CS223 Final Project:
Dynamic multi-agent average consensus via asynchronous gossip

Florian Berlinger, Moritz A. Graule [*†‡]

December 10, 2019

Abstract

The successful application of multi-agent robotic systems in search and rescue missions or the exploration of hazardous or remote environments requires average consensus when evaluating options and making decisions. We study average consensus finding in low-density swarms consisting of agents with limited communication range. In this setting, agents are required to move through their environment in order to exchange information with each other.

As locomotion is generally costly, it is important that each of the agents has a good sense of whether the swarm has reached a sufficiently accurate estimate of the true average. We built a simulator to empirically investigate the convergence of average consensus algorithms under various specifications and adapt the problem formulation and simulations to model the specific behavior of the Kilobot collective. We further leverage this simulator to test and compare various heuristics for the local detection of global convergence. Finally, we present an attempt at bounding the time required to reach average consensus in a swarm of agents that move randomly on a random graph and exchange information whenever they meet.

1 Introduction

1.1 Motivation

Multi-agent robotic systems consisting of low-cost individual agents with basic locomotion, sensing, and communication capability can be invaluable in many applications: they could be deployed to detect hazardous biological or chemical agents, to set up an ad-hoc communication network in challenging or changing environments, or in the exploration of potential new (off-world) habitats. Many of these applications require the robots to individually measure a characteristic of their environment (e.g., temperature, pH, or humidity), and then reach a consensus about the measured value across the collective. When the swarm's aim is to obtain the average value of all agent's measurements, this task is referred to as "average consensus finding" [1, 2].

If the communication range of the individual robots is impeded either through agent level limitations (e.g., low cost, low power, or a desire to remain hard to be detected by a third-party system), or through environmental factors (e.g., medium in which signal transmission is challenging, or distance-dependent disturbances), agent motion can facilitate the dissemination of information across a swarm [2, 3, 4]. Robotic

[*]The authors made equal contributions and their names appear in alphabetical order.

[†]John A. Paulson School of Engineering and Applied Sciences, Harvard University, 02139 Cambridge MA, United States.

[‡]E-mails: {fberlinger, graulem}@g.harvard.edu

motion is costly. Therefore, with moving agents, it is particularly desirable to derive local heuristics that allow each agent to reason about whether a sufficiently accurate consensus has been reached.

To empirically test various local termination heuristics (Section 3), we implemented a framework that allows us to simulate average consensus finding in robotic swarms in which information is disseminated through random agent motion. We aim to verify the most promising approaches with physical experiments on the Kilobot collective.

1.2 Related work

1.2.1 The Kilobot collective

The Kilobot collective consists of up to 1024 independent agents of diameter $d = 32mm$ that each are capable of basic communication, motion, and sensing (Figure 1). The collective has demonstrated intelligent maneuvers such as self-assembly and dispersion [5, 6]. In the current experimental setup, the agents' motion is mechanically restricted to be within an arena of size 2m $\times$ 2m. Each agent can communicate with other agents by broadcasting short messages to either one or multiple agents in its close proximity; the broadcasting range diameter is limited to $r = 3d$. When the agents do not execute a specific maneuver, each of the agents moves around in the arena according to an independent, continuous random walk with velocity v (this can be adjusted; chosen here as $v = r$ to facilitate comparisons between physical experiments and our simulations). Each agent's communication and motion is triggered by an independent Poisson process with rate parameter $\lambda = 1$.

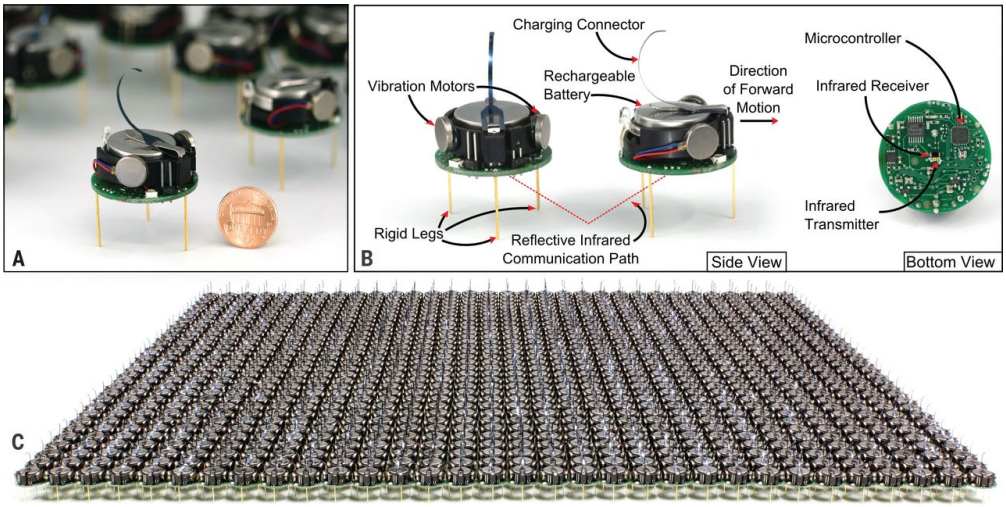

Figure 1: The Kilobot collective. An individual Kilobot agent next to a US penny for scale (A), depictions of a Kilobot and its circuit board with key mechanical and electronic components highlighted (B), and a photograph of the complete Kilobot collective (C) (adapted from [5]).

1.2.2 Average consensus finding and information dissemination

Consider the following average consensus finding task for a collective of n independent agents i: each agent is initialized with a measured value $x_i(t = 0)$ at initial time $t = 0$. The swarm's objective is to achieve a state in which each agent is aware of the true average measurement $x_{ave} = 1/n \sum_i x_i(0)$.

Boyd et al. (2006) derived lower and upper bounds on the time required to achieve average consensus in a network with stationary agents through a simple algorithm in which, upon communication between two agents, these two agents update their estimate of the average to be equal to the average of their previous estimates. Specifically, they bounded the ε-averaging time $T_{ave}(\varepsilon)$, which is defined as the smallest time required to get the normalized error within ε of 0 with high probability.

Boyd et al. (2006) present the following lower and upper bounds on $T_{ave}(\varepsilon)$:

$$T_{ave}(\varepsilon) \leq \frac{3\log\varepsilon^{-1}}{\log\lambda_2(W)^{-1}}$$

$$T_{ave}(\varepsilon) \geq \frac{0.5\log\varepsilon^{-1}}{\log\lambda_2(W)^{-1}}$$

Here, $\lambda_2(W)$ is the second largest eigenvalue of $W = \mathbb{E}[W(k)]$, and $W(k)$ is the probabilistic state transition matrix that governs the evolution of the system as

$$x(k) = W(k)x(k-1)$$

The derivation of these bounds relies on (i) a proof that the $x(k) - x_{ave}$ and $\mathrm{Var}(x(k))$ converge to 0; (ii) the fact that $\lambda_2(W) < 1$ by construction of $W(k)$; and (iii) an application of Markov's inequality to the following term:

$$\Pr\left(\frac{||x(k) - x_{ave}||}{||x(0)||} \geq \varepsilon\right)$$

Please refer to the original paper for a more detailed discussion and the definition of $W(k)$ in this setting. Boyd et al. (2006) further proposed a localized algorithm that allows agents to adjust with which neighbors in the network they communicate at what probability such that the time until average consensus has been achieved is minimized [1].

Oliva et al. (2019) built on the work by Boyd et al. (2006) to show that the same algorithm converges to the accurate average consensus in a setting where agents move on a random location graph according to a random walk and communicate with an agent selected at random from all the agents that share a node [2]. The update rule upon communication is identical to Boyd et al. (2006). Oliva et al. (2019) do not discuss the ε-averaging time in this setting, but we show in Appendix A that bounds on $T_{ave}(\varepsilon)$ can be obtained here that are very similar to the bounds presented by Boyd et al. (2006).

A problem related to average consensus finding is the study of how information disseminates across a swarm of agents following random motion if the agents can only communicate when they meet. Pettarin et al. (for one- and two-dimensional worlds) and Lam et al. (for environments with dimension ≥ 3) studied how a piece of information disseminates from one agent across a swarm of randomly initialized agents if agents communicate whenever they meet and derived probabilistic bounds on the time required until all agents are aware of the given information [3, 4].

Assuming sufficient storage capacity on board of each agent, an average consensus algorithm for moving agents is only practically useful if it achieves convergence faster than the time it would take to disseminate the initial values of all agents to all other agents in the swarm; otherwise, all agents could simply wait until they have achieved all information from the other robots and then compute the exact average value.

1.3 Report outline

We will first provide a formal definition of the task to find the average consensus with robots exhibiting the specifications seen in the Kilobots. We then introduce various heuristics that the robots may employ to locally detect whether the swarm's estimate of the average value is sufficiently close to the true average. This is followed by a section describing the implementation details of the simulator we built combined with a description of the experiments we ran and the respective results. We conclude the report by summarizing our key observations and providing an outlook on future work. Two interesting theoretical expansions of the discussion on the convergence of the algorithm studied in Oliva et al. (2019) are provided in Appendices A and B.

2 Problem statement

The restrictions on motion and communication outlined in this section are derived from the specifications of the Kilobot as discussed in Section 1.2.1. Recall: we consider a swarm of n independently controlled robotic agents i, where each of the agents is initialized to have measured the value $x_i(t=0)$ at initial time $t=0$. The swarm's objective is to achieve a state in which each agent is aware of the true average measurement $x_{ave} = 1/n \sum_i x_i(0)$.

To facilitate a formal analysis of average consensus finding through random gossipping in the Kilobot system, we abstract each agent's motion as a random walk on an undirected graph of agent locations $G = (V,E)$: The physical arena is first split into squares of size $r \times r$. Each of these squares corresponds to a node in G. These nodes are connected if an agent can move from one position to another in one average time step,i.e., two nodes are connected if their center positions are less then v/λ apart (Figure 2). In addition, we simulate the scenario in which agents move on a random location graph to enable a direct comparison with the results presented in [2].

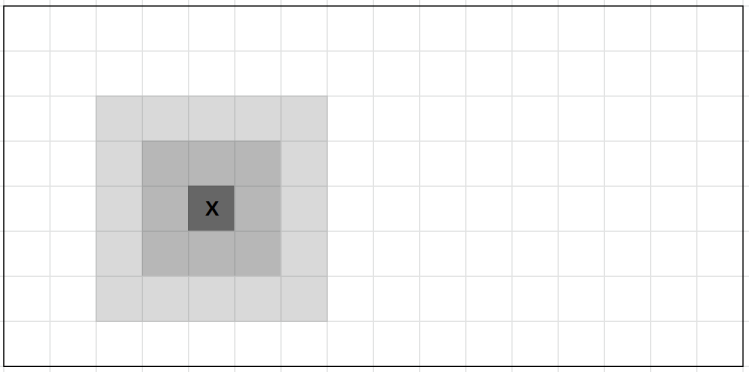

Figure 2: Location graph definition. The physical arena is split into squares that have sides as long as the maximum communication range diameter r. Each square is a node in the location graph. If the distance between two square centers is less or equal to the distance an agent can travel within the expected time between two of its local clock ticks, the corresponding nodes are connected in the location graph. For example, if $v/\lambda = \sqrt{2}r$, the node marked with an **x** would be connected to the nodes corresponding to the dark gray squares. However, if $v/\lambda = 2\sqrt{2}r$, node **x** would be connected to the dark and light grey nodes. We assume that each of the connected nodes is equally likely to be reached by one step of the agent's random walk, which is an approximation.

Each agent's communication is limited to broadcasting to one randomly selected agent on the same node. The problem statement then closely follows the definition given in [2]:

Each agents behavior is governed by an independent clock that ticks according to a Poisson process with rate $\lambda = 1$. When an agent's clock ticks, it communicates with an agent that is randomly selected from all agents that are present at the same node (an agent may select to communicate with itself). When two agents i, j communicate, they each update their estimate of the global average as $x_{i,j}(k) = (x_i(k-1) + x_j(k-1))/2$, where k is a global counter for total clock ticks across all agents. Communication and the update of $x_{i,j}$ happen on a much shorter time scale then agent motion and are assumed to be instantaneous in our model. Once the communication step is completed, the agent takes a step of a random walk on the graph G. Oliva et al. (2019) have shown that this simple update rule eventually achieves convergence to the true average measurement with a variance that approaches 0 for $k \to \infty$ [2].

3 Local convergence detection

In this first iteration, our goal is not to find heuristics with a provable performance, but instead to gain an intuition based on simulations about which simple heuristics work well in practice. This section describes various heuristics that we implemented and tested empirically. In future work, we aim to derive bounds on the performance of some of these heuristics (specifically for H0 and H2).

3.1 Heuristics

3.1.1 H0: Fixed number of interactions

In what is probably the most naive termination heuristic we implemented, each agent switches off after a fixed number of interactions, i.e., after having taken N steps. If an appropriate bound for the ε-averaging time was known for the setting with moving agents, N could be chosen such that the total steps taken across the collective is at least $T_{ave}(\varepsilon)$ for a desirable ε with high probability (we derive an upper bound on $T_{ave}(\varepsilon)$ in Appendix A). Alternatively, one could relate N to the total time required to achieve information dissemination as defined in [3].

3.1.2 H1: Absolute change in x_i

Assuming that the agents' observations are normalized (either to lie within a predefined interval, for example $[-1, 1]$, or transformed to align with samples from the standard normal distribution), a simplistic local stopping heuristic relies on the number of subsequent updates that are below a predefined threshold: if the absolute change in value estimate observed by a given agent is less than δ, its convergence counter c_i is increased by 1. If the agent observes an absolute change that is larger than δ, its counter is reset to 0. Turn the agent off if $c_i \geq m$.

3.1.3 H2: Variance estimation

Ideally, all agents would stop moving once the variance of their values is smaller than a predefined threshold. Unfortunately, this requires global knowledge, which is not available to the agents in our setting. We therefore implement the heuristic H2, in which each agent estimates the population's standard deviation (which is a function of the global time counter k) from the last m interactions with its communication partners j according to Equation 1, where $\bar{x}_i$ is the mean of the values held by the communication partners j of agent

i. An agent can then turn itself off if it estimates the population standard deviation to be smaller than a predefined threshold, i.e., $\hat{\sigma}_i < \delta$.

$$\bar{x}_i = 1/m \sum_{j=1}^{m} x_j$$

$$\hat{\sigma}_i = \sqrt{\frac{1}{m-1.5} \sum_{j=1}^{m} (x_j - \bar{x}_i)^2} \tag{1}$$

Note that this is a slightly biased estimator for the standard deviation, even assuming that the samples are drawn from the whole population independently and at random. In addition, the samples that each agent draws are spatially correlated, which may further deteriorate the precision of the this estimator.

Note further that choosing m may require some careful considerations: it should be large enough such that the variance estimate is accurate and not dominated by local effects (i.e., it should be chosen such that each agent can cover a significant portion of the location graph through its random walk); and it should be small enough such that it can appropriately capture the time-dependence of the true variance (which, in expectation, decreases with increasing time).

3.1.4 H3: Convergence rate

In this heuristic, each agent computes an estimated rate of convergence locally, and turns itself off if this rate falls below a predefined threshold δ.

Let's assume that the absolute value update $\Delta_i = |x_i(k) - x_j(k)|/2$ of robot i converges to 0 as

$$\Delta_i = \Delta_0 e^{-b_i k_i}$$

where k_i can either be the robots local tick counter, or the time since the beginning of the experiments (which is known to each individual agent).

We can then define

$$y_i := \log(\Delta_i) = \log(\Delta_0) - b_i k_i = a_i - b_i k_i$$

and estimate a_i, b_i from the last m samples using standard linear regression.

In this setting, the current rate of improvement can be obtained as

$$\frac{\partial}{\partial k_i} \Delta_i = -b_i \Delta_0 e^{-b_i k_i} = -b_i e^{a_i - b_i k_i}$$

A robot should turn itself off if it estimates its rate of improvement to be smaller than δ. Alternatively, this could be expanded to a more interesting heuristic, in which a robot turns itself off if the expected gain (based on the current rate of improvement estimate) is smaller than the expected cost of continuing operation. (Implementation detail: as $\log(0)$ is undefined, we define Δ_i as $\Delta_i = \max(|x_i(k) - x_j(k)|, \delta)/2$.)

3.1.5 H4: Initial versus current variance

This heuristic is closely related to H2. Instead of estimating the variance continuously and turning off once it falls below a pre-specified threshold, here each agent estimates the initial variance during a pre-defined number of steps at the beginning of operation. It then continues to estimate the current variance and switches

off once the current variance drops below a pre-specified fraction of the initial variance, e.g., below 0.1%. This metric may be less vulnerable to the magnitude of the initial variance than H2, and therefore more robust.

3.2 Positive feedback

Positive feedback is a common heuristic in collective decision making; it is frequently observed in nature [7, 8, 9] and has been shown to facilitate robust decision making in robotic swarms [10, 11]. Generally speaking, with positive feedback, the probability that a given agent in a collective commits to a given decision increases with the number of other agents that have already committed to said decision.

Each of the heuristics for localized convergence detection listed above can be run with or without positive feedback. Positive feedback can either be achieved by introducing a separate counter, or by modifying the metric of the heuristics listed above. If a separate counter c_f is introduced, positive feedback can be achieved as follows: whenever a communication partner is encountered that is already turned off, increase c_f; if c_f reaches a predefined threshold, the agent switches itself off.

Possible ways to adapt the heuristics H0-H4 to achieve positive feedback without a separate counter are discussed in the following section.

3.2.1 Adjusted metrics

In the case of heuristic H0 and H1, positive feedback can be achieved by increasing the step counter (or convergence counter c_i, respectively) every time an agent tries to communicate with another agent that is already turned off.

In heuristic H2, this can be achieved by computing the mean $\bar{x}_i$ and the summation in the Equation 1 only over the m_{on} agents that were still switched on. Similarly, in H3, Δ_i can be set to zero for any agent that is encountered in off state. Note that we are not claiming that these approaches are principled; we merely state that these modifications of H2 and H3 achieve positive feedback.

4 A simulator for the study of dynamic multi-agent consensus

4.1 Basic simulator

We aimed at building a versatile and modular simulator[1] to investigate consensus problems in dynamic multi-agent systems. As a baseline, we reproduced results from Oliva et al. (2019) to verify our implementation. Oliva et al. (2019) studied a gossip algorithm for multi-agent systems via random walk: Multiple agents on a random graph with variable initial values each average values with one single random agent on their node, should there be at least one, and then walk to a random neighboring node. Their actions are asynchronous and follow a Poisson process with rate λ.

To realize such a scenario, we implemented two underlying data structures: (i) location graphs ($G_{n,m}$, $G_{n,p}$, regular grid) representing agent locations (nodes) and their respective options for next moves (edges); (ii) a min-heap ordered according to the times of the next clock tick (next_clock) of each agent, thereby keeping track of agents' actions; the min-heap further stores the identification number (uuid) and graph

[1] Our simulator is implemented in Python 3 and lives on a private github repository including instructions for installation and use (please ask for access). A copy of the repository was submitted together with this report. We documented all experiments in a jupyter notebook, which we encourage others to use and play with simulation parameters. Alternatively, Python files can be run from a terminal and selected (but fewer) results will be shown in figures and print statements.

Algorithm 1 Our basic average consensus algorithm.

Instantiate a graph `G`, min-heap `H`, and lists for storing experimental and status values such as current agent values `val`.

Initialize agents with 0-mean Gaussian random values on random graph locations and insert them into the min-heap according to their respective times of next action, which follow a Poisson process.

Repeat until the end of simulation time (or decentralized consensus is detected):

1. `(uuid, event_time, pos) = H.delete_min()`

   Get agent with minimum clock and its uuid and position on graph from heap `H`

2. `neighbor_id = random.choice(list(G.agents[pos]))`

   Choose *one* random neighbor on the same position in the graph `G`

3. `avg_val = (val[uuid] + val[neighbor_id]) / 2`

   Average values, update values of all agents in `val`, update variance

4. `next_pos = G.graph[pos][random.randint(0, len(G.graph[pos])-1)]`
   `G.agents[pos].remove(uuid) G.agents[next_pos].add(uuid)`

   Move agent to a random neighboring node on graph `G` (or stay if node is isolated)

5. `H.insert(uuid, event_time + exp_rv(clock_rate), next_pos)`

   Update agent's clock and together with its new graph position insert it back into the heap `H`

Analyze and visualize simulation results.

location (pos) of each agent. Running on those two data structures according to user-specified parameters, such as number of agents or graph type, is our average consensus procedure as shown in Algorithm 1.

Oliva et al. (2019) tested $N = 10$ agents with clock rates $\lambda = 1$ on a $G_{n,m}$ random graph with $n = 40$ nodes and $m = 161$ edges for 1000 time steps. Our experiments with identical parameters match the results from Oliva et al. (2019), thereby validating our implementation (Figure 3). Our implementation can be found and tested in `2019_oliva_fig3bcd.ipynb`.

4.2 Principal investigations

In basic scaling experiments we investigated the performance of average consensus under isolated parameter variations for simulation run times of 100s. Compared to Oliva et al. (2019) we did not allow agents to pick themselves instead of available neighbors for averaging values since this slows down convergence and would be costly in practical applications. As a baseline, we used a $G_{n,p}$ random graph with $n = 1000$ nodes and $p = 0.5$ edge probability, and $N = 200$ agents with $\lambda = 1$ clocks initialized on random graph nodes with random zero-mean Gaussian distributed values, for which we found precise average consensus (Figure 4A). The standard deviation of the agents' values converged toward their mean under various experimental parameters (Figure 4B-F). For convergence speed, we confirmed that:

- The more agents on a given graph, the faster they reach consensus since the probability of interactions increases (Figure 4B) (within the parameter range we tested; this is not true beyond a certain number of agents).

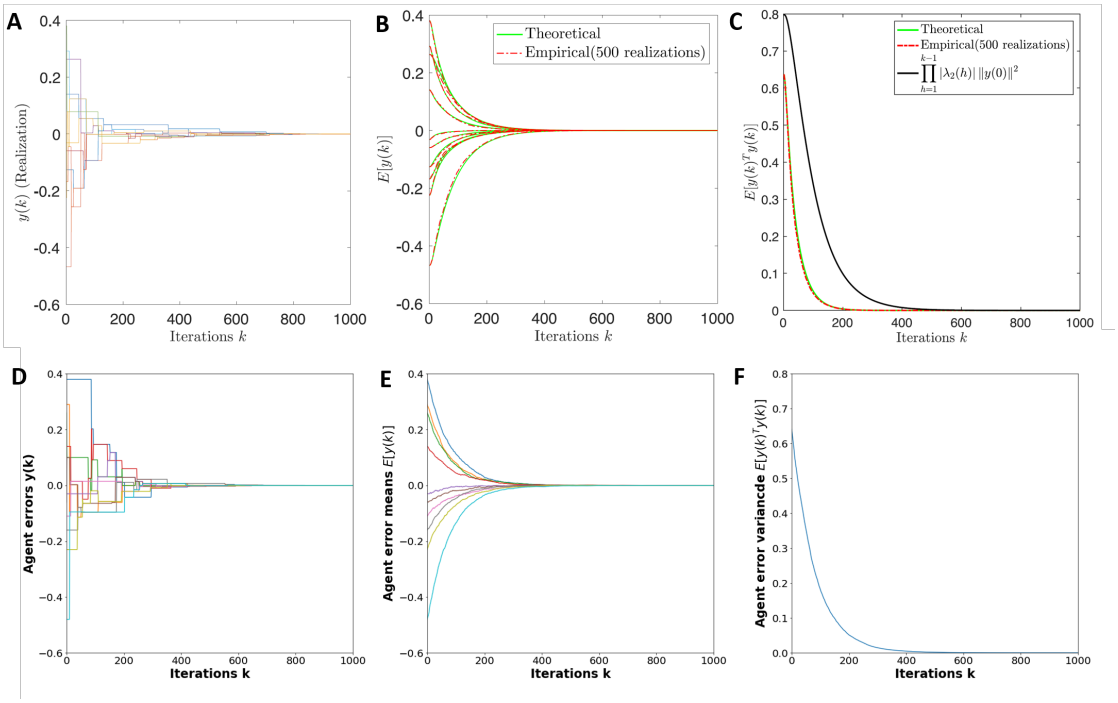

Figure 3: Agent error evolution **(A,D)**, empirical agent error means **(B,E)**, and empirical agent error variance **(C,F)** versus number of iterations in Oliva et al. (2019) **(A-C)** and our simulator **(D-F)**.

- The larger the graph for a given amount of agents, the slower they reach consensus since the probability of interactions decreases (Figure 4C).
- Graph density matters around the critical number of edges for which the graph might or might not be connected. Isolated parts of the graph cannot achieve average consenus with each other (Figure 4D).
- Average consensus takes longer if initial values are further apart (Figure 4E).
- The faster the clock rates of the agents, the more often they take action and communicate with a neighbor, and the faster they reach consensus (Figure 4F).

These observations align with our intuition, and indicate that the algorithm and simulator were implemented correctly. Our implementation can be found and tested in `basic_scalings.ipynb`.

4.3 Relaxation of communication protocol

Oliva et al. (2019) restricted communication to one-on-one. Many other communication protocols for gossiping exist including multiple that allow one-to-many communication. Kilobot, too, are capable of broadcasting and could send and receive values from one or multiple neighbors within one communication window. To investigate how such group communication affects convergence speed, we allowed agents to talk to several neighbors on the same node at a given time. Being able to average values with several neighbors in one time step accelerates consensus. However, at the point at which much more agents could talk than are likely to be found on a single node, group communication cannot be exploited anymore and

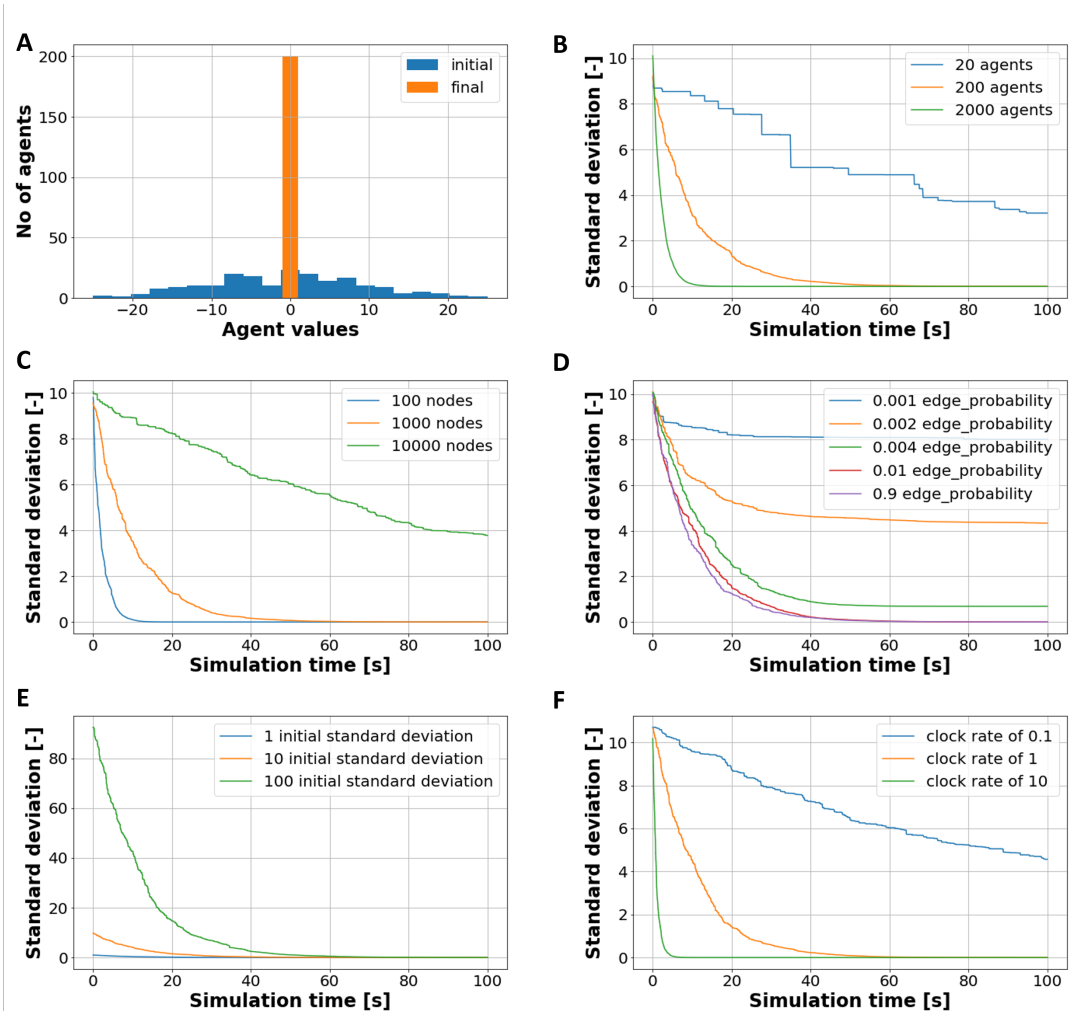

Figure 4: **(A)** Histogram of initial random Gaussian values ($\mu = 0, \sigma = 10$) and final averaged values of 200 agents running for 100s with $\lambda = 1$ clocks on a $G_{n,p}$ graph ($n = 1000, p = 0.5$). **(B-F)** Same simulation with five isolated parameter variations revealed the following scalings: more agents reach faster consensus (B); larger graphs slow down consensus (C); graph density does only affect consensus speed when close to or creating partitioned graphs (D); far apart initial values slow down consensus (E); faster agents reach faster consensus (F).

performance gains vanish (Figure 5A). Other than varying the number of interacting agents, we used the same baseline settings as in our principal investigations. Our implementation can be found and tested in `communication_limits.ipynb`.

4.4 Kilogrid - a Kilobot grid graph

Here we introduce an additional regular grid graph modelling the way Kilobots would move on a table (refer to Figure 2 for notes on how this graph can be constructed). We compare the performance over 100 realizations of average consensus on a random graph with this grid graph, both having equal amounts of nodes and edges. In expectation, agents moving at random on the grid graph achieve average consensus more slowly since their mixing is worse (Figure 5B). Our implementation can be found and tested in `random_vs_gridgraph.ipynb`.

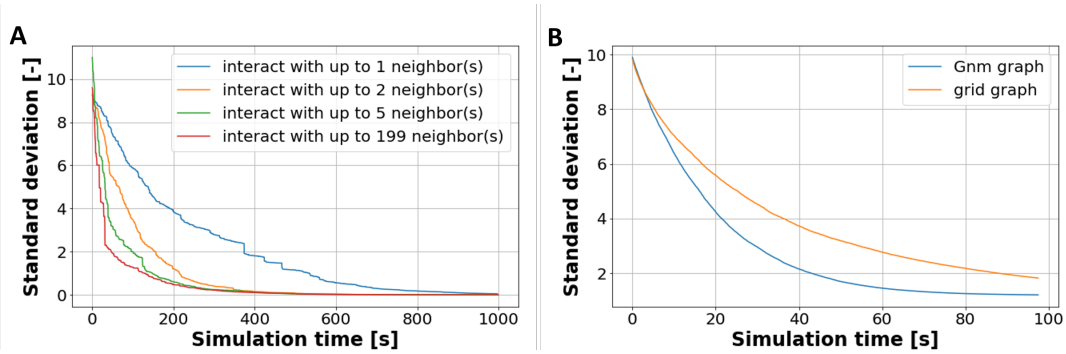

Figure 5: Two adaptions toward a closer representation of Kilobot agents moving on a 2D flat surface. **(A)** Averaging values with several neighbors in one time step accelerates consensus up to the point at which more agents could talk than are likely to be found on a single node. **(B)** In expectation (results are the mean of 100 simulations), agents moving at random on a grid graph achieve average consensus more slowly than on an equally large random graph since their mixing is worse.

4.5 Decentralized convergence detection (DCD)

In theory and for a given graph and agent set, the time required to achieve average consensus can be predicted with high probability. In practice, however, a multi-agent system might be released into an a priori unknown and complex environment to sample one or several of its qualities such as temperature, humidity, or radiation. The cost of agent deployment scales with time and distance travelled. Terminating any real world operation as soon as a good enough consensus has been achieved saves cost.

Instead of making potentially weak assumptions on unknown environments, we conceptualized different heuristics for agents to detect convergence individually in real time and without the need for central control (Section 3). Here we show the evaluation of these heuristics on our simulator. For all heuristics, agents successfully detected consensus and switched themselves off, thereby saving time and edges travelled (Table 1 and Figure 6). Different heuristics performed similarly well in terms of reaching a high quality consensus with low standard deviation as fast as possible (Figure 6A,B) and clearly improved against the controlling experiment with no heuristic applied. Comparing heuristics is left for future work since their performances depend on individual parameter tunings. A good heuristic is one that requires little global or a priori information and reaches consensus with smaller standard deviation quickly, reliably, and robustly. H3 and H4 are

such heuristics. In H4, for instance, each agent estimates the initial standard deviation among all agents by taking n samples at the beginning, and then switches off if the current sampled standard deviation is smaller than an improvement factor ε times the initial estimate; only n, ε have to be chosen and those parameters showed to be robust in simulation. Furthermore, they are largely independent of graph structure, number of agents, or the initial magnitude of the standard deviation.

We have not used positive feedback in any of the results presented here. However, positive feedback can be turned on and generally accelerated consensus in our implementation `dcd_heuristics.ipynb`.

Heuristic for consensus detection	Figure legend	Edges travelled
Ctrl: No decentralized consensus detection	none	29925
H0: Fixed number of interactions	fix_enc	6377
H1: Difference in values smaller threshold	abs_chg	10101
H2: Estimated STD smaller threshold	std_est	11271
H3: Convergence rate smaller threshold	conv_rate	14324
H4: Estimated STD $< \varepsilon \times$ initially sampled STD	unknown_env	12772

Table 1: Tested heuristics for consensus detection, their respective labels in Figure 6 and number of edges travelled before all agents switched off.

5 Conclusion

We designed and implemented a simulator to investigate multi-agent systems moving and interacting on random graphs. Here we studied the performance of dynamic average consensus via asynchronous gossip for a variety of experimental parameters. Furthermore, we added decentralized consensus detection heuristics for agents to terminate operation as soon as a satisfying consensus has been reached, thereby reducing each agent's energy spent on locomotion.

With the underlying data structures (location graphs and min-heap) and basic motion and communication algorithms in place, we can use our simulator to study a variety of more complex problems in future work, such as the speed of information dissemination among moving agents, or the best-of-n problem in which a swarm of robots makes a collective decision on the best out of n available options. We are aware that we need to carefully study the effect of selecting the decision parameters for each of the heuristics, and will do so in continuation of this project. We are further working on probabilistic performance guarantees that the heuristics in Section 3.1 can achieve.

At the same time, our simulator is well-suited to assist in the development of new theory by providing intuition and validation. We have already outlined a time-dependent convergence bound for moving agents compared to static agents in Boyd et al. (2006) in Appendix A, and, compared to Oliva et al. (2019), a simpler convergence proof as run time approaches infinity based on principles we have learnt in class (presented in Appendix B).

Finally, we have tailored our framework to simulate average consensus finding with a robotic platform that exhibits narrowly defined technical specifications, such as the Kilobot collective. We will use this simulator to explore optimal or near-optimal termination heuristics tailored to the Kilobots (including termination parameter selection), and will run experiments with these heuristics on hardware.

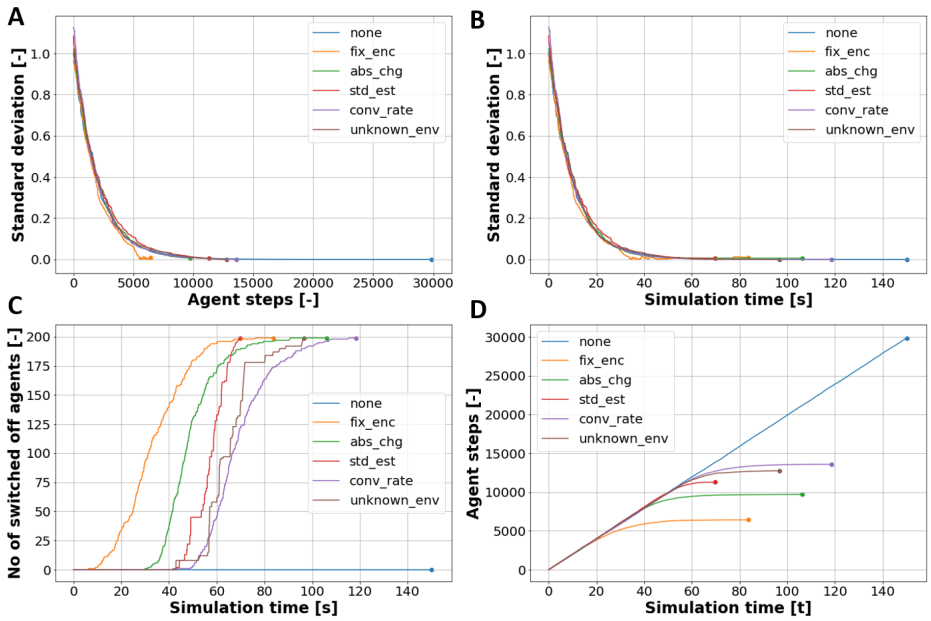

Figure 6: **(A)** All heuristics save on agent steps while achieving consensus with low standard deviation. **(B)** Average consensus procedure can be stopped prematurely since **(C)** agents switch off upon having reached consensus. **(D)** Agent steps stagnate and converge as the number of active agents decreases over time .

References

[1] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *IEEE/ACM Transactions on Networking (TON)*, vol. 14, no. SI, pp. 2508–2530, 2006.

[2] G. Oliva, S. Panzieri, R. Setola, and A. Gasparri, "Gossip algorithm for multi-agent systems via random walk," *Systems & Control Letters*, vol. 128, pp. 34–40, 2019.

[3] A. Pettarin, A. Pietracaprina, G. Pucci, and E. Upfal, "Tight bounds on information dissemination in sparse mobile networks," in *Proceedings of the 30th annual ACM SIGACT-SIGOPS symposium on Principles of distributed computing*, pp. 355–362, ACM, 2011.

[4] H. Lam, Z. Liu, M. Mitzenmacher, X. Sun, and Y. Wang, "Information dissemination via random walks in d-dimensional space," in *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete algorithms*, pp. 1612–1622, Society for Industrial and Applied Mathematics, 2012.

[5] M. Rubenstein, A. Cornejo, and R. Nagpal, "Programmable self-assembly in a thousand-robot swarm," *Science*, vol. 345, no. 6198, pp. 795–799, 2014.

[6] M. Gauci, R. Nagpal, and M. Rubenstein, "Programmable self-disassembly for shape formation in large-scale robot collectives," in *Distributed Autonomous Robotic Systems*, pp. 573–586, Springer, 2018.

[7] D. J. Sumpter and S. C. Pratt, "Quorum responses and consensus decision making," *Philosophical Transactions of the Royal Society B: Biological Sciences*, vol. 364, no. 1518, pp. 743–753, 2008.

[8] D. J. Sumpter, "The principles of collective animal behaviour," *Philosophical transactions of the royal society B: Biological Sciences*, vol. 361, no. 1465, pp. 5–22, 2005.

[9] S. C. Nicolis and J.-L. Deneubourg, "Emerging patterns and food recruitment in ants: an analytical study," *Journal of theoretical biology*, vol. 198, no. 4, pp. 575–592, 1999.

[10] A. Brutschy, A. Scheidler, E. Ferrante, M. Dorigo, and M. Birattari, ""can ants inspire robots?" self-organized decision making in robotic swarms," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4272–4273, IEEE, 2012.

[11] J. Ebert, M. Gauci, F. Mallmann-Trenn, and R. Nagpal, "Bayes bots: Collective bayesian decision-making in decentralized robot swarms," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, in review.

A Upper bound on $T_{ave}(\varepsilon)$ for average consensus finding with random agent motion

In this section, we demonstrate how an upper bound on the ε-averaging time $T_{ave}(\varepsilon)$ can be obtained for the setting described in this report and Oliva et al. (2019). The derivation of this bound follows the steps outlined in Boyd et al. (2006) and as such relies on (i) a proof that the $x(k) - x_{ave}$ and $\mathrm{Var}(x(k))$ converge to 0; (ii) the fact that $|\lambda_2(W(k))| < 1$ by construction for sufficiently large k; and (iii) an application of Markov's inequality. Here, $W = \mathbb{E}[W(k)]$, and $W(k)$ is the probabilistic state transition matrix that governs the evolution of this system as

$$x(k) = W(k)x(k-1)$$

The reader is referred to Oliva et al. (2019) for a proof that $x(k) - x_{ave}$ and $\mathrm{Var}(x(k))$ converge to 0; in addition, we state an outline of an alternative proof for this convergence in Appendix B. The steps (ii) and (iii) required to bound $T_{ave}(\varepsilon)$ are outlined in abbreviated form below.

Oliva et al. (2019) define the probabilistic transition matrix $W(k)$ using the matrix

$$W_{ij} = I - \frac{(e_i - e_j)(e_i - e_j)^T}{2}$$

where e_i is a $n \times 1$ unit vector with its ith component 1 and all other entries 0. In other words, a state transition with W_{ij} occurs if agents i and j meet and exchange information. Oliva et al. (2019) denote the probability that agents i and j communicate as $\omega_{i,j}$; therefore, $W(k) = W_{ij}$ with probability $\omega_{i,j}$. The exact decomposition of the communication probability $\omega_{i,j}$ is discussed in detail in the original paper and omitted here. In brief, $\omega_{i,j}$ is the joint probability of the following events: (i) the clock of agent i ticks, and (ii) agent i selects to communicate with agent j (which can only occur if i and j share a node when the clock of i ticks). W can then be obtained as

$$W = \mathbb{E}[W(k)] = \sum_i \sum_j \omega_{ij}(k) W_{ij}$$

As a crucial step in their work, Oliva et al. (2019) demonstrate that $|\lambda_2(W(k))| < 1$ for $k \geq D$, where D is the diameter of the location graph G on which the agents move. We will rely on this preposition to show that the ε-averaging time is bounded.

As part (iii) of our derivation of a bound on $T_{ave}(\varepsilon)$, consider the following probability

$$\Pr\left(\frac{||x(k) - x_{ave}\mathbf{1}||}{||x(0)||} \geq \varepsilon\right) = \Pr\left(\frac{||x(k) - x_{ave}\mathbf{1}||^2}{||x(0)||^2} \geq \varepsilon^2\right)$$

By definition of the variance and denoting the error at time k as $y(k)$, this is the same as

$$\Pr\left(\frac{y(k)^T y(k)}{x(0)^T x(0)} \geq \varepsilon^2\right)$$

Applying Markov's inequality to this probability yields

$$\Pr\left(\frac{y(k)^T y(k)}{x(0)^T x(0)} \geq \varepsilon^2\right) \leq \varepsilon^{-2} \frac{\mathbb{E}[y(k)^T y(k)]}{x(0)^T x(0)} \tag{2}$$

Following Equation (11) in Oliva et al. (2019), we know that

$$\mathbb{E}[y(k)^T y(k)] \leq \prod_{h=0}^{k-1} |\lambda_2(h)| ||y(0)||^2 \leq |\lambda_{2,max}|^k ||y(0)||^2$$

where $\lambda_{2,max}$ is the maximum second largest eigenvalue of $\mathbb{E}[W(k)]$: $\lambda_{2,max} = \max_k |\lambda_2(k)|$. If $k > D$, we can further state that

$$\mathbb{E}[y(k)^T y(k)] \leq \lambda_{2,maxD}^{k-D} ||y(0)||^2$$

where $\lambda_{2,maxD} = \max_{k>D} |\lambda_2(k)|$, which is positive and strictly smaller than 1 (see also Corollary 1 in Oliva et al. (2019)). We further know that $||y(0)||^2 = y(0)^T y(0) = x(0)^T x(0) - n x_{ave}^2 \leq x(0)^T x(0)$, which allows us to rewrite Equation 2 as

$$\Pr\left(\frac{y(k)^T y(k)}{x(0)^T x(0)} \geq \varepsilon^2\right) \leq \varepsilon^{-2} \frac{\mathbb{E}[y(k)^T y(k)]}{x(0)^T x(0)} \leq \varepsilon^{-2} |\lambda_{2,maxD}|^{k-D} \ \ \forall \ \ k > D \tag{3}$$

The upper bound on the ε-averaging time then follows from Equation 3:

$$T_{ave}(\varepsilon) \leq D + \frac{3 \log \varepsilon^{-1}}{\log \lambda_{2,maxD}^{-1}}$$

As a final step, we would like to derive a bound on $\lambda_{2,maxD}$ given that the location graph, the density of agents, and potentially additional information on the initial agent distribution, are known. We have not managed to do so yet.

B Convergence proof

In the following, we provide a proof that the gossiping scheme presented in Oliva et al. (2019) converges to the true average of the initial values. Our proof is much shorter, mathematically less involved than the proof employed in [2], and based on probabilistic analysis.

Average consensus setting. We are dealing with a $G_{n,p}$ random graph on which n agents move around, each following a random walk. The agents each were given a random value at initialization. Whenever an agent's clock ticks (modeled by a Poisson process), that agent (i) gossips with a random neighbor on the same node, should there be at least one neighbor, and then (ii) moves to a random neighboring node on the location graph. If two agents gossip, they each set their value to the average of the pair's previous values.

Convergence-to-mean-claim. The agents' values converge to their mean for large enough run times.

Proof. Our proof makes four key observations: (i) each agent will eventually gossip with each other agent; (ii) a gossip step will never increase the mean error; (iii) if two agents gossip, there is a non-zero probability that the mean error is reduced, i.e., if the values held by the two agents are on different sides of the initial mean $x_{ave}(t=0)$; (iv) as long as the consensus has not converged, there exist at least two agents that hold values which are on different sides of the initial mean $x_{ave}(t=0)$. Given (i-iv), each agent's error is monotonically decreasing towards 0.

For (i) we rely on the fact that two agents following independent random walks on a random graph will eventually meet.

For (ii) we look at a case in which two values from equal sides of the mean get paired and averaged: by taking an unweighted average of these two values, the sum of the averaged values will be equally far away from the mean than the some of the initially paired values, i.e., by contradiction the values will not move further away from the mean.

For (iii) we use a simple expectation argument: for the mean to exist, there needs to be at least one value at least as big as the mean and one value at most as big as the mean. Therefore, a non-zero probability exists that such two values get paired, averaged, and the new resulting value is closer to the mean.

For (iv) our justification relies on two arguments, of which the first one closely resembles an expectation argument: First, if the average of all agent values at time t is $x_{ave}(t)$, there must be at least two agents i and j such that the values these agents hold at time t are on different sides of $x_{ave}(t)$. Second, as shown below, a gossip step never changes the population mean (i.e., $x_{ave}(t) = x_{ave}(t=0)\ \forall\, t$) under the given update rule:

$$x_{ave}(t) = \frac{1}{n}\sum_{h=1}^{n} x_h(t)$$

$$x_{ave}(t+1) = \frac{1}{n}\left(\sum_{h \setminus \{i,j\}} x_h(t) + x_i(t) + x_j(t)\right) = \frac{1}{n}\left(\sum_{h \setminus \{i,j\}} x_h(t) + 2\frac{x_i(t)+x_j(t)}{2}\right) = x_{ave}(t)$$