Computing in the Presence of Concurrent Solo Executions

Sergio Rajsbaum, Michel Raynal, Julien Stainer

Publications Internes de l'IRISA
ISSN : 2102-6327
PI 2004 – May 2013

Computing in the Presence of Concurrent Solo Executions

Sergio Rajsbaum[*] Michel Raynal[**] ,[***] Julien Stainer[***]

Abstract: In a *wait-free* model any number of processes may crash. A process runs *solo* when it computes its local output without receiving any communication from other processes, either because they crashed or they are too slow. While in wait-free shared-memory models at most one process may run solo in an execution, any number of processes may have to run solo in an asynchronous wait-free message-passing model.

This paper is on the computability power of models in which several processes may concurrently run solo. It first introduces a family of round-based wait-free models, called the d-*solo* models, $1 \leq d \leq n$, where up to d processes may run solo. The paper gives then a characterization of the colorless tasks that can be solved in each d-*solo* model. The paper introduces the (d, ϵ)-*solo approximate agreement* problem, which generalizes ϵ-approximate agreement. It proves that (d, ϵ)-solo approximate agreement can be solved in the d-solo model, but it cannot be solved in the $(d+1)$-solo model. The paper also studies the relation linking d-set agreement and (d, ϵ)-solo approximate agreement in asynchronous wait-free message-passing systems.

These results establish for the first time a hierarchy of wait-free models weaker than the basic read/write model, that are nevertheless still strong enough to solve many tasks. The new failure model, based on taking into account solo executions, has message passing as well as shared-memory instantiations.

Key-words: Approximate agreement, Asynchronous system, Communication object, Distributed computability, Message-passing, Read/write shared memory, Solo execution, System partitioning, Task solvability, Topology, Wait-freedom

---

Calcul en présence d'exécutions solo concurrentes

Résumé : *Dans un modèle* wait-free*, un nombre quelconque de processus peut s'arrêter de fonctionner de façon inopinée. Un processus s'exécute* en solo *lorsqu'il calcule sa valeur de sortie sans communiquer avec les autres processus. Dans les modèles* wait-free *dans lesquels les processus communiquent via une mémoire partagée, au plus un processus peut s'exécuter en solo, alors que dans les modèles* wait-free *dans lesquels la communication se fait par échange de messages, un nombre quelconque d'entre eux peut avoir à s'exécuter en solo. Cet article étudie la calculabilité dans des modèles intermédiaires, appelés d-solo modèles, modèles dans lesquels jusqu'à* d *processus s'exécutent en solo.*

Mots clés : *Accord approché, système asynchrone, objet de communication, calculabilité distribuée, système à passage de messages, mémoire partagée, exécution en solo, partitionnement de système, solvabilité d'une tâche, topologie, synchronisation sans attente.*

---

[*] Instituto de Mathematicas, UNAM, D.F. 04510, Mexico
[**] Institut Universitaire de France
[***] ASAP : équipe commune avec l'Université de Rennes 1 et Inria

1 Introduction

Distributed computability The computability power of a distributed model depends on its communication, timing, and failure assumptions. A basic result is the impossibility to solve consensus in an asynchronous read/write or message-passing system even if only one process may crash [14, 24]. When looking at the communication medium and assuming asynchronous processes prone to crash failures, a read/write system and a message-passing system have the same computability power if and only if less than half of the processes may crash [1]. If a majority of the processes may crash, the message passing model is weaker than the shared memory model because partitions can occur.

The power of a model has been studied in detail with respect to *tasks*, which are the distributed equivalent of a function in sequential computing. Each process gets only one part of the input, and after communicating with the others, decides on an output value, such that collectively, the various local outputs produced by the processes respect the task specification, which is defined from the local inputs of the processes. This paper concentrates on the class of *colorless tasks* (e.g., [4, 6, 18, 19, 20]), where the specification is in terms of possible inputs and outputs, but without referring to which process gets which input or produces which output. Among the previously studied notable tasks, many are colorless, such as consensus [14], set agreement [9], approximate agreement [11] and loop agreement [19], while some are not, like renaming [2, 8].

Wait-freedom and solo execution This paper considers *wait-free* distributed asynchronous crash-prone computation models. Wait-free has two (complementary) meanings. First, it means the model allows up to $n-1$ processes to crash, where n is total number of processes. The term *wait-freedom* is also used to state a liveness condition, that requires every non-faulty process to progress in its computation and eventually decide (i.e., compute a result) whatever the behavior of the other processes [17].

In a wait-free model where processes must satisfy the wait-freedom liveness condition, a process has to make progress in its computation even in the extreme case where all other processes have crashed, or are too slow, and consequently be forced to decide without knowing their input values. Hence, for each process, there are executions where this process perceives itself as being the only process participating in the computation.

More generally, we say that a process executes *solo* if it computes its local output without knowing the input values of the other processes.

Two extreme wait-free models: shared memory and message passing In a model where processes communicate by reading and writing to shared registers, at most one process can run solo in the same execution. This is because, while a process is running solo, its writes and reads from the shared memory, and once it finishes its computation it writes to the memory its decision. Any other process that starts running, will be able to read the history left by the solo process in the memory.

When considering message-passing communication, all processes may have to run solo concurrently in the extreme case, where messages are arbitrarily delayed, and each process perceives the other processes as having crashed. Only tasks that can be solved without communication can be computed in this model.

Investigating the computability power of intermediary models The aim of the paper is to study the computability power of asynchronous models in which several processes may run solo in the same execution. More precisely, assuming that up to d processes may run solo, the paper addresses the following questions:

- How to define a computation model in which up to d processes may run solo?
- Which tasks can be computed in such a model?

Our aim is to study these questions in a clean theoretical framework, and investigate for the first time models weaker than the basic wait-free read/write model. However, we hope that our results might be relevant to other intermediate models, such as distributed models over fixed or wireless networks, and models where processes communicate via multi-writer/multi-reader registers, when the number of registers is smaller than the number of processes [10].

To simplify the technical development, following [5], the paper develops a theoretical framework based on a round-based, iterated model (IIS) that has been proved useful in many other papers, (e.g., [29]). Processes execute an infinite sequence of asynchronous rounds and communicate through specific objects called *immediate snapshot* objects. An immediate snapshot object is a high-level read/write object such that a new object is associated with each round and, when it executes round r, a process can access only the object associated with round r. A main interest of the IIS model is that, from a task computability point of view, it has the same power as the read/write wait-free model [5]. Also, the topology of the IIS model is easier to analyze, establishing a good foundation to analyze task solvability in various distributed computing models (e.g., [21]).

Contributions of the paper The following results provide some answers to the previous questions:

- The definition of a family of d-*solo models,* each parameterized with an integer d, $1 \le d \le n$. The 1-solo model corresponds to the IIS model (which is equivalent to the read/write wait-free model [5]), while the n-solo model corresponds to the round-based wait-free message-passing model.
- A characterization of the set of colorless tasks that can be solved in the d-solo model, $1 \le d \le n$. The characterization connects topology via a new form of complex subdivisions with *colorless* algorithms. Moreover, more tasks can be solved in the d-solo model than in the $(d+1)$-solo model, for $1 \le d < n$, which establishes a hierarchy of solo models.
- Any d-*solo* model with $d \ge 2$, is weaker than the read/write wait-free model, yet there are natural, non-trivial problems that can be solved, called (d,ϵ)-*solo approximate agreement* (in short (d,ϵ)-SAA). The (d,ϵ)-SAA can be solved in the d-solo model, for any $\epsilon > 0$, but not in the $(d+1)$-solo model.
- Finally, the d-solo model is related to d-set agreement. This relation shows that, for $d < n$, d-set agreement is strong enough to solve (d,ϵ)-solo approximate agreement but is too weak to solve $(d-1,\epsilon)$-solo approximate agreement in the wait-free message-passing model. This provides us with a better insight on a bound on the "maximal partitioning" allowed to solve $(d-1,\epsilon)$-solo approximate agreement in the wait-free message-passing model.

The (d,ϵ)-solo approximate agreement problem is a generalization of approximate agreement [11]. The input of each process consists of a point in the Euclidean space $\mathbb{R}^N$ ($N \ge d$). The *validity* property states that each process p_i has to decide a point which is in the convex hull of all the input points. The *agreement* property states that at most d processes may decide any point in the convex hull of the input points (let CH be the convex hull defined by these at most d points), while the other processes have to decide values whose distance to CH is at most ϵ. Actually, the convex hull of solo processes is an "attractor" for the set of decided values.

When $d = 1$, validity and agreement imply that the Euclidean distance between any pair of points decided by the processes has to be upper bounded by a predefined constant. Thus, $(1,\epsilon)$-solo approximate agreement problem in $\mathbb{R}^m$ is essentially the problem that has been recently considered in the context of t Byzantine failures and asynchronous message-passing systems [26, 33], where it is shown that it can be solved iff $n > t(m+2)$.

Consider the application described in [26], where n asynchronous robots move in a common d-dimensional space (CORDA model where $d \in \{2,3\}$ [15, 28]). The initial position of a robot constitutes its input value. The algorithm described in [26] directs the robots to meet within the convex hull of their initial positions, and the final distance between any two of them is upper bounded by some predefined constant. In our d-solo model, any algorithm solving the (d,ϵ)-solo approximate agreement problem allows up to d robots to remain input-isolated and hence are allowed to remain at their initial positions (they permanently commit "receive omission" failures), while the distance of any other non-faulty robot to the convex hull of the solo robots is at most ϵ.

The colorless tasks that are solvable in the wait-free iterated immediate snapshot (IIS) model have been characterized in [20]. Due to the simulations in [5, 16], the characterization holds for the usual read/write wait-free model (and in fact extends to t-resilient models by [6]). Section 4 extends the characterization of [20] to the d-solo model, $1 \le d \le n$. Our characterization in terms of colorless algorithms permits the use of standard subdivisions, instead of chromatic subdivisions used in previous papers. We believe colorless algorithms are interesting in themselves, and indeed, for $d = 1$, if a colorless task is solvable, it is solvable by a colorless algorithm. For $d > 1$ we defer the proof that colorless algorithms and general algorithms can solve a very similar class of tasks.

One of the central results of topology is the Simplicial Approximation Theorem [27], which establishes what is a "discrete version" of a continuous map. This theorem is also central for the wait-free characterization theorem of [22] and t-resilient extension (e.g., [20]). However, this theorem cannot be used in a d-solo model, $d > 1$, because it is no longer the case that the diameter of the simplexes in a subdivision is reduced. Not even the Relative Simplicial Approximation Theorem [34] can be directly used.

Roadmap The paper is composed of 7 sections. Section 2 introduces base definitions, the communication objects, and the d-solo model. Section 3 investigates colorless tasks in the d-solo model, while Section 4 focuses on what can be computed in the presence of concurrent solo executions. Then, Section 5 defines the (d,ϵ)-solo approximate agreement problem, shows that it can be solved in the d-solo model and cannot in the $(d+1)$-solo model, thereby defining a strict hierarchy of distributed computing models. Section 6 is on the relation between d-set agreement and (d,ϵ)-solo approximate agreement in wait-free message-passing systems. Finally, Section 7 concludes the paper. Topology notions relevant to the paper, and additional technical developments are given in appendices.

2 Tasks, Processes, Communication Object, and Iterated Model

Tasks A task is a one-shot distributed computing problem specified in terms of an input/output relation Δ. Each process starts with a private input value and must eventually compute a private output value. The task specifies the possible initial configurations. An initial configuration I specifies the input value of each process. Similarly, the output values produced by the processes in an execution represents an output configuration O.

A task $(\mathcal{I}, \mathcal{O}, \Delta)$ is defined by a set of input configurations $\mathcal{I}$, a set of possible output configurations $\mathcal{O}$, and a relation Δ which specifies which output configurations $O \in \mathcal{O}$ are correct for each input $I \in \mathcal{I}$. A more formal description appears in Section 3.1 and in previous papers such as in [21, 22].

Processes The system model is made up of n asynchronous (deterministic) sequential processes, $p_1, \ldots, p_n$, which proceed in in asynchronous rounds. The index i of process p_i is sometimes used to denote p_i. Up to $n-1$ processes may crash. Once a process crashes, it never recovers. We say the model is *wait-free*.

Rounds and communication objects A communication object $CO[r]$ is associated with each round r and this object is the only means for the processes to communicate during round r. The rounds are *communication-closed* [12] which means that, when a process executes a round, it can communicate with other processes only through the object associated with this round.

More precisely, $CO[r]$ is a one-shot object (i.e., each process accesses it only once) which provides the processes with a single operation denoted $\mathsf{communicate}(i, v)$, where v is the value that the invoking process p_i wants to communicate to the other processes during round r. Such an invocation returns to p_i a set of pairs (process identity, value) deposited into $CO[r]$ by other processes during round r.

Iterated model Each process p_i executes the algorithm skeleton described in Figure 1, where the local computation parts are related to the particular task that is solved. The local variable r_i is the local round number, ℓs_i contains p_i's local state, while $view_i$ contains all the pairs $(j, \ell s_j)$ communicated to p_i during the current round. The local transition function $\delta_i()$ defines the new local state of p_i according to its previous local state and the pairs $(j, \ell s_j)$ it has obtained from $CO^d[r]$ (the parameter d is explained below in Section 2.1). To solve a task, it is necessarily to instantiate accordingly $\delta_i()$, the predicate $\mathsf{decision}()$ and the function $\mathsf{dec\_val}()$: $\mathsf{decision}()$ allows p_i to decide, while $\mathsf{dec\_val}()$ allows it to compute the decided value. As we are interested in computability and not efficiency, we assume a *full information* algorithm, i.e., at the end of each round r_i, ℓs_i contains the value of $view_i$, and δ_i can be task independent. However, we will see in Section 3 that in some cases, tasks can be solved without communicating all a process knows.

$r_i \leftarrow 0$; $\ell s_i \leftarrow$ initial local state;
loop forever $r_i \leftarrow r_i + 1$;
    $view_i \leftarrow CO^d[r_i].\mathsf{communicate}(i, \ell s_i)$;
    $\ell s_i \leftarrow \delta_i(\ell s_i, view_i)$;
    if $\mathsf{decision}(\ell s_i)$ **then** $\mathsf{dec\_val}(\ell s_i)$ **end if**
end loop.

Figure 1: Iterated model

2.1 Communication object

The communication objects $CO^d[1]$, $CO^d[2]$, etc., of an execution are parameterized by a solo-dimension d, $1 \leq d \leq n$. As previously indicated, an object $CO^d[r]$ contains a set of pairs, one per process. Each pair (i, v) is such that i is a process index and v the value communicated by p_i, and $CO^d[r]$ contains at most one pair per process.

Definition The behavior of every object CO^d is defined as follows. Considering an execution during which each of the n processes $\{p_1, \ldots, p_n\}$ accesses the object (at most once) using its local state ℓs_i as input, one can represent this execution by an *ordered partition*, i.e., a tuple of non-empty sets $(P_1, \ldots, P_z)$ such that (1) for any distinct $i, j \in \{1, \ldots, z\}$: $P_i \cap P_j = \emptyset$, and (2) $\bigcup_{i=1}^{z} P_i = \{p_1, \ldots, p_n\}$. From an operational view, the ordered partition $(P_1, \ldots, P_z)$ describes the sequence of concurrent accesses to the object CO^d.

The behavior of CO^d is defined from a *d-ordered partition*, where a d-ordered partition is an ordered partition $(\pi_1, \ldots, \pi_{z'})$ such that $0 \leq |\pi_1| \leq d$ (the size of the first set of the partition can be 0 and cannot exceed d). More precisely, the d-ordered partition $(\pi_1, \ldots, \pi_{z'})$ associated with CO^d is:

- If $|P_1| > d$: $(\pi_1, \ldots, \pi_{z'}) = (\emptyset, P_1, \ldots, P_z)$, and
- If $|P_1| \leq d$: $(\pi_1, \ldots, \pi_{z'}) \in \{(\emptyset, P_1, \ldots, P_z), (P_1, \ldots, P_z)\}$.

$(\pi_1, \ldots, \pi_{z'}) = (P_1, \ldots, P_z)$ captures the cases where, initially, d (or less) processes execute solo. In the other cases we have $(\pi_1, \ldots, \pi_{z'}) = (\emptyset, P_1, \ldots, P_z)$, because initially either too many processes execute concurrently (first item), or, while no more than d processes execute concurrently, none of them executes solo.

We are now in oder to define the values $view_i$, $1 \leq i \leq n$, obtained by the processes when the behavior of CO^d is represented by the *d-ordered partition* $(\pi_1, \ldots, \pi_{z'})$ are defined as follows:

$$(i \in \pi_1) \Rightarrow (view_i = \{(i, \ell s_i)\}), \quad \text{and}$$
$$(x > 1 \wedge i \in \pi_x) \Rightarrow \big(view_i = \{(j, \ell s_j) \ : \ j \in \pi_y \wedge y \leq x\}\big).$$

This means that the view of each process p_i belonging to π_1 (where $0 \leq |\pi_1| \leq d$) contains only its own contribution, namely the pair (i, ls_i). Differently, the view of a process p_i in π_x, where $x > 1$, contains all the pairs $(j, \ell s_j)$ deposited in CO^d by the processes p_j of the sets π_y such that $y \leq x$. Thus, each process of π_1 appears as executing solo, while each other process of a set p_x, $x \neq 1$, sees the contributions provided (a) by all the processes p_i belonging to the "previous" sets π_y ($y < x$), and (b) by all the processes from its "concurrency" set π_x. (The immediate snapshot object described in [4] implements CO^d for $d = 1$.)

Object properties Given an object CO^d, the next properties follows from its definition.

- Solo execution upper bound. $0 \leq |\{i \text{ such that } |view_i| = 1\}| \leq d$.
- Self-inclusion. $\forall\, i : (i, -) \in view_i$.
- Containment. $\forall\, i, j : \big((|view_i| \leq |view_j|) \wedge |view_j| > 1)\big) \Rightarrow (view_i \subseteq view_j)$.

2.2 Examples of communication objects

Considering a system of $n = 3$ processes, this section describes two communication objects, corresponding to the cases $d = 1$, and $d = n - 1 = 2$. (Their aim is also to show connection between these objects and topology.)

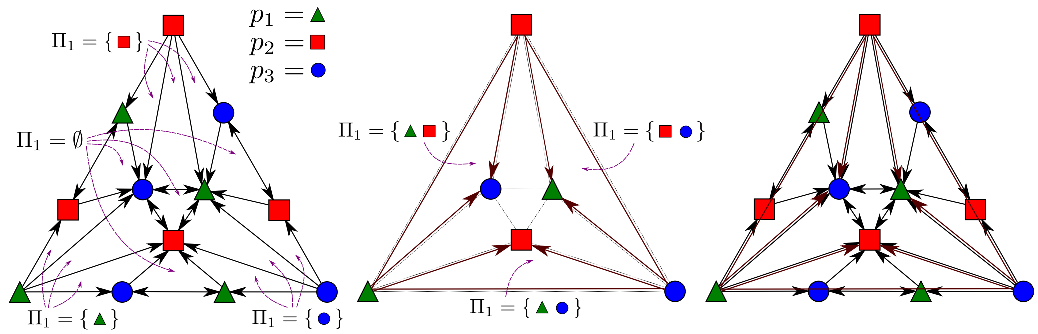


Figure 2: All possible executions for 3 processes

Object $CO^1[r]$ All the possible behaviors of $CO^1[r]$ that can occur are described on the left of Figure 2. An arrow from p_i to p_j means that the set $view_j$ (obtained by p_j when it returns from the invocation $CO^1[r].\mathsf{communicate}(j, -)$) is such that $(i, \ell s_i) \in view_j$. On the contrary, the absence of an arrow from p_i to p_j means that $(i, \ell s_i) \notin view_j$. In the topology parlance, the internal triangles are simplexes defining the possible subdivision of the (external triangle which defines the) complex associated with the execution at the beginning of round r.

The possible sets π_1 that can appear during an execution of $CO^1[r]$ are indicated for each small triangle (simplex) on the figure at the left. To simplify the notation, let $v_i = \ell s_i[r-1]$. As an example, the small triangle as the center corresponds to the case where $\pi_1 = \emptyset$ and $view_1 = view_2 = view_3 = \{(1, \ell v_1), (2, v_2), (3, v_3)\}$. For the three triangles at the bottom of the figure at the left, we have the following:

- Small triangle at the left side: $\pi_1 = \{p_1\}$, $view_1 = \{(1, v_1)\}$, $view_3 = \{(1, v_1), (3, v_3)\}$, and $view_2 = \{(1, v_1), (2, v_2), (3, v_3)\}$.
- Small triangle in the middle: $\pi_1 = \emptyset$, $view_1 = view_3 = \{(1, v_1), (3, v_3)\}$, and $view_2 = \{(1, v_1), (2, v_2), (3, v_3)\}$.
- Small triangle on the right side: $\pi_1 = \{p_3\}$, $view_3 = \{(3, v_3)\}$, $view_1 = \{(1, v_1), (3, v_3)\}$, and $view_2 = \{(1, v_1), (2, v_2), (3, v_3)\}$.

It is easy to see that the previous iterated computation model, where the communication objects are instantiated with $d = 1$, is nothing more than the *iterated immediate snapshot* model introduced in [4]. It has been shown in [5] that, from a task solvability point of view, this model is equivalent to the base wait-free asynchronous read/write model [17].

Object $CO^2[r]$ The possible behaviors of $CO^2[r]$ are represented on the right side of Figure 2. The new behaviors added to the ones of $CO^1[r]$ are represented in the middle of Figure 2 (the figure at the right is consequently the "addition" to the figure at the left of the possible behaviors described in the middle).

The new additional values for π_1 are described on the figure in the middle. The case $\pi_1 = \{1, 3\}$ that appears at the bottom of the figure represents the execution in which each of p_1 and p_3 executes as if it was alone: none of them sees the pair value communicated by the other. Differently p_3 sees both of them. Hence, this triangle represents the additional execution where $view_3 = \{(3, v_3)\}$, $view_1 = \{(1, v_1)\}$, and $view_2 = \{(1, v_1), (2, v_2), (3, v_3)\}$.

It is easy to see that this model is weaker than the base wait-free asynchronous read/write model: in the execution corresponding to the bottom triangle where $\pi_1 = \{1, 3\}$, none of p_1 and p_3 "writes" its pair before the other. More generally, if $d = n$, the object $CO^n[r]$ gives an account wait-free message-passing executions where, due to message asynchrony and process crashes, it is possible that an arbitrary number of processes do not receive messages from the other processes.

2.3 A spectrum of solo models

It follows from their definition that CO^d is stronger (more constraining) than CO^{d+1} in the sense that the subdivided complex of CO^d is included the one of C^{d+1}. From an operational point of view, this means that CO^d includes "more synchrony" than CO^{d+1}.

The d-solo model The generic framework described in Figure 2 instantiated with CO^d objects is called the *d-solo model*. It is denoted $\mathcal{ACS}^d_{n,n-1}$(ASC stands for Asynchronous Concurrent Solo) where the first subscript denotes the total number of processes, while the second subscript denotes the upper bound on the number of processes allowed to crash.

Hierarchy of d-solo models Let $A \succeq_T B$ mean that any task that can be solved in the model B can be solved in the model A, and $A \simeq_T B \stackrel{def}{=} (A \succeq_T B) \wedge (B \succeq_T A)$.

Let $\mathcal{ARW}_{n,n-1}$ denote the base wait-free (asynchronous) read/write model. It follows from the fact that (for task solvability) the IIS model and $\mathcal{ARW}_{n,n-1}$ have the same computability power [5], and IIS is nothing more than $\mathcal{ACS}^1_{n,n-1}$, that we have $\mathcal{ARW}_{n,n-1} \simeq_T \mathcal{ACS}^1_{n,n-1}$.

Let $\mathcal{AMP}_{n,n-1}$ denote the classical (non-iterated) message-passing system where up to $(n-1)$ processes may crash. As all processes except one may crash and communication is asynchronous (hence messages can be arbitrarily delayed), the tasks that can be solved in $\mathcal{AMP}_{n,n-1}$ are the tasks that can be wait-free solved without communication. But, this set of tasks is exactly the set of tasks that can be solved in $\mathcal{ACS}^n_{n,n-1}$. Hence, $\mathcal{ACS}^n_{n,n-1} \simeq_T \mathcal{AMP}_{n,n-1}$.

It follows from the definition of the communication objects CO^d and CO^{d+1} that any task solvable in $\mathcal{ACS}^{d+1}_{n,n-1}$ is solvable in $\mathcal{ACS}^d_{n,n-1}$. We have consequently the following hierarchy of models:

$$\mathcal{ARW}_{n,n-1} \simeq_T \mathcal{ACS}^1_{n,n-1} \succeq_T \ldots \succeq_T \mathcal{ACS}^d_{n,n-1} \succeq_T \ldots \succeq_T \mathcal{ACS}^n_{n,n-1} \simeq_T \mathcal{AMP}_{n,n-1}.$$

We will see in Section 5 that $A \succeq_T B$ can be replaced by $A \succ_T B$ (all the tasks solvable in B are solvable in A, and there is at least one task solvable in A and not in B).

3 Colorless Tasks and the d-Solo Model

This section focuses on colorless tasks that can be solved in the d-solo model. After having defined colorless tasks it shows that, for these tasks, one can use a restricted form of the algorithm in Figure 1. It then, introduces the notions of a (d, R)-subdivision task and a (d, R)-agreement task. (More topology notions in Appendix A.)

3.1 Colorless tasks

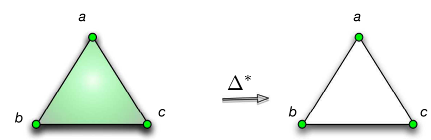
A colorless task is a special kind of task where the processes cannot use their ids during the computation. This implies that the task specification is not in terms of ids. A colorless task specifies which sets of values are valid input configurations, and which are valid output decisions, but not which value is assigned to which process. Thus, a process may adopt the input value or the output value of another process.

Formally, a *colorless task* is a triple $(\mathcal{I}^*, \mathcal{O}^*, \Delta^*)$, where $\mathcal{I}^*$ is a colorless *input complex*, $\mathcal{O}^*$ is a colorless *output complex*, and $\Delta^* : \mathcal{I}^* \to 2^{\mathcal{O}^*}$ is a *carrier map*. A colorless complex is a family of sets, over some basic set of values, such that if a set is in the complex, then all its subsets are also in the complex. A set in the complex is called a *simplex*. Simplexes of size 1, are called *vertices*, and of size 2, *edges*. Indeed, a graph is a 1-dimensional complex. In the case of a colorless complex, a vertex is just a value, either an input or an output value, while in a colored complex, a vertex is a pair of values, one is a process id, and the other is an input our output value. If σ is an input simplex in $\mathcal{I}^*$, the carrier map $\Delta^*(\sigma)$ is a subcomplex of $\mathcal{O}^*$ satisfying *monotonicity:* $\forall \sigma, \sigma' \in \mathcal{I}^* \ : \ \Delta^*(\sigma \cap \sigma') \subseteq \Delta^*(\sigma) \cap \Delta^*(\sigma')$.

Operationally, the meaning of a colorless task is the following. If $\sigma \in \mathcal{I}^*$, then the processes can start an execution with input values from σ; different processes may propose the same vertex or different vertices from σ. Processes eventually decide (not necessarily distinct) vertices that belong to the same output simplex $\tau \in \mathcal{O}^*$, such that $\tau \in \Delta^*(\sigma)$. If the system consists of n processes, then the processes can start with at most n different input values, and hence, processes will never start on a simplex σ of $\mathcal{I}$ of dimension greater than $n-1$ (the dimension of σ is $|\sigma|-1$). Thus, for n processes, only the simplexes of $\mathcal{I}$ of dimension $\leq n-1$ are relevant, i.e., the $n-1$ *skeleton* of $\mathcal{I}$, denoted $\text{Skel}^{n-1}\mathcal{I}$. For example, in a system of two processes, $n = 2$, only the 1-skeleton of $\mathcal{I}$ is of interest, which is the graph consisting of the vertices and 1-simplices of $\mathcal{I}$.

The most famous colorless task is k-*set agreement*. Every process proposes a value, from some base set of possible input values, D. Each process that does not crash has to decide a value (termination), such that a decided value is a proposed value (validity), and at most k different values are decided (agreement). This may be called the $|D|$-dimensional version of set agreement. The task version for n processes is trivial if $|D| \leq k$, and wait-free unsolvable in a read/write system if $|D| > k$, provided $n > k$ [4, 22, 32].

In the figure on the right the 2-set agreement colorless task is illustrated, over 3 possible input values, $D = \{a, b, c\}$. The input complex $\mathcal{I}^*$ consists of all sets over D, while the output complex $\mathcal{O}^*$ consists of all sets over D of size at most 2, which is the boundary of $\mathcal{I}^*$, denoted $\partial\mathcal{I}^*$. The carrier map sends a simplex σ in $\mathcal{I}^*$ to $\Delta^*(\tau)$, all subsets of σ in $\mathcal{O}^*$.



$\mathcal{I}^* = \sigma^2$ $\quad$ $\mathcal{O}^* = \partial\sigma^2$

3.2 Colorless algorithms

A *colorless algorithm* is an algorithm in the form of Figure 1, but where the local computation made by δ_i in line (4) is very restricted. Although a colorless algorithm is not as powerful as an algorithm with no restrictions, it simplifies that exposition, and in the full version we show that they can solve a similar class of colorless tasks.

Informally, in a colorless algorithm processes behave in an anonymous[1] way: they never use their ids in the computation. In each round, processes consider the shared memory as if it is a set. In each round, a process deposits its input in the set, and gets back a view of the contents of the set. If two processes deposit the same value in the set, only one copy is stored. When a process gets back a set of values, there is no information of which process deposited which value. A process "forgets" which is its own value in the set. The set of values that a process receives at the end of a round, becomes its input to the next round.

Formally, in an execution, the initial local state of a process p_i is a vertex v_i of $\mathcal{I}^*$, and is assigned in line 1 to ℓs_i. Furthermore, the set of all initial states v_i (not necessarily distinct) is a simplex σ of $\mathcal{I}^*$. We may write, $\sigma = \{\ell s_1[0], \ldots, \ell s_n[0]\}$, where $\ell s_i[0]$ denotes the initial value of ℓs_i. Notice that $|\sigma|$ may be less than n because different processes may start with the same input value.

The local transition δ_i eliminates process ids. Namely, during any round r and for any process p_i, if we denote by $\ell s_i[r]$ the value of ℓs_i at the end of round r, in line 3 of the algorithm, $view_i$ is assigned the value returned by the invocation $CO^d[r].\mathsf{communicate}(i, \ell s_i[r-1])$, and this value is a set of pairs $\{(i_1, \ell s_{i_1}[r-1]), \ldots, (i_k, \ell s_{i_k}[r-1])\}$ that includes ids $i_1, \ldots, i_k$, but when the function δ_i is applied to this set it returns a set $\sigma_i^r = \{\ell s_{i_1}[r-1], \ldots, \ell s_{i_k}[r-1]\}$. We assume every process executes the same number of rounds, $R \geq 0$, and in the last round, produces an output value $\mathsf{dec\_val}(\ell s_i)$ (all processes use the same function $\mathsf{dec\_val}$).

For an R round colorless algorithm in the d-dimensional model, the *algorithm complex* is defined as follows. For each input simplex $\sigma \in \mathcal{I}^*$, the subcomplex $\mathcal{P}^*(\sigma)$ represents the executions r where all processes start with inputs from σ (at least one process starts with each of the vertices in σ). Moreover, in the algorithm complex for the d-dimensional model we do not want to include the $(d-1)$-dimensional model, so we consider only runs where the processes that in a round see more than one process, they see at least $d+1$ processes. The complex $\mathcal{P}^*(\sigma)$ contains a top dimensional simplex $\tau = \{\ell s_i\}$ for each such R round execution of the algorithm starting in σ, where the vertices ℓs_i of τ are the values of $\ell s_i[r]$ at the end of this execution, for each process p_i (without repetitions, as the simplex is a set). The complex $\mathcal{P}^*$ is the union of $\mathcal{P}^*(\sigma)$ over all $\sigma \in \mathcal{I}^*$. It is easy to prove that $\mathcal{P}^*(\cdot)$ is a strict carrier map from $\mathcal{I}^*$ to the algorithm complex $\mathcal{P}^*$.

For example, if $|\sigma| = 1$, say $\sigma = \{v\}$, then $\mathcal{P}^*(\sigma)$ consists of only one vertex: every process starts with the same input value, v, and in each runs sees always the same set, $\{v\}$, so after executing R rounds, every process decides the same value, v'. Thus, $\mathcal{P}^*(\sigma) = \{v\}$. We will explain the significance of the next lemma later on, when we discuss subdivisions.

[1]The name "colorless" comes from previous papers. In a colored complex each vertex of a simplex has a distinct process id, while in a colorless complex vertices do not have id labeling. To stress this a colorless complex is denoted with a $*$ superscript, as in $\mathcal{K}^*$.

Lemma 1 *Consider a 1-round colorless algorithm and an input simplex $\sigma \in \mathcal{I}^*$. The simplexes of $\mathcal{P}^*(\sigma)$ are of the form $\tau = \{\tau_1, \ldots, \tau_z\}$, where each $\tau_i \subseteq \sigma$, and there is an $l, 0 \le l \le d$ such that (1) for all $i, 0 \le i \le l$, $|\tau_i| = 1$, so $\cup_{0 \le i \le l} \tau_i$ is a face σ' of σ, (2) for all $j, l < j \le z$, $\sigma' \subsetneq \tau_j$, and (3) for all $j, l < j \le z-1$, $\tau_j \subsetneq \tau_{j+1}$.*

Proof Each simplex τ associated to a sequence of faces $F_0, \ldots, F_z$ and to an integer l as above, corresponds to an allowed behavior for the object CO^d. Namely the one represented by the d-ordered set partition $(\bigcup_{i=1}^{l} F_i, F_{l+1} \setminus \bigcup_{i=1}^{l} F_i, \ldots, F_z \setminus \bigcup_{i=1}^{z-1} F_i)$. Since $0 \le l \le d$, the set $\bigcup_{i=1}^{l} F_i$ contains at most d values, moreover (2) and (3) imply that the sets of the partition are pairwise disjoint

Reciprocally, if $(\pi_1, \ldots, \pi_z)$ corresponds to an allowed behavior for the object CO^d, then one can build a sequence of faces $(F_1, \ldots, F_{z+l-1})$ by choosing $l = |\{\ell s_i, p_i \in \pi_1\}|$, $\{F_1, \ldots, F_l\} = \{\{\ell s_i\}, p_i \in \pi_1\}$ and $\forall i, l < i < z+l-1 \,:\, F_i = \bigcup_{j=1}^{i+|\pi_1|-1} \ell s_j$. The properties of the communication object CO^d ensure that $l \le |\pi_1| \le d$, but also that the properties (2) and (3) hold. Consequently the simplex τ whose vertices are the faces $F_i, 1 \le i \le z+l-1$, belongs to $\mathcal{P}^*(\sigma)$. $\square_{Lemma\ 1}$

If $\mathcal{P}^*(\cdot)$ is a carrier map from $\mathcal{I}^*$ to the algorithm complex $\mathcal{P}^*$, and dec_val is a simplicial map from $\mathcal{P}^*$ to $\mathcal{O}^*$, we say that dec_val *is carried by* Δ^* if for each $\sigma \in \mathcal{I}^*$ and each $\tau \in \mathcal{P}^*(\sigma)$, the simplex dec_val$(\tau)$ belongs to $\Delta^*(\sigma)$.

Lemma 2 *If the colorless task $(\mathcal{I}^*, \mathcal{O}^*, \Delta^*)$ is solvable by a colorless algorithm then there exists an algorithm complex $\mathcal{P}^*$, and a simplicial map* dec_val *from $\mathcal{P}^*$ to $\mathcal{O}^*$ that is carried by Δ^*.*

Proof The output value decided by a process in line 5 is based on ℓs_i, which is a set of values, with no process ids. If the r-round colorless algorithm solves the colorless task $(\mathcal{I}^*, \mathcal{O}^*, \Delta^*)$, at the end of the r-th round, processes have to decide an output value, by executing dec_val(ℓs_i) in line 5. The result of dec_val(ℓs_i) is a vertex in $\mathcal{O}^*$. Different processes may decide different vertices as long as they belong to the same simplex τ of $\mathcal{O}^*$. Moreover, if $\sigma \in \mathcal{I}^*$ is the input simplex of the execution, the output simplex τ must be in $\Delta^*(\sigma)$, to satisfy the task's specification. $\square_{Lemma\ 2}$

3.3 (d, R)-Subdivision and (d, R)-agreement tasks

The (d, R)-subdivision task Which is the simplest task a colorless algorithm can solve in the d-dimensional model? It is the task solved when each process executes R rounds, then stops, and its decision function is the identity! Namely, dec_val$(\ell s_i) = \ell s_i$ i.e. a process decides the set of values $\ell s_i[R]$ it retrieves from the communication object during the R^{th} round. Given any input complex $\mathcal{I}^*$ and any integer $R \ge 0$, we call this task the (d, R)-*subdivision task* over $\mathcal{I}^*$. The output complex $\mathcal{O}^*$ of this task is of course equal to the algorithm complex $\mathcal{P}^*$, with the simplicial map dec_val being the identity.

For the carrier map, $\Delta^*(\sigma)$ includes all simplexes τ that correspond to executions starting in σ, i.e., $\Delta^*(\sigma) = \mathcal{P}^*(\sigma)$. In particular, for $R = 0$, $\mathcal{I}^* = \mathcal{O}^*$, and Δ^* is the identity carrier map, which sends a simplex σ to the complex consisting of σ and all its faces (which we often denote by σ, abusing notation).

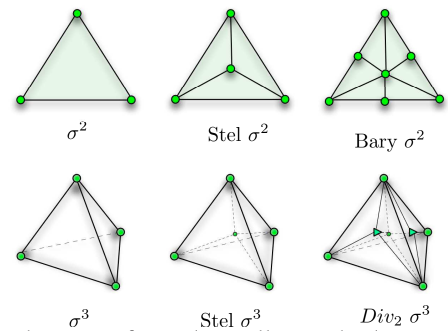
By definition, the (d, R)-subdivision task over $\mathcal{I}^*$ is solvable in the d-dimensional model, and moreover, by a colorless algorithm. In fact, it is the basic building block to solve every other colorless task, as shown in Theorem 1. We will justify the name "subdivision task" when we see how to specify the task without resorting to executions of some model in Section 3.4.

The (d, R)-agreement task When the vertices of $\mathcal{I}^*$ are points in Euclidean space, the (d, R)-subdivision task can be used directly to solve a task that we call (d, R)-*agreement task* over $\mathcal{I}^*$, which is defined combinatorially in Section 5. In the (d, R)-subdivision task, processes propose sets of values in each round. We can encode such a set of values as its barycenter b, and then the process can directly propose b. We shall see in Section 5, that, although both tasks are essentially the same, when we work with barycenters processes compute output values distant of less than ϵ from the polytope formed by the values decided by processes that run solo, and we can make ϵ as small as we want, by choosing a large enough value of R.

Operationally, the (d, R)-agreement task over $\mathcal{I}^*$ is defined as follows. Processes execute R rounds of a colorless algorithm in the d-dimensional model. In each round r, each process p_i computes its value $\ell s_i[r]$ that will be the input to the next round, in line 4 of the algorithm, by taking the barycenter of the values that it gets back from the object in line 3. The barycenter computed in round R is the output of of the process.

3.4 The structure of colorless algorithms

The structure of a colorless complex is explained in terms of *subdivisions* (Appendix A). Examples of subdivisions of a simplex are illustrated on the figure that follows at the right of the page.



Perhaps the simplest subdivision is the *stellar* subdivision. Given a complex (abusively denoted σ^m) consisting of an m-simplex $\sigma^m = \{s_0, \ldots, s_m\}$ and all its faces, the complex $\text{Stel}(\sigma^m, b)$ is constructed by taking a *cone* with apex b over the boundary complex $\partial\sigma^m$ The *barycentric* subdivision, Bary σ^m, is perhaps the most widely used in topology. A simplex τ is in Bary σ^m if and only if there exists a sequence $\sigma_0 \subset \ldots \subset \sigma_z$ of faces of σ^m, and the set of vertices of τ is the set of the barycenters of the these faces, denoted $\hat{\sigma}_i, 0 \leq i \leq z$.

For the d-solo models, we need to define a family of subdivisions that goes from the stellar to the barycentric subdivision. The *d-dimensional subdivision* of a complex $\mathcal{K}$ denoted $\text{Div}_d\ \mathcal{K}$, is the barycentric subdivision of $\mathcal{K}$ relative to $\text{Skel}^{d-1}\mathcal{K}$. Intuitively, we do not subdivide $\text{Skel}^{d-1}\mathcal{K}$ because we consider executions where up to d processes run solo, they get their own view in an invocation of a CO^d object. See the construction of Figure 3 and Appendix A. As usual, the *R-iterated d-dimensional subdivision*, $\text{Div}_d^R\ \mathcal{K}$, is obtained by repeating the subdivision process R times.

$\text{Div}_d\ \text{Skel}^{d-1}\sigma^m \leftarrow \text{Skel}^{d-1}\sigma^m$; % each vertex is labeled by its name
for k **from** d **to** m **do** % Construct $\text{Div}_d\ \text{Skel}^k\sigma^m$ %
    for each simplex σ^k in σ^m **do**
        insert a vertex b in the barycenter of σ^k; % this barycenter is labeled with the set of vertices of σ_k
        construct the cone with apex at b over $\text{Div}_d\ \partial\sigma^k$; % over the already subdivided boundary of σ^k %
        add the cone to $\text{Div}_d\ \text{Skel}^k\sigma^m$
    end for loop
end for loop.

Figure 3: Constructing the subdivision $\text{Div}_d\ \sigma^m$ of a simplex σ^m for the d-solo model

The next lemma follows from the fact that the construction of Div_d in Figure 3 corresponds exactly to the description given in Lemma 1, and the fact in the system there are n processes, so they can start with at most n different input values (so only the input simplexes in $\mathcal{I}^*$ of dimension at most $n-1$ are relevant).

Lemma 3 *If $\mathcal{P}_R^*$ is the R-round algorithm complex of a colorless algorithm in the d-solo model with input complex $\mathcal{I}^*$, then $\mathcal{P}_R^*$ is an R-iterated, d-dimensional subdivision of the $n-1$ skeleton of $\mathcal{I}^*$.*

Returning to the (d, R)-subdivision task, we can now justify its name, simply by recalling that its output complex is equal to the algorithm complex:

Lemma 4 *The (d, R)-subdivision task over $\mathcal{I}^*$ for n processes is a triple $(\mathcal{I}^*, \mathcal{O}^*, \Delta^*)$, where $\mathcal{O}^*$ is the R-iterated, d-dimensional subdivision of the $n-1$ skeleton of $\mathcal{I}^*$, and Δ^* is equal to the corresponding subdivision carrier map.*

4 What Can Be Computed in the Presence of Concurrent Solo Executions?

This section presents a characterization of the colorless tasks that can be solved in each one of the d-solo models.

4.1 The general case

Consider an r round colorless algorithm that solves the colorless task $(\mathcal{I}^*, \mathcal{O}^*, \Delta^*)$. At the end of the r-th round, processes have to decide an output value, by executing $\mathsf{dec\_val}(\ell s_i)$ in line 5. The result of $\mathsf{dec\_val}(\ell s_i)$ is a vertex in $\mathcal{O}^*$, and different processes may decide different vertices as long as they belong to the same simplex of $\mathcal{O}^*$. This means that $\mathsf{dec\_val}$ is a simplicial map from $\mathcal{P}_r^*$ to $\mathcal{O}^*$. Moreover, $\mathsf{dec\_val}$ is carried by Δ^*, in the sense that for $\sigma \in \mathcal{I}^*$,

$$\mathsf{dec\_val}(\mathcal{P}_r^*(\sigma)) \subseteq \Delta^*(\sigma),$$

which means that for any input simplex σ, any r round execution ends in a simplex τ of $\mathcal{P}_r^*$, and the decision that the processes make in τ, form an output simplex $\mathsf{dec\_val}(\tau)$ of $\mathcal{O}^*$. This output simplex $\mathsf{dec\_val}(\tau)$ must be in $\Delta^*(\sigma)$, to satisfy the task's specification.

Theorem 1 *The colorless task $\mathcal{T}^* = (\mathcal{I}^*, \mathcal{O}^*, \Delta^*)$ is solvable with n processes in the d-solo model by a colorless algorithm if and only if there is an $R \geq 0$ and a simplicial map $\phi : Div_d^R\ Skel^{n-1}\mathcal{I}^* \to \mathcal{O}^*$ carried by Δ^*.*

Proof (Sketch) If $\mathcal{T}^*$ is solvable in the d-solo model then there exists the simplicial map $\phi : \text{Div}_d^R \text{ Skel}^{n-1}\mathcal{I}^* \to \mathcal{O}^*$ carried by Δ^*, by Lemma 2 and Lemma 3.

Conversely, notice that Lemma 4 says that the (d, R)-subdivision task over $\mathcal{I}^*$ for n processes has as output complex the R-iterated, d-dimensional subdivision of the $n-1$ skeleton of $\mathcal{I}^*$. By definition there is a colorless algorithm that solves this task. Thus, when each process p_i runs this algorithm, on an input simplex σ of $\mathcal{I}^*$, it gets as output a vertex v_i in $\text{Div}_d^R\text{Skel}^{n-1}\mathcal{I}^*$. Then p_i produces as output to $\mathcal{T}^*$ the value $\phi(v_i)$. Since the outputs v_i span a simplex of $\text{Div}_d^R\text{Skel}^{n-1}\mathcal{I}^*$, the outputs $\phi(v_i)$ span a simplex τ of $\mathcal{O}^*$, and τ is in $\Delta^*(\sigma)$. $\square_{Theorem\ 1}$

4.2 More developments and the case $d = 1$

Our main characterization theorem is about which colorless tasks $\mathcal{T}^* = (\mathcal{I}^*, \mathcal{O}^*, \Delta^*)$ are solvable with n processes in the d-solo model by a colorless algorithm. There are colorless tasks that are not solvable by a colorless algorithm, yet they are solvable (with n processes in the d-solo model, $d > 1$). To explain this we need to recall that the structure of an algorithm complex in the wait-free iterated model (our model with $d = 1$) is a chromatic subdivision of the input chromatic complex [23]. A chromatic subdivision is constructed similarly to the algorithm in Figure 3, taking cones repeatedly, except that now we have to take *chromatic cones* ([7], Definition 37). The *d-dimensional chromatic subdivision* of a chromatic complex $\mathcal{I}$ holds the $(d-1)$-skeleton of $\mathcal{I}$ fixed. It corresponds to an algorithm complex of a one round algorithm in the d-dimensional model. In Figure 4 there is an example of an input complex $\mathcal{I}^*$ and a colorless algorithm complex for 1 and 2 rounds, in the $d = 2$ solo model (hence edges are not subdivided). In contrast, when the algorithm is chromatic, each vertex must be labeled with a value and a process id. In Figure 5 we see that even each input simplex σ is chromatic (labeled with ids), and the algorithm complex for $d = 2$ is also chromatic, and is constructed by inserting a triangle (instead of a vertex) in the place of the barycenter, and taking the chromatic cone. In the triangle, each vertex is labeled with an id, and the values the process sees after one round.

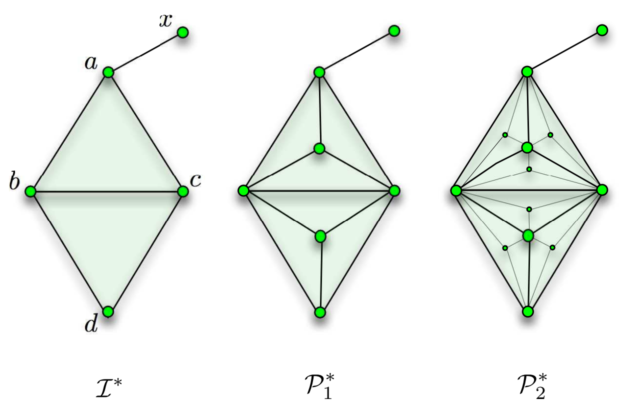


Figure 4: Example of a 1-round and 2-round algorithm complex, $\mathcal{P}_1^*$, $\mathcal{P}_2^*$, over an input complex $\mathcal{I}^*$ which as input values a, b, c, d, x, in the $d = 2$ solo model

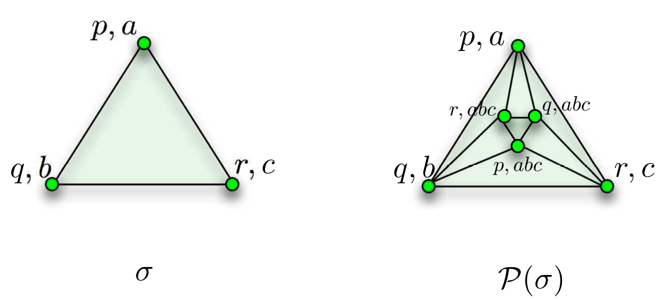


Figure 5: One chromatic input simplex σ and the one round chromatic algorithm complex

Figure 6 presents a colorless task, $\mathcal{T}_1^*$, where processes can start with any input value from the set $\{a, b, c\}$. The carrier map Δ^* states that $\Delta^*(a) = a$, $\Delta^*(b) = b$, $\Delta^*(c) = c$, $\Delta^*(e) = e$ for each edge e in $\mathcal{I}^*$. For the simplex σ^2 consisting of all three values, $\Delta^*(\sigma^2)$ allows the processes to decide in any of the simplexes of $\mathcal{O}^*$. This task is not solvable by a colorless algorithm in the $d = 2$ solo model by a colorless algorithm. However, it is solvable by a 1-round algorithm in the $d = 2$ solo model, if processes use their ids.

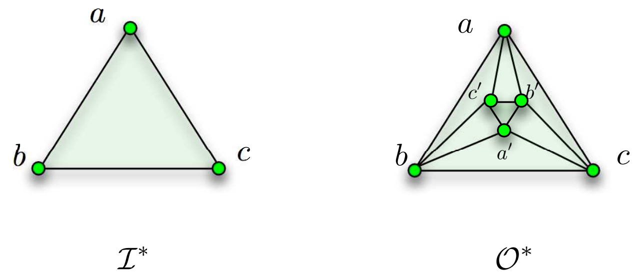


Figure 6: A colorless task $\mathcal{T}_1^*$ that is not solvable by a colorless algorithm in the model with $d = 2$

Figure 7 presents $\mathcal{T}_{imp}^*$, a similar colorless task, where processes can start with any input value from the set $\{a, b, c\}$, and is the same on the boundary, the carrier map Δ^* states that $\Delta^*(a) = a$, $\Delta^*(b) = b$, $\Delta^*(c) = c$, $\Delta^*(e) = e$ for each edge e in $\mathcal{I}^*$. However, but the output complex is different, and for the simplex σ^2 consisting of all three values, $\Delta^*(\sigma^2)$ allows more simplexes. This task is not solvable by any algorithm in the $d = 2$ solo mode.

Figure 7: A colorless task $\mathcal{T}_{imp}^*$ that is not solvable by any algorithm in the model with $d = 2$

The task $\mathcal{T}_{imp}^*$ of Figure 7 illustrates the difficulty in using the Simplicial Approximation Theorem, or even the Relative Simplicial Approximation Theorem, as in previous papers such as [20], to relate the characterization of Theorem 1 to the existence of a continuous function. Notice that there is a continuous function f from $|\mathcal{I}^*|$ to $|\mathcal{O}^*|$ that respects the task specification of $\mathcal{T}_{imp}^*$ and moreover, f is simplicial on the 1 skeleton of $\mathcal{I}^*$, so the Relative Simplicial Approximation Theorem says there is a subdivision of $\mathcal{I}^*$ that does not modify its 1-skeleton, and where a simplicial map can be constructed to $\mathcal{O}^*$, however, the subdivision would subdivide internal edges that cannot be subdivided by an algorithm in the model with $d = 2$. We explore this issue further in a sequel paper. However, for $d = 1$, indeed every simplex is subdivided, and hence the diameter of every simplex is reduced each time a new subdivision is taken. Thus we can prove the following characterization, which is the same as the one in [20], except that it shows that colorless algorithms can be used without loss of generality, for colorless tasks, when $d = 1$.

Theorem 2 *The colorless task $\mathcal{T}^* = (\mathcal{I}^*, \mathcal{O}^*, \Delta^*)$ is solvable with n processes in the d-solo model, $d = 1$, if and only if either of the following equivalent conditions holds:*

1. *there exist a colorless algorithm that solves $\mathcal{T}^*$ with n processes in the d-solo model, $d = 1$;*
2. *there is an $R \geq 0$ and a simplicial map $\phi : Div_1^R\ Skel^{n-1}\mathcal{I}^* \to \mathcal{O}^*$ carried by Δ^*;*
3. *there is a continuous map $f : |Skel^{n-1}\mathcal{I}^*| \to |\mathcal{O}^*|$ carried by Δ^*, that is simplicial on $|Skel^0\mathcal{I}^*|$.*

Proof We are going to prove the theorem by proving the following implications $(0) \Leftarrow (1) \Leftarrow (2) \Leftarrow (0)$, where (0) $(0) \Leftarrow (1) \Leftarrow (2) \Leftarrow (3) \Leftarrow (0)$, where (0) means: the colorless task $\mathcal{T}^* = (\mathcal{I}^*, \mathcal{O}^*, \Delta^*)$ is solvable with n processes in the d-solo model, with $d = 1$. Although to give a more complete picture of our theoretical framework, we prove other implications as well.

Condition 1 implies Condition 0, because a colorless algorithm is a special case of general algorithm.

Condition 1 implies condition 2 follows directly from Lemma 2 and Lemma 3.

Condition 2 implies condition 1, because Lemma 4 says that the (d, R)-subdivision task over $\mathcal{I}^*$ for n processes has as output complex the R-iterated, d-dimensional subdivision of the $n - 1$ skeleton of $\mathcal{I}^*$. By definition there is a colorless algorithm that solves this task. Thus, when each process p_i runs this algorithm, on an input simplex σ of $\mathcal{I}^*$, it gets as output a vertex v_i in $\text{Div}_d^R\text{Skel}^{n-1}\mathcal{I}^*$. Then p_i produces as output to $\mathcal{T}^*$ the value $\phi(v_i)$. Since the outputs v_i span a simplex of $\text{Div}_d^R\text{Skel}^{n-1}\mathcal{I}^*$, the outputs $\phi(v_i)$ span a simplex τ of $\mathcal{O}^*$, and τ is in $\Delta^*(\sigma)$.

Now, we prove that the conditions 2 and 3 of the theorem are equivalent.

Now, condition 2 implies condition 3. A simplicial map $\phi : \text{Div}_d^R \text{ Skel}^{n-1}\mathcal{I}^* \rightarrow \mathcal{O}^*$ carried by Δ^* yields a continuous map $|\phi| : |\text{Div}_d^R \text{ Skel}^{n-1}\mathcal{I}^*| \rightarrow |\mathcal{O}^*|$ also carried by Δ^*, by simply extending ϕ linearly (see Section A). Since $|\text{Div}_d^R \text{ Skel}^{n-1}\mathcal{I}^*|$ is homeomorphic to $|\text{Skel}^{n-1}\mathcal{I}^*|$, and $\text{Div}_d^R \text{ Skel}^{d-1}\mathcal{I}^* = \text{Skel}^{d-1}\mathcal{I}^*$, there is a continuous map $f : |\text{Skel}^{n-1}\mathcal{I}^*| \rightarrow |\mathcal{O}^*|$ carried by Δ^*, that is simplicial on $|\text{Skel}^{d-1}\mathcal{I}^*|$.

Condition 3 implies condition 2. Assume we have a continuous map $f : |\text{Skel}^{n-1}\mathcal{I}^*| \rightarrow |\mathcal{O}^*|$ carried by Δ^* that is simplicial on $|\text{Skel}^{d-1}\mathcal{I}^*|$. By the Simplicial Approximation Theorem f has a simplicial approximation: there is an R together with a simplicial map $\varphi : \text{Div}_d^R\text{Skel}^n\mathcal{I}^* \rightarrow \mathcal{O}^*$ carried by Δ^*, with $f = \varphi$ on $|\text{Skel}^{d-1}\mathcal{I}^*|$. Since the two maps are homotopic relative to this skeleton, the homotopy is a straight line in the carrier of points not in the skeleton, we have that φ is carried by Δ^*. Thus, condition 3 implies condition 2.

Finally, assume there is a algorithm (not necessarily colorless) that solves $\mathcal{T}^* = (\mathcal{I}^*, \mathcal{O}^*, \Delta^*)$ with n processes in the d-solo model. We prove that condition 3 of the theorem holds. Namely, there is a continuous map $f : |\text{Skel}^{n-1}\mathcal{I}^*| \rightarrow |\mathcal{O}^*|$ carried by Δ^*, that is the identity on $|\text{Skel}^{d-1}\mathcal{I}^*|$. The argument is similar to Lemma 4.1 in [20], extended to the observation above, that an R round algorithm in the d-model has as algorithm complex the R-iterated d-dimensional chromatic subdivision of the chromatic version of the input complex $\mathcal{I}$, holding its $(d-1)$-skeleton fixed. $\square_{Theorem\ 2}$

5 (d, ϵ)-Solo Approximate Agreement and Strict Hierarchy of Models

We now study the properties of the (d, R)-agreement task of Section 3.3 in terms of a precision parameter ϵ, showing that this task can be solved in the d-solo model while it cannot be solved in the $(d+1)$-solo model.

The (d, ϵ)-solo approximate agreement problem (in short (d, ϵ)-SAA) is a generalization of the *ϵ-approximate agreement* problem [11]. The $(1, \epsilon)$-solo approximate agreement instance implies 2ϵ-approximate agreement. Assuming the input of each process is a point of the d-dimensional Euclidean space $\mathbb{R}^d$, (d, ϵ)-solo approximate agreement is defined by the following properties. (This definition is discussed in Appendix B.)

- Validity. Any output lies within the convex hull of the inputs.
- Agreement. There is a set of processes S, $1 \leq |S| \leq d$, such that any process p_i that is not in S decides a value o_i (point) such that the Euclidean distance between o_i and CH is at most ϵ, where CH is the convex hull of the points decided by the processes in S.
- Termination. If a process p_i does not crash, it decides a value.

It follows from this definition that up to d processes are allowed to decide any set of points within the convex hull (as an example each of them may decide the point it proposes). These processes define the set S, and intuitively, the values they decide are collectively "represented" by their convex hull CH. Finally, the values decided by the other processes are constrained by the values decided by the processes in S. A said in the Introduction, the convex hull CH is an attractor for the set of decided values.

The next theorem shows that, from a task solvability point of view, the d-solo model is stronger than the $(d+1)$-solo model. As (d, ϵ)-solo approximate agreement can be solved in the d-solo model (Lemma 5), it is then sufficient to show that this problem cannot be solved in the $(d+1)$-solo model (Lemma 6).

Lemma 5 *If the volume of any d-face of the input complex is less than V, and $R > \frac{\log(V)+\log(d!)-d\log(\epsilon)}{\log(d+1)}$, then the (d, R)-agreement task solves the (d, ϵ)-solo approximate agreement problem.*

The proof shows that the smallest height of any d-simplex after R subdivisions is less than ϵ, which entails that, for any $(d+1)$ distinct values decided in the same execution, one is closer than ϵ from the $(d-1)$-face (convex hull) formed by the d other values.

Proof The validity property comes directly from the fact that, the values decided in the (d, R)-agreement task are barycenters of a subset of the input values. The termination follows from the fact that any correct process decides in R rounds.

Let us consider $\mathcal{P}_R^*$ the R-round complex of the colorless algorithm. If we show that the minimal height of any d-face of $\mathcal{P}_R^*$ is less than ϵ then the agreement property of the (d, ϵ)-solo approximate agreement problem follows.

Since, during each subdivision and in each d-face, a cone is built over the boundary with apex at the barycenter, the volume of the d-faces is multiplied by $\frac{1}{d+1}$ during each subdivision. It follows that, after R subdivisions with $V \cdot (\frac{1}{d+1})^R < \frac{\epsilon^d}{d!}$, the volume of any d-face of $\mathcal{P}_R^*$ is less than $\frac{\epsilon^d}{d!}$.

Let h_{min}^d be the smallest height of a d-face σ of $\mathcal{P}_R^*$, it is the distance between a vertex v_d of σ and the $(d-1)$-face of σ with the largest volume V_{max}^{d-1}. The volume of σ is $V(\sigma) = \frac{1}{d} h_{min}^d \cdot V_{max}^{d-1} < \frac{\epsilon^d}{d!}$. Consequently either $h_{min}^d < \epsilon$ or $V_{max}^{d-1} < \frac{\epsilon^{d-1}}{(d-1)!}$. In the first case the smallest height of σ is less than ϵ. In the second case, if $d-1 = 1$ then $V_{max}^1 < \epsilon$ is the largest distance between two vertices of σ and then the smallest height of σ is less than ϵ. If $d-1 > 1$ then consider the smallest height h_{min}^{d-1} of $\sigma \setminus \{v_d\}$ whose length is the distance between a vertex v_{d-1} of $\sigma \setminus \{v_d\}$ and the face of $\sigma \setminus \{v_d\}$ with the largest volume V_{max}^{d-2}. Since h_{min}^{d-1} is smaller than the distance between v_{d-1} and its complementary face in σ, $h_{min}^{d-1} \geq h_{min}^d$. The volume of $\sigma \setminus \{v_d\}$ is $V_{max}^{d-1} = \frac{1}{d-1} h_{min}^{d-1} \cdot V_{max}^{d-2} < \frac{\epsilon^{d-1}}{(d-1)!}$, consequently either $h_{min}^d \leq h_{min}^{d-1} < \epsilon$ or we can iterate with $V_{max}^{d-2} < \frac{\epsilon^{d-2}}{(d-2)!}$. $\Box_{Lemma\ 5}$

Lemma 6 *For $d > 1$, if the domain of the possible inputs contains a simplex of dimension $(d-1)$ whose edge length is strictly greater than $2\epsilon(d-1)\sqrt{\frac{2(d-1)}{d}}$, then, for $n \geq d$, the $(d-1, \epsilon)$-solo approximate agreement problem is impossible to solve in $\mathcal{ACS}_{n,n-1}^d$.*

Proof The proof is by contradiction. Assuming that there is an algorithm A that solves $(d-1,\epsilon)$-AA in $\mathcal{ACS}_{n,n-1}^d$, let us consider its executions in which the processes p_{d+1}, ..., p_n crash before executing any step, and each process p_i, $1 \leq i \leq d$ retrieve only its own value from the CO^d objects in every round. As no more than d processes invoke A, there is a subset of executions in which the behavior of each process p_i, $1 \leq i \leq d$, is indistinguishable from a solo execution.

Hence, to show that $(d-1,\epsilon)$-AA cannot be solved in $\mathcal{ACS}_{n,n-1}^d$, let the values v_1, ..., v_d proposed to $(d-1,\epsilon)$-AA by the processes p_1, ..., p_d be d distinct vertices of a regular simplex of dimension $(d-1)$ whose the edge length α is such that $\alpha > 2\epsilon(d-1)\sqrt{\frac{2(d-1)}{d}}$ (this is possible since, by hypothesis, the domain of possible inputs contains such a simplex). In the following $dist(a, X)$ denotes the Euclidean distance between a and X, where a is a vertex, and X is a vertex, a polytope, or a hyperplane. It follows from the termination, agreement and validity properties of A that there exists a plane $\mathcal{P}$ of dimension $(d-2)$ such that $\forall\, i \in \{1, \ldots, d\} : dist(v_i, \mathcal{P}) \leq \epsilon$. This is a consequence of the agreement property. (Up to $d-1$ processes can decide any value. The convex hull of this set of values is a polytope P' of dimension $(d-2)$ or less and all the other processes decide a value distant of at most ϵ from P'.Since the dimension of P' is at most $d-2$, any $(d-2)$-plane $\mathcal{P}$ containing P' verifies the property.)

Remark that, since they are the vertices of a regular $(d-1)$-dimensional simplex, the points v_1, ..., v_d do not belong to the same plane of dimension $d-2$ (hence, the vectors $v_1v_2, \ldots v_1v_d$ are linearly independent).

Let N_i be a normalized vector, orthogonal to the hyperplane $\mathcal{P}_i$ containing the points $v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_d$. Then, $\forall i \in [2, d]$, $dist(v_i, \mathcal{P}_i) = |\langle N_i | v_1 v_i \rangle|$ (where $|\langle N_i | v_1 v_i \rangle|$ denotes the scalar product of the vector N_i by the vector v_1v_i).

Let B_i be the d balls of center v_i and radius ϵ. Since $\forall i,\ dist(v_i, P) \leq \epsilon$, there exists d vectors $\epsilon_1, \ldots, \epsilon_d$ such that the points $v_1' = v_1 + \epsilon_1, \ldots, v_n' = v_d + \epsilon_d \in B_1, \ldots, B_d$ and $\forall\, i,\ v_i' \in P$. The points $v_1', \ldots, v_d'$ are coplanar and the vectors $v_1'v_2', \ldots, v_1'v_d'$ are not independent, and consequently there exist $d-1$ reals $\lambda_2, \ldots, \lambda_d \neq 0, \ldots, 0$ such that $\sum_{i=2}^{d} \lambda_i v_1' v_i' = 0$.

Let λ_j be such that $|\lambda_j| = \max_i |\lambda_i|$. We have then

$$0 = \langle N_j | \sum_{i=2}^{d} \lambda_i v_1' v_i' \rangle, \tag{1}$$

$$= \langle N_j | \sum_{i=2}^{d} \lambda_i (v_1 v_i + \epsilon_i - \epsilon_1) \rangle, \tag{2}$$

$$= \lambda_j \langle N_j | v_1 v_j \rangle + \sum_{i=2}^{d} \lambda_i \langle N_j | (\epsilon_i - \epsilon_1) \rangle. \tag{3}$$

Thus,

$$\lambda_j \langle N_j | v_1 v_j \rangle = -\sum_{i=2}^{d} \lambda_i \langle N_j | (\epsilon_i - \epsilon_1) \rangle, \tag{4}$$

$$|\lambda_j \langle N_j | v_1 v_j \rangle| = |\sum_{i=2}^{d} \lambda_i \langle N_j | (\epsilon_i - \epsilon_1) \rangle|, \tag{5}$$

$$|\lambda_j| dist(v_j, \mathcal{P}_j) = |\sum_{i=2}^{d} \lambda_i \langle N_j | (\epsilon_i - \epsilon_1) \rangle|. \tag{6}$$

But

$$|\sum_{i=2}^{d} \lambda_i \langle N_j | (\epsilon_i - \epsilon_1) \rangle| \leq \sum_{i=2}^{d} |\lambda_i| (|\epsilon_i| + |\epsilon_1|), \tag{7}$$

$$\leq |\lambda_j| \times 2\epsilon \cdot (d-1), \tag{8}$$

which would imply that $dist(v_j, \mathcal{P}_j) \leq 2\epsilon \cdot (d-1)$, while, according to the definition of the points v_i, $dist(v_j, \mathcal{P}_j) = \alpha\sqrt{\frac{d}{2(d-1)}} > 2\epsilon \cdot (d-1)$. ($\alpha\sqrt{\frac{d}{2(d-1)}}$ is the height of a regular $d-1$-simplex of edge size α) This contradicts the existence of $\mathcal{P}$ and thus the existence of the algorithm A. $\Box_{Lemma\ 6}$

The next theorem follows from the previous Lemmas 5 and 6.

Theorem 3 *If the domain of the possible input values (a) is bounded and (b) contains a simplex of dimension d whose edge length is strictly greater than $2\epsilon d\sqrt{\frac{2d}{d+1}}$, then the (d,ϵ)-solo approximate agreement problem is solvable in $\mathcal{ACS}^d_{n,n-1}$but not in $\mathcal{ACS}^{d+1}_{n,n-1}$.*

6 Relating d-Set Agreement and (d,ϵ)-Solo Approximate Agreement

The d-set agreement (in short d-SA) problem [9] is defined as follows. Assuming that every process proposes a value, each process that does not crash has to decide a value (termination), such that a decided value is a proposed value (validity), and at most d different values are decided (agreement). Similarly to ϵ-approximate agreement which is a weakened version of consensus, (d,ϵ)-SAA is a weakened version of d-SA.

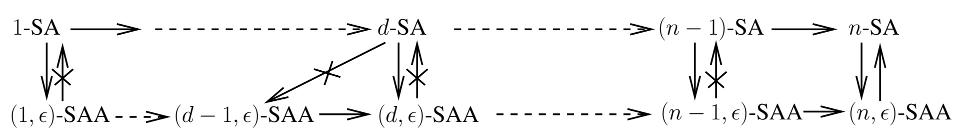


Figure 8: Relating d-SA with both (d,ϵ)-SAA and $(d-1,\epsilon)$-SAA

Considering the wait-free asynchronous message-passing model enriched with an algorithm solving d-SA (denoted $\mathcal{AMP}_{n,n-1}[d\text{–SA}]$ in the following), this section shows that (d,ϵ)-SAA can be solved in this model while $(d-1,\epsilon)$-SAA cannot. It also shows that d-SA cannot be solved in $\mathcal{ACS}^d_{n,n-1}$ (which is stronger than $\mathcal{AMP}_{n,n-1}[d\text{–SA}]$). The resulting computability map is represented in Figure 8. An arrow means that a reduction exists, while a crossed out arrow means that no reduction exists. (Let us observe that the arrows for $d = n$ are trivial as, in this case, each process is allowed to decide the value it proposes without communicating with other processes.)

Theorem 4 *It is possible to solve (d,ϵ)-SAA in $\mathcal{AMP}_{n,n-1}[d\text{–SA}]$.*

Proof Let the input of each process p_i be the d-solo coordinates of a point in $\mathbb{R}^d$. The processes execute first a d-set agreement algorithm, at the end of which they agree on at most d points of $\mathbb{R}^d$. As these points have been proposed by processes, they belong to the convex hull of proposed values, and consequently satisfy the validity property of (d,ϵ)-SAA. Moreover, as no more than d different points are output by d-set agreement algorithm, they trivially satisfy the agreement property of (d,ϵ)-SAA, which concludes the proof. $\Box_{Theorem\ 4}$

The following theorem strengthens Lemma 6, namely, $(d-1,\epsilon)$-SAA cannot be solved in $\mathcal{ACS}^d_{n,n-1}$ enriched with a solution to d-SA.

Theorem 5 *It is impossible to solve $(d-1,\epsilon)$-SAA in $\mathcal{ACS}^d_{n,n-1}[d\text{–SA}]$.*

Let us notice that, as $\mathcal{ACS}^d_{n,n-1}[d-\mathrm{SA}]$ is stronger than $\mathcal{AMP}_{n,n-1}[d-\mathrm{SA}]$, it follows from this theorem that $(d-1,\epsilon)$-solo approximate agreement cannot be solved in $\mathcal{AMP}_{n,n-1}[d-\mathrm{SA}]$.

Proof The proof is by contradiction. Assuming that there is an algorithm A that solves $(d-1,\epsilon)$-AA in $\mathcal{ACS}^d_{n,n-1}[d-\mathrm{SA}]$, let us consider its executions in which the processes p_{d+1}, ..., p_n crash before executing any step, and the messages among the processes p_1, ..., p_d are delayed until each of them has decided. each process p_i, $1 \leq i \leq d$ retrieve only its own value from the CO^d objects in every round. As no more than d processes invoke A, there is a subset of executions in which the behavior of each process p_i, $1 \leq i \leq d$, is indistinguishable from a solo execution. Let us observe that, in these executions of A, each process can obtain from the underlying d-SA algorithm the value it has proposed to it (i.e., in these executions, the d-SA algorithm provides none of the processes p_1, ..., p_d with new information).

The rest of the proof is then the same as the proof of Lemma 6, starting now after its first paragraph by the sentence "Hence, to show that $(d-1,\epsilon)$-AA cannot be solved in $\mathcal{AMP}_{n,n-1}[d-\mathrm{SA}]$, let the values v_1, ..., v_d", etc. $\Box_{Theorem\ 5}$

Theorem 6 *For $d < n$, it is impossible to solve d-SA in $\mathcal{ACS}^d_{n,n-1}$.*

Proof Let us first observe that $\mathcal{ACS}^d_{n,n-1}$ can be simulated in $\mathcal{ARW}_{n,n-1}$. Hence, if d-SA can be solved in $\mathcal{ACS}^d_{n,n-1}$, it can also be solved in $\mathcal{ARW}_{n,n-1}$. But this contradicts the theorem stating that it is impossible to solve d-SA in $\mathcal{ARW}_{n,n-1}$ [4, 22, 32], which completes the proof. $\Box_{Theorem\ 6}$

The next corollary follows from the fact that, for $d < n$ and task solvability, (a) $\mathcal{ACS}^d_{n,n-1}$ is a stronger model than $\mathcal{AMP}_{n,n-1}$, (b) (d,ϵ)-SAA can be solved in $\mathcal{ACS}^d_{n,n-1}$, and (c) Theorem 6. Let $\mathcal{AMP}_{n,n-1}[(d,\epsilon)-\mathrm{SAA}]$ denote the asynchronous wait-free message-passing model augmented with (d,ϵ)-SAA.

Corollary 1 *For $d < n$, it is impossible to solve d-SA in $\mathcal{AMP}_{n,n-1}[(d,\epsilon)-\mathrm{SAA}]$.*

7 Conclusion

A process executes solo when its computes its local result without knowing the input values of the other participating processes. This paper addressed round-based asynchronous wait-free executions in which up to d processes may execute solo in each round. Among several contributions, the paper presented a strict hierarchy of wait-free iterated models, called d-solo models, and a topology-based characterization of the colorless tasks which can be solved in such d-solo models, $1 \leq d \leq n$. The paper also introduced a colorless task, denoted (d,ϵ)-solo approximate agreement (a generalization of the classic approximate agreement task), which can be solved in the d-solo model and cannot be solved in the $(d+1)$-solo model.

The d-solo hierarchy is reminiscent of the t-resilient hierarchy encountered in fault-tolerant computing. Replacing the wait-free model by a t-resilient model, and considering the (x,ϵ)-solo approximate agreement problem, it would be interesting to see how are related the parameters n, t, d, and x for the problem to be solvable.

Acknowledgments

This work has been partially supported by the French ANR project DISPLEXITY, which is devoted to computability and complexity in distributed computing) and a UNAM-PAPIIT grant. The authors want to thank Damien Imbs for discussions related to the topic addressed in the paper and Philippe Rannou for his help in the proof of Lemma 6.

References

[1] Attiya H., Bar-Noy A., and Dolev D., Sharing memory robustly in message passing systems. *Journal of the ACM*, 42(1):121-132, 1995.

[2] Attiya H., Bar-Noy A., Dolev D., Peleg D., and Reischuk R., Renaming in an asynchronous environment. *Journal of the ACM*, 37(3):524-548, 1990.

[3] Attiya H. and Welch J., *Distributed computing: fundamentals, simulations, and advanced topics.* Wiley-Interscience (Second Ed.), 2004.

[4] Borowsky E. and Gafni E., Generalized FLP impossibility results for t-resilient asynchronous computations. *Proc. 25th ACM Symposium on Theory of Computing (STOC'93)*, ACM Press, pp. 91-100, 1993.

[5] Borowsky E. and Gafni E., A simple algorithmically reasoned characterization of wait-free computations (Extended abstract). *Proc. 16th ACM Symposium on Principles of Distributed Computing (PODC'97)*, pp. 189-198, 1997.

[6] Borowsky E., Gafni E., Lynch N., and Rajsbaum, S. The BG distributed simulation algorithm. *Distributed Computing,* 14(3): 127-146 (2001).

[7] Castañeda A. and Rajsbaum S. New combinatorial topology bounds for renaming: the lower bound. *Distributed Computing* 22(5-6): 287-301 (2010).

[8] Castañeda A., Rajsbaum S., and Raynal M., The renaming problem in shared memory systems: an introduction. *Computer Science Review*, 5(3): 229-251, 2011.

[9] Chaudhuri S., More *choices* allow more *faults:* set consensus problems in totally asynchronous systems. *Information and Computation,* 105:132-158, 1993.

[10] Delporte-Gallet C., Fauconnier H., Gafni E., and Rajsbaum S., Black art: obstruction-free k-set agreement with $|MWMR$ registers$| <$ |proccesses|. *Proc. Int'l Conference on Networked Sytems (NETYS'13),* May 2-4, 2013, Marrakech (Morocco). To appear in Springer LNCS.

[11] Dolev D., Lynch N.A., Pinter S.S., Stark E.W., and Weihl W.E., Reaching approximate agreement in the presence of faults. *journal of the ACM* 33(3):499-516, 1986.

[12] Elrad T.E. and Francez N., Decomposition of distributed programs into communication-closed layers. *Science of Computer Programming*, 2(3):155-173, 1982.

[13] Feder T., Hell P., Huang J., Rafiey A., Interval graphs, adjusted interval digraphs, and reflexive list homomorphisms. *Discrete Applied Mathematics*, 160(6):697-707 (2012).

[14] Fischer M.J., Lynch N.A., and Paterson M.S., Impossibility of distributed consensus with one faulty process. *Journal of the ACM*, 32(2):374-382, 1985.

[15] Flochini P., Prencipe G., and Santoro N., *Distributed computing by oblivious mobile robots*. Morgan & Claypool Publishers, 171 pages, 2012.

[16] Gafni E. and Rajsbaum S., Distributed programming with tasks. *14th Int'l Conference Principles of Distributed Systems (OPODIS'10)*, Sprionger LNCS 6490, pp. 205–218, 2010.

[17] Herlihy M.P., Wait-free synchronization. *ACM Transactions on Programming Languages and Systems*, 13(1):124-149, 1991.

[18] Herlihy M.P. and Rajsbaum S., The decidability of distributed decision tasks (extended abstract). *Proc. 29th annual ACM Symposium on Theory of computing (STOC'97)*, ACP Press, pp. 589–598, 1997.

[19] Herlihy M.P. and Rajsbaum S., A classification of wait-free loop agreement tasks. *Theoretical Computer Science*, 291(1):55–77, 2003.

[20] Herlihy M.P. and Rajsbaum S., The topology of shared-memory adversaries. *Proc. 29th ACM Symposium on Principles of Distributed Computing (PODC 2010)*, ACM Press, pp. 105–113, 2010.

[21] Herlihy M.P., Rajsbaum S., and Raynal M., Power and limits of distributed computing shared memory models. *Theoretical Computer Science*, To appear. http://dx.doi.org/10.1016/j.tcs.2013.03.002, 22 pages, 2013.

[22] Herlihy M.P. and Shavit N., The topological structure of asynchronous computability. *Journal of the ACM*, 46(6):858-923, 1999.

[23] Kozlov, D. Chromatic subdivision of a simplicial complex. *Homology, Homotopy and Applications,* 14(1):1Ð13, 2012.

[24] Loui M.C., and Abu-Amara H.H., Memory requirements for agreement among unreliable asynchronous processes. *Par. and Distributed Computing: Vol. 4 of Advances in Comp. Research,* JAI Press, 4:163-183, 1987.

[25] Maunder C.R.F., *Algebraic topology.* Dover Publications, 393 pages, 1990.

[26] Mendes H. and Herlihy M.P., Multidimensional approximate agreement in Byzantine asynchronous systems. *Proc. 45th ACM Symposium on the Theory of Computing (STOC'13)*, ACM Press, 2013.

[27] Munkres J.R., *Elements of algebraic topology.* Addison-Wesley Publishing Company, 547 oages, 1984.

[28] Potop-Butucaru M., Raynal M., and Tixeuil S., Distributed computing with mobile robots: an introductory survey. *Proc. 14th Int'l Conference on Network-Based Information Systems (NBiS'11)*, pp. 318-324, IEEE Press, 2011.

[29] Rajsbaum, S. Iterated shared memory models. *Proc. 9th Latin American Symposium Theoretical Informatics (LATIN 2010)*, Springer LNCS 6034, pp. 407-416, 2010.

[30] Raynal M., *Concurrent programming: algorithms, principles, and foundations*. Springer, 515 pages, 2013 (ISBN 978-3-642-32027-9).

[31] Stillwell J., *Classical topology and combinatorial group theory*. Springer (2nd Edition), 343 pages, 1993.

[32] Saks M. and Zaharoglou F., Wait-free k-set agreement is impossible: the topology of public knowledge. *SIAM Journal on Computing*, 29(5):1449-1483, 2000.

[33] Vaidya N.H. and Garg V.K., Byzantine vector consensus in complete graphs. *Proc. 32nd ACM Symposium on Principles of Distributed Computing (PODC'13)*, ACM Press, 2013.

[34] Zeeman E.C., Relative simplicial approximation. *Proc. Mathematical Proceedings of the Cambridge Philosophical Society*, 60:39-43, 1964.

A Notations and Basic Topology Notions

We use standard notions from distributed computing (e.g. [3, 30]) and topology (e.g., [27, 31]).

A.1 Table of notations

The notations used in the paper are summarized in the next table.

Notation	Meaning
d-SA	d-Set agreement
(d, ϵ)-SAA	(d, ϵ)-Solo approximate agreement
$\mathcal{ACS}^d_{n,n-1}$	Asynchronous wait-free concurrent d-solo model
$CO^d[r]$	Communication object used in round r of $\mathcal{ACS}^d_{n,n-1}$
$\mathcal{ACS}^d_{n,n-1}[d\text{−SA}]$	Asynchronous wait-free concurrent d-solo ,augmented with d-SA
$\mathcal{ARW}_{n,n-1}$	Asynchronous wait-free read/write model
$\mathcal{AMP}_{n,n-1}$	Asynchronous wait-free message-passing model
$\mathcal{AMP}_{n,n-1}[d\text{−SA}]$	Asynchronous wait-free message-passing model augmented with d-SA
$\mathcal{AMP}_{n,n-1}[(d,\epsilon)\text{−SAA}]$	Asynchronous wait-free message-passing model augmented with (d, ϵ)-SAA
IIS	Iterated immediate snapshot model
σ^m	m-simplex $\{s_0, \ldots, s_m\}$
Baryσ^m	Barycentric subdivision of σ^m
Bary$(\mathcal{K}/\mathcal{K}_0)$	Barycentric subdivision of $\mathcal{K}$ holding $\mathcal{K}_0$ fixed
Skel$^d\sigma^m$	d-dimensional skeleton of σ^m, all simplexes of dimension at most d
Stel(σ^m, b)	Stellar subdivision of σ^m at vertex b
Div$_d$ σ^m	d-dim. subdiv. of σ^m, barycentric subdiv. leaving the $(d-1)$-skeleton untouched
$\partial\sigma^m$	Boundary complex consisting of all simplexes of dim $\leq m-1$
Stv, St$(v, \mathcal{K})$	Star of v in $\mathcal{K}$; simplices that include v, Open Star if only their interiors is taken

A.2 Basic topology notions

A *geometric simplex* σ of dimension n, is the set of all points spanned by the set $\{a_0, \ldots, a_n\}$ of geometrically independent points in R^N, all points x such that

$$x = \sum_{i=0}^{n} t_i a_i, \text{ where } \sum_{i=0}^{n} t_i = 1, \ t_i \geq 0.$$

The numbers t_i are the *barycentric coordinates* of x with respect to points $a_0, \ldots, a_n$, which are the *vertices* of σ. Any simplex spanned by a subset of these vertices is a *face* of σ, and the *boundary* of σ, $\partial\sigma$, consists of the faces of σ different from σ. The *interior* of σ is $\sigma - \partial\sigma$, and consist of all points x as above where all $t_i > 0$.

A *geometric complex* $\mathcal{K}$ in R^N is a collection of simplices such that every face of a simplex of $\mathcal{K}$ is in $\mathcal{K}$ and the intersection of any two simplexes of $\mathcal{K}$ is a face of each of them. Notice that a point of $\mathcal{K}$ lies in the interior of a unique simplex, called the *carrier* of the vertex in $\mathcal{K}$. The collection of all simplices of $\mathcal{K}$ of dimension at most p is a subcomplex of $\mathcal{K}$ called its p-*skeleton*. The subset of R^N that is the union of the simplices of $\mathcal{K}$ is the *polytope* of $\mathcal{K}$, denoted $|\mathcal{K}|$. A *polyhedron* is a space that is the polytope of a geometric complex.

The *star* of a vertex v in $\mathcal{K}$, denoted St$(v, \mathcal{K})$, or Stv when $\mathcal{K}$ is understood, is the union of interiors of the simplices having v as a vertex. While the open star consists of the interiors of these simplexes, the *closed star* contains all of them.

Often one is interested only in the combinatorics of a geometric complex and the analytic geometry details are irrelevant, so one uses an *abstract complex*, $\mathcal{K}$, which is a collection of finite non-empty sets, called *abstract simplices*, such that if σ is an element of $\mathcal{K}$ then so is every non-empty subset of σ. The *dimension* of σ is one less that its number of elements, called *vertices*. Often we do not distinguish between an abstract simplicial complex $\mathcal{K}$ and its *geometric realization*, consisting of a geometric complex with the same combinatorial structure.

A *simplicial map* δ from $\mathcal{K}$ to $\mathcal{K}'$ sends each vertex of $\mathcal{K}$ to a vertex of $\mathcal{K}'$, such that if $\{v_i\}$ are vertices of a simplex σ in $\mathcal{K}$ then $\delta(v_i)$ are (not necessarily distinct) vertices of a simplex of $\mathcal{K}'$. Thus δ induces a map from simplexes of $\mathcal{K}$ to simplexes of $\mathcal{K}'$. The map δ can be defined only on the abstract complex, or extended to an *induced* continuous map from the geometric complex $\mathcal{K}$ to the geometric complex $\mathcal{K}'$ linearly: if $x = \sum t_i v_i$ then $\delta(x) = \sum t_i \delta(v_i)$.

A *carrier map* Ξ sends each simplex of $\mathcal{K}$ to subcomplex of $\mathcal{K}'$, such that if $\sigma \subseteq \sigma'$ then $\Xi(\sigma) \subseteq \Xi(\sigma')$. The carrier map is *strict* if $\Xi(\sigma) \cap \Xi(\sigma') = \Xi(\sigma \cap \sigma')$.

We recall now homotopy and contractibility notions. Two maps $f, g : (X, A) \to (Y, B)$ (where $f, g : X \to Y$, such that $f(A) \subseteq B$, $g(A) \subseteq B$) are *homotopic* if there exists a map $F : (X \times I), (A \times I) \to (Y, B)$, such that $F(x, 0) = f(x)$ and $F(x, 1) = g(x)$ for all $x \in X$. Recall that if f, g are maps from (X, A) to (Y, B), such that $f|A = g|A$, f is *homotopic to g relative to A* if there exists homotopy F, such that $F(a, t) = f(a) = g(a)$ for all $a \in A$, $t \in I$, so that it is fixed on A. A *loop* λ can be viewed as a simplicial map from a subdivision of the 1-skeleton of σ^2 to $\mathcal{O}^*$. The continuous version is a loop $|\lambda|$, a map from $|\text{Skel}^1\sigma^2|$ to $\mathcal{O}^*$ obtained linearly from λ. Since $|\text{Skel}^1\sigma^2|$ is homeomorphic to S^1 (a 1-dimensional sphere), then a loop can also be viewed as a map from S^1 to $\mathcal{O}^*$. Intuitively, a deformation of one loop into another loop λ', is an homotopy, a continuous map that defines loops, starting with λ at time $t = 0$ and by small modifications ends up with the loop λ' at time $t = 1$. A *trivial loop* consists of just on vertex. A loop *contractible* if it is homotopic to a trivial one. Equivalently, a loop $\lambda : S^1 \to \mathcal{O}^*$ is contractible if it can be extended to the interior of S^1, i.e. to a map of the unit disk $\lambda : D^2 \to \mathcal{O}^*$.

A complex $\mathcal{K}'$ is a *subdivision* of a geometric complex $\mathcal{K}$ if each simplex of $\mathcal{K}'$ is contained in a simplex of $\mathcal{K}$ and each simplex of $\mathcal{K}$ equals the union of finitely many simplices of $\mathcal{K}'$. Thus, $|\mathcal{K}'|$ and $|\mathcal{K}'|$ are equal as sets.

Given a complex (abusively denoted σ^m) consisting of an m-simplex $\sigma^m = \{s_0, \ldots, s_m\}$ and all its faces, and a vertex b in the interior of σ^m, the *stellar subdivision* Stel(σ^m, b) is constructed by taking a cone with apex b over the boundary complex $\partial\sigma^m$. While any point in the interior can be used as the apex, it is natural to use the barycenter of σ^m. The *barycenter* of $\sigma = \{v_0, \ldots, v_p\}$ is the point $\hat{\sigma} = \sum_{i=0}^{p} 1/(p+1)v_i$. More generally, a *cone* on a complex $\mathcal{K}$ with vertex b, where b is a point such that each ray emanating from w intersects $|\mathcal{K}|$ in at most one point, is sometimes denoted $b * \mathcal{K}$. It contains all simplices of the form $\{b, a_0, \ldots, a_p\}$, where $\{a_0, \ldots, a_p\}$ is a simplex of $\mathcal{K}$.

The *barycentric subdivision* of $\mathcal{K}$ is a complex $\mathcal{K}'$, denoted Bary$\mathcal{K}$. It is constructed by a sequence of subdivisions of the skeletons of $\mathcal{K}$, starting with its 0-skeleton, L_0. In general, if L_p is a subdivision of the p-skeleton of $\mathcal{K}$, then L_{p+1} is the subdivision of the $p+1$ skeleton obtained by inserting a vertex $\hat{\sigma}$ in the barycenter of each $p+1$ simplex σ of $\mathcal{K}$, and constructing a cone with apex $\hat{\sigma}$ on the subdivided boundary of σ. As an abstract complex, Bary$\mathcal{K}$ equals the collection of all simplices of the form $\hat{\sigma}_1, \hat{\sigma}_2, \ldots, \hat{\sigma}_k$, such that $\sigma_1 \succ \sigma_2 \succ \cdots \succ \sigma_k$, where $\sigma_i \succ \sigma_{i+1}$ means σ_{i+1} is a proper face of σ_i. We may think of Bary as a strict carrier map from $\mathcal{K}$ to $\mathcal{K}'$, that sends each simplex σ of $\mathcal{K}$ to the subcomplex of $\mathcal{K}'$ consisting of the barycentric subdivision of σ. Inductively, the r-th barycentric subdivision of $\mathcal{K}$, denoted Bary$^r\mathcal{K}$, is equal to the barycentric subdivision of Bary$^{r-1}\mathcal{K}$, and Bary$^1\mathcal{K} =$ Bary$\mathcal{K}$.

The *diameter* of σ is the maximum distance between vertices of σ. The following is well know.

Lemma 7 *Given a finite complex $\mathcal{K}$ and $\epsilon > 0$, there is an R such that each simplex of Bary$^R\mathcal{K}$ has diameter less than ϵ.*

A continuous map $h : |\mathcal{K}| \to |\mathcal{L}|$ can be approximated by a simplicial map, f, in the following sense. A *simplicial approximation* f : Bary$^R\mathcal{K} \to \mathcal{L}$, $R \geq 1$, is a simplicial map such that $h(St\, v) \subset St\, f(v)$ for each vertex v of Bary$^R\mathcal{K}$. Equivalently, given $x \in |\mathcal{K}|$, if $h(x)$ is in the interior of simplex τ (its carrier) in $\mathcal{L}$, then $f(x)$ is also in τ. (recall that we may assume the simplicial map is the induced map on the geometric complex, so f is defined on every such point x). Notice that simplicial approximations compose.

Theorem 7 (Simplicial Approximation Theorem) *Given a continuous map $h : |\mathcal{K}| \to |\mathcal{L}|$ from a finite complex $\mathcal{K}$ to a complex $\mathcal{L}$, there is an R such that h has a simplicial approximation f : Bary$^R\mathcal{K} \to \mathcal{L}$.*

Any simplicial approximation f to h is homotopic to h, and the homotopy is relative to those points x such that $f(x) = h(x)$. Moreover, if $h : (|\mathcal{K}|, |\mathcal{K}'|) \to (|\mathcal{L}, |\mathcal{L}'|)$, then any simplicial approximation f to h satisfies $f(|\mathcal{K}'|) \subseteq |\mathcal{L}'|$ and h and f are homotopic as maps of pairs.

For our results we need a generalized version of barycentric subdivision, that leaves a subcomplex unchanged. If $\mathcal{K}$ is a complex and $\mathcal{K}_0$ a subcomplex, then Bary$(\mathcal{K}/\mathcal{K}_0)$ is the *barycentric subdivision of $\mathcal{K}$ holding $\mathcal{K}_0$ fixed,* or *relative* to $\mathcal{K}_0$. It is constructed by a sequence of subdivisions of the skeletons of $\mathcal{K}$, starting with L_0 the 0-skeleton. In general, if L_p is a subdivision of the p-skeleton of $\mathcal{K}$, and each simplex of $\mathcal{K}_0$ of dimension at most p belongs to L_p, define L_{p+1} to be the union of L_p with all $p+1$ simplexes of $\mathcal{K}_0$, and the cones $\hat{\sigma} * L_\sigma$, were σ ranges over all $p+1$ simplexes σ of $\mathcal{K}$ not in $\mathcal{K}_0$ (L_σ is the subcomplex of L_p whose polytope is $\partial\sigma$). As an abstract complex, Bary$(\mathcal{K}/\mathcal{K}_0)$ has simplexes of the form

$$\tau = \{\hat{\sigma}_1, \ldots, \hat{\sigma}_q, v_0, \ldots, v_p\},$$

where $s = \{v_0, \ldots, v_p\}$ is a simplex of $\mathcal{K}_0$ and $\sigma_1, \ldots, \sigma_q$ are simplexes of $\mathcal{K}$ not in $\mathcal{K}_0$ with $\sigma_1 \succ \ldots \succ \sigma_q \succ s$. Notice that either $v_0, \ldots, v_p$ or $\hat{\sigma}_1, \ldots, \hat{\sigma}_q$ may be missing from the expression above. By iterating the process, we get the *r-th barycentric subdivision of $\mathcal{K}$ relative to* $\mathcal{K}_0$, Bary$^r(\mathcal{K}/\mathcal{K}_0)$, which holds $\mathcal{K}_0$ fixed. Notice, that this iterated version is not what we use in a d-model, because in a d model, in each round the $d-1$ skeleton is again not subdivided.

If h is a continuous map $h : |\mathcal{K}| \to |\mathcal{L}|$ that is already simplicial in $\mathcal{M}$, then the Simplicial Approximation Theorem, shows that any simplicial approximation f : Bary$^R\mathcal{K} \to \mathcal{L}$, will satisfy that f remains consistent with h

inside of $\mathcal{M}$, i.e., $f(\sigma) \subseteq h(\sigma)$. We cannot demand that f and h coincide in $\mathcal{M}$, because the simplexes of $\mathcal{M}$ are unchanged under subdivision. The relative simplicial approximation theorem due to [34], see also [25], states that it is possible to obtain an f that is equal to h on $\mathcal{M}$ and is homotopic to h.

Theorem 8 (Relative Simplicial Approximation Theorem) *Given a continuous map $h : |\mathcal{K}| \to |\mathcal{L}|$ from a finite complex $\mathcal{K}$ to a complex $\mathcal{L}$, a subcomplex $\mathcal{M}$ of $\mathcal{K}$ with h simplicial on $\mathcal{M}$, there is an R and a simplicial map $f : Bary^R(\mathcal{K}/\mathcal{M}) \to \mathcal{L}$ such that h and f are homotopic relative to $\mathcal{M}$.*

B On Approximate Agreement

B.1 On the agreement property of (d, ϵ)-solo approximate agreement

As in all approximate agreement problems (see below) the idea is to force processes to decide values that are not too far the ones from the others. But, as each process that executes solo sees only the value it proposes, it can decide only its value. Differently, the other processes see the values decided by the processes which executed solo. This motivates the definition given in Section 5.

More precisely, the fact that up to d processes are allowed to decide any values in the convex hull of the proposed values is directly related to the possibility of up to d processes executing solo in an execution. This set of at most d processes defines the set S used in the definition of (d, ϵ)-SAA. As indicated, when this occurs, the values decided by these processes are collectively represented by their convex hull CH and any other process has to decide a value "not too far" from CH.

If no process executes solo, the agreement property states that the set S has nevertheless to contain at least one process: $1 \leq |S| \leq d$ (hence S has not to be confused with the operational set π_1 used in the definition of the communication objects involved in the d-solo model, Section 2.1). As the values decided by the processes in S are then within the convex hull of the proposed values, it is as these processes have executed solo.

Remark One could think to have an agreement property composed of two parts, defined as follows:

- Solo execution agreement. If at least one process executes solo, the agreement property is the one described in Section 5.
- No-solo execution agreement. If no process executes solo, the Euclidean distance between any two decided values is at most ϵ.

Unfortunately, as in the non-blocking atomic commit problem (NBAC), this definition involves the behavior of the run, which becomes an input of the problem. It follows that, as NBAC, the problem captured by this extended definition is not a task (a task is defined by an application from input vectors to output vectors and this application has to be independent of the execution pattern). *End of remark.*

B.2 Related work

Assuming that each process proposes a value from the set $\mathbb{R}$, approximate agreement requires that the decided values belong to the range of proposed values (validity property), and the difference between any two decided values is upper bounded by a predefined constant ϵ (approximate agreement property) [11]. This problem was introduced in the context of asynchronous systems prone to Byzantine process failures (in this case, the validity and agreement properties apply only to the non-faulty processes). An aim of this paper was to introduce a problem that, contrarily to consensus, can be solved despite asynchrony and process failures. This paper showed that ϵ-approximate agreement can be solved iff $n > 5t$ where t is the maximum number of Byzantine processes.

Differently, in the context of asynchronous read/write systems where processes are prone to crash failures, approximate agreement can be solved for any number of process crashes.

Considering Byzantine failures and asynchronous message-passing systems, ϵ-approximate agreement has very recently been generalized to the case where the proposed values are points in an m-dimensional space $\mathbb{R}^m$ [26, 33]. A decided value (point) must then belong to the convex hull of the points proposed by the non-faulty processes (validity), and the Euclidean distance between any pair of values – points – decided by non-faulty processes has to be upper bounded by a predefined constant. It is shown in the previous papers, that the problem can be solved despite Byzantine behaviors and asynchrony iff $n > t(m + 2)$.

Our definition of (d, ϵ)-approximate agreement considers another *uncertainty feature* of distributed computing, namely, the maximal number of processes which may execute solo whose decided values are abstracted by their convex hull.

B.3 An example of application

Let us consider a set of n asynchronous robots able to move in a common d-dimensional space (CORDA model where $d \in \{2, 3\}$ [15, 28]). The *robot gathering* problem requires that the robots move to meet in a given area which depends on their initial positions. The initial position of a robot constitutes its input value.

In the context of asynchronous systems where processes can commit Byzantine failures, the approximate agreement algorithm described in [26] solves the robot gathering problem. The area where the non-faulty robots (processes) have to meet is within the convex hull of their initial positions, and the final distance between any two of them is upper bounded by some predefined constant.

As described in the Introduction, when considering robots in the d-solo model, any algorithm solving the (d, ϵ)-solo approximate agreement problem allows the robots to meet in an area within the convex hull defined by their initial positions, and this area is defined by the agreement property of (d, ϵ)-SAA, which is an approximate agreement with respect the convex hull defined by the values decided by the solo processes.