

Accelerating Approximate Consensus with Self-Organizing Overlays

Jacob Beal
Raytheon BBN Technologies
Cambridge, MA 02138
Email: jakebeal@bbn.com

Abstract—Laplacian-based approximate consensus algorithms are an important and frequently used building block in many distributed applications, including formation control, sensor fusion, and synchronization. These algorithms, however, converge extremely slowly on networks that are more than a few hops in diameter and where values are spatially correlated. This paper presents a new algorithm, Power Law Driven Consensus, which uses a self-organizing virtual overlay network to accelerate convergence, at a cost of decreasing the predictability of the final converged value. Experimental comparison with the Laplacian approach confirms that PLD-consensus allows for drastically faster convergence in spatial networks.

I. INTRODUCTION

Approximate consensus algorithms are an important building block for many distributed algorithms, including robot formation control [1], flocking and swarming [2], sensor fusion [3], modular robotics [4] and synchronization [5]. The dominant algorithmic approach to distributed approximate consensus is a Laplacian-based approach in which each device finds a weighted local average of its own current value with the values held by its neighbors. In effect, this approach is operating like particle diffusion, such that as the differences between devices eventually equalize, the network is brought into consensus. Although this approach supports a number of elegant mathematical results [6], [7], including an exponential rate of convergence derived from the graph Laplacian, the bounds on the rate of convergence are extremely loose, and may actually indicate a very slow rate of convergence indeed.

Spatial computers, which tend to have mesh-like network structure many hops across, are an example of where Laplacian-based consensus performs poorly. As demonstrated in [8], on spatial computers, the expected convergence time is actually $O(\text{diameter}^2)$ with a high constant factor, such that Laplacian-based approximate consensus is expected to converge extremely slowly whenever there is both significant diameter and also a spatial correlation in the initial values of devices. Since many applications of approximate consensus, including those mentioned above, are commonly executed on spatial computers with spatially correlated values, this can severely limit the efficacy and applicability of consensus-based applications.

This paper introduces a new algorithm, Power Law Driven Consensus, which uses a self-organizing virtual overlay network to accelerate convergence, at a cost of decreasing the predictability of the final converged value. Following a brief

review and specification of problem context in Section II, I present the new PLD-consensus algorithm in Section III. Section IV then compares the new algorithm with the Laplacian approach in simulation, confirming that PLD-consensus allows for drastically faster convergence in spatial networks.

II. APPROXIMATE CONSENSUS ON SPATIAL COMPUTERS

A spatial computer is generally defined as any collection of devices in which the difficulty of moving information between any two devices is strongly dependent on the distance between them, and where the functional goals of the system are linked to its spatial structure. Examples include ad-hoc communication networks, swarms of unmanned aerial vehicles (UAVs), sensor networks, and colonies of engineered biological cells.

For purposes of this paper, we will consider a more restricted class of spatial computer, in which a set of n devices are arranged in a graph $G = \{V, E\}$ and also embedded by a function $p : V \rightarrow M$ into a Riemannian manifold M with distance function d . Edges are assumed to be bidirectional, to have equal weight, and not to exist between two vertices i and j if $d(p(i), p(j))$ is greater than some threshold.

Laplacian-based average consensus algorithms [7] solve a specific class of distributed consensus problem: given a real-number initial local value $l_i(0)$ for each device i , Laplacian-based consensus computes the approximate mean of $l_i(0)$ by iterative applying the transformation:

$$l_i(t+1) = l_i(t) + \epsilon \sum_{j \in \mathcal{N}(i,t)} l_j(t) - l_i(t) \quad (1)$$

where $\mathcal{N}(i, t)$ is the set of graph neighbors of device i at time t and the constant $\epsilon > 0$ is the step size of the algorithm. For simplicity, we give only the synchronous specification; only minor modification is required for a more general non-synchronized algorithm.

Laplacian-based consensus has been proven [7] to converge exponentially toward the mean value of $l_i(0)$, but the rate of convergence is set by the second eigenvalue of the graph Laplacian, which is very small for high-diameter mesh networks. In fact, as shown in [8], Laplacian-based consensus performs badly on spatial computers, with a convergence time that is $O(\text{diameter}^2)$ and a high constant factor due to the generally large potential number of neighbors.

When there is no correlation in the distribution of values on devices, this does not matter as much, since a rough estimate of

global values can be made from locally sampled information. When values are correlated by their location in the network, however, no reasonable estimate of consensus can be made without moving information over long distances. Since spatial computers often have a large diameter and values are often highly correlated with location, a faster algorithm is clearly needed.

III. POWER LAW DRIVEN CONSENSUS

If the problems with Laplacian-based consensus come from the high diameter of the network, then one clear approach to accelerating consensus would be to reduce the effective network diameter through construction of an overlay network containing long-distance links.

Power Law Driven Consensus is a simple implementation of such an approach, where the overlay network self-organizes using a $1/f$ distribution to break symmetry, selecting certain devices to have their values spread over a longer distance. Under PLD-consensus, devices compete to become “dominant” over one another, drawing their bids for dominance from the scale-free $1/f$ distribution. The value of using a scale-free distribution of this sort is that, no matter the size or arrangement of the network, the expected distribution contains one device that receives a value large enough to dominate the network. Note that this is an initial version, and the process can likely be optimized significantly through modulation of this distribution.

Dominance decays over both time and space, so this competition results in a partition of the network into “dominance regions” that shift over time. Values then spread outward from dominant devices through the regions that they dominate, and each device blends the local dominant value with its own value.

Since this distribution is scale-free, the regions of dominance begin small, allowing local blending of values, much like Laplacian-based consensus. The regions then expand rapidly until the entire network is dominated by a single device, effectively reducing diameter and allowing for rapid convergence. Finally, because dominance decays over time, any dominant device that fails will eventually be replaced by other dominant devices.

A. Formal Algorithm Specification

Having giving some intuitions for how PLD-consensus will function, we will now provide a formal specification of the algorithm. For simplicity, this specification will be stated in terms of synchronous rounds. In fact, however, there is no requirement for synchrony, both the overlay construction and blending operations are relaxation methods, meaning that any local updates (within the stable range) is expected to move the system as a whole closer to a converged state. The algorithm is thus expected to operate well under a wide range of non-synchronous conditions as well (indeed, the simulations in Section IV are non-synchronous).

Let us begin the specification by considering the computation of the dominance overlay. This sub-algorithm serves two

functions: first, determining the relative dominance level of each device and second, flowing values down the dominance gradient from more dominant to less dominant devices.

The dominant value state for each device i at each round t is a tuple $(d_i(t), u_i(t), v_i(t))$ of the current dominance level $d_i(t)$, a tie-breaker unique identifier $u_i(t)$, and a dominant value $v_i(t)$. In addition, each device i has a local value $l_i(t)$, which is its current candidate for a consensus value.

At each round, the algorithm considers three sources for the new dominant value state. The driven state S_d is:

$$S_d = (\lfloor \frac{1}{u(0,1)} \rfloor, u_i(t), l_i(t)) \quad (2)$$

where $u(0,1)$ is a uniform random distribution over the interval $(0,1)$. Taking the inverse of $u(0,1)$ produces $1/f$ -noise, giving a power-law distribution of candidate new dominance levels being injected into the network at each round.

For each neighbor $n \in \mathcal{N}(i,t)$, where $\mathcal{N}(i,t)$ is the set of neighbors of device i at time t , the neighbor-derived state S_n is:

$$S_n = (d_n(t) - 1, u_n(t), v_n(t)) \quad (3)$$

which takes the neighbor’s value at a decremented dominance level.

Finally, if no neighbor has a higher dominance level, the leader state S_l is:

$$S_l = (L_i, u_i(t), l_i(t)) \quad (4)$$

where the leader dominance L_i is

$$L_i = \begin{cases} d_i(t) - 1 & \text{if } \forall n \in \mathcal{N}(i,t), d_i(t) > d_n(t) \\ 0 & \text{else} \end{cases} \quad (5)$$

which decrements the old dominance level but inserts the new local value.

The new state is then set to whichever of these three sources has the highest dominance level:

$$(d_i(t+1), u_i(t+1), v_i(t+1)) = \text{Lmax}(S_d \cup S_l \cup \{S_n | n \in \mathcal{N}(i,t)\}) \quad (6)$$

where Lmax is a lexicographic maximum, such that the first elements of a tuple are compared, then if they are equal the second elements are compared, etc.

Figure 1 shows an example of the regions of dominance produced by this computation. Initially, no devices are dominant over their neighbors, but as dominance levels are injected via the $1/f$ -noise, regions of dominance grow rapidly until eventually some device is able to dominate the entire network.

Once the dominance overlay has been established, computation of consensus is relatively straightforward. Given an initial local value of $l_i(0)$ at each device i , the value at round t may be computed as a simple proportional blend:

$$l_i(t) = \alpha \cdot v_i(t) + (1 - \alpha) \cdot l_i(t-1) \quad (7)$$

where $v_i(t)$ is the dominant value as provided from the overlay, and α is the proportional blending constant. Note,

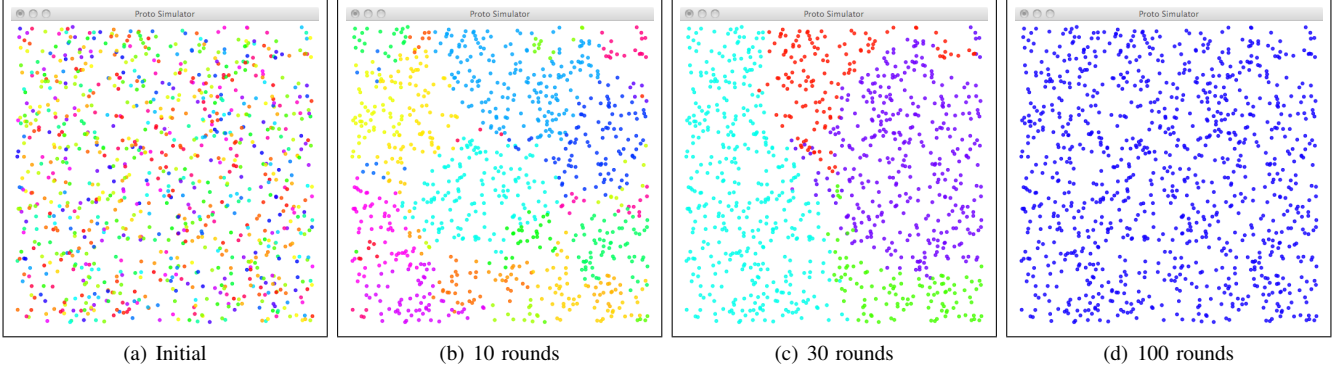


Fig. 1. Visualization of dominance competition driven by $1/f$ -noise, by having each device spread a unique RGB color computed from its UID. From an initial state where no devices are dominant (a), regions of dominance are expected to grow rapidly (b,c) until eventually some device is able to dominate the entire network.

```
(def dominant-value (value)
  (3rd
    (rep ;; Declare three state variables; only third will be returned
      (tup dominance id spread-value)
      ;; Initialize to no dominance, local ID and value
      (tup 0 (mid) value)
      (let* ;; Three options for finding dominant value:
          ;; 1: Drive new dominance with 1/f noise
          ((local (floor (/ 1 (rnd 0 1))))
           ;; 2: Take a decremented dominance from a neighbor
           (hoodmax (max-hood
                     (nbr (tup (- dominance 1)
                               id spread-value))))
           ;; 3: Decrement old dominance, but use new value
           (leader (= (2nd hoodmax) (mid))))
          ;; Use whichever option has greater dominance
          (if (or leader (> local (1st hoodmax)))
              (tup (max local (1st hoodmax)) (mid) value)
              hoodmax))))))
```

Fig. 2. Proto code for computing dominant value. Comments are in blue, state variables in green.

however, that updating in this way uses only overlay values and no neighbor values at all.

To soften this non-locality, PLD-consensus mixes overlay blending and the local blending of Laplacian-based consensus by the simple expedient of adding the standard Laplacian differential term:

$$l_i(t) = \alpha \cdot v_i(t) + (1 - \alpha) \cdot l_i(t - 1) + \beta \sum_{n \in \mathcal{N}(i,t)} w(e_{i,n}) \cdot (l_n(t - 1) - l_i(t - 1)) \quad (8)$$

where β is the step size for Laplacian-based consensus. PLD-consensus thus makes use of both neighbor values and values delivered through the overlay, and reduces to pure overlay blending when $\beta = 0$ and to Laplacian-based consensus when $\alpha = 0$.

IV. EXPERIMENTAL VALIDATION

In this section, I present experimental validation of the PLD-consensus algorithm in simulation. These experiments have two aims: first, to quantitatively compare the performance

```
(def PLD-consensus (initial alpha beta)
  (rep
    value ;; Declare one state variable: current consensus value
    initial ;; Initialize consensus with local initial value
    ;; Find regionally-dominant value ...
    (let ((dominant (dominant-value value)))
      ;; ... and blend incrementally with local value ...
      (+ (+ (* alpha dominant) (* (- 1 alpha) value))
         (* beta (sum-hood (- (nbr value) value))))))
```

Fig. 3. Proto code for PLD-consensus algorithm. Comments are in blue, state variables in green.

of PLD-consensus against Laplacian-based consensus, and second, to gain an initial understanding of the behavior of PLD-consensus under different configurations and conditions of execution.

For these experiments, I implemented the PLD-consensus algorithm in Proto [9], [10]. The code for creating and propagating values through the dominance overlay is listed in Figure 2, and the code for the full PLD-consensus algorithm is listed in Figure 3. Proto was a desirable language for implementation and experimental validation for two reasons: first, Proto's programming model allows a concise and direct implementation of the mathematical specification given in the previous section, and, second, the network simulator distributed with MIT Proto made it simple to run and analyze experiments on large spatial networks.

Except where otherwise noted, all experiments are run with the following parameters: the network consists of 1000 devices distributed uniformly randomly in a 100 meter by 100 meter rectangle. Devices use a unit disk model of communication, communicating with all other devices within r meters, where r is computed to give an expected 10 neighbors per device. The algorithms are run for 1000 rounds of partially synchronous execution (equal frequency, random phase), and PLD-consensus is run with $\alpha = 0.02$ and $\beta = 0$, while Laplacian-based consensus is run with a step size of $\epsilon = 0.02$. For each experimental condition, 10 trials are run. When analyzing the convergence statistics of a network, the devices

with the top and bottom 2.5% of values are ignored, to ensure that any devices that may be disconnected due to the random distribution are excluded. A network is then considered to have converged if the difference between the minimum and maximum value within the median 95% of devices is less than 1% of the initial difference (e.g., less than 0.01 if all devices start with values between 0 and 1).

A. Illustrative Comparison of Algorithms

Let us begin with an illustrative comparison of Laplacian-based consensus and PLD-consensus, to provide an initial intuition of the difference between these two approaches. For this experiment, we use a set of 1000 devices distributed uniformly randomly in a 100 meter by 100 meter rectangle. Devices can communicate via broadcast to all other devices within 7 meters, giving approximately 15 expected neighbors per device. Devices are initialized to one of two values based on their spatial location, with devices in the left hand side of the plane given an initial value $l_i(0) = 10$, and on the right hand side $l_i(0) = 30$, producing a highly spatially correlated distribution. To have the purest comparison of PLD-consensus and Laplacian-based consensus, we consider the case of $\alpha = 0.01$ and $\beta = 0$ for PLD-consensus, meaning that only dominance overlay values are used, and compare against a step size of $\epsilon = 0.01$ for Laplacian-based consensus.

Figure 4 shows the evolution of values held by the various devices over time during a single trial run. Although both cases begin the same, PLD-consensus converges rapidly, within a few hundred rounds, while even after 5000 rounds of computation Laplacian-based consensus has not converged, with a difference of 1.87 between the minimum and maximum of the median 95%—nearly 10% of the initial value difference.

The trade-off for the faster convergence of PLD-consensus, however, is a decreased accuracy in finding the overall mean value. In this case, the Laplacian consensus values at $t = 5000$ have a mean of 19.56, which is quite close to the true mean of 20. The converged value of PLD-consensus, on the other hand, is somewhat farther off at 14.35.

B. Convergence Rate

For a more thorough comparison of convergence rate, let us compare the evolution of value distributions in PLD-consensus and Laplacian-based consensus. For this experiment, trials were run for two initial distribution conditions: a homogeneous condition in which each device's initial value $l_i(t)$ is drawn uniformly randomly from the interval $[0, 1]$, and a spatially-correlated condition in which devices in the left half of the distribution start with $l_i(t) = 0$ and devices in the right half start with value $l_i(t) = 1$. Values were then recorded every 10 rounds for 1000 rounds.

Figure 5 plots the difference between highest and lowest value in the median 95% against time. In both cases, PLD-consensus clearly greatly outperforms Laplacian-based consensus. Under the homogeneous condition, Laplacian-based consensus initially converges more quickly than PLD-consensus, but greatly slows when further convergence re-

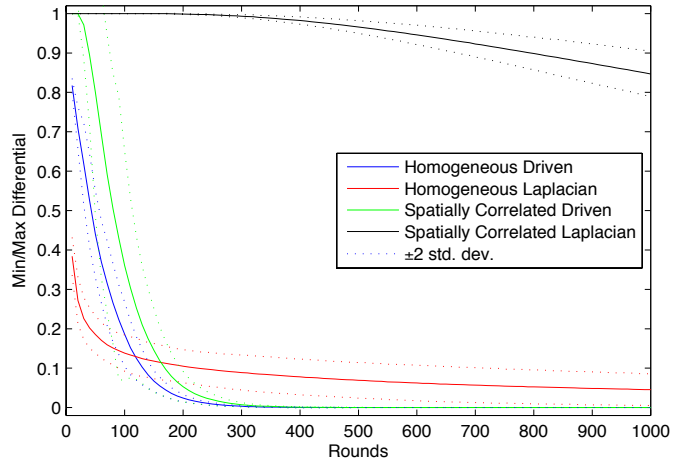


Fig. 5. PLD-consensus progresses toward convergence much faster than Laplacian-based consensus, though Laplacian-based consensus initially progresses more quickly when initial values are homogeneously distributed.

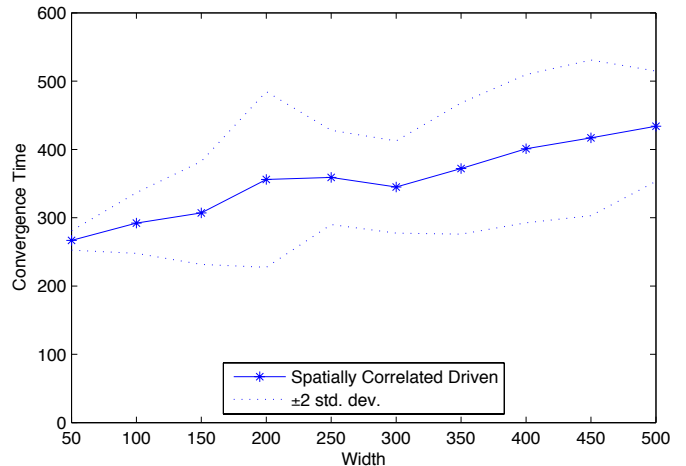


Fig. 6. Convergence time for PLD-consensus increases linearly with the width of the network.

quires equalization of values over many hops. PLD-consensus converges rapidly in both the homogeneous and spatially-correlated conditions. With spatially-correlated initial values, however, Laplacian-based consensus has barely even begun to converge by $t = 1000$, and will not complete its convergence for many thousands of rounds more, as seen in the prior illustrative comparison.

C. Scaling with Diameter and Mobility

The poor scaling of Laplacian-based consensus as network diameter increases has already been demonstrated in [8]. Since the $1/f$ -noise driving PLD-consensus is scale-free, however, the limiting factor should instead be communication time, and thus the algorithm should scale in $O(\text{diameter})$. To test this, I ran the PLD-consensus algorithm with spatially-correlated initial values on networks with dimensions of X meters width by 50 meters, where the width X ranged from 50 to 500 meters

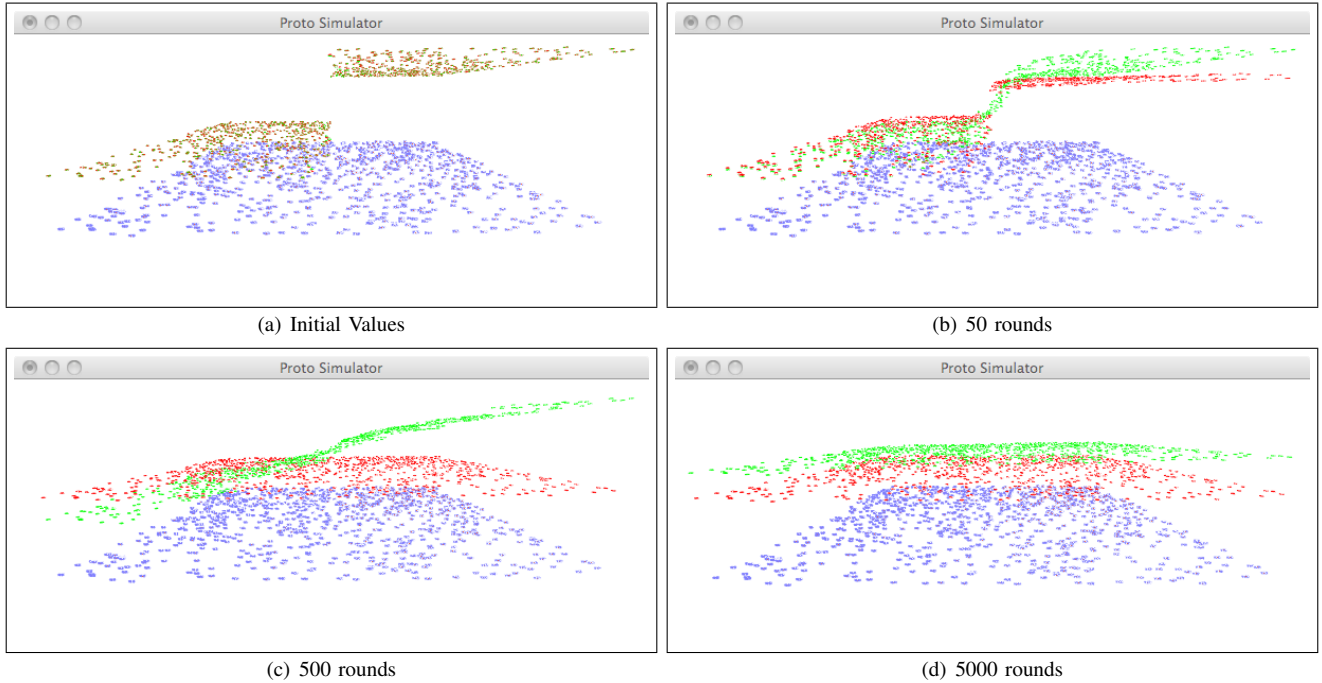


Fig. 4. PLD-consensus vs. Laplacian-based consensus on a mesh network of 1000 devices with approximately 15 neighbors each. The devices are shown as blue dots distributed in a plane, viewed at an angle, with their current values for consensus showed as the height of red (PLD-consensus) and green (Laplacian-based) dots above the plane. From an initial spatially-correlated distribution (a), PLD-consensus begins to converge rapidly (b), arriving at an approximation of the mean value within a few hundred rounds (c), while Laplacian-based consensus is eventually closer to the true mean, but even after 5000 rounds still retains nearly 10% of the initial value difference (d).

in steps of 50, and with a constant density of devices (thus ranging in number from 250 to 2500).

For this experiment, convergence is nearly universal, the only exceptions being one trial for $X = 50$ and one for $X = 100$, in which the random distribution contained too many disconnected devices. Figure 6 shows the scaling of convergence time with respect to network width X : as expected the convergence time increases approximately linearly with the diameter, atop an approximately constant base convergence time determined by the blending rate.

In many consensus applications, the participating devices are not stationary. This could both be helpful, in lowering the effective diameter of the network, and problematic, in scrambling the structure of the overlay. To investigate the effect of mobility, I ran the PLD-consensus algorithm with spatially-correlated initial values with devices moving at a velocity v varying geometrically from 0.01 to 1.0. Each device moves towards a randomly chosen point in space at velocity v , and upon reaching it (within quantization error) chooses a new point to move towards.

Figure 7 shows the results of this experiment: PLD-consensus converges for every trial, which is unsurprising since mobility means that no devices will remain disconnected. Perhaps more surprisingly, device movement does not appear to have any significant effect on convergence rate. It is likely that mobility does have some effect, but it is small enough to not be observable under these experimental conditions. Laplacian-based consensus, on the other hand, only converges

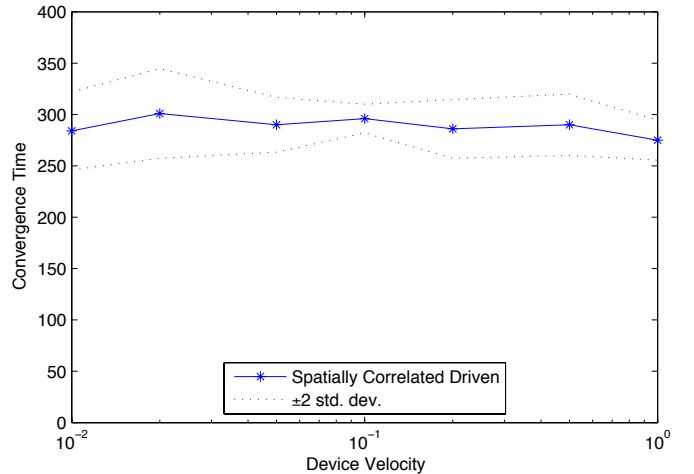
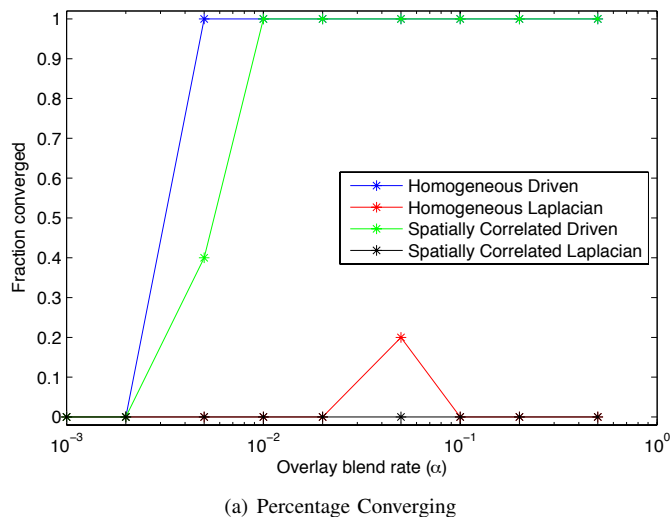


Fig. 7. Device movement does not have any significant effect on the rate at which PLD-consensus converges within the range of parameters studied.

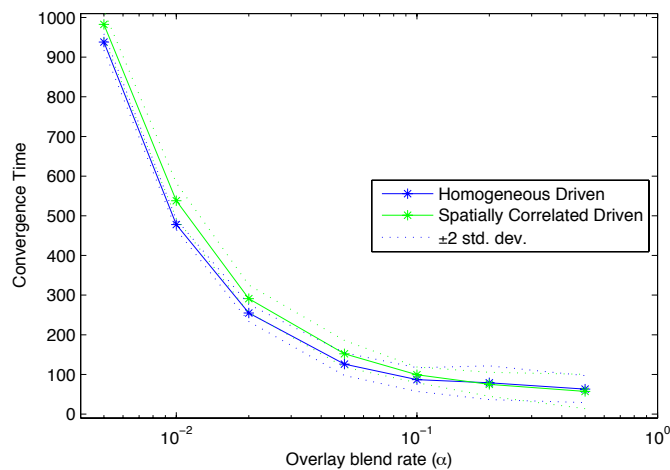
at all for $v = 1$, when 3 of the 10 trials converge just barely before $t = 1000$.

D. Effect of α and β Parameters

Finally, we consider the effects of the blending parameters α and β . For both of these parameters, there is a tension between speed and stability: the higher they are, the faster the network moves towards convergence, but if they are too high then it may become unstable.



(a) Percentage Converging



(b) Time of Convergence

Fig. 8. PLD-consensus converges reliably for all $\alpha \geq 0.01$, while Laplacian-based consensus nearly never converges (a). When PLD-consensus converges, the convergence time appears to be dominated by blend rate for low α and by overlay construction for high α (b).

To study the effect of varying α , I ran the PLD-consensus algorithm with spatially-correlated initial values, holding $\beta = 0$ while ranging α geometrically from 0.001 to 0.5. The experiment for β is the same, except that $\alpha = 0.02$, and β ranges from 0.001 to 0.5. For comparison with α variation, Laplacian-based consensus is run with a step size ϵ that is varied identically.

Figure 8 shows the results of varying α . PLD-consensus converges reliably for all $\alpha \geq 0.01$. For those trials that converge, the convergence time for low α is close to inversely proportional to α , indicating that convergence is likely dominated by blend rate. For high α , convergence time is nearly flat, indicating that convergence is likely dominated by overlay construction. Laplacian-based consensus, on the other hand only converges at all for two moderate-alpha trials: with low α it converges too slowly and with high α it becomes unstable and values diverge rapidly.

Variation of β , on the other hand, produces no significant effect on either the convergence time or on the converged value, until $\beta = 0.1$, where the Laplacian component becomes unstable and values diverge. These results indicate that the Laplacian component is likely to be operating so slowly on spatially-correlated distributions as to have no measurable effect at the diameter studied by this experiment; only at lower α or diameter is there likely to be a significant effect.

V. CONTRIBUTIONS AND FUTURE WORK

This paper has presented a new approximate consensus algorithm, PLD-consensus, which uses a self-organizing overlay network to accelerate consensus. This allows much faster convergence, at a cost of higher expected deviation from the mean of the network's initial values.

This new algorithm has the potential for a major improvements across a wide class of consensus-based applications. Further study is required, however, before such applications can be made. For example, some applications may depend more on obtaining an accurate mean of initial values, while others may be able to tolerate more inaccuracy in exchange for speed. Likewise, this paper considered only the problem of one-shot approximate consensus, while applications such as flocking depend on tracking of a changing consensus over time. Similarly, PLD-consensus only considers the simplest overlay structure and means of combining Laplacian- and overlay-based consensus approaches; more sophisticated approaches may be able to produce greatly improved performance.

REFERENCES

- [1] M. Egerstedt and X. Hu, "Formation constrained multi-agent control," *IEEE Trans. on Robotics and Automation*, vol. 17, no. 6, pp. 947–951, 2001.
- [2] R. Olfati-Saber, "Flocking for multi-agent dynamic systems: Algorithms and theory," *IEEE Trans. on Automatic Control*, vol. 51, no. 3, March 2006.
- [3] L. Xiao, S. Boyd, and S. Lall, "A scheme for asynchronous distributed sensor fusion based on average consensus," in *Fourth International Symposium on Information Processing in Sensor Networks*, 2005.
- [4] C.-H. Yu and R. Nagpal, "Self-adapting modular robotics: A generalized distributed consensus framework," in *International Conference on Robotics and Automation (ICRA)*, 2009.
- [5] J.-J. Slotine and W. Wang, "A study of synchronization and group cooperation using partial contraction theory," *Cooperative Control*, pp. 443–446, 2005.
- [6] R. Olfati-Saber and R. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *Automatic Control, IEEE Transactions on*, vol. 49, no. 9, pp. 1520–1533, Sept. 2004.
- [7] R. Olfati-Saber, J. Fax, and R. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, Jan. 2007.
- [8] N. Elhage and J. Beal, "Laplacian-based consensus on spatial computers," in *AAMAS 2010*, 2010.
- [9] J. Beal and J. Bachrach, "Infrastructure for engineered emergence in sensor/actuator networks," *IEEE Intelligent Systems*, vol. 21, pp. 10–19, March/April 2006.
- [10] "MIT Proto," software available at <http://proto.bbn.com/>, Retrieved June 22nd, 2012.