Theoretical Computer Science 948 (2023) 113733

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Wait-free approximate agreement on graphs [☆]

Dan Alistarh [a], Faith Ellen [b], Joel Rybicki [a,*]

[a] *IST Austria, Austria*
[b] *University of Toronto, Canada*

ARTICLE INFO

Article history:
Received 3 December 2021
Received in revised form 15 November 2022
Accepted 20 January 2023
Available online 25 January 2023

Keywords:
Approximate agreement
Wait-free algorithms
Bridged graphs

ABSTRACT

Approximate agreement is one of the few variants of consensus that can be solved in a wait-free manner in asynchronous systems where processes communicate by reading and writing to shared memory. In this work, we consider a natural generalisation of approximate agreement on arbitrary undirected connected graphs. Each process is given a node of the graph as input and, if non-faulty, must output a node such that

- all the outputs are within distance 1 of one another, and
- each output value lies on a shortest path between two input values.

From prior work, it is known that there is no wait-free algorithm among $n \geq 3$ processes for this problem on any cycle of length $c \geq 4$, by reduction from 2-set agreement (Castañeda et al., 2018).

In this work, we investigate the solvability of this task on general graphs. We give a new, direct proof of the impossibility of approximate agreement on cycles of length $c \geq 4$, via a generalisation of Sperner's Lemma to convex polygons. We also extend the reduction from 2-set agreement to a larger class of graphs, showing that approximate agreement on these graphs is unsolvable. On the positive side, we present a wait-free algorithm for a different class of graphs, which properly contains the class of chordal graphs.

1. Introduction

Understanding the solvability and complexity of coordination tasks is one of the key questions in distributed computing. The difficulty of coordination often arises from *uncertainty*: processes have limited knowledge about each other's inputs, the relative speed of computation and communication between processes can vary, and processes may fail during computation.

Tasks which require perfect agreement, such as *consensus* [42], are typically hard to solve: Fischer, Lynch, and Paterson [28] proved that consensus cannot be reached in asynchronous message-passing systems if even one process may crash. Later, this was extended to shared memory systems where processes communicate using shared registers [17,38].

While perfect agreement is not needed for many applications, it is known that agreeing on at most $k > 1$ different values is still hard: There exists no wait-free algorithm for k-set agreement in the asynchronous setting for $n > k$ processes, where processes communicate by reading and writing to shared registers [11,33,43]. In contrast, approximate agreement – agreeing on values that are sufficiently close to one another – can be considerably easier [9,22,26,27,39,44].

[☆] This article belongs to Section A: Algorithms, automata, complexity and games, Edited by Paul Spirakis.
[*] Corresponding author.
E-mail addresses: dan.alistarh@ist.ac.at (D. Alistarh), faith@cs.toronto.edu (F. Ellen), joel.rybicki@ist.ac.at (J. Rybicki).

https://doi.org/10.1016/j.tcs.2023.113733

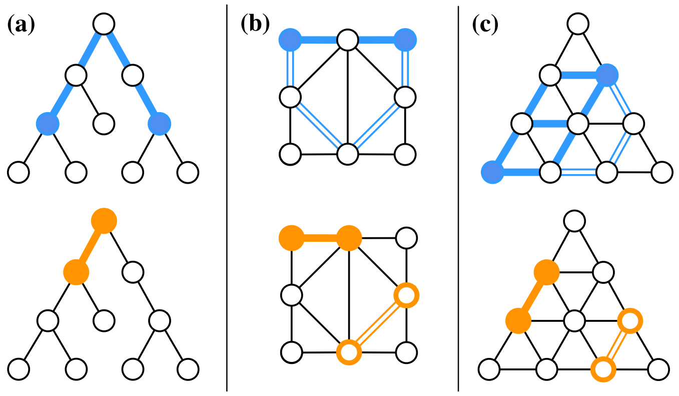

Fig. 1. Examples of approximate agreement with $n = 2$ processes. In the top row, blue nodes are input values for a particular instance. Solid blue lines denote the edges on *shortest* paths connecting input nodes. Blue double lines denote the additional edges that are on some *minimal* path connecting input nodes. Solid orange nodes denote some outputs that satisfy shortest path validity. Non-solid orange nodes denote some outputs that only satisfy minimal path validity. (a) Agreement on a tree. All minimal paths are also shortest paths. (b) Agreement on a chordal graph. (c) Agreement on a non-chordal bridged graph. (d)–(e) Instances of 6-cycle agreement. (For interpretation of the colours in the figure(s), the reader is referred to the web version of this article.)

1.1. Graphical approximate agreement

In this work, we study solvability and complexity of approximate agreement when the set of input and output values are nodes of a graph. Consider a distributed system with n processes and let $G = (V, E)$ be a connected undirected graph. The graph G is not assumed to be related to the communication topology of the system, but it is assumed to be known by all processes.

In *(graphical) approximate agreement on* G, each process p_i is given a node $x_i \in V$ as input and has to output a node $y_i \in V$ subject to the following constraints:

- (clique) agreement: every two output values are adjacent in G, and
- (shortest path) validity: each output value lies on a shortest path between two input values.

Note that the output values form a clique. Fig. 1(a) gives an example of graphical approximate on a tree.

Approximate agreement on a path The special case when G is a path has been well-studied. This case is typically considered in the continuous setting, where the values reside on the real line, $\mathbb{R}$, and the goal is to output values within distance $\varepsilon > 0$ of each other. However, a discrete version of the problem can be obtained by considering integer-valued inputs and outputs and taking $\varepsilon = 1$.

The approximate agreement problem was originally introduced by Dolev, Lynch, Pinter, Stark and Weihl [22]. They showed that, in asynchronous message-passing systems, approximate agreement can be solved with $f < n/5$ Byzantine faults. This result was later improved by Abraham, Amit, and Dolev [1] to allow $f < n/3$ Byzantine faults, matching a lower bound by Fischer, Lynch, and Merritt [29]. Fekete gave efficient algorithms for crash, omission and Byzantine fault models in the synchronous message-passing setting [26] and asynchronous message-passing algorithms that tolerate crash and omission faults [27].

In the shared-memory setting with crash faults, Attiya, Lynch, and Shavit [9] showed that the step complexity of wait-free solutions to approximate agreement on $\mathbb{R}$ using single-writer registers is $\Theta(\log n)$. Using multi-writer registers, Schenk [44] established that the step complexity of obtaining agreement is $O(\log(M/\varepsilon))$, where M is the maximum magnitude of any input value.

Approximate agreement under minimal path validity Nowak and Rybicki [41] studied approximate agreement on chordal graphs under a slightly different validity condition, where output values have to lie on a *minimal* path between any two input values. A path in G is *minimal* if no two non-consecutive nodes in the path are connected by an edge, i.e. if $v_0, \ldots, v_k$ is a minimal path and $0 \le i < j - 1 \le k - 1$, then $\{v_i, v_j\} \notin E$. This validity condition is weaker, since every shortest path between two nodes is a minimal path, but the converse is not true. Figs. 1(b)–(c) illustrate the difference between minimal and shortest paths. If G is chordal, then there exists an algorithm tolerating f *Byzantine* faults in the asynchronous message-passing model for $n > (\omega(G) + 1)f$ processes, where $\omega(G)$ is the size of the largest clique in G [41].

Approximate gathering on graphs Alcántara, Castañeda, Flores-Peñaloza, and Rajsbaum [5] investigated approximate agreement with a weaker validity condition: The *1-gathering* task requires that the output values must be a subset of the input values if the input values form a clique. This condition is called *clique validity*. They also considered a related problem, *edge gathering*, which has different agreement and validity conditions: *edge agreement* requires that all outputs belong to the

Table 1
Algorithms for asynchronous approximate agreement on graphs. SM denotes models in which processes communicate by reading and writing to shared memory. MP denotes the asynchronous message-passing model with Byzantine faults.

Graph class	Validity	Model	Faults	Resiliency	Reference
Clique graph is a tree	Clique validity	SM	crash	$f < n$	[5]
Radius one					[5]
Chordal	Minimal path	MP	Byzantine	$f < n/(\omega + 1)$	[41]
Paths	Shortest path	SM	crash	$f < n$	[9,44]
Paths		MP	Byzantine	$f < n/3$	[1]
Nicely bridged or radius one		SM	crash	$f < n$	**here**
Any		SM	crash	$f = 1$	**here**

same edge and *edge validity* requires that the output values must be a subset of the input values if all inputs belong to the same edge.

Alcántara, Castañeda, Flores-Peñaloza, and Rajsbaum show that their algorithms satisfy clique validity, but at least some of their algorithms can be shown to satisfy shortest path validity. However, in general, algorithms for 1-gathering or edge gathering do not solve graphical approximate agreement. The difference is best illustrated by the simple case of a path, where minimal path validity and shortest path validity require that the outputs always lie between the smallest and largest input values, while clique validity and edge validity do not have this requirement. Other examples are shown in Figs. 1 (b), (c), and (e), where clique validity and edge validity hold vacuously, since the inputs do not form a clique.

Approximate agreement on cycles When G is a cycle of length $c \geq 4$, approximate agreement under minimal path validity and approximate agreement under clique validity are the same problem. We refer to this special case as *c-cycle agreement*. When $c = 3$, the problem is trivial, since each process can output its input.

Castañeda, Rajsbaum, and Roy [15] showed that 2-set agreement reduces to the edge gathering problem on graphs with cycles of length at least three. On cycles of length $c \geq 4$, edge gathering is equivalent to c-cycle agreement, which implies that there is no 2-resilient asynchronous algorithm using shared registers for this problem. Hence, approximate agreement on cycles of length at least 4 is harder than on paths and chordal graphs.

1.2. Contributions

In this work, we establish additional positive and negative results on the solvability and complexity of graphical approximate agreement.

Positive results We present a wait-free algorithm for $n \geq 2$ processes that solves approximate agreement on a large subclass of bridged graphs, and on any radius one graph. A *bridged graph* is a graph in which each of its cycles of length at least 4 contains 2 nodes that are connected by a shorter path than either path in the cycle connecting them [24,25]. All chordal graphs are bridged, but the converse is not necessarily true; for an example, see Fig. 1(c).

Our algorithm solves the graphical approximate agreement problem on graphs of radius one and a large class of bridged graphs, which includes chordal graphs and graphs whose clique graphs are trees. Thus, our algorithm handles all graphs handled by previous algorithms, while guaranteeing shortest path validity. See Table 1 for a comparison.

In addition, we give a 1-resilient asynchronous algorithm for graphical approximate agreement using only registers on *any* connected graph for $n \geq 2$ processes. Note that, when $n = 2$, this algorithm is wait-free. For the fully-connected synchronous message-passing model, we also present an f-resilient synchronous algorithm for the fully-connected message-passing model with $n > f$ processes. The algorithm solves approximate agreement on any connected graph G in $\lfloor f/2 \rfloor + \lceil \log_2 \operatorname{diam}(G) \rceil + 1$ rounds, where $\operatorname{diam}(G)$ is the diameter of G.

Negative results In Section 4, we provide a new, direct proof of the impossibility of approximate agreement on cycles of length $c \geq 4$. It uses a generalisation of Sperner's lemma (on triangles) to convex polygons. Furthermore, in Section 5, we present a simplified version of the existing reduction [15] from 2-set edge gathering, use it to extend the impossibility of 1-gathering and graphical approximate agreement to a larger class of graphs. It then follows from standard reductions [13,31] that there is no 2-resilient asynchronous algorithm using registers and any f-resilient synchronous algorithm requires at least $\lfloor f/2 \rfloor + 1$ rounds for $n > f$ processes.

2. Related work

Multidimensional approximate agreement Mendes, Herlihy, Vaidya and Garg [40] generalised approximate agreement to the multidimensional setting, where the input values are points in m-dimensional Euclidean space $\mathbb{R}^m$, for $m > 0$. In the multidimensional approximate agreement problem, the output values should be within distance $\varepsilon > 0$ of one another and be contained in the convex hull of the input values of the non-faulty processes. When $m = 1$, this is approximate agreement on a line. Multidimensional approximate agreement on $\mathbb{R}^m$ is solvable with f Byzantine faults in the asynchronous completely-connected message-passing setting if and only if $n > (m+2)f$ holds [40]. In the synchronous setting, the problem is solvable if and only if $n > \max\{3f, (m+1)f\}$. Recently, Függer and Nowak [30] established asymptotically tight convergence rates for multidimensional approximate agreement by removing the dependence on the dimension m of the space.

Unlike approximate agreement on the real line, it is not straightforward to obtain a discrete version of multidimensional approximate agreement when $m \geq 2$. For example, in the two-dimensional integer lattice $\mathbb{Z}^2 \subseteq \mathbb{R}^2$, one can find a pair of points arbitrarily far apart such that they are the only integral points in their convex hull. For example, take the points $(0, 0)$ and $(1, k)$ for any $k > 0$: the only integral points in the convex hull of these points are the points themselves. In this case, solving approximate agreement is the same as solving consensus.

Approximate agreement in simplicial complexes In barycentric agreement [34], processes are given inputs that lie on a simplex σ of a simplicial complex and must output values that are on a simplex of the barycentric subdivision of σ. This problem can be solved using m-dimensional approximate agreement [39].

Simplex agreement [36] is a mild generalisation in which processes are given vertices in a simplicial complex as input, the set of outputs must lie on a simplex of this complex, and, if the set of inputs lies on a simplex of this complex, the outputs must be a subset of the inputs. Edge gathering on a graph is the same as simplex agreement on the graph when viewed as a simplicial complex and 1-gathering on a graph is the same as simplex agreement in the complex $\kappa(G)$ of cliques of the graph G. Ledent [36] conjectured this problem is solvable on G if and only if $\kappa(G)$ is contractible. Note that a graph is chordal if and only if the complex of cliques of every induced subgraph is contractible [2].

Loop agreement [32,34] is a variant of a special case of simplex agreement. In this problem, the simplicial complex has three distinguished nodes with a fixed simple path connecting each pair of these nodes. These paths intersect only at their endpoints. The input of each process is a distinguished node. Like simplex agreement, the outputs must lie on a simplex of the complex and, if all inputs are the same, all processes must output this node. However, if the set of inputs consists of two distinguished nodes, the set of outputs can lie on the endpoints of any one edge of the path connecting the two inputs.

Herlihy and Rajsbaum [32] proved that there is a wait-free algorithm for loop agreement using only registers if and only if the loop formed by the three paths is contractible in the simplicial complex. In particular, if the simplicial complex is a cycle of length $c \geq 3$, loop agreement has no wait-free solution using only registers. Although this task appears similar to c-cycle agreement, there are two important differences when $c \geq 4$. First, there are c choices for each input, rather than just 3 choices. Furthermore, the validity condition for c-cycle agreement is weaker. For example, if the set of inputs consists of two diametrically opposite nodes, then the set of outputs can lie on any one edge of the cycle.

Approximate robot gathering in graphs Robot gathering problems were first studied in continuous settings [4,19], but have also been studied in discrete settings, where n robots reside on nodes in a graph. The inputs represent the initial positions of the robots, the outputs represent the final positions of the robots, and the goal is that the outputs are close to one another.

Exact gathering of asynchronous robots, where the goal is to get all robots to the same node, has been studied extensively in various models; see a recent survey by Cicerone, Di Stefano, and Navarra [18]. Castañeda, Rajsbaum, and Roy [15] and Alcántara, Castañeda, Flores-Peñaloza, and Rajsbaum [5] studied several variants of approximate gathering of asynchronous robots on a graph, where robots communicate via atomic snapshot objects.

There is a wait-free algorithm for edge gathering on graph G if and only if G is a tree [5]. On cliques, edge gathering is the same as the 2-set agreement task, whereas 1-gathering and graphical approximate agreement are trivial. Alcántara et al. [5] also showed that 1-gathering is solvable in a wait-free manner on graphs whose clique graphs are trees and on graphs of radius one (i.e., graphs with a dominating set of size one). The clique graph $K(G)$ of G is the graph whose nodes are the maximal cliques of G and whose edges are the pairs of maximal cliques that share one or more nodes. Note that there are chordal graphs whose clique graphs are not trees. For example, see Fig. 1(b).

Castañeda et al. [15] and Alcántara et al. [5] gave reductions showing that edge gathering in a graph that contains a cycle is at least as hard as 2-set agreement for $n = 3$ processes. In triangle-free graphs, edge gathering and 1-gathering are exactly the same problem. Thus, their result implies that on triangle-free graphs with a cycle, 1-gathering is as hard as 2-set agreement for $n = 3$ processes. When the graph is a cycle of length $c \geq 4$, both edge gathering and 1-gathering are the same as approximate agreement with minimal path validity, i.e., c-cycle agreement.

In Section 4, we give a direct topological proof of the impossibility of c-cycle agreement. Then in Section 5, we simplify and adapt the reduction from 2-set agreement to prove that wait-free 1-gathering, and hence also graphical approximate agreement under shortest path validity, is impossible on a larger class of graphs. In particular, this includes graphs that can contain triangles.

3. Models

We consider distributed systems with n processes, where some processes may fail by crashing. In particular, we focus on the setting where processes communicate using atomic snapshot objects. We also consider a synchronous message-passing model in which every process can directly communicate with every other process.

3.1. Asynchronous shared memory models

A *single-writer snapshot* object has n components, each of which has initial value $-$. It supports two atomic operations, update and scan. An update(x) by process p_i changes the value of component i to $x \neq -$. A scan returns the value of each component. This object can be implemented from registers, which support only read and write [3].

In the *iterated snapshot model*, n processes, $p_0, \ldots, p_{n-1}$, communicate using a finite or infinite sequence of shared single-write snapshot objects. We use S_i to denote the i'th snapshot object in this sequence. Each process performs an update on a snapshot object, starting with S_1. Next, it repeatedly performs scan on this object until there are at least a certain number of components with values other than $-$. Then it updates its state and determines whether to output a value and terminate or to continue to the next snapshot object in the sequence.

An *execution* of an algorithm consists of a sequence of steps in which processes perform operations on snapshot objects or output values and terminate, as specified by the algorithm. In particular, once a process has terminated, it no longer performs any steps. Note that it suffices to consider executions where all accesses to each snapshot object occur before any accesses to the next snapshot object in the sequence. This is because, if one process performs an operation on S_i immediately after another process performs an operation on S_j, where $j > i$, then interchanging the order of these two operations does not change the resulting states of the two processes.

A *configuration* consists of the contents of each snapshot object and the state of each process. In an *initial configuration*, each process is in an initial state, which includes its identifier and its input, and all components of all snapshot objects contain the initial value $-$. A configuration is *terminal* if every process is in a terminated state. A configuration C is *reachable* if there is an execution from an initial configuration that results in configuration C. An algorithm is *wait-free* if it has no infinite execution from any initial configuration.

A process *crashed* in an execution if the process takes a finite number of steps in the execution, but does not terminate. An algorithm is *f-resilient* if it has no infinite execution (from an initial configuration) in which at most f processes crashed. In particular, an algorithm is wait-free if and only if it is $(n-1)$-resilient.

The *iterated immediate snapshot model*, introduced by Hoest and Shavit [35], is a restricted version of the iterated snapshot model. The sequence of steps in an execution can be divided into blocks in which a set of processes each performs an update to the same snapshot object and then each of them performs one scan of this snapshot object. Moreover, whenever a process performs an update, it writes its entire state to its component and, after performing a scan, its new state consists of its identifier and the result of the scan. Note that a process remembers its entire history, because its previous state is stored in the snapshot object. Thus, the contents of every snapshot object in a configuration can be determined from the states of all processes in that configuration.

Each initial configuration in the iterated immediate snapshot model corresponds to a *simplex* containing one vertex for each process, which specifies its input. The collection of all such simplexes is called the *input complex*. Likewise, for any algorithm, each reachable terminal configuration corresponds to a simplex containing one vertex for each process, which specifies its state, including the value it outputs. The collection of all such simplexes (or n-node cliques) is called the *protocol complex*. There is an edge between two vertices in the protocol complex if they represent the states of different processes and there is a reachable terminal configuration containing both these states.

Hoest and Shavit proved that the protocol complex of any wait-free algorithm can be obtained from the input complex by performing a finite number of *non-uniform chromatic subdivisions* of simplexes. In the special case when there are $n = 3$ processes, each simplex is a triangle and the non-uniform chromatic subdivision of a simplex is a triangulation of the simplex.

The asynchronous shared memory model in which processes communicate using shared registers (that support only read and write) is known to be equivalent to both the iterated snapshot model and the iterated immediate snapshot model: Any problem that has a deterministic, wait-free solution in one of these models has deterministic, wait-free solutions in the other two models For deterministic, wait-free computation, both the iterated snapshot model and the iterated immediate snapshot model are equivalent to the asynchronous shared memory model in which processes communicate using shared registers (which support only read and write) [3,10,12].

3.2. The synchronous message-passing model

In the synchronous message-passing model, there is no uncertainty regarding the relative speeds of processes. An *execution* is divided into synchronous rounds. In each round, each process can send a (possibly different) message to every other process. If a process crashes in round r, it sends a message to an arbitrary subset of the other processes in round r, but does not send any messages in subsequent rounds. During round r, a process that has not crashed (in round r or earlier) receives all the messages that were sent to it in round r. At the end of round r, a process that has not crashed does local computation to determine whether to output a value and terminate or continue to the next round. A process that has terminated does not send or receive messages in subsequent rounds.

A *configuration* consists of the state of each process at the beginning of a round. An *initial configuration* is a configuration at the beginning of the first round, so each process is in an initial state (which includes its identifier and its input). A configuration is *terminal* if every process is either crashed or in a terminated state. A synchronous algorithm is f-resilient if it has no infinite execution (from an initial configuration) in which at most f processes crashed. An f-resilient synchronous

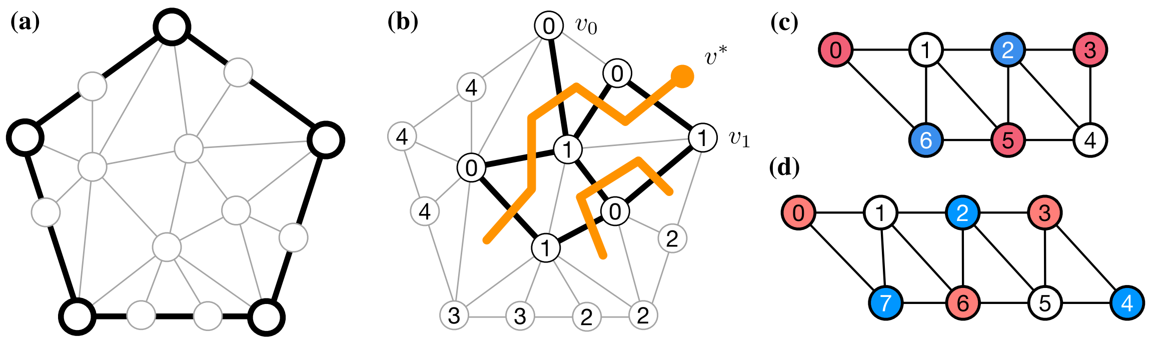

Fig. 2. (a) A triangulation T of a pentagon. (b) A Sperner labelling of T with the edges of the graph G', as defined in Lemma 1, superimposed in orange. (c) The subcomplex $\mathbb{H}$ for $c = 7$. (d) The subcomplex $\mathbb{H}$ for $c = 8$.

algorithm has round complexity T if every process either crashes or outputs a value and terminates by the end of round T in every execution (from an initial configuration) in which at most f processes crashed.

4. Impossibility of asynchronous wait-free cycle agreement

In this section, we give a new direct topological proof of the impossibility of wait-free c-cycle agreement among $n \geq 3$ processes in the iterated immediate snapshot model. Our proof relies on a slight generalisation of Sperner's lemma from triangles to convex polygons, originally shown by Atanassov [8] and generalised to convex polytopes of any dimension by de Loera, Peterson, and Su [20]. However, for us, a special case in the two-dimensional setting suffices, because the key argument in the lower bound is for three processes. We note that the proof closely follows the classic proof of Sperner's lemma for triangles; see e.g. [14, Section 1.9].

Let H be a polygon with c vertices and let T be a finite triangulation of H. A *Sperner labelling* of T is a function from the vertices of T to the set $\{0, \dots, c-1\}$ such that each vertex of H gets a different label and each vertex in T that lies on an edge of H gets the same label as one of the endpoints of this edge. See Figs. 2(a) and 2(b).

Lemma 1. *Let H be a convex polygon with c vertices. Any Sperner labelling of a triangulation of H has a triangle whose vertices have three different labels.*

Proof. Let T be a triangulation of H and consider any Sperner labelling of T. Without loss of generality, suppose there are two adjacent vertices v_0 and v_1 of H labelled with 0 and 1, respectively. Consider the graph $G' = (V' \cup \{v^*\}, E')$, where V' is the set of triangles of T. There is an edge in E' between triangles τ and τ' if and only if they have exactly two vertices in common, one of which is labelled 0 and the other of which is labelled 1. There is an edge in E' between v^* and triangle τ if and only if two of the vertices of τ have different labels and both lie on the edge of H between v_0 and v_1. This is illustrated in Fig. 2(b).

Each of the vertices of T that lies on the edge of H between v_0 and v_1 is labelled by 0 or 1. Since v_0 and v_1 have different labels, there are an odd number of edges of T whose endpoints have different labels and which lie on the edge of H between v_0 and v_1. Thus, v^* has odd degree. If a triangle has two vertices labelled 0 and one node labelled 1 or vice versa, it has degree 2 in G'. If a triangle has one vertex labelled 0, one vertex labelled 1, and one vertex with some other label, it has degree 1 in G'. Otherwise, it has degree 0 in G'.

The handshaking lemma [23] says that any finite graph contains an even number of nodes with odd degree. Since v^* has odd degree, there exists a triangle $\tau \in V'$ with odd degree. The vertices of this triangle have three different labels. □

Theorem 2. *For $c \geq 4$, there is no wait-free algorithm for the c-cycle agreement problem among $n \geq 3$ processes in the iterated immediate snapshot model.*

Proof. Consider a complete tripartite graph, where part $i \in \{0, 1, 2\}$ has c vertices, corresponding to the initial states in which process p_i has input $x \in \{0, \dots, c-1\}$. Each triangle corresponds to an initial configuration for c-cycle agreement among 3 processes, p_0, p_1, and p_2. In this case, the input complex consists of a simplex for each triangle in this graph.

Let $\mathbb{H}$ denote the part of this input complex consisting of the simplexes corresponding to the following $c-2$ input configurations:

- p_0 has input $3a$, p_1 has input $3a+1$, and p_2 has input $c-3a-1$, for $0 \leq a \leq \lfloor (c-3)/6 \rfloor$.
- p_1 has input $3a+1$, p_2 has input $c-3a-1$, and p_0 has input $c-3a-2$, for $0 \leq a \leq \lfloor (c-4)/6 \rfloor$.
- p_1 has input $3a+1$, p_2 has input $3a+2$, and p_0 has input $c-3a-2$, for $0 \leq a \leq \lfloor (c-5)/6 \rfloor$.
- p_2 has input $3a+2$, p_0 has input $c-3a-2$, and p_1 has input $c-3a-3$, for $0 \leq a \leq \lfloor (c-6)/6 \rfloor$.

– p_2 has input $3a+2$, p_0 has input $3a+3$, and p_1 has input $c-3a-3$, for $0 \le a \le \lfloor (c-7)/6 \rfloor$.
– p_0 has input $3a+3$, p_1 has input $c-3a-3$, and p_2 has input $c-3a-4$, for $0 \le a \le \lfloor (c-8)/6 \rfloor$.

Note that $\mathbb{H}$ contains c vertices, one for each possible input value. The cases $c=7$ and $c=8$ are illustrated in Figs. 2(c) and 2(d). The processes p_0, p_1, and p_2 are denoted by the colours red, white, and blue, respectively. The *border* of $\mathbb{H}$ is a polygon H with all c vertices and with edges between vertices whose input values differ by 1 modulo c.

Consider any wait-free algorithm for c-cycle agreement among 3 processes in the iterated immediate snapshot model. Let $\mathbb{S}$ denote its protocol complex. It is finite, since the input complex is finite and the algorithm is wait-free. Let $\mathbb{T}$ denote the subcomplex of $\mathbb{S}$ consisting of all terminal configurations reachable from configurations in $\mathbb{H}$, where each vertex is labelled with the output value it contains. The vertices and edges of $\mathbb{T}$ form a triangulation T of H. Label each vertex of T with the label of the corresponding vertex in $\mathbb{T}$.

The vertex v_x in $\mathbb{H}$ with input value $x \in \{0, \ldots, c-1\}$ is a vertex in $\mathbb{T}$ that corresponds to the solo execution by some process with input x. If v_x is not labelled by the value x, then the algorithm does not solve c-cycle agreement, because it violates validity. Consider any edge of $\mathbb{T}$ whose endpoints correspond to vertices in T that lie on the edge of H between v_x and $v_{x'}$, where $x' = (x+1) \bmod c$. This edge corresponds to an execution by only two processes, one with input x and the other with input x'. If an endpoint of this edge is not labelled by x or x', the algorithm does not solve c-cycle agreement, because it violates validity.

If the algorithm is correct, then the labelling of the triangulation T is a Sperner labelling. By Lemma 1, T contains a triangle whose vertices have three different labels. The corresponding configuration in $\mathbb{T}$ is the result of an execution in which the three processes output different values, so the algorithm cannot be solving c-cycle agreement among three processes.

Since all but three processes can crash before taking any steps, any algorithm that solves c-cycle agreement among $n \ge 3$ processes is also an algorithm that solves c-cycle agreement among 3 processes. Therefore no such algorithm exists. □

5. Impossibility results via reductions

In this section, we show that wait-free approximate agreement on $n \ge 3$ processes is impossible on graphs that admit a certain labelling of the nodes. This is by a reduction from 2-set agreement. Using standard techniques, this implies the impossibility of 2-resilient graphical approximate agreement in the asynchronous shared memory model (where processes communicate by reading from and writing to registers) and a lower bound on the round complexity of the problem in the synchronous message-passing model.

We note that while the reduction is similar to the proof of the impossibility of edge gathering by Castañeda et al. [15], our result also applies to the 1-gathering and graphical approximate agreement problems, and applies to a different class of graphs.

5.1. Graphs on which wait-free approximate agreement is impossible

We now show that approximate agreement is hard on graphs that admit a certain labelling of its nodes. We do so by a reduction from 2-set agreement among $n \ge 3$ processes. In this problem, each process has an input value in $\{0, 1, 2\}$ and, if it does not crash, it must output one of the inputs such that at most two different values are output.

A labelling $\ell\colon V \to \{0, 1, 2\}$ of the nodes of a graph $G = (V, E)$ is an *impossibility labelling* if the following conditions hold:

(1) G contains no triangle with three different labels, and
(2) G contains a cycle C in which exactly one node has label 1 and its two neighbours in C have labels 0 and 2.

It is easy to check that any cycle graph of length $c \ge 4$ admits an impossibility labelling: pick three consecutive nodes, label them with 0,1,2, and label all other nodes with 2. A wheel graph, which consists of a cycle and one central node that is a neighbour of all nodes in the cycle, does not admit an impossibility labelling. On the other hand, if one edge adjacent to the central node is removed, the resulting graph does admit an impossibility labelling: label the other endpoint of the removed edge with 1, label one of its neighbours with 0, and label all other nodes with 2.

Theorem 3. *Suppose G is a graph that admits an impossibility labelling. Then there is no wait-free algorithm among $n \ge 3$ processes that solves 1-gathering or graphical approximate agreement on G.*

Proof. Consider an impossibility labelling ℓ of G. Let C be a cycle in G that contains exactly one node, v_1, with label 1, a neighbour v_0 of v_1 with label 0, and a neighbour v_2 of v_1 with label 2. As mentioned in Section 1.1, there is a wait-free approximate agreement algorithm on any path. Let A be a wait-free approximate agreement algorithm on the path $C \setminus \{v_1\}$.

To obtain a contradiction, suppose there is a wait-free algorithm B for 1-gathering or graphical approximate agreement on G. The following wait-free algorithm solves 2-set agreement:

- Processes with input value $x \in \{0, 2\}$ run the approximate agreement algorithm A on the path $C \setminus \{v_1\}$ using v_x as input. The node each of these processes outputs in A is used as its input for algorithm B.
- Processes with input value 1 use v_1 as their input for algorithm B.
- Each process p_i outputs the label $\ell(y_i)$ of the node y_i it outputs in B.

By the clique agreement property of graphical approximate agreement, the values output in B lie on a clique. The first property of an impossibility labelling implies that the nodes in this clique have at most two distinct labels. Thus, at most two different values are output by the processes.

If there are three distinct input values, then validity is immediately satisfied. If all input values are the same, then all output values are this input value, since this is true for algorithms A and B. It remains to consider instances of set agreement with exactly two input values. First, suppose the inputs for set agreement are in the set $\{0, 1\}$. All processes with input 0 output v_0 in algorithm A, since v_0 is the only value input to A. Thus, each process uses either v_0 or v_1 as its input to algorithm B. As v_0 and v_1 are adjacent in G, each process outputs one of these two values in B, by clique validity and, hence, shortest path validity. Hence, each process outputs a value in $\{\ell(v_0), \ell(v_1)\} = \{0, 1\}$ for set agreement, satisfying validity. The case $\{1, 2\}$ is symmetric.

Now, suppose that the inputs for set agreement are in the set $\{0, 2\}$. Then each process uses either v_0 or v_2 as its input to algorithm A. Their outputs in A, and hence their inputs to algorithm B, all lie on some edge $\{u, v\}$ on the path $C \setminus \{v_1\}$. By both clique validity (of 1-gathering problem) and shortest path validity (of graphical approximate agreement), each process outputs either u or v in B. From the second property of an impossibility labelling, we get that all values in $C \setminus \{v_1\}$ are labelled with either 0 or 2 because the label 1 is used exactly once in C. Thus, each process outputs 0 or 2 for set agreement, satisfying validity. $\square$

5.2. There exists no 2-resilient asynchronous algorithm

A task is *colourless* if the input of any process may be the input of any other process, the output of any process may be the output of any other process, and the specifications of valid outputs only depend on the set of inputs of the processes [13,32,34]. Similarly to approximate agreement on a path, also graphical approximate agreement is an example of a colourless task.

The BG simulation technique [13] shows that the impossibility of wait-free algorithms for a colourless task for $n \geq 3$ processes implies the impossibility of 2-resilient algorithms for that task.

Theorem 4. *If there exists a k-resilient asynchronous algorithm for $n > k$ processes that solves a colourless task, then there is a wait-free asynchronous algorithm for $k + 1$ processes that solves the task.*

Together with Theorem 2, the BG simulation immediately implies that there is no 2-resilient asynchronous algorithm for graphical approximate agreement on graphs that admit an impossibility labelling. This also implies a lower bound for the cycle agreement problem.

Corollary 5. *Suppose G is a graph that admits an impossibility labelling. Then for any $n \geq 3$, there is no 2-resilient asynchronous algorithm that solves graphical approximate agreement on G.*

5.3. Time lower bounds for synchronous algorithms

We can now lift the impossibility results to time lower bounds for the synchronous model using the round-by-round simulation by Gafni [31, Theorem 4.3], who showed the following.

Theorem 6. [31] *Let $0 < k < f < n$ such that $n - k - f > 0$. Fix $T \leq f/k$. Suppose there exists a synchronous f-resilient algorithm for n nodes that solves a colourless task in T rounds. Then there exists a k-resilient asynchronous algorithm that solves the task.*

Applying Corollary 5 and Theorem 6, we obtain a time lower bound for synchronous algorithms.

Corollary 7. *Suppose G is a graph that admits an impossibility labelling. For any $n > f \geq 3$, any f-resilient synchronous message-passing algorithm for graphical approximate agreement on G requires at least $\lfloor f/2 \rfloor + 1$ rounds.*

6. Upper bounds for asynchronous systems

In this section, we provide upper bounds for graphical approximate agreement. In the iterated snapshot model, we give

- a 1-resilient algorithm on general graphs for $n \geq 2$ processes (Section 6.1), and
- a wait-free algorithm on any nicely bridged graph for $n \geq 2$ processes (Section 6.2).

Preliminaries Let $G = (V, E)$ be a connected graph. For any set $U \subseteq V$, the subgraph of G induced by U is the graph $G[U] = (U, F)$, where $F = \{e \in E : e \subseteq U\}$. The distance between two nodes u and v in G is denoted by $d(u, v)$. The *eccentricity* $\epsilon(v)$ of a node $v \in V$ is $\max\{d(u, v) : u \in V\}$. The *diameter* of G is $\operatorname{diam}(G) = \max\{\epsilon(v) : v \in V\}$ and the *radius* $\operatorname{rad}(G)$ of G is $\min\{\epsilon(v) : v \in V\}$. For any nonempty set $U \subseteq V$, let $D(U) = \max\{d(u, v) : u, v \in U\}$. In particular, $\operatorname{diam}(G) = D(V)$.

6.1. A 1-resilient algorithm for general graphs

Let $G = (V, E)$ be an arbitrary connected graph, for example, a c-cycle for some $c \geq 4$. We show that we can solve the approximate agreement problem on G assuming at most one process crashes. The intuitive idea of the algorithm is simple: First use 2-set agreement to reduce the number of input values to at most 2 and then run approximate agreement on a path for $T = \lceil \log_2 \operatorname{diam}(G) \rceil$ steps.

There is an easy 1-resilient algorithm for 2-set agreement. However, the second step is not immediate, as there may be many paths of G on which the approximate agreement algorithm could be run. However, since all processes know the graph G, we can avoid this difficulty by fixing in advance a shortest path between every pair of nodes. We note that the second step of the algorithm is similar to that of the 2-process algorithm by Castañeda et al. [15].

The rest of this section is dedicated to proving the following result.

Theorem 8. *Let $G = (V, E)$ be a connected graph. Then for all $n \geq 2$, there exists a 1-resilient algorithm which solves graphical approximate agreement on G in the iterated snapshot model.*

Solving 2-set agreement Fix a total order on V. For any nonempty subset $X \subseteq V$, let $\min(X)$ be the smallest element of X under this order. Let $x_i(0) \in V$ be the input of process p_i. We will use a single-writer atomic snapshot object, S_0, whose components are initialised with the special value $-$. Each process p_i:

- performs update on the ith component of the snapshot object S_0, setting it to the value $x_i(0)$,
- repeatedly performs scan on the snapshot object S_0 until at least $n-1$ components have values other than $-$,
- lets $X_i(0)$ be the *set* of nodes returned by its last scan, and
- lets $x_i(1) = \min(X_i(0))$.

Approximate agreement on a path For any two nodes $u, v \in V$, fix a shortest path between u and v in G and let $g(u, v)$ be a fixed node in the middle of this path. Then $d(u, g(u, v)), d(v, g(u, v)) \leq \lceil d(u, v)/2 \rceil$. For any nonempty set $X \subseteq V$ of size at most two, define

$$\psi(X) = \begin{cases} u & \text{if } X = \{u\} \\ g(u, v) & \text{if } X = \{u, v\}. \end{cases}$$

We will use a sequence $S_1, \ldots, S_T$ of single-writer atomic snapshot objects, whose components are initialised with the special value $-$. For $t = 1, \ldots, T$, each process p_i:

- performs update on component i of the snapshot object S_t, setting it to the node $x_i(t)$,
- repeatedly performs scan on the snapshot object S_t until at least $n-1$ components have values other than $-$,
- lets $X_i(t)$ be the *set* of nodes returned by its last scan, and
- lets $x_i(t+1) = \psi(X_i(t))$.

The output of process p_i is the node $x_i(T+1)$.

Correctness Let $0 \leq t \leq T$. If process p_i crashes before computing $X_i(t)$, we define $X_i(t)$ to be the empty set. Observe that each process p_i first performs update on S_t with $x_i(t)$ before performing scan on S_t. Thus, if p_i computes $X_i(t)$, then $X_i(t)$ is nonempty.

Each component of S_t is updated at most once. Since scan is an atomic operation, the set of nodes returned in a scan is a subset of the set of nodes returned in any later scan. Therefore, $X_i(t) \subseteq X_j(t)$ or $X_j(t) \subseteq X_i(t)$ for any i and j. Each process continues performing scan until it crashes or S_t contains at most one $-$. Thus $\{X_j(t) : 0 \leq j \leq n-1\}$ contains at most two nonempty sets. Since $x_j(t+1)$ is a function of $X_j(t)$, it follows that $\{x_j(t+1) : X_j(t) \neq \emptyset\}$ contains at most two different nodes. These are the only values that are used to update components of S_{t+1}, so $X_i(t+1) \subseteq \{x_j(t+1) : X_j(t) \neq \emptyset\}$. Hence, $|X_i(t+1)| \leq 2$ and, if $X_i(t+1) \neq \emptyset$, then $x_i(t+2) = \psi(X_i(t+1))$ is defined.

Let $X(t) = \bigcup\{X_i(t) : 0 \leq i < n\}$. We use $X(T+1)$ to denote the set of output values. Note that $X(t) \subseteq V$ for $0 \leq t \leq T+1$ and $X(0)$ is a subset of the input values. If $t \geq 1$, then $X(t) \subseteq \{x_j(t) : X_j(t-1) \neq \emptyset\}$, so $|X(t)| \leq 2$.

Lemma 9. *Let $1 \leq t \leq T$. Then $D(X(t+1)) \leq \lceil D(X(t))/2 \rceil$.*

Proof. If $X_i(t) = X(t)$ for every nonempty set $X_i(t)$, then $x_i(t+1) = \psi(X(t))$. Hence $X(t+1)$ will contain only one node and $D(X(t+1)) = 0$. Otherwise, $X_i(t)$ is a nonempty, proper subset of $X(t)$ for some $0 \le i < n$. Recall that $|X(t)| \le 2$, so $X_i(t) = \{u\}$ and $X(t) = \{u, v\}$ for some nodes $u \ne v$. Since $X_j(t) \subseteq X_i(t)$ or $X_i(t) \subseteq X_j(t)$ for all $0 \le j < n$, it follows that every nonempty set $X_j(t)$ is either equal to $\{u\}$ or $\{u, v\}$ and $x_j(t+1)$ is either equal to $\psi(\{u\}) = u$ or $\psi(\{u, v\}) = g(u, v)$. By definition of g, we have that $d(u, g(u, v)) \le \lceil d(u, v)/2 \rceil$. Since $X(t+1) \subseteq \{u, g(u, v)\}$, it follows that $D(X(t+1)) \le \lceil D(X(t))/2 \rceil$. □

Proof of Theorem 8. We verify that the agreement and validity properties of graphical approximate agreement are satisfied. We proceed by induction to show that nodes in $X(t+1)$ lie on some shortest path between the nodes in $X(t)$ for all $1 \le t \le T$. The case $t = 1$ is true because $X(1) \subseteq X(0)$. Suppose the claim holds for some $X(t)$ such that $1 \le t \le T$. By definition of g and ψ, all nodes in $X(t+1)$ lie on some shortest path between the nodes in $X(t)$. Thus, validity is satisfied. Since $X(1) \subseteq V$, $D(X(1)) \le \mathrm{diam}(G)$. As $T = \lceil \log_2 \mathrm{diam}(G) \rceil$, Lemma 9 implies that the distance $d(u, v)$ between any two output nodes $u, v \in X(T+1)$ is at most

$$\max\{d(u, v) : u, v \in X(T+1)\} = D(X(T+1)) \le \lceil \mathrm{diam}(G)/2^T \rceil \le 1. \quad \square$$

In Section 7, we extend the same algorithmic idea to the synchronous message-passing setting under crash faults.

6.2. A wait-free asynchronous algorithm for nicely bridged graphs

Preliminaries The *center* of G is the set $\{v \in V : \epsilon(v) = \mathrm{rad}(G)\}$ of nodes with minimum eccentricity in G. A graph G is *k-self-centered* if every node has eccentricity k. This means that every node is in the center of G and $\mathrm{diam}(G) = \mathrm{rad}(G) = k$. A graph is *chordal* if it does not contain any induced cycles of length greater than three. The *3-sun*, also known as the Hajós graph, is obtained from a triangle $\{u, v, w\}$ by subdividing each of its edges and connecting the resulting three nodes $\{x, y, z\}$ to be a clique. This graph is 2-self-centered and chordal.

A set $K \subseteq V$ of nodes is (shortest path) *convex* if, for any $u, v \in K$, all nodes on all shortest paths between u and v are contained in K. For any $U \subseteq V$, the *convex hull* $\langle U \rangle$ of U is the smallest convex superset of U. If $A \subseteq B$, then $\langle A \rangle \subseteq \langle B \rangle$. A node v is *simplicial* in the graph G if the neighbours of v in G form a clique.

Bridged and nicely bridged graphs A subgraph H of G is *isometric* if the distances between any two nodes of H are the same in H and G. A graph is *bridged* if it contains no isometric cycles of length greater than three [24]. All chordal graphs are bridged, but a bridged graph may contain induced cycles of length greater than five.

We say that $G = (V, E)$ is *nicely bridged* if any 2-self-centered subgraph $H = G[S]$ induced by a convex set $S \subseteq V$ is chordal. In Appendix A we give examples of nicely bridged graphs. Chordal graphs, 3-sun-free bridged graphs, and bridged graphs with no four cliques are examples of nicely bridged graphs.

We now list some useful properties of bridged graphs. Farber gave the following result about the radius and diameter of bridged graphs [24].

Lemma 10. *For any bridged graph G, we have* $3 \cdot \mathrm{rad}(G) \le 2 \cdot \mathrm{diam}(G) + 2$. *If G is bridged and does not contain a 3-sun as an induced subgraph, then* $2 \cdot \mathrm{rad}(G) \le \mathrm{diam}(G) + 1$ *holds.*

We use the following fact due to Farber and Jamison [25, Theorem 6.5].

Lemma 11. *If* $G = (V, E)$ *is bridged, then* $D(\langle U \rangle) = D(U)$ *for any nonempty* $U \subseteq V$.

Next, we prove the following simple lemma.

Lemma 12. *If* $G = (V, E)$ *is bridged and* $H = G[\langle U \rangle]$ *for* $U \subseteq V$, *then* $\mathrm{diam}(H) = D(\langle U \rangle)$.

Proof. Let $u, v \in \langle U \rangle$. Consider any shortest path between u and v in G. By definition of $\langle U \rangle$, all nodes on this path are in $\langle U \rangle$. Thus, this is also a path between u and v in H. Since H is an induced subgraph of G, any shortest path between u and v in H is also a path between u and v in G. Hence the distance between u and v in G is the same as the distance between u and v in H. It follows that $D(\langle U \rangle) = \mathrm{diam}(H)$. □

Note that an induced subgraph of a bridged graph is not necessarily a bridged graph. For example, consider wheel graphs. However, the subgraph of a bridged graph induced by a convex set is bridged.

Lemma 13. *Let* $G = (V, E)$ *be a bridged graph and* $S \subseteq V$. *Then the subgraph* $G[\langle S \rangle]$ *is bridged.*

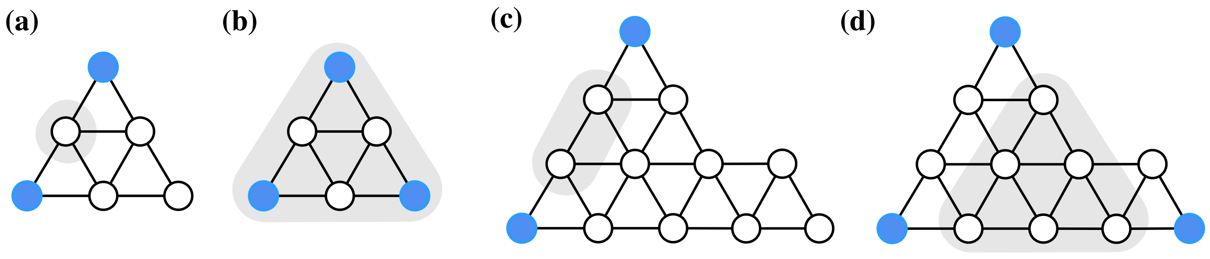

Fig. 3. Examples of the update rule of the algorithm. Blue nodes represent values in the set X. The grey area represents the center of the subgraph induced by $\langle X\rangle$. (a) The update rule outputs the only node in the center. (b) The update rule selects one of the white nodes in the center, as these are non-simplicial in the graph induced by the convex hull of the blue vertices. (c) The update rule selects one of the white nodes in the center. (d) All of the white nodes in the center are non-simplicial, so the update rule can select any one of these nodes.

Proof. Let $H = G[\langle S\rangle]$. Consider a cycle C of length at least four in H. Since C is a cycle in G and G is bridged, there exist nodes u and v in C such that the distance between them in G is less than the distance between them in C. Consider a shortest path between u and v in G. The shortest path convex hull $\langle S\rangle$ contains this path, since u and v are nodes of $\langle S\rangle$. Thus $H = G[\langle S\rangle]$ also contains this path. Hence C is not isometric. □

The algorithm For any nonempty set of nodes $X \subseteq V$, we choose a node $\psi(X)$ from the subgraph H induced by $\langle X\rangle$ as follows:

- If the center of H has a node that is non-simplicial in H, then let $\psi(X)$ be any such node.
- Otherwise, let $\psi(X)$ be any node in the center of H.

By definition, $\psi(X)$ has minimum eccentricity in the subgraph of G induced by $\langle X\rangle$. Since $\psi(X)$ is a node in the convex hull of X, it is on some shortest path between two nodes in X. Fig. 3 illustrates how the node $\psi(X)$ is chosen. Note that Fig. 3(a) and Fig. 3(b) give the views of two different processes on the same graph, as do Fig. 3(c) and Fig. 3(d).

Let $x_i(0)$ be the input of process p_i and let $T^* = \lceil \log_{3/2} \operatorname{diam}(G)\rceil + 1$. The processes communicate using a sequence $S_0, \ldots, S_T$ of single-writer snapshot objects, where $T = \max\{|V|, T^*\}$. In each iteration $t = 0, \ldots, T$, each process p_i:

- performs update on the ith component of the snapshot object S_t, setting it to the node $x_i(t)$,
- performs scan on the snapshot object S_t,
- defines $X_i(t)$ be the *set* of nodes returned by its scan, and
- sets $x_i(t+1) = \psi(X_i(t))$.

Once p_i has computed $x_i(T+1)$, the process outputs this node and terminates.

Correctness As before, if p_i crashes before computing the set $X_i(t)$, we define $X_i(t)$ to be the empty set. Let $X(t) = \bigcup\{X_i(t) : 0 \le i < n\}$. Note that $X(t) \subseteq V$ for $0 \le t \le T+1$ and $X(0)$ is a subset of the input nodes. In particular, $X(t)$ is the set of nodes returned by the last scan performed on the snapshot object S_t. Observe that if $y \in X(t)$, then $y = x_i(t)$ for some $0 \le i < n$. This is because each process p_j that performs update on S_t does so only with node $x_j(t)$. Thus, component j of S_t is either $x_j(t)$ or the special initial value $-$.

We use $X(T+1)$ to denote the set of output nodes. We show that $X(T+1)$ satisfies agreement (all the nodes are contained in a clique) and validity (all the nodes are in the shortest path convex hull of the inputs) of approximate agreement on G. We start with validity.

Lemma 14 *(Validity). Let $0 \le t \le T+1$. Then $X(t) \subseteq \langle X(0)\rangle$.*

Proof. We proceed by induction on t. For $t = 0$, we have $X(0) \subseteq \langle X(0)\rangle$. Suppose the claim holds for some $0 \le t \le T$. Let $y \in X(t+1)$. Since $y = x_i(t+1) = \psi(X_i(t))$, for some $0 \le i < n$, the node y is in $\langle X_i(t)\rangle$. As $X_i(t) \subseteq X(t)$, it follows that $\langle X_i(t)\rangle \subseteq \langle X(t)\rangle$. Thus, $y \in \langle X_i(t)\rangle \subseteq \langle X(t)\rangle$. By the induction hypothesis, $X(t) \subseteq \langle X(0)\rangle$, so $y \in \langle X(t)\rangle \subseteq \langle X(0)\rangle$. □

We show that, for $0 \le t \le T^*$, if the set of nodes $X(t)$ does not form a clique, then the diameter of $X(t+1)$ is roughly half the diameter of $X(t)$. Recall that $D(U)$ is the maximum distance in G between the any two nodes in U. Note that $D(U') \le D(U)$ for $U' \subseteq U$. As in Section 6.1, $X_i(t) \subseteq X_j(t)$ or $X_i(t) \subseteq X_j(t)$ for $i, j \in \{0, \ldots, n-1\}$.

Lemma 15. *Let $0 \le t \le T^*$. Then $D(X(t+1)) \le \frac{2}{3}(D(X(t))+1)$. Moreover, if $\langle X(t)\rangle$ does not contain a 3-sun as an induced subgraph, then $D(X(t+1)) \le \frac{1}{2}(D(X(t))+1)$.*

Proof. Let $x_i(t+1), x_j(t+1) \in X(t+1)$. Recall that, by definition, $x_i(t+1) = \psi(X_i(t)) \in \langle X_i(t)\rangle$ and $x_j(t+1) = \psi(X_j(t)) \in \langle X_i(t)\rangle$. Without loss of generality, assume that $X_j(t) \subseteq X_i(t)$. Let H be the subgraph of G induced by $\langle X_i(t)\rangle$. Since G is bridged, the induced subgraph H is also bridged, by Lemma 13. Since $X_j(t) \subseteq X_i(t) \subseteq \langle X_i(t)\rangle$, both $x_j(t+1)$ and $x_i(t+1)$ are nodes of H. Moreover, $X_i(t) \subseteq X(t)$ implies that $D(X_i(t)) \le D(X(t))$. By Lemma 11 and Lemma 12,

$$\operatorname{diam}(H) = D(\langle X_i(t)\rangle) = D(X_i(t)) \le D(X(t)).$$

By definition, $x_i(t+1) = \psi(X_i(t))$, which is a node in the center of H. Hence, by Lemma 10,

$$d\left(x_i(t+1), x_j(t+1)\right) \le \operatorname{rad}(H) \le \frac{2}{3}\left(\operatorname{diam}(H)+1\right) \le \frac{2}{3}\left(D(X(t))+1\right).$$

For the second claim, if $\langle X(t)\rangle$ does not contain a 3-sun as an induced subgraph, Lemma 10 yields

$$d\left(x_i(t+1), x_j(t+1)\right) \le \operatorname{rad}(H) \le \frac{1}{2}\left(\operatorname{diam}(H)+1\right) \le \frac{1}{2}\left(D(X(t))+1\right). \quad \square$$

We can apply Lemma 15 repeatedly to ensure that we quickly end up in a subgraph with diameter at most two.

Lemma 16. *The set $X(T^*)$ has diameter at most two.*

Proof. First, we show by induction that for all $0 \le t \le T^*$, we have

$$D(X(t)) \le \left(\frac{2}{3}\right)^t (\operatorname{diam}(G)-2)+2.$$

The base case $t=0$ is vacuous as the distance between any two nodes is at most the diameter $\operatorname{diam}(G)$. For the inductive step, suppose the claim holds for some $0 \le t < T^*$. By Lemma 15,

$$\begin{aligned} D(X(t+1)) &\le \frac{2}{3}\left[D(X(t))+1\right] \\ &\le \frac{2}{3}\left[\left(\frac{2}{3}\right)^t (\operatorname{diam}(G)-2)+3\right] = \left(\frac{2}{3}\right)^{t+1} (\operatorname{diam}(G)-2)+2. \end{aligned}$$

Since $T^* = \lceil \log_{3/2} \operatorname{diam}(G)\rceil + 1$ and the diameter is an integer, we get that

$$D(X(T)^*) \le \left\lfloor \frac{2}{3\operatorname{diam}(G)}(\operatorname{diam}(G)-2)+2 \right\rfloor \le \left\lfloor \frac{2}{3}+2 \right\rfloor = 2. \quad \square$$

Lemma 17. *If the subgraph induced by $\langle X(t)\rangle$ has radius one, then $X(t+1)$ is a clique.*

Proof. Let $x_i(t+1), x_j(t+1) \in X(t+1)$ and assume without loss of generality that $X_j(t) \subseteq X_i(t)$. Now $\langle X_j(t)\rangle \subseteq \langle X_i(t)\rangle$. Since $x_i(t+1) = \psi(X_i(t))$ is a node in the center of subgraph induced by $\langle X_i(t)\rangle$, it is adjacent to the node $x_j(t+1)$. Hence, any two nodes in $X(t+1)$ are adjacent. $\square$

Thus, by Lemma 17, when the convex hull of $X(t)$ has radius 1, one more iteration suffices to reach agreement. Moreover, the algorithm solves the problem on any graph of radius one in one iteration. If the graph does not contain a 3-sun as an induced subgraph, then the algorithm converges in T^*+1 iterations. However, the above lemmas do not guarantee progress when the convex hull of $X(t)$ has diameter two and radius two. We handle this case next.

Handling 2-self-centered graphs In bridged graphs, the algorithm converges either to a clique or to a set whose convex hull induces a 2-self-centered subgraph. We show that if G is nicely bridged, i.e., any 2-self-centered convex subgraph is chordal, our algorithm makes progress. However, our approach does not work for all bridged graphs, as there are non-chordal 2-self-centered bridged graphs which do not have any simplicial nodes. (For example, see Appendix A.)

Recall that, if the center of $\langle X\rangle$ contains a non-simplicial node, then $\psi(X)$ is a non-simplicial node from the center of the subgraph induced by $\langle X\rangle$. This allows us to exclude simplicial nodes, which always exist in any chordal graph [21]. By removing any simplicial node, the convex hull shrinks, as shown by the next lemma.

Lemma 18. *Let U be a convex set. If $s \in U$ is simplicial in U, then $\langle U \setminus \{s\}\rangle = U \setminus \{s\}$.*

Proof. Suppose $s \in \langle U \setminus \{s\}\rangle$. This means that there are two nodes $u, v \in U \setminus \{s\}$ such that the node s lies on some shortest path $u = w_0, \dots, w_k = v$ between u and v. Let $w_i = s$ for some $0 < i < k$. Since w_i is simplicial in H, the nodes w_{i-1} and w_{i+1} are adjacent in H. But now the path $w_0, \dots, w_{i-1}, w_{i+1}, \dots, w_k$ is a shorter path from u to v, a contradiction. $\square$

We now recall a useful result about simplicial nodes in chordal graphs shown by Dirac [21].

Lemma 19. [21] *Every chordal graph G has a simplicial node. If G is not a clique, then it has two non-adjacent simplicial nodes.*

We can now show that if there are two values that are at least distance two apart, then the convex hull of $X(t)$ shrinks.

Lemma 20. *Let $0 \le t \le T$. If $D(X(t)) \ge 2$, then $\langle X(t+1)\rangle \subsetneq \langle X(t)\rangle$.*

Proof. Let H be the subgraph induced by $\langle X(t)\rangle$. Assume that H has diameter and radius two; otherwise, the claim follows from Lemma 16 and Lemma 17. Since G is nicely bridged, H is chordal. Let S be the set of nodes that are simplicial in H. Since H is chordal and has diameter at least two, S has two non-adjacent nodes, by Lemma 19. Let $S' = S \cap X(t+1)$. Observe that if $S' \subsetneq S$, then Lemma 18 implies $\langle X(t+1)\rangle \subsetneq \langle X(t)\rangle$. We show that the set $S' = S \cap X(t+1)$ is either empty or a clique, which implies that $S' \subsetneq S$.

For the sake of contradiction, let $x_i(t+1), x_j(t+1) \in S' \subseteq X(t+1)$ be two non-adjacent nodes. We may assume that $X_j(t) \subseteq X_i(t)$, which implies that $\langle X_j(t)\rangle \subseteq \langle X_i(t)\rangle$. Since $x_i(t+1)$ and $x_j(t+1)$ are non-adjacent, they are connected by a node $v \in \langle X_i(t)\rangle$. Note that v is not simplicial in $\langle X_i(t+1)\rangle$, but $x_i(t)$ is simplicial in $\langle X_i(t)\rangle$. By definition, $x_i(t+1) = \psi(X_i(t))$ is a node in the center of $\langle X_i(t)\rangle$, so it has eccentricity two. This means that v also has eccentricity two and v is also in the center. But, now, the center of $\langle X_i(t)\rangle$ contains a non-simplicial node v, which means that $x_i(t+1) = \psi(X_i(t))$ is non-simplicial in $\langle X_i(t)\rangle$ by definition of ψ. $\square$

Theorem 21. *Let G be a connected graph. If G is nicely bridged or has radius one, then there exists a wait-free algorithm that solves graphical approximate agreement on G in the iterated snapshot model.*

Proof. The case of radius one graphs is covered by Lemma 17. For nicely bridged graphs, it remains to verify that agreement and validity conditions of graphical approximate agreement are satisfied. Lemma 14 shows that validity is satisfied. Repeated application of Lemma 20 implies that $X(T+1)$ is a clique and, thus, agreement is satisfied. $\square$

7. Upper bound for synchronous message-passing systems

Finally, we adapt the algorithm of Section 6.1 to the synchronous message-passing setting under crash faults. This establishes the following upper bound. Note that Corollary 7 gives a lower bound of $\lfloor f/2\rfloor + 1$ rounds.

Theorem 22. *Let G be a connected graph. For any $0 \le f < n$, there exists an f-resilient synchronous message-passing algorithm for n processes that solves approximate agreement on G in $\lfloor f/2\rfloor + \lceil \log_2 \operatorname{diam}(G)\rceil + 1$ rounds.*

To show this, we use the following result [16].

Lemma 23. *For any $0 \le f < n$, there exists an f-resilient synchronous message-passing algorithm for n processes that solves 2-set agreement in $\lfloor f/2\rfloor + 1$ rounds.*

Overview The synchronous message-passing algorithm for graphical approximate agreement on G follows the same idea as the asynchronous algorithm given in Section 6.1. All processes:

– use 2-set agreement to reduce the size of the set of inputs to at most 2, and then
– run approximate agreement on a path for $\lceil \log_2 \operatorname{diam}(G)\rceil$ steps.

By Lemma 23 the first part takes $\lfloor f/2\rfloor + 1$ rounds. By using similar arguments as in Section 6.1, we show that the second part takes $\lceil \log_2 \operatorname{diam}(G)\rceil$ rounds, and that the validity and agreement properties of graphical approximate agreement are satisfied.

Preliminaries Let $x_i(0)$ be the input of process p_i for the graphical approximate agreement problem. As in Section 6.1, we let $g(u,v)$ be a fixed node in the center of a shortest path between u and v, so $d(u, g(u,v)), d(v, g(u,v)) \le \lceil d(u,v)/2\rceil$ holds. For any nonempty set $X \subseteq V$ of size at most two, define

$$\psi(X) = \begin{cases} u & \text{if } X = \{u\}, \\ g(u,v) & \text{if } X = \{u,v\}. \end{cases}$$

The algorithm Let $T = \lceil \log_2 \mathrm{diam}(G) \rceil$. Each process p_i:

(1) runs the 2-set agreement algorithm of Lemma 23 with input $x_i(0)$ for $\lfloor f/2 \rfloor + 1$ rounds,
(2) lets $x_i(1)$ be its output in the 2-set agreement algorithm,
(3) for $t = 1, \ldots, T$ rounds,
  – sends the node $x_i(t)$ to all processes in the system,
  – receives a set $X_i(t)$ of nodes from other processes,
  – lets $x_i(t+1) = \psi(X_i(t))$, and
(4) outputs the node $x_i(T+1)$.

Correctness The proof of correctness of the synchronous algorithm closely follows the proof of correctness of the asynchronous algorithm given in Section 6.1. If process p_i crashes before computing $X_i(t)$, we define $X_i(t)$ to be the empty set. Let $X(t) = \bigcup\{X_i(t) : 0 \le i < n\}$ be the set of values received by any process during the tth round of Step (3). Note that each non-faulty process p_i always sends the value $x_i(t)$ to itself, so $X_i(t)$ is nonempty if p_i has not crashed by round t of Step (3). We use $X(T+1)$ to denote the set of output values.

Lemma 24. *Let* $1 \le t \le T$. *Then* $1 \le |X(t)| \le 2$.

Proof. We proceed by induction on t. For the base case $t = 1$, observe that $|X(1)| \le 2$ holds by the agreement property of 2-set agreement. For the inductive step, suppose that $1 \le |X(t)| \le 2$ holds for some $1 \le t \le T$. Since $X_i(t) \subseteq X(t)$, it follows that $Z = \{X_i(t) : 0 \le i \le n-1\}$ contains at most two nonempty sets.

To see why, suppose that Z contains more than one nonempty set. Then there exists some processes p_i and p_j such that $X_i(t) = \{a\}$ for some value $a \in X(t)$ and $X_j(t) \neq \{a\}$ is nonempty. Since the algorithm is synchronous, this implies that p_i sent the value a to all other processes before computing the set $X_i(t)$, and so p_j received the value a before computing the set $X_j(t)$. Thus, $a \in X_j(t)$, so $X_j(t) \in \{\{a\}, X(t)\}$. Finally, since $x_i(t+1) = \psi(X_i(t))$, it follows that $X(t+1)$ will contain at most two different nodes. □

The proof of the following lemma is analogous to the proof of Lemma 9, so we do not present it.

Lemma 25. *Let* $1 \le t \le T$. *Then* $D(X(t+1)) \le \lceil D(X(t))/2 \rceil$.

Finally, we observe that by construction, the algorithm takes

$$\lfloor f/2 \rfloor + 1 + T = \lfloor f/2 \rfloor + \lceil \log_2 \mathrm{diam}(G) \rceil + 1$$

rounds. Using the same reasoning as in the proof of Theorem 8, it follows that the outputs of the algorithm satisfy agreement and shortest path validity. Thus, we have proved Theorem 22.

8. Conclusions

We now conclude by discussing how our results connect to recent work on approximate agreement on graphs. In addition, we highlight some open problems.

8.1. Solvability in the asynchronous setting

A new topological proof In this work, we investigated the solvability of approximate agreement on graphs in asynchronous systems, where processes communicate by reading and writing to shared memory. We gave a new topological proof, using a variant of Sperner's lemma, for the impossibility of solving 2-resilient c-cycle agreement and, hence, approximate agreement on cycles of length c, for $c \ge 4$. This impossibility result also holds for the 1-gathering problem on cycles of length $c \ge 4$, since clique validity and minimal path validity are equivalent on cycles. Because we gave a direct proof, we obtained additional insight into the graphical approximate agreement problem.

Graphical approximate agreement vs. 2-set agreement We also defined the notion of an impossibility labelling of a graph and used a reduction from 2-set agreement to prove that 2-resilient 1-gathering and graphical approximate agreement for $n \ge 3$ processes are also impossible on any graph that admits an impossibility labelling.

Moreover, there is simple reduction in this model from approximate agreement on any connected graph to 2-set agreement: as in Section 6.1, the processes first solve 2-set agreement on their inputs and then solve approximate agreement on a path in the graph between the outputs from 2-set agreement. Hence, 2-set agreement is solvable if and only if approximate agreement is solvable on a graph that admits an impossibility labelling.

Towards a complete characterisation of graphical approximate agreement Ledent [36] mentions that if a graph G admits an impossibility labelling, then its complex of cliques, $\kappa(G)$, is not simply connected. He conjectures that the converse is also true (when G is connected). This would provide an interesting characterisation of the family of connected graphs that admit impossibility labellings.

Recently, Liu [37] extended our work by introducing a different set of conditions and proving that there are no wait-free algorithms solving 1-gathering and, hence, approximate agreement on graphs satisfying these conditions. His proof, which is also topological, relies on a different generalisation of Sperner's lemma. In particular, his conditions are satisfied by all triangulated spheres (except the tetrahedron graph, which is a clique). Ledent [36] had previously conjectured that wait-free 1-gathering on these graphs is impossible. More generally, he conjectured that wait-free 1-gathering is solvable on a graph G if and only if its complex of cliques $\kappa(G)$ is contractible.

Although clique validity (the validity condition for 1-gathering) is weaker than shortest-path validity (the validity condition for graphical approximate agreement), resolving Ledent's conjecture will likely also shed light on graphical approximate agreement. Another open question is whether there are graphs on which 1-gathering is solvable, but approximate agreement with shortest path validity or minimal path validity is not solvable.

Extension-based proofs It is also interesting to consider whether these impossibility results can be proved using other techniques. Alistarh, Aspnes, Ellen, Gelashvili and Zhu [6] introduced extension-based proofs to model inductive impossibility arguments, such as valency arguments. They proved that extension-based proofs cannot be used to prove the impossibility of $(n-1)$-set agreement among $n \geq 3$ processes.

In an earlier version of this paper [7], we showed that extension-based proofs cannot be used to prove the impossibility of 4-cycle agreement. Recently, Liu [37] generalised this result to show that extension-based proofs cannot prove the impossibility of 1-gathering on any graph.

New algorithms On the positive side, we gave algorithms for solving graphical approximate agreement with shortest path validity (and, hence, for minimal path validity and clique validity). We showed that there is a 1-resilient asynchronous algorithm for any connected graph and a wait-free algorithm for any graph that is nicely bridged. This extends the class of graphs for which approximate agreement algorithms were known. It remains open to find an algorithm that solves graphical approximate agreement on all bridged graphs (not just nicely bridged graphs) or show that such an algorithm does not exist.

8.2. Complexity in the synchronous setting

Finally, we investigated the round complexity of graphical approximate agreement in the synchronous message-passing model. In this model, we showed that any f-resilient approximate agreement algorithm for a graph that admits an impossibility labelling requires at least $\lfloor f/2 \rfloor + 1$ rounds for $3 \leq f < n$.

We also gave an f-resilient algorithm that solves graphical approximate agreement on any graph G in $\lfloor f/2 \rfloor + \lceil \log_2 \operatorname{diam}(G) \rceil + 1$ rounds, for $0 \leq f < n$. On graphs that admit an impossibility labelling, these results leave a gap of $\lceil \log_2 \operatorname{diam}(G) \rceil$ rounds between the lower and upper bounds. On paths, it is known that iterative algorithms require $\Theta(\log \operatorname{diam}(G))$ rounds to converge [26]. What is the exact round complexity of f-resilient graphical approximate agreement, for $f < n$, in the synchronous message-passing model?

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 805223 ScaleML) and under the Marie Skłodowska-Curie grant agreement No. 840605 and from the Natural Sciences and Engineering Research Council of Canada grant RGPIN-2020-04178. Part of this work was done while Faith Ellen was visiting IST Austria.

Appendix A. Examples of nicely bridged graphs

In this section, we give some sufficient conditions for a bridged graph to be nicely bridged. If $C \cup \{x\}$ is an induced wheel of G, then we say that the wheel $C \cup \{x\}$ is *uniquely centered* in G if there is no $y \neq y$ such that $C \cup \{y\}$ is also an induced wheel in G.

Theorem 26. *Let G be a bridged graph. Then G is nicely bridged if any of the following hold:*

(a) *G is chordal.*

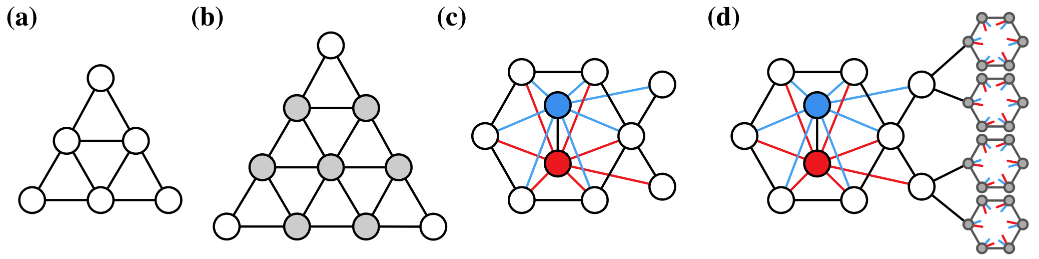

Fig. 4. Examples of nicely bridged and not nicely bridged graphs. (a) The 3-sun is chordal and nicely bridged. (b) A nicely bridged graph. The grey nodes form a uniquely centered wheel. (c) A non-chordal 2-self-centered bridged graph. The cycle has a wheel that is not uniquely centered: both red and blue nodes are both axles of the wheel. (d) A 2-self-centered bridged graph with no simplicial nodes. Each grey node is connected to the red and blue node.

(b) *G does not contain a 3-sun as an induced subgraph.*
(c) *Every induced wheel of G is uniquely centered.*
(d) *G has no cliques of size four.*

The 3-sun is depicted in Fig. 4(a). It is chordal, and hence, nicely bridged. Fig. 4(b) shows a bridged graph whose every wheel is uniquely centered. In contrast, Fig. 4(c) shows a bridged graph with a wheel that is not uniquely centered. This graph is also not nicely bridged. Fig. 4(d) gives an example of a bridged graph which has no simplicial nodes.

Recall that bridged graphs do not contain any induced cycles of length $4 \le k \le 5$, as every such cycle would be an isometric cycle of length at least four. In particular, any non-chordal bridged graph will have an induced cycle of length at least six. To establish Theorem 26, we start with the following lemma.

Lemma 27. *Suppose G is a non-chordal bridged graph of diameter two. Let C be a shortest induced k-cycle of length $k \ge 6$ in G. Then there exists a node x such that $C \cup \{x\}$ induces a k-wheel in G.*

Proof. Let $C = \{c_1, \dots, c_k\}$ be the shortest induced cycle of length $k \ge 6$. Since G has diameter two and C is an induced cycle, we have $d(c_1, c_{k-2}) = 2$. Thus, there is some node y connecting c_1 and c_{k-2}. Since G is bridged, then no subset of $\{c_1, x, c_{k-2}, c_{k-1}, c_k\}$ can form an induced four or five cycle. Thus, so x is adjacent to c_k and c_{k-1}. The $(k-1)$-cycle $\{c_1, \dots, c_{k-2}, x\}$ cannot be an induced cycle either, as the shortest induced cycle had length k. This also implies that y has to be adjacent to each $c_1, \dots, c_{k-2}$ and $C \cup \{x\}$ induces a k-wheel. □

Lemma 28. *Suppose $G = (V, E)$ is a bridged graph that contains an induced cycle of length $k > 3$. Let C be the shortest such cycle and suppose $C \cup \{x\}$ induces a uniquely centered wheel. For any $v \in V$, let $A(v)$ be the neighbours of v in C. If $A(v) \ne \emptyset$, then the following hold:*

(a) *The set $A(v)$ induces a path of length at most three.*
(b) *If v is not adjacent to x, then $A(v)$ induces a path of length at most two.*

Proof. Suppose $A(v)$ is nonempty and does not induce a path. Choose from $A(v)$ a pair of two such nodes c_i and c_j which have the shortest distance in C. Without loss of generality, we may assume these are the nodes c_1 and c_i for some $2 < i \le k/2 + 1$. Now $\{v, c_1, \dots, c_i\}$ induces a cycle of length $3 < i + 1 \le k/2 + 2 < k$. This contradicts the fact that k was the length of the shortest induced cycle of length at least four.

Next we show that the path induced by $A(v)$ has length at most three. Without loss of generality, assume that $A(v) = \{c_1, \dots, c_h\}$. For the sake of contradiction, assume that $h > 3$. Since C is uniquely centered, v cannot be adjacent to all nodes of C. Hence, this path has length $3 < h < k$. Now $\{c_1, v, c_h, \dots, c_{h+1}, \dots c_k\}$ induces a cycle of length $3 < k - h + 2 < k - 1$, which is a contradiction. For the last claim, observe that if v is not adjacent to x and $A(u) = \{c_1, \dots, c_3\}$, then $\{x, c_1, v, c_3\}$ induces a four cycle. □

Lemma 29. *If G is bridged and its every induced wheel is uniquely centered, then G is nicely bridged.*

Proof. Suppose G is not nicely bridged, that is, there is a H is 2-self-centered convex subgraph that is not chordal. By Lemma 13 the graph H is bridged. Since H is bridged, but not chordal, H contains some induced cycle of length at least six. Let $k > 5$ be the length of the shortest induced cycle in H. Fix $C = \{c_1, \dots, c_k\}$ to be some induced cycle of length k. By Lemma 27 there exists some node x in H such that $C \cup \{x\}$ induces a k-wheel.

Because H is 2-self-centered, x has eccentricity two. Thus, there exists some node y in H such that $d(x, y) = 2$. We show that the existence of such y leads to the existence of a induced cycle of length 4, 5, or $k - 1$, which contradicts the assumption that C was the shortest cycle of length $k > 3$. Clearly, $y \notin C$. By Lemma 28, y is adjacent to at most three consecutive nodes $A(y)$ of C. Without loss of generality, assume that $A(y) \subseteq \{c_1, c_2, c_3\}$.

First, we show that $A(y)$ must be empty. Observe that $d(y, c_5) = 2$. Hence, there is some node $v \notin C \cup \{x\}$ that is adjacent to both y and c_5. However, v can be adjacent to at most three consecutive nodes $A(v)$ of C. If v is not adjacent to c_3, then either $\{v, y, c_3, c_4\}$ induces a four cycle or $\{v, y, c_3, c_4, c_5\}$ induces a five cycle. Thus, v is adjacent to c_3. But then $\{c_1, y, z, c_5, \ldots, c_k\}$ induces a $(k-1)$-cycle. Thus, $A(y)$ must be empty.

Since $A(y)$ is empty, the node y is not adjacent to any node $c_i \in C$. But since H is 2-self-centered, $d(y, c_i) = 2$ for all $c_i \in C$. Choose a neighbour u of y that is connected to c_1. By Lemma 28 the set $A(u)$ induces a path. Without loss of generality, assume that $A(u) = \{c_1, \ldots, c_j\}$ for some $1 \le j \le 3$. Since $d(y, c_5) = 2$, there is some v that is adjacent to y and c_5. By Lemma 28 the set $A(v)$ induces a path of length at most three and so $A(v) \subseteq \{c_3, \ldots, c_k, c_1\}$. Note that $A(u) \cap A(v)$ can intersect either at c_1 or c_3, since C has length $k \ge 6$.

(1) Consider the case $A(u) \cap A(v) \neq \{c_1\}$. Now $A(v) \subseteq \{c_3, \ldots, c_h\}$ for $5 \le h \le k$. If u and v are adjacent, then $\{c_1, u, v, c_h, \ldots, c_k\}$ is an induced cycle of length at least four and less than $k - (h-5) \le k$, which is a contradiction. Hence, u and v are not adjacent. This means that a subset of $C' = \{y, u, c_1, x, c_5, v\}$ induces a cycle of length at least four. Since G cannot have any induced cycles of length four or five, C' must be an induced 6-cycle. Since C was the shortest induced cycle of length $k \ge 6$, it follows that $k = 6$. By Lemma 27 there is some $z \neq x$ connected to all nodes of C'. Now $c_1, c_5 \in A(z)$. Since $A(z)$ is an induced path of length at most three, this implies that $c_6 \in A(z)$. Now either $\{y, v, c_6, c_1, u\}$ is an induced 5-cycle or $\{y, v, c_1, u\}$ is an induced 4-cycle, a contradiction.

(2) Consider the case $A(u) \cap A(v) = \{c_1\}$. This means that $k = 6$ and $A(v) = \{c_5, c_6, c_1\}$ and v is adjacent to x by Lemma 28. Since $d(y, c_3) = 2$, there is some w adjacent to y and c_3. Now $A(w)$ induces a path of length at most three. Suppose w is not adjacent to v. Then either $\{y, w, c_3, x, v\}$ or $\{y, w, x, v\}$ is an induced cycle. Hence, w is adjacent to v. This implies that $A(w) = \{c_3, c_4, c_5\}$, as otherwise we could find another induced cycle of length either four or five. Thus w is adjacent to x by Lemma 28.
If w is not adjacent to u, then either $\{y, u, x, w\}$ or $\{y, u, c_1, x, w\}$ is an induced cycle of length four or five, respectively. Thus, w is adjacent to u. If $j \le 2$ we have that $\{u, c_j, \ldots, c_3, w\}$ is an induced cycle of length four or five. Hence $A(u) = \{c_1, c_2, c_3\}$. But then $\{u, c_3, c_4, c_5, v\}$ is an induced 5-cycle, which is a contradiction. □

Proof of Theorem 26. (a) The claim follows from the fact that every induced subgraph of a chordal graph is also chordal. Hence, this also holds for any subgraph induced by a convex set.

(b) Suppose H is a diameter two subgraph of G induced by a convex set S. By Lemma 13 $H = G[S]$ is bridged. Since G does not contain an induced 3-sun, neither does H. Thus, by Lemma 10 we have $\mathrm{rad}(H) \le (\mathrm{diam}(H)+1)/2 = 3/2$. Since the radius must be integral, H has radius one, and cannot be 2-self-centered. Therefore, G is nicely bridged.

(c) This is the claim from Lemma 29.

(d) If G is bridged and has no induced wheels, then Lemma 27 implies that it is chordal, and thus, also nicely bridged. Hence, suppose there exists an induced wheel in G. We show the claim by establishing that the nonexistence of cliques of size four implies that every induced wheel of G is uniquely centered. Then claim (c) implies that G is nicely bridged.

Suppose there exists a k-wheel for $k > 3$ that is not uniquely centered. Let $C = \{c_0, \ldots, c_{k-1}\}$ be the induced k-cycle forming the wheel and $x \neq y$ be two nodes such that $C \cup \{x\}$ and $C \cup \{y\}$ both induce a k-wheel. Note that x and y are not adjacent, as otherwise $\{c_0, c_1, x, y\}$ would be a clique of size four. But since x and y are not adjacent, the set $\{x, c_0, y, c_2\}$ induces a four cycle, which contradicts the fact that G was bridged. □

References

[1] Ittai Abraham, Yonatan Amit, Danny Dolev, Optimal resilience asynchronous approximate agreement, in: Proceedings of the Eighth International Conference on Principles of Distributed Systems (OPODIS), 2004, pp. 229–239.
[2] Karim A. Adiprasito, Eran Nevo, Jose A. Samper, Higher chordality: from graphs to complexes, Proc. Am. Math. Soc. 144 (8) (2016) 3317–3329.
[3] Yehuda Afek, Danny Dolev, Eli Gafni, Michael Merritt, Nir Shavit, Atomic snapshots of shared memory, J. ACM 40 (4) (1993) 873–890.
[4] Agmon Noa, David Peleg, Fault-tolerant gathering algorithms for autonomous mobile robots, SIAM J. Comput. 36 (1) (2006) 56–82.
[5] Manuel Alcántara, Armando Castañeda, David Flores-Peñaloza, Sergio Rajsbaum, The topology of look-compute-move robot wait-free algorithms with hard termination, Distrib. Comput. 32 (3) (2019) 235–255.
[6] Dan Alistarh, James Aspnes, Faith Ellen, Rati Gelashvili, Leqi Zhu, Why extension-based proofs fail, in: Proceedings of the Fifty First Annual ACM Symposium on Theory of Computing (STOC), ACM, 2019.
[7] Dan Alistarh, Faith Ellen, Joel Rybicki, Wait-free approximate agreement on graphs, https://arxiv.org/abs/2103.08949, 2021.
[8] Atanassov Krassimir, On Sperner's lemma, Studia Sci. Math. Hung. 32 (1996).
[9] Hagit Attiya, Nancy Lynch, Nir Shavit, Are wait-free algorithms fast?, J. ACM 41 (4) (July 1994) 725–763, https://doi.org/10.1145/179812.179902.
[10] Elizabeth Borowsky, Eli Gafni, Immediate atomic snapshots and fast renaming (extended abstract), in: Proceedings of the Twelfth ACM Symposium on Principles of Distributed Computing (PODC), 1993, pp. 41–51.
[11] Elizabeth Borowsky, Eli Gafni, Generalized FLP impossibility result for t-resilient asynchronous computations, in: Proceedings of the Twenty Fifth Annual ACM Symposium on Theory of Computing (STOC), 1993, pp. 91–100.
[12] Elizabeth Borowsky, Eli Gafni, A simple algorithmically reasoned characterization of wait-free computation, in: Proceedings of the Sixteenth ACM Symposium on Principles of Distributed Computing (PODC), 1997, pp. 189–198.
[13] Elizabeth Borowsky, Eli Gafni, Nancy Lynch, Sergio Rajsbaum, The BG distributed simulation algorithm, Distrib. Comput. 14 (3) (2001) 127–146, https://doi.org/10.1007/PL00008933.

[14] J.A. Bundy, U.S.R. Murty, Graph Theory with Applications, Elsevier Science Publishing Co., ISBN 0-444-19451-7, 1982.
[15] Armando Castañeda, Sergio Rajsbaum, Matthieu Roy, Convergence and covering on graphs for wait-free robots, J. Braz. Comput. Soc. 24 (1) (2018) 1.
[16] Soma Chaudhuri, Maurice Herlihy, Nancy A. Lynch, Mark R. Tuttle, Tight bounds for k-set agreement, J. ACM 47 (5) (2000) 912–943, https://doi.org/10.1145/355483.355489.
[17] Benny Chor, Amos Israeli, Ming Li, On processor coordination using asynchronous hardware, in: Proceedings of the Sixth Annual ACM Symposium on Principles of Distributed Computing (PODC), 1987, pp. 86–97.
[18] Serafino Cicerone, Gabriele Di Stefano, Alfredo Navarra, Asynchronous Robots on Graphs: Gathering, Springer International Publishing, Cham, ISBN 978-3-030-11072-7, 2019, pp. 184–217.
[19] Mark Cieliebak, Paola Flocchini, Giuseppe Prencipe, Nicola Santoro, Distributed computing by mobile robots: gathering, SIAM J. Comput. 41 (4) (2012) 829–879.
[20] Jesus A. De Loera, Elisha Peterson, Francis Edward Su, A polytopal generalization of Sperner's lemma, J. Comb. Theory, Ser. A 100 (1) (2002) 1–26, https://doi.org/10.1006/jcta.2002.3274.
[21] Gabriel Andrew Dirac, On rigid circuit graphs, in: Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg, vol. 25, Springer, 1961, pp. 71–76.
[22] Danny Dolev, Nancy A. Lynch, Shlomit S. Pinter, Eugene W. Stark, William E. Weihl, Reaching approximate agreement in the presence of faults, J. ACM 33 (3) (May 1986) 499–516, https://doi.org/10.1145/5925.5931.
[23] Leonhard Euler, Solutio problematis ad geometriam situs pertinentis, in: Commentarii academiae scientiarum Petropolitanae, 1741, pp. 128–140.
[24] Martin Farber, On diameters and radii of bridged graphs, Discrete Math. 73 (3) (1989) 249–260.
[25] Martin Farber, Robert E. Jamison, On local convexity in graphs, Discrete Math. 66 (3) (1987) 231–247, https://doi.org/10.1016/0012-365X(87)90099-9.
[26] Alan David Fekete, Asymptotically optimal algorithms for approximate agreement, Distrib. Comput. 4 (1) (1990) 9–29.
[27] Alan David Fekete, Asynchronous approximate agreement, Inf. Comput. 115 (1) (1994) 95–124.
[28] Michael J. Fischer, Nancy A. Lynch, Michael S. Paterson, Impossibility of distributed consensus with one faulty process, J. ACM 32 (2) (1985) 374–382, https://doi.org/10.1145/3149.214121.
[29] Michael J. Fischer, Nancy A. Lynch, Michael Merritt, Easy impossibility proofs for distributed consensus problems, Distrib. Comput. 1 (1) (1986) 26–39.
[30] Matthias Függer, Thomas Nowak, Fast multidimensional asymptotic and approximate consensus, in: Proceedings of the Thirty Second International Symposium on Distributed Computing (DISC), vol. 121, 2018, pp. 27:1–27:16.
[31] Eli Gafni, Round-by-round fault detectors (extended abstract): unifying synchrony and asynchrony, in: Proceedings of the Seventeenth Annual ACM Symposium on Principles of Distributed Computing (PODC), New York, NY, USA, in: Association for Computing Machinery, 1998, pp. 143–152.
[32] Maurice Herlihy, Sergio Rajsbaum, The decidability of distributed decision tasks (extended abstract), in: Proceedings of the Twenty Nineth Annual ACM Symposium on Theory of Computing (STOC), 1997, pp. 589–598.
[33] Maurice Herlihy, Nir Shavit, The topological structure of asynchronous computability, J. ACM 46 (6) (1999) 858–923, https://doi.org/10.1145/331524.331529.
[34] Maurice Herlihy, Dmitry Kozlov, Sergio Rajsbaum, Distributed Computing Through Combinatorial Topology, 1st edition, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2013.
[35] Gunnar Hoest, Nir Shavit, Toward a topological characterization of asynchronous complexity, SIAM J. Comput. 36 (2) (2006) 457–497.
[36] Jérémy Ledent, Brief announcement: variants of approximate agreement on graphs and simplicial complexes, in: Proceedings of the Fortieth ACM Symposium on Principles of Distributed Computing (PODC), 2021, pp. 427–430.
[37] Shihao Liu, The impossibility of approximate agreement on a larger class of graphs, in: Proceedings of the Twenty Sixth International Conference on Principles of Distributed Systems (OPODIS), 2022, pp. 25:1–25:12.
[38] Michael C. Loui, Hosame H. Abu-Amara, Memory requirements for agreement among unreliable asynchronous processes, Adv. Comput. Res. 4 (163–183) (1987) 31.
[39] Hammurabi Mendes, Christine Tasson, Maurice Herlihy, Distributed computability in byzantine asynchronous systems, in: Proceedings of the 46th ACM Symposium on Theory of Computing, 2014, pp. 704–713.
[40] Hammurabi Mendes, Maurice Herlihy, Nitin Vaidya, Vijay K. Garg, Multidimensional agreement in Byzantine systems, Distrib. Comput. 28 (2015) 423–441, https://doi.org/10.1007/s00446-014-0240-5.
[41] Thomas Nowak, Joel Rybicki, Byzantine approximate agreement on graphs, in: Proceedings of the Thirty Third International Symposium on Distributed Computing (DISC), vol. 146, 2019, pp. 29:1–29:17.
[42] Marshall C. Pease, Robert E. Shostak, Leslie Lamport, Reaching agreement in the presence of faults, J. ACM 27 (2) (1980) 228–234, https://doi.org/10.1145/322186.322188.
[43] Michael Saks, Fotios Zaharoglou, Wait-free k-set agreement is impossible: the topology of public knowledge, SIAM J. Comput. 29 (5) (2000) 1449–1483, https://doi.org/10.1137/S0097539796307698.
[44] Erik Schenk, Faster approximate agreement with multi-writer registers, in: Proceedings of the Thirty Sixth Annual IEEE Symposium on Foundations of Computer Science (FOCS), 1995, pp. 714–723.