Ppsim: A Software Package for Efficiently Simulating and Visualizing Population Protocols

David Doty(✉) and Eric Severson

University of California, Davis, CA 95616, USA
{doty,eseverson}@ucdavis.edu

Abstract. We introduce `ppsim` [28], a software package for efficiently simulating population protocols, a widely-studied subclass of chemical reaction networks (CRNs) in which all reactions have two reactants and two products. Each step in the dynamics involves picking a uniform random pair from a population of n molecules to collide and have a (potentially null) reaction. In a recent breakthrough, Berenbrink, Hammer, Kaaser, Meyer, Penschuck, and Tran [6] discovered a population protocol simulation algorithm quadratically faster than the naïve algorithm, simulating $\Theta(\sqrt{n})$ reactions in *constant* time (independently of n, though the time scales with the number of species), while preserving the *exact* stochastic dynamics.

`ppsim` implements this algorithm, with a tightly optimized Cython implementation that can exactly simulate hundreds of billions of reactions in seconds. It dynamically switches to the CRN Gillespie algorithm for efficiency gains when the number of applicable reactions in a configuration becomes small. As a Python library, `ppsim` also includes many useful tools for data visualization in Jupyter notebooks, allowing robust visualization of time dynamics such as histogram plots at time snapshots and averaging repeated trials.

Finally, we give a framework that takes any CRN with only bimolecular (2 reactant, 2 product) or unimolecular (1 reactant, 1 product) reactions, with arbitrary rate constants, and compiles it into a continuous-time population protocol. This lets `ppsim` exactly sample from the chemical master equation (unlike approximate heuristics such as τ-leaping or LNA), while achieving asymptotic gains in running time. In linked Jupyter notebooks, we demonstrate the efficacy of the tool on some protocols of interest in molecular programming, including the approximate majority CRN and CRN models of DNA strand displacement reactions.

Keywords: Population protocol · Chemical reaction network

1 Introduction

A foundational model of chemistry used in natural sciences is that of chemical reaction networks (CRNs) [22]: finite sets of reactions such as $A + B \to C + D$,

Supported by NSF award 1900931 and CAREER award 1844976.

E. Cinquemani and L. Paulevé (Eds.): CMSB 2021, LNBI 12881, pp. 245–253, 2021.
https://doi.org/10.1007/978-3-030-85633-5_16

representing that molecules A and B, upon colliding, can change into C and D. This gives a continuous time, discrete state, Markov process [22] modelling discrete counts[1] of molecules.

Population protocols [3], a well-studied model of distributed computing with limited agents, are a restricted subset of CRNs (with two reactants and two products in each reaction, and unit rate constants) that nevertheless capture many of the interesting features of CRNs. Different terminology is used: in reaction $A + B \to C + D$, two *agents* (molecules), whose *states* (species types) are A, B, have an *interaction* (reaction), changing their states respectively to C, D. *Gillespie kinetics for CRNs.* The standard Gillespie algorithm [22] simulates the Markov process mentioned above. Given a fixed volume $v \in \mathbb{R}^+$, the *propensity* of a unimolecular reaction $r : X \xrightarrow{k} \ldots$ is $\rho(r) = k \cdot \#X$, where $\#X$ is the count of X. The propensity of a bimolecular reaction $r : X + Y \xrightarrow{k} \ldots$ is $\rho(r) = k \cdot \frac{\#X \cdot \#Y}{v}$ if $X \neq Y$ and $k \cdot \frac{\#X \cdot (\#X - 1)}{2v}$ otherwise. The Gillespie algorithm calculates the sum of the propensities of all reactions: $\rho = \sum_r \rho(r)$. The time until the next reaction is sampled as an exponential random variable T with rate ρ, and a reaction r_{next} is chosen with probability $\rho(r_{\text{next}})/\rho$ to be applied.

Population Protocols. The population protocols model comes with simpler dynamics. At each step, a scheduler chooses a random pair of agents (molecules) to interact in a (potentially null) reaction. The discrete time model counts each interaction as $\frac{1}{n}$ units of time, where n is the population size. A continuous time variant [18] gives each agent a rate-1 Poisson clock, upon which it interacts with a randomly chosen other agent. The expected time until the next interaction is $\frac{1}{n}$, so up to a re-scaling of time, which by straightforward Chernoff bounds is negligible, these two models are equivalent. `ppsim` can use either time model.

There is an important efficiency difference between the algorithms: the Gillespie algorithm automatically skips null reactions. For example, a reaction such as $L + L \to L + F$, when $\#L = 2$ and $\#F = n - 2$, is much more efficient in the Gillespie algorithm, which simply increments the time until the $L + L \to L + F$ reaction by an exponential random variable in one step. A naïve population protocol simulation iterates through $\Theta(n)$ expected null interactions ($L + F \to L + F$ and $F + F \to F + F$) until the two L's react. To better handle cases like this, `ppsim` dynamically switches to the Gillespie algorithm when the number of null interactions is sufficiently large; see documentation [28] for implementation details.

Other Simulation Algorithms. Variants of the Gillespie algorithm reduce the time to apply a single reaction from $O(|R|)$ to $O(\log |R|)$ [21] or $O(1)$ [31], where $|R|$ is the number of types of reactions. However, the time to apply n reactions still scales with n. A common speedup heuristic for simulating $\omega(1)$ reactions in $O(1)$ time is τ-*leaping* [10,23,24,29,32], which "leaps" ahead by time τ, by assuming reaction propensities will not change and updating counts in a single batch step

[1] Another modelling choice are ODEs that describe real-valued concentrations, the "mean-field" approximation to the discrete behavior in the large scale limit [26].

by sampling according these propensities. Such methods necessarily approximate the kinetics inexactly, though it is possible in some cases to prove bounds on the approximation accuracy [32]. Linear noise approximation (LNA) [11] can be used to approximate the discrete kinetics, by adding stochastic noise to an ODE approximation. A speedup heuristic for population protocol simulation is to sample the number of each interaction that would result from a random matching of size m, and update species counts in a single step. This, too, is an inexact approximation: unlike the true process, it prevents any molecule from participating in more than one of the next m interactions.

The algorithm implemented by `ppsim`, due to Berenbrink, Hammer, Kaaser, Meyer, Penschuck, and Tran [6], builds on this last heuristic. Conditioned on the event that no molecule is picked twice during the next m interactions, these interacting pairs are a random disjoint matching of the molecules. Define the random variable C as the number of interactions until the same molecule is picked twice. Their basic algorithm samples this collision length C according to its exact distribution, then updates counts in batch assuming all pairs of interacting molecules are disjoint until this collision, and finally simulates the interaction involving the collision. By the Birthday Paradox, $\mathrm{E}[C] \approx \sqrt{n}$ in a population of n molecules, giving a quadratic factor speedup over the naïve algorithm. The time to update a batch scales quadratically with q, the total number of states. The "multibatch" variant, used by `ppsim`, samples multiple successive collisions to process an even larger batch, and uses $O\left(q\sqrt{\frac{\log n}{n}}\right)$ time per simulated interaction.

See [6] for details. An advantage of such a fast simulator, specifically for population protocols implementing *algorithms*, is that the very large population sizes it can handle (over 10^{12}) allow one to tell the difference (on a log-scale plot of convergence time) between a protocol converging in time $O(\log n)$ versus, say, $O(\log^2 n)$.

2 Usage of the Ppsim Tool

We direct the reader to [28] for detailed installation, usage instructions, and examples. Here we highlight basic usage examples for specifying protocols.

There are three ways one can specify a population protocol, each best suited for different contexts. The most direct specification of a protocol directly encodes the mapping of input state pairs to output state pairs using a Python `dict` (the following is the well-studied *approximate majority* protocol, which has been studied theoretically [4,13] and implemented experimentally with DNA [12]):

```
a,b,u = 'A','B','U'
approx_majority = {(a,b):(u,u), (a,u):(a,a), (b,u):(b,b)}
```

More complex protocols with many possible species are often specified in pseudocode instead of listing all possible reactions. `ppsim` supports this by allowing the *transition function* mapping input states to output states to be computed

by a Python function. The following allows species to be integers and computes an integer average of the two reactants:

```
def discrete_averaging(s: int, r: int):
    return math.floor((s+r)/2), math.ceil((s+r)/2)
```

States and transition rules are converted to integer arrays for internal Cython methods, so there is no efficiency loss for the ease of representing protocol rules, since a Python function defining the transition function is not called during the simulation: producible states are enumerated before starting the simulation.

For complicated protocols, an advantage of `ppsim` over standard CRN simulators is the ability to represent species/states as Python objects with different fields (as they are often represented in pseudocode), and to plot counts of agents based on their field values.[2]

Finally, protocols can be specified using CRN-like notation for CRNs with reactions that are bimolecular (2-input, 2-output) or unimolecular (1-input, 1-output), with arbitrary rate constants. For instance, this code specifies the CRN

$$A + B \underset{4}{\overset{0.5}{\rightleftharpoons}} 2C, \qquad C \xrightarrow{5} D$$

```
a,b,c,d = species('A B C D')
crn = [(a+b | 2*c).k(0.5).r(4), (c >> d).k(5)]
```

This will then get compiled into a continuous time population protocol that samples the same distribution as Gillespie. See full paper [14] for details.

Any of the three specifications (`dict`, Python function, or list of CRN reactions) can be passed to the `Simulation` constructor. The `Simulation` can be run to generate a history of sampled configurations.

```
init_config = {a: 51, b: 49}
sim = Simulation(init_config, approx_majority)
sim.run(16, 0.1)    # 160 samples up to time 16
sim.history.plot() # Pandas dataframe with counts
```

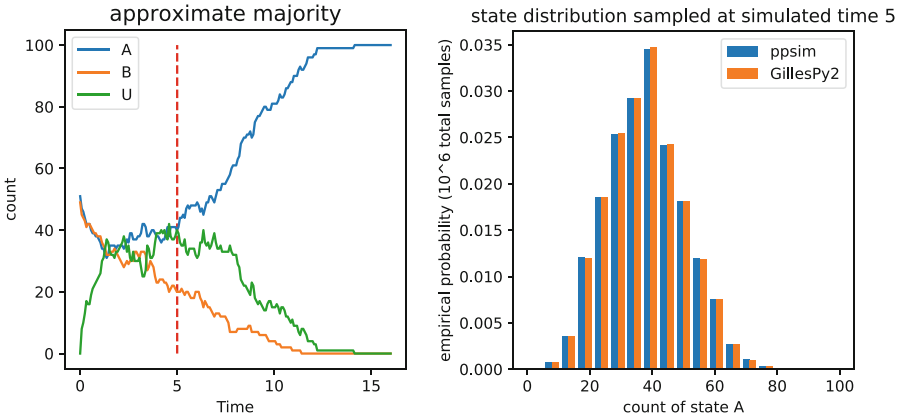
This would produce the plot shown in Fig. 1a. When the input is a CRN, `ppsim` defaults to continuous time and produces the exact same distributions as the Gillespie algorithm. Figure 1b shows a test against the package GillesPy2 [25] to confirm they sample the same distribution.

3 Speed Comparison with Other CRN Simulators

We ran speed comparisons of `ppsim` against both GillesPy2 [25] and StochKit2 [30], the latter being the fastest option we found for Gillespie simulation. Figure 2 shows that `ppsim` is able to reach significantly larger population sizes. Other tests shown in an example notebook[3] show how each package scales with the number of species and reactions.

[2] Download and run https://github.com/UC-Davis-molecular-computing/ppsim/blob/main/examples/majority.ipynb to visualize such large state protocols.

[3] https://github.com/UC-Davis-molecular-computing/ppsim/blob/main/examples/crn.ipynb shows further plots and explanations.



(a) Plot of sim.history. (b) Comparison with Gillespie algorithm.

Fig. 1. Time 5 (dotted line in Fig. 1a) was sampled 10^6 times with `ppsim` and GillesPy2 to verify they both sample the same chemical master equation distribution (Fig. 1b).

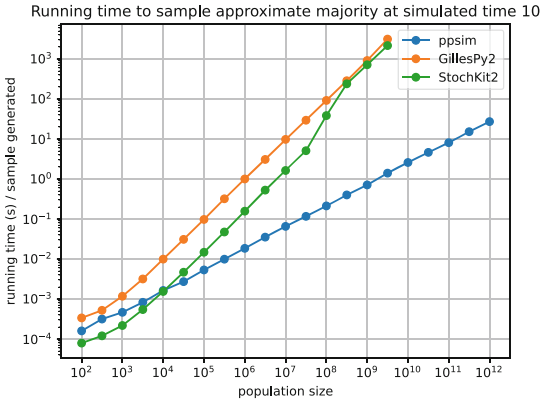
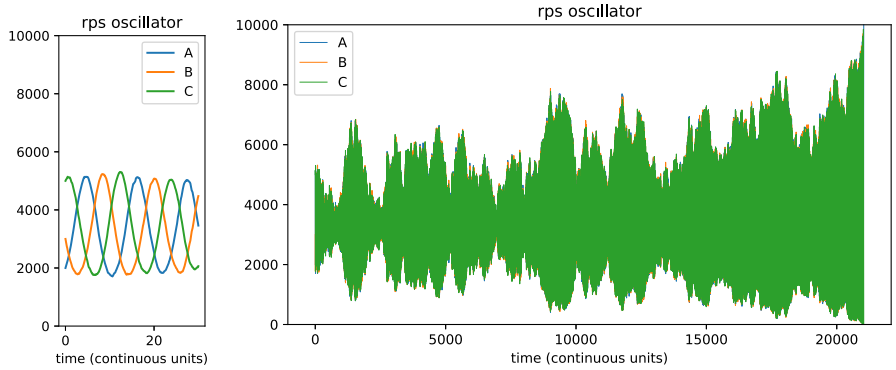


Fig. 2. Comparing runtime with population size n shows $O(n)$ scaling for Gillespie (slope 1 on log-log plot) versus $O(\sqrt{n})$ scaling for `ppsim` (slope 1/2).

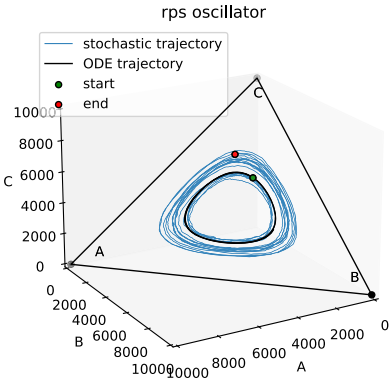
4 Issues with Other Speedup Methods

It is reasonable to conjecture that exact stochastic simulation of large-count systems is unnecessary, since Gillespie is fast enough on small-count systems, and faster ODE approximation is "reasonably accurate" for large-count systems. However, there are example large count systems with stochastic effects not observed in ODE simulation, and where τ-leaping introduces systematic inaccuracies that disrupt the fundamental qualitative behavior of the system, demonstrating the need for exact stochastic simulation. A simple such example is the

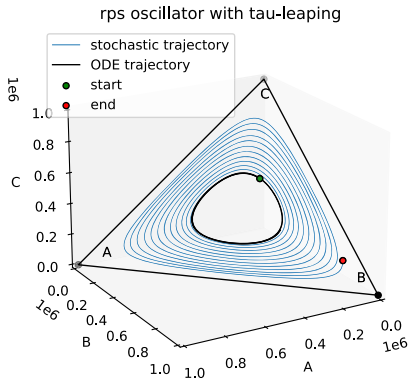


(a) Short timescale oscillations.

(b) Over a long $\Theta(n)$ timescale, the varying amplitudes will cause two species to go extinct.



(c) Dynamics from Figs 3a, 3b in phase space. The ODE solution has a neutrally stable orbit.



(d) τ-leaping adds a consistent outward drift that will lead to extinction on a much shorter timescale.

Fig. 3. The rock-paper-scissors oscillator has qualitative dynamics missed by both ODE simulation (never goes extinct) and τ-leaping (too quickly goes extinct).

3-state rock-paper-scissors oscillator: $B + A \to 2B,\ C + B \to 2C,\ A + C \to 2A$. Fig. 3 compares exact simulation of this CRN to τ-leaping and ODEs.

The population protocol literature furnishes more examples, with problems such as leader election [2,5,7–9,15,17,19,20,34,35] and single-molecule detection [1,16],[4] that crucially use small counts in a very large population, a regime not modelled correctly by ODEs. See also [27] for examples of CRNs with

[4] Download and run https://github.com/UC-Davis-molecular-computing/ppsim/blob/main/examples/rps_oscillator.ipynb to see visualizations of the generalized 7-state rps oscillator used for single-molecule detection in [16].

qualitative stochastic behavior not captured by ODEs, yet that behavior appears only in population sizes too large to simulate with Gillespie.

5 Conclusion

Unfortunately, the algorithm of Berenbrink et al. [6] implemented by `ppsim` seems inherently suited to population protocols, not more general CRNs. For instance, reversible dimerization reactions $A + B \rightleftharpoons C$ (used, for example, in [33] to model toehold occlusion reactions in DNA systems) seem beyond the reach of the batching technique of [6]. Although such reactions can be *approximated* by $A + B \rightleftharpoons C + F$ for some anonymous "fuel" species F, the count of F influences the rate of the reverse reaction $F + C \to A + B$, with a different rate than $C \to A + B$.

Another area for improvement is the handling of null reactions. There could be a way to more deeply intertwine the logic of the Gillespie and batching algorithms, to gain the simultaneous benefits of each, skipping the null reactions while simulating many non-null reactions in batch.

References

1. Alistarh, D., Dudek, B., Kosowski, A., Soloveichik, D., Uznański, P.: Robust detection in leak-prone population protocols. In: Brijder, R., Qian, L. (eds.) DNA 2017. LNCS, vol. 10467, pp. 155–171. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66799-7_11
2. Alistarh, D., Gelashvili, R.: Polylogarithmic-time leader election in population protocols. In: Halldórsson, M.M., Iwama, K., Kobayashi, N., Speckmann, B. (eds.) ICALP 2015. LNCS, vol. 9135, pp. 479–491. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-47666-6_38
3. Angluin, D., Aspnes, J., Diamadi, Z., Fischer, M.J., Peralta, R.: Computation in networks of passively mobile finite-state sensors. Distrib. Comput. **18**(4), 235–253 (2006)
4. Angluin, D., Aspnes, J., Eisenstat, D.: A simple population protocol for fast robust approximate majority. Distrib. Comput. **21**(2), 87–102 (2008)
5. Berenbrink, P., Giakkoupis, G., Kling, P.: Optimal time and space leader election in population protocols. In: STOC 2020: Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, pp. 119–129. Association for Computing Machinery, New York (2020). https://doi.org/10.1145/3357713.3384312
6. Berenbrink, P., Hammer, D., Kaaser, D., Meyer, U., Penschuck, M., Tran, H.: Simulating population protocols in sub-constant time per interaction. In: ESA 2020: 28th Annual European Symposium on Algorithms, vol. 173, pp. 16:1–16:22 (2020). https://drops.dagstuhl.de/opus/volltexte/2020/12882
7. Berenbrink, P., Kaaser, D., Kling, P., Otterbach, L.: Simple and efficient leader election. In: 1st Symposium on Simplicity in Algorithms (SOSA 2018), vol. 61, pp. 9:1–9:11 (2018)

8. Bilke, A., Cooper, C., Elsässer, R., Radzik, T.: Brief announcement: population protocols for leader election and exact majority with $O(\log^2 n)$ states and $O(\log^2 n)$ convergence time. In: PODC 2017: Proceedings of the ACM Symposium on Principles of Distributed Computing, pp. 451–453. ACM (2017)
9. Burman, J., et al.: Time-optimal self-stabilizing leader election in population protocols. In: PODC 2021: Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing (2021)
10. Cao, Y., Gillespie, D.T., Petzold, L.R.: Efficient step size selection for the tau-leaping simulation method. J. Chem. Physi. **124**(4), 044109 (2006)
11. Cardelli, L., Kwiatkowska, M., Laurenti, L.: Stochastic analysis of chemical reaction networks using linear noise approximation. Biosystems **149**, 26–33 (2016). https://doi.org/10.1016/j.biosystems.2016.09.004, https://www.sciencedirect.com/science/article/pii/S0303264716302039, selected papers from the Computational Methods in Systems Biology 2015 conference
12. Chen, Y.J., et al.: Programmable chemical controllers made from DNA. Nat. Nanotechnol. **8**(10), 755–762 (2013)
13. Condon, A., Hajiaghayi, M., Kirkpatrick, D., Maňuch, J.: Approximate majority analyses using tri-molecular chemical reaction networks. Nat. Comput. **19**(1), 249–270 (2020)
14. Doty, D., Severson, E.: ppsim: A software package for efficiently simulating and visualizing population protocols. Technical Report 2105.04702, arXiv (2021). arXiv:2105.04702
15. Doty, D., Soloveichik, D.: Stable leader election in population protocols requires linear time. Distrib. Comput. **31**(4), 257–271 (2018), special issue of invited papers from DISC 2015
16. Dudek, B., Kosowski, A.: Universal protocols for information dissemination using emergent signals. In: Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, pp. 87–99 (2018)
17. Elsässer, R., Radzik, T.: Recent results in population protocols for exact majority and leader election. Bull. EATCS **3**(126) (2018)
18. Fanti, G., Holden, N., Peres, Y., Ranade, G.: Communication cost of consensus for nodes with limited memory. Proc. Natl. Acad. Sci. **117**(11), 5624–5630 (2020)
19. Gąsieniec, L., Stachowiak, G.: Fast space optimal leader election in population protocols. In: SODA 2018: ACM-SIAM Symposium on Discrete Algorithms, pp. 2653–2667. SIAM (2018)
20. Gąsieniec, L., Stachowiak, G., Uznański, P.: Almost logarithmic-time space optimal leader election in population protocols. In: SPAA 2019: 31st ACM Symposium on Parallelism in Algorithms and Architectures, pp. 93–102 (2019)
21. Gibson, M.A., Bruck, J.: Efficient exact stochastic simulation of chemical systems with many species and many channels. J. Phys. Chem. A **104**(9), 1876–1889 (2000)
22. Gillespie, D.T.: Exact stochastic simulation of coupled chemical reactions. J. Phys. Chem. **81**(25), 2340–2361 (1977)
23. Gillespie, D.T.: Approximate accelerated stochastic simulation of chemically reacting systems. J. Phys. Chem. **115**(4), 1716–1733 (2001)
24. Gillespie, D.T.: Stochastic simulation of chemical kinetics. Annu. Rev. Phys. Chem. **58**, 35–55 (2007)
25. GillesPy2. https://github.com/StochSS/GillesPy2
26. Kurtz, T.G.: The relationship between stochastic and deterministic models for chemical reactions. J. Phys. Chem. **57**(7), 2976–2978 (1972)

27. Lathrop, J.I., Lutz, J.H., Lutz, R.R., Potter, H.D., Riley, M.R.: Population-induced phase transitions and the verification of chemical reaction networks. In: Geary, C., Patitz, M.J. (eds.) DNA 26: 26th International Conference on DNA Computing and Molecular Programming. Leibniz International Proceedings in Informatics (LIPIcs), vol. 174, pp. 5:1–5:17. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Dagstuhl (2020). https://doi.org/10.4230/LIPIcs.DNA.2020.5
28. ppsim Python package. source code (2021). https://github.com/UC-Davis-molecular-computing/ppsim API documentation, https://ppsim.readthedocs.io/ Python package for installation via pip: https://pypi.org/project/ppsim/
29. Rathinam, M., El Samad, H.: Reversible-equivalent-monomolecular tau: a leaping method for "small number and stiff" stochastic chemical systems. J. Comput. Phys. **224**(2), 897–923 (2007)
30. Sanft, K.R., Wu, S., Roh, M., Fu, J., Rone, K.L., Petzold, L.R.: Stochkit2: software for discrete stochastic simulation of biochemical systems with events. Bioinformatics **27**(17), 501–522 (2011). https://academic.oup.com/bioinformatics/article/27/17/2457/224105
31. Slepoy, A., Thompson, A.P., Plimpton, S.J.: A constant-time kinetic monte carlo algorithm for simulation of large biochemical reaction networks. J. Chem. Phys. **128**(20), 05B618 (2008)
32. Soloveichik, D.: Robust stochastic chemical reaction networks and bounded tau-leaping. J. Comput. Biol. **16**(3), 501–522 (2009)
33. Srinivas, N., Parkin, J., Seelig, G., Winfree, E., Soloveichik, D.: Enzyme-free nucleic acid dynamical systems. Science **358**(6369), eaal2052 (2017)
34. Sudo, Y., Masuzawa, T.: Leader election requires logarithmic time in population protocols. Para. Process. Lett. **30**(01), 2050005 (2020)
35. Sudo, Y., Ooshita, F., Izumi, T., Kakugawa, H., Masuzawa, T.: Time-optimal leader election in population protocols. IEEE Trans. Parallel Distrib. Syst. **31**(11), 2620–2632 (2020). https://doi.org/10.1109/TPDS.2020.2991771