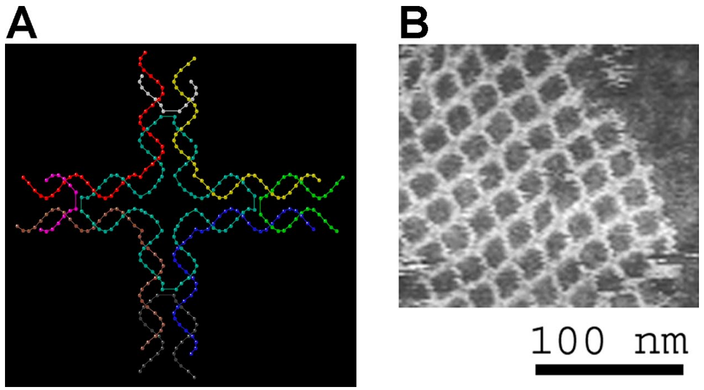
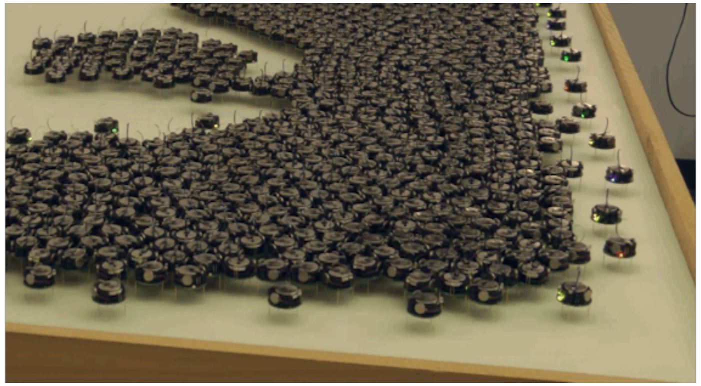
Convex Hull Formation for Programmable Matter

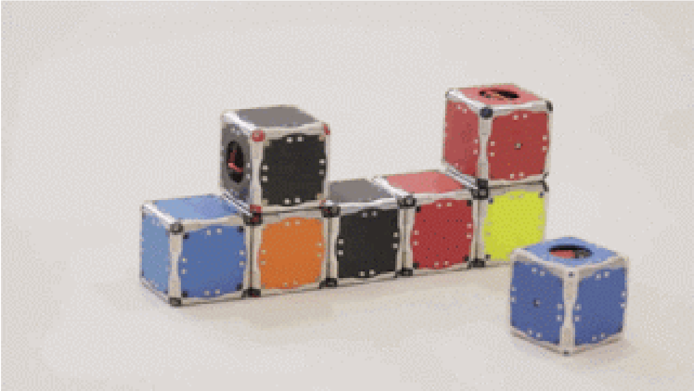
JOSHUA J. DAYMUDE AND ANDRÉA W. RICHA – ARIZONA STATE UNIVERSITY

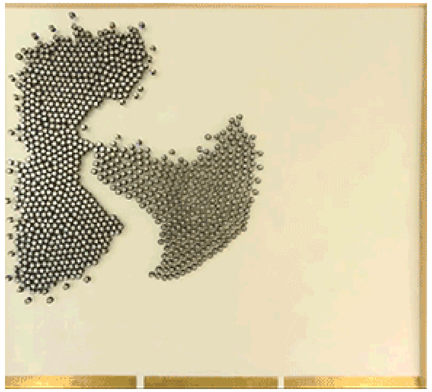
ROBERT GMYR, CHRISTIAN SCHEIDELER, AND THIM STROTHMANN – UNIVERSITY OF PADERBORN

Current Programmable Matter



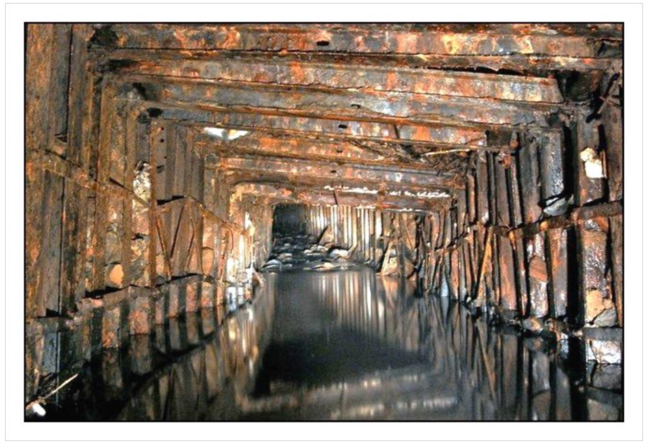


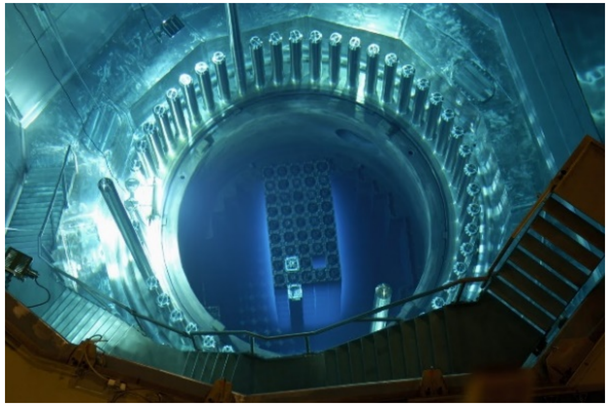


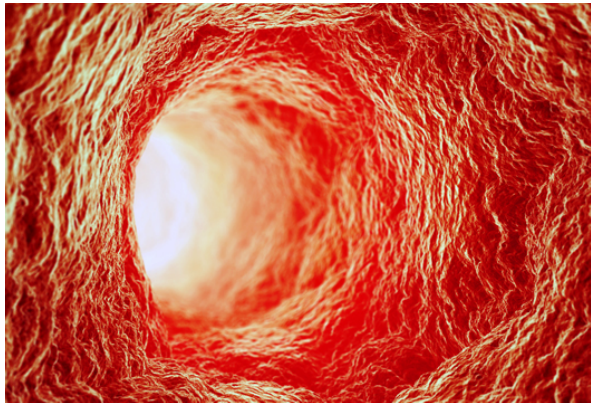


[1] RGR 2013: "M-blocks: Momentum driven, magnetic modular robots" [2] RCN 2014: "Programmable self-assembly in a thousand-robot swarm"

Inspirations & Applications



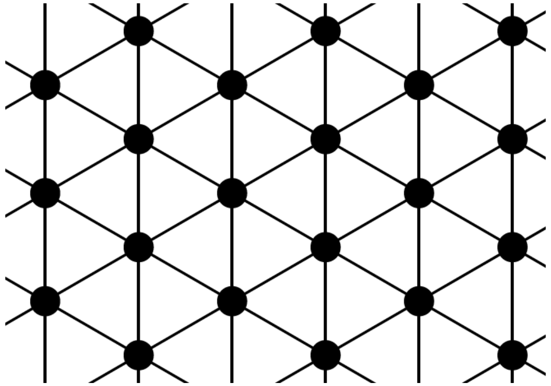


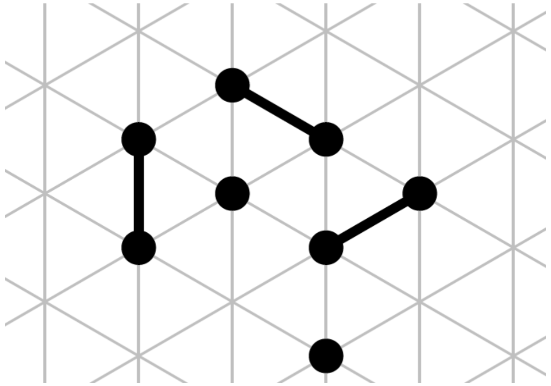


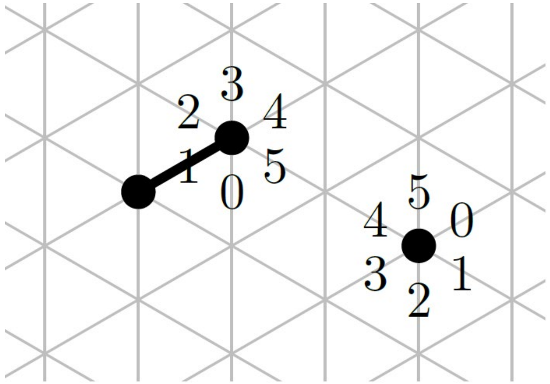
The Amoebot Model

Particles move by *expanding* and *contracting,* and are:

- Anonymous (no unique identifiers)
- Without global orientation or compass (no shared sense of "north")
- Limited in memory (constant size)
- Activated asynchronously





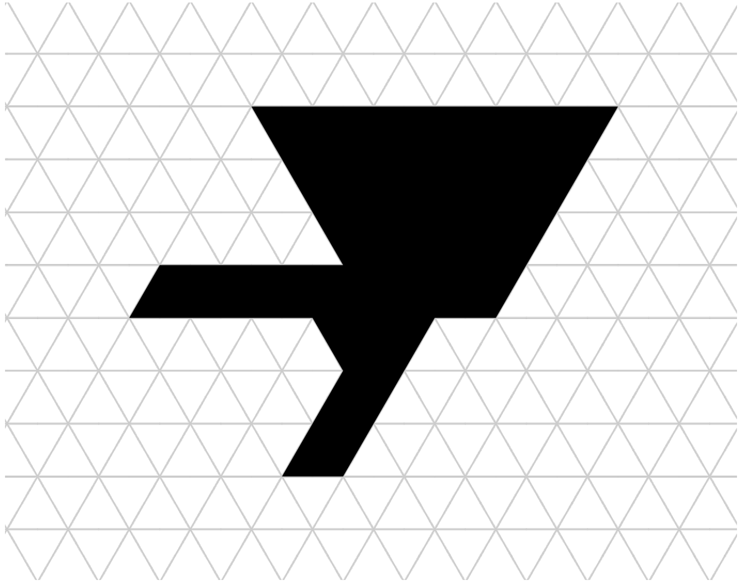


Our Past Work

- Leader Election [DNA21, ALGOSENSORS '17]
- Shape Formation [NANOCOM '15, SPAA '16]
- Object Coating [*Theoretical Computer Science, Natural Computing*]
- Full list of publications can be found at: sops.engineering.asu.edu/publications-press/.

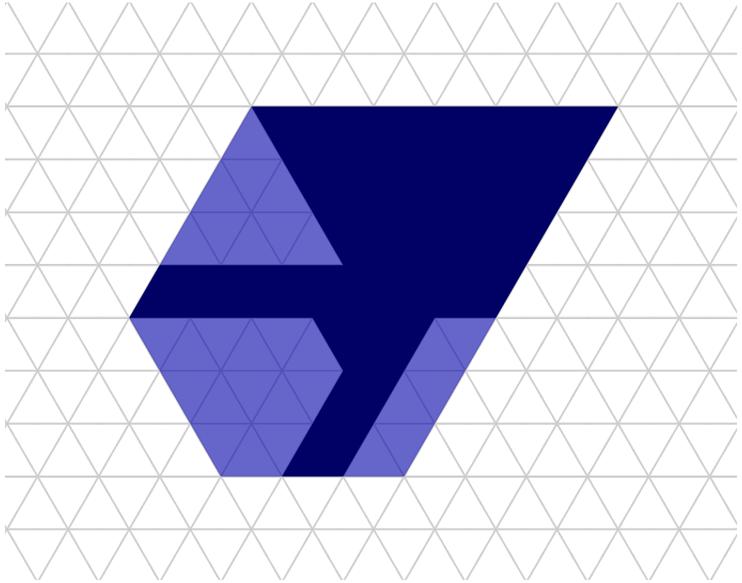
Convex Hull: Definitions

- We begin with an object *O*, which is a connected set of nodes in our graph *G = (V, E)*.



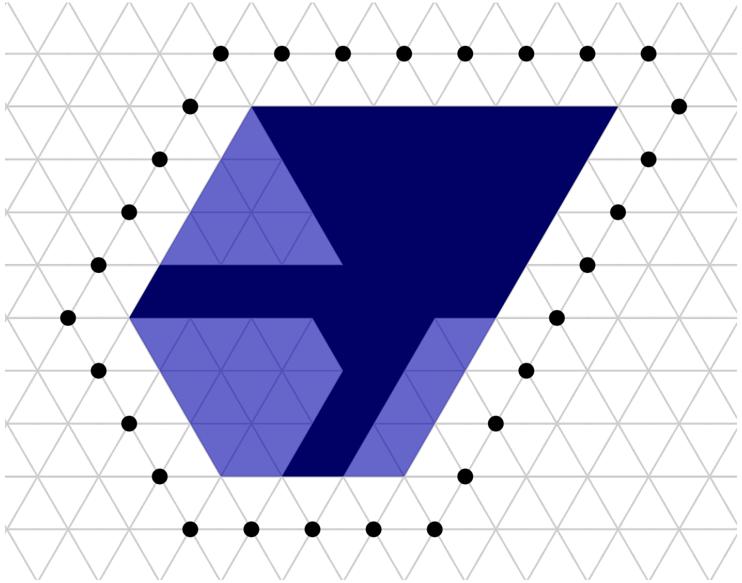
Convex Hull: Definitions

- We begin with an object *O*, which is a connected set of nodes in our graph *G = (V, E)*.
- Let *O** be the minimal convex set of nodes containing *O*.



Convex Hull: Definitions

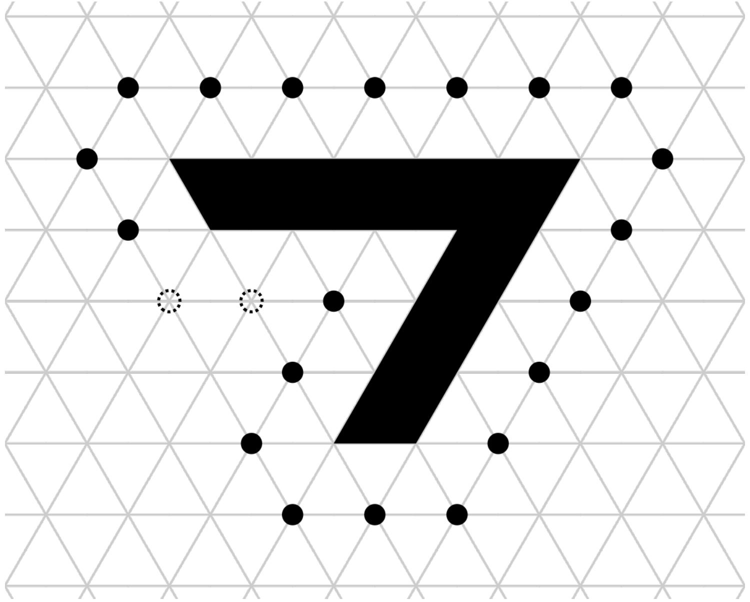
- We begin with an object *O*, which is a connected set of nodes in our graph *G = (V, E)*.
- Let *O** be the minimal convex set of nodes containing *O*.
- The *convex hull* of *O*, denoted *C(O)*, is the set of nodes in *V* \ *O** adjacent to some node(s) of *O**. (Essentially the "external boundary" of *O**).



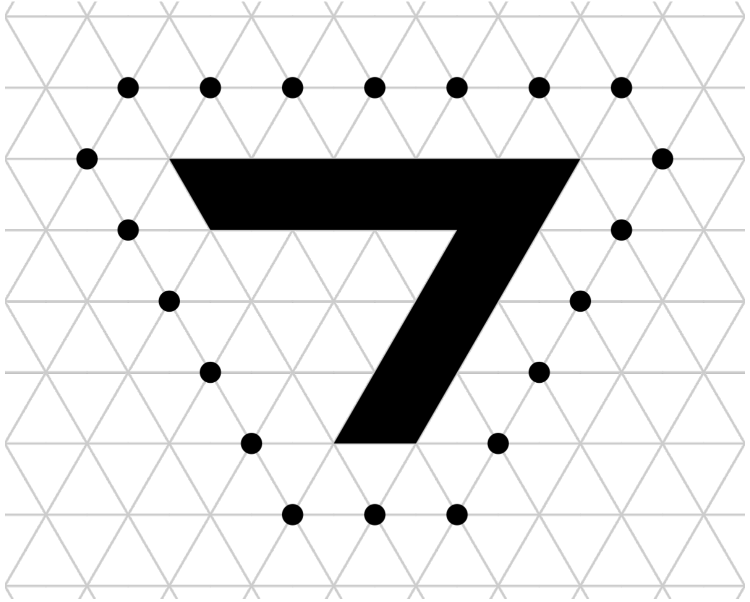
Why Convex Hulls?

- Interesting problem in computational geometry, especially in distributed settings.
- Can be viewed as a relaxation of object coating.

Incomplete Coating



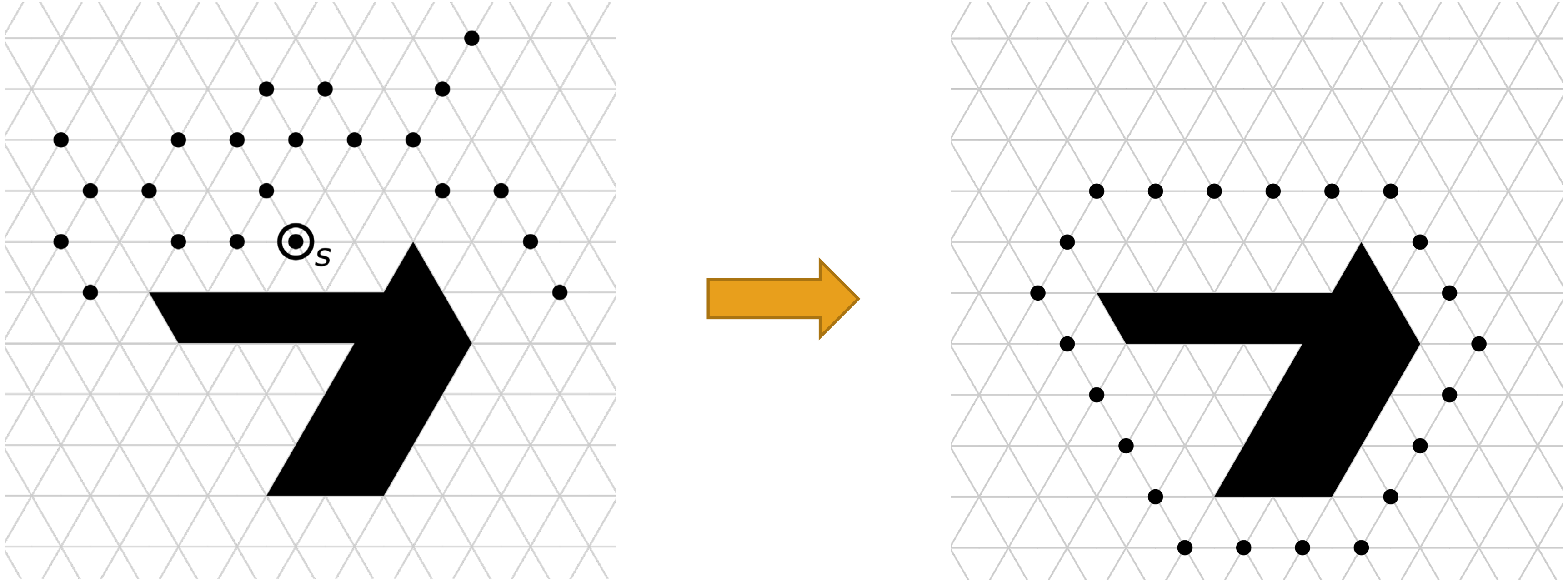
Convex Hull



Our Goal

Given: a connected object *O* with no holes, a connected particle system *P* such that $|P| \geq |C(O)|$, and a unique seed particle *s* which is adjacent to *O.*

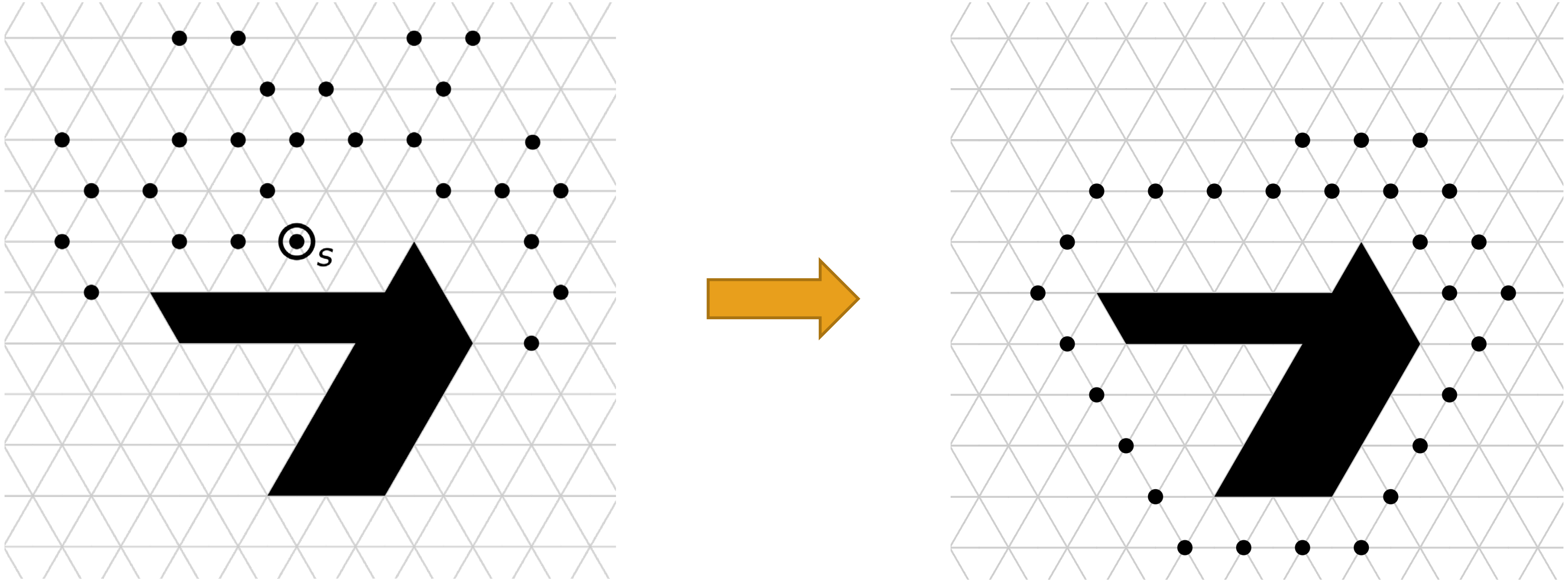
Goal: reconfigure *P* so that every node of *C(O)* is occupied by a contracted particle.



Our Goal

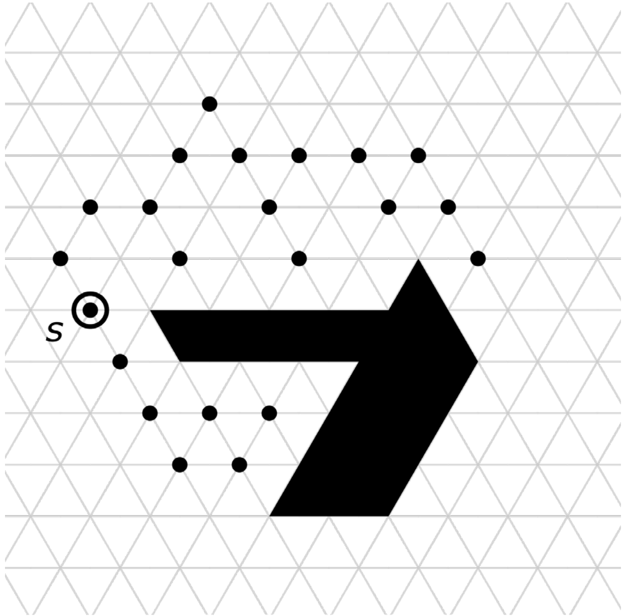
Given: a connected object *O* with no holes, a connected particle system *P* such that $|P| \geq |C(O)|$, and a unique seed particle *s* which is adjacent to *O*.

Goal: reconfigure *P* so that every node of *C(O)* is occupied by a contracted particle.



Algorithm: High Level

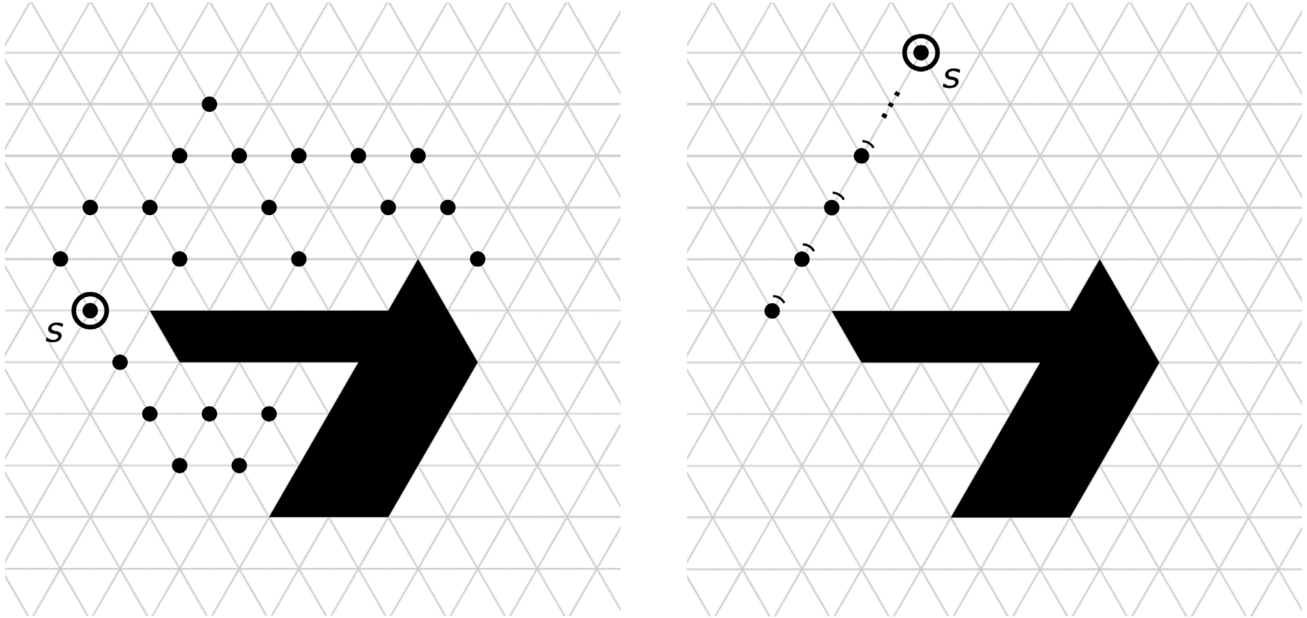
Our algorithm is broken up into two main phases:



Algorithm: High Level

Our algorithm is broken up into two main phases:

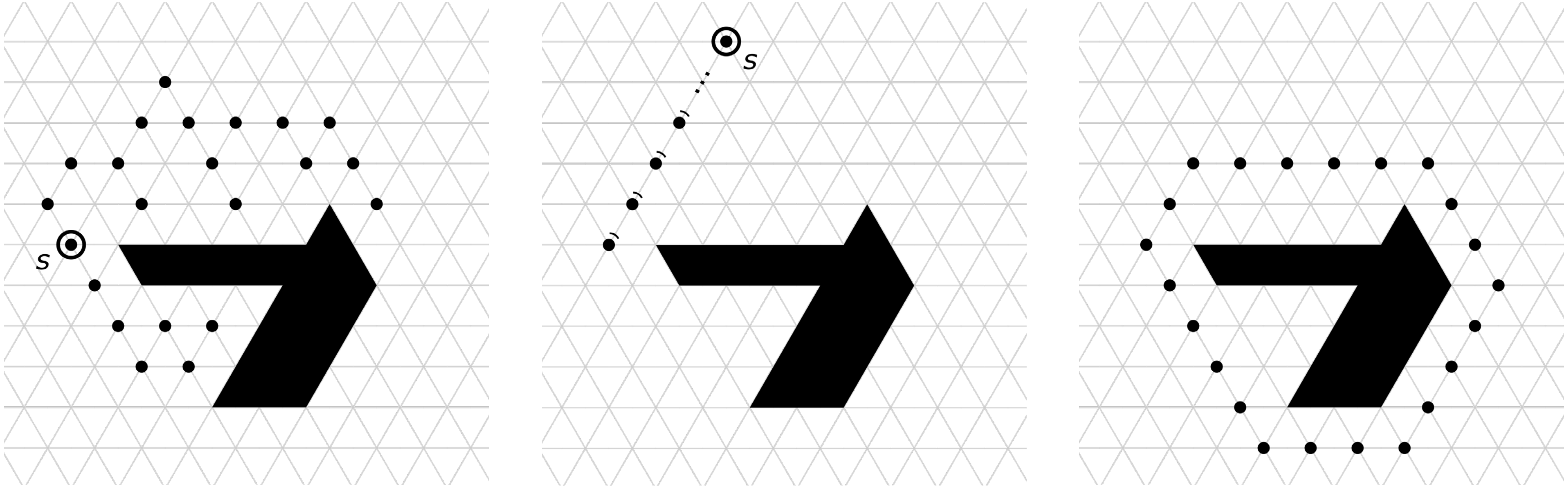
1. Phase I: Escaping the Object



Algorithm: High Level

Our algorithm is broken up into two main phases:

1. Phase I: Escaping the Object
2. Phase II: Constructing the Convex Hull



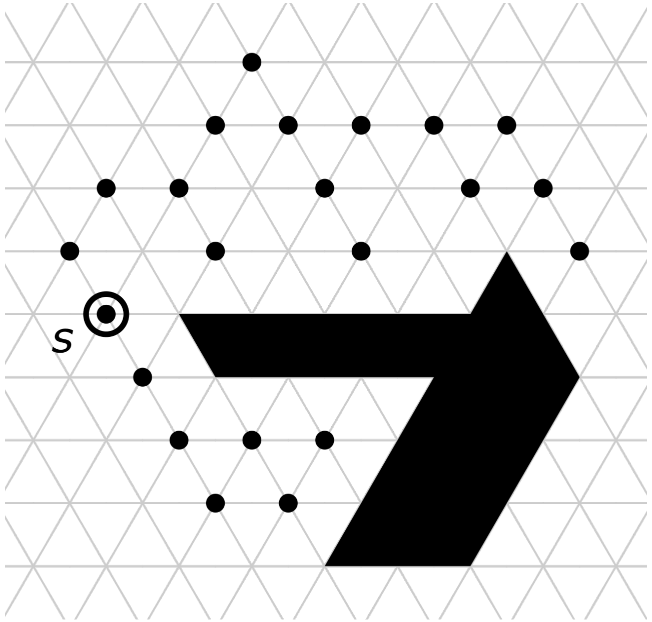
Phase I: Escaping the Object

Phase I is responsible for reorganizing P into a straight line of particles, which must necessarily reach outside O^* (recall: $|P| \geq |C(O)|$).

Phase I: Escaping the Object

Phase I is responsible for reorganizing P into a straight line of particles, which must necessarily reach outside O^* (recall: $|P| \geq |C(O)|$).

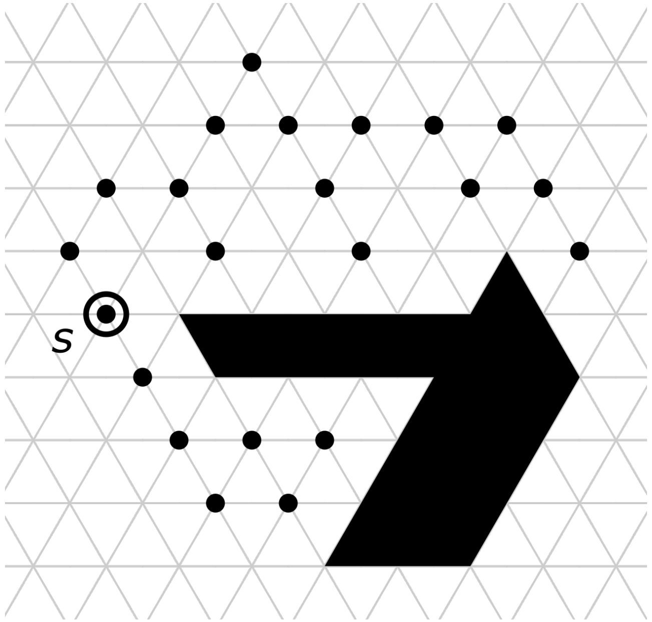
First, organize P using the "spanning forest primitive".

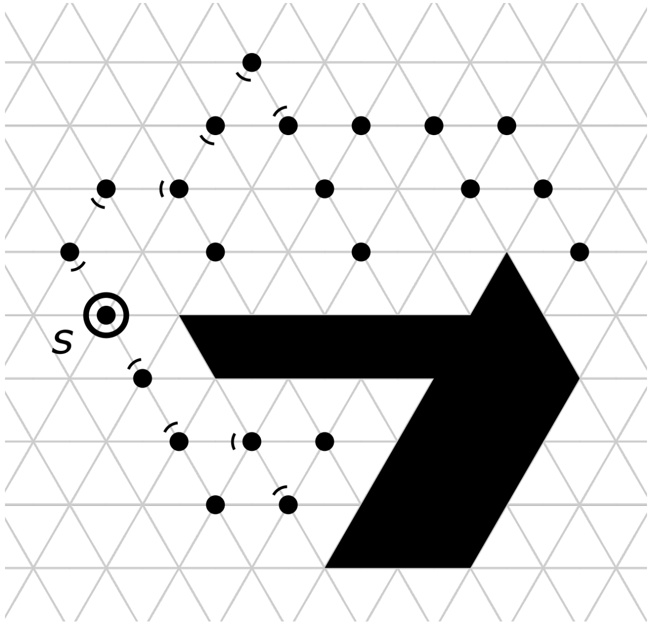


Phase I: Escaping the Object

Phase I is responsible for reorganizing *P* into a straight line of particles, which must necessarily reach outside *O** (recall: $|P| \geq |C(O)|$).

First, organize *P* using the "spanning forest primitive".

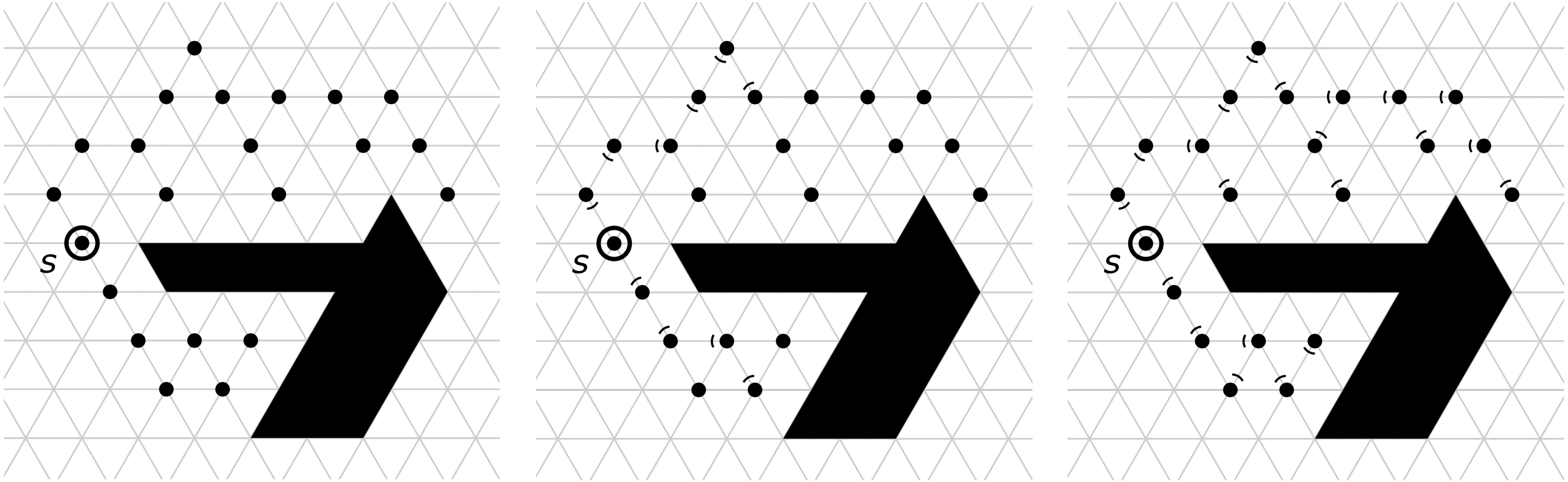




Phase I: Escaping the Object

Phase I is responsible for reorganizing *P* into a straight line of particles, which must necessarily reach outside *O** (recall: $|P| \geq |C(O)|$).

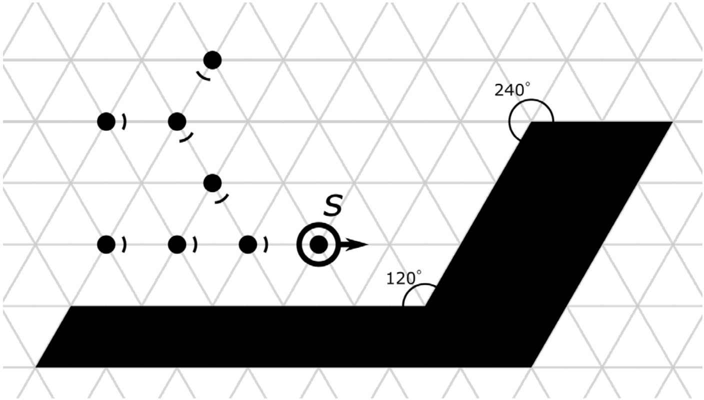
First, organize *P* using the "spanning forest primitive".



Phase I: Escaping the Object

Next, alternate between the following subphases until Phase I is complete:

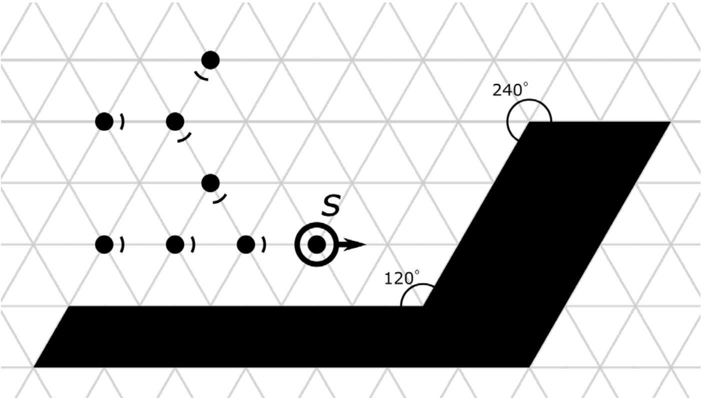
1. Wall Following: Follow the object using right-hand-rule until finding a “concave turn”.

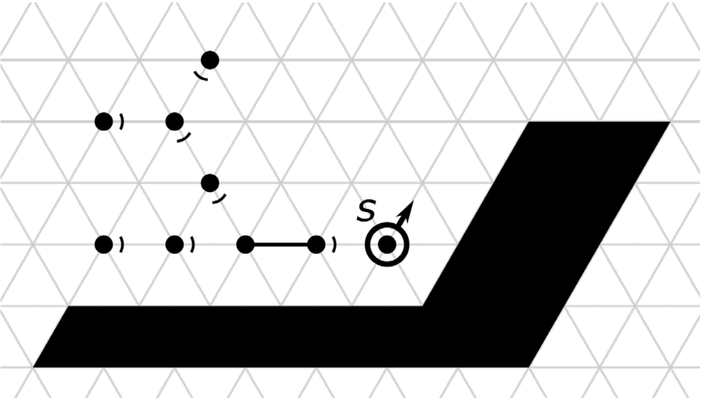


Phase I: Escaping the Object

Next, alternate between the following subphases until Phase I is complete:

1. Wall Following: Follow the object using right-hand-rule until finding a “concave turn”.

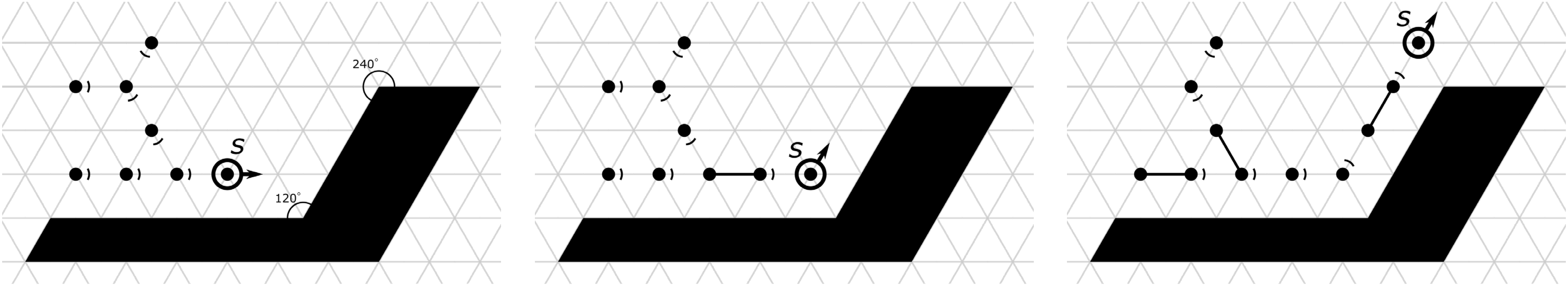




Phase I: Escaping the Object

Next, alternate between the following subphases until Phase I is complete:

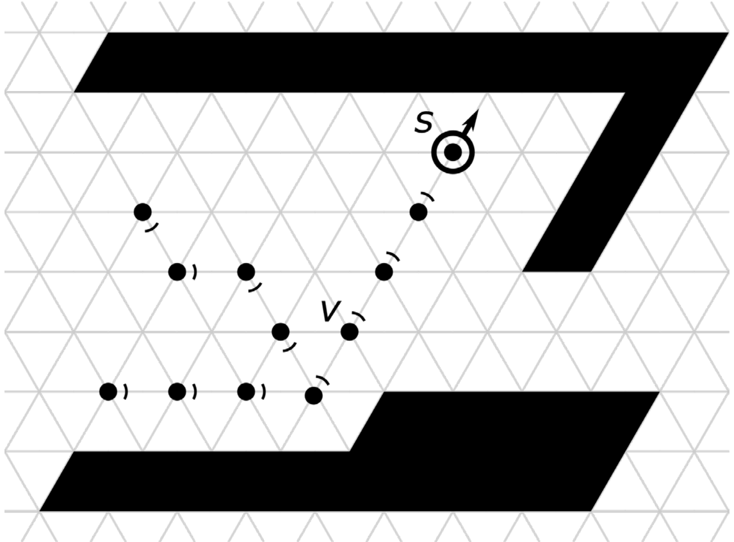
1. Wall Following: Follow the object using right-hand-rule until finding a "concave turn".



Phase I: Escaping the Object

Next, alternate between the following subphases until Phase I is complete:

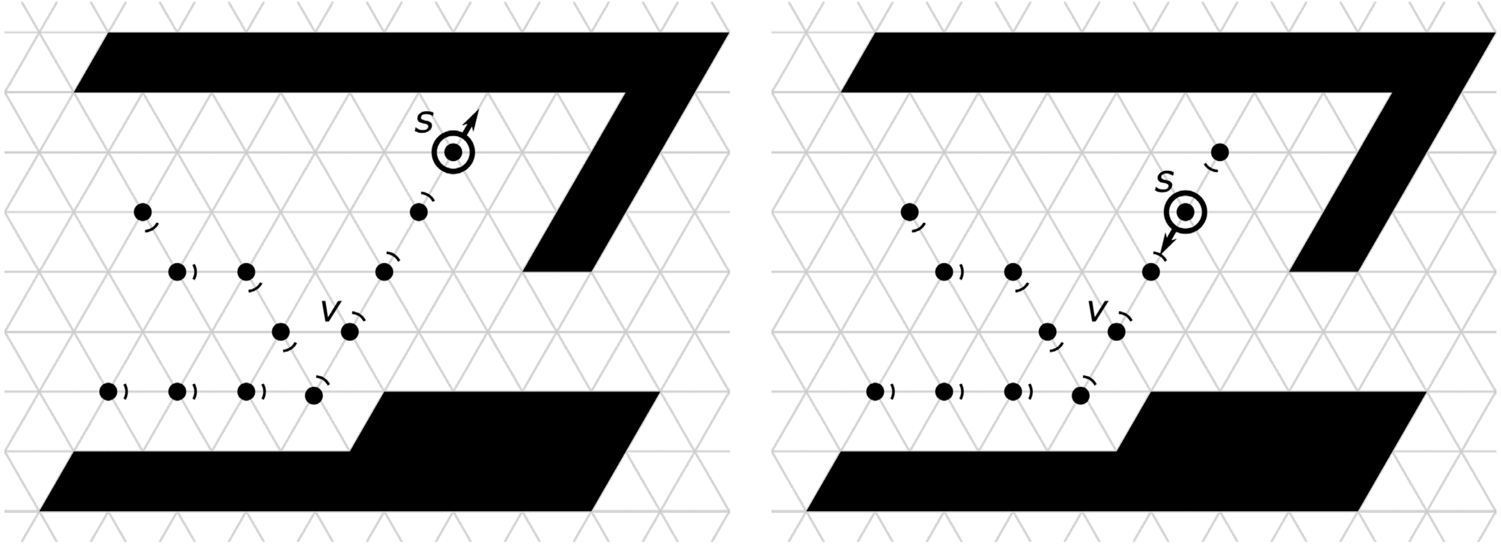
1. Wall Following: Follow the object using right-hand-rule until finding a “concave turn”.
2. Line Probing: Attempt to build the desired line, and backtrack on failure.



Phase I: Escaping the Object

Next, alternate between the following subphases until Phase I is complete:

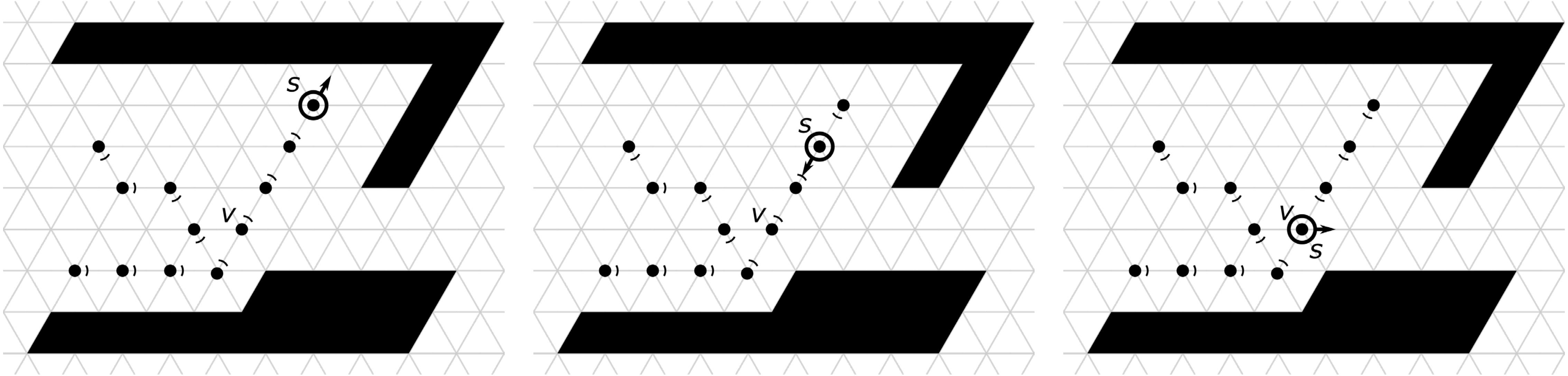
1. Wall Following: Follow the object using right-hand-rule until finding a “concave turn”.
2. Line Probing: Attempt to build the desired line, and backtrack on failure.



Phase I: Escaping the Object

Next, alternate between the following subphases until Phase I is complete:

1. Wall Following: Follow the object using right-hand-rule until finding a “concave turn”.
2. Line Probing: Attempt to build the desired line, and backtrack on failure.



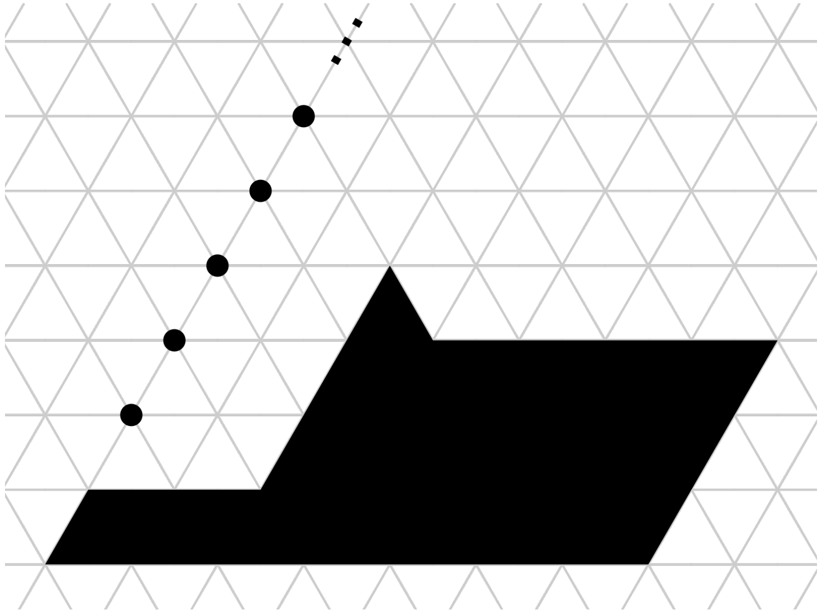
Line Bending: A Helpful Tool for Phase II

Bending a straight line by some angle is easy in a synchronous setting, but we have asynchronous activations.

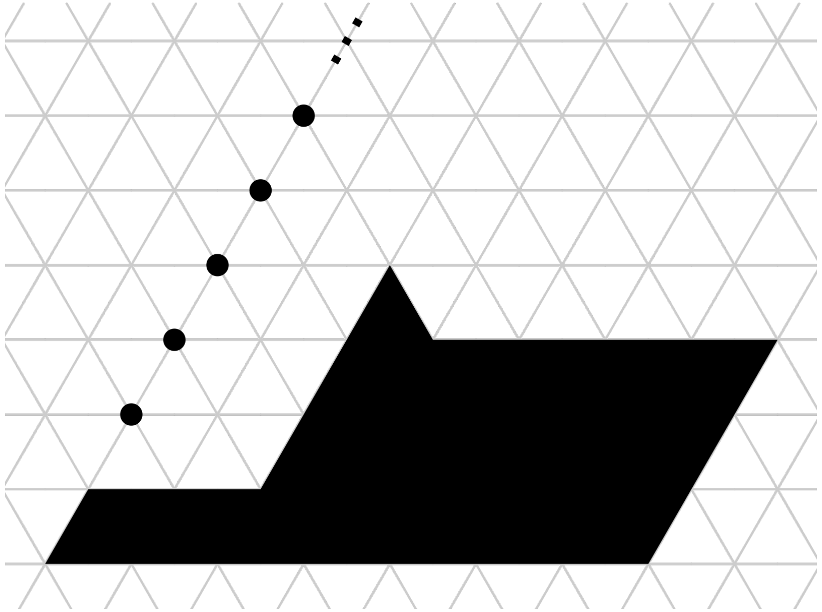
Line Bending: A Helpful Tool for Phase II

Bending a straight line by some angle is easy in a synchronous setting, but we have asynchronous activations.

Synchronous



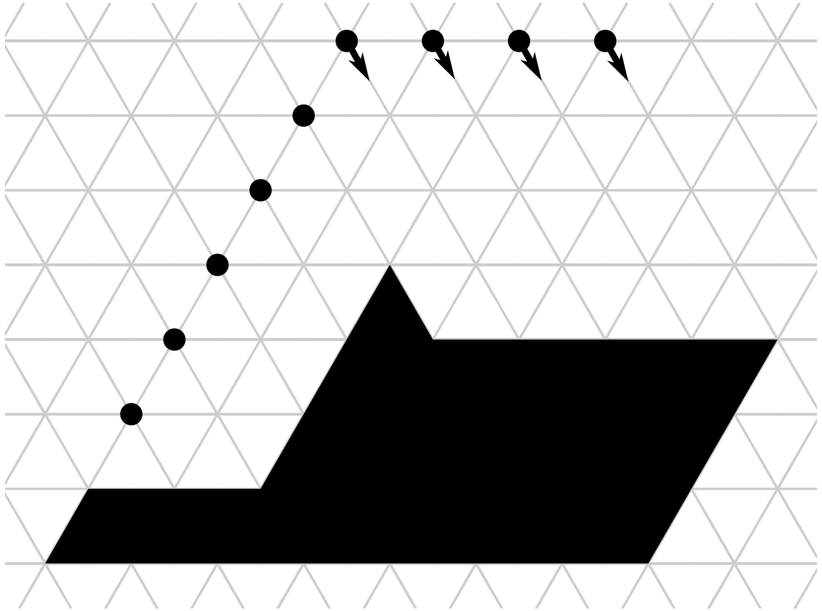
Asynchronous



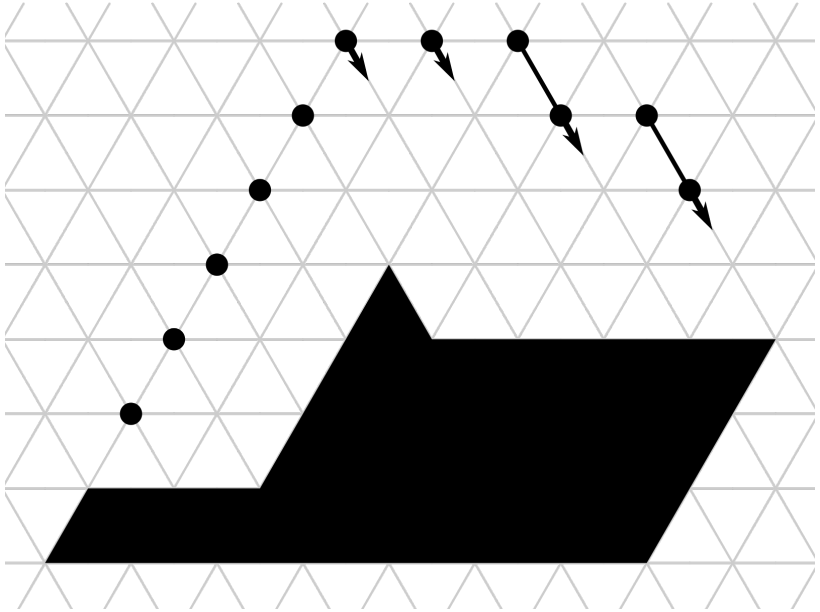
Line Bending: A Helpful Tool for Phase II

Bending a straight line by some angle is easy in a synchronous setting, but we have asynchronous activations.

<u>Synchronous</u>



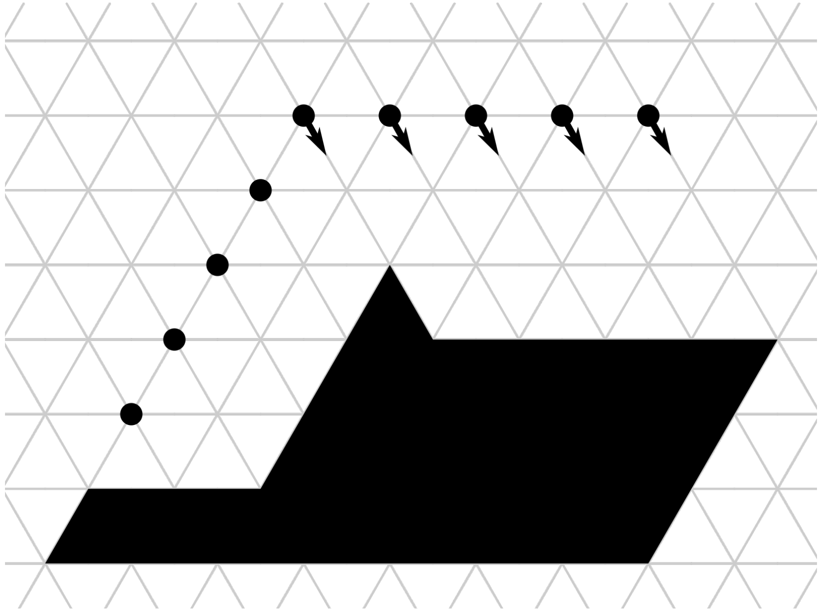
<u>Asynchronous</u>



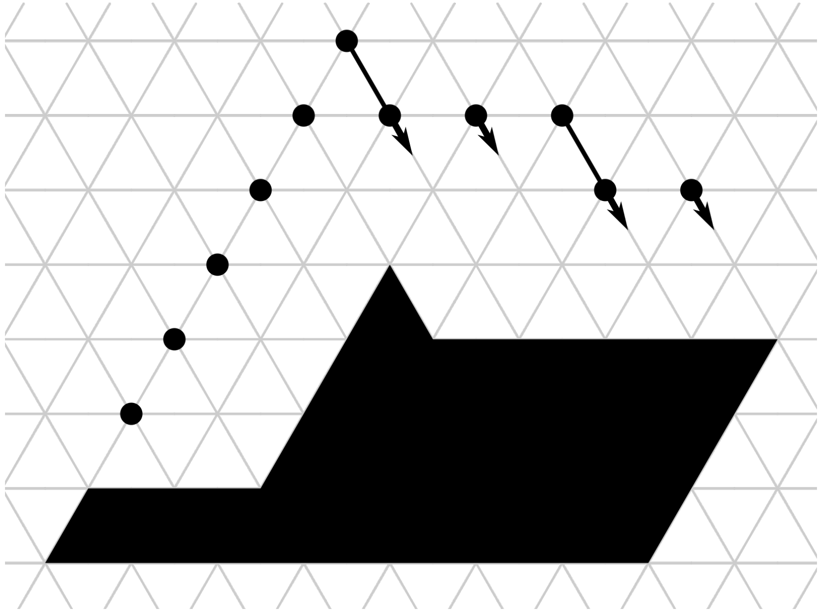
Line Bending: A Helpful Tool for Phase II

Bending a straight line by some angle is easy in a synchronous setting, but we have asynchronous activations.

Synchronous



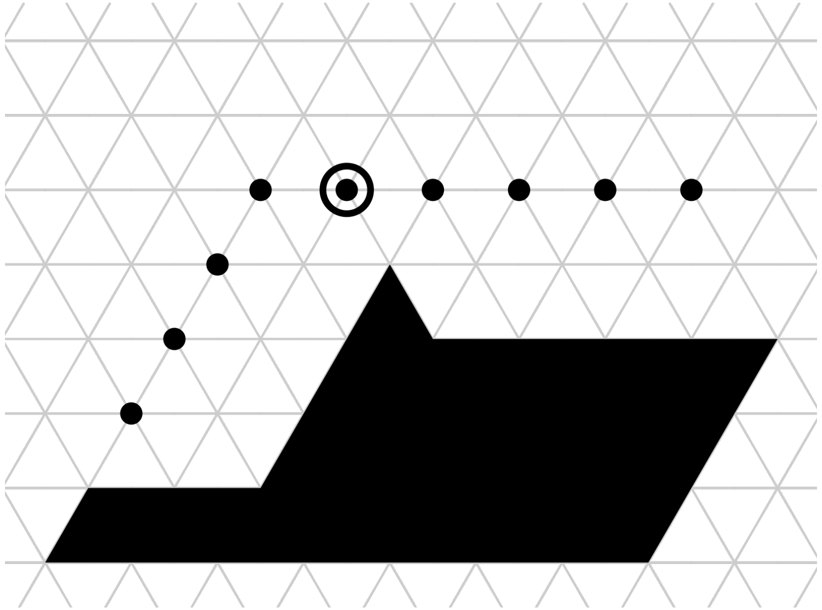
Asynchronous



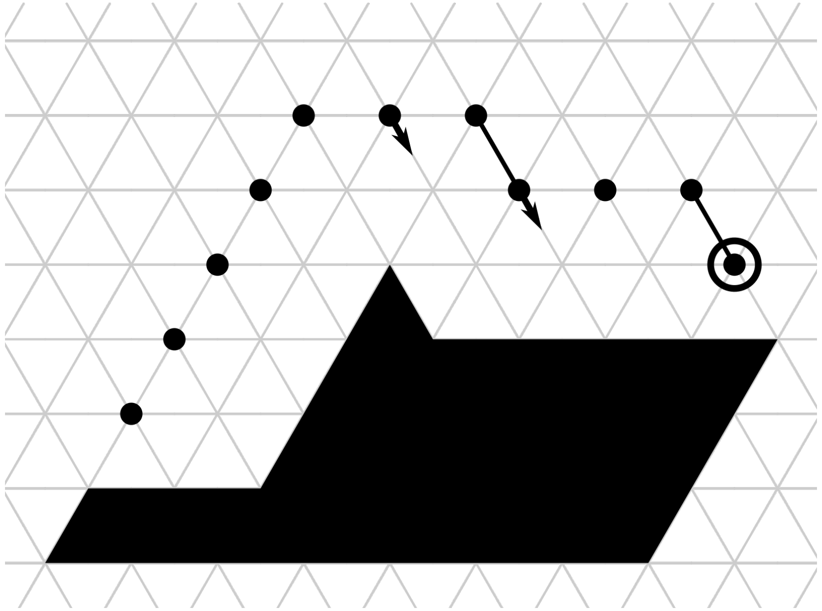
Line Bending: A Helpful Tool for Phase II

Bending a straight line by some angle is easy in a synchronous setting, but we have asynchronous activations.

Synchronous



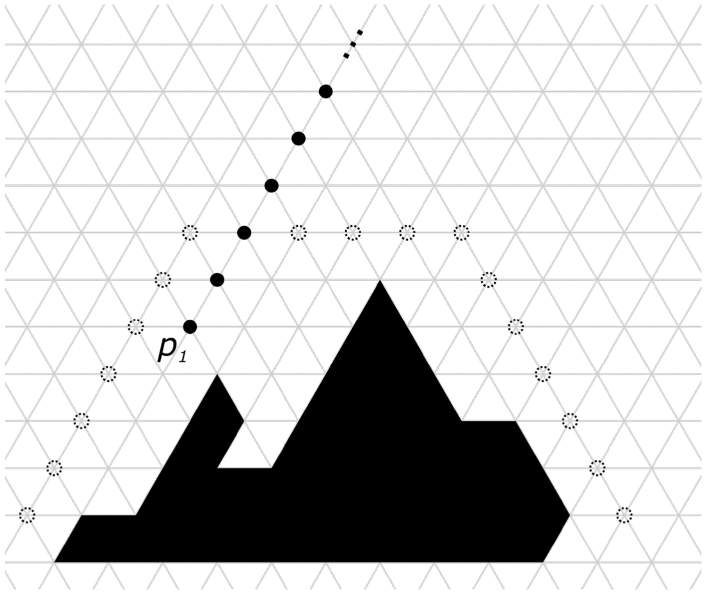
Asynchronous



Phase II: Constructing the Convex Hull

Phase II begins from the straight line of particles obtained in Phase I. This phase is also divided into two subphases (but these don't alternate):

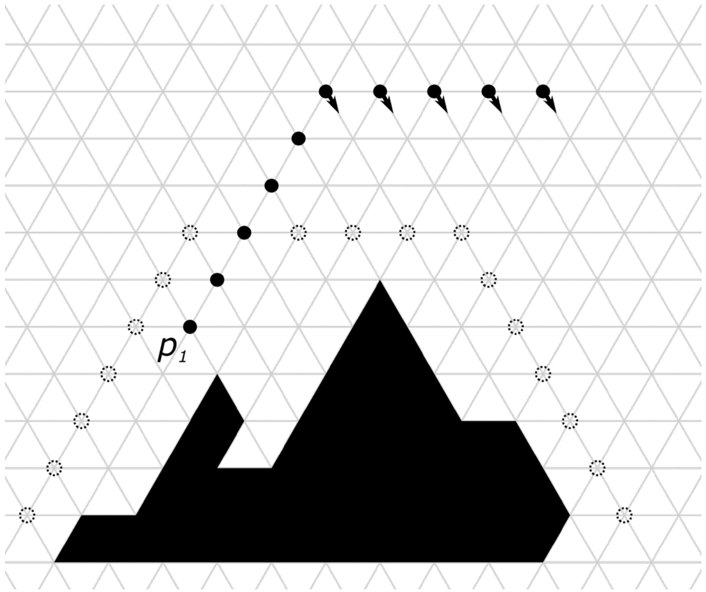
1. Moving the root of the particle line to some position in *C(O)* by bending + forwarding.



Phase II: Constructing the Convex Hull

Phase II begins from the straight line of particles obtained in Phase I. This phase is also divided into two subphases (but these don't alternate):

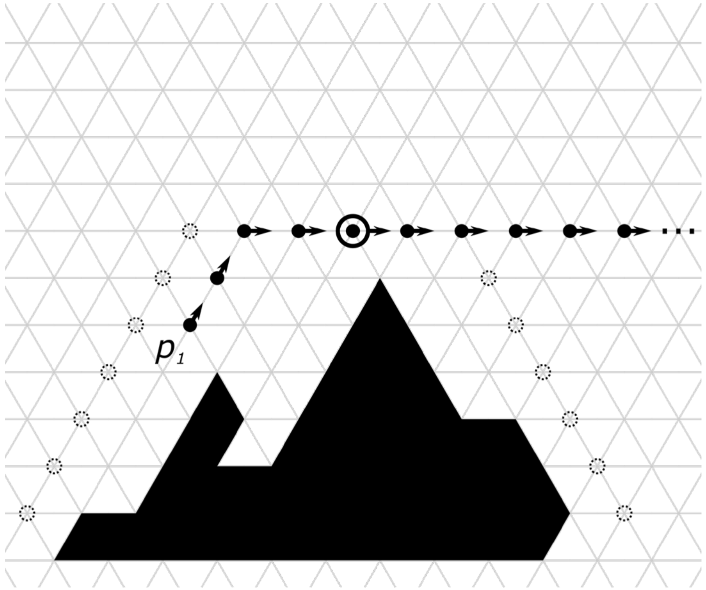
1. Moving the root of the particle line to some position in *C(O)* by bending + forwarding.



Phase II: Constructing the Convex Hull

Phase II begins from the straight line of particles obtained in Phase I. This phase is also divided into two subphases (but these don't alternate):

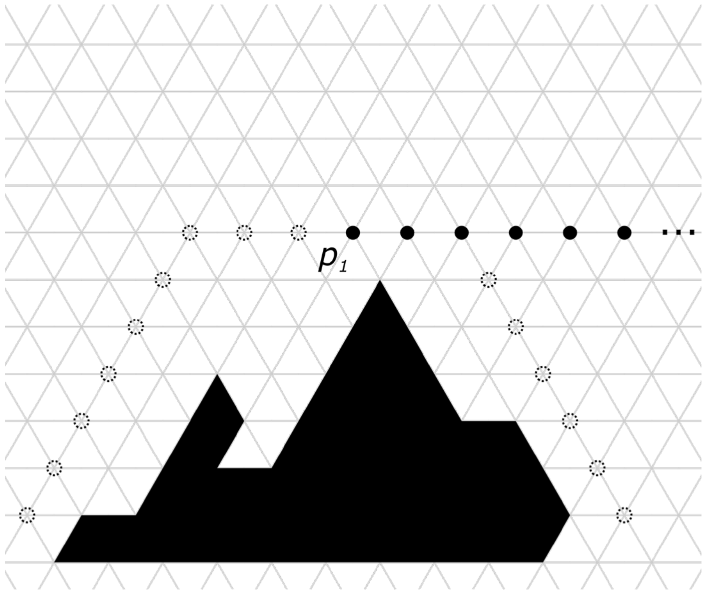
1. Moving the root of the particle line to some position in *C(O)* by bending + forwarding.



Phase II: Constructing the Convex Hull

Phase II begins from the straight line of particles obtained in Phase I. This phase is also divided into two subphases (but these don't alternate):

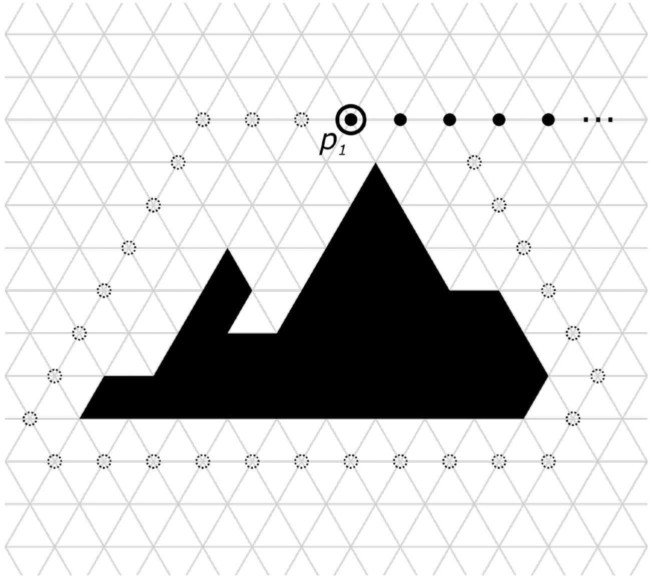
1. Moving the root of the particle line to some position in *C(O)* by bending + forwarding.



Phase II: Constructing the Convex Hull

Phase II begins from the straight line of particles obtained in Phase I. This phase is also divided into two subphases (but these don't alternate):

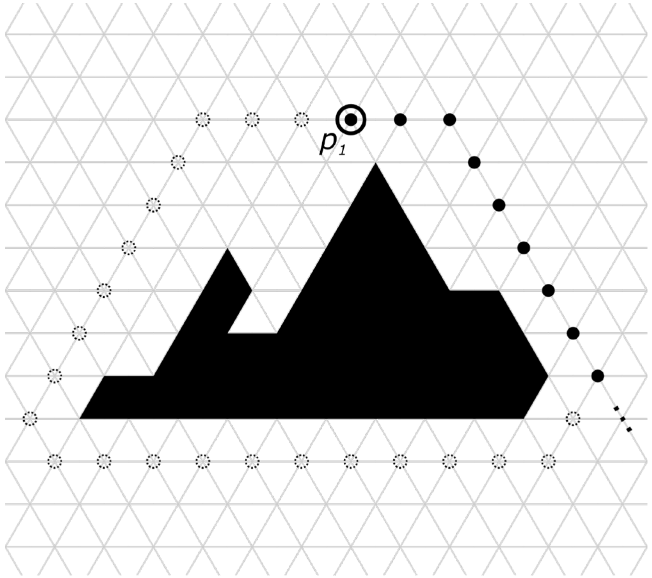
1. Moving the root of the particle line to some position in *C(O)* by bending + forwarding.
2. Bending the line around the object, occupying the rest of *C(O)*.



Phase II: Constructing the Convex Hull

Phase II begins from the straight line of particles obtained in Phase I. This phase is also divided into two subphases (but these don't alternate):

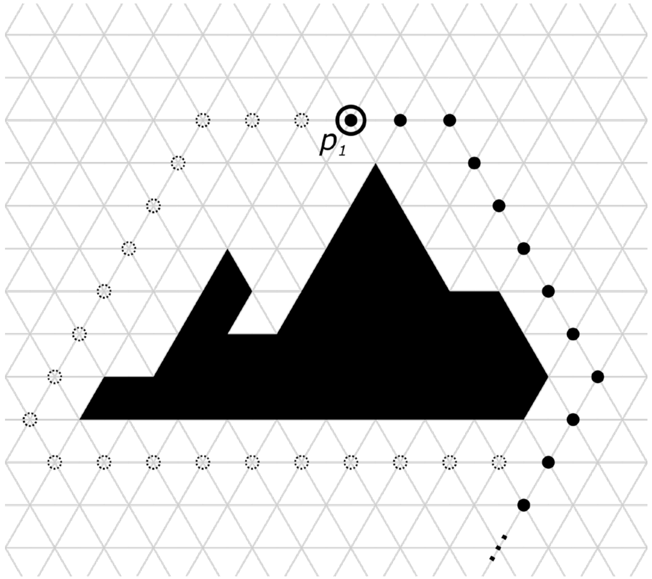
1. Moving the root of the particle line to some position in *C(O)* by bending + forwarding.
2. Bending the line around the object, occupying the rest of *C(O)*.



Phase II: Constructing the Convex Hull

Phase II begins from the straight line of particles obtained in Phase I. This phase is also divided into two subphases (but these don't alternate):

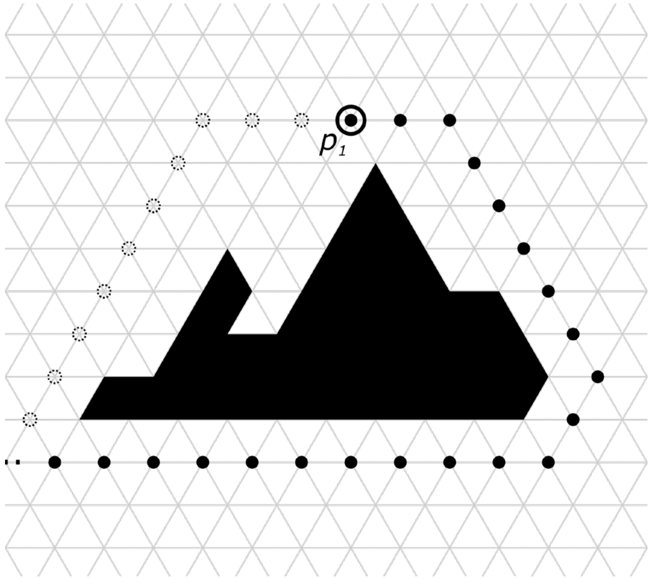
1. Moving the root of the particle line to some position in *C(O)* by bending + forwarding.
2. Bending the line around the object, occupying the rest of *C(O)*.



Phase II: Constructing the Convex Hull

Phase II begins from the straight line of particles obtained in Phase I. This phase is also divided into two subphases (but these don't alternate):

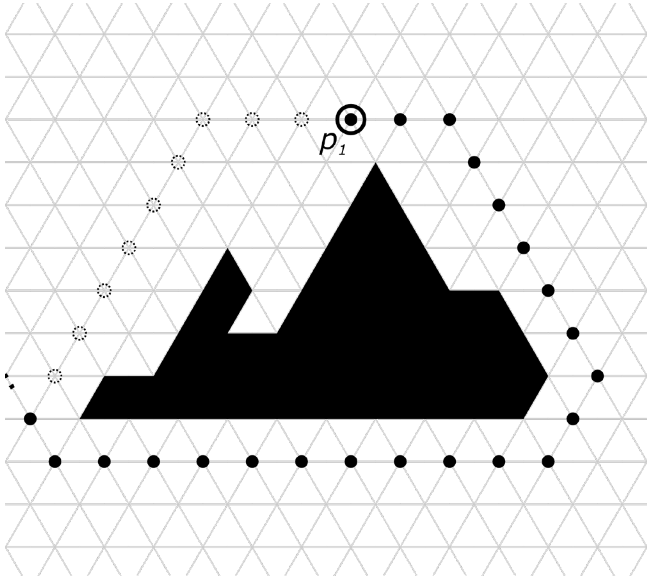
1. Moving the root of the particle line to some position in *C(O)* by bending + forwarding.
2. Bending the line around the object, occupying the rest of *C(O)*.



Phase II: Constructing the Convex Hull

Phase II begins from the straight line of particles obtained in Phase I. This phase is also divided into two subphases (but these don't alternate):

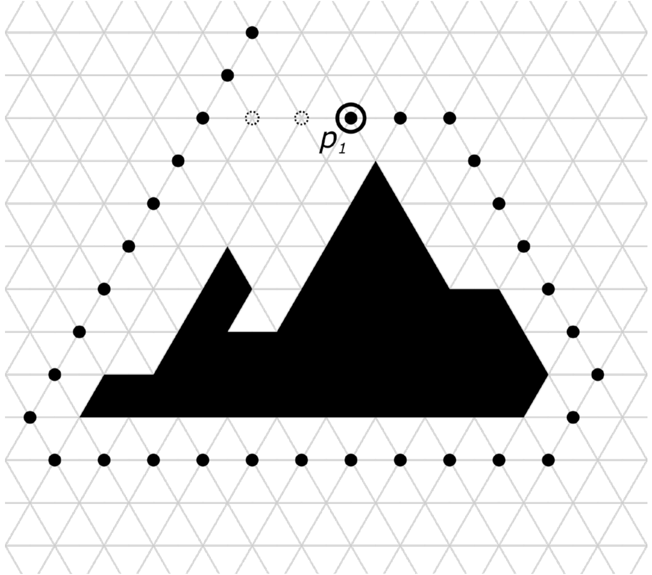
1. Moving the root of the particle line to some position in *C(O)* by bending + forwarding.
2. Bending the line around the object, occupying the rest of *C(O)*.



Phase II: Constructing the Convex Hull

Phase II begins from the straight line of particles obtained in Phase I. This phase is also divided into two subphases (but these don't alternate):

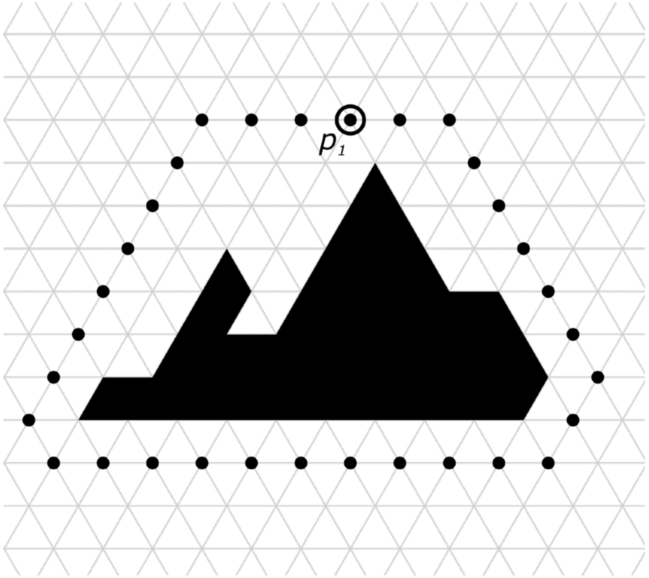
1. Moving the root of the particle line to some position in *C(O)* by bending + forwarding.
2. Bending the line around the object, occupying the rest of *C(O)*.



Phase II: Constructing the Convex Hull

Phase II begins from the straight line of particles obtained in Phase I. This phase is also divided into two subphases (but these don't alternate):

1. Moving the root of the particle line to some position in *C(O)* by bending + forwarding.
2. Bending the line around the object, occupying the rest of *C(O)*.



Preliminary Worst-Case Runtime Analysis

Let $n = |P|$ and m be the area occupied by O.

We measure runtime in *asynchronous rounds*.

Phase I: $\mathbf{O}(n + m)$ rounds. **?**

- Spanning forest primitive: $\mathbf{O}(n)$ rounds. ✓
- Wall following subphase: $\mathbf{O}(m)$ rounds. ✓
- Line probing subphase: $\mathbf{O}(m)$ rounds. **?**

Phase II: $\mathbf{O}(n)$ rounds. **?**

- Each line bending: $\mathbf{O}(n)$ rounds. **?**
- Move the root to the hull: ≤ 6 line bends. ✓
- Wrap the rest of the line: 6 line bends. ✓

All together: $\mathbf{O}(n + m)$ rounds…?

Future Work

- For convex hull formation (work-in-progress):
  - Formalize the ideas outlined here into a fully developed distributed algorithm.
  - Theoretical results: work out the details of correctness and runtime proofs.
- For Self-Organizing Particle Systems in general:
  - Pushing towards applications: bridging/filling gaps, etc.
  - Investigate more fault tolerant algorithms.
  - Generalize the existing model and algorithms to 3-dimensional space, if possible.

Collaborators

Arizona State University


Andréa W. Richa


Joshua J. Daymude

UNIVERSITÄT PADERBORN


Christian Scheideler


Robert Gmyr


Thim Strothmann

Thank you!

sops.engineering.asu.edu