Laplacian-Based Consensus on Spatial Computers

Nelson Elhage
MIT CSAIL
77 Massachusetts Ave
Cambridge, MA USA
nelhage@mit.edu

Jacob Beal
BBN Technologies
10 Moulton Street
Cambridge, MA USA
jakebeal@bbn.com

ABSTRACT

Robotic swarms, like all spatial computers, are a challenging environment for the execution of distributed consensus algorithms due to their scale, diameter, and frequent failures. Exact consensus is generally impractical on spatial computers, so we consider approximate consensus algorithms. In this paper, we show that the family of self-organizing protocols based on the graph Laplacian of a network[19] are impractical as well. With respect to the structure of a finite-neighborhood spatial computer, we find that these protocols have an expected convergence time of $O(diameter^2)$ when the inputs are strongly correlated with location. Verifying this result in simulation, we further determine that the constant factor on the convergence time is high, rendering Laplacian-based approximate consensus unsuitable for general use on spatial computers.

Categories and Subject Descriptors

I.2.11. [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent Systems*

General Terms

Algorithms

Keywords

Spatial computing, Proto, Amorphous computing

1. INTRODUCTION

Multi-robot systems are intimately tied to the problem of consensus. Many of the basic activities of groups of robots, such as moving together in a coherent group, choosing between possible tasks, and electing leaders, can be viewed as distributed consensus problems, in which a collection of networked devices agree on a single representative value derived from the initial inputs of all the participating devices. Large swarms of robots, however, are a challenging environment for the execution of distributed consensus algorithms due to their scale, diameter, and frequent failures.

A key part of the challenge comes from the spatially-correlated network structure typical of robotic swarms with short-range communication. Although distributed consensus is a generally well-studied topic in computer science, there has been little exploration of the practicalities of distributed consensus on these sorts of *spatial computers*—potentially large collections of devices distributed to fill a space, where communication between devices is highly localized. A robotic swarm is one example of a spatial computer; others include sensor networks, peer-to-peer wireless, pervasive computing, and engineered bacterial colonies. Distributed consensus tends to arise as a problem on other spatial computers as well, and viewing the problem through this common framework will allow results in the area of robotic swarms to apply across all these domains as well.

Exact consensus is generally impractical on spatial computers, so we consider approximate consensus algorithms. One attractive possible approach to approximate consensus is the family of self-organizing protocols based on the graph Laplacian of a network[19], which compute an averaging-based consensus. In recent years, theoretical study using the tools of algebraic graph theory and control theory have showed these protocols to be robust and exponentially convergent (Olfati-Saber et al. summarize a number of key mathematical results in their survey of the field [18]). Moreover, these Laplacian-based consensus algorithms have been successfully applied to many networked systems, including robotic tasks such as flocking and swarming (e.g. [17]), formation control (e.g. [7]), and shape-formation in a modular robot (e.g. [22]).

Despite their theoretical elegance and promising applications, however, we find that these algorithms scale poorly on finite-neighborhood spatial computers. Analysis shows that when inputs are strongly correlated with location (which is the typical case on spatial computers), they have an expected convergence time of $O(diameter^2)$ with respect to the structure of the spatial computer—far from the desired lower bound of $O(diameter)$. Verifying this result in simulation, we further determine that the constant factor on the convergence time is high, rendering Laplacian-based approximate consensus unsuitable for general use on spatial computers.

2. SPATIAL COMPUTING

Spatial computing is the study of collections of local computational devices distributed through a physical space, in which:

- the difficulty of moving information between any two devices is strongly dependent on the distance between them, and

Cite as: Laplacian-Based Consensus on Spatial Computers, Nelson Elhage, Jacob Beal, *Proc. of 9th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, van der Hoek, Kaminka, Lespérance, Luck and Sen (eds.), May, 10–14, 2010, Toronto, Canada, pp. XXX-XXX.

- the "functional goals" of the system are generally defined in terms of the system's spatial structure.

Systems that can be viewed as spatial computers are abundant, both natural and man-made—examples include wireless sensor networks, animal and robot swarms, reconfigurable hardware like FPGAs, biological embryos, peer-to-peer wireless, pervasive computing, and engineered bacterial colonies.

The roots of spatial computing as a unified field lie in amorphous computing, biological modelling, pervasive computing, and reconfigurable computing. Amorphous computing[1] considers the programming of myriad simple and error-prone devices arranged in an irregular locally-connected network. Amorphous computing approaches are often biologically inspired, as in the pattern-formation work of Coore[5] and Nagpal[14]. This is also the source of the continuous space abstraction[3] and Proto programming language[4] that we use for simulations in this paper. At the same time, the challenge of modelling developing biological systems led to the development of the MGS topological language[8], and a general inquiry into the relationship between space and computation[9], while consideration of scaling in reconfigurable computing (e.g. [6]) and the long-term implications of pervasive computing (e.g. [23]) also converge on the same spatial problems for study.

2.1 Formal Definition of a Spatial Computer

In this work, we consider the consensus problem in the context of finite-neighborhood spatial computers that communicate via short-range broadcast.

Formally, let us take the definition of a spatial computer to be the combination of:

- A set of n devices in a weighted graph $G = \{V, E\}$, with a weighting function w that maps each edge $e_{i,j} \in E$ to a non-negative real number.
- A Riemannian manifold M with distance function d (a manifold is a space that is locally Euclidean; a Riemannian manifold also provides geometric properties like angles and surface integrals)
- A mapping $p : V \to M$ of each device to a unique position in the manifold

with the additional constraint that the weight $w(e_{i,j}) = O(1/d(p(i), p(j)))$. In other words, the weight of edges decreases at least linearly with distance (except for a possible initial transient).

A finite-neighborhood spatial computer is one in which there is a fixed range r such that $d(p(i), p(j)) > r$ implies that $w(e_{i,j}) = 0$. In other words, there are no direct connections between devices farther than some fixed range away from one another.

A short-range radio broadcast network, as is commonly used in sensor robotic swarms, is an example of a finite-neighborhood spatial computer: even though radio waves propagate without limit, beyond some fixed range there is effectively zero communication. If modelled as a unit disc network (a severe simplification), the weight function is

$$w(e_{i,j}) = \begin{cases} 1 & \text{if } d(p(i), p(j)) \leq r \\ 0 & \text{if } d(p(i), p(j)) > r \end{cases}$$

Although it is not formally included in the definition, the case in which spatial computers are most interesting is generally when both n and distances between most devices may be large. For example, a large robotic swarm performing search and rescue, might include thousands of devices and an extended "pseudopod" of the swarm might be hundreds of hops long.

As is the case in amorphous computing[1], we commonly assume that the large scale of a spatial computing system means that at any given instant there is a high likelihood that an error is occurring somewhere within the system—devices are crashing or misbehaving, messages are being dropped, new devices are being added. The specifics of the error model vary from system to system, but in all of them the robustness of the system is assumed to be constantly challenged.

3. DISTRIBUTED CONSENSUS

In distributed consensus, every device i begins with its own initial value $x_i(0)$ and within finite time comes to agree on the same value that every other device agrees on, where that value depends in some way on the set of initial values.

In an exact consensus algorithm, the values may be discrete and there is some time t following which every node has decided on precisely the same value. Unfortunately, even though recent developments such as quorum consensus[2] have greatly reduced the cost and fragility of available algorithms, there are a plethora of lower bounds and impossibility results (see [12] for some key examples) that render exact consensus generally impractical given the large scale, diameter, and frequent errors typical of spatial computers. For example, maintaining a quorum consensus algorithm on a changing collection of devices requires frequent updates of the quorums, and if any update fails the algorithm is irrevocably stalled.

We thus turn instead to approximate consensus algorithms, which evade these difficulties by weakening the agreement conditions. One particularly successful family of approximate consensus algorithms are Laplacian-based *average-consensus* algorithms[19, 18]. These solve a specific variety of distributed consensus problems: given real-number initial values, the values held by every device converge asymptotically toward the mean of the initial values—making it possible to establish that there is some time t after which the difference in values is always less than a desired error e. In recent years, these have been successfully applied to a number of spatial computing problems at relatively small scale, such as flocking and swarming (e.g. [17]), sensor fusion (e.g. [21, 20]), shape-formation in modular robotics (e.g. [22]), and formation control for multi-robot systems (e.g. [7])

Let us briefly summarize the general algorithm and relevant convergence results, as presented in [18], which reviews many variants of the algorithm and a number of key mathematical results. For the spatial computer model under consideration, the applicable variant is the following discrete-time consensus algorithm:

$$x_i(k+1) = x_i(k) + \epsilon \sum_{j \in N_i} w(e_{i,j}) \cdot (x_j(k) - x_i(k)) \qquad (1)$$

where $x_i(t)$ is the value at device i at time t, N_i is the set of device i's neighbors, and $w(e_{i,j})$ is the weight of the edge from i to j. The constant $\epsilon > 0$ is the step size of the algorithm. The analysis of Olfati-Saber and others indicates

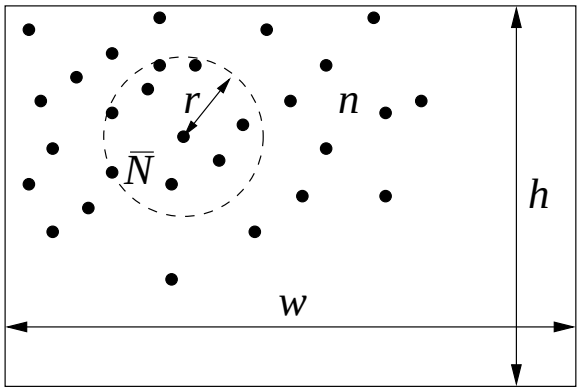


Figure 1: A spatial computer with unit disc communication and uniform random distribution of devices through a rectangular area can be described with five parameters: width w and height h, number of devices n, radius of communication r, and expected number of neighbors $\bar{N}$.

that in order for the algorithm to converge, we require $\epsilon < \frac{1}{\Delta}$, where Δ is the maximum degree of the network. If this condition is satisfied, for an undirected graph, Algorithm (1) asymptotically converges for all initial conditions to an average-consensus of $x_i(\infty) = \frac{1}{n}\sum_i x_i(0)$.

The convergence speed of Algorithm (1) is governed by the eigenvalues of the graph Laplacian matrix for the connectivity graph of a given network. Let $A = [w(e_{i,j})]$ be the adjacency matrix for the connectivity graph, and let $D = \text{diag}(d_1, \ldots, d_n)$ be the degree matrix, with diagonal elements $d_i = \sum_{j \neq i} w(e_{i,j})$. The matrix $L = D - A$ is known as the graph Laplacian of the graph, and governs the dynamics of the consensus algorithm.

In particular, the values converge exponentially from their initial values toward the average with a rate constant of $\mu_2 = 1 - \epsilon\lambda_2$, where the size of the disagreement $\delta(k)$ at time k is bounded above by $\mu_2^k||\delta(0)||$ and λ_2 is the second eigenvalue of the graph Laplacian.

This convergence rate is not as good as it may first appear. Although it converges exponentially with respect to the initial disagreement, the critical λ_2 that regulates the speed of the convergence is dependent on the structure of the graph, and may vary greatly. The available bounds on the eigenvalues of the graph Laplacian are not very tight when applied to spatial computers. Letting Δ denote the maximum degree of a graph, [15] gives wide bounds in terms of the graph's diameter:

$$\frac{4}{n \cdot diameter} \leq \lambda_2 \leq \Delta - 2\sqrt{\Delta - 1} + \frac{2\sqrt{\Delta - 1} - 1}{\lfloor diameter/2 \rfloor}$$

The author notes that, for fixed n, large diameter corresponds to small λ_2, but does not attempt to precisely characterize the relationship. [13] and [11] demonstrate some additional bounds, but they are also insufficiently tight to accurately estimate the order of the convergence time for large spatial computers. The existing theoretical results on convergence rate of Laplacian-based approximate consensus are thus insufficient on their own to determine whether this algorithm is practical for spatial computers.

4. ANALYSIS

Lacking sufficiently tight theoretical bounds on λ_2, we instead begin our analysis from the perspective of the spatial computer.

Assuming unit disc communication and a uniform random distribution through a rectangular area, our spatial computer has five parameters—number of devices n, expected number of neighbors $\bar{N}$, radius of communication r, the width w and height h of the the distribution area (Figure 1). There are only four degrees of freedom, though, since expected number of neighbors can be derived from n, r, and the area. The expected diameter of the network can also be derived from these parameters (using the relations derived in [10]).

The Laplacian-based consensus algorithm adjusts the local function value by a small fraction of the difference between it and its neighbors. When the algorithm runs on a spatial computer that approximates space well ($\bar{N} \geq 6$, per [10]) with a small ϵ, we may thus expect it to behave like diffusion. We now consider the time that the algorithm takes to converge such that every device is expected to be within some fixed error e of the true average.

Diffusive phenomena scale time-wise with the square of distance—as is to be expected given the relationship of molecular diffusion and random walks. Thus, if the initial distribution of $x_i(0)$ values is spatially correlated across a distance c, it should require time proportional to c^2 for all values in the system to converge to within e of the average. In the general case of spatial correlation, then, the convergence time is proportional to $diameter^2$.

Since the algorithm's update rule is not normalized with respect to the number of neighbors, that will also have an effect on the convergence time. Double the neighbors means double the amount of change for a fixed ϵ, so we may expect an inversely proportional relationship between $\bar{N}$ and the time to converge to within e of the average. As $\bar{N}$ increases, the diameter may be expected to shrink slightly as well (per [10]).

The effect from the initial difference $\delta(0)$ and the update constant ϵ are, of course, just as before. Putting all of these terms together, we find that to a first order of approximation the expected time for the values of a spatial computer to converge to within e of the true average is:

$$O(\frac{diameter^2 \cdot \ln(\delta(0))}{\bar{N} \cdot \epsilon})$$

The value of ϵ, however, is related to the number of neighbors. If Δ is the highest degree of any device in the network, then the algorithm is only guaranteed to converge if ϵ is less than $1/\Delta$ [18]. Thus we have

$$\epsilon < 1/\Delta \leq 1/\bar{N}$$

$$\bar{N} \cdot \epsilon < 1$$

Assuming that a larger ϵ (and thus a faster convergence time) is always preferred, we may assume that $\bar{N} \cdot \epsilon$ is bounded below by some constant as well, and therefore our final bound on expected convergence time is:

$$O(diameter^2 \cdot \ln(\delta(0)))$$

where *diameter* is the only variable relating to the structure of the spatial computer.

5. EXPERIMENTAL VERIFICATION

In order to verify our analysis, and to establish the range of constant factors for finite-neighborhood spatial computers, we turn to experiments in simulation. We performed a series of experiments to verify the predictions of the previous section with respect to the asymptotic scaling of the convergence time.

We begin with an experiment to confirm the effect of correlating the initial conditions with spatial position. Having verified that convergence times scale badly for networks with spatially correlated initial conditions, we carried out experiments for each relevant variable in the analysis above to confirm its effect.

We confirmed the quadratic effect of diameter in two separate experiments. We first varied n and w while fixing $\bar{N}$, thus varying the diameter with n. In the second experiment, we fixed $\bar{N}$ and n, but changed the shape of the space by varying l and w, to confirm that aspect ratio has no significant effect on the scaling from spatial correlation. In both cases we observed the expected quadratic dependence.

We also ran experiments varying $\bar{N}$, and confirmed both the expected direct linear effect, as well as the smaller effect from the changing network diameter. Finally, we ran experiments varying ϵ and $\delta(0)$ and observed the expected effects on the convergence time.

5.1 Experimental Setup

We implemented the approximate consensus algorithm in Proto[4], a language for programming aggregate behavior on spatial computers. In Proto, computation is specified as operations on fields, where a field is a function mapping each point in space to a value. The implementation of the algorithm in Proto is:

```
(def consensus (epsilon init)
   (rep val init
      (+ val
         (* epsilon
            (sum-hood (- (nbr val) val))))))
```

This **consensus** function takes two arguments, the consensus parameter ϵ and the field assigning initial values to each device. The **rep** construct creates a variable **val**, which will hold the current approximate consensus value at each device. This starts at the initial value **init** for each device, then evolves according to the consensus law.

Simulations were then performed in the MIT Proto network simulator using a unit disc model of communication, in which each device communicates with all neighbors within r meters.

Except when noted otherwise, experiments were performed with $n = 2000$ devices uniformly randomly distributed on a rectangular $h = 100$ meter by $w = 250$ meter space, with the radio range r set to give an expected $\bar{N} = 10$ neighbors per device (approximately 6.3 meters). Devices executed asynchronously, transmitting once per second. We used a conservative step size of $\epsilon = 0.02$, which is expected to be well within the convergent region for the algorithm (although in occasional high $\bar{N}$ cases the random distribution may cause violation of the $1/\Delta$ convergence constraint). Initial conditions are set with $x_i(0) = 0$ in the left half of the space and $x_i(0) = 50$ in the right half of the space—a worst-case spatial arrangement for diffusion. Note that this means the convergence time is expected to be regulated by width rather than the full diameter (along the diagonal), since the initial conditions are vertically homogeneous.

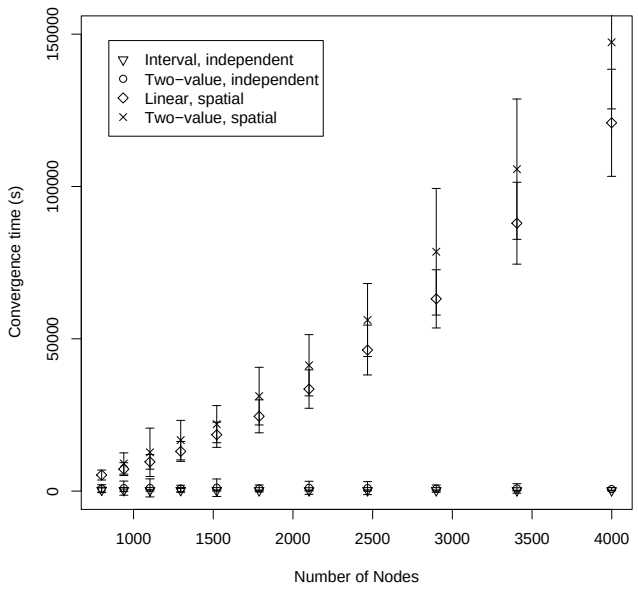


Figure 2: When initial values of devices are strongly correlated with a device's location in a spatial computer, convergence time scales much worse than when they are independent of position.

In every case, we measure end-to-end convergence time of the algorithm, recording how many simulated seconds t it takes for 95% of all nodes in the network to reach a value within $e = 2$ units of the actual mean value based on the known initial conditions. Since each simulated node updates once per simulated second, these convergence times in seconds are thus equal to the number of update cycles of the algorithm.

For all plots, the data points shown are the average of the convergence time for 50 runs with the same parameters. Error bars are drawn at $\pm$ two standard deviations of the distribution for the same 50 experiments.

5.2 Initial

We performed an initial empirical exploration to confirm the importance of correlating the initial condition of a node with its spatial position. We varied n from 800 to 4000 nodes, scaling w linearly with n from 100 up to 500, and thus varying the width while maintaining $\bar{N} = 10$ expected neighbors. At each point, we ran simulations under 4 different sets of initial conditions:

1. $x_i(0)$ chosen uniformly randomly from the interval $[0, 50]$.
2. $x_i(0)$ set to 0 or 50 with 50% probability.
3. $x_i(0)$ varying linearly from 0 to 50 with the horizontal position of the node.
4. $x_i(0)$ set to 0 in the left half of the space and 50 in the right half.

For networks with randomized initial conditions, uncorrelated with position we find consistently fast convergence

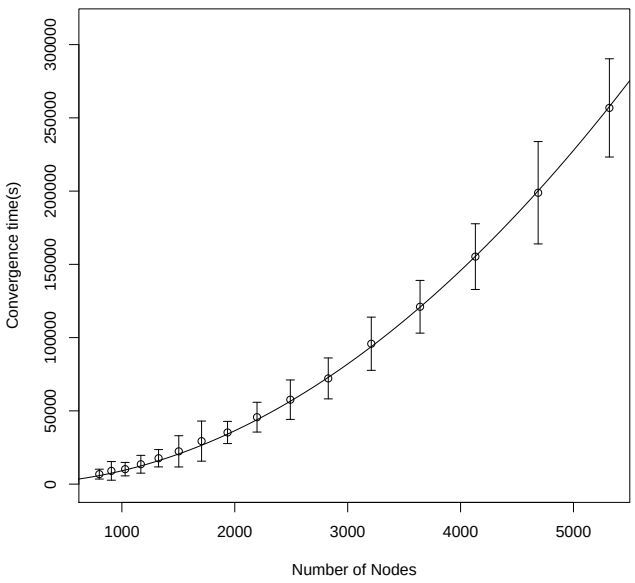


Figure 3: Convergence time scales quadratically as a function of network width. This plot shows a quadratic fit for number of nodes where width scales with number of nodes, holding expected number of neighbors fixed.

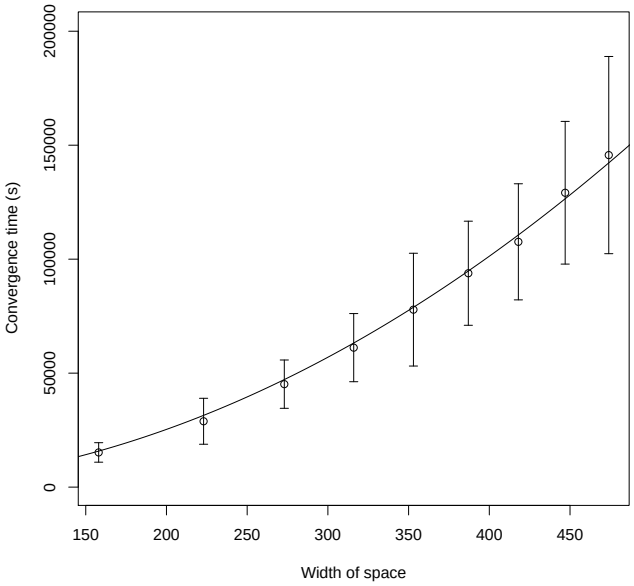


Figure 4: Convergence time depends on width, not number of devices. This plot shows a quadratic fit for width on simulations of $n = 2000$ devices in a fixed area with varying aspect ratio.

times, independent of network size (Figure 2). When initial conditions are correlated with the position of the node, however, we see the predicted quadratic scaling.

In both cases, changing the initial conditions between ones that only have $x_i(0)$ at either 0 or 50 and ones that are smoothly distributed across the range has only a small effect on the convergence time, as the diffusive process eliminates drastic local differences quickly, but then takes a long time to smooth out smaller disagreements across space.

5.3 Scaling with Network Diameter

We next performed a series of experiments to verify the consensus algorithm's scaling characteristics with network diameter. We varied n and the width of the space together, fixing $\bar{N} = 10$ expected neighbors while varying the diameter. We varied n between 800 and 6000 nodes, distributed randomly over a space varying from $100m \times 100m$ to $750m \times 100m$. Given $\bar{N} = 10$, by the results of Kleinrock and Silvester [10], we can estimate the effective width as varying from about 20 hops to around 170 hops.

As predicted, convergence time scales quadratically with width. Figure 3 shows the results along with a best-fit quadratic regression. Note, however, that even for modest numbers of hops the number of rounds to converge is in the thousands. The quadratic term of the fit shown in Figure 3 is $11.3w^2$, measuring w in expected hops. Although this is not a massive constant, it does mean that that impact of the quadratic scaling is increased by another order of magnitude—and cannot be significantly improved upon due to the $\epsilon < 1/\Delta$ convergence constraint.

5.4 Effect of Aspect Ratio

In order to verify that the quadratic scaling is an effect of the network width, rather than n, we performed another series of experiments. In these, we fixed the number of nodes at $n = 2000$, and varied the shape of the space on which they were distributed, between a $158m \times 158m$ square to a $1 : 10$ rectangle $500m \times 50m$. By our analysis, we expect convergence time to be a function of the width of the space – since the process is symmetrically distributed across the height, only the horizontal distance is relevant.

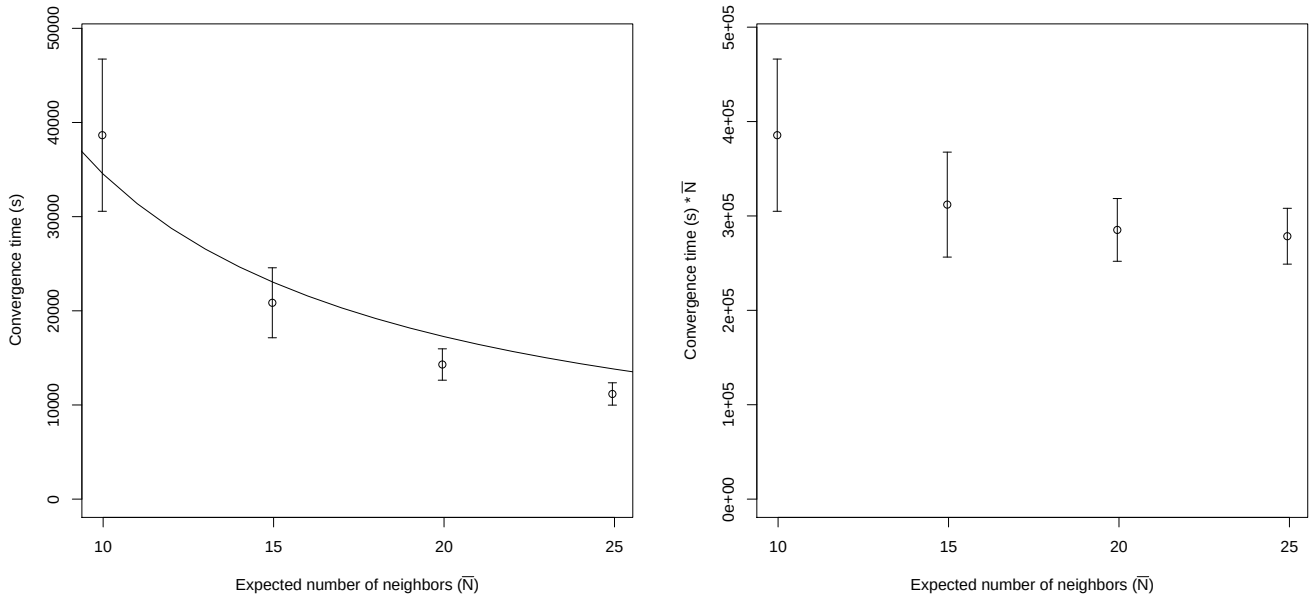
We once again observe a quadratic dependency on the width of the network, even though all experiments have the same number of devices (Figure 4). Note that the although the mean is regulated by width, the variance increases as the height becomes smaller. This is to be expected given that smaller height means less randomly distributed nodes in the cross-section of the spatial computer.

5.5 Effect of Degree

As further verification of the effect of diameter, we performed a third series of simulations, in which we fixed the space on which nodes were distributed, but varied the number of nodes, in order to vary $\bar{N}$ between 10 and 25.

We expect to observe two effects by changing the number of neighbors. First, as noted in Section 4, there should be an inverse linear effect from number of neighbors. Second, there should be a smaller effect due to the decreasing of the diameter of the network. As the number of neighbors increases, the expected progress towards any destination gained by a single radio hop also increases gradually [10], decreasing the hop distance between the edges of the space.

We confirmed both effects in our simulations. Figure 5(a)



(a) (b)

Figure 5: End-to-end convergence time (a) scales primarily with the inverse number of neighbors (inverse fit shown by line), plus a smaller factor from higher $\bar{N}$ increasing the expected progress per hop and therefore decreasing diameter slightly. In (b) we plot $\bar{N}$ against $t \cdot \bar{N}$ to highlight the secondary effect from higher $\bar{N}$.

plots $\bar{N}$ against t, showing the inverse linear relationship between number of neighbors and convergence time. In Figure 5(b), we plot $\bar{N}$ against $t \cdot \bar{N}$, removing the expected inverse linear effect of $\bar{N}$ and highlighting another smaller effect from decreasing the diameter of the network, with diminishing gains as the progress factor approaches 1.

5.6 Effects of Step Size and Initial Disagreement

We performed an additional pair of experiments to confirm that well-understood aspects of the approximate consensus algorithm worked as believed in our simulations. In the first experiment, we examined the effect of the step size ϵ. With all other parameters fixed, we recorded convergence time for values of ϵ ranging between 0.01 and 0.1.

As analyzed in Section 4, we expect an inverse linear impact from the step size in the convergence process, so long as the system converges. By the results of Olfati-Saber and others, we also know that the system is guaranteed to converge only for $\epsilon < 1/\Delta$. With an expected average degree of 10, some nodes will have degrees potentially much greater, and so we should expect instability some time before $\epsilon = 0.1$.

The results of the experiments with ϵ appear in Figure 6. For values of $\epsilon > 0.08$, more than half the experiments diverged $x_i(\infty) = \pm\infty$. For $\epsilon = 0.08$, 5 of 50 runs diverged; those are omitted from the data plotted in Figure 6.

The curve shows a best-fit $t \sim \frac{1}{\epsilon}$ curve. It fits within the error bars, but there appears to be some systematic error, as the curve consistently undershoots all points for $\epsilon \geq 0.03$. As the system approaches the unstable region, there seems to be some small second-order effect in ϵ, perhaps coming from oscillation of devices with high degree. We have not attempted to precisely characterize this effect.

As a final confirmation of the behavior of the process, we ran a series of experiments on the effect of the difference between values in the initial conditions. Fixing all other variables, we varied $x_i(0)$ for all nodes in the right half of the space between 10 and 100. Because Algorithm (1) converges exponentially quickly, we expect a logarithmic increase in convergence time with an increase in the step size.

The result of the experiments is shown in Figure 7. A best-fit logarithmic curve is plotted, matching the data well.

6. CONTRIBUTIONS

Although Laplacian-based averaging consensus is an attractive approach to implementing distributed consensus on spatial computers, analysis shows that it has an expected convergence time of $O(diameter^2)$ to reach a fixed amount of error, with respect to the structure of the spatial computer. Experimental verification further determines that the constant factors on this convergence rate are relatively large, rendering these algorithms generally impractical for large spatial computers.

There are two basic paths by which the impact of this disappointing result might be alleviated. On the one hand, even a very small fraction of non-spatial links added to a spatial computer may greatly shrink the effective diameter. Preliminary investigations indicate that adding 1% or less non-spatial links may make these algorithms competitive with the theoretical best performance of a purely spatial algorithm on computers more than a few hundred hops in diameter—a result not unexpected given the small-world findings of [16]. For those application domains where it is

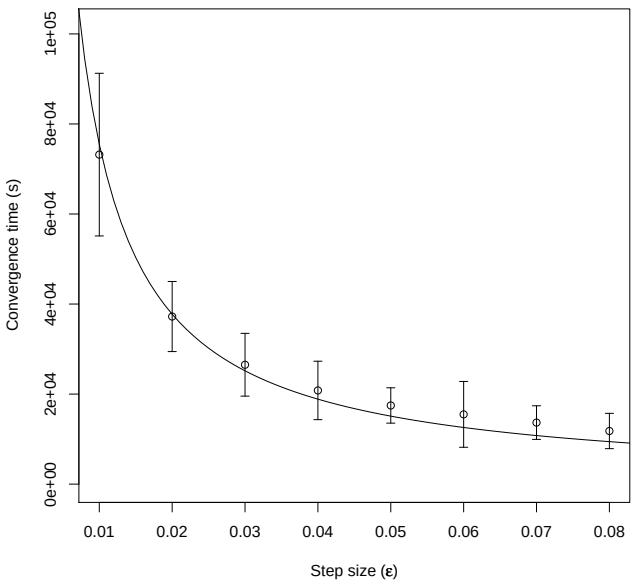


Figure 6: Convergence time is inverse linear with step size ϵ, though this relationship breaks down as the system approaches instability. The line shown is an inverse linear fit, and the system is generally unstable for $\epsilon > 0.08$.

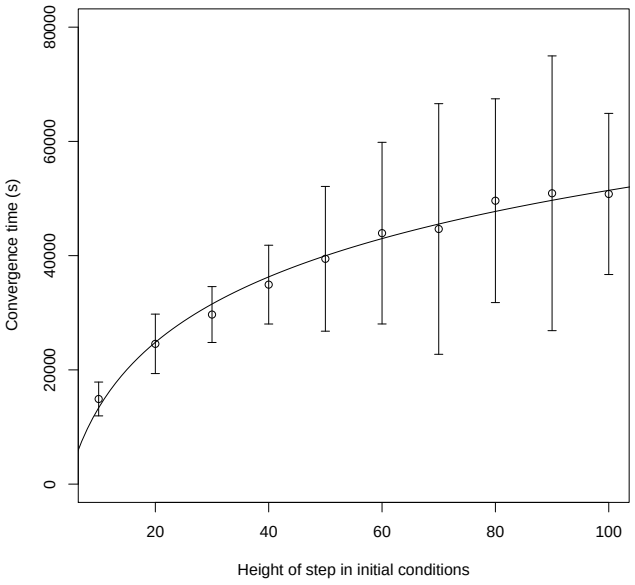


Figure 7: Convergence time scales logarithmically with the initial disagreement between nodes in the left and right halves of the distribution.

not reasonable to add non-spatial links to the computer, on the other hand, it is an open question how to design a practical algorithm that might approach the fixed lower bound of $O(diameter)$ convergence.

7. REFERENCES

[1] H. Abelson, D. Allen, D. Coore, C. Hanson, G. H. y, T. Knight, R. Nagpal, E. Rauch, G. Sussman, and R. Weiss. Amorphous computing. Technical Report AIM-1665, MIT, 1999.

[2] H. Attiya, A. Bar-Noy, and D. Dolev. Sharing memory robustly in message-passing systems. *Journal of the ACM*, 42:124–142, January 1995.

[3] J. Beal. Programming an amorphous computational medium. In J.-P. B. et al., editor, *Unconventional Programming Paradigms 2004 (LNCS 3566)*, pages 121–136. Springer-Verlag, 2005.

[4] J. Beal and J. Bachrach. Infrastructure for engineered emergence on sensor/actuator networks. *IEEE Intelligent Systems*, 2006.

[5] D. Coore. *Botanical Computing: A Developmental Approach to Generating Interconnect Topologies on an Amorphous Computer*. PhD thesis, MIT, 1999.

[6] A. DeHon. Very large scale spatial computing. In *Third International Conference on Unconventional Models of Computation*, pages 27–37, 2002.

[7] M. Egerstedt and X. Hu. Formation constrained multi-agent control. *IEEE Trans. on Robotics and Automation*, 17(6):947–951, 2001.

[8] J.-L. Giavitto, C. Godin, O. Michel, and P. Prusinkiewicz. Computational models for integrative and developmental biology. Technical Report 72-2002, Univerite d'Evry, LaMI, 2002.

[9] J.-L. Giavitto, O. Michel, J. Cohen, and A. Spicher. Computation in space and space in computation. Technical Report 103-2004, Univerite d'Evry, LaMI, 2004.

[10] L. Kleinrock and J. Silvester. Optimum transmission radii for packet radio networks or why six is a magic number. In *NTC '78; National Telecommunications Conference, Birmingham, Ala., December 3-6, 1978, Conference Record. Volume 1. (A79-40501 17-32) Piscataway, N.J., Institute of Electrical and Electronics Engineers, Inc., 1978, p. 4.3.1-4.3.5.*, volume 1, pages 4–+, 1978.

[11] M. Lu, L. zhu Zhang, and F. Tian. Lower bounds of the laplacian spectrum of graphs based on diameter. *Linear algebra and its applications*, 420:400–406, 2007.

[12] N. Lynch. *Distributed Algorithms*. Morgan Kaufman, 1996.

[13] B. Mohar. The laplacian spectrum of graphs. In Y. Alavi, G. Chartrand, O. R. Oellermann, and A. J. Schwenk, editors, *Graph Theory, Combinatorics, and Applications*, volume 2, pages 871–898. Wiley, 1991.

[14] R. Nagpal. *Programmable Self-Assembly: Constructing Global Shape using Biologically-inspired Local Interactions and Origami Mathematics*. PhD thesis, MIT, 2001.

[15] M. W. Newman. The laplacian spectrum of graphs. Master's thesis, University of Manitoba, 2000.

[16] R. Olfati-Saber. Ultrafast consensus in small-world networks. *American Control Conference, 2005. Proceedings of the 2005*, pages 2371–2378 vol. 4, June

2005.

[17] R. Olfati-Saber. Flocking for multi-agent dynamic systems: Algorithms and theory. *IEEE Trans. on Automatic Control*, 51(3), March 2006.

[18] R. Olfati-Saber, J. Fax, and R. Murray. Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, 95(1):215–233, Jan. 2007.

[19] R. Olfati-Saber and R. Murray. Consensus problems in networks of agents with switching topology and time-delays. *Automatic Control, IEEE Transactions on*, 49(9):1520–1533, Sept. 2004.

[20] D. P. Spanos, R. Olfati-Saber, and R. M. Murray. Approximate distributed kalman filtering in sensor networks with quantifiable performance. In *Fourth International Symposium on Information Processing in Sensor Networks*, pages 133–139, 2005.

[21] L. Xiao, S. Boyd, and S. Lall. A scheme for asynchronuous distributed sensor fusion based on average consensus. In *Fourth International Symposium on Information Processing in Sensor Networks*, 2005.

[22] C.-H. Yu and R. Nagpal. Self-adapting modular robotics: A generalized distributed consensus framework. In *International Conference on Robotics and Automation (ICRA)*, 2009.

[23] F. Zambonelli and M. Mamei. Spatial computing: An emerging paradigm for autonomic computing and communication. Technical report, Universita di Modena e Reggio Emilia, 2004.