Ant Geometers

Sean Luke, Katherine Russell, and Bryan Hoyle

George Mason University
sean@cs.gmu.edu

Abstract

Just how much can a pheromone-enabled swarm do? Motivated by robotic construction, we set out to show that a swarm of computationally simple ants, communicating only via pheromones, can in fact perform classic compass-straightedge geometry, and thus can make many shapes and perform many nontrivial geometric tasks. The ants do not need specially-designed stigmergic building materials, a prepared environment, local or global direct communication facilities (such as radio or line-of-sight signaling), or any localization beyond initial starting points for drawing. We describe the proof of concept in replicable detail. We then note that its accuracy and efficiency can be greatly improved through augmentation with a simple embeddable broadcast mechanism.

Introduction

One of the biggest difficulties in swarm robotics, and in swarm agent simulation in general, lies in how to communicate and coordinate. Due to their large numbers, swarm agents often cannot communicate through a common broadcast medium such as radio, both because they would overwhelm the medium, and because it would require every agent to receive and deal with messages from all N other agents. Instead artificial swarm schemes often use either local communication or *indirect* communication, whereby agents leave messages for one another — virtual breadcrumbs, if you will.

The biggest source of inspiration for indirect communication in robotics is surely the use of pheromones by ants, termites, etc. to coordinate behaviors. These insects are known to use pheromones for many tasks: but most swarm robotics and swarm simulation literature has focused on their most famous use, namely establishing foraging trails.

In prior work we have demonstrated trail optimization, adaptation to environmental changes, and even multi-waypoint, self-intersecting tours using only pheromones and swarms of very simple agents (Panait and Luke, 2004). Our later work extended the pheromone model to swarm foraging robots which store and read pheromone information in intelligent breadcrumbs (in the form of wireless sensor motes), and then deploy, move, and retrieve these devices from the environment (Hrolenok et al., 2010; Russell et al., 2015).

The use of pheromones to build foraging trails is straightforward and well studied. We are instead interested in showing that pheromones can be used for something much more ambitious. Our research area is collective building construction, and among the first tasks in construction is the laying out of survey lines to define the form of the object being built. As such, we have chosen to demonstrate, as an elaborate proof of concept, that swarms of very simple agents, communicating only via pheromones, can achieve all the operations necessary to perform collective **compass-straightedge construction** (or "classical construction") from geometry, and thus can build many nontrivial geometric shapes.

This is not easy: some of the basic operations are challenging to achieve and, while we show that such operations are possible, they can be costly. We will compare against agents that differ only in their communications medium (broadcast beacons), but nevertheless can do the task much more rapidly.

In this paper we first discuss existing literature in pheromones, swarm robotics, and collaborative construction. We then explain classic compass-straightedge geometric construction and its background. We then detail the pheromone and swarm agent model being used, and describe the basic procedures necessary to do compass-straightedge construction. Finally, we compare this approach against similar agents using broadcast beacons instead of pheromones.

Previous Work

Swarm robotics and swarm agent research is naturally inspired by social insects (Brambilla et al., 2013) and stigmergic approaches to collective behavior Dorigo et al. (2000). Swarms are highly parallel, can be built with simple agents, and are robust in the face of noise, agent failure, and dynamic environments (Bonabeau, 1996). Swarms are commonly used in tasks such as exploration and foraging (Wodrich and Bilchev, 1997; Panait and Luke, 2004; Russell et al., 2015; Prabhakar et al., 2012), but one recent application has been in collective construction (Ardiny et al., 2015). Swarms are typically limited to indirect, stigmergic, and local communication, which leads to one of the three following implementation trends:

First: agents may use inert, local, environmental features, which can be sensed but do not directly communicate with the agents or with each other. Landmarks or the presence of other agents are common features which can be used to clear ground for site preparation (Parker and Zhang, 2006) and build circles around given locations (Pitonakova and Bullock, 2013). Agents using these techniques do not need accurate localization and can use a variety of building materials to perform their tasks. However, the agents cannot easily do planning or coordination as they neither know where they are nor what has been accomplished so far.

Second: agents may exist in or create "smart" environments which can be used to localize the agents, such as writable blocks (Allwright et al., 2014; Werfel and Nagpal, 2008; Sugawara and Doi, 2014) or countable building materials (Werfel et al., 2011). This allows grid-world models to be directly implemented with robots and has produced swarms capable of making 3D user-defined shapes. The use of such environments allows agents to be fully localized relative to the structure they are building, but they must use highly specialized building materials or contrived environments.

Third: agents may lay temporary stigmergic markers in the environment, such as breadcrumbs or pheromones (Deneubourg et al., 1990; Russell, 1999; Panait and Luke, 2004; Chibaya and Bangay, 2007). This technique has produced behaviors such as circle building (Pitonakova and Bullock, 2013), exploration (Wodrich and Bilchev, 1997), and wall building (Stewart and Russell, 2006). One difficulty with the method lies in the medium in which these stigmergic markers are placed. In prior work we have attempted to address this with portable local beacons (Russell et al., 2015) but other methods, such as RFID tags (Mamei and Zambonelli, 2007; Ziparo et al., 2007), lights (Stewart and Russell, 2006), and chemical dispersion (Kowadlo and Russell, 2008) have also been tried.

Our work in this paper is most comparable to that of "smart" environments, such as in Werfel et al. (2011), as it allows the collective construction of a very large number of possible structures: but instead of using specially-made stigmergic construction materials, we explore whether such tasks may be performed solely through a general-purpose indirect communication method such as pheromones.

Compass-Straightedge Construction

The *compass-straightedge* technique has existed since the ancient Greeks and has a long history of constructive proofs to build complex shapes and do many nontrivial geometric tasks. Traditionally the only two tools permitted are a collapsing compass and an arbitrarily long straightedge. The compass loses its angle as soon as it is lifted off the drawing surface, and so one cannot preserve distance by raising the compass and moving it somewhere else. However, this is an artificial limitation, as there is a way to transfer distance between two points using a finite number of axiomatic steps (Sarhangi, 2007). It has also been shown that the straightedge is not required at all, only a compass, (Mohr, 1672; Mascheroni, 1797) but the resulting proofs can be much more complicated.

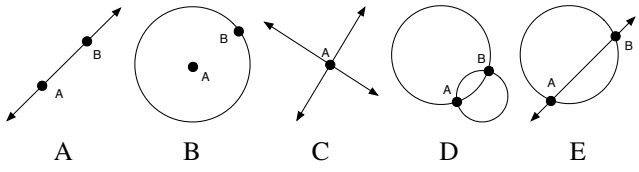
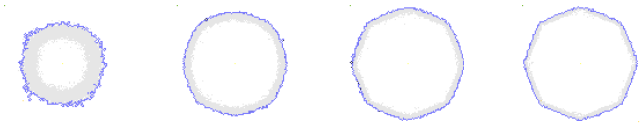


Figure 1: The five basic compass-straightedge geometry procedures. (A) Drawing a line between two points. (B) Drawing a circle centered at one point and passing through another. (C) Identifying the point at the intersection of two lines. (D) Identifying and distinguishing the two points at the intersection of two circles (one point if they are tangent). (E) Identifying and distinguishing the two points at the intersection of a circle and a line (one point if they are tangent).

Figure 1 shows the five basic abstract procedures sufficient to do all compass-straightedge construction: drawing a line through two points, drawing a circle centered at one point and passing through another, identifying the intersection of two lines, identifying the intersections of a line and a circle, and identifying the intersections of two circles. Note that for the last two procedures, not only must one identify the points, but one must also *uniquely* identify and distinguish them from one another. This problem does not arise for human proofs, which are visual: but ants have only local information and so must include ways to distinguish the points.

Composing these simple techniques, one can build much more complicated constructions. One simple example for constructing an equilateral triangle is as follows: start with a line segment representing the desired base; use a compass to construct two circles centered at the endpoints with radius equal to the segment length; and, finally, draw two more line segments connecting the desired intersection point to the two endpoints of the base. Other basic things which can be constructed include: bisecting arbitrary angles with a line; constructing a square with twice the area of another square; constructing a circle tangent to another circle; trisecting an arbitrary line segment; and building any regular polygon whose number of edges is equal to some power of 2 times the product of zero or more primes of the form $2^{2^n}+1$, for some integer n; and many, many more. Overall the first ten constructible polygons have 3, 4, 5, 6, 8, 10, 12, 15, 16, and, thanks to Gauss, 17 sides (Gauss, 1801).

There are many things that *cannot* be constructed: any regular polygon not of the form above (such as the heptagon or the nonagon), trisections of arbitrary angles, squares with equal area to arbitrary circles (the legendary *squaring the circle*), and many other classes of shapes. Several extensions have been proposed: for example, the ability to make marks on the straightedge permits angle trisection and the construction of additional regular polygons (Gleason, 1988).



A: 10% B: 1% C: 0.1% D: 0.01%

Figure 2: Effect of Evaporation on Circle Development. Too high evaporation (A) produces incorrectly small and overly noisy circles, but too low evaporation (D) produces octagons.

The Ant Model

Out ants' world is a non-toroidal, bounded 200×200 square grid environment holding 1000 ants. Any number of ants may share the same grid cell, and can move horizontally, vertically, or diagonally. Each grid cell can also hold multiple kinds of *pheromones*. The ants may move about in the environment, may read pheromone values in their local (9-cell) neighborhood, and may write or update pheromone values only to their current cell. As shown later, a grid world is not critical to the model, but was chosen for simplicity.

Pheromones are used by the ants for various tasks: to mark points of interest, to establish gradients to and from those points of interest, to build up estimates of shapes, and to make final line drawings in the environment. Each pheromone has a *pheromone type* and a current numerical *value* ≥ 0 which by default automatically reduces (*evaporates*) at a rate of 0.5% per timestep. A pheromone in a cell can also be set to be *non-evaporating*. Non-evaporating pheromones are used to draw lines and circles and to mark points of interest, which in turn serve as permanent maxima in a pheromone gradient. We call cells holding non-evaporating pheromones *sites*.

Because we ultimately will migrate this model to robots, the pheromones in the model do not diffuse on their own, but rather pheromone information can only be spread from one cell to another by an ant. This is because, while diffusion is used in biological pheromone models, it is not easy to employ with physical robots, as it requires chemical sensors and dispersion methods (Kowadlo and Russell, 2008), or cells or breadcrumbs which communicate with one another.

Lacking diffusion, evaporation becomes very useful for a grid-world model as it causes the ants to naturally build gradients with more circular and less octagonal cross-sections. This results in straight paths at any angle, not just in the eight compass directions, and more circular circles (see for example Figure 2). Additionally, evaporation can be seen as a potential benefit in more interesting dynamic environments where old information should be treated skeptically. The disadvantage of evaporation is that it is a major source of noise: agents cannot rely on pheromone gradients being consistent at infrequently visited locations as neighboring cells may have been allowed to evaporate while the current cell has just recently been "topped off". This significantly complicates our task and makes our procedures less efficient.

The Ants Each ant is a simple machine which iteratively *updates* pheromone values in its cell, then either follows a *procedure* to perform some task, or (with 0.1 probability) *moves randomly*. Random movement encourages exploration and nondeterminism: some procedures may temporarily disable it so as to reduce noise.

An ant can move in any of the eight compass directions. If an ant moves horizontally or vertically, it must wait one timestep before it may continue. If an ant moves diagonally, it must wait 1.5 steps (a discrete approximation of $\sqrt{2}$, though neither has any real impact on the results over just using 1.0).

An ant can also sense pheromones in its grid cell or any of the eight neighboring cells, and can determine if those pheromones are set to evaporate or not. An ant also knows where it was last timestep, and can perceive relative orientation ("to the left of me", etc.). An ant can set a temporary *timer* to know roughly how long it has been doing a task. Finally, an ant is capable of storing a single pheromone value in its sole *register* for later retrieval.

Updating Pheromones Every timestep, the ant first updates all the pheromones at its cell location. If a pheromone P at the ant's location i is marked as non-evaporating, the ant leaves it alone. Otherwise, it is updated as:

$$P_i \leftarrow \max_{j \in \text{Neighbors}(i)} \begin{cases} \alpha^{\sqrt{2}} P_j & \text{if } j \text{ is a diagonal neighbor of } i \\ \alpha P_j & \text{otherwise} \end{cases}$$

We set α to 0.88 based on experiment. As the ants wander about, this equation effectively builds a pheromone gradient away the "tent-pole" locations in the grid (where a non-evaporating value for P has been set). Note that even though $\alpha^{\sqrt{2}}$ is used to (properly) cut down diagonal neighboring cells, this is not sufficient to create a true circular gradient in a grid world.

The Procedures

As this paper is a proof of concept of a challenging task, and the procedures below are nontrivial, we describe them in detail for replicability, and beg the reader's forgiveness.

Though we describe the ant's procedures below in algorithmic pseudocode, we in fact implement each of them as a finite-state automaton (a DFA) which iterates through its process one step at a time as it is pulsed. Each state in the automaton is associated with some behavior to iteratively perform, which may be some simple action (like laying a pheromone) or a call to a lower-level procedure (such as wall following). Such a recursive DFA is known as a *Hierarchical Finite-State Automaton* (HFA). Any procedure can signal *done*, which informs its calling procedure that it has finished.

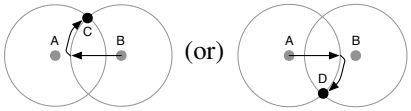
The procedures make heavy use of building gradients from various points in order to establish loci. They take two kinds of arguments: *pheromones*, which are capitalized (like *PointA*), and simple numerical *values*, which are lowercase (like *direction* or *m*). A pheromone typically establishes

a gradient leading to a maximal spot in the environment. For brevity, we often use the pheromone name (like *PointA*) to also refer to the location of its maximum (typically a *site*). There is always a single starting location for the ants: *Home*.

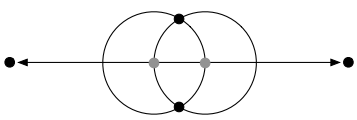
Some procedures assume a world border, however in an environment with no border it could just as well be replaced with following along the locus of points for which the *Home* gradient is equal to some very small value. Only cells within the world border are considered valid places to move and read pheromones from.

We first describe various basic procedures, then the five geometric procedures. Here is a summary of the five:

- **Draw a Circle** Draw a line segment from the center to the edge point and use it to determine the radius (in terms of pheromone gradient from the center). Build a gradient from the center. Identify points whose gradient value matches the radius. Trace a line through those points.
- **Mark Line/Line Intersection** This is simple: fan out until the intersection is discovered, then mark it.
- **Mark Circle/Circle Intersections** Identify the intersections *and* determine which is which. To do this, each ant randomly chooses to go to one or the other of the two circle centers. It then heads towards the other center until it finds the circle edge of its initial center, then follows clockwise along the circle until it reaches an intersection. The intersection is uniquely labeled according to which circle was followed when it was discovered. The process is illustrated below:



- **Mark Line/Circle Intersections** Again, the trick is to distinguish the intersections from each other. To do this, the ants go to a certain extreme point on the line outside the circle, then follow down the line, and as they find intersections, mark them in the order they were found.
- **Draw an Extended ("Infinite") Line** This is tricky. Noise and the restriction to local information make the obvious approaches impossible for drawing an extended line, such going down the gradient away the circle centers, or walking straight using some notion of momentum. Instead we set up a perpendicular bisector. The ants draw a circle centered at the first point and passing through the second point, then a circle centered at the second point, passing through the first. They identify the two circle/circle intersections, then draw the extended line that is the the perpendicular bisector between the two points. The various elements are shown below:



Basic Procedures

Backup( ) Back up one step, to the ant's previous location. Further backups are not possible (the ant has no history).

GoUp(*P*) Iteratively, in the current nine-cell neighborhood around the ant, move to the cell where the value of pheromone *P* is highest. Break ties randomly. If the highest value is zero, *MoveRandomly( )*. When at a local maximum, signal *done*.

GoHome( ) The same as *GoUp(Home)*.

GoDown(*P, m (default = 0)*) Iteratively, in the current nine-cell neighborhood around the ant, move to the cell where the value of pheromone *P* is lowest. Break ties randomly. If the lowest value is at or below *m* or the ant cannot move because the surrounding area has a higher gradient (as happens at the world border), signal *done*.

MoveRandomly( ) Move to a random location in the nine-cell neighborhood around the ant. Because our environment is bounded, we include an additional protective measure: if the ant reaches a world border, *GoHome( )*.

MarkSite(*P*) Set the value of pheromone *P* at the ant's location to maximal, and mark it non-evaporating.

LoadRegister(*P*) Store in the ant's register the value of pheromone *P* at the ant's location.

FollowWorldBorder(*direction*) Temporarily turn off randomness so as not to lose the border. Head along the border of the world in *direction* (clockwise or counterclockwise).

FollowLine(*P, End*) Temporarily turn off randomness to not to lose the line. Among the eight-cell neighborhood around the ant, iteratively move to the cell where *P* is highest. Break ties by preferring forward-facing directions. When the ant has reached a cell with the *End* pheromone, signal *done*.

WallFollow(*P*) Head clockwise such that pheromone *P* is always maximal in the cell immediately to the ant's right (that is, follow along a "wall" of cells marked as sites for *P*).

MakeLine(*PointA, PointB*) This procedure sets up the gradients for straight line from *PointA* to *PointB*, but does not draw it. Iterate: *GoUp(PointA)*, then *GoUp(PointB)*. This causes the ants to go back and forth between the points, optimizing the trail until it is straight. After some time (perhaps 5000 steps), signal *done*.

BuildGradientDown(*P, m (default = 0)*) Build a gradient away from *P*, stopping when the gradient is well established down to value *m*. This is done by repeatedly iterating: *GoUp(P)*, then *GoDown(P,* $\alpha \times$ *m)*. The α makes the ant go one cell further than needed. The ant initially doesn't go straight down, but makes many random moves: and so we only signal *done* when *GoDown(...)* signals *done* and *all* the neighboring cells around the ant have nonzero values for *P*, indicating that it has likely built out the gradient well.

DrawLine(*PointA, PointB, Trace*) This procedure draws a straight line of pheromone *Trace* from *PointA* to *PointB*. First, *MakeLine(PointA, PointB)*. Then *GoUp(PointA)*. Temporarily turn off random moves to make a straight line, then *GoUp(PointB)* while calling *MarkSite(Trace)* on each new grid cell. Upon reaching *PointB*, signal *done*.

MakePerpendicularBisectorLine(*PointA, PointB, MarkA, MarkB, Temp*) The bisector line is the locus of cells where the gradients from points *PointA* and *PointB* are equal. Its ends are defined by *MarkA* and *MarkB*. This splits the swarm into two groups to build the two gradients in parallel.

Randomly do either: (1) *GoUp(PointA)*, then *MakeBisectorHalf(PointA, PointB, MarkA, MarkB, Temp)*; or (2) *GoUp(PointB)*, then *MakeBisectorHalf(PointB, PointA, MarkB, MarkA, Temp)*. Then signal *done*.

MakeBisectorHalf(*MyPoint, OtherPoint, MyMark, OtherMark, Temp*) This handles one sub-swarm. First, *BuildGradientDown(MyPoint)*. While doing so, when the pheromone value of *MyPoint* is less than or equal to the pheromone value for *OtherPoint*, *MarkSite(Temp)*; and whenever *Temp* is set at the ant's current location but the pheromone value of *MyPoint* is greater than the pheromone value of *OtherPoint*, remove it, as it has been set incorrectly due to pheromone evaporation.

Occasionally (with 0.1 probability) stop gradient-building and do the following. *GoDown(OtherPoint)* until the agent hits the world border, then *FollowWorldBorder(clockwise)* until one of three things happens: (1) If the value of *OtherPoint* is greater than *MyPoint*, the ant is too far: *GoUp(MyPoint)*, then continue *BuildGradientDown(MyPoint)* as before. (2) If the ant finds a cell with *Temp* set, this is the far end of the line. *MarkSite(MyMark)*, *FollowWorldBorder(counterclockwise)* a short distance (perhaps 100 steps), *GoUp(MyPoint)*, *GoUp(OtherPoint)*, *GoUp(Home)*, and *GoUp(MyPoint)*, then continue to *BuildGradientDown(MyPoint)* as before. This spreads the *MyMark* pheromone. (3) If the ant finds a non-zero *MyMark* gradient, the task is already completed: *GoUp(MyPoint)* and continue to *BuildGradientDown(MyPoint)* as before. Whenever both *MyMark* and *OtherMark* have been set at the ant's cell, *GoUp(MyPoint)*, then *BuildGradientDown(MyPoint)*: this erases *Temp* in all cells. After some time (perhaps 1000 steps), signal *done*.

The Five Geometric Construction Procedures

DrawCircle(*Center, EdgePoint, Temp, Circle*) This procedure draws a circle with pheromone *Circle*, centered at *Center*, and which passes through *EdgePoint*. A temporary and much thicker circle is marked first, then the outer border traced, since fluctuations in pheromones and random movements of the ants can cause variation in what the ants perceive as being the correct distance from the center.

MakeLine(Center, EdgePoint), then *GoUp(Center)*, then *GoUp(EdgePoint)*. At this point, *LoadRegister(Center)* to measure the gradient from *Center*, which will define the radius of the circle. *GoUp(Center)*. Next, repeatedly iterate through *BuildGradientDown(Center, register)*, then *GoDown(Center)*, then *MarkSite(Temp)*. This causes the ant to mark the outer edge of the circle with *Temp*. After some time (perhaps 3000 steps), enough marks have been set to form a solid circular wall. At this point, the ant must find its way to the outside of this wall. To do this, *GoUp(Center)*, then *GoDown(Center,* $\beta \times$ *register)*, which causes the agent to move out well beyond circle whose radius is defined by *register*, then *GoUp(Center)* until it finds a cell with a *Temp* pheromone value (the wall). We set $\beta = 0.01$ based on experiment. Finally, trace the circle: *WallFollow(Temp)* while simultaneously doing *MarkSite(Circle)* on each new grid cell. After some time (perhaps 500 steps), signal *done*.

MarkCircleCircleIntersections(*CenterA, CenterB, TempA, TempB, CircleA, CircleB, Mark1, Mark2*) This procedure identifies and uniquely distinguishes the intersections of two circles, one centered at *CenterA* and traced with *CircleA*, and one centered at *CenterB* and traced with *CircleB*. The *Temp* pheromones were those used to generate the original circles: they are called upon again to assist in wall-following.

First randomly *GoUp(CenterA)* or *GoUp(CenterB)*. If the ant chose *CenterA*, then *GoUp(CenterB)* until it finds *CircleA*, then *WallFollow(TempA)* until the ant reaches a cell with both *CircleA* and *CircleB*: then *MarkSite(Mark1)*. If the ant chose *CenterB*, then *GoUp(CenterA)* until it finds *CircleB*, then *WallFollow(TempB)* until the ant reaches a cell with both *CircleA* and *CircleB*: then *MarkSite(Mark2)*. In either case, *GoHome( )* and stay there for a while (perhaps 500 steps) to ensure other ants have seen the markings, then signal *done*.

MarkCircleLineIntersections(*PointA, PointB, Circle, Line, Mark1, Mark2*) This procedure identifies and uniquely distinguishes the (up to) two intersections of a circle traced out with *Circle*, and a line delimited by *PointA* and *PointB*, and traced out with *Line*. To make things simple, we assume that *PointA* and *PointB* are at the extrema of the line: this is not a problem, as the procedure for marking extended lines will mark the points at the borders of the environment.

First *GoUp(PointA)*, then *FollowLine(Line, PointB)*, and as the ant is doing so, *MarkSite(...)* the first intersection (which has both *Line* and *Circle* pheromones) with *Mark1*, and the second such intersection, if any, with *Mark2*.

MarkLineLineIntersection(*LineA, LineB, Mark*) This identifies the intersection between two lines. One simple (and inefficient) approach is to search the space until we have discovered the unique intersection, then mark it.

GoUp(Mark) (which moves randomly unless *Mark* is nonzero) until the agent has found a cell marked as both *LineA* and as *LineB*: at this point, *MarkSite(Mark)* then *GoHome( )*, then signal *done*. If another ant finds the intersection first, we may see the *Mark* pheromone already, in which case *GoHome( )* to tell the other ants, then signal *done*.

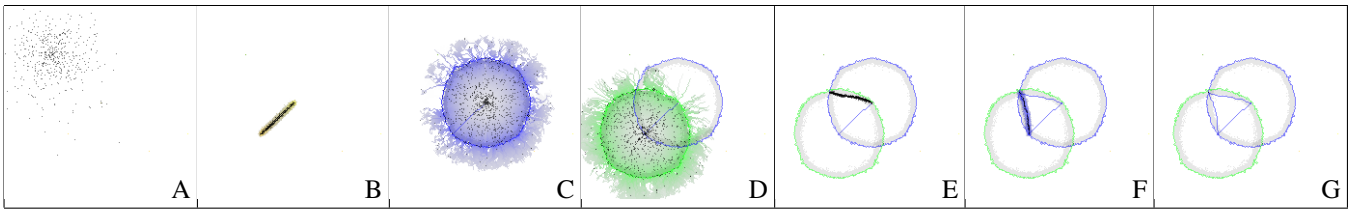


Figure 3: Building an Equilateral Triangle from Two Provided Points. (A) Emerging from Home. (B) Building the first line between the provided points. (C) Building the circle. (D) Building the second circle. (E) Determining the intersections, then building the second line. (F) Tracing the third line. (G) Finished.

DrawExtendedLine(*PointA, PointB, TempA, TempB, CircleA, CircleB, IntersectionA, IntersectionB, MarkA, MarkB, Line*) This procedure requires *nine* distinct pheromones because of its approach to drawing an extended (arbitrarily long) line between two points *PointA* and *PointB*. It first draws circles with *PointA* and *PointB* each as centers and passing through the other point, respectively. It then identifies the intersections of these circles. The extended line is the perpendicular bisector of these two intersection points.

The procedure is as follows. First *DrawLine(PointA, PointB, Line)*. Then *DrawCircle(PointA, PointB, TempA, CircleA)*. Then *DrawCircle(PointB, PointA, TempB, CircleB)*. Then *MarkCircleCircleIntersections(PointA, PointB, TempA, TempB, CircleA, CircleB, IntersectionA, IntersectionB)*. Next, *MakePerpendicularBisectorLine(IntersectionA, IntersectionB, MarkA, MarkB)* to identify the extrema of the extended line. Finally, *DrawLine(MarkA, PointA, Line)*, then *DrawLine(MarkB, PointB, Line)*.

Demonstration and Comparison

At this stage, if a shape can be provably built with compass-straightedge geometry, the ants can theoretically build it, given a finite-state automaton coded with the steps necessary. As a simple example, Figure 3 shows the process of building an equilateral triangle from two prespecified points. Using this approach, we have built a variety of structures: see for example the hexagon and angle bisector in Figure 4. Noise can cause some failures: in our simulator triangles presently have a 95% success rate, angle bisection is 91%, and hexagons are 87%: we believe these rates can still be improved.

While we have demonstrated that a basic pheromone model is *capable* of a nontrivial task such as this, we admit that it is not *efficient*: the agents build a variety of gradients throughout the environment, and if the system must use evaporation, then they must also maintain gradients until they have finished a subtask. Because of the locality of the pheromone model, and the need to move randomly, the approach is also quite *noisy*. As can be seen in Figures 3 and 4, the resulting lines, shapes, and angles are not ideal.

Broadcasting With *small* amount of global embedded communication, we can dramatically outperform a pheromone model. To show this, we compare the model against the same model augmented with *broadcast beacons*: objects which an ant may deploy at any time and associate with a pheromone. Ants can detect the presence, distance to, and relative angle to a broadcast beacon anywhere in the environment. As a result, they can easily follow along a circle (the locus of points a certain distance from a beacon), or head down a line passing through two beacons (the locus of points where both beacons are at the same relative angle or opposite angles). This eliminates the need to build and maintain gradients, and so ants with broadcast beacons can generally complete most tasks over dramatically faster. Furthermore, as they do not use local updating on a square grid, broadcast beacons' "gradients", so to speak, are circular without evaporation.

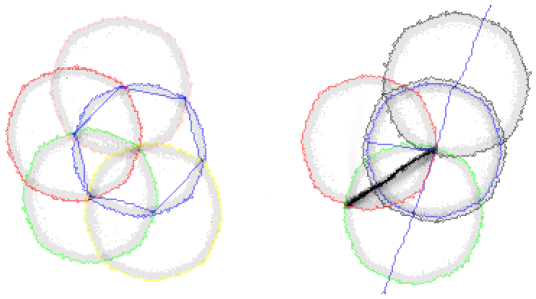


Figure 4: Hexagon (left) and Angle Bisection (right).

Except for the addition of broadcast beacons deployment and sensing, the ants are the same: in fact the revised procedures work largely the same way as the all-pheromone approach. One exception: because relative angle to beacons is reliable, the ants can do *DrawExtendedLine(...)* by simply going away from both beacons (that is, having both beacons directly behind the ant), though the original perpendicular bisector approach could still have been used.

For some examples, consider Figure 5, and compare against the same tasks done in Figures 3 and 4: there is something to be said for globally accessible information.[1]

Here we describe the additional basic procedures, then the geometric construction procedures, for ants using beacons.

[1] With three beacons, you could just do triangulation! But we want to show the ants' capability without sophisticated trigonometry.

Further Basic Procedures with Broadcast Beacons

GotoOrPlaceBeacon(*P*) If a beacon for *P* exists, head towards the beacon, and signal *done* on arrival. If the beacon does not exist, *GoUp(P)* until a site is discovered marked with *P*, then place a beacon for *P* at that site, and signal *done*.

GotoOrPlaceBeaconShortCircuit(*P*) If a beacon for *P* exists, simply signal *done*. If the beacon does not exist, *GoUp(P)* until a site is discovered marked with *P*, then place a beacon for *P* at that site, and signal *done*.

DrawLine(*LineA, LineB, P*) *GotoOrPlaceBeaconShortCircuit(LineB)*. *GotoOrPlaceBeacon(LineA)*. This ensures that beacons are located at both *LineA* and *LineB*, and that the ant is at *LineA*. Then head along the line between *LineA* and *LineB*, towards *LineB*, drawing the line with pheromone *P*. When at *LineB*, signal *done*.

Geometric Construction with Broadcast Beacons

MarkLineLineIntersection(*Line1A, Line1B, Line2, B*) *GotoOrPlaceBeacon(Line1A)*. Then head down the line which passes through *Line1A* and *Line1B*. It doesn't matter what direction. If the ant has gone "too far" (we define this as being at the world border, but any large measure is fine), turn around and head the other direction. When the ant discovers the pheromone *Line2*, denoting the trace of the second line, this is the intersection of the two lines. Place beacon *B* at this position and signal *done*.

MarkCircleLineIntersections(*LineA, LineB, Circle, A, B*) *GotoOrPlaceBeacon(LineA)*. Then head down the line which passes through *LineA* and *LineB* in the direction of (and past) *LineB*. When the ant has gone "too far", turn around and head the other direction. When the ant discovers the pheromone *Circle*, denoting the first intersection of the line and circle, place beacon *A* at this position. Then head down the line past *LineA*. When the ant has again gone "too far", turn around and head the other direction. When the ant discovers the pheromone *Circle*, denoting the second intersection, place beacon *B* at this position and signal *done*.

MarkCircleCircleIntersections(*CenterA, CenterB, CircleA, CircleB, A, B*) *GotoOrPlaceBeacon(CenterA)*, then *GotoOrPlaceBeacon(CenterB)*. Next, randomly head either to *CenterA* from *CenterB* or to *CenterB* from *CenterA*. If the ant goes to *CenterA*, then when the ant finds the circle traced with *CircleB*, follow clockwise along the *CircleB* pheromone until an intersection point is found (having both *CircleA* and *CircleB*), then *PlaceBeacon(A)*. Continue along *CircleB* until another intersection point is found, then *PlaceBeacon(B)*. On the other hand, if the ant goes to *CenterB*, then when the ant finds the circle traced with *CircleA*, follow clockwise along the *CircleA* pheromone until an intersection point is found (having both *CircleA* and *CircleB*), then *PlaceBeacon(B)*. Continue along *CircleB* until another intersection point is found, then *PlaceBeacon(A)*. Either way, finally signal *done*.

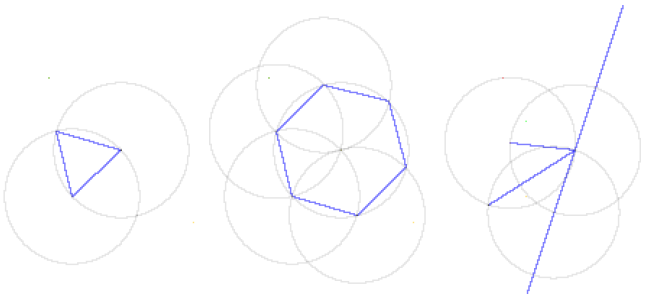


Figure 5: Equilateral Triangle (left), Hexagon (center), and Angle Bisection (right) using Broadcast Beacons. Compare with Figures 3 and 4.

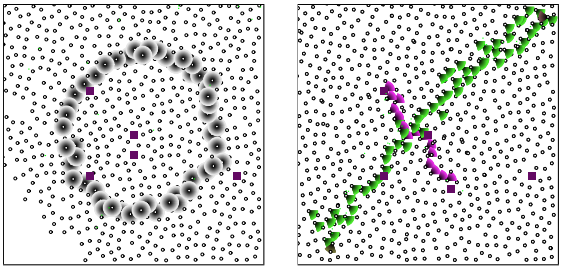


Figure 6: Circle (left) and Perpendicular Line Bisector (right) using a simulator for robots with deployable sensor motes.

DrawExtendedLine(*LineA, LineB, P*) *GotoOrPlaceBeaconShortCircuit(LineA)*. Then *GotoOrPlaceBeacon(LineB)*. This ensures that beacons are located at both *LineA* and *LineB*, and that the ant is at *LineB*. Then head along the line between *LineB* and *LineA*, towards and ultimately past *LineA*, drawing the line with pheromone *P*. When the ant has gone "too far", *GotoOrPlaceBeacon(LineA)*. Once at *LineA*, head along the line between *LineA* and *LineB*, towards and ultimately past *LineB*, drawing the line with pheromone *P*. When the ant has again gone "too far", signal *done*.

DrawCircle(*Center, EdgePoint, P*) *GotoOrPlaceBeaconShortCircuit(Center)*. Then *GotoOrPlaceBeacon(EdgePoint)*. This ensures that beacons are located at both *Center* and *EdgePoint*, and that the ant is at *EdgePoint*. *LoadRegister(Center)*. Then head along the path (either direction) where the *Center* pheromone is equal to the register (this essentially traces along the circle), while drawing the line with pheromone *P*. When the ant is back at *EdgePoint*, signal *done*.

Conclusion

We have demonstrated through proof of concept that, using only non-diffusing pheromones as a communication model, a swarm of ants can work together to perform compass-straightedge construction. This is a nontrivial task, but it demonstrates that indirect communication can enable sophisticated collaborative work.

This demonstration also illustrates a potential weakness in our pheromone model: it can be *very* inefficient, as whole ar-

eas must be painted with pheromones and perhaps constantly updated. For this reason we feel our demonstration falls near the limit of what these models are realistically capable of supporting. The broadcast beacons method, on the other hand, still allows for efficient constructions while using only sparse, robot-deployable devices.

Both methods are applicable to real robots, and immediate future work is to demonstrate this on an actual robot swarm. We have gathered preliminary (and noisy) results for the pheromone model in a simulator in which robots use deployable wireless sensor motes to create a sparse pheromone graph (as in Hrolenok et al. (2010) and Russell et al. (2015)); this is shown in Figure 6. To use such capabilities in a real-world scenario, however, will require solutions to a number of additional issues, including: obstacles in the environment, dynamic environments where marked areas might be removed, and environments without a known border.

Acknowledgments

This research was funded by NSF NRI grant 1317813.

References

Allwright, M., et al. (2014). SRoCS: Leveraging stigmergy on a multi-robot construction platform for unknown environments. In *Swarm Intelligence*, 158–169.

Ardiny, H., Witwicki, S., and Mondada, F. (2015). Construction automation with autonomous mobile robots: A review. In *RSI International Conference on Robotics and Mechatronics (ICROM)*, 418–424.

Bonabeau, E. (1996). Marginally stable swarms are flexible and efficient. *Journal de Physique I*, 6(2):309–324.

Brambilla, M., et al. (2013). Swarm robotics: a review from the swarm engineering perspective. *Swarm Intelligence*, 7(1):1–41.

Chibaya, C. and Bangay, S. (2007). A probabilistic movement model for shortest path formation in virtual ant-like agents. In *Conference of the South African Institute of Computer Scientists and Information Technologists on IT Research in Developing Countries*, 9–18.

Deneubourg, J.-L., et al. (1990). The self-organizing exploratory pattern of the argentine ant. *Journal of Insect Behavior*, 3(2):159–168.

Dorigo, M., Bonabeau, E., and Theraulaz, G. (2000). Ant algorithms and stigmergy. *Future Generation Computer Systems*, 16(8):851–871.

Gauss, C. (1801). *Disquisitiones Arithmeticae*, chap. 366.

Gleason, A. (1988). Angle trisection, the heptagon, and the triskaidecagon. *The American Mathematical Monthly*, 95(3):185–194.

Hrolenok, B., et al. (2010). Collaborative foraging using beacons. In *AAMAS*, 1197–1204.

Kowadlo, G. and Russell, R. A. (2008). Robot odor localization: A taxonomy and survey. *International Journal of Robotics Research*, 27(8):869–894.

Mamei, M. and Zambonelli, F. (2007). Pervasive pheromone-based interaction with RFID tags. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 2(2):4.

Mascheroni, L. (1797). *La Geometria del Compasso*.

Mohr, G. (1672). *Euclides Danicus*.

Panait, L. and Luke, S. (2004). A pheromone-based utility model for collaborative foraging. In *AAMAS*, 36–43.

Parker, C. A. C. and Zhang, H. (2006). Collective robotic site preparation. *Adaptive Behavior*, 14(1):5–19.

Pitonakova, L. and Bullock, S. (2013). Controlling ant-based construction. In *ALIFE*, 151–158.

Prabhakar, B., Dektar, K. N., and Gordon, D. M. (2012). The regulation of ant colony foraging activity without spatial information. *PLoS Computational Biology*, 8(8):1–7.

Russell, K., et al. (2015). Swarm robot foraging with wireless sensor motes. In *AAMAS*, 287–295.

Russell, R. (1999). Ant trails — an example for robots to follow? In *ICRA*, 2698–2703.

Sarhangi, R. (2007). Geometric constructions and their arts in historical perspective. In *Proceedings of the Bridges Donostia Conference*, 233–240.

Stewart, R. L. and Russell, R. A. (2006). A distributed feedback mechanism to regulate wall construction by a robotic swarm. *Adaptive Behavior*, 14(1):21–51.

Sugawara, K. and Doi, Y. (2014). Collective construction of dynamics structure: collaboration between semi-active blocks and simple robots. In *2014 IEEE/SICE International Symposium on System Integration (SII)*, 118–121.

Werfel, J. and Nagpal, R. (2008). Three-dimensional construction with mobile robots and modular blocks. *International Journal of Robotics Research*, 27(3-4):463–479.

Werfel, J., Petersen, K., and Nagpal, R. (2011). Distributed multi-robot algorithms for the TERMES 3d collective construction system. In *Workshop on Reconfigurable Modular Robotics, (at IROS)*.

Wodrich, M. and Bilchev, G. (1997). Cooperative distributed search: The ants' way. *Control and Cybernetics*, 26.

Ziparo, V., et al. (2007). RFID-based exploration for large robot teams. In *ICRA*, 4606–4613.