mathscr{C}|} (\lambda_c - \mu_c)(\vec{e_c} - \vec{e_1}).$$

By definition of volume, the determinant $det((\vec{e_2} - \vec{e_1}), (\vec{e_3} - \vec{e_1}), \ldots, (\vec{e_{|\mathscr{C}|}} - \vec{e_1}))$ is not equal to zero which means that the vectors $(\vec{e_2} - \vec{e_1}), (\vec{e_3} - \vec{e_1}), \ldots, (\vec{e_{|\mathscr{C}|}} - \vec{e_1})$ are linearly independent. Therefore, for any $c \geq 2$, $\lambda_c = \mu_c$, but $\sum_{c=1}^{|\mathscr{C}|} \lambda_c = \sum_{c=1}^{|\mathscr{C}|} \mu_c = 1$; hence, $\lambda_c = \mu_c$ for every $c = 1, \ldots, |\mathscr{C}|$. ∎

The regularity of $S_{|\mathscr{C}|}$ is not used in the proof of Lemma I.1, but it gives a very simple criterion to find the majority color $\arg\max_i n_i$: it is the color corresponding to the vertex of $S_{|\mathscr{C}|}$ which is the closest to the barycenter B of the system $\{(O_1, n_1), \ldots, (O_{|\mathscr{C}|}, n_{|\mathscr{C}|})\}$.

Core Idea of This Paper: The first idea that would come naturally to mind after the observations made from Lemma I.1 is to design the following algorithm:

1) Assign to each agent the vertex O_c corresponding to its voting color c.
2) Have the agents compute distributively the center of mass of the assigned vertices, i.e., the barycenter B of the system $\{(O_1, n_1), \ldots, (O_{|\mathscr{C}|}, n_{|\mathscr{C}|})\}$.
3) Have the agents compute locally the vertex $O_{C_{\text{maj}}}$ which is the closest to B, and output C_{maj} as the majority color. If there are several such vertices $O_{C_{\text{maj}}}$, then the agents deduce that there is a tie between the corresponding set of colors.

Pairwise gossip for average consensus [2] is a distributed algorithm which computes averages with real states; when two agents i and j exchange their states x_i and x_j, they both update their state to $(x_i + x_j)/2$. This procedure conserves the global average, and with time, the states in the network approximate the true average up to any desired level of precision, if the network is jointly connected in time. This continuous-valued algorithm is obviously generalizable to multiple dimensions and, in theory, it could be used to achieve the second step of the above algorithm.

This first idea, however, does not work well for the same reason as in the 1-D case (Section I-A). When votes are close to a tie, the barycenter B is close to several vertices. At finite times, the agents running gossip hold different estimates of B and do not agree on a single value, which can lead to different color outputs. A second issue is that pairwise gossip needs to be quantized in practice, and a careless quantization could lead to larger imprecisions and make the first issue worse. In this paper, we derive quantized versions of pairwise gossip using a finite number of states, such that based on its state, an agent can eventually tell which is the closest vertex to the center of mass. Or, in words adapted to the interval/volume consensus framework, instead of computing the barycenter B of votes itself, agents can decide in which (convex) majority hyper-volume B is located. *Our generalization includes the 2 bit automaton*, and necessitates more sophisticated notations and tools to prove convergence than in [1]. This explains why the notations in this paper significantly differ from the ones used in [1].

More precisely, we will derive a list of sufficient conditions for an algorithm to converge to the majority vote. These conditions will allow us to claim that our automata converge correctly. Checking that a given automaton fulfills the sufficient conditions is easy, but constructing an adequate automaton is challenging. We managed to build correct automata for the binary, ternary, and quaternary cases, but constructing them for arbitrary sets $\mathscr{C}$ remains an open problem.

It is important to mention that the *robustness* of gossip algorithms against the loss of a message or of an agent remains present in our generalization. In gossip, if a node transforms by mistake its state x in a state y, then the error induced in the computed barycenter $\tilde{B}$ compared to the correct barycenter B is $\|B - \tilde{B}\|_1 = |x - y|/n$. This error is small when the network size n is large and it has no consequence if $\tilde{B}$ stays in the correct majority hyper-volume, which is likely to happen if the votes are far enough from a tie. If too many errors add up, then the algorithm looses track of the initial barycenter. It is thus robust to some amount of message or agent losses, which depends on n, on the initial barycenter B and on the specific automaton that is being used.

C. History of the Battles to Solve the Binary Voting Problem

The binary voting problem, particularly popular among the cellular automaton (CA) community, is well-known for its contrast between its simplicity of statement and its difficulty to be solved. Before [1], research has unsuccessfully focused on automata coded on 1 bit. This section summarizes the years of work on the problem and highlights the key ingredients that a solution needs and that are possible to gather by coding states on 2 bits instead of 1.

In [6], Land *et al.* showed that a synchronous deterministic two-state automaton, which always successfully classifies density in a connected network, does not exist. Several binary automata were tentatively designed on the ring network, the most successful one being able to solve the problem in at most 83% of the initial voting configurations [7]. The behavior of the simplest automaton, which applies the local majority rule, has been carefully studied [8], [9]. In the local majority rule, agents update their state to the majority state of their neighborhood. The inability of this rule to lead all the states to a consensus constitutes its major flaw.[2] For example, consider the line topology with initial configuration (0, 0, 1, 1, 1). This configuration does not evolve with time, because the second agent sees (0, 0, 1) and keeps state 0, and the third agent sees (0, 1, 1) and keeps state 1. Additionally, even when a consensus is reached, the bit on which agents agree is not necessarily the initial majority bit. As a consequence of this failure, research has tried to avoid frozen

[2]On a positive note, this lack of convergence to a consensus can be used to detect communities in large-scale networks. See label propagation algorithms [10].

situations as (0, 0, 1, 1, 1) with the local majority rule, and CAs were developed with a crucial new feature: *information flow*.

For example, by slightly modifying the problem setting, two exact solutions were found on the ring network [11], [12]. Both are based on the traffic flow rule, also called 184 rule, in which the number of 0 s and 1 s stays unchanged, while the 0 s and 1 s interlace with time, until the minority bits are systematically surrounded by 2 majority bits. Inevitably, some neighboring agents have common majority bit states. Both solutions are based on simple observations. In the first solution, at each iteration, this interlaced configuration shifts by one position in the network so that all the agents eventually discover two consecutive identical bits, which are equal to the majority bit. In the second solution, the network switches to the local majority rule, which is successful on any interlaced configuration reached by rule 184. Both solutions confirm the power of information flow in the density classification task. Furthermore, they highlight the importance of *conserving the global information* through iterations. In [13], Capcarrere and Sipper proved that density conservation is a necessary condition for a CA to be successful. Despite their success on rings, neither solutions however are extendable to arbitrary graph topologies, so the research effort was devoted to design rules adapted to any network, focusing back on bringing states to a consensus. To avoid the impossibility result [6], asynchronous and probabilistic automata were proposed [14]. An automaton is used asynchronously when nodes wake up at random times (usually exponentially distributed), communicate with their neighbors and update their state according to the automaton. The order in which nodes update their state is thus random, which could appear as an additional difficulty [15]. Most probabilistic automata are run synchronously. Examples of probabilistic automata converging to a consensus are numerous; in probabilistic polling, nodes update their state to the state of a randomly chosen neighbor. If the probabilities are well tuned, this algorithm can reach proportionate agreement: the probability of reaching a consensus on 1 is equal to $\rho(1)$ (the initial density of 1 s), and consensus on 0 has probability $\rho(0)$. In [16], Fuks designs a probabilistic automaton on the ring which conserves density in expectation and thus achieves proportionate agreement. However, none of these algorithms converges to the correct consensus with probability 1.

In arbitrary connected networks, the asynchronous graph automaton using 4 states instead of 2 described in Section I-A solves the binary voting problem in finite time with probability 1 for any vote configuration presenting a strict majority. The idea is that two states indicate that 0 is the majority bit, and two other states indicate that 1 is the majority bit. When running the 4-state automaton, any connected network reaches in finite time a configuration using the two possible states which agree on the initial majority bit. The algorithm uses asynchronicity to ensure that information flows, and adding a bit allows to code the extra global information, necessary to guarantee convergence to the correct bit. Our main challenge is to generalize this simple graph automaton to automata that solve the *multiple* voting problem ($|\mathscr{C}| \geqslant 2$).

The general structure of our approach is the following. First, in Section II, we define pairwise asynchronous graph automata (PAGA). In particular, the 4-state automaton which solves the binary voting problem is a PAGA. In Section IV, we state a number of sufficient conditions for a PAGA to compute the majority color almost surely (a.s.) in finite time in any connected network. The conditions on the updating rules depend only on the number $|\mathscr{C}|$ of possible votes: they are advantageously independent of the network topology. A number of notions are defined to write the conditions on the PAGA and to prove convergence based on these conditions (the proof is in Section V). To illustrate these notions with concrete examples, we describe beforehand, in Section III, the 4-state automaton, which solves the binary voting problem, as well as a simple 15-state automaton, which solves the ternary voting problem. The theory we develop in this paper is able to handle more complicated voting automata, which we describe in Section VI. In particular we build a 100-state automaton, which fulfills all conditions for $|\mathscr{C}|4$, the quaternary voting problem.

II. Automata and the Voting Problem

A. Pairwise Asynchronous Graph Automata (PAGA)

Assume that the agents $\mathcal{V}$ are fixed, but that the edges $\mathcal{E}$ connecting them can be time-varying. We are thus given a sequence of graphs $\{G(t) = (\mathcal{V}, \mathcal{E}(t))\}_{t\geq 0}$. We denote by $\mathcal{G}$ the (finite) set of graphs representing the network over time, and by $\mathcal{D}$ the set of discrete probability distributions.

Definition II.1 [Pairwise Asynchronous Graph Automata (PAGA)]: A pairwise asynchronous graph automaton $\mathcal{A}$ is an ordered quadruple (Q, I, Δ, f), where:

- Q is a finite set of states;
- $I \subseteq Q$ is a set of initial states;
- Δ is a local transition function of the form: $Q \times Q \to Q$;
- f is a function of the form $\mathcal{G} \to \mathcal{D}$. For a given graph $G = (\mathcal{V}, \mathcal{E})$, $f(G)$ is a probability distribution over $\mathcal{E}$.

A PAGA is run asynchronously. At each iteration t, an edge of the current graph $G(t) \in \mathcal{G}$ is drawn at random with the probability distribution $f(G(t))$. The two end agents i and j of the selected edge exchange their states q_i and q_j. Agent i updates its state q_i to $\Delta(q_i, q_j)$, and agent j updates its state q_j to $\Delta(q_j, q_i)$.

There is a classical example of function f to implement a PAGA in a distributed way. Agents are given random independent exponential clocks. When its clock ticks, an agent calls a neighboring agent in the current graph $G(t)$ at random. These two agents form the selected edge. If the neighbor choice is done uniformly then, for an edge $e = (i, j)$, where i has degree d_i and where j has degree d_j, $f(G)(e) = (1/d_i + 1/d_j)/|V|$. Note that f does not rule the construction of the sequence of graphs $\{G(t) = (\mathcal{V}, \mathcal{E}(t))\}_{t\geq 0}$, which is imposed to us, but it defines which edge e is selected in $\mathcal{E}(t)$ for an update at every iteration.

We have not defined final states; a PAGA theoretically never terminates. Therefore, a termination algorithm should be run in parallel. It is well known that an algorithm which detects consensus requires at least $O(\log n)$ bits at each node [17]. If there are x processes of votes, which majorities are to be computed in parallel, then each node needs $O(x \log(|Q|) + \log n)$ memory bits: $\log(|Q|)$ bits per process to code the PAGA states, and $\log n$

bits, which are common to the x processes, to terminate the parallel computations. In that case, the termination algorithm detects consensus, when the x processes have converged to consensus.

B. Voting PAGA

The possible votes are also called colors for simplicity. Let C_{maj} be the set of the initial majority color(s) (there might be a tie between several colors).

Definition II.2 (Voting PAGA for Set $\mathscr{C}$, and Graphs $\mathcal{G}$): Let $\mathscr{C}$ be the set of colors. A voting PAGA is a pairwise asynchronous graph automaton (Q, I, Δ, f) which admits the following properties.

- There is a set $\mathcal{P}$ such that $Q \subseteq \mathcal{P} \times \mathscr{C}$. Any state q can be written as $q = (x, c)$; x is called the coordinate (or point) of q, and c its color. By extension, an agent with state q is also said to have coordinate x and color c.
- For every color $c \in \mathscr{C}$, there is a single state $(o_c, c) \in I$, which is adopted initially by the agents voting for c; $I = \{(o_c, c) : c \in \mathscr{C}\}$.
- When running with this initial state configuration, there is almost surely a finite time T after which all the agents in $\mathcal{G}$ have color in C_{maj}.

According to this definition, a voting PAGA solves the voting problem for colors $\mathscr{C}$ and network $\mathcal{G}$. Indeed, after iteration T, all the agents know the majority vote based on their states. The goal of this paper is to state sufficient conditions on Q, I, Δ, f and $\{G(t)\}_{t\geqslant 0} \in \mathcal{G}^{\mathbb{N}}$ for a PAGA to be a voting PAGA.

In this paper, $\mathcal{P}$ is a set of points in $S_{|\mathscr{C}|}$ that is carefully chosen, because the set of states Q is determined by $\mathcal{P}$. Let $A \in \mathcal{P}$, and x be the coordinates of A, then we use state notations (A, c) and (x, c) interchangeably.

Definition II.3 (Valid States With Respect to $\mathcal{P}$): Let $\mathcal{P}$ be a set of points in $S_{|\mathscr{C}|}$. For any point $A \in S_{|\mathscr{C}|}$, $d_{min}(A) = \min_{\gamma\in\mathscr{C}} d(A, O_\gamma)$, where d is the Euclidean distance. A valid state with respect to $\mathcal{P}$ is defined as follows:

$$(A, c) \text{ is a valid state } \iff \begin{cases} A \in \mathcal{P}, \\ d(A, O_c) = d_{min}(A). \end{cases}$$

In our work, the state set Q is taken as the set of valid states with respect to a set $\mathcal{P}$: $Q = \{(A, c) : A \in \mathcal{P} \text{ and } d(A, O_c) = d_{min}(A)\}$. By extension, we say that a point $A \in S_{|\mathscr{C}|}$ has color c if $d(A, O_c) = d_{min}(A)$. Note that point $A \in S_{|\mathscr{C}|}$ may have several colors, whereas a state $(x, c) \in Q$ has a unique color c.

In the next section, we give two simple examples of PAGA so that the theory developed in Sections IV and V can already be supported by concrete examples.

III. Simple Examples of Voting PAGA

The two following PAGA are voting PAGA if the graphs $\{G(t)\}_{t\geqslant 0}$ are independent and identically distributed (i.i.d.) in $\mathcal{G}$, and if $\widetilde{G} = (\mathcal{V}, \bigcup_{\mathcal{G}} \mathcal{E})$ is a connected graph, i.e., graphs in $\mathcal{G}$ are *jointly connected*.

1) The 4-State Binary Voter: We recall here the 4-state automaton described in Section I-A and already derived in [1], in order to get used to the geometrical framework in a simple example. In the binary voter problem, $\mathscr{C} = \{1, 2\}$ and $S_{|\mathscr{C}|} = S_2$ is a line segment O_1O_2. The barycenter G of the segment is thus its center. To construct Q, we first define $\mathcal{P} = \{O_1, G, O_2\}$. Then, according to Definition II.3,

$$Q = \{(O_1, 1), (G, 1), (G, 2), (O_2, 2)\}.$$

$I = \{(O_1, 1), (O_2, 2)\}$. We choose for example the classical function f described in Section II. For every pair of states (q_1, q_2), we define $\Delta(q_1, q_2)$ as follows:

q_2 \ q_1	$(O_1,1)$	$(G,1)$	$(G,2)$	$(O_2,2)$
$(O_1,1)$	$(O_1,1)$	$(G,1)$	$(G,1)$	$(G,2)$
$(G,1)$	$(O_1,1)$	$(G,1)$	$(G,2)$	$(O_2,2)$
$(G,2)$	$(O_1,1)$	$(G,1)$	$(G,2)$	$(O_2,2)$
$(O_2,2)$	$(G,1)$	$(G,2)$	$(G,2)$	$(O_2,2)$

(1)

As updates are pairwise asynchronous, we recommend to read the table by pairs: $\Delta(q_1, q_2)$ and $\Delta(q_2, q_1)$, and to check that, as in pairwise gossip, the center of mass is conserved by the automaton updates. The reader can also check that this automaton is the same as the one described in Fig. 1.

2) The 15 State Ternary Voter: In the ternary voter problem, $\mathscr{C} = \{1, 2, 3\}$ and $S_{|\mathscr{C}|} = S_3$ is an equilateral triangle $O_1O_2O_3$. The barycenter G of the simplex is located at the intersection of the medians $O_1C_{2,3}$, $O_2C_{1,3}$, and $O_3C_{1,2}$, where $C_{a,b}$ is the middle of O_aO_b. For i =1,2,3, let M_i be the middle of GO_i. To construct Q, we first define $\mathcal{P} = \{O_1, O_2, O_3, C_{1,2}, C_{1,3}, C_{2,3}, M_1, M_2, M_3, G\}$. Then, according to Definition II.3, $Q = \{(O_1, 1), (O_2, 2), (O_3, 3), (C_{1,2}, 1), (C_{1,2}, 2), (C_{1,3}, 1), (C_{1,3}, 3), (C_{2,3}, 2), (C_{2,3}, 3), (M_1, 1), (M_2, 2), (M_3, 3), (G, 1), (G, 2), (G, 3)\}$. To simplify notations, the states are numbered from 1 to 15 as in Fig. 3(b). The set of initial states is $I = \{(O_1, 1), (O_2, 2), (O_3, 3)\}$. We choose the same function f as in previous automaton. For every pair of states (q_1, q_2), we define $\Delta(q_1, q_2)$ as in Table I. Note that 15 states can be coded on 4 bits.

To get familiar with the automaton, we suggest that the reader checks some transitions (together with their symmetric transitions), and locates them on Fig. 3(b): $\Delta(1, 6), \Delta(1, 5), \Delta(1, 7), \Delta(1, 9), \ldots,$. The ternary automaton can be downloaded at [18].

IV. Sufficient Conditions to Construct Voting PAGA

Our goal is to build a theoretical framework in which we can show that all the automata we formulate in this paper do converge and solve the voting problem in finite time with probability 1. In this section, we fix a set of colors $\mathscr{C}$, and we state sufficient conditions on Q, I, Δ, f and $\{G(t)\}_{t\geqslant 0}$ for a PAGA to be a voting PAGA on $\mathscr{C}$. We design the conditions such that they are fulfilled by all the automata we present in this paper, so that we can conclude that our automata are indeed voting PAGA.

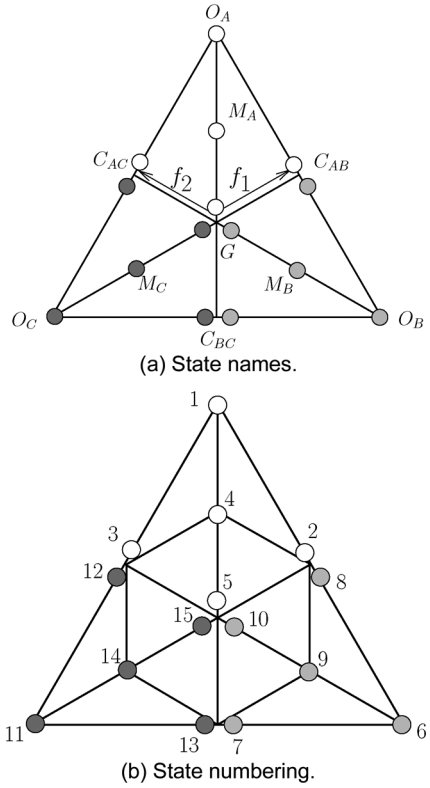


Fig. 3. Ternary voting problem. (a) The 15 states of the automaton described in Section III-2 and (b) their numbering and their associated tiling.

TABLE I
TERNARY AUTOMATON

q_1 \ q_2	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	2	3	4	4	8	5	2	2	4	12	3	5	3	4
2	1	2	4	4	5	6	10	8	9	10	14	4	10	15	10
3	1	5	3	4	5	9	14	5	10	15	11	12	14	14	15
4	1	2	3	4	5	9	10	2	10	5	14	3	15	12	5
5	4	2	3	4	5	9	7	8	9	10	14	12	13	14	15
6	2	8	10	8	9	6	7	8	9	9	13	10	7	7	9
7	4	9	15	5	10	6	7	9	9	10	11	15	13	14	15
8	1	2	4	4	5	6	10	8	9	10	14	4	10	15	5
9	4	8	5	2	10	6	7	8	9	10	14	15	7	15	10
10	4	2	3	4	5	9	7	8	9	10	14	12	13	14	15
11	3	15	12	12	14	7	13	15	13	14	11	12	13	14	14
12	1	5	3	4	5	9	14	5	10	5	11	12	14	14	15
13	4	9	15	5	10	6	7	9	9	10	11	15	13	14	15
14	4	5	12	5	15	9	13	10	7	15	11	12	13	14	15
15	4	2	3	4	5	9	7	8	9	10	14	12	13	14	15

Before we state the conditions, we first need to set up a few definitions. $S_{|\mathscr{C}|}$ is a regular simplex with $|\mathscr{C}|$ vertices embedded in the Euclidean space of dimension $|\mathscr{C}|-1$.

Definition IV.1 (Tiling Associated to $\mathcal{P}$): A tiling associated to $\mathcal{P} \subset S_{|\mathscr{C}|}$ is a set of tiles, which are convex polytopes with positive volume, such that 1) the tile interiors are disjoint, 2) the union of all tiles is equal to $S_{|\mathscr{C}|}$, and 3) any tile vertex is in $\mathcal{P}$.

Example IV.1: $\{O_1G, GO_2\}$ is a tiling associated to $\mathcal{P} = \{O_1, G, O_2\}$ from Section III-1. Fig. 4 shows two tilings associated with points $\mathcal{P}$ of Section III-2. □

We design voting automata with a finite number of states only. Therefore, the set of state coordinates $\mathcal{P}$ is finite, and if the barycenter B of votes does not coincide with a point of $\mathcal{P}$, the automaton cannot reach a consensus on the coordinates of B. Instead, the idea is to have the agents coordinates converge to several points in $\mathcal{P} \bigcap \mathcal{T}$, where $\mathcal{T}$ is the tile which contains barycenter B.

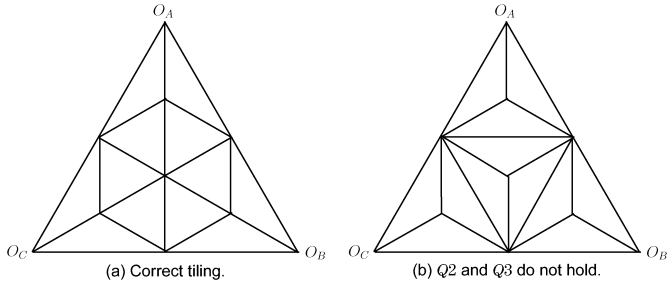


Fig. 4. Tilings for the triangle. (a) The tiling of the ternary automaton of Section III-1. (b) Using the same points $\mathcal{P}$ as in (a), a tiling which does not follow conditions $Q2$ and $Q3$.

Definition IV.2 (Tile-Neighbor): Two points in $\mathcal{P}$ are tile-neighbors if and only if they belong to the same tile.

Definition IV.3 (i-Face, Colors of an i-Face.): For any $i = 0, 1, \ldots, |\mathscr{C}|-1$, the i-faces of a tile $\mathcal{T}$ are recursively defined as follows.

- If $i = |\mathscr{C}|-1$, then the only $(|\mathscr{C}|-1)$-face is $\mathcal{T}$ itself.
- If $i < |\mathscr{C}|-1$, then the i-faces are the intersections of the tangent hyperplanes of dimension i with the $(i+1)$-face(s) of $\mathcal{T}$.

In addition, the **colors of an i-face** $\mathcal{F}$ are the colors of its barycenter $G_{\mathcal{F}}$ (center of mass of its vertices); their set is $\{c : d(G_F, O_c) = d_{min}(G_F)\}$.

The vertices of an i-face of a tile $\mathcal{T}$ are also vertices of $\mathcal{T}$. Therefore, they belong to $\mathcal{P}$.

The 0-faces are the vertices of the tiles, the 1-faces are their edges, etc. In 3-D for example, a tetrahedral tile has four 0-faces, six 1-faces, four 2-faces, and one 3-face (the whole tetrahedron itself).

The study of i-faces is important because it is possible that the barycenter B of votes lies on a face and not in the interior of a tile. In this case, B belongs to more than one tile. To avoid any ambiguity, we will always consider the i-face it belongs to, and not only the tile(s) which contain(s) B.

We are now ready to state the conditions on Q.

Condition IV.1 (on Q): Q is constructed as in Definition II.3, with a set of points $\mathcal{P}$, which admits an associated tiling such that:

- **Q1**: No tile interior contains a point with several colors: for any tile $\mathcal{T}$, for any A in the interior $\mathcal{T}^\circ$ of tile $\mathcal{T}$, $|\{c : d(A, O_c) = d_{min}(A)\}| = 1$.
- **Q2**: Any i-face $\mathcal{F}_i$ of any tile has at most one $(i-1)$-face $\mathcal{F}_{i-1}$ that does not have exactly the same colors as $\mathcal{F}_i$. If there is such an $(i-1)$-face $\mathcal{F}_{i-1}$, then, for any point $A \in \mathcal{F}_i \setminus \mathcal{F}_{i-1}$, A has exactly the same colors as $\mathcal{F}_i$.
- **Q3**: The tiling is consistent: if a set of points are pairwise tile-neighbors, then they all belong to the same tile.

Remark IV.1: $Q1$ implies that all the interior points of a tile share the same unique color. Indeed, if there were two points of different colors, then, by continuity of the distance function, there would be a two-color point somewhere between them; this

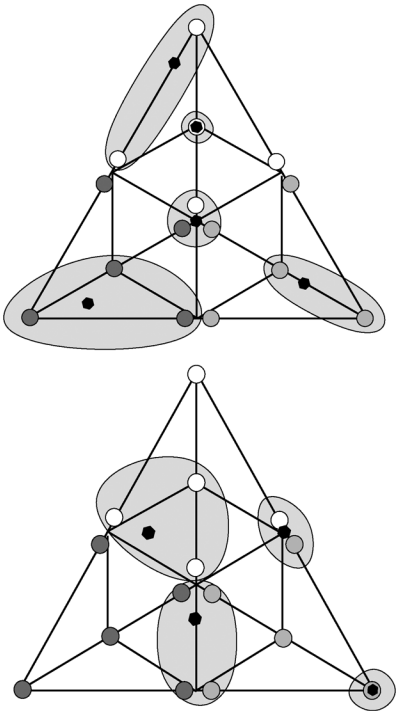


Fig. 5. Example of vote barycenters (black hexagonal dots) with their corresponding convergence cells (gray circular shapes) in the ternary voting PAGA of Section III-1.

is forbidden by $Q1$ since the tiles are convex. The set of points in $S_{|\mathscr{C}|}$ that have more than one color form *boundaries* between the strict majority zones, and they lie on tile faces of dimension smaller than $|\mathscr{C}|-1$.

Fig. 4 shows two tilings associated with the set $\mathcal{P}$ of the automaton of Section III-2, one that follows Condition IV.1 [Fig. 4(a)] and one that does not [Fig. 4(b)].

Condition IV.2 (on I): The simplex vertices $\{O_c\}_{c\in\mathscr{C}} \subset \mathcal{P}$ and I is the set $\{(O_c, c)\}_{c\in\mathscr{C}}$, where the initial state of an agent with vote c is (O_c, c).

When running a PAGA, each agent should eventually hold a state with the majority color (or a majority color in case of a tie). In our setting, at any time, the agents hold states whose coordinates have barycenter B (by definition, B is the barycenter of the initial votes). These coordinates converge to some tile points $\mathcal{T}\cap\mathcal{P}$, whereas the states (coordinates + colors) converge to a set of states, called a *cell*, which depends on B. Then, if the PAGA is correct, the states in this cell should admit the color(s) of B, and only the color(s) of B. In Fig. 5, we show examples of cells on the ternary automaton of Section III-1. The black dots represent some locations of B, and their corresponding convergence cells are circled out. Note that a cell is defined relatively to a face. It excludes the states of the vertices of this face, whose colors are different from the colors of the face interior. Before we can state conditions on Δ, the notion of tiling needs therefore to be refined. We will now properly define the notion of cell.

Definition IV.4 (Cell $\mathcal{C}$ of a Face $\mathcal{F}$): For any $i = 0, 1, \ldots, |\mathscr{C}|-1$, and for any i-face $\mathcal{F}$ of any tile, a cell $\mathcal{C}$ is constructed. Let $c_1, \ldots, c_\ell$ be the ℓ colors of $\mathcal{F}$ and $(x_1, \ldots, x_{n_\mathcal{F}})$ the coordinates of the $n_\mathcal{F}$ points of $\mathcal{F}\cap\mathcal{P}$. Then $\mathcal{C} = \{(x_1, c_1), \ldots, (x_1, c_\ell), \ldots\ldots, (x_{n_\mathcal{F}}, c_1), \ldots, (x_{n_\mathcal{F}}, c_\ell)\}$ is the subset of $\mathcal{P}\times\mathscr{C}$ with coordinates $x_1, \ldots, x_{n_\mathcal{F}}$ and colors $c_1, \ldots, c_\ell$. $\mathcal{C}$ is said to have dimension i.

Example IV.2: The binary voting PAGA (Section III-1) with tiling $\{O_1G, GO_2\}$ has cells $\{(O_1,1),(G,1)\}, \{(O_2,2),(G,2)\}, \{(O_1,1)\}, \{(O_2,2)\}$ and $\{(G,1),(G,2)\}$. See Fig. 5 for some examples of cells of the ternary voting PAGA (Section III-1)

Example IV.3: For any tile $\mathcal{T}$, we construct a cell of dimension $|\mathscr{C}|-1$ with points $\mathcal{T}\bigcap\mathcal{P}$, and *one* color: the color of the interior of $\mathcal{T}$ (Remark IV.1). □

Example IV.4: Let G be the barycenter of $S_{|\mathscr{C}|}$. It is easy to show that $G\in\mathcal{P}$. The cell of dimension 0 constructed with G is $\{(G,1), \ldots, (G, |\mathscr{C}|)\}$. □

Lemma IV.1: Condition $Q1$ implies that all the elements of a cell are valid states with respect to $\mathcal{P}$: $\mathcal{C}\subset Q$.

Proof: Let $\mathcal{T}$ be a tile from which an i-face $\mathcal{F}$ with ℓ colors was constructed and consider its corresponding cell $\mathcal{C}$. From Remark IV.1, the interior $\mathcal{T}^\circ$ of tile $\mathcal{T}$ has a unique color; denote this color by c_1. By continuity of the distance function, any point in $\mathcal{F}\subseteq\mathcal{T}$ has color c_1. In particular points in $\mathcal{F}\cap\mathcal{P}$ have color c_1. Therefore, if $\ell=1$, by definition of the state space Q ($(x,c)\in Q$ if $x\in\mathcal{P}$ and if x has color c), $\mathcal{C}\subset Q$. Suppose now that $\ell>1$ (which excludes $i=|\mathscr{C}|$), then by Definition IV.3, the barycenter $G_\mathcal{F}$ of $\mathcal{F}$ has colors $c_1, \ldots, c_\ell$. In particular $G_\mathcal{F}$ lies on the perpendicular bisector of segment $[O_{c_1}O_{c_2}]$, which is a hyperplane of dimension $|\mathscr{C}|-2$. If $\mathcal{F}$ is not entirely included in the bisector, then the bisector crosses $\mathcal{T}^\circ$, which is excluded by $Q1$. Indeed, if this was the case, there would be points in $\mathcal{T}^\circ$ that are equidistant from O_{c_1} and O_{c_2}: c_2 would also be a color of $\mathcal{T}^\circ$. Therefore $\mathcal{F}$ is included in the perpendicular bisectors of all segments $[O_{c_1}O_{c_j}]$ for $2\leqslant j\leqslant\ell$, and any point of $\mathcal{F}$ is equidistant from $O_{c_1}, \ldots, O_{c_\ell}$. The points in $\mathcal{F}$ already have color c_1, therefore they also have colors $c_2, \ldots, c_\ell$. In particular, points in $\mathcal{F}\cap\mathcal{P}$ have colors $c_1, c_2, \ldots, c_\ell$, which proves that all states in $\mathcal{C}$ are valid states (see Definition II.3) ■

Remark IV.2: The previous proof also shows that all the points in the i-interior of an i-face have exactly the same colors, which are the colors of the i-face. Indeed, we can replace $G_\mathcal{F}$ by any other interior point in the proof. Therefore, the second part of Condition Q2 is a little redundant with Condition Q1 and could be restricted to "If there is such an $(i-1)$-face $\mathcal{F}_{i-1}$, then, for any point $A\in\mathcal{F}_i\setminus\{\mathcal{F}_{i-1}\bigcup\mathcal{F}_i^\circ\}$, A has exactly the same colors as $\mathcal{F}_i$," where $\mathcal{F}_i^\circ$ refers to the i-interior of $\mathcal{F}_i$.

Definition IV.5 (Adjacent States): Two states are adjacent if they belong to the same cell.

A state is adjacent with itself. The points of two adjacent states are tile-neighbors, but the reverse statement is not true. On the other hand, the following holds.

Lemma IV.2: Two states (x_1,c_1) and (x_2,c_2) are adjacent if and only if points of coordinates x_1 and x_2 are tile-neighbors and if $(x_1,c_2)\in Q$ and $(x_2,c_1)\in Q$.

Proof: ($\Rightarrow$) obvious. ($\Leftarrow$) Suppose that x_1 and x_2 are tile-neighbors and that they both have colors c_1 and c_2. Let $\mathcal{F}$ be the face of smallest dimension containing x_1 and x_2 and let i be its dimension. Consider the Voronoi cells of points $O_{c_1}, O_{c_2}, \ldots, O_{|\mathscr{C}|}$. Then x_1 and x_2 belong to the Voronoi cells of both O_{c_1} and O_{c_2}. Voronoi cells being convex, any points of the segment $[x_1,x_2]$ also belong to these two Voronoi cells. In

other words any points of the segment $[x_1, x_2]$ have colors c_1 and c_2. Face $\mathcal{F}$ being the face of smallest dimension, then such points are in the i-interior of $\mathcal{F}$. By Remark IV.2, $\mathcal{F}$ has colors c_1 and c_2, and the cell constructed with $\mathcal{F}$ has these colors as well. ∎

Definition IV.6 (Smallest Cell): Let $\mathcal{Z}$ be a subset of a tile. The smallest cell of $\mathcal{Z}$ is the cell of smallest dimension whose points' convex hull contains $\mathcal{Z}$.

In the proofs of Theorems V.1 and V.2, we will show that the states of the agents eventually belong to the smallest cell of the barycenter B of the votes. This will guarantee that the agents eventually adopt the colors of B, which are the majority colors.

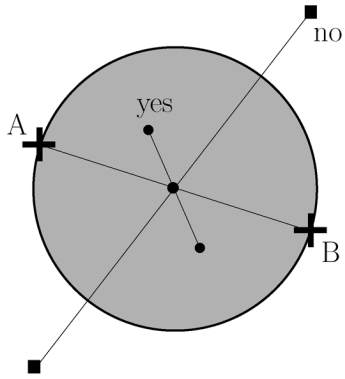
Condition IV.3 (on Δ): Condition $Q1$ holds so that Q is organized in cells as in Definition IV.4. For any states $q_1 = (x_1, c_1)$ and $q_2 = (x_2, c_2)$, let $q_1' = (x_1', c_1') = \Delta(q_1, q_2)$ and $q_2' = (x_2', c_2') = \Delta(q_2, q_1)$. Δ enjoys the following properties.

- $\mathbf{\Delta 1}$: Conservation of the center of mass: $x_1' + x_2' = x_1 + x_2$.
- $\mathbf{\Delta 2}$: Cell contraction: q_1' and q_2' are adjacent.
- $\mathbf{\Delta 3}$: Mixing: if x_1 and x_2 are tile-neighbors, then $x_1' = x_2$ and $x_2' = x_1$.
- $\mathbf{\Delta 4}$: Color swap: Let $\mathcal{C}$ be the smallest cell of $\{x_1', x_2'\}$. If c_2 is a color of $\mathcal{C}$, then $c_1' = c_2$. Similarly, if c_1 is a color of $\mathcal{C}$, then $c_2' = c_1$.
- $\mathbf{\Delta 5}$: Spatial contraction: There is a potential function $V : \mathcal{P} \to \mathbb{R}^+$ such that $V(x_1') + V(x_2') \leqslant V(x_1) + V(x_2)$, where the inequality is an equality if and only if $x_1' = x_2$ and $x_2' = x_1$.

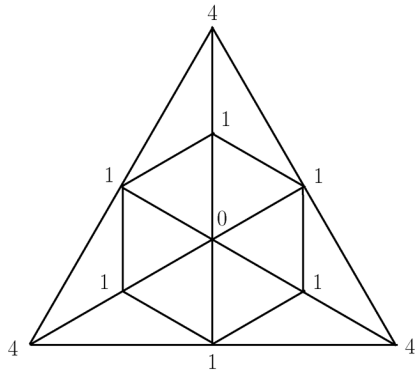
Remark IV.3: In practice, it is worth refining $\Delta 3$ to "Δ swaps tile-neighbor coordinates unless it is possible to further contract them within their tile $\mathcal{T}$." If there are coordinates y and z in $\mathcal{T} \cap \mathcal{P}$ such that $x_1 + x_2 = y + z$ and such that $d(y, z) < d(x_1, x_2)$, then it is interesting to set x_1' and x_2' equal to y and z in either order.

Remark IV.4: Let G (barycenter of $S_{|\mathcal{C}|}$) be the origin of the Euclidean space where $S_{|\mathcal{C}|}$ lies. Condition $\Delta 5$ is called *spatial contraction* condition because $V(x) = x^T x$, which is a valid potential function for the automata of Section III, has a simple geometrical contraction interpretation. For any point of coordinates x, $x^T x$ is the square distance to the origin G of the Euclidian space. Assume that $\Delta 1$ holds and let $\epsilon = x_1' - (x_1 + x_2)/2 = (x_1 + x_2)/2 - x_2'$. Then V satisfies $\Delta 5$ if and only if $\|\epsilon\|_2 < \|(x_1 - x_2)/2\|_2$ (except when $x_1' = x_2$ and $x_2' = x_1$): x_1' and x_2' are inside the circle centered at $(x_1 + x_2)/2$ which goes through x_1 and x_2. Indeed,

$$
\begin{aligned}
&V(x_1) + V(x_2) - (V(x_1') + V(x_2')) \\
&\quad = x_1^T x_1 + x_2^T x_2 - \left(\frac{x_1^T + x_2^T}{2} + \epsilon^T\right)\left(\frac{x_1 + x_2}{2} + \epsilon\right) \\
&\qquad - \left(\frac{x_1^T + x_2^T}{2} - \epsilon^T\right)\left(\frac{x_1 + x_2}{2} - \epsilon\right) \\
&\quad = 2\left[\left(\frac{x_1 - x_2}{2}\right)^T \left(\frac{x_1 - x_2}{2}\right) - \epsilon^T \epsilon\right] \\
&\quad = 2\left[\left\|\frac{x_1 - x_2}{2}\right\|_2^2 - \|\epsilon\|_2^2\right].
\end{aligned}
$$



(a) geometrical interpretation



(b) V for ternary automaton

Fig. 6. Spatial contraction property (a) The pair of state updates for states in A and B should lie in the gray circle for the function $V(x) = x^T x$ to be a spatially contracting potential function. (b) $V(x) = x^T x$ is a valid potential function of the ternary automaton of Section III-2. (a) Geometrical interpretation. (b) V for ternary automaton.

The potential function V satisfies the spatial contraction property if and only if

$$\|\epsilon\|_2 \leq \left\|\frac{x_1 - x_2}{2}\right\|_2.$$

The inequality should be strict as soon as states do not swap coordinates. For the ternary automaton of Section III, V can be visualized in Fig. 6(b).

Condition IV.4 (on f and $\{G(t)\}_{t\geqslant 0}$): $\{G(t)\}_{t\geqslant 0}$ is i.i.d. Let $\mathcal{G}$ be the finite set defined by $\mathcal{G} = \{G : \mathbb{P}[G(t) = G] > 0\}$. Let $\mathcal{E}' \subseteq \bigcup_{\mathcal{G}} \mathcal{E}$ be the set of edges e that are chosen with positive probability: $\mathcal{E}' = \{e \in \bigcup_{\mathcal{G}} \mathcal{E} : \exists G \in \mathcal{G}, f(G)(e) > 0\}$. Graph $G' = (\mathcal{V}, \mathcal{E}')$ is connected, i.e., graphs $\{G(t)\}_{t\geqslant 0}$ are jointly connected.

V. Almost Sure Convergence in Finite Time

We prove in this section that our automata solve the multiple voting problem.

Theorem V.1: Assume that Conditions 1, 2, and 3 hold for a given PAGA. If there is a finite time T after which all agents states belong to the same cell, then, after T, every agent color is in C_{maj}.

Proof: Let $\mathcal{C}$ be the cell of smallest dimension the agents states belong to after time T, let $\mathcal{F}$ be the face from which cell $\mathcal{C}$ is built, and let R be the set of agents coordinates. Let B be the barycenter of the initial votes; B has color(s) C_{maj}. In this proof, we show that $\mathcal{C}$ is the smallest cell of B, that it implies that $\mathcal{C}$ has colors C_{maj}, and thus that agents have colors C_{maj}. By Conditions IV.2 and $\Delta 1$ (conservation of center of mass), B lies in the convex hull of the points of $\mathcal{C}$, i.e., $B \in \mathcal{F}$. Let $\mathcal{C}_s$ be the smallest cell of B, and $\mathcal{F}_s$ be the face from which it is built. We need to show that $\mathcal{F} = \mathcal{F}_s$. First $\mathcal{F}_s \subseteq \mathcal{F}$ ($\mathcal{F}_s$ is a face of $\mathcal{F}$) because B belongs to both faces, and $\mathcal{F}_s$ is the one of smallest dimension. Suppose that $\mathcal{F} \neq \mathcal{F}_s$, then there is a point of R in $\mathcal{F} \setminus \mathcal{F}_s$. Tiles are convex, thus their faces are convex too. Hence, any point in $\mathcal{F} \setminus \mathcal{F}_s$ is not in the tangent hyperplane $\mathcal{H}$ to $\mathcal{F}$ containing $\mathcal{F}_s$, and $\mathcal{F} \setminus \mathcal{F}_s$ actually lies on one side only of $\mathcal{H}$, but B is a weighted barycenter of the points in R, thus B cannot be in $\mathcal{H}$; hence, $B \notin \mathcal{F}_s$. This contradicts $B \in \mathcal{F}_s$. Therefore, $\mathcal{F} = \mathcal{F}_s$, and $\mathcal{C}$ is the smallest cell of B. Two cases appear: 1) either B constitutes a 0-face and then $\mathcal{C}$ is the cell built from the 0-face $\{B\}$ and color(s) C_{maj}; 2) or B is not a 0-face. Then $B \in \mathcal{F}^{\circ}$, and $\mathcal{C}$ has exactly the colors C_{maj} of B (Remark IV.2). The agents states are in $\mathcal{C}$; thus, they have colors in C_{maj}. □ ■

Theorem V.2 (Main Theorem): If $\{G(t)\}_{t\geqslant 0}$ and $\mathcal{A} = (Q, I, \Delta, f)$ satisfy Conditions 1, 2, 3, and 4, then $\mathcal{A}$ is a **voting PAGA**, i.e., there is a.s. a finite time T after which all the agents colors are in C_{maj}.

Let T be the first time when all agents states are in the same cell. By $\Delta 3$ and $\Delta 4$, if q_1 and q_2 are adjacent states, then Δ swaps states: $\Delta(q_1, q_2) = q_2$ and $\Delta(q_2, q_1) = q_1$. Therefore, states remain in the same cell forever after T. Note that states remain in the same cell even when Remark IV.3 is implemented. Thus, according to Theorem V.1, it is sufficient to show that there is a time $T < \infty$ when all agents are a.s. in the same cell to prove Theorem V.2.

Proof: This proof has two parts: "Coordinates" and "Colors." In the "coordinates" part, we use the spatial contraction property to set up a time T_1 after which states coordinates are systematically swapped whenever two agents communicate. We show that any pair of coordinates is eventually swapped with probability 1 because the network is jointly connected. By $\Delta 2$, updated states are adjacent, hence swapped coordinates are tile-neighbors. As a consequence, at time T_1, every pair of agents a.s. has tile-neighbor coordinates. The tiling being consistent, all the coordinates are a.s. in the same tile. Then we need to show that states eventually adopt the "right colors," i.e., the colors of a cell containing the coordinates of the agents. This is proven in the "color part" of the proof. Condition $Q2$ guarantees that there is an agent whose coordinates x_a admit only right colors. Each time an agent with coordinates x_a communicates with a neighbor, it transforms its neighbor's color into a right color. Since this coordinate x_a travels in the network thanks to $\Delta 3$, it eventually attracts all the colors into C_{maj}. This is how the algorithm proceeds, and the proof translates this phenomenon in the following way. Formally, we show that there is a time T_2 after which the number of right colors is maximum. Then we show that with probability 1, coordinate x_a eventually meets all the colors. Therefore, with probability 1, at time T_2, all the colors of the agents were already right colors.

Coordinates: For $i = 1, \ldots, n$, let $q_i(t)$ be the state of agent i at time t. The sum of potential functions $\{E(t)\}_{t\geqslant 0} = \{\sum_{i=1}^{n} V(q_i(t))\}_{t\geqslant 0}$ is a non-increasing sequence ($\Delta 5$), i.e., it is a Lyapunov function. Since there is a finite number of states, the Lyapunov function can take only a finite number of values. Thus, it reaches a minimum in finite time T_1. We first show that, after T_1, all agents states are a.s. pairwise tile-neighbors, i.e., they all belong to a tile $\mathcal{T}$ ($Q3$). Take a pair of agents a and b (with coordinates x_a and x_b at time T_1) and build the following sequence: $a(T_1) = a$, $b(T_1) = b$ and for any $t \geqslant T_1$,

$$a(t+1) = \begin{cases} i(t), & \text{if } a(t) \text{ and } i(t) \\ & \text{communicate at time } t. \\ a(t), & \text{if } a(t) \text{ does not} \\ & \text{communicate at time } t. \end{cases} \tag{2}$$

$$b(t+1) = \begin{cases} i(t), & \text{if } b(t) \text{ and } i(t) \\ & \text{communicate at time } t. \\ b(t), & \text{if } b(t) \text{ does not} \\ & \text{communicate at time } t. \end{cases} \tag{3}$$

After T_1, by $\Delta 5$, pairs of communicating agents swap coordinates (otherwise E decreases); therefore, for $t \geqslant T_1$, agent $a(t)$ (respectively, $b(t)$) has coordinates x_a (resp., x_b). We consider the joint process $\{a(t), b(t)\}_{t\geq T_1}$, which is a *homogeneous* Markov chain over the state space $H = \{1, 2 \ldots, n\}^2 \setminus \{(1,1), (2,2), \ldots, (n,n)\}$ (for any t, $a(t) \neq b(t)$). Indeed, at any time t, $(a(t+1), b(t+1))$ depends on $(a(t), b(t))$ and on the edge $e(t)$ that is selected at random in an i.i.d. fashion at time t: $\mathbb{P}[e(t) = e] = \sum_{G\in\mathcal{G}} f(G)(e)\mathbb{P}[G(t) = G]$ is independent of t ($\{G(t)\}_{t\geqslant 0}$ is i.i.d.). By Condition 4, the set $\mathcal{E}'$ of edges e such that $\mathbb{P}[e(t) = e] > 0$ forms a connected graph $G' = (\mathcal{V}, \mathcal{E}')$. Therefore $\{a(t), b(t)\}_{t\geq T_1}$ is irreducible. Moreover, for any edge $e = (i, j) \in \mathcal{E}'$, the Markov chain $\{a(t), b(t)\}_{t\geq T_1}$ admits a positive transition probability $p = \mathbb{P}[e(t) = e]$ from state (i, j) to state (j, i). It was shown in [1] that any transition with positive probability of an irreducible Markov chain over a finite state set is used in finite time with probability 1. Therefore, with probability 1, there is a finite time T_{ab} such that agents $a(T_{ab})$ and $b(T_{ab})$ communicate and swap their coordinates x_a and x_b at time T_{ab}. Condition $\Delta 2$ imposes that updated states are adjacent. Thus, x_a and x_b are tile-neighbors with probability 1. Therefore, at time T_1, states coordinates are a.s. pairwise tile-neighbors.

Colors: Let $\mathcal{C}$ be the smallest cell that the agents coordinates occupy at time T_1, and we claim that all the agents states end up in $\mathcal{C}$. If $\mathcal{C}$ has dimension 0, then the agents states are necessarily in $\mathcal{C}$ at time T_1 (such a cell gathers all the colors of a given tile vertex). Suppose now that $\mathcal{C}$ has at least two states of different coordinates. Let $i > 0$ be its dimension and let $\mathcal{N}(t)$ be the set of agents whose states are in $\mathcal{C}$, i.e., whose colors are colors of $\mathcal{C}$. The sequence $|\mathcal{N}(t)|$ is a non-decreasing integer sequence; thus, it reaches a maximum in finite time $T_2 \geqslant T_1$. Indeed, any cell of smaller dimension than $\mathcal{C}$, and whose points are in $\mathcal{C}$,

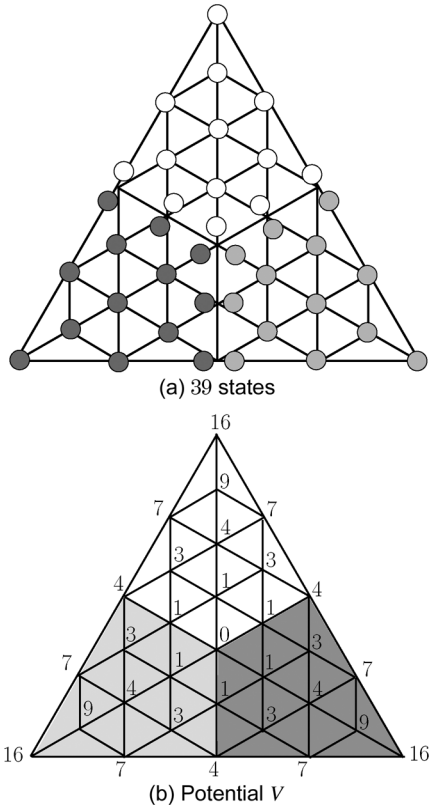


Fig. 7. Ternary voter automaton with 39 states. The potential function in (b) is $V(x) = xx^T$. (a) 39 states. (b) Potential V.

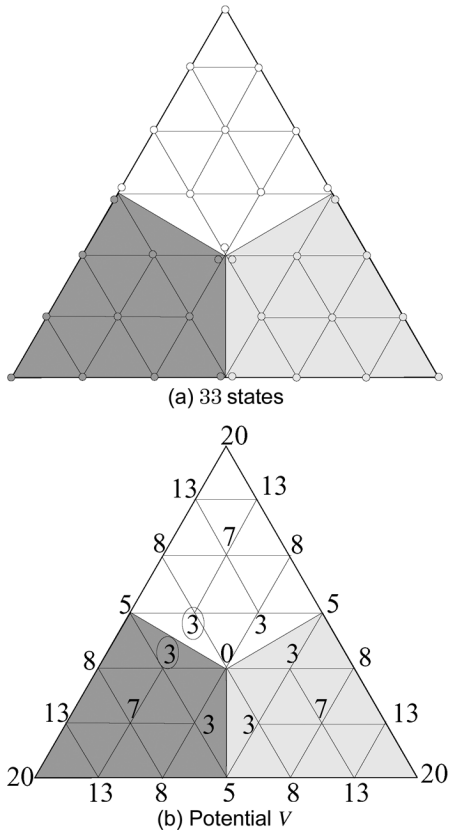


Fig. 8. Ternary voter automaton with 33 states. The potential function in (b) is *not* $V(x) = xx^T$. When the two circled states with $V = 3$ situated on both sides of GC_{12} communicate, they update to G $(V = 0)$ and C_{12} $(V = 5)$, one with color 1 (white), the other with color 2 (gray). $\Delta 5$ holds, but this updating is not spatially contracting in the Euclidian sense. (a) 33 states. (b) Potential V.

has the colors of $\mathcal{C}$ and maybe some more; thus, $\Delta 4$ implies that any time an agent with a state in $\mathcal{N}(t)$ is activated, then its activated neighbor in $G(t)$ adopts its color (and thus its state). Cell $\mathcal{C}$ being the smallest cell of the agents coordinates, and considering condition $Q2$, $(*)$ there is at least one agent a in the network whose coordinates x_a at time T_2 admits *only* colors of $\mathcal{C}$. Let c_a be its color. Now, take any other agent b with state (x_b, c_b) at time T_2, and build the same sequences $\{a(t)\}_{t \geqslant T_2}$ and $\{b(t)\}_{t \geqslant T_2}$ as in (2), (3), with $a(T_2) = a$, $b(T_2) = b$. Then, for $t \geqslant T_2$, agent $a(t)$ has state (x_a, c_a). Furthermore, if c_b is not a color of $\mathcal{C}$, then $(**)$ the color of agent $b(t)$ at time t is *not* in $\mathcal{C}$ either (otherwise $|\mathcal{N}(t)|$ would increase). At any time t, the coordinates of $b(t)$ are x_b. From the "coordinates" part of the proof, we know that $a(t)$ and $b(t)$ communicate in finite time with probability 1. Therefore, there is a.s. a finite time T'_{ab} when agents $a(T'_{ab})$ and $b(T'_{ab})$ communicate. Agent $b(T'_{ab})$ updates its state to (x_a, c_a) and agent $a(T'_{ab})$ updates to (x_b, c) for some color c, such that states (x_a, c_a) and (x_b, c) are adjacent $(\Delta 2)$, but x_a admits only colors in $\mathcal{C}$, and Lemma IV.2 shows that the states (x_a, c_a) and (x_b, c) are adjacent only if x_a has color c, i.e., by statement $(*)$, only if c is a color of $\mathcal{C}$. Therefore, with probability 1, c is a color of $\mathcal{C}$, which is possible only if c_b is a color of $\mathcal{C}$ as well (by statement $(**)$). In other words, for any agent b, at time $T_2 < \infty$ (and after time T_2), its state (x_b, c_b) is a.s. in $\mathcal{C}$; at time T_2, agents states a.s. belong to a single cell $\mathcal{C}$.■

VI. Other Examples of Voting PAGA

A. Binary and Ternary Voters

For the binary and the ternary voting problems at least, voting PAGA are not unique. Sets $\mathcal{P}$ from Section III-1 can be made larger by adding for example a point in the middle of each edge of each tile. This leads to automata $\mathcal{A}_{2k}$ with $2k$ states that solve the binary voting problem. For the ternary voting problem, refer to Fig. 7 to see the new set $\mathcal{P}$, as well as its tiling and its associated potential function V.

A large set $\mathcal{P}$ with a radically different tiling can also be designed for the ternary voting problem, as shown on Fig. 8. The resulting automaton is interesting because it is not spatially contracting in the Euclidian sense. However, a potential function $V(x) \neq x^T x$ can be found to show that the automaton does classify majority among three votes. Unfortunately, the automaton has $33 = 2^5 + 1$ states, which still requires 6 bits. Remember that the ternary PAGA of Section III-1 has 15 states, that can be coded on 4 bits.

B. Practical Considerations on Larger Automata

Larger sets $\mathcal{P}$ lead to voting PAGA which tend to reach consensus on C_{maj} faster. Indeed, in situations close to a tie between two colors, the barycenter of votes are close to the frontier between the two color zones. Having more possible states

close to the frontier allows to have more agents in the network holding a state which is strictly inside a color zone. These nodes have the power of attracting the states that are on the frontier to the correct color.

Their transition functions Δ are omitted because they do not fit on a page. However, if a proper tiling and a potential function V are given, the reader can easily build a correct Δ function by following the "instructions" of Condition IV.3. In a few words, for any pair of states $q_1 = (x_1, c_1)$ and $q_2 = (x_2, c_2)$.

1) Compute the center of mass $m = (x_1 + x_2)/2$.
2) If m is inside a tile $\mathcal{T}$,
   - Look for two points y and z in $\mathcal{T}$, that have center of mass m.
   - If there are several such pairs, take the pair that minimizes $V(y) + V(z)$.
   - Assign (y, c_m) and (z, c_m) as updates $q_1' = (x_1', c_1')$ and $q_2' = (x_2', c_2')$. Out of the two possible assignments, it is heuristically interesting (better speed of convergence) to assign them in an order that maximizes

$$d(x_1, x_1') + d(x_2, x_2'). \tag{4}$$

3) If m is on a face $\mathcal{F}$.
   - Look for two points y and z in $\mathcal{F}$, that have center of mass m, that minimize the sum of energies V.
   - Assign the update coordinates in order to maximize the sum of distances as in (4).
   - The update colors should be colors of m. Follow $\Delta 4$ when possible. In the case where $\Delta 4$ does not impose any color, then choose arbitrarily any color c of m. We recommend that if m has several colors, then the two updates should have different colors. Also it is more elegant to respect the symmetry of the triangle, and to impose the same global number of updates of color 1, color 2, and color 3. The goal is not to give the advantage to any color in the automaton in case of a tie between two or three colors. Note that the triangle has a rotational symmetry, which makes perfectly balanced automata possible, whereas the tetrahedron is not rotationally symmetrical. It is thus difficult to construct well-balanced automata on the tetrahedron.

To summarize, the tiling and the potential function V contains all the necessary information to code a correct automaton. From a design point of view, constructing a new voting PAGA boils down to finding a new tiling on which Conditions IV.1 and IV.3 can be applied.

C. Quaternary Voter

We have successfully built a quaternary voting PAGA.

1) Construction of Set $\mathcal{P}$: $\mathcal{S}_4$ is a regular tetrahedron with vertices O_1, O_2, O_3, and O_4 and barycenter G. Denote by C_{ij} the middle of edge O_iO_j and form the orthogonal basis $(G\vec{C}_{12}, G\vec{C}_{13}, G\vec{C}_{14})/3$ with origin at G. Consider the 3-D-lattice $\mathcal{L}$ generated by the $\mathbb{Z}$-span of the basis, and let $\mathcal{P}$ be the intersection of $\mathcal{L}$ with $\mathcal{S}_4$ minus the points situated on the edges

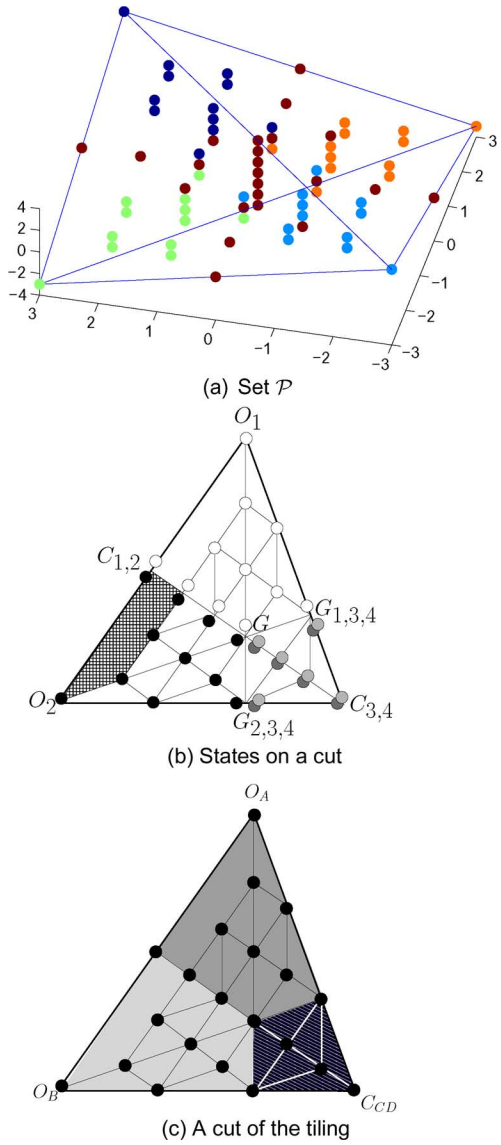


Fig. 9. (a) Set $\mathcal{P}$ in the tetrahedron. (b) States and tiling for the quaternary PAGA on the cut of $\mathcal{S}_4$ along the hyperplane $O_1O_2C_{3,4}$. A state (x, c) is represented by a circle of color c at position x. Color 1 is white, 2: black, 3: light gray, 4: dark gray. $G_{i,j,k}$ is the barycenter of face $O_iO_jO_k$. Note that Remark IV.3 can be applied on the shaded 2-face. (c) The tiling, without the states. (a) Set $\mathcal{P}$. (b) States on a cut. (c) A cut of the tiling.

of $\mathcal{S}_4$ different from $\{O_i\}_{i\in[1,4]}$ and from $\{C_{ij}\}_{i\neq j\in[1,4]^2}$.[3] Set $\mathcal{P}$ generates 100 states, 25 of each color. See Fig. 9(b) and (a) to visualize the 100 states, which are listed in Table II. Note that 100 states can be coded on 7 bits.

2) Tiling: The 3-D-lattice $\mathcal{L}$ cuts the tetrahedron in unit cubes. Each unit cube is divided in six tiles, shaped as tetrahedra. The six tetrahedra in each cube share a common edge, which is one of the four big diagonals.[4] Therefore, for each unit cube, one should choose one big diagonal. There are two case.

[3]This is useful to limit the number of states. Also, note that the quaternary PAGA, restricted to each of the four faces of the tetrahedron should constitute a ternary PAGA, which appears to be impossible if these points are not removed. This simplex face constraint is the main difficulty in the design of voting PAGA for arbitrary $\mathcal{C}$.

[4]In more technical terms, the big diagonal is called the circumradius.

TABLE II
STATE INDEXING FOR THE 4-VOTER AUTOMATON WITH 100 STATES. STATES 1, 26, 51, AND 76 ARE THE INITIAL STATES

color A				color B				color C				color D			
1	3	3	3	26	-3	-3	3	51	-3	3	-3	76	3	-3	-3
2	2	2	2	27	-2	-2	2	52	-2	2	-2	77	2	-2	-2
3	1	2	2	28	-1	-2	2	53	-1	2	-2	78	1	-2	-2
4	2	2	1	29	-2	-2	1	54	-2	2	-1	79	2	-2	-1
5	2	1	2	30	-2	-1	2	55	-2	1	-2	80	2	-1	-2
6	1	1	2	31	-1	-1	2	56	-1	1	-2	81	1	-1	-2
7	1	2	1	32	-1	-2	1	57	-1	2	-1	82	1	-2	-1
8	2	1	1	33	-2	-1	1	58	-2	1	-1	83	2	-1	-1
9	1	1	1	34	-1	-1	1	59	-1	1	-1	84	1	-1	-1
10	1	1	0	35	-1	-1	0	60	-1	1	0	85	1	-1	0
11	1	0	1	36	-1	0	1	61	-1	0	-1	86	1	0	-1
12	0	1	1	37	0	-1	1	62	0	1	-1	87	0	-1	-1
13	0	0	3	38	0	0	3	63	0	0	-3	88	0	0	-3
14	0	3	0	39	0	-3	0	64	0	3	0	89	0	-3	0
15	3	0	0	40	-3	0	0	65	-3	0	0	90	3	0	0
16	0	0	2	41	0	0	2	66	0	0	-2	91	0	0	-2
17	0	2	0	42	0	-2	0	67	0	2	0	92	0	-2	0
18	2	0	0	43	-2	0	0	68	-2	0	0	93	2	0	0
19	0	0	1	44	0	0	1	69	0	0	-1	94	0	0	-1
20	0	1	0	45	0	-1	0	70	0	1	0	95	0	-1	0
21	1	0	0	46	-1	0	0	71	-1	0	0	96	1	0	0
22	-1	1	1	47	1	-1	1	72	1	1	-1	97	-1	-1	-1
23	1	1	-1	48	-1	-1	-1	73	-1	1	1	98	1	-1	1
24	1	-1	1	49	-1	1	1	74	-1	-1	-1	99	1	1	-1
25	0	0	0	50	0	0	0	75	0	0	0	100	0	0	0

1) The cube does not cross a frontier zone. Then, the big diagonal goes through the cube vertex closest to O_c, where is c is the color of the cube.
2) The cube crosses a frontier zone. Then the big diagonal is included in the frontier. Moreover no tetrahedron should cross the frontier zone.

Finally, for cubes that are close to the edges of the tetrahedron, we had to adapt the shape of the tiles to make the automaton work. See Fig. 9(c).

3) Potential Function V: Building the potential function V is slightly more complicated than for the triangle, since the quaternary PAGA is not strictly spatially contracting in the Euclidian sense. Indeed, if the middle m of the coordinates of two states is equal to the center of mass of a cube, then the updated states should be the vertices of the big diagonal of the cube chosen in the tiling construction. Now, if the two states are originally another big diagonal of the same cube, then the updates do not spatially contract, i.e., they lie on the same sphere centered in m. Therefore, $V = x^T x$ is not appropriate. However, $V(x) = x^T x + \sqrt{x^T x}$ works, as it adds a perturbation factor that favors states aligned with G. The resulting potential table is given in Table III.

4) Transition Function Δ: It is important to understand that only the function Δ is needed to implement a voting algorithm. As we have already computed Δ, one just needs to get Δ, to know which are the initial states and what are the states' colors. The transition function Δ is a simple $|Q| \times |Q|$ table that indicates how to update the states for any pair of states. For the tetrahedron, 10 000 entries have thus been computed, using all the possible symmetries of the tetrahedron. The detailed construction is omited here. The automaton can be downloaded at [19].

5) Simulations: Fig. 10 shows a simulation, where the automaton runs on a ring network. The algorithms induced by the automata can be slow and there are heuristics to make them converge faster. For example, if nodes with state coordinate G do not wake up, it is easy to see that the algorithm still converges. The algorithm is especially slow when the votes are close to a tie. In that case, many nodes have a state at G, and it is thus worth making them silent. In a wireless network, by reducing interference, the remaining nodes can decide to wake up more often and attract states of other nodes to their color more efficiently.

TABLE III
POTENTIAL FUNCTION FOR THE 100 STATES FOR THE 4-VOTER AUTOMATON. $V(x) = 0.5(x^T x + \sqrt{x^T x})$

states				potential
1	26	51	76	16.0981
2	27	52	77	7.7321
3	28	53	78	6
4	29	54	79	6
5	30	55	80	6
6	31	56	81	4.2247
7	32	57	82	4.2247
8	33	58	83	4.2247
9	34	59	84	2.3660
10	35	60	85	1.7071
11	36	61	86	1.7071
12	37	62	87	1.7071
13	38	63	88	6
14	39	64	89	6
15	40	65	90	6
16	41	66	91	3
17	42	67	92	3
18	43	68	93	3
19	44	69	94	1
20	45	70	95	1
21	46	71	96	1
22	47	72	97	2.3660
23	48	73	98	2.3660
24	49	74	99	2.3660
25	50	75	100	0

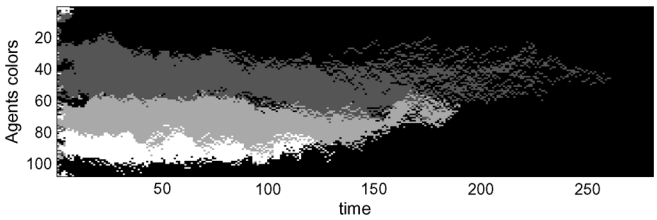


Fig. 10. Quaternary PAGA running on a ring of 108 agents with votes: 33 blacks, 30 dark grays, 25 light grays and 20 whites. The successive state color configurations are shown with time increasing towards the right ($2 * 108 = 216$ iterations per column).

VII. CONCLUSION

A. Summary

In this paper, we focused on the multiple voting problem, where every node votes for one candidate among $|\mathscr{C}| \geqslant 2$ candidates and needs to learn the majority vote of the network. We stated a number of sufficient conditions for an automaton to solve the multiple voting problem. We then designed automata fulfilling these conditions for the cases of $|\mathscr{C}| = 2$, 3, and 4 possible votes. These automata reach correctly consensus on the majority color, but they do not have final states and at convergence nodes keep swapping their coordinates. Simple parallel

procedures need therefore to be implemented to detect that color consensus is reached, and terminate the algorithm.

B. Further Work: More Than Four Votes and Extension to the Ranking Problem

When more than four votes coexist in the network, we can create algorithms based on the binary, ternary or quaternary automata to solve the multiple *voting* problem or even to rank all votes by frequency of apparition (*ranking* or *sorting* problem).

For large $|\mathscr{C}|$, one can *sequentially* use the algorithms designed for small $|\mathscr{C}|$ to perform successive comparisons. For example, ranking $|\mathscr{C}| > 2$ votes can be done with the binary automaton sequentially performing pairwise comparisons. To compare votes A and votes B with the binary automaton in the presence of other votes $(C, D, \ldots)$, nodes with vote A start with initial state 0, nodes with vote B start with initial state 1, and nodes which hold another vote initialize their state to coordinate G (here 0.5^- or 0.5^+) to be neutral. All in all, this sorting algorithm requires 2 communication bits and uses $O(|\mathscr{C}| \log |\mathscr{C}|)$ times the binary voting algorithm.

If time is more critical than memory, another voting or sorting solution is to run the k-ary automata (k =2, 3, or 4) *in parallel*, in a way that allows to compare the frequencies of apparition of all pairs of votes $(AB, AC, BC, \ldots)$. To rank votes using the binary automaton $(k = 2)$, one needs $\binom{|\mathscr{C}|}{2}$ parallel processes, thus requiring $2\binom{|\mathscr{C}|}{2} = |\mathscr{C}|(|\mathscr{C}| - 1)$ bits.

The number of parallel processes needed to compute the majority vote(s) for $|\mathscr{C}| > 4$ with the ternary (respectively, the quaternary) automaton is more complicated: it is the minimum number of triangles (respectively, complete graphs of 4 nodes) needed to cover the complete graph of $|\mathscr{C}|$ nodes. This number scales as $\Theta(|\mathscr{C}|^2)$. For $|\mathscr{C}| = 5$, it turns out that the best solution is hybrid and demands a total of 15 bits. To compute majority out of votes A, B, C, D, E, one can compare in parallel

$$\begin{array}{ll} A, B, C, D & \text{using the quaternary automaton} \quad (7 \text{ bits}) \\ A, B, E & \text{using the ternary automaton} \quad (4 \text{ bits}) \\ C, D, E & \text{using the ternary automaton} \quad (4 \text{ bits}). \end{array}$$

Note that this procedure only computes the majority vote, but we are only a small step away from a *full sorting* of the vote frequencies, i.e., from solving the ranking problem. In the ternary automaton of Section III-1, in the ternary automaton of Fig. 7 and in the quaternary automaton of Section VI-C, every cell corresponds to a specific vote ranking (see Fig. 11). Therefore, if the algorithm which detects consensus computes the converging cells of the parallel processes, every node can deduce the whole vote ranking from the partial rankings.

C. Open Problems

Many related questions remain open. Do voting automata exist for any $|\mathscr{C}|$? The speed of convergence of the 2-bit automaton was studied in [20], but what is the speed of convergence of a multiple voting automaton? How can the distribution f of activated edges be optimized to speed up convergence? Given a spatio-temporal process of message/agent losses, what is the probability of error of the algorithm? Most importantly, what is the minimum number of bits that agents need to store/exchange in order to solve the multiple voting problem for a given $|\mathscr{C}|$?

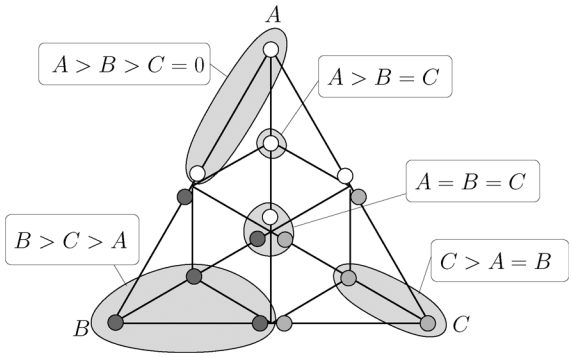


Fig. 11. Deducing ranking from cells.

REFERENCES

[1] F. Bénézit, P. Thiran, and M. Vetterli, "Interval consensus: From quantized gossip to voting," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, 2009, pp. 3661–3664.
[2] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Gossip algorithms: Design, analysis and applications," in *Proc. 24th Conf. IEEE Commun. Soc. (INFOCOM'05)*, 2005, pp. 1653–1664.
[3] S. Kar and J. M. F. Moura, "Distributed consensus algorithms in sensor networks: Quantized data," *CoRR*, vol. abs/0712.1609, 2007.
[4] T. C. Aysal, M. Coates, and M. Rabbat, "Distributed average consensus using probabilistic quantization," in *Proc. IEEE/SP 14th Workshop Statist. Signal Process. (SSP '07)* , Aug. 2007, pp. 640–644.
[5] A. Kashyap, T. Başar, and R. Srikant, "Quantized consensus," *Automatica*, vol. 43, no. 7, pp. 1192–1203, 2007.
[6] M. Land and R. K. Belew, "No perfect two-state cellular automata for density classification exists," *Phys. Rev. Lett.*, vol. 74, no. 25, pp. 5148–5150, Jun. 1995.
[7] P. G. de Sá and C. Maes, "The Gacs–Kurdyumov–Levin automaton revisited," *J. Statist. Phys.*, vol. 67, pp. 507–522, May 1992.
[8] N. H. Mustafa and A. Pekec, "Majority consensus and the local majority rule," in *Proc. 28th Int. Coll. Automata, Lang. Program. (ICALP '01)* , London, U.K., 2001, pp. 530–542, Springer-Verlag.
[9] D. Peleg, "Local majority voting, small coalitions and controlling monopolies in graphs: A review," Jerusalem, Israel, 1996, Tech. Rep..
[10] U. Raghavan, R. Albert, and S. Kumara, "Near linear time algorithm to detect community structures in large-scale networks," *Phys. Rev. E*, vol. 76, no. 3, p. 36106, 2007.
[11] H. Fukś, "Solution of the density classification problem with two cellular automata rules," *Phys. Rev. E*, vol. 55, no. 3, pp. R2081–R2084, Mar. 1997.
[12] M. S. Capcarrere, M. Sipper, and M. Tomassini, "Two-state, $r = 1$ cellular automaton that classifies density," *Phys. Rev. Lett.*, vol. 77, no. 24, pp. 4969–4971, Dec. 1996.
[13] M. S. Capcarrère and M. Sipper, "Necessary conditions for density classification by cellular automata," *Phys. Rev. E*, vol. 64, no. 3, p. 036113, Aug. 2001.
[14] Y. Hassin and D. Peleg, "Distributed probabilistic polling and applications to proportionate agreement," in *Proc. ICALP*, 1999, pp. 402–411.
[15] M. Tomassini and M. Venzi, "Evolution of asynchronous cellular automata for the density task," in *PPSN VII: Proc. 7th Int.Conf. Parallel Problem Solving From Nature*, London, U.K., 2002, pp. 934–944, Springer-Verlag.
[16] H. Fukś, "Nondeterministic density classification with diffusive probabilistic cellular automata," *Phys. Rev. E*, vol. 66, no. 6, p. 066106, Dec. 2002.
[17] D. Angluin, J. Aspnes, Z. Diamadi, M. Fischer, and R. Peralta, "Computation in networks of passively mobile finite-state sensors," *Distrib. Comput.*, vol. 18, no. 4, pp. 235–253, 2006.
[18] F. Bénézit, http://lcavwww.epfl.ch/~benezit/ternaryAutomaton.mat, Ternary Automaton.
[19] F. Bénézit, http://lcavwww.epfl.ch/~benezit/automaton.mat, Quaternary Automaton.
[20] M. Draief and M. Vojnovic, "Convergence speed of binary interval consensus," Microsoft Research, Tech. Rep. MSR-TR-2009-86, Aug. 2009.

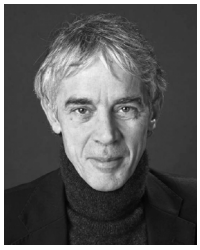
Florence Bénézit graduated from Ecole Polytechnique, Paris, France, in 2006 and received the Ph.D. degree from EPFL, Lausanne, Switzerland, in 2009.

She is currently a Postdoctoral Scholar at Ecole Normale Supérieure/INRIA, Paris. Her research interests include distributed signal processing, network automata and network tomography.

Patrick Thiran received the electrical engineering degree from the Université Catholique de Louvain, Louvain-la-Neuve, Belgium, in 1989, the M.S. degree in electrical engineering from the University of California, Berkeley, in 1990, and the Ph.D. degree from EPFL, Lausanne, Switzerland, in 1996.

He is an Associate Professor at EPFL. He became an Adjunct Professor in 1998, an Assistant Professor in 2002, and an Associate Professor in 2006. From 2000 to 2001, he was with Sprint Advanced Technology Labs, Burlingame, CA. His research interests include communication networks, performance analysis, dynamical systems, and stochastic models. He is currently active in the analysis and design of wireless multihop networks and in network monitoring.

Dr. Thiran served as an Associate Editor for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS from 1997 to 1999, and he is currently an Associate Editor for the IEEE/ACM TRANSACTIONS ON NETWORKING. He was the recipient of the 1996 EPFL Ph.D. award and of the 2008 Crédit Suisse Teaching Award.



Martin Vetterli (M'86–SM'90–F'95) received the Dipl.El.-Ing. degree from ETHZ, Zurich, Switzerland, in 1981, the M.S. degree from Stanford University, Stanford, CA, in 1982, and the Doctorat ès Sciences degree from EPFL, Lausanne, Switzerland, in 1986.

He was a Research Assistant at Stanford and EPFL and has worked for Siemens and AT&T Bell Laboratories. In 1986, he joined Columbia University, New York, where he was last an Associate Professor of Electrical Engineering and codirector of the Image and Advanced Television Laboratory. In 1993, he joined the University of California, Berkeley, where he was a Professor in the Department of Electrical Engineering and Computer Sciences until 1997, and now holds an Adjunct Professor position. Since 1995, he has been a Professor of Communication Systems at EPFL, where he chaired the Communications Systems Division (1996/1997), and heads the Audiovisual Communications Laboratory. From 2001 to 2004, he directed the National Competence Center in Research on mobile information and communication systems. He has also been a Vice-President at EPFL since October 2004 in charge of, among others, international affairs and computing services. He has held visiting positions at ETHZ (1990) and Stanford (1998). He has published about 140 journal papers on a variety of topics in signal/image processing and communications and holds a dozen patents. His research interests include sampling, wavelets, multirate signal processing, computational complexity, signal processing for communications, digital image/video processing, joint source/channel coding, signal processing for sensor networks, and inverse problems like acoustic tomography.

He is a fellow of ACM and of EURASIP and a member of SIAM. He is on the editorial boards of *Applied and Computational Harmonic Analysis*, *The Journal of Fourier Analysis and Application* and the IEEE JOURNAL OF SELECTED TOPICS IN SIGNAL PROCESSING. He received the Best Paper Award from EURASIP in 1984, the Research Prize of the Brown Bovery Corporation (Switzerland) in 1986, the IEEE Signal Processing Society's Senior Awards in 1991, in 1996 and in 2007 (for papers with D. LeGall, K. Ramchandran, and P. Marziliano and T. Blu, respectively). He won the Swiss National Latsis Prize in 1996, the SPIE Presidential award in 1999, the IEEE Signal Processing Technical Achievement Award in 2001, and is an ISI highly cited researcher in engineering. He was a member of the Swiss Council on Science and Technology from 2000 to 2003. He was a plenary speaker at various conferences (e.g., IEEE ICIP, ICASSP, ISIT) and is the coauthor of three books with J. Kovacevic, *Wavelets and Subband Coding* (Prentice-Hall, 1995), with P. Prandoni *Signal Processing for Communications* (EFPL Press, 1998), and with J. Kovacevic and V. K. Goyal, *The World of Fourier and Wavelets*.