A Topological Perspective on Distributed Network Algorithms

Armando Castañeda[1], Pierre Fraigniaud[2(✉)], Ami Paz[2], Sergio Rajsbaum[1], Matthieu Roy[3], and Corentin Travers[4]

[1] UNAM, Mexico City, Mexico
{armando.castaneda,rajsbaum}@im.unam.mx
[2] CNRS and Université de Paris, Paris, France
{pierref,amipaz}@irif.fr
[3] CNRS, Toulouse, France
roy@laas.fr
[4] CNRS and University of Bordeaux, Bordeaux, France
travers@labri.fr

Abstract. More than two decades ago, combinatorial topology was shown to be useful for analyzing distributed fault-tolerant algorithms in shared memory systems and in message passing systems. In this work, we show that combinatorial topology can also be useful for analyzing distributed algorithms in networks of arbitrary structure. To illustrate this, we analyze consensus, set-agreement, and approximate agreement in networks, and derive lower bounds for these problems under classical computational settings, such as the LOCAL model and dynamic networks.

Keywords: Distributed computing · Distributed graph algorithms · Combinatorial topology

1 Introduction

1.1 Context and Objective

A breakthrough in distributed computing was obtained in the 1990's, when *combinatorial topology*, a branch of Mathematics extending graph theory to higher dimensional objects, was shown to provide a framework in which a large variety of models can be studied [29,41]. Combinatorial topology provides a powerful arsenal of tools, which considerably expended our understanding of the solvability and complexity of many distributed problems [2,9,10,30]. We refer to the book by Herlihy et al. [25] for an extended and detailed description of combinatorial topology applied to distributed computing, in a wide variety of settings.

In a nutshell, combinatorial topology allows us to represent all possible executions of a distributed algorithm, along with the relations between them, as a single mathematical object, whose properties reflect the solvability of a problem. Combinatorial topology was primarily used to study distributed computing

K. Censor-Hillel and M. Flammini (Eds.): SIROCCO 2019, LNCS 11639, pp. 3–18, 2019.
https://doi.org/10.1007/978-3-030-24922-9_1

in the context of shared memory and message passing systems, but not in the context of systems in which the presence of a network connecting the processing elements needs to be taken into account. On the other hand, a large portion of the study of distributed computing requires to take into account the structure of the network connecting the processors, e.g, when studying *locality*. This paper is a first attempt to approach distributed network computing through the lens of combinatorial topology.

The base of the topological approach for distributed computing consists of modeling all possible input (resp., output) configurations as a single object called input *complex* (resp., output complex), and specifying a task as a relation between the input and output complexes. Moreover, computation in a given model results in a topological deformation that modifies the input complex into another complex called the *protocol complex*. The fundamental result of combinatorial topology applied to distributed computing [25] is that a task is solvable in a computational model if and only if there exists a simplicial mapping, called *decision map*, from the protocol complex to the output complex, that agrees with the specification of the task. In other words, for every input configuration, (1) the execution of the algorithm should lead the system into one or many configurations, forming a subcomplex of the protocol complex, and (2) the decision map should map every configuration in this subcomplex (i.e., each of its *simplexes*) into a configuration of the output complex that is legal for the given input configuration, with respect to the specification of the task.

Understanding the power and limitation of a distributed computing model with respect to solving a given task requires to understand under which condition the decision map exists. This requires to understand the nature of topological deformations of the input complex resulting from the execution of an algorithm, and the outcome of this deformation, i.e., the protocol complex. That is, one needs to establish the connections between the distributed computing model at hand, and the topological deformations incurred by the input complex in the course of a computation under this model.

The connections between the computational models and the topological deformations are now well understood for several distributed computing models. For instance, in shared-memory wait-free systems, the protocol complex results from the input complex by a series of specific *subdivisions* of its simplexes. Note that the impossibility result for consensus in shared-memory wait-free systems is a direct consequence of this fact, as the input complex of consensus is connected, subdivisions maintain connectivity, but the output complex of consensus is not connected—this prevents the existence of a decision map, independently of how long the computation proceeds. Similarly, in shared-memory t-resilient systems, the protocol complex results from the input complex not only by a series of specific subdivisions, but also by the appearance of some *holes* in the course of the computation. This is because every process can wait for hearing from at least $n - t$ other processes in any n-node t-resilient system. These holes enable the existence of a decision map in the case of $(t + 1)$-set-agreement, but are not sufficient to enable the existence of a decision map for consensus, as long

as $t \geq 1$. And indeed, the FLP result [19] implies that consensus is not solvable in asynchronous systems even in the presence of at most one failure.

This paper addresses the following issues: What is the nature of the topological deformations incurred by the input complex in the context of network computing, i.e., when nodes are bounded to interact only with nearby nodes according to some graph metric? And, what is the impact of these deformations on the ability to solve tasks efficiently (e.g., locally) in networks? As a first step towards answering these questions in general, we tackle them in the framework of synchronous failure-free computing, which is actually the framework in which most studies of distributed network computing are conducted [37].

1.2 Our Results

We place ourselves in the context of synchronous failure-free computing in networks [37]. As a first step towards understanding the nature of computation in this model from a topological perspective, we focus on lower bounds. We make a simplifying assumption which significantly strengthens the model, and therefore strengthens our lower bounds as well. We assume *structure awareness*. This assumption essentially asserts that each processing node is fully aware of the network it belongs to. More specifically, it assumes that all processes are given the same adjacency matrix of the network, and every process is given the index in the matrix of the vertex it occupies in the network. Structural awareness makes many tasks trivial. This is, for instance, the case of graph problems such as computing a vertex-coloring, an independent set, or a matching, which are among the main concerns of distributed network computing. Nevertheless, input-output tasks such as consensus and set-agreement, which are less studied in networks, yet important tasks as far as distributed computing and combinatorial topology are concerned [40], remain non-trivial.

The main contribution of this paper is in studying the topological model of distributed computing in networks, under the assumption of structure awareness. In particular, we show that the protocol complex involves deformations that were not observed before in the context of distributed computing, deformations which we call *scissor cuts*. These cuts appear between the facets of the input complex, and depend on the structure of the underlying network governing the way the information flows between nodes.

We show that this characterization is useful for deriving lower bounds on agreement tasks. For this purpose, we model the way information flows between nodes in the network by the so-called *information-flow graph*, and establish tight connections between structural properties of this graph, and the ability to solve agreement tasks in the network. This is achieved thanks to our understanding of the topology of the protocol complex. For instance, we show that if the domination number of the information-flow graph is at least $k+1$, then the protocol complex is at least $(k-1)$-connected, and if the protocol complex is at least $(k-1)$-connected, then k-set agreement is not solvable.

Interestingly, our results connecting the structure of the information-flow graph with the topology of the protocol complex, imply lower bounds for solving

agreement problems in the classical LOCAL model, as well as in dynamic networks. For instance, a consequence of our results is that, in the LOCAL model, solving k-set agreement in a network requires at least r rounds, where r is the smallest integer such that the r-th power of the network (two nodes are adjacent when their distance in the network is at most r) has domination number at most k. Similarly, we show that solving k-set agreement in a dynamic network $(H_t)_{t\geq 1}$ requires at least r rounds, where r is the smallest integer such that $(H_t)_{1\leq t\leq r}$ has temporal dominating number at most k.

Applying the topological approach to network computing also enables to derive fine grained results. For instance, we show that in every n-node network where consensus is not solvable, ϵ-approximate agreement is also not solvable whenever $\epsilon < \frac{1}{n-1}$. This bound is tight, in the sense that there exists a network where consensus is impossible, while $\frac{1}{n-1}$-approximate agreement is solvable.

1.3 Related Work

The deep connections between combinatorial topology and distributed computing were concurrently and independently identified in [29] and [41]. Since then, numerous outstanding results were obtained using combinatorial topology for various types of tasks, including agreement tasks such as consensus and set-agreement [40], and symmetry breaking tasks such as renaming [2,9,10]. A recent work [1] provides evidence that topological arguments are sometimes necessary. All these contributions were obtained in the asynchronous shared memory model with crash failures, but combinatorial topology was shown to be applicable to Byzantine failures as well [36]. Note that the message passing model restricts itself to complete graphs [16,28]. Recent results showed that combinatorial topology can also be applied in the analysis of mobile computing [38], demonstrating the generality and flexibility of the topological framework applied to distributed computing. The book [25] provides an extensive introduction to combinatorial topology applied to distributed computing.

In contrast, distributed network computing has not been impacted by combinatorial topology. This domain of distributed computing is extremely active and productive this last decade, analyzing a large variety of network problems in the so-called LOCAL model [37], capturing the ability to solve task locally in networks[1]. We refer to [4,5,8,13,18,20,21,24,42] for a non exhaustive list of achievements in context. However, all these achievements were based on an operational approach, using sophisticated algorithmic techniques and tools solely from graph theory. Similarly, the existing lower bounds on the round-complexity of tasks in the LOCAL model [3,8,23,32,35] were obtained using graph theoretical and combinatorial arguments. The question of whether adopting a higher dimensional approach by using topology would help in the context of local computing, be it for a better conceptual understanding of the algorithms, or providing stronger technical tools for lower bounds, is, to our knowledge, entirely open.

[1] The CONGEST model has also been subject of tremendous progresses, but this model does not support full information protocols, and thus is out of the scope of our paper.

Similarly to (static) distributed network computing, the fundamental research on dynamic networks [6,11,12,34] has rarely been impacted by combinatorial topology. Relevant works in this framework study consensus [17,33], set-agreement [7,22] and approximate agreement [14]. We also refer to [15,31,39] which analyze distributed computation in a model where all processes broadcast messages at each round, but the recipients of these messages are defined by a graph which may change from round to round. The information-flow graph introduced and analyzed in this paper can be viewed as an abstraction of computation in dynamic networks, as this graph contains a summary of how information was transmitted among processes in the network during some interval of time.

2 Model and Definitions

In this section, we describe an abstract model of computation that captures various models of distributed computing, including the LOCAL model, and computing in dynamic graphs. This model is called KNOW-ALL, for reason that will soon be apparent.

2.1 The KNOW-ALL Model

We consider a set of n synchronous fault-free processes, with distinct names in $\{1,\ldots,n\}$, all running the same algorithm. The processes can model computing entities exchanging messages through a network, but also software agents or physical robots moving in space and exchanging messages whenever they meet, or computing entities in a dynamic network whose links evolve over time. The processes communicate using some communication medium, and the interactions are specified by a sequence $\mathcal{H}$ of n-node directed labeled graphs $(H_t)_{1\leq t\leq T}$. The label of a node of H_t is a value in $\{1,\ldots,n\}$, different from the labels of all other nodes. The process with name $p \in \{1,\ldots,n\}$ occupies the node labeled p in each of the graphs H_t, $1 \leq t \leq T$. The arcs in H_t represent the interactions that can take place at the t-th rounds of an algorithm. The core property of the KNOW-ALL model is that every process is a priori given its name, and the sequence $\mathcal{H} = (H_t)_{1\leq t\leq T}$, so every node is given the complete knowledge of the communication patterns occurring during the T rounds. The only uncertainty is about the inputs to the nodes.

The KNOW-ALL model is stronger than several classical distributed computing models. For example, the LOCAL model is also synchronous, fault-free model but with a fixed communication graph H, i.e., $H_t = H$ for every $t \geq 1$, and the nodes learn only some of the graph topology during an execution. A dynamic graph computation is defined by a sequence of graphs on the same set of nodes, and the nodes only gain partial information on the graph sequence during the execution. This is generalized by the KNOW-ALL model, where all the graph sequence is given in advance to the processes. Hence, in both cases the KNOW-ALL model is stronger than the classical model, and lower bounds proven for the KNOW-ALL model imply lower bounds for the other models as well.

By no means we claim the KNOW-ALL model to be practical. We make several simplifying assumptions that are typical in these settings: unbounded computational power, unbounded communication, failure-freeness, and also structural awareness, which is not a typical assumption. However, this strong model is sufficient for exhibiting lower bounds, and for establishing impossibility results for weaker, more realistic models. More important perhaps, it enables us to exhibit interesting phenomenon regarding the impact of the communication pattern on the topology of the protocol complex.

2.2 Input-Output Problems and the Information-Flow Graph

We focus on input-output problems, naturally defined as follows. A task (I, O, F) in the n-process KNOW-ALL model is described by a set I of input values, a set O of output values, and a mapping

$$F : I^n \to 2^{O^n}$$

specifying, for every n-tuple of input values, the set of possible legal n-tuple of output values. (In the topological sense, we focus on tasks for which the input complex is a pseudosphere, as explained below.) The input value of process p is denoted by $in(p) \in I$.

A distributed algorithm solving a task has two components: a communication protocol enabling each process to gather information about the inputs of other processes, and a decision function f that maps the gathered information to an output value. In the KNOW-ALL model, we can restrict our attention to considering only *flooding* protocols. At round t of such a protocol, every process p sends to all its out-neighbors in H_t all the name-input pairs it is aware of, that is, the pair $(p, in(p))$, and all the pairs it has received in the previous rounds. After T rounds, the process takes a decision based on the set of pairs it is aware of. Considering only flooding protocols does not reduce the computational power, as the structural awareness allows each process to simulate any other protocol.

Assuming flooding protocols, designing an algorithm boils down to designing a decision function f which allows each process, given the set of received input values, to compute an output value such that the collection of output values produced by the processes is consistent with the collection of input values. More specifically, for every vector of input values $(v_1, \ldots, v_n) \in I^n$, given to process $(p_1, \ldots, p_n)$, respectively, let w_i be the vector where for every $j \in \{1, \ldots, n\}$,

$$w_i[j] = \begin{cases} v_j \text{ if } j = i, \text{ or process } i \text{ receives the pair } (j, v_j) \text{ when flooding in } \mathcal{H}; \\ \perp \text{ otherwise.} \end{cases}$$

Then, every process $i \in \{1, \ldots, n\}$ must compute an output value

$$v'_i = f\,(i, w_i)$$

such that the resulting n-tuple $(v'_1, \ldots, v'_n)$ is in $F(v_1, \ldots, v_n)$.

In order to analyze flooding protocols, we define the *information-flow graph*, which describes the execution of a flooding protocol in the KNOW-ALL model.

Definition 1. *Let $\mathcal{H} = (H_t)_{1 \leq t \leq T}$ be an instance of the* KNOW-ALL *model. The* information-flow graph *associated with $\mathcal{H}$ is the directed graph G whose n nodes are labeled by $1, \ldots, n$, and there is an arc (p, q) from p to q in G if q receives the pair $(p, in(p))$ when flooding in $\mathcal{H}$.*

A crucial observation is that whenever two instances $\mathcal{H}$ and $\mathcal{H}'$ of the KNOW-ALL model yield the same information flow graph, then these two instances have the same computational power. The structure of the information-flow graph has a crucial impact on the ability to solve input-output problems in the KNOW-ALL model, an impact which we study in this paper. In order to clarify the impact of the structure of the information flow graph on the ability to solve problems, we apply techniques of combinatorial topology.

3 Topological Description of the KNOW-ALL Model

3.1 Basics Definitions

A *simplicial complex* is a finite set V along with a collection of nonempty subsets $\mathcal{K}$ of V closed under containment (i.e., if $A \in \mathcal{K}$ and $\emptyset \neq B \subset A$, then $B \in \mathcal{K}$). An element of V is called a *vertex* of $\mathcal{K}$, and the vertex set of $\mathcal{K}$ is denoted by $V(\mathcal{K}) = V$. Each set in $\mathcal{K}$ is called a *simplex*. A subset of a simplex is called a *face* of that simplex. The *dimension* $\dim \sigma$ of a simplex σ is one less than the number of elements of σ, i.e., $|\sigma| - 1$. We use "d-face" as shorthand for "d-dimensional face". A simplex σ in $\mathcal{K}$ is called a *facet* of $\mathcal{K}$ if σ is not contained in any other simplex. Note that a set of facets uniquely defines a simplicial complex. The dimension of a complex is the largest dimension of any of its facets. A complex is *pure* if all its facets have the same dimension.

Let $\mathcal{K}$ and $\mathcal{L}$ be complexes. A *vertex map* is a function $h : V(\mathcal{K}) \to V(\mathcal{L})$. If h also carries simplexes of $\mathcal{K}$ to simplexes of $\mathcal{L}$, it is called a *simplicial map*. We add one or more *labels* to the vertices, $\lambda : V \to D$, where D is an arbitrary domain. In particular, we have the set $\{1, \ldots, n\}$ of process names, and a label associating each vertex with a name. Typically, each simplex is *properly colored* by these names: if u and v are distinct vertices of a simplex σ, then $\text{name}(u) \neq \text{name}(v)$. A simplicial map h is *chromatic* if it preserves names, i.e., $\text{name}(h(v)) = \text{name}(v)$ for any vertex v. In this paper, all simplicial maps between colored complexes will be chromatic. Given two complexes $\mathcal{K}$ and $\mathcal{L}$, a *carrier map* Φ maps each simplex $\sigma \in \mathcal{K}$ to a sub-complex $\Phi(\sigma)$ of $\mathcal{L}$, such that for every two simplexes τ and τ' in $\mathcal{K}$ that satisfy $\tau \subseteq \tau'$, we have $\Phi(\tau) \subseteq \Phi(\tau')$.

Roughly speaking, a *geometric realization* $|\mathcal{K}|$ of a simplicial complex $\mathcal{K}$ is a geometric object defined as follows. Each vertex in $V(\mathcal{K})$ is mapped to a point in a Euclidean space, such that the images of the vertices are affinely independent. Each simplex is represented by a polyhedron, which is the convex hull of points representing its vertices. Figure 1 displays the geometric representations of several simplicial complexes that are detailed later.

Let k be a positive integer. We say that a complex has *a hole in dimension k* if the k-sphere S^k embedded in a geometric realization of the complex cannot be continuously contracted to a single point within that realization. Informally, a complex is *k-connected* if it has no holes in dimension k. A complex $\mathcal{K}$ is *k-connected* if every continuous map $h : S^k \to |\mathcal{K}|$ can be extended to a continuous map $h' : D^{k+1} \to |\mathcal{K}|$ where D^{k+1} denotes the $(k+1)$-disk. In dimension 0, this property simply states that any two points can be linked by a path, i.e., the complex is path-connected. In dimension 1, it states that any loop can be filled into a disk, i.e., the complex is simply connected. By convention, a (-1)-connected complex is just a non-empty complex.

Finally, given a set I, a *pseudosphere* $\Psi(\{1,\ldots,n\}, I)$ is the complex defined as follows: (1) every pair (i, v) is a vertex, where $v \in I$, and (2) for every index set $J \subseteq \{1,\ldots,n\}$, and every multi-set $\{v_j : j \in J\}$ of values, the set $\{(j, v_j) : j \in J\}$ is a simplex. Pseudospheres offer a convenient way to describe all possible initial configurations where each process input is an arbitrary value from I.

3.2 The Topology of Computing in the KNOW-ALL Model

Given a distributed computing task (I, O, F) to be solved in the KNOW-ALL model, two complexes play a major role in this framework, the *input complex*, denoted by $\mathcal{I}$, and the *output complex*, denoted by $\mathcal{O}$. Let us fix an information flow graph G. The input complex $\mathcal{I}$ is the pseudosphere $\Psi(\{1,\ldots,n\}, I)$, also defined by its set of facets

$$\big\{\{(1, v_1),\ldots,(n, v_n)\} : v_i \in I\big\}.$$

The set of all facets of the output complex $\mathcal{O}$ is

$$\big\{\{(1, v'_1),\ldots,(n, v'_n)\} : v'_i \in O, \text{ and } \exists v \in I^n, (v'_1,\ldots,v'_n) \in F(v)\big\}.$$

Note that the output complex includes only combinations of output values that are legal with respect to the problem at hand. Note also that the input and output complexes do not depend on the communication medium considered, and that both complexes are pure—all their facets have the same dimension.

For instance, in the case of binary consensus in an n-process system (see Fig. 1), the set of facets of the input complex is

$$\big\{\{(1, v_1),\ldots,(n, v_n)\} :\ v_i \in \{0, 1\}\big\}.$$

This complex is homeomorphic to the $(n-1)$-dimensional sphere S^{n-1}. For the same example, the output complex is composed of two disjoints $(n-1)$-facets, τ_0 and τ_1:

$$\tau_0 = \{(1, 0),\ldots,(n, 0)\}, \text{ and } \tau_1 = \{(1, 1),\ldots,(n, 1)\}.$$

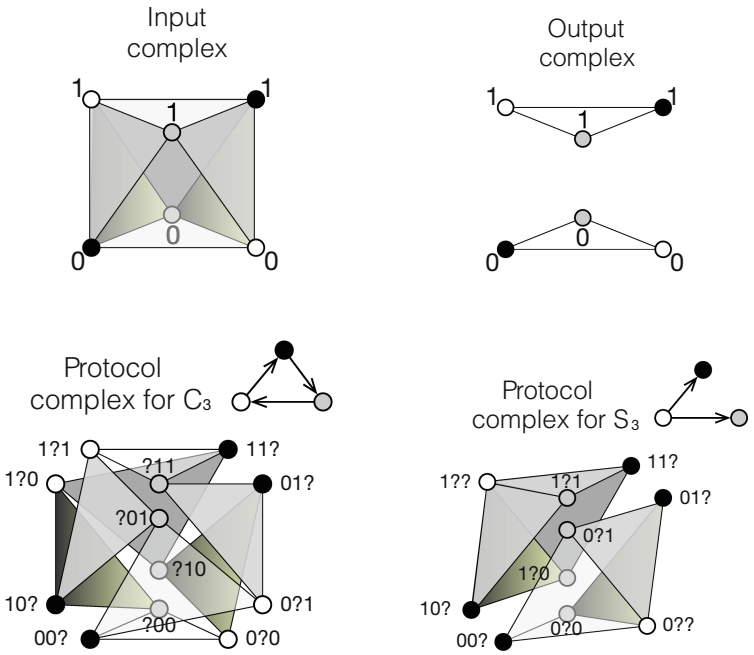

Fig. 1. Impact of the information flow graph on the protocol complex for binary consensus with three processes. Labels next to vertices are input and output values, in the input and output complexes respectively, or views in protocol complexes. A view "xyz" labeling a vertex means that the process corresponding to this vertex knows the input values x from process $\circ$, y from process $\bullet$, and z from process $\bullet$. A question mark in a label indicates that the process does not know the corresponding input value.

One can rephrase the operational definition (I, O, F) of *task* in Sect. 2.2 in the framework of combinatorial topology as follows: a task $(\mathcal{I}, \mathcal{O}, \Delta)$ is described by a carrier map Δ from $\mathcal{I}$ to $\mathcal{O}$. Note that, in absence of failures and asynchrony, a task can be described merely by a mapping Δ from the facets of $\mathcal{I}$ to subsets of facets of $\mathcal{O}$. For a given facet $\sigma = \{(1, v_1), \ldots, (n, v_n)\} \in \mathcal{I}$, the set of facets of $\Delta(\sigma)$ is defined by

$$\{(1, v'_1), \ldots, (n, v'_n)\} \in \Delta(\sigma) \iff (v'_1, \ldots, v'_n) \in F(v_1, \ldots, v_n). \tag{1}$$

The carrier map Δ of binary consensus maps every input facet σ containing both input values 0 and 1 to the two $(n-1)$-facets τ_0 and τ_1, and maps each $(n-1)$-facet σ_b with a unique input value $b \in \{0, 1\}$ to the output $(n-1)$-facet τ_b.

In any distributed computing model, in each point in time during the execution of an algorithm, one can define a complex whose vertices are pairs (p, w) where w is the state of process p, i.e., its view of the computation. A set of vertices with distinct process names forms a *protocol simplex* if there is a protocol execution where those processes collect those views. All possible protocol

simplexes make up the *protocol complex*. The following fact is a direct consequence of the definition of the information flow graph.

Fact 1. *Given an information flow graph G, and a task $(\mathcal{I}, \mathcal{O}, \Delta)$, the protocol complex $\mathcal{P}$ associated with G and $\mathcal{I}$ is the complex whose facet are all the sets of the form $\{(1, w_1), \ldots, (n, w_n)\}$ for which there exists a facet $\{(1, v_1), \ldots, (n, v_n)\}$ of $\mathcal{I}$ such that, for $i = 1, \ldots, n$, $w_i = \{(j, v_j) : i = j \text{ or } (j, i) \in E(G)\}$. We define a carrier map $\Xi : \mathcal{I} \to \mathcal{P}$ which carries each facet of $\mathcal{I}$ to a single facet of $\mathcal{P}$, satisfying*

$$\Xi(\{(1, v_1), \ldots, (n, v_n)\}) = \{(1, w_1), \ldots, (n, w_n)\}.$$

An important observation is that the facets of the input complex are preserved in the protocol complex, i.e., there is a one-to-one correspondence between the facets of these two complexes. This is because the computation is synchronous and failure-free, from which it follows that each input configuration yields a single configuration in the protocol complex.

Example. Figure 1 displays two illustrations of the protocol complex for binary consensus, for two different information flow graphs on three processes: the consistently directed cycle C_3, and the directed star S_3 whose center has two out-neighbors. Process names are omitted, and instead are represented by the colors of the circles ($\circ$, $\bullet$, and $\bullet$). The number of vertices in the protocol complexes depends on the information flow graph.

Let us focus first on process $\circ$. A vertex $(\circ, v)$ in the input complex yields two vertices in the protocol complex for C_3, depending on the input value received from process $\bullet$. Instead, a vertex $(\circ, v)$ in the input complex yields a single vertex in the protocol complex for S_3 because, according to this information flow graph, process $\circ$ receives no inputs from other processes. On the other hand, every vertex $(\bullet, v)$ in the input complex yields two vertices in both protocol complexes. This is because, in both information flow graphs, C_3 and S_3, process $\bullet$ receives the input from process $\circ$. Similarly, every vertex $(\bullet, v)$ in the input complex yields two vertices in both protocol complexes, because in both information flow graphs process $\bullet$ receives the input from another process, from process $\bullet$ in C_3 and from process $\circ$ in S_3.

3.3 Topological Characterization of Task Solvability

So far, we have proceeded in two parallel paths. The first, *operational* path, was about algorithms in the KNOW-ALL model, where information propagates between processes according to some information flow pattern G. The second, *topological* path, relates the inputs of processes defined by an input complex, their views modeled in the protocol complex, and their desired outputs, appearing in the output complex. The connections between these paths is established in the next fact, which directly follows from the definitions.

Fact 2. *A task (I, O, F) is solvable in the* KNOW-ALL *model with information flow graph G if and only if, for the topological formulation $(\mathcal{I}, \mathcal{O}, \Delta)$ of the task,*

there exists a chromatic simplicial map $\delta : \mathcal{P} \to \mathcal{O}$ *satisfying* $\delta(\Xi(\sigma)) \in \Delta(\sigma)$ *for every facet* $\sigma \in \mathcal{I}$*, where* $\mathcal{P}$ *is the protocol complex associated with* G *and* $\mathcal{I}$*.*

The simplicial map $\delta : \mathcal{P} \to \mathcal{O}$ is called *decision map*. If $\delta(i, w_i) = (i, v'_i)$, then the corresponding algorithm specifies that process i with view w_i outputs $f(i, w_i) = v'_i$.

Example. Let us consider Fig. 1 again. The protocol complex for S_3 is disconnected, while for C_3 it is 0-connected (i.e., path connected). The presence of a universal node $\circ$ (dominating all other nodes) in the information flow graph S_3 results in all processes becoming aware of the input of the process corresponding to that node. Therefore, the protocol complex for S_3 is split into two subcomplexes, the one corresponding to process $\circ$ with input 0, and the other corresponding to process $\circ$ with input 1. Similarly, the protocol complex for the complete graph K_3 with bidirectional edges is entirely disconnected, i.e., composed of eight pairwise non-intersecting facets, because there is no uncertainty under the complete information flow graph, as every process receives the input of every other process.

Since the protocol complex for S_3 is disconnected, consensus is solvable in this graph. To see why, consider δ that maps every vertex $(p, 0**)$ of the protocol complex to vertex $(p, 0)$ of the output complex, and every vertex $(p, 1**)$ of the protocol complex to vertex $(p, 1)$ of the output complex. This is a chromatic simplicial map, and thus, by Fact 2 consensus is solvable. In contrast, there is no such mapping $\delta : \mathcal{P} \to \mathcal{O}$ for the protocol complex $\mathcal{P}$ corresponding to C_3, because $\mathcal{P}$ is 0-connected. Let us consider the path $((\circ, 1?1), (\bullet, ?01), (\bullet, 00?))$ in the protocol complex for C_3. Vertex $(\circ, 1?1)$ must be mapped to vertex $(\circ, 1)$ in the output complex because $(\circ, 1?1)$ belongs to a facet with all processes having input value 1. Similarly, vertex $(\bullet, 00?)$ must be mapped to vertex $(\bullet, 0)$ because $(\circ, 00?)$ belongs to a facet with all processes having input value 0. If a mapping δ maps $(\bullet, ?01)$ to $(\bullet, 1)$, then the simplex $\{(\bullet, ?01), (\bullet, 00?)\}$ is mapped to $\{(\bullet, 1), (\bullet, 0)\}$, which is not a simplex of $\mathcal{O}$. The same occurs if $(\bullet, ?01)$ is mapped to $(\bullet, 0)$, as $\{(\circ, 1), (\bullet, 0)\}$ is not a simplex of $\mathcal{O}$. Thus, there is no simplicial map δ, and, by Fact 2, consensus is not solvable. We generalize this result to every information flow graph G, and to k-set agreement, for every $k \geq 1$.

4 Applications to Agreement Tasks

In this section, we illustrate the power of using topology for analyzing the KNOW-ALL model, and its implications on standard models such as LOCAL and dynamic networks. First, we establish a connection between the structure of the information flow graph resulting from some instance of the KNOW-ALL model on the one hand, and the topology of the protocol complex induced by this instance on the other hand. Recall that the *domination number* $\gamma(G)$ of a graph G is the number of vertices in a smallest dominating set for G, where, in directed graphs, a vertex u dominates a vertex v if $(u, v) \in E(G)$.

Theorem 1. *Let $\mathcal{H}$ be an instance of the* KNOW-ALL *model, and G be the information flow graph associated with it. If $\gamma(G) > k$, then the protocol complex $\mathcal{P}$ for $\mathcal{H}$ is at least $(k-1)$-connected.*

Recall that, in the k-set agreement task, the processes must agree on at most k of the input values. In the context of asynchronous shared memory computing, the level of connectivity of the protocol complex is closely related to the ability to solve k-set agreement [26,27,30]. Using a similar connection, Theorem 1 implies the following.

Theorem 2. *Let $\mathcal{H}$ be an instance of the* KNOW-ALL *model, and G be the information flow graph associated with it. If $\gamma(G) > k$, then k-set agreement is not solvable in $\mathcal{H}$.*

To establish Theorem 2, we show that if the protocol complex $\mathcal{P}$ for $\mathcal{H}$ is at least $(k-1)$-connected, then k-set agreement is not solvable in $\mathcal{H}$, and then we apply Theorem 1. Observe that the converse of Theorem 2 also holds, i.e., if $\gamma(G) \leq k$ then k-set agreement is solvable in $\mathcal{H}$. The algorithm performs as follows. Let D be a dominating set for G, with $|D| \leq k$. Since D is dominating, every process p receives the input value of at least one process in D, and can decide on such a value as an output. In total, at most $|D| \leq k$ values are decided.

Theorem 2 implies that, in particular, consensus solvability requires the information flow graph to contain a universal node, i.e., a node that dominates all the other nodes. This theorem has implications for more traditional computational models, including the LOCAL model. Given a graph H, and $r \geq 1$, let H^r denote the graph on the same set of nodes as H, but in which two nodes are adjacent if their distance in H is at most r.

Corollary 1. *In the* LOCAL *model, solving k-set agreement in a network H requires at least r rounds, where r is the smallest integer such that $\gamma(H^r) \leq k$.*

Theorem 2 also applies to dynamic networks, in which edges appear and disappear over time. A dynamic network is a sequence $\mathcal{G} = (G_t)_{t \geq 1}$ of graphs on the same set of nodes V, where G_t is the actual network at round t. A set $D \subseteq V$ is a *temporal* dominating set for $(G_t)_{1 \leq t \leq r}$ if, for every node $v \notin D$, there is a temporal path from some node $u \in D$ to v, i.e., a sequence $(u_0, \ldots, u_s)$ of nodes with $u_0 = u$ and $u_s = v$, and a sequence $1 \leq t_0 < t_1 < \cdots < t_s \leq r$ of rounds such that $\{u_i, u_{i+1}\} \in E(G_{t_i})$ for every $i = 0, \ldots, s-1$.

Corollary 2. *Solving k-set agreement in dynamic network $\mathcal{G} = (G_t)_{t \geq 1}$ requires at least r rounds, where r is the smallest integer such that $(G_t)_{1 \leq t \leq r}$ has a temporal dominating set D with $|D| \leq k$.*

Finally, recall that, for $\epsilon \in [0, 1]$, binary ϵ-approximate agreement requires the processes to output values that are not more than ϵ apart, under the condition that if all the processes have the same input value $v \in \{0, 1\}$, then they all should output v. Using topological arguments applied to the information flow graph associated with the given instance $\mathcal{H}$ of the KNOW-ALL model, we show the following.

Theorem 3. *Let $\mathcal{H}$ be an instance of the* KNOW-ALL *model. If consensus is impossible under $\mathcal{H}$, then, for every $\epsilon < \frac{1}{n-1}$, ϵ-approximate agreement is also not solvable under $\mathcal{H}$. This bound is tight in the sense that there exists an instance $\mathcal{H}$ of the* KNOW-ALL *model for which consensus is impossible, while $\frac{1}{n-1}$-approximate agreement is solvable.*

The same way Theorem 2 has consequences on the complexity of solving k-set agreement in concrete computational models such as the LOCAL model and dynamic networks, Theorem 3 has similar consequences on the complexity of solving approximate agreement in these latter models.

5 Conclusion and Further Work

We demonstrate that combinatorial topology is applicable to distributed network computing. Of course, this is just a first step, and further work will require incorporating features of every distributed network model, in order to capture the specific characteristics of each of them. For instance, fully capturing the popular LOCAL model requires removing the structure awareness assumption, and studying the details of how the protocol complex evolves round after round.

Incorporating asynchrony and failures into network computing, from a topological perspective, requires understanding the topological impact of simultaneously stretching the facets, introducing holes resulting from t-resiliency, and introducing scissor cuts resulting from the presence of a network. This is definitely technically challenging, but our paper shows that there are no conceptual obstacles preventing us from addressing these questions.

Acknowledgments. Pierre Fraigniaud and Corentin Travers are supported by ANR projects DESCARTES and FREDA; Pierre Fraigniaud receives additional support from INRIA project GANG; Ami Paz is supported by the Fondation Sciences Mathématiques de Paris (FSMP); Sergio Rajsbaum is supported by project unam-papiit IN109917.

References

1. Alistarh, D., Aspnes, J., Ellen, F., Gelashvili, R., Zhu, L.: Why extension-based proofs fail. CoRR abs/1811.01421 http://arxiv.org/abs/1811.01421 (2018). To appear in STOC 2019
2. Attiya, H., Castañeda, A., Herlihy, M., Paz, A.: Bounds on the step and namespace complexity of renaming. SIAM J. Comput. **48**(1), 1–32 (2019). https://doi.org/10.1137/16M1081439
3. Balliu, A., Brandt, S., Hirvonen, J., Olivetti, D., Rabie, M., Suomela, J.: Lower bounds for maximal matchings and maximal independent sets. CoRR abs/1901.02441 http://arxiv.org/abs/1901.02441 (2019)
4. Barenboim, L., Elkin, M., Goldenberg, U.: Locally-iterative distributed $(\delta + 1)$-coloring below szegedy-vishwanathan barrier, and applications to self-stabilization and to restricted-bandwidth models. In: Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing, (PODC), pp. 437–446 (2018). https://dl.acm.org/citation.cfm?id=3212769

5. Barenboim, L., Elkin, M., Pettie, S., Schneider, J.: The locality of distributed symmetry breaking. In: 53rd IEEE Symposium on Foundations of Computer Science (FOCS), pp. 321–330 (2012). https://doi.org/10.1109/FOCS.2012.60
6. Bhadra, S., Ferreira, A.: Computing multicast trees in dynamic networks and the complexity of connected components in evolving graphs. J. Internet Services Appl. **3**(3), 269–275 (2012). https://doi.org/10.1007/s13174-012-0073-z
7. Biely, M., Robinson, P., Schmid, U., Schwarz, M., Winkler, K.: Gracefully degrading consensus and k-set agreement in directed dynamic networks. Theor. Comput. Sci. **726**, 41–77 (2018). https://doi.org/10.1016/j.tcs.2018.02.019
8. Brandt, S., et al.: A lower bound for the distributed Lovász local lemma. In: 48th ACM Symposium on Theory of Computing (STOC), pp. 479–488 (2016). https://doi.org/10.1145/2897518.2897570
9. Castañeda, A., Rajsbaum, S.: New combinatorial topology bounds for renaming: the lower bound. Distrib. Comput. **22**(5–6), 287–301 (2010). https://doi.org/10.1007/s00446-010-0108-2
10. Castañeda, A., Rajsbaum, S.: New combinatorial topology bounds for renaming: the upper bound. J. ACM **59**(1), 3:1–3:49 (2012). https://doi.org/10.1145/2108242.2108245
11. Casteigts, A., Flocchini, P., Godard, E., Santoro, N., Yamashita, M.: On the expressivity of time-varying graphs. Theor. Comput. Sci. **590**, 27–37 (2015). https://doi.org/10.1016/j.tcs.2015.04.004
12. Casteigts, A., Flocchini, P., Quattrociocchi, W., Santoro, N.: Time-varying graphs and dynamic networks. Int. J. Parallel Emergent Distrib. Syst. **27**(5), 387–408 (2012). https://doi.org/10.1080/17445760.2012.668546
13. Chang, Y., Li, W., Pettie, S.: An optimal distributed$(\Delta + 1)$-coloring algorithm? In: 50th ACM Symposium on Theory of Computing (STOC), pp. 445–456 (2018). https://doi.org/10.1145/3188745.3188964
14. Charron-Bost, B., Függer, M., Nowak, T.: Approximate consensus in highly dynamic networks: the role of averaging algorithms. In: Halldórsson, M.M., Iwama, K., Kobayashi, N., Speckmann, B. (eds.) ICALP 2015. LNCS, vol. 9135, pp. 528–539. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-47666-6_42
15. Charron-Bost, B., Schiper, A.: The heard-of model: computing in distributed systems with benign faults. Distrib. Comput. **22**(1), 49–71 (2009). https://doi.org/10.1007/s00446-009-0084-6
16. Chaudhuri, S., Herlihy, M., Lynch, N.A., Tuttle, M.R.: Tight bounds for k-set agreement. J. ACM **47**(5), 912–943 (2000). https://doi.org/10.1145/355483.355489
17. Coulouma, E., Godard, E., Peters, J.G.: A characterization of oblivious message adversaries for which consensus is solvable. Theor. Comput. Sci. **584**, 80–90 (2015). https://doi.org/10.1016/j.tcs.2015.01.024
18. Fischer, M., Ghaffari, M., Kuhn, F.: Deterministic distributed edge-coloring via hypergraph maximal matching. In: 58th IEEE Annual Symposium on Foundations of Computer Science (FOCS), pp. 180–191 (2017). https://doi.org/10.1109/FOCS.2017.25
19. Fischer, M.J., Lynch, N.A., Paterson, M.: Impossibility of distributed consensus with one faulty process. J. ACM **32**(2), 374–382 (1985). https://doi.org/10.1145/3149.214121
20. Ghaffari, M.: An improved distributed algorithm for maximal independent set. In: 27th ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 270–277 (2016). https://doi.org/10.1137/1.9781611974331.ch20

21. Ghaffari, M., Kuhn, F., Maus, Y.: On the complexity of local distributed graph problems. In: 49th ACM Symposium on Theory of Computing (STOC), pp. 784–797 (2017). https://doi.org/10.1145/3055399.3055471
22. Godard, E., Perdereau, E.: k-set agreement in communication networks with omission faults. In: 20th International Conference on Principles of Distributed Systems (OPODIS), pp. 8:1–8:17 (2016). https://doi.org/10.4230/LIPIcs.OPODIS.2016.8
23. Göös, M., Hirvonen, J., Suomela, J.: Linear-in-Δ lowerbounds in the LOCAL model. Distrib. Comput. **30**(5), 325–338 (2017). https://doi.org/10.1007/s00446-015-0245-8
24. Harris, D.G., Schneider, J., Su, H.: Distributed $(\Delta+1)$-coloring in sublogarithmic rounds. In: 48th ACM Symposium on Theory of Computing (STOC), pp. 465–478 (2016). https://doi.org/10.1145/2897518.2897533
25. Herlihy, M., Kozlov, D., Rajsbaum, S.: Distributed Computing Through Combinatorial Topology. Morgan Kaufmann, San Francisco (2013)
26. Herlihy, M., Rajsbaum, S.: Set consensus using arbitrary objects. In: Proceedings of the Thirteenth Annual ACM Symposium on Principles of Distributed Computing (PODC), pp. 324–333 (1994). https://doi.org/10.1145/197917.198119
27. Herlihy, M., Rajsbaum, S.: Algebraic spans. Math. Struct. Comput. Sci. **10**(4), 549–573 (2000). http://journals.cambridge.org/action/displayAbstract?aid=54601
28. Herlihy, M., Rajsbaum, S., Tuttle, M.R.: An axiomatic approach to computing the connectivity of synchronous and asynchronous systems. Electr. Notes Theor. Comput. Sci. **230**, 79–102 (2009). https://doi.org/10.1016/j.entcs.2009.02.018
29. Herlihy, M., Shavit, N.: The asynchronous computability theorem for t-resilient tasks. In: 25th ACM Symposium on Theory of Computing (STOC), pp. 111–120 (1993). https://doi.org/10.1145/167088.167125
30. Herlihy, M., Shavit, N.: The topological structure of asynchronous computability. J. ACM **46**(6), 858–923 (1999). https://doi.org/10.1145/331524.331529
31. Kuhn, F., Lynch, N.A., Oshman, R.: Distributed computation in dynamic networks. In: 42nd ACM Symposium on Theory of Computing (STOC), pp. 513–522 (2010). https://doi.org/10.1145/1806689.1806760
32. Kuhn, F., Moscibroda, T., Wattenhofer, R.: Local computation: lower and upper bounds. J. ACM **63**(2), 17:1–17:44 (2016). https://doi.org/10.1145/2742012
33. Kuhn, F., Moses, Y., Oshman, R.: Coordinated consensus in dynamic networks. In: 30th ACM Symposium on Principles of Distributed Computing (PODC), pp. 1–10 (2011). https://doi.org/10.1145/1993806.1993808
34. Kuhn, F., Oshman, R.: Dynamic networks: models and algorithms. SIGACT News **42**(1), 82–96 (2011). https://doi.org/10.1145/1959045.1959064
35. Linial, N.: Locality in distributed graph algorithms. SIAM J. Comput. **21**(1), 193–201 (1992). https://doi.org/10.1137/0221015
36. Mendes, H., Tasson, C., Herlihy, M.: Distributed computability in Byzantine asynchronous systems. In: 46th Symposium on Theory of Computing (STOC), pp. 704–713 (2014). https://doi.org/10.1145/2591796.2591853
37. Peleg, D.: Distributed Computing: A Locality-Sensitive Approach. SIAM, Philadelphia (2000)
38. Rajsbaum, S., Castañeda, A., Flores-Peñaloza, D., Alcantara, M.: Fault-tolerant robot gathering problems on graphs with arbitrary appearing times. In: 2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS), pp. 493–502 (2017). https://doi.org/10.1109/IPDPS.2017.70
39. Rajsbaum, S., Raynal, M., Travers, C.: The iterated restricted immediate snapshot model. In: 14th International Conference on Computing and Combinatorics (COCOON), pp. 487–497 (2008). https://doi.org/10.1007/978-3-540-69733-6_48

40. Sakavalas, D., Tseng, L.: Network topology and fault-tolerant consensus. Synth. Lect. Distrib. Comput. Theory **9**, 1–151 (2019)
41. Saks, M.E., Zaharoglou, F.: Wait-free k-set agreement is impossible: the topology of public knowledge. In: 25th ACM Symposium on Theory of Computing (STOC), pp. 101–110 (1993). https://doi.org/10.1145/167088.167122
42. Suomela, J.: Survey of local algorithms. ACM Comput. Surv. **45**(2), 24:1–24:40 (2013). https://doi.org/10.1145/2431211.2431223