Infection algorithms for distributed agents

R. Charturvedi, P. Dymond, M. Jenkin
Computer Science and Engineering, York University, Canada
(rahul, dymond, jenkin)@cse.yorku.ca

***Abstract*—Variations on the population protocol model can be used to analyze algorithms for tasks undertaken by a collection of autonomous agents, with simple, pairwise interactions between them such as, leader election (choosing one of the agents to have special authority,) and simple summation (collecting the sum of values held by individual agents.) Here we present infection or epidemic algorithms for leader election for a collection of numbered (but unbounded) set of agents. We demonstrate a distributed probabilistic algorithm that the agents can use to elect a leader and to decide that the election is complete. Similarly, we demonstrate an algorithm for the distributed summation problem. Simulations exhibit the efficacy of this approach. An interesting aspect of these algorithms is a method for the agents to probabilistically decide when the election or summation has been completed, without *a priori* knowledge of the number of agents.**

***Keywords*-leader election; summation; robotics, infection; epidemics**

I. Introduction

We consider a set of independent agents each executing an algorithm in order to jointly complete some task. One basic task is /em leader election, identifying one of the agents to have a special role in subsequent activity. Leader election is a basic component for many other distributed algorithms. We also consider the *summation* task: aggregating the sum of the values held by all agents. Our computational model is a strengthened version of the population protocol model introduced by Angluin et al [1]. Population protocols were originally developed to model sensor networks in which agents had very limited resources and no control over their movements or communications. Key features of the basic population protocol model are

- Agents: The system consists of a large population of identical finite-state agents.
- Pairwise interactions: Agents do not send messages or share memory. Instead, an interaction between two agents updates both of their states according to a joint transition table. Interactions are atomic and independent of other activity within the collection of agents.
- Unpredictable interaction patterns: Individual agents have no control over their movement or interactions.
- Distributed inputs and outputs: The input to a population protocol is distributed across the initial state of the entire population. Similarly, the output is distributed to all agents.
- Convergence rather than termination: Agents converge to the correct value at the end of any computation, but may themselves be unaware that the computation is finished.

In contrast, here we are interested in algorithms in which termination can be decided by agents probabilistically.

Angluin et al. developed a leader election algorithm for anonymous agents that takes $\Theta(n^2)$[1] pairwise agent interactions, assuming uniform "fair" interactions. The population protocol described in [1] assumes that the agents are anonymous. For many real world applications the agents are not anonymous but rather are equipped with unique serial or identification numbers. We address here the issues of finding more efficient algorithms be developed for leader election in such applications, and devising efficient methods for recognizing the end of the computation.

In this paper (see also [2], [3]) we study a generalization of the approach of [1] in which each agent has a unique identifier or serial number. Interactions may involve comparison or exchange of these serial numbers by a pair of interacting agents. Interactions occur randomly between paris of agents, as in the population protocol models. Here we are specifically interested in random interactions where each pair of agents is equally likely to meet in each interaction. We call this the *uniform interaction* assumption, and note that this constraint implies the "fairness" property used by Angluin et al. Roughly speaking, an (infinite) computation is called *fair* if interactions occur infinitely often between all pairs of agents. These assumptions are obviously substantial restrictions, which we will discuss further.

[4] proposed a protocol for fast computation in populations of anonymous agents with a single predetermined leader. based on the use of *epidemics* to propagate information quickly through the population. In epidemic algorithms (e.g [5], [6]) an agent that wishes to disseminate a new piece of information sends the information to a set of peer agents, chosen at random. In turn, these peer agents forward the information to randomly selected agents, and so forth. Using epidemic algorithms, the protocols of [4] are able to carry out arithmetic operations such as addition, subtraction, comparison, and multiplication and division by constants using $O(n \ln^c n)$ interactions. The *community protocol model* was

[1]A function $f(n)$ is said to be $\Theta(n^k)$ if there exist positive constants c_1 and c_2 such that for sufficiently large n $c_1 n^k \leq f(n) \leq c_2 n^k$.

introduced in [7]. Community protocols allow agents to be initially assigned unique identifiers from a large set, and to store a constant number of identifiers in their memory. These additional capabilities preserve much of the simple nature of the agents as in population protocols, but allow them to be uniquely identifiable. The internal algorithms of the agents cannot be dependent on the values of the identifiers: the identifiers can only be tested for equality with one another. We use a variation on their model here, allowing comparison of identifiers, as well as allowing a limited amount of other data to be exchanged.

II. Leader election for numbered agents

For ease of exposition let us assume that the identifiers of the agents are drawn from the set of natural numbers, that there are n agents, that each agent i has its unique identifier id_i stored in its memory, and that the leader to be chosen is the one with the highest identifier value (which is *a priori* unknown.)

Each agent i also maintains a local variable $leader_i$ for recording the identifier of the elected leader or candidates throughout the process, and a boolean value $isleader_i$ for maintaining a value (initially $true$), that will become $false$ for all agents other than the eventual leader as the leader election algorithm proceeds.) Initially every agent is a candidate for leadership and $(leader_i, isLeader_i) = (id_i, true)$. That is, each agent initially regards itself as the potential leader. When two agents interact, the higher numbered agent will "infect" the lower-numbered agent, so that the agent with the lower value of $leader$ replaces its value with the higher value from the other agent's $leader$ variable and turns its boolean variable $false$.

Using this method of infection the value of the highest identifier in the population is propagated among the agents. Assuming fair interactions among agents, the population elects a unique leader using this method. This process is described in Algorithm 1.

Initially: For all agents i:;
  $leader_i \leftarrow aid_i$;
  $isLeader_i \leftarrow true$;
Interactions: Between agents i and j;
  if $leader_i > leader_j$ **then**
    $leader_j \leftarrow leader_i$;
    $isLeader_j \leftarrow false$;
  else
    $leader_i \leftarrow leader_j$;
    $isLeader_i \leftarrow false$;
  end

Algorithm 1: Electing a leader from a population of agents with identifiers drawn from the set of Natural numbers.

Assuming interactions in which every possible pair of interactions occurs infinitely often, after a sufficient number of interactions only one agent will have a true value for $isLeader$. Unfortunately we know of no deterministic way for the agents to know when the election is complete. One approach to determining probabilistically that the election process is complete is to use timer agents (e.g., as in [1]), but this will slow down the election process. In the following we analyze the complexity of the above leader election algorithm, and explore properties of epidemics that can help the agents detect the completion of the leader election process efficiently.

We first show, as in [1], that using the epidemic method of Algorithm 1, all agents can be expected to have the same $leader_i$ value in $\mathrm{O}(n \ln n)$ interactions, thus properly identifying the new leader of the population.

Call the agent with the highest identifier the eventual leader. Every agent with a $leader$ value equal to that of the eventual leader (although they do not know this) in the population is called a $follower$. (Using this definition, the eventual leader is also a follower. Note also that an agent's leader value may first have already been changed to some other agent's identifier prior to the agent becoming a follower.) Agents that are not followers are called *free agents*. The process of changing a free agent into a follower is called a $conversion$. Given that the agents are unaware of n, a technique is required that allows the eventual leader and others to decide when the conversion of all agents is complete. We begin by proving certain properties of the epidemic:

Theorem 1. Using Algorithm 1 all agents are expected to have the same $leader_i$ value after a total of $\mathrm{O}(n \ln n)$ interactions.

Proof For simplicity we assume that the n agents are numbered from 1 to n and that self-interaction of agents is allowed. Note that this last assumption can only increase the expected number of interactions required by an algorithm. Consider an interaction in the agent population when i agents are followers. Then the probability that this round will increase the number of followers by one is given by

$$\begin{aligned} P_{bi} &= p(\text{follower meets non-follower}) + \\ &\quad\ p(\text{non-follower meets follower}) \\ &= \tfrac{i}{n} * \tfrac{n-i}{n} + \tfrac{n-i}{n} * \tfrac{i}{n} \\ &= 2\tfrac{i}{n}\tfrac{n-i}{n}. \end{aligned}$$

Since the probability above is a geometric distribution, the expected number of rounds required to increase the number of followers by 1 is given by

$$E_{bi} = \frac{1}{P_{bi}} = \frac{n^2}{2i(n-i)}.$$

By linearity of expectation, summing over i provides the expected number of interactions required to convert all of the agents to followers, so

$$E_b = \sum_{i=1}^{n-1} E_{bi} = \sum_{i=1}^{n-1} \frac{n^2}{2i(n-i)}.$$

Thus,

$$E_b = \frac{n^2}{2} \sum_{i=1}^{n-1} \frac{1}{i(n-i)}$$

Simplifying the value of $\sum_{i=1}^{n-1} \frac{1}{i(n-i)}$ (see [8] Ex. 7.35, attributed therein to T. Feder)

$$E_b = \frac{n^2}{2} * \frac{2}{n} H(n-1) < nH(n) = \Theta(n \ln n)$$

where $H(n-1)$ and $H(n)$ are the $n-1^{\text{th}}$ and n^{th} Harmonic numbers respectively.

Theorem 2. Using Algorithm 1 the eventual leader is involved in an expected $\Theta(\ln n)$ conversions.

Proof When the election is complete every agent will have become aware of the identification number of the eventual leader. Call the process of an agent becoming aware of the identification number of the eventual leader a conversion. Assuming fair interactions then the probability of the eventual leader being involved in the first conversion is 1, in the second conversion $\frac{1}{2}$, in the third conversion $\frac{1}{3}$ and so on. Therefore, the expected number of conversions that the eventual leader is involved in can be expressed as E_{lca}

$$E_{lca} = \sum_{i=1}^{n-1} \frac{1}{i} < H(n) < \Theta(\ln n).$$

Theorem 3. The eventual leader is expected to be involved in $\Theta(\ln n)$ non-conversion interactions with followers before all the free agents are converted.

Proof Consider an interaction in which there are i followers. The probability that the number of followers increases in this round is P_{bi}. The expected number of rounds required to increase the number of followers from i to $i+1$ is E_{bi} This is the expected number of non-conversion rounds that occur between the $(i-1)^{th}$ and i^{th} conversions.

The probability of a non-conversion round in which the eventual leader interacts with an already converted follower is P_{b2} is given by

$$\begin{aligned} P_{b2} &= p(\text{eventual leader meets a follower}) + \\ &\quad p(\text{follower meets the eventual leader}) \\ &= \frac{1}{n}\frac{i}{n} + \frac{i}{n}\frac{1}{n} = \frac{2i}{n^2} \end{aligned}$$

We want to find the number of times that the eventual leader is expected to encounter a follower between the $(i-1)^{th}$ and i^{th} conversions. This is $P_{b2} * E_{bi} = \frac{2i}{n^2}\frac{n^2}{2i(n-i)} = \frac{1}{n-i}$. Summing this over all i gives E_{lmf}, the expected number of times a leader sees one of its followers before all free agents are converted.

$$E_{lmf} = \sum_{i=1}^{n-1} \frac{1}{n-i} = H(n-1) < H(n) = \Theta(lnn).$$

Theorems 2 and 3 show that the expected number of conversions carried out by the eventual leader is approximately equal to the number of interactions between the eventual leader and its followers. This property can be used by a potential leader to estimate if the election process is complete with some degree of confidence. When a potential leader has been involved in the same number of conversion and non-conversion interactions it can estimate that the election is probably nearly complete. In practice, rather than testing for equality it is desirable to ensure that the number of followers met exceeds the number of conversions by some margin. This buffer factor takes into account the large number assumptions in the order estimates (the expected size is $\Theta(\ln n)$) which ignores lower order terms. Thus we introduce two parameters, A and B, and have all agents in the running for leadership conclude that the election is finished when the number of conversions they have performed is less than $B + A\times$ number of non-conversion interactions.

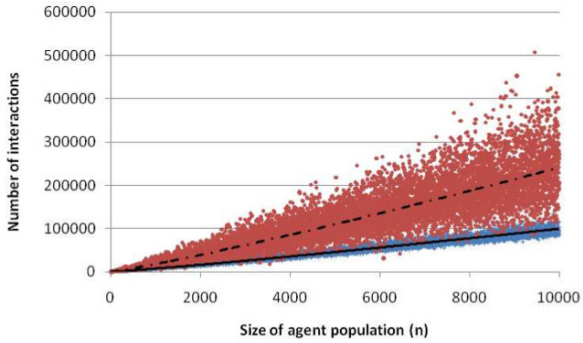

Figure 1. Election algorithm performance. For experiments with uniform random interactions we show the number of interactions required to elect a single leader is plotted for different population sizes. Blue points indicate when the election actually completed, fitted by the solid line. Red points indicate when the eventual leader agent declared the election complete, fitted by the dashed line. In these experiments, A=4, B= 0. The solid line fits the points at which the election is complete. The dashed line fits the points at which the eventual leader agent declares that the election is complete.

Algorithm 2 summarizes the basic process of leader election for numbered leaders with numbers drawn form $\mathcal{N}_0$. Algorithm 2 assumes large numbers of agents and sets the termination criteria based on using parameters A and B. A is the multiplicative factor, and B is an additive factor, used for smaller-sized populations so that the termination test requires that a minimum number of interactions has occurred. This helps to avoid cases where agents falsely conclude the election is over after only a few interactions.

III. Experimental demonstration

Figure 1 shows results of a particular experimental evaluation of the leader election and termination algorithm. Populations of various sizes from 1000 to 10000 were simulated and the leader election algorithm was run. The value used for B was 0, and for A was 4. Because the algorithm declares that the election is complete only probabilistically, this declaration may not always be correct. In the figure,

Termination Parameters: int A,B
Initially: For all agents i:;
  $leader_i \leftarrow aid_i$;
  $isLeader_i \leftarrow false$;
  $electionComplete_i \leftarrow false$;
  $conversions_i \leftarrow 0$;
  $metFollowers_i \leftarrow 0$;
Interaction: Between agents i and j;
  if $electionComplete_i$ *or* $electionComplete_j$ **then**
    $electionComplete_i \leftarrow true$;
    $electionComplete_j \leftarrow true$;
  else
    if $leader_i > leader_j$ **then**
      $leader_j \leftarrow leader_i$;
      if $leader_i = aid_i$ *% i is still possible leader* **then**
        $conversions_i \leftarrow conversions_i + 1$;
        if $B + A \times conversions_i < metFollowers_i$ **then**
          $isLeader_i \leftarrow true$;
          $electionComplete_i \leftarrow true$;
        end
      end
    else if $leader_i < leader_j$ **then**
      $leader_i \leftarrow leader_j$;
      if $leader_j = aid_j$ **then**
        $conversions_j \leftarrow conversions_j + 1$;
        if $B + A \times conversions_j < metFollowers_j$ **then**
          $isLeader_j \leftarrow true$;
          $electionComplete_j \leftarrow true$;
        end
      end
    else
      if $leader_i = aid_i$ **then**
        $metFollowers_i \leftarrow metFollowers_i + 1$;
      end
      if $leader_j = aid_j$ **then**
        $metFollowers_j \leftarrow metFollowers_j + 1$;
      end
    end
  end

Algorithm 2: Election with termination.

blue points indicate the point at which the election was actually complete. Red points indicate the point at which the eventual leader agent actually determined that the election was complete. A solid line fits the blue responses and a dashed line fits the red ones. For smaller values of n (under 3335) the algorithm terminated correctly in over 99.2 per cent of cases, improving to 99.8 per cent for the largest third of the populations simulated.

IV. Conclusion

Leader election is a critical problem in a range of tasks from network controllers to autonomous agents. Elections have long been studied in collections of agents lacking unique identification numbers; however this model of agents may be limiting for many real world tasks in which simple agents are assigned unique numbers, such as MAC addresses or serial numbers. For such environments, we exhibit a probabilistic algorithm with very simple local computation for deciding that the election process has completed. The algorithm operates probabilistically in O($n \ln n$) interactions to both establish the leader of the collection of agents and decide that the election is complete. Future work will explore the development of tighter bounds on the buffer used to ensure that the algorithm is probably complete, and evaluation of the performance of the algorithm under non-uniform agent interaction schedules. This will be done both through additional simulations and a refinement of the analysis. Our preliminary experiments indicate that the value of parameter A can be chosen very close to 1, and that small values for B are usually successful, suggesting that the probabilistic statement of Theorem 3 can be improved substantially.

Acknowledgments

The financial support of NSERC is gratefully acknowledged. The authors thank Eric Ruppert for helpful discussions.

References

[1] D. Angluin, J. Aspnes, Z. Diamadi, M. J. Fischer, and R. Peralta, "Computation in networks of passively mobile finite-state sensors," in *PODC '04: Proceedings of the twenty-third annual ACM symposium on Principles of Distributed Computing*. New York: ACM, 2004, pp. 290–299. [Online]. Available: http://dx.doi.org/10.1145/1011767.1011810

[2] R. Chaturvedi, "Electing and maintaining leaders in populations of autonomous agents," Master's thesis, York University Department of Computer Science and Engineering, 2010.

[3] R. Chaturvedi, P. Dymond, and M. Jenkin, "Efficient leader election among numbered agents," in *Proc Int. Conf. Data Engineering and Internet Technology*, March 2011. [Online]. Available: http://vgrserver.cs.yorku.ca/ jenkin/papers/2011/rahuldeit2011.pdf

[4] D. Angluin, J. Aspnes, and D. Eisenstat, "Fast computation by population protocols with a leader," in *DISC*, ser. Lecture Notes in Computer Science, S. Dolev and S. Dolev, Eds., vol. 4167. Berlin, Heidelberg: Springer, 2006, pp. 61–75. [Online]. Available: http://dx.doi.org/10.1007/11864219_5

[5] K. P. Birman, M. Hayden, O. Ozkasap, Z. Xiao, M. Budiu, and Y. Minsky, "Bimodal multicast," *ACM Trans. Comput. Syst.*, vol. 17, no. 2, pp. 41–88, May 1999. [Online]. Available: http://dx.doi.org/10.1145/312203.312207

[6] P. Eugster, R. Guerraoui, A. M. Kermarrec, and L. Massoulie, "From epidemics to distributed computing," *IEEE Computer*, vol. 37, no. 5, pp. 60–67, May 2004.

[7] R. Guerraoui and E. Ruppert, "Even small birds are unique: Population protocols with identifiers," Dept of Computer Science and Engineering, York University, Tech. Rep., September 2007. [Online]. Available: http://infoscience.epfl.ch/record/111565

[8] R. Graham, D. Knuth, and O. Patashnik, *Concrete mathematics*, 2nd ed. Reading, MA: Addison-Wesley, 1994.