Programmable self-assembly in a thousand-robot swarm

Michael Rubenstein, Alejandro Cornejo, Radhika Nagpal.

By- Swapna Joshi
1st year Ph.D Computing Culture and Society.

Authors

Michael Rubenstein
Assistant Professor Department of Electrical Engineering and Computer Science Department of Mechanical Engineering Northwestern University

Alejandro Cornejo
Tech Start Up
PhD in C.S - MIT

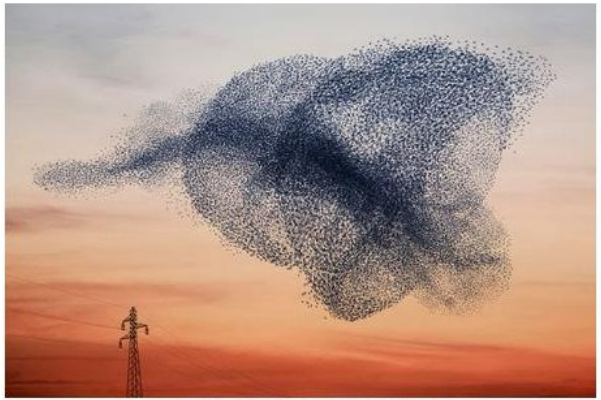
Radhika Nagpal
Fred Kavli Prof. of Computer Sc.
School of Eng and Applied Sc.
Wyss Institute for Biologically Inspired Engineering
Harvard University

COMPUTATIONAL BEAUTY- In Nature

Some social systems in Nature can present an intelligent collective behavior although they are composed by simple individuals.

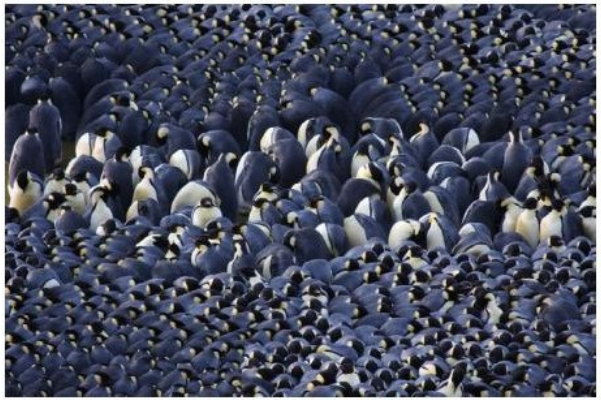
The intelligent solutions to problems naturally emerge from the self-organization and communication of these individuals.

These systems provide important techniques that can be used in the development of artificial intelligent systems.



Bird flock (@ Alain Delorme)

Fish school (@ Francis Pérez)



Penguins (@ Robyn Mundy)

Wildebeest (@ Burrard-Lucas.com)

COLLECTIVE INTELLIGENCE- learning from nature

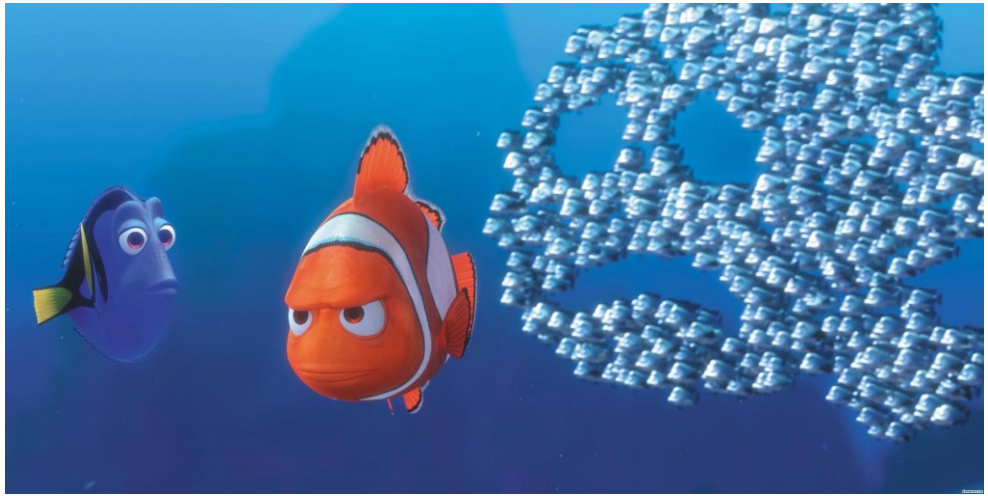
Rules of Engagement:

- One vs Collective
- Any one individual could have a really limited view of what is going on
- Intelligence is not limited to individual, but property of the group
- No leadership
- Sheer oneness, emerging from interactions or local rules of engagement
- Single entity, single mind making collective decisions
- Local and simple interactions
- New properties emerge , such as phase transition, pattern formation, group movement

COLLECTIVE INTELLIGENCE – Inspiration

“ A lot of my research is built around this idea that if you have a collective of individuals and they all have simple local rules, what you cant do is design the rules bottom up. Because you are just going to be stuck trying to see every variation of what goes on, and if you look what an individual is doing, it is not well connected with the global one. So is there a way to go inverse? Is it possible to write, like, a compiler where I say, Well, what I want the group to achieve is this. And then, the computer sort of figures out what it is that all of the individuals should do.” – Nagpal

IEEE History of Robotics Interviews
Conducted by Prof.Selma Sabanovic

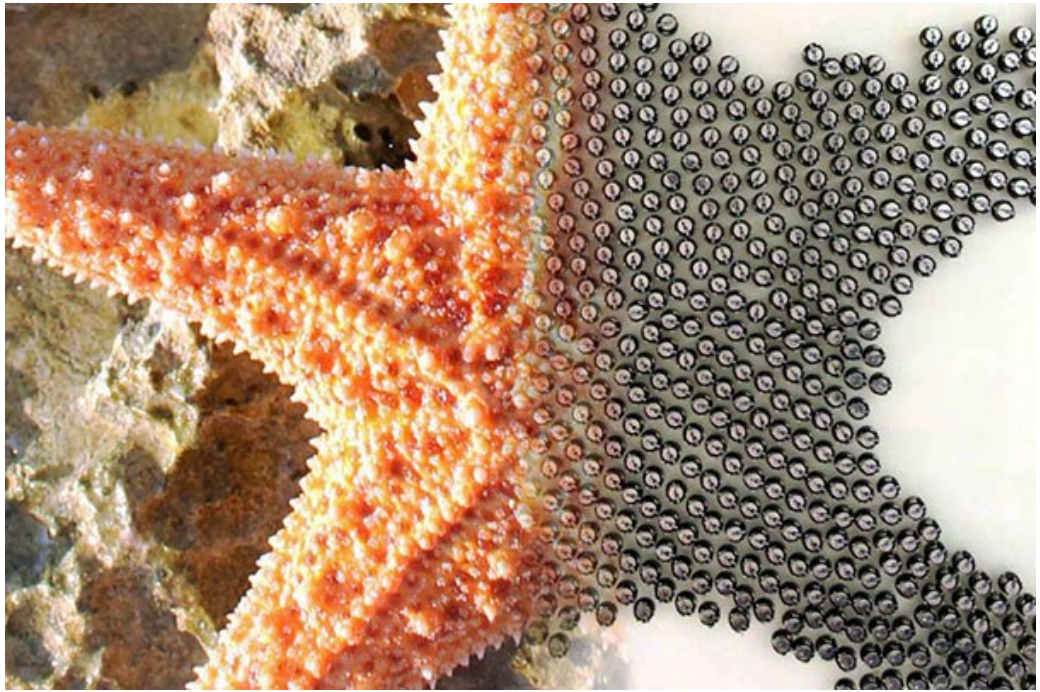


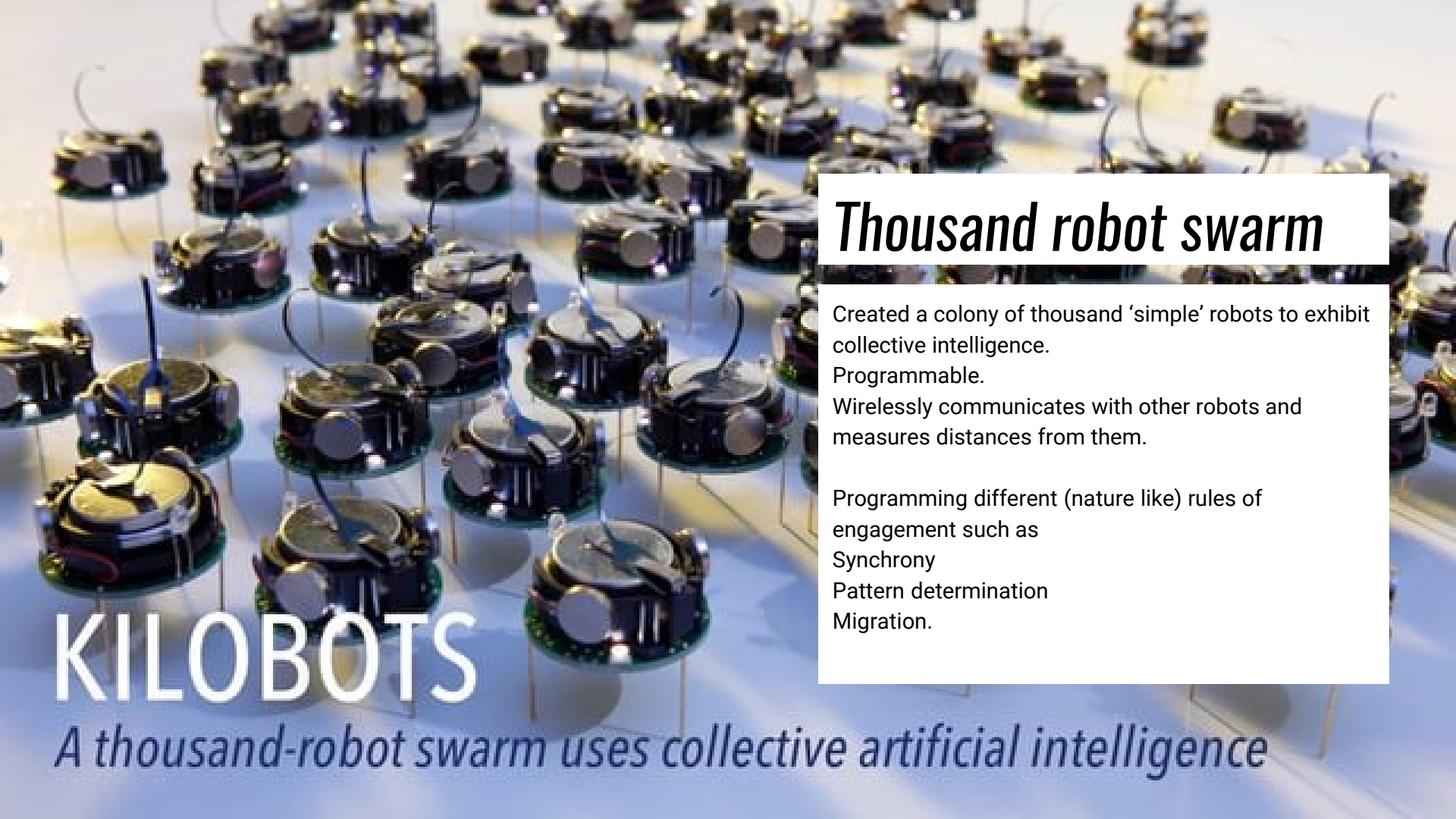
COLLECTIVE INTELLIGENCE– Intelligent Machines

A key theme in AI is to create computational intelligence the way we see it in nature.
Creating these rules of engagement and local interactions for computing collective intelligence.

Two main questions:

- How do we take a global goal and translate it into local interactions between identical agents?
- How do we engineer robust, predictable behavior from large number of unreliable agents?





Thousand robot swarm

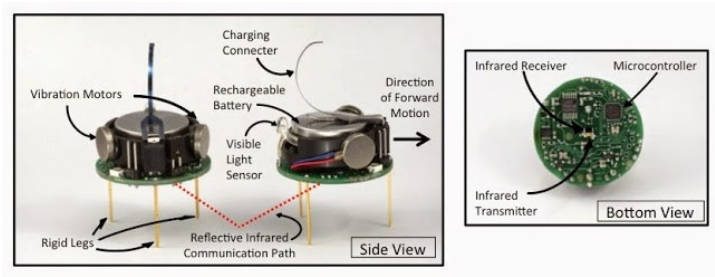
Created a colony of thousand ‘simple’ robots to exhibit collective intelligence.
Programmable.
Wirelessly communicates with other robots and measures distances from them.

Programming different (nature like) rules of engagement such as
Synchrony
Pattern determination
Migration.

Algorithms

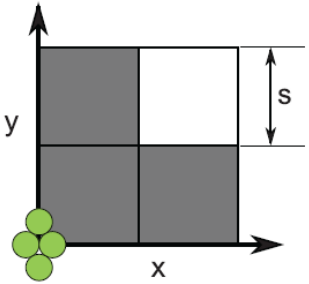
Design

Creation of complicated self assemblies by combining different rules of engagement.



User-specified shape

The desired shape is given to all robots in the form of a binary bitmap. Four pre-localized seed robots (green) define the origin and orientation of the coordinate system.



The desired shape is aligned with the coordinate system and scaled by the input parameter 's'.

1. Robots have the ability to approximate holonomic motion (move straight, turn in place).
2. Robots can communicate with neighboring robots within a fixed radius.
3. Robots can measure distance to communicating neighbors within that radius.
4. Robots have basic computation capabilities and internal memory. All robots, except the four seed robots, are given an identical program, which includes the self-assembly algorithm and a description of the desired shape.

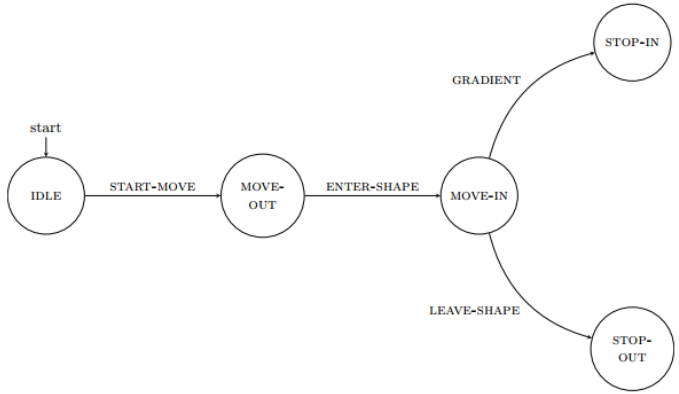


Figure S7: Shape Formation State Machine Representation

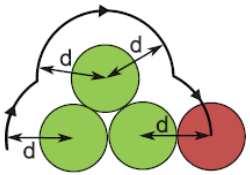
Algorithms

Edge-following

In edge-following, a moving robot attempts to follow the boundary of a set of stationary robots in a clockwise direction.

Edge-following

A robot (red) moves by maintaining a fixed distance 'd' to the center of the closest stationary robot (green).



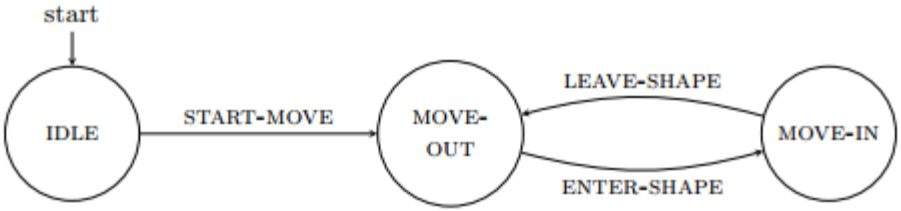


Figure S9: Perpetual Edge-Following Algorithm

Algorithm 1 Edge follow nearest neighbor at DESIRED_DISTANCE

```
1:  prev ← DISTANCE_MAX// prev will store previously measured distance to nearest neighbor
2:  loop
3:      current ← DISTANCE_MAX // current will store current distance to nearest neighbor
4:      for all neighbors n do
5:          if measured_distance(n) < current then
6:              current ← measured_distance(n)
7:      if current < DESIRED_DISTANCE then // desired edge-following distance
8:          if prev < current then // getting closer to desired distance
9:              move straight forward
10:         else
11:             move forward and counterclockwise
12:     else
13:         if prev > current then // getting closer to desired distance
14:             move straight forward
15:         else
16:             move forward and clockwise
17:     prev ← current
```

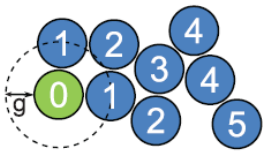
Algorithms

Gradient Formation

Individual robots can measure distances between each other; the purpose of gradient formation is to create a long-range sense of distance across a swarm.

Gradient formation

Each robot sets its gradient value to 1 + the minimum value of all neighbors closer than distance 'g'. The source robot (green) maintains a gradient value of 0.



Algorithm 2 Gradient formation. G represents the gradient-distance and GRADIENT_MAX is infinity.

```
1:  loop
2:    if gradient_seed = TRUE then // check if robot is designated as gradient source
3:      gradient_value(self)← 0
4:    else
5:      gradient_value(self)←GRADIENT_MAX
6:      for all neighbors n do
7:        if measured_distance(n) <G then // only consider neighbors closer than G
8:          if gradient_value(n) <gradient_value(self) then
9:            gradient_value(self)← gradient_value(n)
10:     gradient_value(self)← gradient_value(self)+1
11:     transmit gradient_value(self)
```

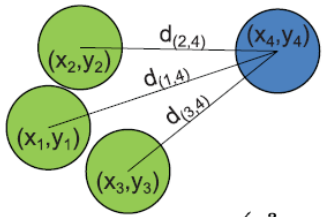
Algorithms

Localization

The self-assembly algorithm relies on robots ability to localize in a coordinate system that is generated and shared by robots inside the desired shape.

Localization

A robot (blue) determines its position in the coordinate system by communicating with already localized robots (green).



$$(x_4, y_4) = \min_{x_4, y_4} \left(\sum_{i=1}^{3} |d_{(i,4)} - \alpha_i| \right)$$

where $\alpha_i = \sqrt{(x_i - x_4)^2 + (y_i - y_4)^2}$

Algorithm 3 Localization using greedy search to minimize trilateration equation

```
1:  if not a seed robot then
2:      position(self)← (0, 0)
3:  loop
4:      nlist← [] // clear list of localized and stationary neighbors
5:      for all neighbors i do
6:          if i is localized and stationary then
7:              nlist.append(i)
8:      if nlist contains 3 or more non_collinear robots then
9:          for all l in nlist do
10:             c ← distance_between(position(self),position(l))
                // calculated distance based on position in coordinates, as opposed to measured distance based
                on signal strength
11:             v ← (position(self)−position(l))/c
                // unit vector pointing from position(l) towards position(self)
12:             n ← position(l)+measured_distance(l)∗v          // compute new position
13:             position(self)←position(self)−(position(self)−n)/4
                // move 1/4 of the way from old calculated position towards new one
```

Algorithm

State Diagram

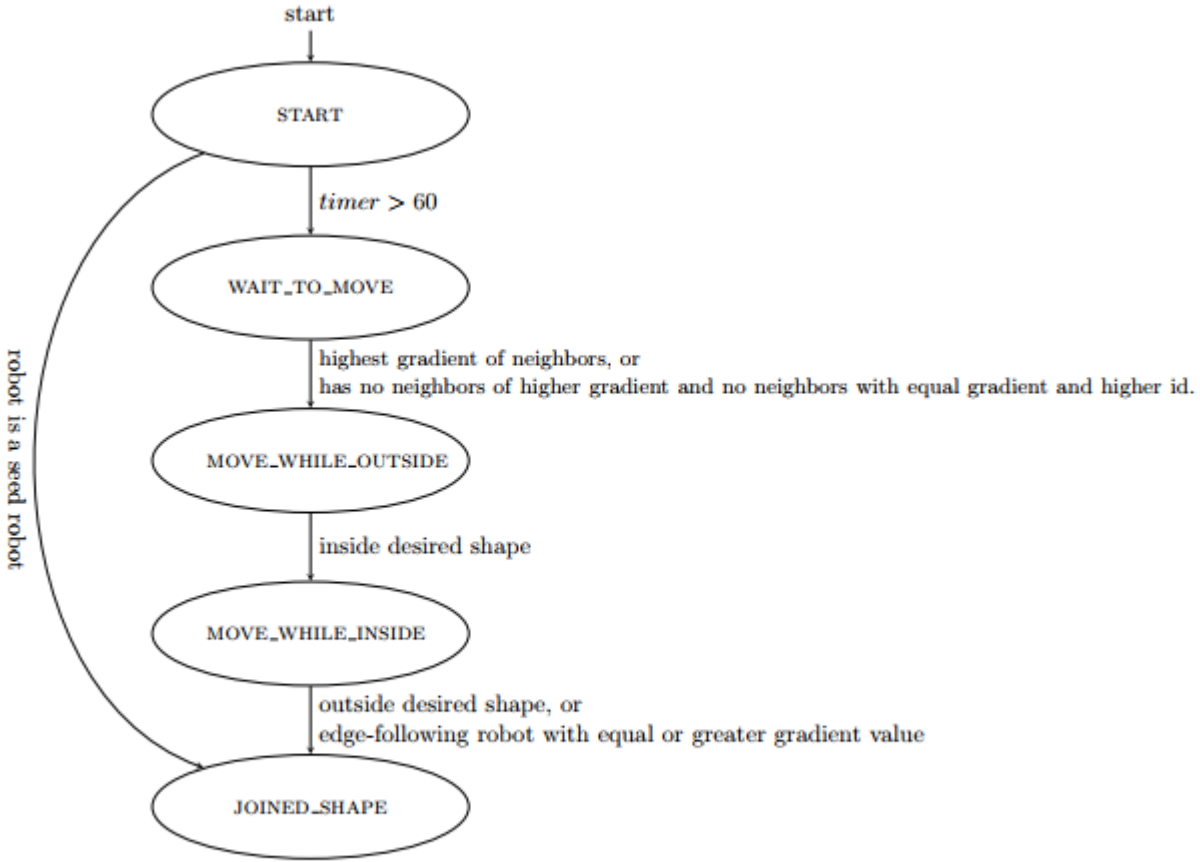


Figure S1: A state diagram of the self-assembly algorithm.

Algorithms

Self Assembly Algorithm

Any single robot is talking to a small number of robots nearby it, using its motion rule to move around the half built structure to decide a place to fit in based on its pattern rules.

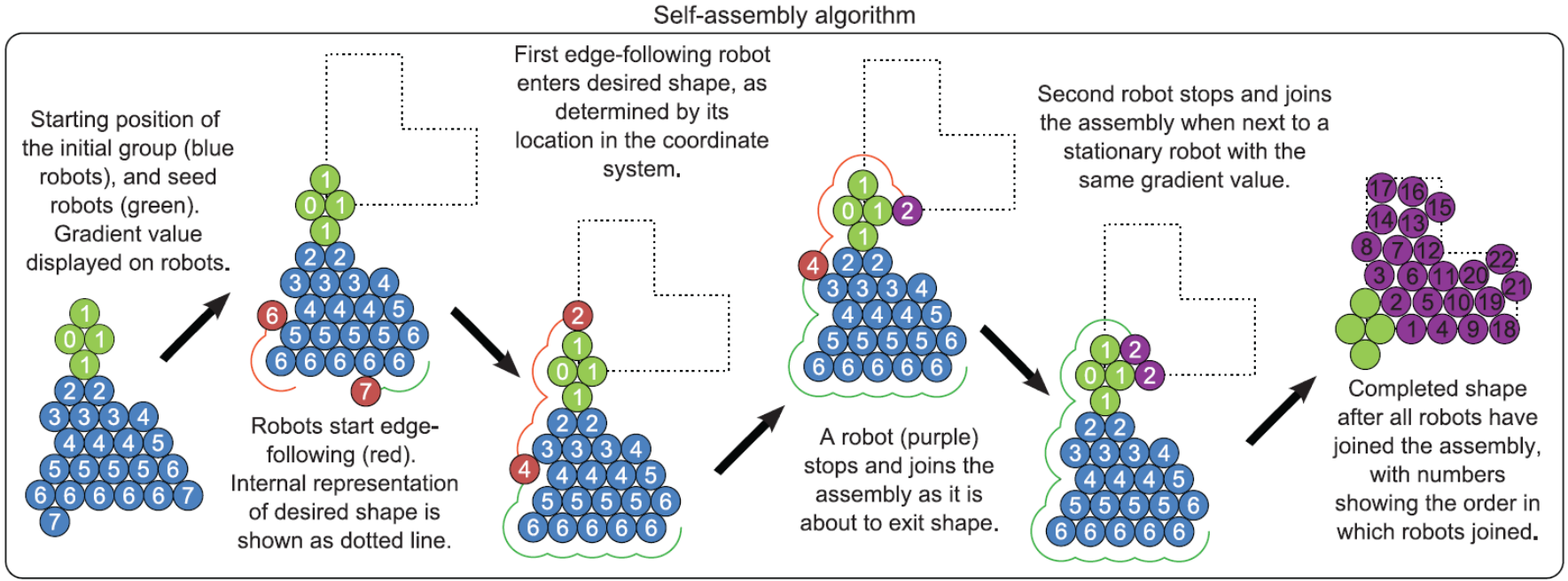
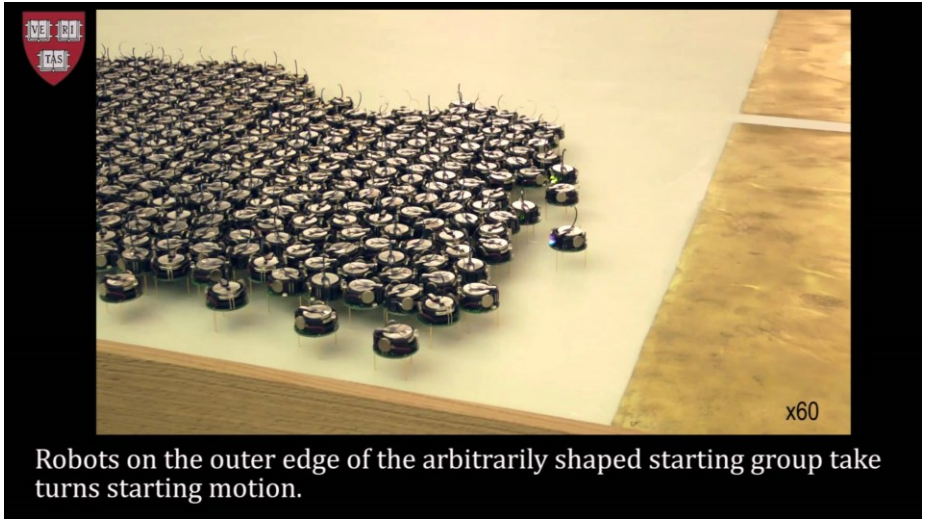


Fig. 2. Collective self-assembly algorithm. Top left: A user-specified shape is given to robots in the form of a picture. Top right: The algorithm relies on three primitive collective behaviors: edge-following, gradient formation, and localization. Bottom: The self-assembly process by which a group of robots forms the user-defined shape.

Algorithms

Shape self Assembly

Creation of complicated self assemblies by combining different rules of engagement.



Robots on the outer edge of the arbitrarily shaped starting group take turns starting motion.

Thousand Robots Swarm

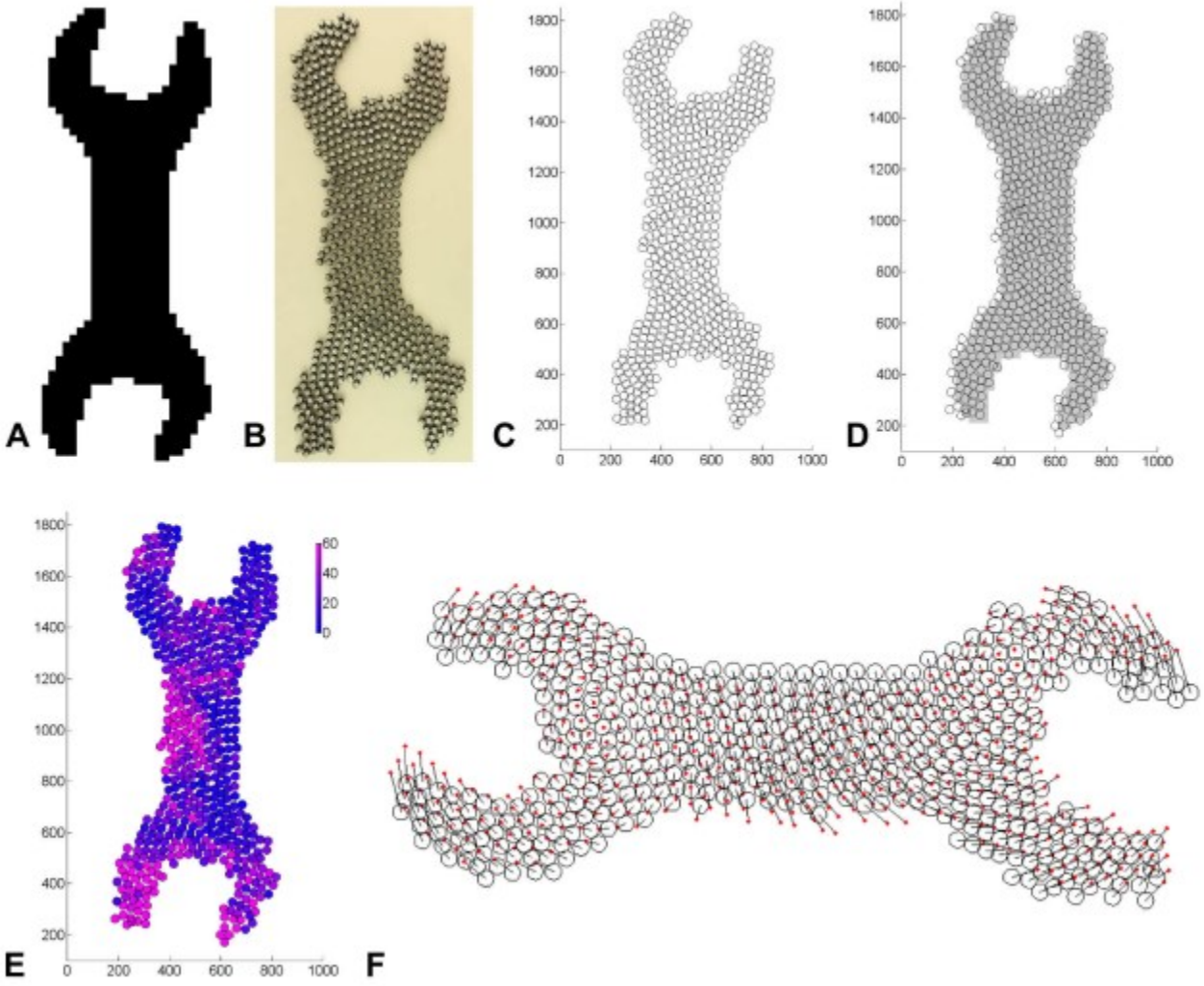
Algorithm 5 Shape self-assembly

```
1:  motion: stop
2:  state← start
3:  timer← 0
4:  loop
5:    if state=start then // wait for fixed amount of time so all robots are on
6:      if robot is a seed robot then
7:        state← joined_shape
8:      else
9:        begin gradient formation (Alg. 2)
10:       begin localization (Alg. 3)
11:       timer←timer+1
12:       if timer > startup_time then
13:         state← wait_to_move
14:   else if state=wait_to_move then // decide when to start moving
15:     if no moving neighbor seen then
16:       h ← 0        // find highest gradient value among neighbors
17:       for all neighbors n do
18:         if h <gradient_value(n) then
19:           h ←gradient_value(n)
20:       if gradient_value(self)> h then
21:         state← move_while_outside
22:       else if gradient_value(self) = h then
23:         if ID(self) > ID of all neighbors with equal gradient_value then
24:           state← move_while_outside
25:   else if state=move_while_outside then // edge-follow while outside desired shape
26:     if position(self) is inside desired shape then
27:       state← move_while_inside
28:     if distance to front edge-following robot > yield_distance then
29:       motion: edge-follow (Alg. 1)
30:     else // yield to edge-following robot ahead
31:       motion: stop
32:   else if state=move_while_inside then // edge-follow while inside desired shape
33:     if position(self) is outside desired shape then
34:       state← joined_shape
35:     if gradient_value(self) ≤ gradient_value(closest neighbor) then
36:       state← joined_shape
37:     if distance to front edge-following robot > yield_distance then
38:       motion: edge-follow (Alg. 1)
39:     else // yield to edge-following robot ahead
40:       motion: stop
41:   else if state=joined_shape then // become stationary as part of shape
42:     motion: stop
43:     stop localization
44:     stop gradient_value update
```

Algorithms

Accuracy

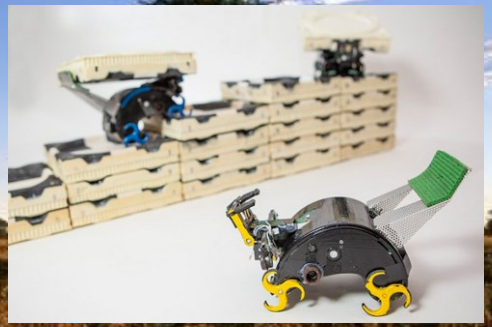
Even though no robot is doing anything accurately, the rules are such that the collective achieves its goal, working like a single entity rather than individuals.

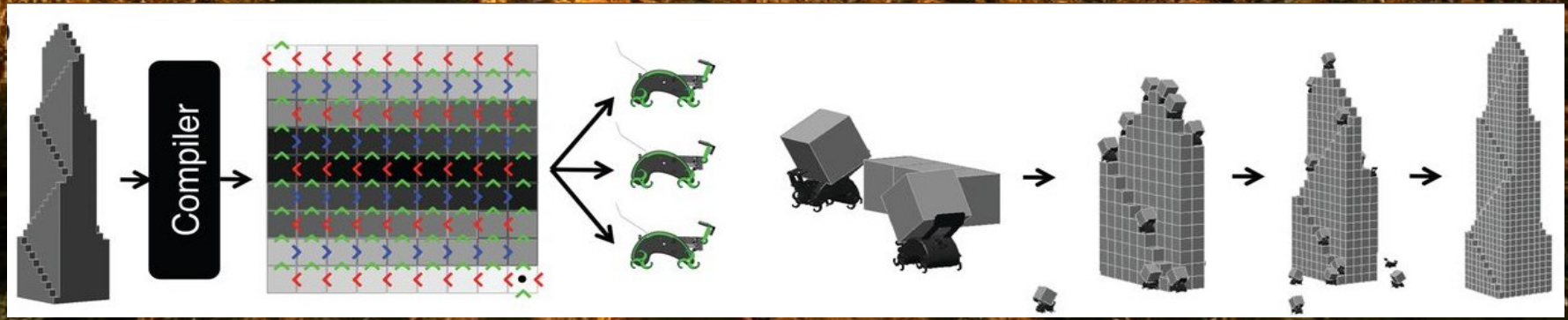


Collective intelligence

Other projects

3Dimensions
Inspirations from Social insects that use pattern rules that help them determine what to build.





Collective Intelligence

Application conceptualizations

Many different applications are possible by creating such artificial collective intelligence or computing them. They are mathematical and conceptual tools to create our own versions of collective power.

Questions

What are some of the ethical considerations to computing collective intelligence or articficially creating it? link

In what ways is human collective behavior different from that of other organisms?

What about rules that apply to our own human collective? Should we ever engineer the human collective? What could errors mean in the context of human collective?