Information and Computation 260 (2018) 51–71

Contents lists available at ScienceDirect

Information and Computation

www.elsevier.com/locate/yinco

On mobile agent verifiable problems ☆

Evangelos Bampas [a,*], David Ilcinkas [b,*]

[a] LIS, Aix-Marseille University and CNRS, Marseille, France
[b] LaBRI, CNRS and Univ. Bordeaux, France

ARTICLE INFO

Article history:
Received 25 July 2017
Received in revised form 4 February 2018

ABSTRACT

We consider decision problems that are solved in a distributed fashion by synchronous mobile agents operating in an unknown, anonymous network. Each agent has a unique identifier and an input string and they have to decide collectively a property which may involve their input strings, the graph on which they are operating, and their particular starting positions. Building on recent work by Fraigniaud and Pelc [J. Parallel Distrib. Comput, vol. 109, pp. 117–128], we introduce several natural new computability classes allowing for a finer classification of problems below MAV or its complement class co-MAV, the former being the class of problems that are verifiable when the agents are provided with an appropriate certificate. We provide inclusion and separation results among all these classes. We also determine their closure properties with respect to set-theoretic operations. Our main technical tool, which is of independent interest, is a new meta-protocol that enables the execution of a possibly infinite number of mobile agent protocols essentially in parallel, similarly to the well-known dovetailing technique from classical computability theory.

1. Introduction

1.1. Context and motivation

The last few decades have seen a surge of research interest in the direction of studying computability- and complexity-theoretic aspects for various models of distributed computing.

Significant examples of this trend include the investigation of unreliable failure detectors (introduced in [8]), as well as wait-free hierarchies (introduced in [22]), which both concern crash-fault-tolerance in distributed asynchronous systems. An unreliable failure detector is an external failure detection mechanism that can make mistakes. It is composed of local modules, one on each node, which output a set of processes that the failure detector module suspects have crashed. Chandra and Toueg [8] introduced this notion and a way to compare unreliable failure detectors, and they exhibited and studied an infinite hierarchy of failure detector classes, leading to a way of classifying distributed tasks, according to the weakest failure detector allowing the given task to be solved. Another approach consists in directly classifying concurrent objects, which are data structures shared by concurrent processes. This line of work, inspired by the seminal paper [22], considers

☆ A preliminary version of this work appears in the Proceedings of the 12th Latin American Theoretical Informatics Symposium, LNCS vol. 9644, pp. 123–137, Springer, 2016.

* Corresponding authors.
E-mail addresses: evangelos.bampas@gmail.com (E. Bampas), david.ilcinkas@labri.fr (D. Ilcinkas).

https://doi.org/10.1016/j.ic.2018.03.003
0890-5401/

wait-free implementations. These are implementations such that any operation on the object by a process terminates in a finite number of steps, even if other processes crash or have different progress speeds. A so-called wait-free hierarchy of objects is then constructed by classifying objects depending on whether an object has a wait-free implementation only using instances of another object as communication primitives. Finally, a more recent work [20] deals with checkability of distributed tasks: the decision problem associated to a task consists in determining whether a given output is valid with respect to the task specification.

Computability- and complexity-theoretic studies for decision problems also concern the fault-free distributed $\mathcal{LOCAL}$ and $\mathcal{CONGEST}$ models. In both models, a communication graph describes which nodes are able to directly communicate. The nodes have unique IDs, and they operate in synchronous rounds, in which any node is able to send a (possibly different) message to each of its neighbor. There are no restrictions on the memory or computing capabilities of the nodes. In the $\mathcal{LOCAL}$ model, there are even no restrictions on the size of the messages, while the $\mathcal{CONGEST}$ model usually assumes that each message has size at most $O(\log n)$ bits.

In both models, several papers studied different classes of distributed languages. A distributed language is basically a set of labeled networks. For example, the set of properly colored networks is a distributed language. In particular, decision and verification were studied. A distributed language is decidable if there exists a distributed algorithm able to globally determine whether the input labeled network is in the language, while a distributed language is verifiable if the membership of an instance to the language can be checked by a distributed algorithm with the help of certificates, in a similar manner as certificates are used in the centralized complexity class NP. System-wide acceptance is usually defined as all nodes locally accepting. System-wide rejection is usually defined as at least one node rejecting.

Deterministic and randomized decision classes in the $\mathcal{LOCAL}$ model were introduced in [26] and [18]. The latter paper also introduced one verification class, which is somehow a variant of another verification class introduced by Korman, Kutten, and Peleg [23]. The impact of identifiers or the lack of them has also been investigated [13,16,17]. In the $\mathcal{CONGEST}$ model, decision and verification were also considered [11,21], as well as a distributed version of property testing [5]. For a much more detailed survey, see [14].

A different approach considers the characterization of problems that can be solved under various notions of termination detection or various types of knowledge about the network in message-passing systems [3,4,6,7,28]. Finally, recent works focus on the computational power of teams of mobile agents [10,19]. Our work lies in this latter direction.

The mobile agent paradigm has been proposed since the 90's as a concept that facilitates several fundamental networking tasks including, among others, fault tolerance, network management, and data acquisition [24], and has been of significant interest to the distributed computing community (see, e.g., the surveys on graph exploration [9], identification of hostile nodes [25], or rendezvous [27]). As such, it is highly pertinent to develop a computability theory for mobile agents, that classifies different problems according to their degree of (non-)computability, insofar as we are interested in really understanding the computational capabilities of groups of mobile agents.

One may argue about the usefulness of developing a theory specifically for mobile agent decision problems, apart from its inherent theoretical interest. There are several reasons.

On the one hand, we believe that such a study is bound to yield intermediate results, tools, intuitions, and techniques that will prove useful when one moves on to consider from a computability/complexity point of view other, perhaps more traditional, mobile agent problems, such as exploration [9], rendezvous [27], graph searching [15], etc., which are not decision problems. One such tool is the protocol that we develop in this paper, which enables the interleaving of the executions of a possibly infinite number of mobile agent protocols.

On the other hand, we think that decision problems are inherently interesting, despite the relative shortage of studies devoted to them ([10,19]). Most studies so far in mobile agent computing indeed concern "complex" problems, in the sense that either the output is not binary (constructing the map of the network, counting the number of agents or nodes, etc.) or the problem requires specific terminal configurations (like in rendezvous) or even properties on the sequences of configurations (like in exploration or graph searching). Decision problems are however closely related to these more complex problems. First, a significant proportion of the studies make initial assumptions on the maximum and/or minimum number of agents in the network [2,12], on the topology (like assuming that the agents are on a tree [1]), or more generally on the possible initial configurations. Algorithms solving decision problems can be used to check that such assumptions actually hold, before running the algorithm dedicated to solve the problem at hand, which may consume resources. Second, certain contexts are subject to faults. In such cases, algorithms solving decision problems may be used for fault tolerance purposes: agents can check (possibly by means of certificates that were constructed while solving the main problem) whether the achieved configuration or output satisfies the desired properties.

In this paper, we consider one of the most broadly used models of mobile agent computing, the same as the one studied in [19]. More precisely, we consider a distributed system in which computation is performed by one or more deterministic mobile agents, operating in an unknown, anonymous network (nodes have no identifiers and edges are only locally distinguished). Each agent is modeled as a deterministic Turing machine, has a unique identifier and is provided with an input string, and they have to collectively decide a property which may involve their input strings, the graph on which they are operating, and their particular starting positions.