Brief Announcement: Fast Aggregation in Population Protocols

Ryota Eguchi[1] and Taisuke Izumi[2]

1 **Graduate School of Engineering, Nagoya Institute of Technology, Nagoya, Japan**
`28414018@stn.nitech.ac.jp`
2 **Graduate School of Engineering, Nagoya Institute of Technology, Nagoya, Japan**
`t-izumi@nitech.ac.jp`

Abstract

The coalescence protocol plays an important role in the population protocol model. The conceptual structure of the protocol is for two agents holding two non-zero values a, b respectively to take a transition $(a, b) \to (a + b, 0)$, where $+$ is an arbitrary commutative binary operation. Obviously, it eventually aggregates the sum of all initial values. In this paper, we present a fast coalescence protocol that converges in $O(\sqrt{n}\log^2 n)$ parallel time with high probability in the model with an initial leader (equivalently, the model with a base station), which achieves an substantial speed-up compared with the naive implementation taking $\Omega(n)$ time.

1998 ACM Subject Classification C.2.4 Distributed Systems, I.1.2 protocols

Keywords and phrases population protocol, aggregation

Digital Object Identifier 10.4230/LIPIcs.DISC.2017.49

1 Introduction

A *passively-mobile* system, which is an abstract notion of wireless ad-hoc networks, consists of a collection of moving objects (called *agents*) in a certain region, with computing devices that do not control over how they move. Despite the restrictions on communication range, memory, and computational power caused by the mobility requirement, the devices must execute cooperatively some task through tiny local computation and short-range communication with other devices located nearby. Typical examples of passively-mobile systems are the network of smart devices attached to cars or animals. *Population protocol* is one of the promising models for such a system, which is first introduced by Angluin et al. [2]. A Population protocol consists of anonymous and identical n agents, which are defined as deterministic state machines. The communication among agents is performed by pairwise interactions, where two interacting agents change their states following a transition function (protocol) deployed to all agents. An execution of a population protocol is a sequence of pairwise interactions. In the basic model, the scheduling of interactions is worst-case but guaranteed to be *fair*, which means that if in the infinitely-many interactions every two agents interact infinitely often.

Recent trends of this model are to design fast protocols for popular problems (e.g. leader election, majority) converging in $O(polylog(n))$ time, and to reveal trade-offs between time and space for several problems. To measure time in the runs of the population protocol models, the (uniform) probabilistic scheduler is often assumed. In the model, two agents interacting at each step are selected at random uniformly and independently. In the literature of this model, time complexity is defined as the number of interactions divided by the number of agents n, and space complexity is the number of states used by each agent.

We consider some abstract problem, called the *aggregation* problem. Precisely, the aggregation problem is defined by any monoid $(X, +)$, where initially each agent i has a value $x_i \in X$, and eventually one specific agent must output the value of $s = \sum_i x_i$. This problem can be solved by the traditional *coalescence* protocol, whose transition rule for two agents with values a and b respectively is specified by $(a, b) \to (a + b, 0)$. One can see that, the standard coalescence protocol needs $\Theta(n)$ time for convergence, since the probability that the last two agents having non-zero values interact is $1/n^2$.

In this paper, we present a new coalescence protocol. It achieves $O(\sqrt{n}\log^2 n)$-convergence time in the special model with existence of one unique leader (equivalently, the model with a base station). On the space complexity side, agents (including the leader) uses $O(|X|^3)$ states.

Problem Statement

Let $(X, +)$ be an arbitrary commutative monoid whose identity element is zero (where $+$ is not necessarily the standard arithmetic sum), and $\hat{X} = X \setminus \{0\}$. In the aggregation problem for $(X, +)$, each agent i initially has a value $x_i \in X$, and the goal of the task is that the leader computes the value $s = \sum_i x_i$. More precisely, we assume that the leader equips an output register storing a value in X. The value of the output register must be converged and stabilized into s. Note that the leader does not have to detect the termination of an execution, and is allowed to update answers multiple times. The computation time of the aggregation problem is defined as the time taken until the convergence of the output register.

Outline of Our Algorithm

Our algorithm utilizes several algorithmic tools proposed in past literature as building blocks. Before the presentation of our protocol, we illustrate three tools. The first algorithm called *epidemics* (or propagation) is a straightforward subroutine used in many algorithms. The abstract structure of the epidemics is as follows: At first there are at least one agent with value v, which wishes to propagate v to all other agents, and the other agents initially with value $\perp$. The transition rule is $(v, \perp)$ or $(\perp, v)$ to (v, v). The analysis by Angluin et al. [3] shows that under the random scheduler the epidemics algorithm finishes within $O(\log n)$ parallel time with high probability.

The second tool is a synchronization mechanism called *phase clock* which counts approximately $O(\log n)$ time or $O(\log^2 n)$ time. The phase clock is first presented in the paper by Angluin et al. [3]. The phase clock is mainly introduced for a unique leader to detect the end of the epidemics (i.e. $O(\log n)$ time), and by a simple extension, it is also possible to count $O(\log^2 n)$ time [4]. A non-trivial advantage of the phase clock mechanism is that it uses only $O(1)$ states per agent.

The third tool is synthetic coin flips due to Alistarh et al. [1], which provides the accessibility of private random bits to each agent. It gives a coin flipping mechanism with reasonably small bias to the agents. The randomness of the synthetic coin flips is extracted from the random interaction-pattern of the scheduler, and thus it works only on the random scheduler.

The idea of our algorithm is very simple: The bottleneck of the standard coalescence algorithm is the situation where the number of agents with non-zero values becomes small. If only m agents have non-zero values, an interaction selected by the scheduler gets no progress of the algorithm with probability $1-\Theta((m/n)^2)$. In the naive coalescence algorithm, spending $O(\sqrt{n}\text{polylog}(n))$ parallel time, $\Theta(\sqrt{n})$ agents still have non-zero values. To accelerate the following coalescence process, when only $O(\sqrt{n})$ agents have non-zero values, we utilizes another mechanism, called *sequential absorption*. The sequential absorption first chooses an agent (which is not leader) with a non-zero value as an absorption agent only spending $O(\log n)$ parallel time. This process is achieved by utilizing the phase clock and the synthetic coin flips: At each phase, the agents with non-zero values flip the synthetic coin, the agents which get value 1 start epidemics, and the epidemics kill the agents with value 0. The number of phases to elect one unique absorption agent is $O(\log n)$, thus the total time to elect the agent is $O(\log^2 n)$. The absorption agent runs the epidemics its value, and immediately become an agent with value zero. The value reaches to the leader within $O(\log n)$ time. Repeating this procedure $\Theta(\sqrt{n})$ times, we can complete the aggregation. Since both the election and epidemics take $O(\log^2 n)$ time, the total running time of the sequential absorption is $O(\sqrt{n}\log^2 n)$. The remaining issue is to combine those two algorithms. While the sequential composition is obviously correct, it requires the timer for (exactly or approximately) counting $\Theta(\sqrt{n})$ parallel time. To avoid consuming extra memory space, instead we choose fair composition, that is, simply running them concurrently. This composition does not affect the correctness of our protocol, since the absorption agent behave following way so that the value of the sum does not change: When the absorption agent with value x_i detect its uniqueness by the phase clock counting $O(\log^2 n)$ time, it immediately change its state to the value zero, and thus x_i is never aggregated in the standard coalescence side. Here we present our main theorem. Note that our algorithm have a low probability of error, that is conversely the algorithm convergences only with high probability.

▶ **Theorem 1.** *Our algorithm solves an aggregation problem for $(X, +)$ in expected $O(\sqrt{n}\log^2 n)$ time using $O(|X^3|)$ states per agent, with high probability.*

Discussion and Research Direction

In [2], authors show the simple coalescence protocol can compute semilinear predicates, which are exact characterization of the basic population protocol model, and thus our protocol computes the predicates in $O(\sqrt{n}\log^2 n)$ time. However for computation of semilinear predicates with leader, there is much faster protocol presented by Angluin et al. [3] which converges in $O(\log^4 n)$ time with high probability. We believe that due to its simplicity and generality, there are some applications of our algorithm in population protocol models.

References

1 Dan Alistarh, James Aspnes, David Eisenstat, Rati Gelashvili, and Ronald L Rivest. Time-space trade-offs in population protocols. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2560–2579. SIAM, 2017. doi:10.1137/1.9781611974782.169.

2 Dana Angluin, James Aspnes, Zoë Diamadi, MichaelJ. Fischer, and Rene Peralta. Computation in networks of passively mobile finite-state sensors. *Distributed Computing*, 18(4):235–253, 2006. doi:10.1007/s00446-005-0138-3.

3 Dana Angluin, James Aspnes, and David Eisenstat. Fast computation by population protocols with a leader. *Distributed Computing*, 21(3):183–199, September 2008. doi:10.1007/s00446-008-0067-z.

4 Leszek Gasieniec and Grzegorz Stachowiak. Fast space optimal leader election in population protocols. *arXiv preprint arXiv:1704.07649*, 2017.