Bio-Inspired Energy Distribution for Programmable Matter

Joshua J. Daymude, **Andréa W. Richa**, and **Jamison W. Weber**

(Arizona State University)

ICDCN 2021, Online Event — January 6, 2021

Arizona State University

Programmable Matter

Programmable matter is a substance that can change its physical properties autonomously based on user input or environmental stimuli.

"Catoms"

"Kilobots"

"M-Blocks"

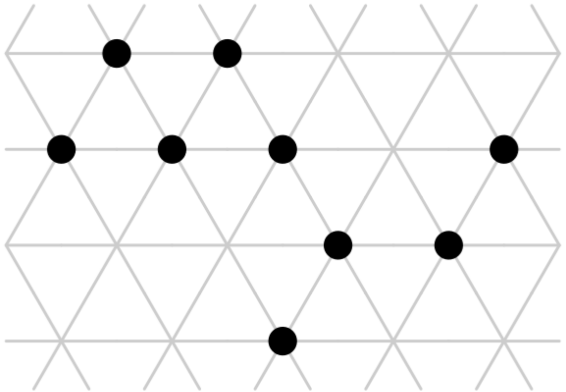
"Particle Robots"

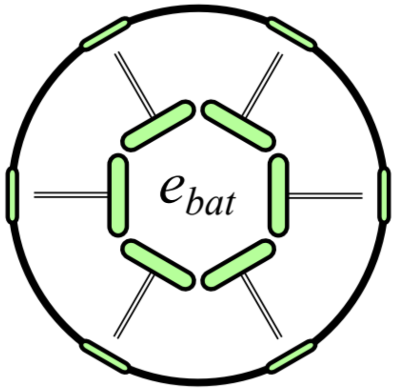
Access vs. need for power is inhomogeneous: some modules may have access to an external energy source, but all modules need power to function.

Energy Distribution: Model & Problem

How can energy be efficiently harvested and distributed via module-to-module power transfer so that all modules can perform their functions?

We investigate this problem under the **amoebot model**.





We assume each particle P has a battery $P.e_{bat}$ with capacity $\kappa > 0$.

Particles with access to the external energy source can directly harvest energy.

All other particles depend on their neighbors to share energy with them.

Energy Distribution: Model & Problem

How can energy be efficiently harvested and distributed via module-to-module power transfer so that all modules can perform their functions?

Formally, the **energy distribution problem** has instances $(\mathcal{P}, \kappa, \delta)$ where:

- $\mathcal{P}$ is a finite, connected particle system containing at least one particle with access to an external energy source.
- κ is the capacity of each particle's battery
- $\delta(P, i) \leq \kappa$ denotes the energy cost for a particle P to perform its i-th action.

A particle is stressed if $P.e_{bat} < \delta(P)$, meaning it has insufficient energy to meet its demand.

An algorithm $\mathcal{A}$ solves the energy distribution problem in time t if:

- No particle remains stressed for more than t rounds
- At least one particle performs an action every t rounds.

Energy Distribution: Approaches

An algorithm $\mathcal{A}$ solves the energy distribution problem in time t if:

- No particle remains stressed for more than t rounds.
- At least one particle performs an action every t rounds.

Fully Selfish: Whenever $P.e_{bat} \geq \delta(P)$, spend the energy to perform an action.

- Problem! Some particles may remain stressed indefinitely.

Fully Altruistic: Whenever a neighbor Q has $Q.e_{bat} < \kappa$, transfer energy to Q.

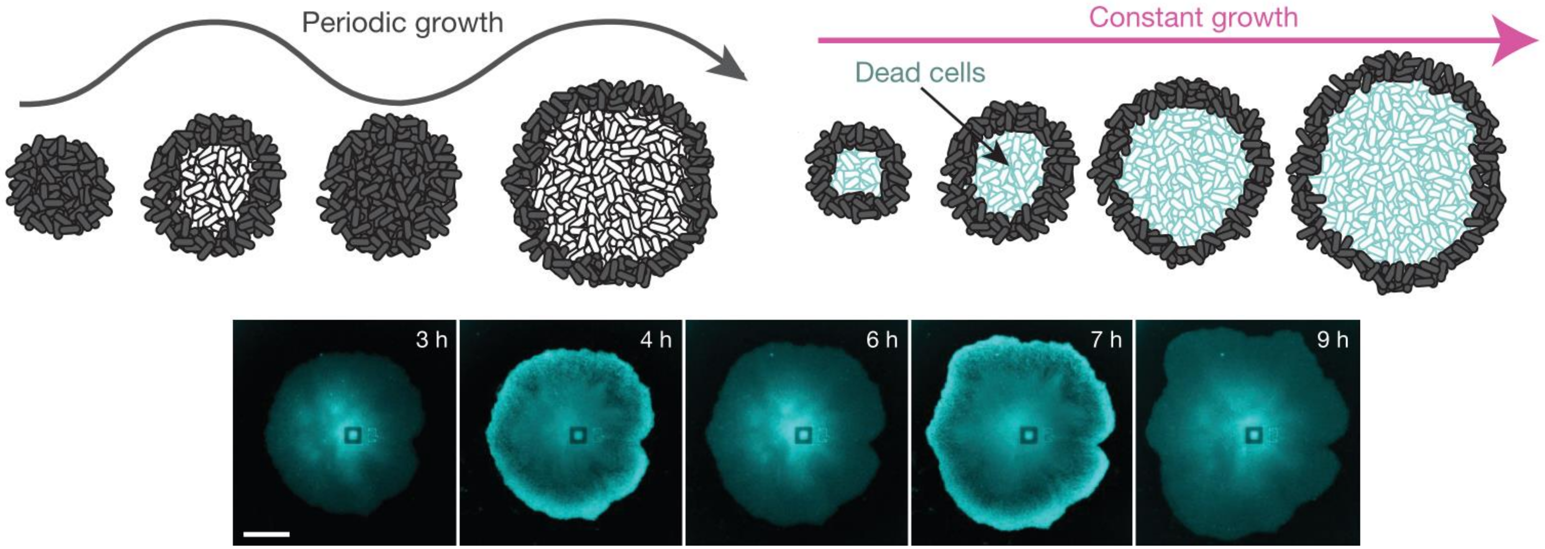
- Problem! No particle knows when it can use its stored energy to perform an action.

Biological Inspiration

Our approach alternates between selfish and altruistic energy usage, inspired by *Bacillus Subtilis* bacterial biofilm colonies [Liu and Prindle et al., Nature 2015].

There is a carefully balanced internal conflict for these biofilms: the peripheral bacteria protect the internal bacteria from attack, but also starve them of environmental nutrients.

The internal bacteria signal their stress (lack of nutrients) using long-range electrochemical signaling that temporarily inhibits the periphery from consuming nutrients, allowing more nutrients to reach the interior.



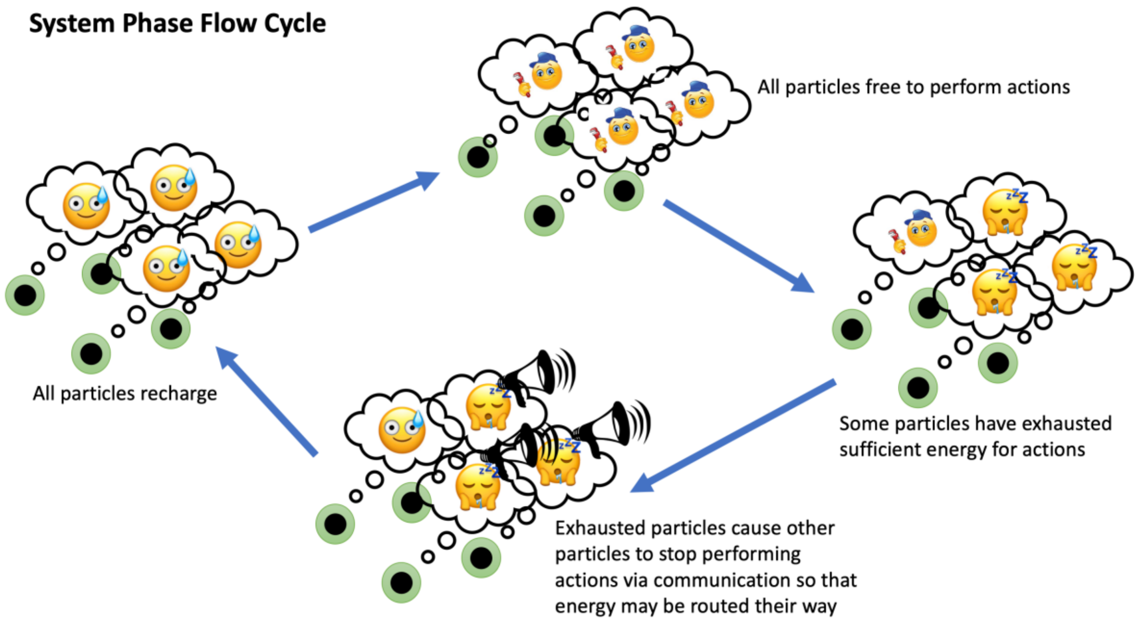
The Energy-Sharing Algorithm

Initially, the particle system self-organizes into a spanning forest rooted at particles with access to external energy sources. They then continuously loop through three phases:

1. Communication. Particles propagate signals from stressed particles towards their tree's root and propagate inhibition from their root to their descendants.

2. Sharing. Root particles harvest energy from the source and all particles attempt to transfer energy to one of their children.

3. Usage. Uninhibited particles spend energy on actions according to their collective behavior.



The Energy-Sharing Algorithm

Theorem. The Energy-Sharing algorithm solves the energy distribution problem in $\mathcal{O}(n)$ rounds, where n is the number of particles in the system.

Proof Outline:

- $\mathcal{O}(n)$ rounds to form the spanning forest. All trees are independent, so pick one tree $\mathcal{T}$.
- $2d_{\mathcal{T}}$ rounds to inhibit all particles in $\mathcal{T}$ when $\mathcal{T}$ contains a stressed particle.
- $\mathcal{O}(|\mathcal{T}|)$ rounds for all inhibited particles to fully recharge, in the worst case. (Tricky!)
- Within $2d_{\mathcal{T}}$ rounds, at least one particle will be uninhibited and perform its action.

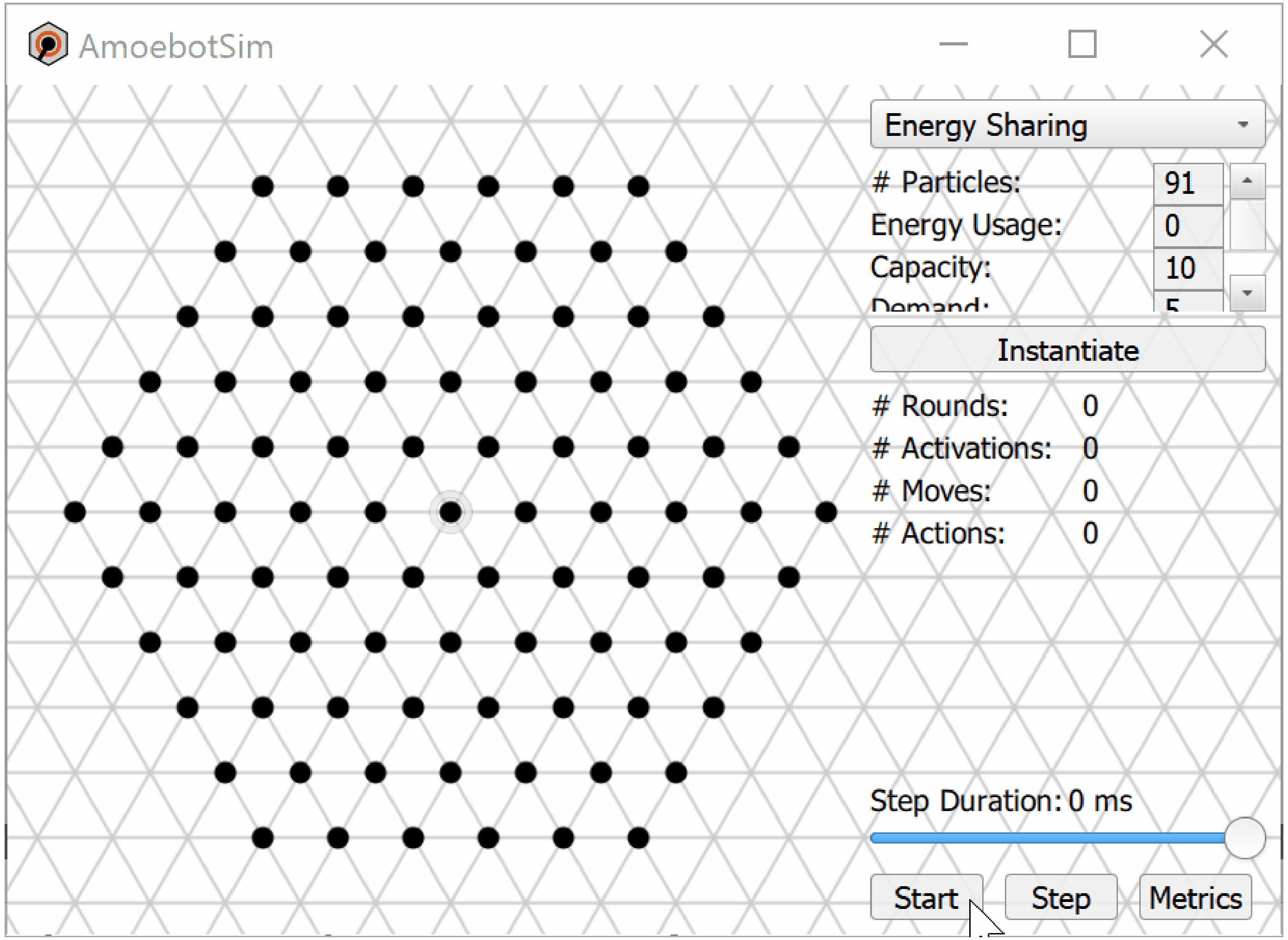
Theorem. In the worst-case, no local control algorithm can solve the energy distribution problem in fewer than $\Omega(n/s)$ rounds when s particles have access to external energy sources.

Proof Idea: Total energy to harvest is $n\kappa$, but only $s\alpha$ energy may be harvested per round.

Corollary. When s is a fixed constant, Energy-Sharing is asymptotically optimal.

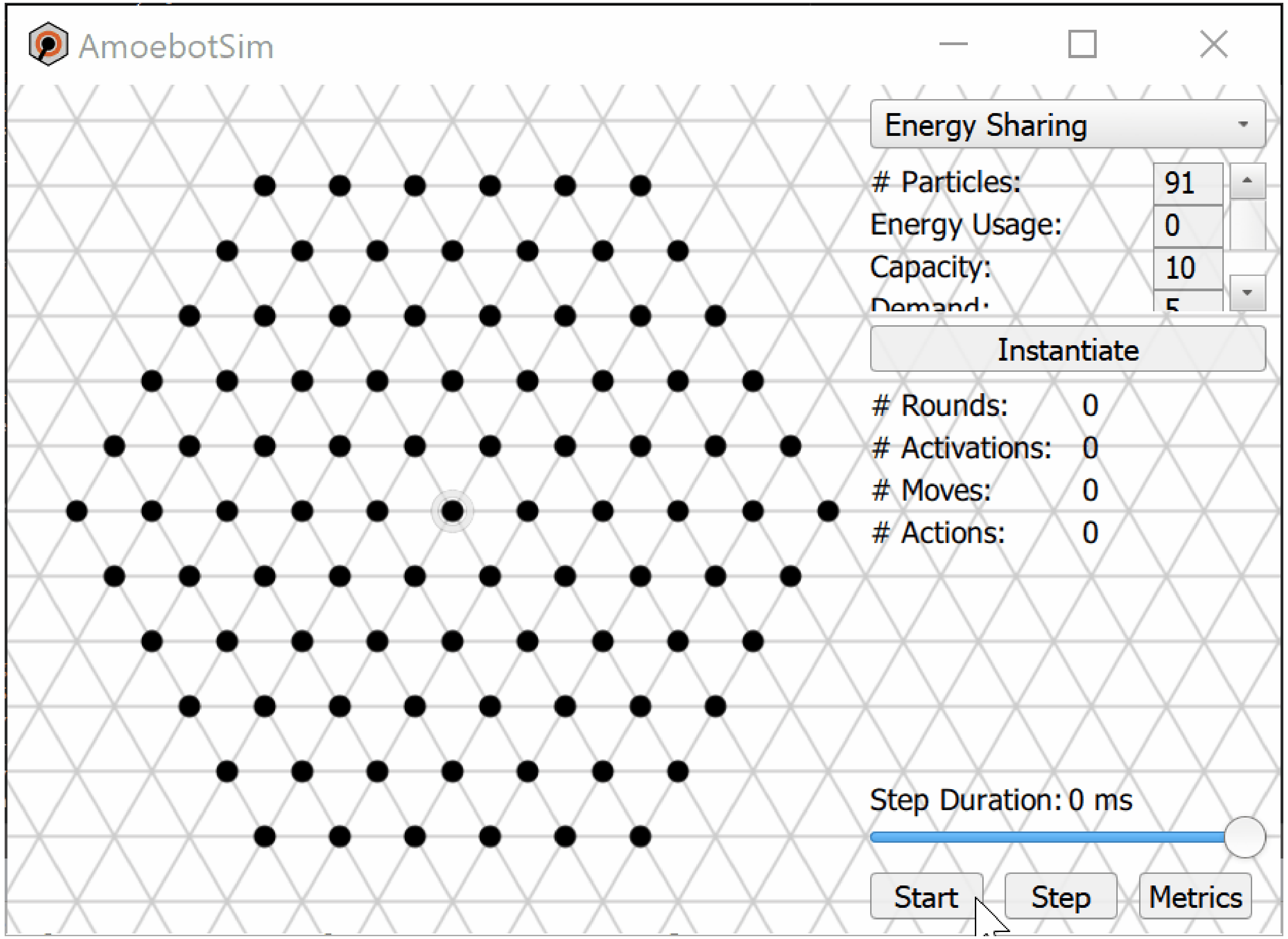
Simulations: Energy-Sharing

Energy-Sharing where each particle has a constant uniform energy demand.



Simulations: Energy-Sharing

Energy-Sharing without communication:



The Forest-Prune-Repair Algorithm

Crash failures pose a key challenge for Energy-Sharing: they disrupt the spanning forest used for communication and routing energy!

To address crash failures, we introduce **Forest-Prune-Repair**, an independent amoebot algorithm that can repair spanning forest structures.

Three (simplifying) assumptions:

1. Detection. The neighbors of a crashed particle can detect that it is crashed.
2. Connectivity. The non-crashed particles must remain connected.
3. Root-reliability. There must always be at least one non-crashed root particle.

The Forest-Prune-Repair Algorithm

The Forest-Prune-Repair algorithm works in two phases:

1. Pruning. When a particle sees that its parent is crashed, it sends a prune signal to all its descendants, causing them to reset their memories and clear their parent pointers. This dissolves the subtree rooted at the crashed particle.

2. Rejoining. Pruned particles choose one of their root or active neighbors to become their new parent (in a round-robin manner).

Theorem. The Forest-Prune-Repair algorithm repairs the spanning forest in $\mathcal{O}(m^2)$ rounds, where m is the number of particles disconnected from the forest by crash failures.

Proof Outline:

- $\mathcal{O}(d_{\mathcal{T}})$ rounds to prune a faulty subtree $\mathcal{T}$.
- A pruned particle can rejoin a faulty subtree at most 6 times before rejoining the forest.
- $d_{\mathcal{T}} \leq m$, so one disconnected particle rejoints the forest every $\mathcal{O}(m)$ rounds.

Algorithm Composition

We ultimately want to use Energy-Sharing as a subprocess that supplies energy for more complex collective behavior defined by higher-level algorithms.

But collective behaviors often involve movement, which would disrupt the spanning forest!

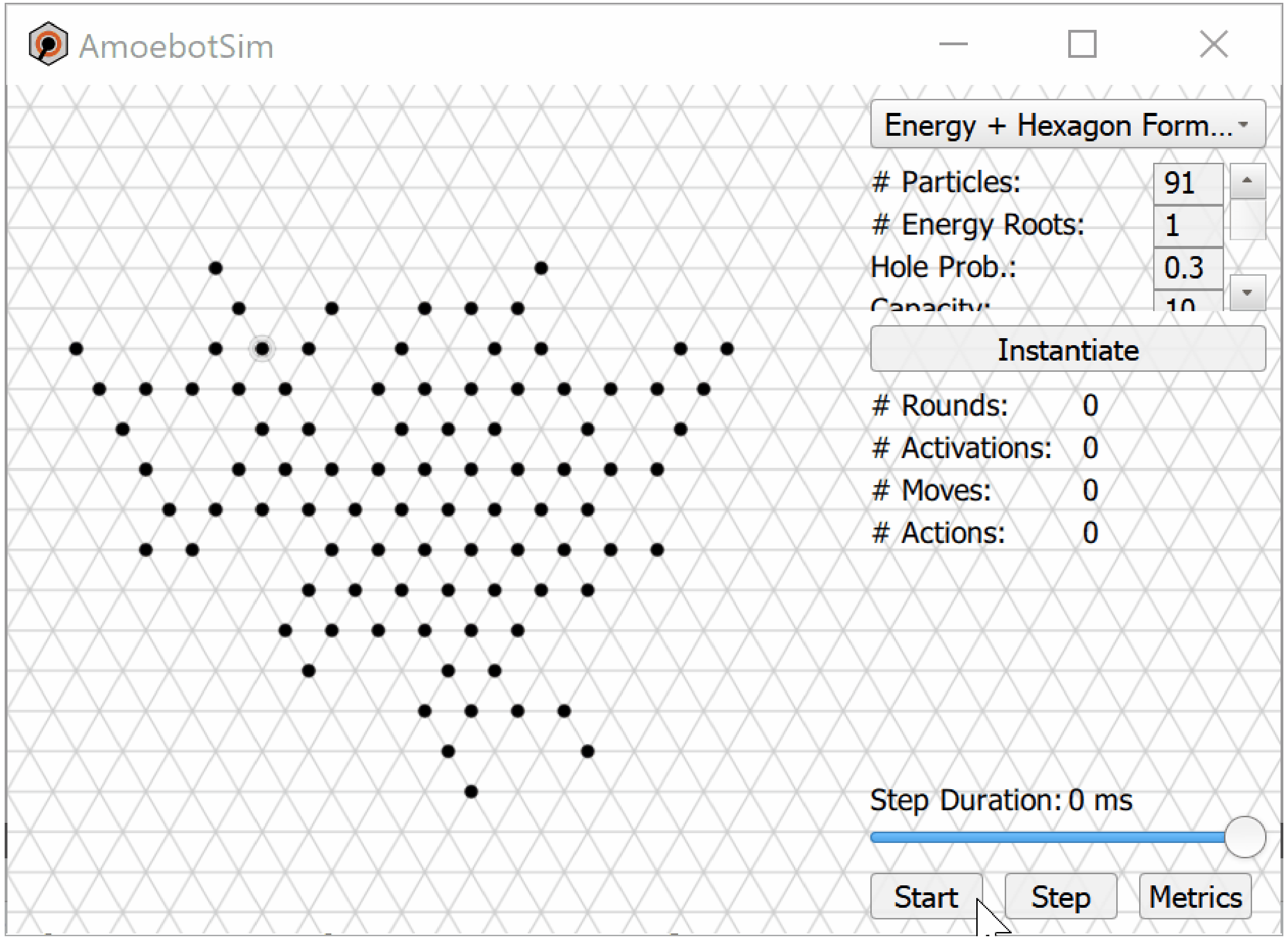
We repurpose Forest-Prune-Repair to address moving particles instead of crashing ones.

Thus, Energy-Sharing can be composed with an amoebot algorithm $\mathcal{A}$ if:

1. Each particle's battery capacity κ is at least as large as the demand of the most energy-intensive action in $\mathcal{A}$.
2. Algorithm $\mathcal{A}$ maintains system connectivity.

Simulations: Algorithm Composition

Energy-Sharing + Forest-Prune-Repair composed with **Hexagon-Formation**:

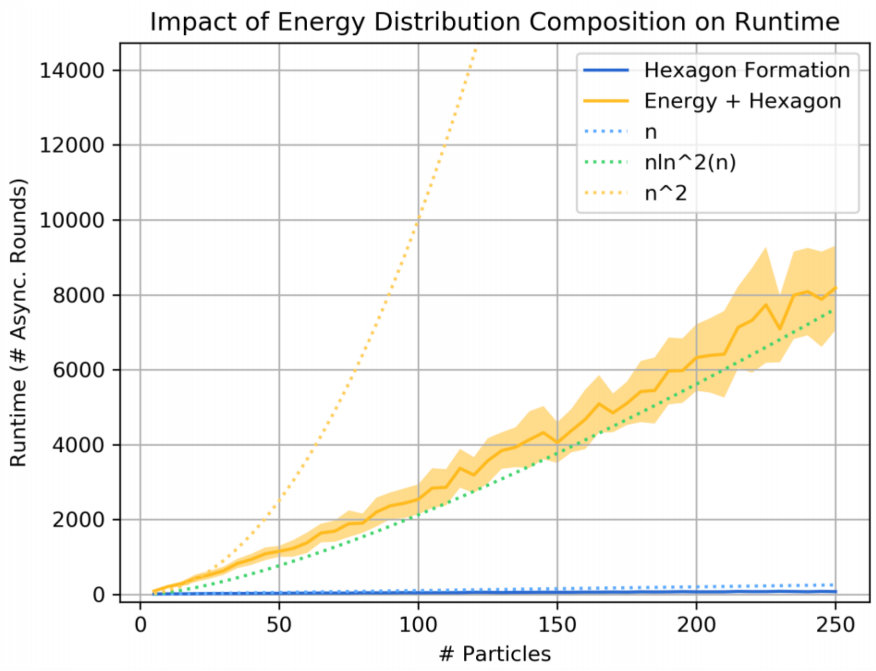


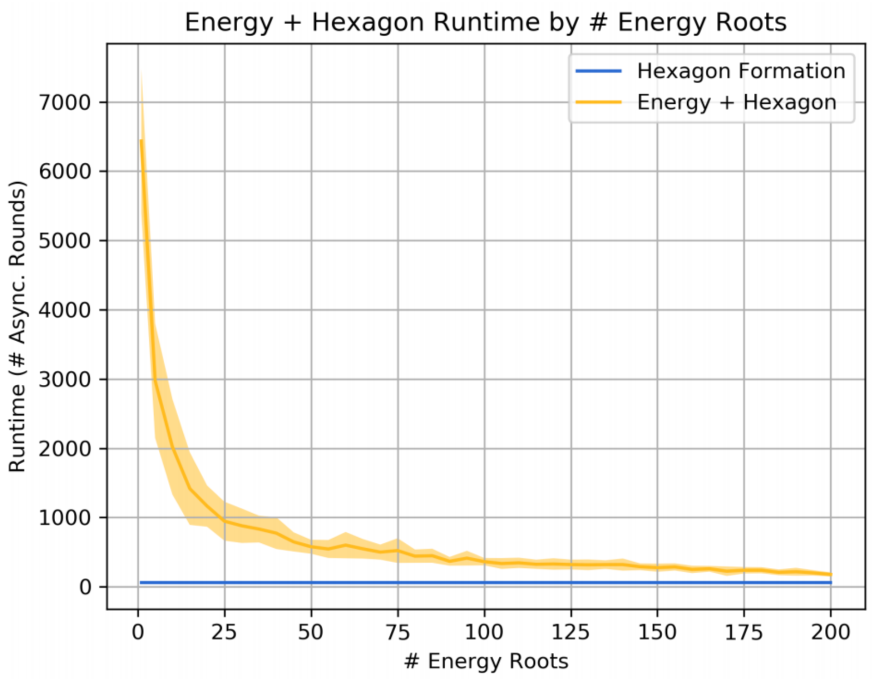
Simulations: Algorithm Composition

Energy-Sharing guarantees that some particle performs an action every $\mathcal{O}(n)$ rounds.

Forest-Prune-Repair rejoins m disconnected particles to the spanning forest in $\mathcal{O}(m^2)$ rounds.

Thus, theoretically the overhead on an algorithm can be quite high, though we observe a roughly $\mathcal{O}(n \log n)$ overhead for Hexagon-Formation.

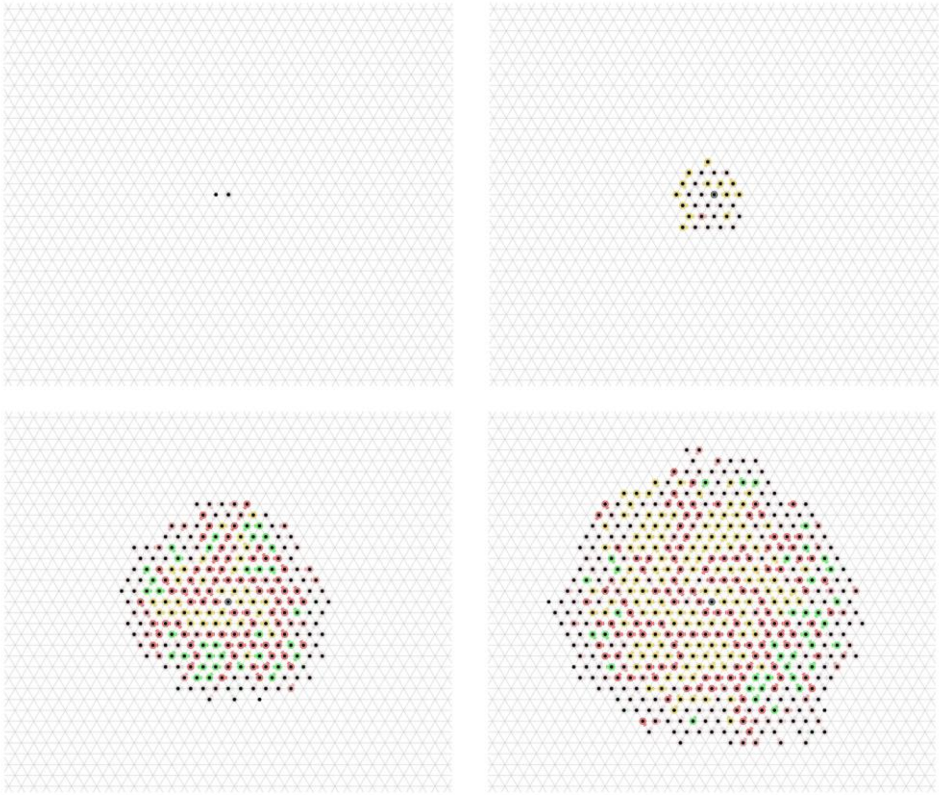


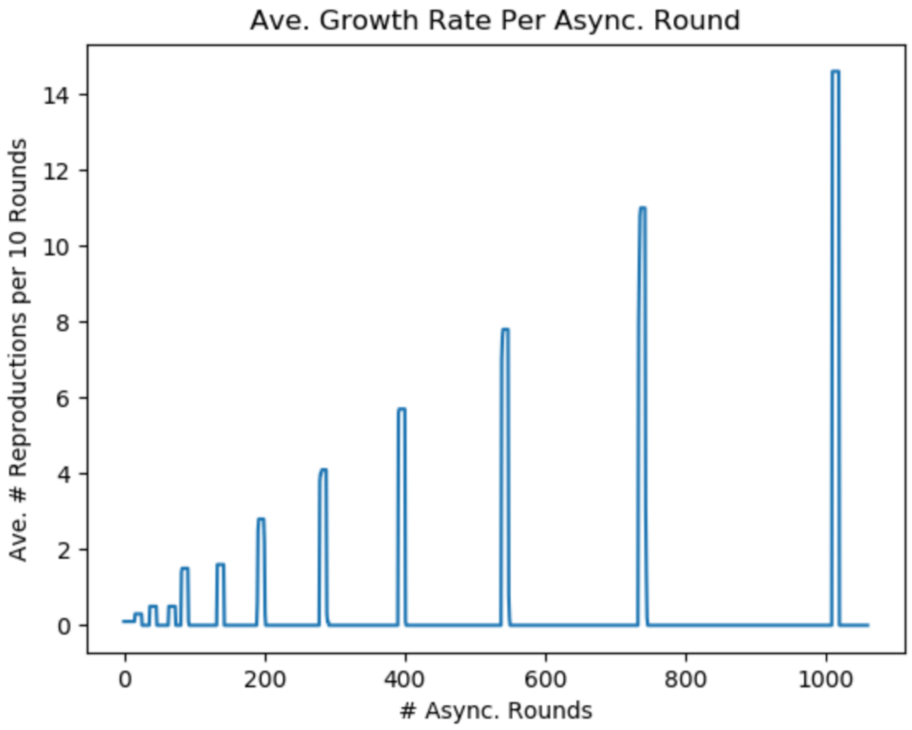


Dynamic Systems

As an informal analogy to the bacterial biofilms, we simulated **system growth** when energy is used for reproduction.

We observe an oscillating growth rate qualitatively similar to those observed for the biofilms.





Thank you!

sops.engineering.asu.edu/sops/energy-distribution