An Infection Algorithm For Leader Election: Experimental Results For A Chain*

Michael Jenkin and Patrick Dymond
Electrical Engineering and Computer Science
York University, Canada
(jenkin,dymond)@eecs.yorku.ca

***Abstract*— A variant of the population protocol model is used in [1] to describe a probabilistic algorithm for leader election (choosing one of the agents to have special authority) in a collection of autonomous numbered agents, with only simple, pairwise interactions allowed between them. Earlier results have shown that leader election for a collection of numbered agents can be accomplished via a probabilistic infection algorithm. The algorithm uses no external or global timers to decide when the election is completed. Here we consider agents distributed through different locations in space. We consider agents as being located on various nodes of a graph and allow only those agents at the same node of the graph to interact. Experimental results using various sizes of chain graph [2] illustrate that the probabilistic algorithm presented in [1], [3] can be successfully applied in this generalized setting.**

***Index Terms*— robotics, leader election, infection, population protocol**

I. Introduction

Communicating information among members of a robotic team can be straightforward if all agents can communicate with each other all the time. This task becomes much more challenging if we assume more limited and realistic communication between team members. To make this problem concrete, suppose that it is desired to elect or choose a leader among a group of autonomous agents where instantaneous and simultaneous communication among the agents is not assumed. How can this team of agents go about electing their leader? To specify the problem suppose that each agent is preassigned a unique positive *id* number, and that the eventual leader is to be that one agent with the largest *id* number. The election process then is simply the process of each agent learning the value of the maximum assigned identification number. The agent whose number matches this number is the leader, and all other agents are not. If we had access to simultaneous broadcast among the agents, then election would be straightforward. Each agent would broadcast their number, and then listen to all other broadcasts and the problem would be solved. If however, the agents can only talk to each other individually in pairs, and there is no way of deterministically ensuring that each agent communicates with every other agent, then the problem becomes more complex. This type of interaction model is similar to the population protocol model introduced by Angluin et al. [4]. Population protocols were developed to model sensor networks in which agents have very limited resources and no control over their movements or communications. Key features of our variant of the population protocol model are:

- Agents: The system consists of a large population of identical simple agents.
- Pairwise interactions: Agents do not send messages or share memory. Instead, an interaction between two agents involves comparing the id numbers of the two agents, and possibly updating the contents of their registers based on the results. Each interaction is atomic and independent of other activity within the collection of agents.
- Probabilistic interaction and movement patterns: Individual agents have no direct control over their interactions or location; instead these are determined probabilistically.
- Convergence and termination: Agents converge to the correct value at the end of any computation. In population protocols the agents may be unaware that the computation is finished, but in the algorithm presented here, with high probability, all agents will be able to ”know” when a leader has been elected.

Angluin et al. developed a leader election algorithm in their *population protocol model* that takes $\Theta(n^2)$ [1] pairwise agent interactions, assuming “fair” interactions (in which every pair of agents interacts infinitely often.) The population protocol described in [4] assumes that the agents are anonymous and finite state. For many real world applications the agents are not anonymous but rather are equipped with unique serial or identification numbers as mentioned above. In [1], [3] we modified Angluin et al.'s approach by assigning to each agent a unique identifier or serial number. Interactions then may involve comparison or exchange of these serial numbers by a pair of interacting agents. We allow agents to remember a fixed number of serial numbers and to contain counters. Interactions occur randomly between paris of agents, as in the population protocol model.

*Research support by Natural Science and Engineering Research Council

[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 $c_1 n^k \leq f(n) \leq c_2 n^k$.

A key assumption of our earlier algorithm is that pairs of individual agents interact with each other with uniform probability: each agent is equally likely to interact with any other agent. But for real robot populations, interaction probability is more likely tied to geographical closeness: one agent is more likely to communicate with another that is closer than with one that is farther away. The notion of distance could be operationalized in many possible ways. Here we define a topological measure of distance by considering the agents as situated on nodes of a graph, and allowing interactions only between pairs of agents located at the same node. After presenting the leader election algorithm from [3] in the next section, we extend the model to a topological world that incorporates the location and movements of the individual agents. Section IV provides some experimental results of the algorithm on a simple class of topological maps, namely chain graphs.

II. Leader election for simple 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. (This value is unknown at the start of the process.) Throughout the process, each agent i maintains a local variable $leader_i$ for recording its current estimate of the id of the agent to be ultimately selected as leader, 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 $(leader_i, isLeader_i) = (id_i, true)$. A leader election computation is a sequence of interactions, each between a pair of agents. For each interaction two agents are chosen uniformly at random and they interact using the algorithm described below. When two agents interact, the higher-numbered agent "infects" 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 gradual infection the value of the highest identifier in the population is propagated among the agents. Assuming fair interactions among agents, the population eventually elects a unique leader using this method. The process is described in Algorithm 1. 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 is done in [4]), but this will slow down the election process. Chaturvedi et al. developed an alternative approach

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
    $leader_i \leftarrow leader_j$
    $isLeader_i \leftarrow false$

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

for estimating that the election is complete based on properties of the interactions among the agents. In the following we analyze the complexity of this leader election algorithm, and explore properties of epidemics that can help the agents detect the completion of the leader election process efficiently.

As shown in [4], using the epidemic method of Algorithm 1, all agents can be expected to have the same $leader_i$ value in $O(n \ln n)$ interactions, thus properly identifying the new leader of the population with high probability. 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.) Agents that are not followers of the leader 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 the expected number of pairwise interactions before all agents have the same $leader_i$ value is $O(n \ln n)$.

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 would only increase the expected number of interactions required by an algorithm. Consider an interaction in the agent population when i agents are followers. The probability that this round will increase the number of followers by one is given by

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

Since the probability above is a geometric distribution, the

expected number of rounds required to increase the number of followers by 1 is

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

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

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

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

Simplifying the value of $\sum_{i=1}^{n-1} \frac{1}{i(n-i)}$ (see [5] Ex. 7.35, attributed therein to T. Feder) yields

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

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, denoted 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.

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

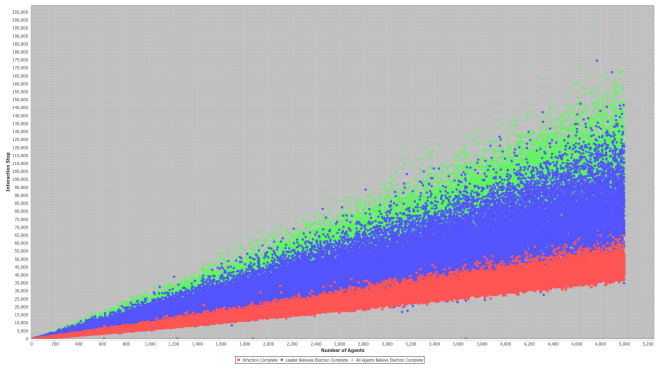


Fig. 1. Election algorithm performance. For experiments with uniform random interactions, the number of interactions required to elect a single leader is plotted for different population sizes. (Each population size is simulated ten times.) Red points indicate when the election actually completed, blue points indicate when the leader believed the election to be complete, and green points indicate when all of the agents believe the election to be complete. The simulation ends when all of the agents believe the election to be complete. If this occurs prematurely, before all agents have been infected by the agent with the largest id, then the election has failed, and the point of completion is plotted as a zero value. (This is an election in error.) Over these simulations the average number of times the algorithm ended in error was 0.86%. In these simulations $A = 2$, $B = 5$.

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 has exceeded the number of conversions made 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. It also takes into account unlikely events that could occur initially during the election process. Thus

Algorithm 2:
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
          $isLeader_i \leftarrow true$
          $electionComplete_i \leftarrow true$
    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
          $isLeader_j \leftarrow true$
          $electionComplete_j \leftarrow true$
    else
      if $leader_i = aid_i$ **then**
        $metFollowers_i \leftarrow metFollowers_i + 1$
      if $leader_j = aid_j$ **then**
        $metFollowers_j \leftarrow metFollowers_j + 1$

Algorithm 2: Election with termination.

we introduce two parameters, A and B, and have each agent in the running for leadership conclude that the election is finished when the number of agents it has modified is less than $B + A\times$ the number of its non-conversion interactions. Algorithm 2 summarizes the basic process of leader election for numbered leaders with numbers drawn from $\mathcal{N}_0$. Algorithm 2 assumes large numbers of agents and sets the termination criteria based on comparing the parameters A and B. Figure 1 shows results of a particular experimental evaluation of the leader election and termination algorithm. Populations of various sizes from 10 to 5000 were simulated and the leader election algorithm was run. The value used for B was 5, and for A was 2. Each population size was

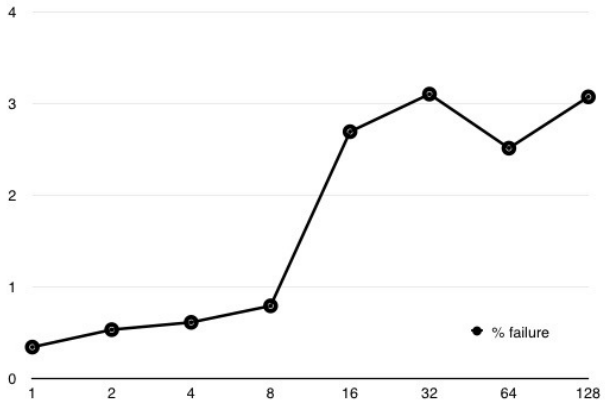


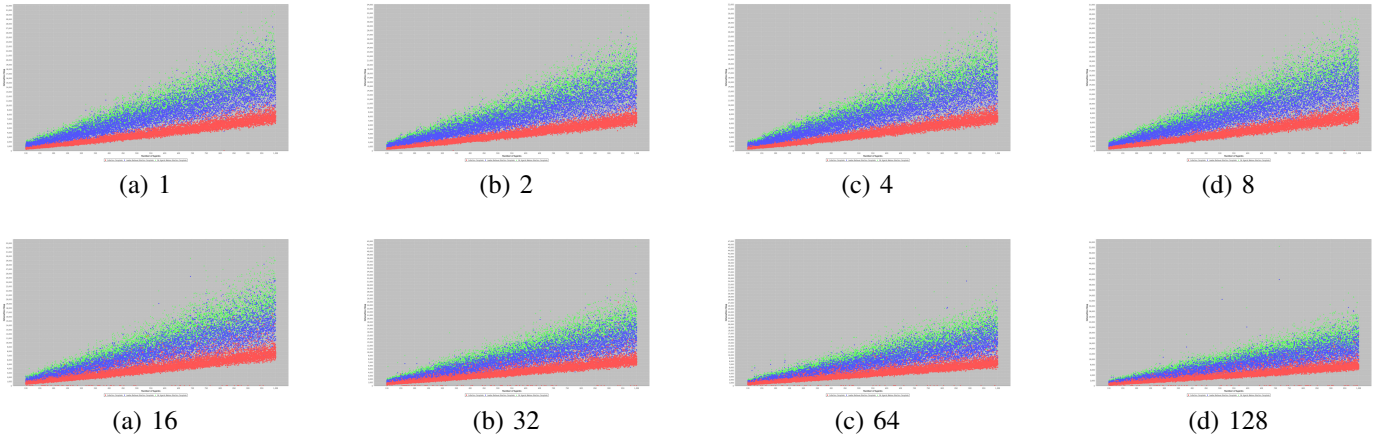
Fig. 2. Failure rate as a function of chain length. Even with a chain length of 128, the failure rate was below 5%.

simulated 10 times. Because the algorithm declares that the election is complete only probabilistically, this declaration may not always be correct. In the figure, red points indicate the number of interactions at which the election was actually complete. Blue points indicate the point at which the eventual leader agent actually decided that the election was complete. Green points indicate the point at which all agents decided that the election was complete. If all of the agents believed that the election was complete before it actually was (i.e. an erroneous conclusion was reached) the actual number of interactions required was plotted as zero.

III. Topological extension

The basic leader election algorithm described above assumes that in any interaction the participating agents are chosen uniformly at random. If agents are distributed in different locations then it is likely that their interaction probability will in some way be a function of the distance between them. There are many different ways in which this interaction probability could be formulated. Here, we use a topological model. We assume that there is an underlying directed graph $G(V, E)$ and that each agent is located at some vertex V_i. Each directed edge leaving vertex i $E_{i,j} = (V_i, V_j)$ is assigned a transition probability $P_{i,j}$ such that $\sum_j P_{i,j} = 1$. This is the probability that an agent located in node i that chooses to *move* to a new vertex (rather than interacting with another agent at the same vertex), would choose to follow edge $E_{i,j}$.

Each agent A_k is located at some specific vertex, and can only interact with other agents present in the same vertex as agent A_k. Some initial distribution of the agents among the graph vertices is assumed. Centrally, we assume a global clock as in the initial population model. At each tick of the clock either an interaction or motion event is chosen with probabilities p and $1 - p$ respectively. If a motion event is



(a) 1 (b) 2 (c) 4 (d) 8

(a) 16 (b) 32 (c) 64 (d) 128

Fig. 3. Performance of the election algorithm on chains of different sizes. Population sizes of 100 through 1000 agents are shown. $A = 2$. $B = 6$. Probability of interaction is 0.5 and all edges are equally probable. Initial agent distribution was uniform over all of the nodes.

chosen then a single agent is chosen to move randomly from its current location based on the probability of the edges outbound from the agent's current node. If an interaction event is chosen then two agents in the same node are chosen randomly to interact.

IV. Experimental results

Figure 3 illustrates the performance of the algorithm of a range of different agent populations on chain graphs of varying size using a simulator constructed in [2]. Here the number of nodes in the chain varies from 1 to 128 while the number of agents range from 100 to 5000. Agents are distributed uniformly across the chain. For all trials $A = 2$ and $B = 6$, and each trial was repeated 10 times. Situations in which the entire population believed that the election was complete even though it was not are marked with true election times of zero. Although the original self-terminating leader election algorithm was intended to operate under a situation in which any pair of agent-agent interactions is equally likely, in the new distance=based setting failure rates are still very low even for chains as long as 128 elements. Figure 2 plots election failure rate as a function of the number of chain elements. Even for quite long chains the failure rate was only approximately 10 times the failure rate for a chain of length one.

V. Discussion and future work

Leader elections has long been studied in collections of simple 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 have shown a probabilistic algorithm with very simple local computation for deciding when the election process has completed. Although originally designed to operate in environments within which interactions between individual agents was equiprobable, the algorithm performs reasonably well in environments within which interaction probability is a function of distance. In this paper a chain graph was introduced in order to model separation between various locations of agents.

The algorithm operates probabilistically using $O(n \ln n)$ 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 theoretical bounds on the buffer used to ensure that the algorithm is probably complete, as well as the theoretical performance of the algorithm under non-uniform agent interaction schedules. 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.

Acknowledgment

The authors thank Rahul Chaturvedi and Omeed Safee-Rad for helpful contributions.

References

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

[2] O. Safee-Rad, "Infection algorithms for distributed agents with variable graph topologies," 4080 Undergraduate Project, York University Computer Science and Engineering, supervised by Patrick Dymond and Michael Jenkin, 2014.

[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, 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

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