A distributed algorithm to maintain and repair the trail networks of arboreal ants

Arjun Chandrasekhar[1], Deborah M. Gordon*[2], and Saket Navlakha*[1]

[1]The Salk Institute for Biological Studies, Integrative Biology Laboratory, La Jolla, CA 92037 USA

[2]Stanford University, Department of Biology, Stanford, CA 94035 USA

Abstract

We study how the arboreal turtle ant (*Cephalotes goniodontus*) solves a fundamental computing problem: maintaining a trail network and finding alternative paths to route around broken links in the network. Turtle ants form a routing backbone of foraging trails linking several nests and temporary food sources. This species travels only in the trees, so their foraging trails are constrained to lie on a natural graph formed by overlapping branches and vines in the tangled canopy. Links between branches, however, can be ephemeral, easily destroyed by wind, rain, or animal movements. Here we report a biologically feasible distributed algorithm, parameterized using field data, that can plausibly describe how turtle ants maintain the routing backbone and find alternative paths to circumvent broken links in the backbone. We validate the ability of this probabilistic algorithm to circumvent simulated breaks in synthetic and real-world networks, and we derive an analytic explanation for why certain features are crucial to improve the algorithm's success. Our proposed algorithm uses fewer computational resources than common distributed graph search algorithms, and thus may be useful in other domains, such as for swarm computing or for coordinating molecular robots.

Introduction

Distributed algorithms allow a collection of agents to efficiently solve computational problems without centralized control [1]. Recent research has uncovered such algorithms implemented by many biological systems, including slime molds during foraging [2] and neural circuits during development [3]. Ants are a diverse taxon of more than 14,000 species that have also evolved distributed algorithms to establish trail networks [4]. Investigating the algorithms used by biological systems can reveal novel solutions to engineering problems [5, 3].

Here we present the first computational analysis, parameterized using data from field observations, of trail networks of an arboreal ant species. The arboreal turtle ant *C. goniodontus* nests and forages in the trees in the tropical dry forest of western Mexico [6]. Because the ants never leave the trees, their foraging trails are constrained by a natural graph: branches and vines form the edges in the graph, and junctions at overlapping branches form the nodes (Figure 1A–C). Each colony has

*Corresponding authors: dmgordon@stanford.edu, navlakha@salk.edu

several nests, located in dead tree branches, that are connected to each other in a circuit or network routing backbone [4, 7, 8]. Moving on the trails along this backbone, the ants distribute resources among the juveniles, workers, and reproductives in all of the nests, while additional temporary trails split from the backbone and lead to food sources. The backbone trail network can be large, often extending over 50 meters in circumference, and encompassing numerous trees [6]. The ants use many junctions in dense vegetation, so trails can be tortuous; each meter of linear distance typically requires ants to traverse approximately 2–5 meters of vegetation [6]. The colony thus chooses paths in the network from a myriad of potential routes, dictated by the graph structure of the vegetation. Ants lay trail pheromone as they move along the edges, and ants use pheromone when choosing edges.

We present a distributed algorithm that can plausibly describe how turtle ants maintain and repair breaks to their routing backbone. Links between branches or vines can be ephemeral, often disrupted by wind, rain, or the movement of an animal through the vegetation. To re-establish connectivity of the routing backbone after a break, the ants must establish a new path that reconnects the two sides of the broken trail. This is an important problem in many network applications [9] and can be solved efficiently using numerous graph algorithms, such as Dijkstra's algorithm or the Bellman-Ford algorithm [10]. However, these classic algorithms require significantly more computation and memory than is likely available to simple biological agents such as turtle ants, who regulate their behavior using local interactions rather than central control [11].

Repairing breaks requires overcoming three challenges. First, the ants must succeed in finding an alternative path by exploring new edges that currently have no pheromone and avoiding dead-ends in the network. One hypothesis for how this could be achieved is to first generate many candidate alternative paths and then converge to one or a few of them over time — a process we call "pruning". Such a strategy, also employed by slime molds [2] and neural circuits [3], has been shown to help quickly discover new paths in distributed settings, in which no agent is aware of the topology of the entire network. Second, all ants must converge to the same new path in order to optimally coordinate resource transport. Turtle ants travel in coherent trails that link nests and food sources [12]. After the vegetation supporting the trail is ruptured, the ants explore outside the previous path, and eventually commit again to a single path. Such convergence prevents ants from getting lost or separated from the rest of the colony. This is also an important goal in computer routing networks, where convergence to a single path ensures in-order delivery of data packets [13]. Third, it may be important to minimize the length of the new trail, which is also a standard measure of efficiency used when evaluating transport network design. However, data from field studies [6, 12] suggest that turtle ant paths are often not the shortest globally. It appears that the second objective, successful convergence, is more important than minimizing trail length, presumably because ants getting lost or separated has a higher cost than the energy spent in walking [12]. A common strategy to increase robustness to edge failures in a graph is to include loops in the path. Prior work [12] showed that loops do form in turtle ant trail networks; however, loops tend to get pruned over time, perhaps reducing the number of foragers needed to maintain the path.

The distributed algorithm used to maintain and repair trail networks must be robust across varying planar network topologies. The forest canopy is highly complex and dynamic, and it is unlikely that turtle ants use different algorithms to accommodate different network structures. Thus, we seek an algorithm that, while likely not "optimal" for any single planar topology, performs well across different planar topologies. The algorithm must also use very limited memory of indi-

vidual agents, as ants are not capable of remembering many of their steps along the graph structure.

Our work seeks to uncover a biologically plausible distributed algorithm that corresponds with field observations of turtle ant behavior in response to experimentally-induced edge breaks (Figure 1B) [12]. We ask:

1. What model is most likely to explain how turtle ants at a node select which edge to traverse next?

2. How well can the algorithm repair broken trails in simulated breaks in synthetic and real-world network topologies when parameterized by the most biologically realistic parameter values?

   (a) Does the algorithm consistently converge to a single consensus path?

   (b) Does the algorithm find short paths?

   (c) Does bi-directional search, using ants from both sides of the broken path concurrently, improve the performance of the algorithm relative to uni-directional search?

   (d) How does allowing an ant to avoid going back to the node it previously visited (backtracking), improve algorithm performance relative to performance when ants are not prevented from backtracking?

   (e) Can we provide any theoretical insights into why certain model features are necessary for any plausible turtle ant algorithm?

3. Can the same algorithm used to repair breaks also be used to keep the established routing backbone intact in the absence of a break?

4. Do turtle ants form multiple alternative paths and then prune some of them over time, as also observed in field studies?

A model that performs well on all of these criteria can be considered a plausible model of turtle ant behavior. Our main contribution is to identify several plausible non-linear models; we also show why one common linear model is likely implausible despite succeeding on some of the criteria listed above.

Related work

To our knowledge, this is the first computational analysis of trail networks of an arboreal ant species, whose movements are constrained to a discrete graph structure rather than continuous space. Compared to previous work, we attempt to solve the network repair problem using different constraints and fewer assumptions about the computational abilities of individual ants.

Species-specific modeling of ant behavior. Previous studies of ant trail networks have largely examined species that forage on a continuous 2D surface [14], including Pharaoh's ants [15], Argentine ants [4, 16, 17], leaf-cutter ants [18], army ants [19], and red wood ants [20]. These species can define nodes and edges at any location on the surface, and form trails using techniques such as random amplification [21, 22, 19], or using their own bodies to form living bridges [23]. Experimental

work on these species sometimes uses discrete mazes or Y-junctions to impose a graph structure; however, these species have evolved to create graph structures in continuous space, not to solve problems on a fixed graph structure, as turtle ants have evolved to do. Turtle ant movements are entirely constrained by the vegetation in which they travel. They cannot form trails with nodes and edges at arbitrary locations; instead, they can use only the nodes and edges that are available to them.

Further, to provide the simplest possible algorithm that is biologically realistic, we assume that turtle ants use only one type of pheromone. There are more than 14,000 species of ants, and they differ in their use of chemical cues. For example, *Monomorium pharoensis* uses several different trail pheromones [24, 25, 26, 27, 28]. There is, however, no evidence that turtle ants lay more than one type of trail pheromone.

Ant colony optimization. Models of ant colony optimization (ACO), first proposed in 1991, loosely mimic ant behavior to solve combinatorial optimization problems, such as the traveling salesman problem [29, 30, 31]. In ACO, individual ants each use a heuristic to construct candidate solutions, and then use pheromone to lead other ants towards higher quality solutions. Recent advances improve ACO through techniques such as local search [32], cunning ants [33], and iterated ants [34]. ACO, however, provides simulated ants more computational power than turtle actually ants possess; in particular, ACO-simulated ants have sufficient memory to remember, retrace, and reinforce entire paths or solutions, and they can choose how much pheromone to lay in retrospect, based on the optimality of the solution.

Prior work inspired by ants provides solutions to graph search problems [35, 36], such as the Hamiltonian path problem [37] or the Ants Nearby Treasure Search (ANTS) problem. The latter investigates how simulated ants collaboratively search the integer plane for a treasure source. These models afford the simulated ants various computational abilities, including searching exhaustively around a fixed radius [38], sending constant sized messages [39], or laying pheromone to mark an edge as explored [40]. Our work involves a similar model of distributed computation, but our problem requires not only that the ants find an alternative path to a nest (a "treasure"), but also that all the ants commit to using the same alternative path. This requires a fundamentally different strategy from that required for just one ant to find a treasure.

Graph algorithms and reinforced random walks. Common algorithms used to solve the general network search and repair problem, including Dijkstra's algorithm, breadth-first search, depth-first search, and A* search [10], all require substantial communication or memory complexity. For example, agents must maintain a large routing table, store and query a list of all previously visited nodes, or pre-compute a topology-dependent heuristic to compute node-to-node distances [41]. These abilities are all unlikely for turtle ants.

Distributed graph algorithms, in which nodes are treated as fixed agents capable of passing messages to neighbors, have also been proposed to find shortest paths in a graph [42, 43], to construct minimum spanning trees [44, 45], and to approximate various NP-hard problems [46, 47]. In contrast, our work uses a more restrictive model of distributed computation, where agents communicate only through pheromone which does not have a specific targeted recipient.

Finally, the limited assumptions about the memory of turtle ants invite comparison to a Markov process. Edge-reinforced random walks [48], first introduced by Diaconis and others [49, 50], proceed as follows: an agent, or random walker, traverses a graph by choosing amongst adjacent edges with

a probability proportional to their edge weight; then the agent augments the weight (or pheromone) of each edge chosen. Our model expands edge-reinforced random walks in two ways: first, we allow many agents to walk the graph concurrently, and second, we decrease edge weights over time. Our work is similar to previous models of the gliding behavior of myxobacteria [51] that consider synchronous, node (rather than edge)-reinforced, random walks with decay. These models seek to determine when bacteria aggregate on adjacent points or instead walk freely on the grid. By contrast, here we ask whether the random walkers converge to a single consensus path between two points on the grid that are not necessarily adjacent.

Results

Our goals are to find an algorithm that can simultaneously explain the movement patterns of turtle ants on a trail network and that can effectively solve the network repair problem. First, we describe a computational framework for evaluating the collective response of turtle ants to edge ruptures. We evaluate the response according to three objectives: the likelihood of finding an alternative path to repair the trail, how well the ants converge to the same new trail, and the capacity to minimize the length of the trail. Second, we derive multiple candidate distributed algorithms for network repair. We parameterize each algorithm using data from field experiments to determine how the model would predict which edge a turtle ant would choose to traverse next from a node, given only local information about adjacent edges and their edge weights. Third, we analyze via simulation how our algorithms perform on different planar network topologies, including simulated breaks on a European road transport network.

A graph-theoretic framework for modeling network repair by turtle ants

We start with a weighted, undirected graph $G = (V, E, W)$, where V is the node set, E is the edge set, and W are the edge weights, as well as two nest nodes $u, v \in V$, and a path $P = (u, \ldots, v)$ from u to v with pheromone along each edge in P. Edges are undirected since turtle ants can walk in both directions over edges. Edge weights correspond to the amount of pheromone on the edges, which can change over time. We mimic a break in the path by removing some edge in P. The challenge for the ants is to find an alternative path that reconnects u and v. The alternative path may build off the existing path, so that the initial and final path may share some edges.

Communication among simulated ants is limited to chemical signals, analogous to pheromone, left on edges traversed. Field observations are consistent with the assumption that, like Argentine ants [52], turtle ants lay trail pheromone continuously as they walk [6, 12]. Though this has not been observed directly, we hypothesize that there are certain exceptional situations in which turtle ants discontinue laying pheromone (Methods). Each ant at a node senses the pheromone level on adjacent edges to inform its next movement. Observations suggest that a turtle ant tends to keep moving in the same direction, indicating that an ant is able to avoid the previous node it visited. We thus assume that simulated ants have one time-step of memory, used to avoid going back and forth along the same edge. As is characteristic of many species of ants [53, 54, 55], simulated ants have no unique identifiers and can use only local information.

Parameters. Our algorithm uses three biological parameters: $q_{\text{add}}, q_{\text{decay}}$, and q_{explore}.

The first parameter (q_{add}) determines how much pheromone is added when an ant traverses an

edge. After each time step, each edge (v_1, v_2) traversed increases its edge weight as:

$$w(v_1, v_2) \leftarrow w(v_1, v_2) + q_{\text{add}}. \tag{1}$$

Without loss of generality, we fix $q_{\text{add}} = 1$, representing a unit of pheromone that an ant deposits on each edge traversed.

The second parameter (q_{decay}) specifies how much pheromone evaporates on each edge in each time step due to natural decay. We model pheromone decay as an exponential decrease in edge weight [56, 57]; thus $q_{\text{decay}} \in (0, 1)$, and at each time step, for each edge (v_1, v_2), its weight is updated as:

$$w(v_1, v_2) \leftarrow w(v_1, v_2) \times (1 - q_{\text{decay}}). \tag{2}$$

Larger values of q_{decay} correspond to more rapid decay of pheromone on the edge.

The third parameter (q_{explore}) specifies the probability that an ant takes an "explore step". The definition of an "explore step" is algorithm-specific (see below), but intuitively, it involves choosing an edge with relatively less or no pheromone. Such deviation is clearly required by any network repair algorithm, since after the routing backbone is ruptured, edges not part of the existing path must be traversed to repair the break. Field observations show that even in the absence of a break, turtle ants explore edges off the main trail. This allows them to discover new food sources and incorporate them into the trail network [12].

Performance metrics. After T time steps, we evaluate the outcome of the algorithm using the following measures (averaged over 50 repeat simulations):

1. Success rate: The probability that the simulated ants succeeded in forming a new path from u to v that does not use the broken edge. In this new path, ants are not required to traverse edges of relatively low weight (Methods). Higher values are better; for example, a success rate of 70% means that in 70% of the simulations, the ants successfully formed an alternative path.

2. Path entropy: An information-theoretic measure of how well the ants converge to a single consensus path, rather than creating multiple, potentially overlapping, $u \to v$ paths with pheromone. Lower values are better, indicating that subsequent ants using the same algorithm on the resulting network will all follow a common path, rather than dispersing along many different paths. This measure is computed only in the simulations in which an alternative path was successfully found. Field observations show that turtle ants consistently converge to a consensus path, and loops in the network are often pruned away over time [12]. This reduces the numbers of lost ants and the numbers of ants traveling in circles.

3. Path length: The length of the new path. Although turtle ants do not always find the globally shortest path [12], we include this measure because it is commonly used to evaluate routing algorithms. Lower values are better, indicating shorter paths. This measure is computed only in the simulations in which an alternative path was successfully found.

A model of computation for individual ants. We assume that all ants are identical and have the following computational abilities:

- Each ant can avoid the node it immediately previously visited. It cannot, however, remember its entire path from the nest up to its current point. The ant may also keep track of a binary state variable that determines whether it is combing back from a dead end and should discontinue laying pheromone.

- In field observations, ants appear to pause at nodes and inspect more than one edge before choosing an edge to take [12]. Thus, each ant can access all adjacent edge weights to decide which node to visit next. To choose its next edge, we allow ants to perform any Turing-computable computation, although we show that a simple, albeit non-linear, function will suffice.

See Methods for full technical details of the model and performance metrics.

Candidate distributed algorithms

Below we introduce several biologically plausible algorithms that attempt to describe how a turtle ant at a node s chooses which edge to traverse next among possible neighboring edges $t_1, t_2, ...t_n$. These algorithms build upon previous linear and non-linear models used to analyze ant trail formation in other species, such as Argentine ants [58, 59, 60] and pharaoh ants [61]. Let $w(s, t_i)$ be the current weight on edge (s, t_i), and let uniform() be a random value drawn uniformly from $[0, 1]$.

In the WEIGHTED random walk (Algorithm 1), each ant chooses the next edge to traverse with probability proportional to the amount of pheromone on that edge: the more pheromone on an edge, the more likely an ant is to traverse that edge. However, with probability q_{explore}, the ant takes an edge that has zero pheromone.

Algorithm 1 WEIGHTED random walk

$X \leftarrow \{t_i : w(s, t_i) > 0\}$ # Explored edges
$Y \leftarrow \{t_i : w(s, t_i) = 0\}$ # Unexplored edges
if uniform() $< q_{\text{explore}}$ **then**
  return $t_i \in Y$ with probability $1/|Y|$
else
  return $t_i \in X$ with probability $w(s, t_i)/\sum_{j \in X} w(s, t_j)$
end if

Note: The algorithm excludes the previously visited node from the sets of candidate edges. If all neighboring edges have weight 0, the ant chooses a zero-weight edge with probability 1 rather than probability q_{explore}. If none of the neighboring edges have weight 0, then the ant chooses an edge with nonzero-weight with probability 1 rather than probability $1 - q_{\text{explore}}$.

In the RANKEDGE random walk (Algorithm 2), with probability $1 - q_{\text{explore}}$, the ant chooses an edge with the highest weight (ties are broken at random). With probability q_{explore}, it bypasses the highest weighted edges and considers edges with the second highest weight. With probability $q_{\text{explore}}(1 - q_{\text{explore}})$, it chooses an edge with the second highest weight. With probability q^2_{explore}, it bypasses both the highest and second highest weighted edges and considers edges with the third highest weight, and so on.

Algorithm 2 RankEdge random walk

$W \leftarrow [w_1, w_2, \ldots, w_k]$ # Sorted unique edge weights, in decreasing order
for $i = 1 \ldots k$ **do**
  $X \leftarrow \{t : w(s,t) = W[i]\}$
  if uniform() $< (1 - q_{\text{explore}})$ **then**
    return $t_i \in X$ with probability $1/|X|$
  end if
end for

Note: The algorithm excludes the previously visited node from the set of candidate edges. If all neighboring edges are tied for the highest weight, then a maximally-weighted edge is chosen with probability 1 rather than probability $1 - q_{\text{explore}}$. If an ant keeps exploring until it gets to the lowest weight, it takes one of the edges tied for the lowest weight with probability 1 rather than probability $1 - q_{\text{explore}}$.

Each algorithm contains additional details inspired by field observations, including a queueing system so ants traverse edges one at a time, the ability to traverse and return from an edge on an explore step in one time-step, and the ability to discontinue laying pheromone on the way back from a dead-end. See Methods for full details.

Other algorithms: We compared these two candidate distributed algorithms to several other non-linear algorithms (MaxEdgeA, MaxEdgeB, MaxEdgeC, MaxWeighted, and Deneubourg), as described in the Supplement. We also compared to a null model, called the Unweighted random walk. The null model uses no parameters; instead, the ants ignore edge weights and choose amongst candidate edges with equal probability.

Summary of conclusions. Overall, we find that non-linear models perform the best at simultaneously explaining field observations and providing a mechanism by which turtle ants could solve the network repair problem. While the linear (Weighted) algorithm does perform well at explaining some aspects of field observations, and it repairs breaks with high probability, it also produces a very high path entropy, with poor convergence to a consensus path. This departs strongly from field observations [12] that show that when repairing broken paths, the ants quickly converge to a single path.

In particular, we find that: (A) RankEdge outperforms all other non-linear algorithms, except for MaxEdgeA, in the likelihood of explaining observed edge choices by turtle ants (Table 1). The log-likelihoods of RankEdge and MaxEdgeA were nearly identical. (B) When parameterized by field data, RankEdge outperforms all other non-linear algorithms in success rate (Table 2); and (C) RankEdge is equivalent to all other non-linear algorithms in path entropy (Table 3). Compared to the linear Weighted algorithm, RankEdge has a lower likelihood of explaining the observed edge choices and a lower success rate. However, RankEdge performs much better in path entropy, path length, and maintaining the trail in the absence of a break. We emphasize that the strong success rate of Weighted is because pheromone is left essentially on every edge in the graph. This guarantees high success but very poor convergence to a single path. Field experiments show that turtle ants converge strongly to a single path.

Q1. Field observations to determine the best algorithm and parameter values

We first determined what parameter values best allow each algorithm to match the data from field observations. We then used these parameter values to test algorithm performance for the network repair problem.

The performance of each candidate algorithm is sensitive to the values chosen for the two free parameters, q_{explore} and q_{decay} (as previously mentioned, we set $q_{\text{add}} = 1$). For example, with low values of q_{explore}, the ants may take a long time to explore enough new edges to find an alternative path; on the other hand, for high values of q_{explore}, the ants will scatter throughout the network and may not converge to a single path. Similarly, for high values of q_{decay} (pheromone decays rapidly), it may be difficult to build and reinforce a single path; for low values of q_{decay}, it may be hard for the colony to eliminate unnecessary edges and commit to one path. These two parameters also affect each other; for example, the higher the decay rate, the fewer edges with pheromone, and thus the more possible edges to explore.

We used data from observations made in the field to evaluate the match between the choices of edges made by turtle ants and the choices predicted by a candidate algorithm (with parameters $q_{\text{explore}}, q_{\text{decay}}$). Observations were made at La Estacion Biologica de Chamela in Jalisco, Mexico [6, 12]. Ants were observed traversing a junction (node) along a foraging trail. We recorded the time at which an ant moved to or from that junction node, and the edge it chose to traverse (Figure 2A–B). Observations were made of six different colonies, with an average of 2.16 junctions per colony, over three days in June 2015 and one day in June 2016. We observed 13 different junctions for time periods ranging from 7 to 24 minutes (mean of 12.3 minutes per observation at a given colony on one day), for a total of 773 edge choices made by turtle ants.

Maximum likelihood estimation. We determined which algorithm and parameter values best explained the observed edge choices made by turtle ants using maximum likelihood estimation (MLE). The data were used to determine the likelihood that a given algorithm, with a given pair of parameter values, would have produced the observed set of edge choices. Figure 2A–B shows an example likelihood calculation, and Figure 2C–D illustrates the results of the MLE for each algorithm over all pairs of parameter values.

Overall, for RankEdge, the maximum likelihood parameter values that best explained the observed turtle ant behavior were: $q_{\text{explore}} = 0.20$, and $q_{\text{decay}} = 0.02$ (Table 1). For Weighted, the maximum likelihood parameter values were $q_{\text{explore}} = 0.05$ and $q_{\text{decay}} = 0.01$. Both candidate algorithms were more likely to explain the data than the null model (Table 1).

Consistency of the maximum likelihood estimation across colonies and days. The maximum likelihood parameter values were similar across colonies and days for the 13 junctions (Figure S2). This suggests that across six colonies, there are similar chemical properties in the pheromone (related to q_{decay}), and that a similar search strategy is used for choosing which edge to traverse next (related to q_{explore}).

Q2. Algorithm performance on synthetic and real-world planar networks

Our goal here is to test how well each algorithm solves the network repair problem on simulated and real-world networks. We were particularly interested in how well each algorithm performed when its parameters were set to the maximum likelihood values derived from observations of turtle

ants. Our main result is that the maximum likelihood parameters for the RankEdge performed well in simulations for network repair across six networks (Figure 3; the black rectangle in both panels shows that the parameter values that best explain the turtle ants' behavior also perform best for solving the network repair problem.) The latter result is substantiated below.

Simulation setup. For all simulations, we ran each algorithm for $T = 1000$ steps using $N = 100$ ants, and repeated each simulation 50 times. To initialize each simulation, we placed each of the N ants at a random node in the original path. This means that at the start of the simulation there were likely ants at nodes on both sides of the rupture in the path. No ants were placed at nodes not part of the original path. Each ant was randomly assigned to walk in search of one of the two nests. All edges that were part of the initial path were initialized with 10 units of pheromone. All other edges were initialized to 0 units of pheromone. When an ant reached its destination nest, it attempted to return to the other nest, and repeated this, going back and forth between nests, for T time-steps. The ants walk synchronously for T time-steps; this is a common assumption in distributed computing problems.

Our first performance metric, called the *success rate*, measures how well the ants succeed in finding an alternative path to repair the break. We simulated breaks under six planar network structures, which have an increasingly complex topology with varying numbers of possible paths. In each evaluation below (Figure 4), we show three panels: the initial network with a break, the final network at the end of the simulation, which is generated using the MLE parameter values, and a heatmap showing the success rate for pairs of parameter values $(q_{\text{explore}}, q_{\text{decay}})$ close to the MLE range. In each synthetic network, a only single link is broken (shown as the 'X' mark in Figure 4); in the Supplement, we describe cases where multiple links are broken.

We analyzed all algorithms but show results only for RankEdge in the main text because Weighted rarely converged onto a single path, thus it did not satisfy our second performance metric. It also did not maintain trails in the absence of a break. These results are described in detail below. Also, see Table 2 and Supplement for analysis of the additional non-linear algorithms.

Minimal graph (Figure 4A). Here we find that RankEdge can solve a basic repair problem in a minimal working example, in which the break causes the existing path to lead to a dead end that should be avoided in favor of a single alternative path to the nest. To favor the alternative path, the simulated ants must largely eliminate the pheromone on the edge leading to the dead end, a process which we call 'pruning'. To favor the alternative path instead of the existing path, the ants should put more pheromone on the edge leading upwards to the alternative route, even though this edge initially had no pheromone.

We find that the RankEdge algorithm succeeds in this task 100% of the time, as long as the ants do not leave pheromone on the way back returning from the dead-end (Methods and Q4).

Simple graph (Figure 4B). Here we increased the complexity of the graph to offer two alternative paths, instead of one in the Minimal graph. We found that RankEdge not only prunes the dead-end, but it can find and commit to one of the two alternatives with a 98% success rate.

Medium graph (Figure 4C). Here we further increased the complexity of the Minimal graph to offer six alternative paths and found that RankEdge not only prunes the dead-end, but can find and

commit to one of the six alternatives with a 84% success rate.

Full grid (Figure 4D). The Full grid presents a different computational challenge: there is no dead-end to prune and the shortest alternative path requires only 3 additional edges. However, the total number of possible new paths is extremely large, which makes it difficult to find and commit to a single path. The Full grid is also a standard benchmark used in the ANTS problem (Related work), in which ants search the integer plane [38, 39, 40].

We found that the highest success rate (70%) occurred for low values of q_{decay}, which closely matches the observed best decay value estimated using maximum likelihood. This highlights an inherent trade-off in the turtle ant algorithm. Low decay rates help preserve the initial path and bias the turtle ants toward finding an alternative route that re-uses as much of the previous path as possible; that is, with low decay rates, repair starts as close to the break as possible. However, low decay rates also limit the capacity to search for other paths that may be shorter even though they re-use less of the previous path. An alternative would be to use higher values of q_{explore} to search for other paths that do not re-use the initial path, but this would make it more difficult for the ants to converge to a single new path.

Spanning grid (Figure 4E). In contrast to the Full grid, the Spanning grid is sparser and requires that the ants go back at least one node from the break to find an alternative path.

We found that the maximum likelihood parameters produced a moderate success rate (54%). As above, the highest success rate occurred for low values of q_{decay} and moderate values of q_{explore}. These values achieve a good trade-off between searching sufficiently far from the break to find an alternative path, and largely preserving the previous path. The performance on the Spanning grid demonstrates that the algorithm is flexible enough to search locally around a break point for new paths, while still maintaining most of the old path.

The results from the Full grid and the Spanning grid together suggest that the algorithm performs best when it preserves as much of the previous path as possible, even if it can not re-use all of the original path. This is consistent with field observations that showed that turtle ants sought alternative paths in a "greedy" manner, by going back up to 1 or 2 nodes from the break point, even though going back more nodes may have resulted in a path with fewer nodes overall [12].

European road transportation network (Figure 5). To demonstrate the utility of this algorithm in a real-world scenario, we applied the RankEdge algorithm to repair networks in a human-designed transport network. We downloaded the network depicting the major roads (edges) connecting intersections (nodes) in the international E-Road in Europe [62] (Methods). We removed an edge from an existing path between two nodes and ran the RankEdge algorithm to repair the simulated closure. The RankEdge algorithm achieved a success rate of 70%, indicating that the turtle ant algorithm can also repair breaks in real-world topologies. This shows how distributed solutions may be useful for new application domains, such as for swarm robotics or molecular robots [63, 64, 65, 66] in remote environments, when centralized or global positioning systems may not be as effective.

We also compared the algorithms on Erdos-Renyi random networks and small-world networks and found similar gains in performance for RankEdge (Supplement).

Q2a. Converging onto a single consensus path (Table 3). Our second performance metric,

called *path entropy*, measures how well foragers commit to a single alternative path.

We find that the RankEdge algorithm consistently achieves a path entropy near 0, indicating that on the final network all ants follow the same path when not taking explore steps (Table 3). This is particularly challenging for the Full and Spanning grids because both contain a large number of possible paths, and thus a large possible path entropy if the simulated ants exploit many paths. Thus, when the algorithm succeeds in repairing the path, RankEdge satisfies our second performance criterion.

More generally, one advantage of non-linear algorithms such as RankEdge is that all simulated ants, by simply following the maximal edge (the adjacent edge with the highest pheromone), can travel from one nest to the other using the same path, thereby achieving a path entropy of 0. On the other hand, linear algorithms such as Weighted do succeed in finding a path; however, Weighted is unable to commit to only one path, and thus has very high path entropy (Figure 6).

Q2b. Finding short paths (Table 4). Our third performance metric measures the path length of the final trail network. We found that RankEdge consistently finds paths of lengths that are close to, though slightly larger than, the globally shortest path lengths. For every network, we compared the average path lengths of RankEdge versus every other algorithm using Welch's unpaired T-test. RankEdge finds significantly shorter paths than Weighted and Unweighted on the Full grid, Spanning grid, and European roads ($p < 0.05$); RankEdge is not significantly different from the other non-linear algorithms (Table 4). The improved performance over Unweighted demonstrates the value of using pheromone to solve the network repair problem collectively, instead of using independent search.

Q2c. The power of bi-directional search (Figure S3). We find that a bi-directional search, in which simulated ants attempt to create an alternative path concurrently from both sides of the break, allows the algorithm to perform significantly better than a uni-directional search using ants from only one side of the break. We tested this on the Full grid, and found that for the MLE parameter values for RankEdge, the success rate was on average 70% for a bi-directional search versus 14% for uni-directional search.

One might predict that uni-directional search would perform as well as the bi-directional search, while simply taking longer. However, we found this not to be true: using a bi-directional search means that once ants from side A of the break reach side B of the break, the rest of their search is directed by the pheromone trail laid by ants that started on side B. In the uni-directional search, even if ants from side A reach side B, they must still find a path from scratch connecting the dead end on side B to the nest on side B. Although uni-directional search has rarely been observed to occur in turtle ant networks, we tested it here to compare it with bi-directional search, which is often used to improve the performance of search algorithms.

Q2d. The power of avoiding backtracking (Figure S4). We find that providing simulated ants the ability to avoid backtracking, i.e., visiting the same node visited in the previous time-step (Methods) allows for a significant improvement in algorithm performance. In contrast, ants that are not given this ability could keep going back and forth along the same edge.

In particular, ants that used the RankEdge algorithm and avoided backtracking produced a success rate of 70% on the Full grid, compared to 0% when an ant was not prevented from potentially returning to the previous node it visited (Figure S4). Thus, providing ants with a basic

node-to-node sense of direction led to a significant improvement in performance.

Q2e. Critical features for the success of a plausible algorithm. We find that there are two important features for non-linear algorithms to perform well in simulation: (1) Simulated ants do not lay pheromone on the way back from a dead end, and (2) simulated ants queue at nodes (Methods). In the Supplement, we provide theoretical analysis for why these two components are critical for any non-linear algorithm to circumvent a dead end. Without either of these two features, the time to circumvent a dead end rises dramatically. In the Supplement, we also confirm these theoretical observations via simulation.

Q3. Maintaining a trail in the absence of a break

Here we consider whether the same algorithm used to repair a path can also keep a path intact when it is not broken. This is important because if different algorithms were used to maintain trails versus repair trails, then the turtle ants would need some signal to toggle between different methods for choosing among candidate edges, depending on the context. We found that a single algorithm, RankEdge, is capable of maintaining trails and responding to breaks.

In particular, we ran the RankEdge algorithm on the Spanning grid without breaking the original path and found that the trail was preserved without any modification to the algorithm or its parameters (Figure 7). In contrast, the Weighted algorithm performed very poorly on this task. In particular, for RankEdge, the path entropy using the MLE parameter values from turtle ant data was optimal (0.00). For Weighted, however, the path entropy for the MLE parameter values was much higher (5.38), indicating poor maintenance of the original path.

Q4. Pruning as a general principle for discovering alternative paths

Field observations show that turtle ants engage in pruning (Figure 8). In our simulations, we also observed that ants explored multiple alternative paths, and then most of these paths were pruned as the colony converged to a single alternative path. Further, the paths tended to become shorter over time. We quantified how many paths were pruned during the simulation using a measure called *path elimination* (Methods). We also quantified how the lengths of the remaining paths changed over time using a measure called *path length pruning* (Methods). All of the non-linear algorithms exhibit some path elimination and pruning, and thus could plausibly be used to explain the observed pruning in observed turtle ants. RankEdge prunes fewer paths than the other algorithms (Table 5) because RankEdge does not form as many initial paths as other algorithms. However, of the paths that are pruned, RankEdge tends to prune more nodes from the paths (Table 6).

For every network, we compared the average path elimination and path length pruning of RankEdge versus every other algorithm using Welch's unpaired T-test. For path elimination, RankEdge does not reduce the number of paths more than other algorithms ($p < 0.05$), as we described above. However, for path length pruning, RankEdge reduces path lengths significantly more than all other algorithms on the Spanning grid and European roads ($p < 0.05$). On the Full grid, RankEdge is significantly better than Weighted ($p < 0.05$), is not significantly different from Deneubourg or MaxEdgeB, and is significantly worse than MaxEdgeA, MaxEdgeC, and MaxWeighted. No statistical difference is observed for the Simple and Medium graphs, likely due to their relatively simple topology. These pruning results suggest that RankEdge explores fewer

paths initially but is better at selecting the shortest of the paths it explores. RankEdge tends to prune fewer paths (path elimination), but of the paths it does explore, RankEdge converges to the shorter paths (path length reduction).

Field observations [12] also showed that when ruptured trails are repaired, new nodes are added to the network, and in subsequent days, some of the nodes are pruned. Such pruning of nodes also led to global pruning of paths (Figure 8). Such an "explore-exploit" strategy may help turtle ants quickly find a solution that re-connects a rupture in a trail, and may also help the colony to optimize the coherence of the trail, by minimizing the number of junctions at which ants could get lost.

Interestingly, using pruning-based strategies to discover the most appropriate edges or paths to keep is a common strategy used by biological systems. In particular, during the development of neural circuits in the brain, synapses are massively over-produced and then pruned-back over time [3]. This strategy is thought to help neural circuits explore possibly topologies and then converge to the most appropriate topology based on environment-dependent feedback. A similar process occurs during the development of vascular (blood flow) networks in the body [67]. Thus, pruning may be a common biological strategy of network design when multiple topologies need to be explored in a distributed manner.

Discussion

Our primary contribution is to address an engineering problem (maintaining a trail network and finding alternative paths to route around broken links in the network) using biologically feasible parameters and models motivated by how turtle ants may solve this problem in the field. Successful performance by the algorithm in simulation, using realistic parameter values, indicates that the algorithm is a plausible candidate to describe how the ants create their networks. The RankEdge algorithm achieved a better maximum likelihood estimate (Figures 2–3) than every other non-linear model except for MaxEdgeA. When parameterized by data from field observations, RankEdge was better able to find a single, short path with high probability compared to other algorithms (Tables 2,3, 4). From this, we conclude that non-linear models, in particular RankEdge and MaxEdgeA, represent the two best plausible models of turtle ant behavior.

By testing performance across six different networks, we found that the turtle ants appear to have evolved an algorithm that may not be optimal for any particular planar network but is robust to some variation in the topology. Further, non-linear algorithms exhibited pruning, which also occurred in field observations [12] (Table 6, Figure 8).

There are several features of our algorithm that are critical for success in repairing breaks to the routing backbone. First, to minimize path entropy it is essential to have a stronger-than-linear bias towards choosing the highest-weighted edge (RankEdge), rather than choosing edges proportional to their edge weight (Weighted). This helps constrain the search space and leads to better convergence to a single consensus path. We emphasize that Weighted has a strong success rate because pheromone is left on every edge in the graph (see e.g., Figure 6A–B). This guarantees that there exists some path with positive probability. However, Weighted does not commit to a single path as effectively as an algorithm with a strong, non-linear bias toward the highest-weighted edge, such as RankEdge. The path entropy of Weighted is high because essentially every path in the graph has high probability, and the average path length is high because the algorithm does little

to eliminate long paths and commit to short paths. Second, to repair breaks it is essential to use bi-directional search and avoid backtracking. Observations show that when turtle ants encounter a break, ants from both sides of the break attempt to repair the trail [12]. When ants from both sides meet, they each encounter a trail that is already strongly reinforced and guided towards the other nest. In addition, the ability to avoid backtracking allows ants to avoid going back and forth along the same edge. Third, we showed theoretically that the time needed to find an alternative path decreases significantly if turtle ants reaching a dead-end in their trail do not leave pheromone while returning back from the dead-end.

The RankEdge algorithm is parsimonious, capable of both maintaining trails and repairing breaks to trails using the same underlying logic. Observed ants encounter diverse situations analogous to breaks in the ongoing maintenance of trails. We find that a single algorithm can solve two diverse problems without requiring the additional complexity of a signal that distinguishes such situations from a rupture in the trail. How each path is established originally is an interesting yet distinct question. Paths are not always the shortest globally, and the physical structure of edges in the canopy appears to affect how these paths are selected.

The algorithm can be extended to improve performance, though this may involve sacrificing biological realism. One possible extension would allow ants to "toggle" between different parameter values or algorithms in different situations. For example, an ant could use RankEdge, but if it encounters a dead-end or massive crowding (determined for example by a large increase in the frequency of antennal contacts with other ants [68, 18]), then it increases its probability of exploring new edges. This would be similar to a distributed version of simulated annealing, with the value of q_{explore} corresponding to the decreasing value of the temperature parameter. A second possible extension would be to use multiple types of pheromone [69]. Ants could use negative pheromone to signal to other ants not to select a certain edge, for example, towards a dead-end. Further work is needed to measure the computational abilities of turtle ants to determine whether such extensions depart from biological realism.

There are some differences between our synthetic networks and the environment of the observed turtle ants. First, turtle ant trail networks in the canopy are 3D planar networks, whereas here, to begin the investigation of arboreal ant trail networks, we used 2D planar networks. The ideal test case would be a suite of synthetic networks that are isomorphic to some portion of the turtle ant canopy. In lieu of this, we describe five synthetic networks, some of which have been used by prior work, that each collectively test the ability of different algorithms to solve the network repair problem. These five networks comprise a necessary (if not exhaustive) set of test cases. Second, in the tropical forest, many edges are physically difficult to traverse, which may provide natural inhibition for selecting certain edges. Third, in the canopy, edges are not all of the same length. In future work that includes variability in edge lengths, synchronous walks will need to be modified since longer edges require more time-steps to traverse. More generally, further work is needed to determine the physical properties of junctions and branches in the canopy and how these properties influence the likelihood of traversing an edge.

Finally, the probabilistic RankEdge algorithm is biologically feasible, requiring less computational complexity and assuming fewer memory requirements than many other distributed graph algorithms commonly used in computer science. This suggests that a biological algorithm evolved to deal with the constraints of the tropical forest canopy may be useful in other applications, such as in swarm robotics or molecular robots [63, 64, 65, 66]. For such applications, the best algorithm to choose depends on the requirements of the problem. We find evidence that RankEdge is the

best algorithm to achieve relatively high success rate and low path entropy. If agents do not need to adhere to a single path, then the WEIGHTED algorithm performs better, though the length of the path may be long. It appears that turtle ants use an algorithm that finds a single short path [12], as RANKEDGE provides. For trivial graphs with only one alternative path, both algorithms perform similarly.

Overall, our work contributes to the growing intersection of distributed algorithms used by natural biological processes [70, 11].

Methods

Maximum likelihood estimation of parameter values

For each candidate algorithm, we varied $q_{\text{decay}}, q_{\text{explore}} \in (0, 1)$ and evaluated the likelihood that the algorithm with a specific set of parameter values would have generated the choices made by turtle ants observed in the field. The edges traversed by the turtle ants, and times the edges were traversed, were used to compute how much pheromone had been added to and had decayed from each edge, to give the amount of pheromone on each edge at any time. In modeling pheromone decay, we treat q_{decay} as the rate of decay per second. When computing the amount of decay between two consecutive ant choices, we decay all of the edges in proportion to the number of seconds elapsed between the two choices. For each candidate algorithm, if we know all of the edge weights at a given time and the value of q_{explore}, we can compute the likelihood of a given choice. Figure 2A–B provides an example of a calculation of the likelihood of a choice for each candidate algorithm.

For a given combination of $q_{\text{decay}}, q_{\text{explore}}$ we performed this likelihood computation for every observed choice in each of the 13 junctions. We updated the edge weights based on the choice and the amount of time that passed between successive choices, and then repeated this process on the next choice made by the next ant. For each junction of observations at a given node on a given day, we computed the maximum likelihood estimate (MLE) for each parameter value pair. We then added the log-likelihoods for all the 13 junctions.

As we formalize in the Supplement, the exponential rate of pheromone decay means that the most recent ant choices have the largest effect on the current edge weights at a junction. Pheromone added far in the past will have largely decayed and will not contribute much to the current weights. Thus, we do not need an extensive history of the choices to perform accurate modeling. It is not currently possible to measure or manipulate pheromone levels on the branches in the canopy.

Additional technical details

Each algorithm includes the following constraints motivated by field observations:

1. Observations suggest that turtle ants tend not to backtrack, but instead tend to keep moving along the trail in the same direction, indicating that turtle ants have at least enough sense of direction to avoid going back and forth over the same edge. Our simulations include three exceptions to this. First, because our simulations include two nests with ants going back and forth, upon reaching the nest, an ant is allowed to backtrack along the same edge it used to reach the nest. Second, if a simulated ant reaches a dead-end node that has no outgoing edges other than the previously traversed edge, it is allowed to backtrack. However, the ant does not lay pheromone on the way back until it reaches a node with two edges, excluding the edge it previously traversed. In field experiments, it is difficult to determine whether a turtle ant is laying pheromone; however, it is known that *Lasius niger* ants down-regulate pheromone deposition at dead-ends to avoid recruitment during crowding [71, 69, 69]. It is possible that turtle ants similarly down-regulate pheromone in response to dead-ends. In the Supplement, we also provide a probabilistic argument for why it is critical that ants do not lay pheromone when returning from a dead-end to repair the break. Thus, in addition to the ability to avoid backtracking, each ant requires one binary state variable that is 0 or 1 depending on whether the ant is coming back from a dead end.

2. Turtle ants queue at a node and leave in a first-in first-out manner. In other words, if more than one ant is at the same node, only one ant chooses an edge in each time-step. In the field, turtle ants walk along narrow branches, almost always one ant at a time in each direction. We find that queueing increases the success rate of the algorithm (Supplement).

3. When turtle ants take an "explore step", they often traverse an edge for a short distance, and then return to the original node [12]. This builds a slight extension off the primary path, which can be extended by subsequent ants. In all algorithms, if a simulated ant takes an explore step, it goes across the edge and comes back in one time-step. Thus, two units of pheromone are left on the edge, and the ant is back at the node it started from. (R3-27) An explore step is defined as any choice that cannot occur unless $q_{\text{explore}} > 0$. For WEIGHTED, this involves taking an edge with zero weight; for RANKEDGE, this involves taking an edge that does not have the highest weight. For UNWEIGHTED, there is no explore step, because ants are not inherently biased towards following any particular pheromone trail.

4. Because pheromone decays exponentially, theoretically once an ant lays pheromone on an edge, that edge's weight will never decay to absolute 0. In practice, if the edge weight stays unchanged even after multiplying by the decay rate (due to numerical computation error, occurring at roughly 10^{-300}), then we reset the weight of that edge to absolute 0. Another possible approach would be to introduce a pheromone detection threshold parameter. If an edge had pheromone below this threshold, the ant would treat the edge as if it had no pheromone. We avoided this approach because it would introduce another parameter to optimize and compare.

5. All edges are assumed to have the same length, and it takes exactly one time step for an ant to cross an edge.

Performance metrics

Below we formally describe the three performance metrics used to evaluate each algorithm, after it ran for T time-steps. Intuitively, the simulated ants have successfully found a path if they can reach one nest from the other without taking any "explore steps". We thus measure the performance of each algorithm assuming $q_{\text{explore}} = 0$, and consider all paths that may be taken with positive probability under this constraint. For RANKEDGE and other non-linear algorithms, ants travel from one nest to the other by following the highest-weighted edges. For WEIGHTED, ants travel from one nest to the other using only edges of positive weight.

Formally, let the *pheromone subgraph* be the subgraph induced by the two nest nodes, all edges with a nonzero weight, and all nodes adjacent to edges with non-zero weight. Let G be a pheromone subgraph and $P = (v_1, v_2, \ldots v_n)$ be a path in G, with nests v_1 and v_n. For a node $v_{i\neq 1} \in P$, define the candidate edges $C_P(v_i) = \{(v_i, u) \in E(G) : u \neq v_{i-1}\}$, i.e., the edges that the ant could take from v_i without backtracking to its previous node. Let $v_i, v_{i+1} \in P$ be consecutive nodes in the path; we say edge (v_i, v_{i+1}) is *maximal with respect to* P if $w(v_i, v_{i+1}) = \max_{u \in C_P(v_i)} w(v_i, u)$. The path P is a *maximal path* if for every pair of consecutive nodes $v_i, v_{i+1} \in P$, the edge (v_i, v_{i+1}) is maximal with respect to P. An ant taking a maximal path always takes an edge with the highest weight; thus, a maximal path allows an ant using one of the non-linear algorithms to commute between two nests with positive probability even if $q_{\text{explore}} = 0$.

Next, define a *pheromone path* to be a path in which all edges have positive weight. Such a path allows an ant following the WEIGHTED algorithm to commute between two nests with positive probability even when $q_{\text{explore}} = 0$.

For a given algorithm, define a *solution path* to be a path that can be traversed with positive probability under that algorithm when $q_{\text{explore}} = 0$. For all the non-linear algorithms discussed here, solution paths are equivalent to maximal paths. For the WEIGHTED algorithm, a solution path is equivalent to a pheromone path.

Let $\hat{p} = (p_1, p_2, \dots), \sum_i p_i = 1$ be a probability distribution. Define the *entropy* of the distribution to be: $S(\hat{p}) = -\sum_i p_i \log(p_i)$.

At the end of the simulation, we evaluate the pheromone subgraph of each algorithm by computing the following measures:

- **Success rate (higher is better):** The probability that the ants form a solution path. This is defined empirically by computing the percentage of the N simulations where a solution path is formed in the final graph.
- **Path entropy (lower is better):** An information-theoretic measure of how well the ants converge onto a single solution path.
  - Let $M_1, M_2, \dots, M_n$ be the set of all n solution paths in the final graph.
  - Let $p_1, p_2, \dots, p_n$ be the probabilities of taking each solution path with $q_{\text{explore}} = 0$. The probabilities $\hat{p} = (p_1, p_2, \dots, p_n)$ form a probability distribution.
  - The path entropy is then: $S(\hat{p})$.
- **Average path length (lower is better):** The average length of the solution paths in the final graph. Path length is defined to be the number of nodes in a path.

To compute the pruning metrics, we first define a *chosen path* as the sequence of nodes $v_1, v_2, \dots, v_n$, after removing cycles, that an ant takes to successfully walk from one nest to another. Figure S1 illustrates why removing cycles is necessary when comparing chosen paths.

Over the course of the simulation, we track all chosen paths for all ants that successfully walk from one nest to the other. This includes the number of times each path was chosen — and thus the distribution over the chosen paths — and the lengths of these paths.

- **Path elimination:** An information-theoretic measure of the degree to which paths from one nest to the other are eliminated over time.
  - Let $S_t = S(\hat{p}_t)$ be the entropy over the distribution of chosen paths $\hat{p}$ that have been completed at or before time t.
  - Let $S_{max} = \max_{1 \le t \le T} S_t$ be the maximum chosen-path entropy over the entire simulation.
  - The path elimination is then the maximum entropy minus the entropy at the end of the simulation: $S_{max} - S_T$.
- **Path length pruning**: A measure of the degree to which ants reduce the lengths of the paths they take over time.

- Suppose at time t the ants have taken chosen paths $p_1, p_2, \ldots, p_n$ with frequencies $c_1, c_2, \ldots, c_n$. Let $l(p_i)$ be the length of path p_i. We define the *weighted-mean chosen path length* at time t to be: $L_t = \dfrac{\sum_i c_i \cdot l(p_i)}{\sum_i c_i}$.
- Let $L_{max} = \max_{1 \leq t \leq T} L_t$ be the maximum weighted mean chosen path length over the entire simulation.
- The path length pruning is then: $L_{max} - L_T$.

Robustness across network topologies. To determine which parameter values performed well across all the planar topologies tested, we defined the *robustness* of a set of parameter values $(q_{\text{explore}}, q_{\text{decay}})$ to be the geometric mean of the success rates for those parameter values on all six networks. We use the geometric mean because it penalizes parameter values that perform poorly on any one particular graph; for a set of parameter values to have a high geometric mean, it must perform well on every graph. When computing robustness, we weight the success rates over all networks equally. This is done for two reasons: first, this highlights algorithms that perform well under a variety of distinct but equal conditions; and second, we do not currently have complete data on which topologies are more or less likely to occur in the canopy, and thus it is not clear how weighting factors should be selected.

Application to the European road network. We sampled a portion of the European road network. This sample contained the same number of nodes as the Full grid ($11 \times 11 = 121$ nodes). Sampling was done by selecting a random node and performing a breadth-first search until 121 nodes were visited. The network contained these 121 nodes and all the edges adjacent to these nodes. We then randomly selected two nodes and removed a randomly-chosen edge in the shortest path between those nodes. If removing this edge disconnected the two nodes, we discarded the pair of nodes and picked a new randomly chosen pair of nodes. We then applied our algorithm to repair the trail.

Algorithm	q_{explore}	q_{decay}	log-likelihood
Unweighted	N/A	N/A	-744.25
Weighted	0.05 ± 0.06	0.01 ± 0.10	-345.05
RankEdge	0.20 ± 0.04	0.02 ± 0.08	-405.42
MaxEdgeA	0.20 ± 0.04	0.02 ± 0.08	-402.63
MaxEdgeB	0.42 ± 0.04	0.01 ± 0.09	-424.61
MaxEdgeC	0.23 ± 0.01	0.02 ± 0.09	-586.87
MaxWeighted	0.22 ± 0.01	0.01 ± 0.09	-568.70
Deneubourg	0.78 ± 0.10	0.01 ± 0.03	-518.18

Table 1: **Maximum likelihood estimates for each algorithm.** For each algorithm, we show the values of q_{explore} and q_{decay} that maximize the likelihood of producing the observed choices made by turtle ants in the field. Standard deviation is computed across 13 junctions, each corresponding to field observations of one junction on a given day. All models are significantly more likely to explain the data than the null model (Unweighted).

Algorithm	Minimal	Simple	Medium	Full	Spanning Grid	Europe	Robustness
Unweighted	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Weighted	1.00	1.00	1.00	1.00	1.00	1.00	1.00
RankEdge	1.00	0.98	0.84	0.70	0.54	0.70	0.78
MaxEdgeA	1.00	0.78	0.74	0.48	0.36	0.62	0.63
MaxEdgeB	0.76	0.60	0.64	0.64	0.44	0.70	0.62
MaxEdgeC	1.00	1.00	0.84	0.48	0.44	0.48	0.66
MaxWeighted	1.00	0.88	0.72	0.63	0.38	0.68	0.68
Deneubourg	1.00	1.00	0.94	0.44	0.67	0.52	0.72

Table 2: **Success rates for each algorithm.** For each algorithm, we show the success rate on each simulated network. The last column summarizes the robustness of each method across all networks. Of the non-linear algorithms, RankEdge performs the best.

Algorithm	Minimal	Simple	Medium	Full	Spanning Grid
Unweighted	0.00	0.69	1.79	13.75	7.24
Weighted	0.00 ± 0.00	0.23 ± 0.25	1.52 ± 0.20	12.38 ± 0.22	5.94 ± 0.15
RankEdge	0.00	0.00	0.00	0.00	0.00
MaxEdgeA	0.00	0.00	0.00	0.00	0.00
MaxEdgeB	0.00	0.00	0.00	0.00	0.00
MaxEdgeC	0.00	0.00	0.00	0.00	0.00
MaxWeighted	0.00	0.00	0.00	0.00	0.00
Deneubourg	0.00	0.00	0.00	0.00	0.00

Table 3: **Path entropy for each algorithm.** For each algorithm, we show the path entropy on each simulated network. Lower values indicate convergence to fewer paths. All non-linear algorithms achieve the optimal path entropy. The standard deviation of the entropy for all non-linear models is 0. We do not report an interval for Unweighted because it does not depend on pheromone amount and thus has the same limiting behavior in all cases.

Algorithm	Minimal	Simple	Medium	Full Grid	Spanning Grid
Unweighted	12.00 ± 0.00	12.90 ± 1.00	13.16 ± 3.00	26.04 ± 6.56	20.05 ± 7.24
Weighted	12.00 ± 0.00	12.97 ± 0.85	13.04 ± 0.64	18.36 ± 0.18	16.93 ± 0.24
RankEdge	12.00 ± 0.00	12.98 ± 1.01	10.95 ± 1.01	13.06 ± 0.34	14.56 ± 3.97
MaxEdgeA	12.00 ± 0.00	12.87 ± 1.00	11.29 ± 0.97	13.17 ± 0.56	15.56 ± 4.38
MaxEdgeB	12.00 ± 0.00	13.30 ± 0.96	10.77 ± 0.99	13.06 ± 0.35	15.09 ± 4.07
MaxEdgeC	12.00 ± 0.00	13.02 ± 1.01	11.00 ± 1.01	13.16 ± 0.56	15.45 ± 3.54
MaxWeighted	12.00 ± 0.00	12.87 ± 0.99	10.89 ± 1.03	13.06 ± 0.58	14.46 ± 3.07
Deneubourg	12.00 ± 0.00	13.16 ± 1.00	11.06 ± 1.30	14.18 ± 3.47	13.94 ± 2.51
Optimal	12.00	12.00	10.00	13.00	13.00

Table 4: **Average path length for each algorithm.** For each algorithm, we show the average path length of the final graph, measured as the number of nodes in the path. RankEdge performed much better than the null model (Unweighted) and close to the globally shortest path length (Optimal).

Algorithm	Simple	Medium	Full Grid	Spanning Grid	European Roads
Weighted	0.94 ± 0.58	0.48 ± 0.32	0.00 ± 0.00	0.001 ± 0.002	0.14 ± 0.30
RankEdge	0.18 ± 0.26	0.25 ± 0.29	0.14 ± 0.20	0.15 ± 0.18	0.10 ± 0.16
MaxEdgeA	0.54 ± 0.84	0.69 ± 1.14	0.26 ± 0.49	0.78 ± 1.37	0.01 ± 0.01
MaxEdgeB	0.25 ± 0.61	0.66 ± 1.06	0.29 ± 0.56	0.84 ± 1.73	1.18 ± 2.44
MaxEdgeC	0.40 ± 0.78	1.29 ± 1.40	0.48 ± 0.92	1.03 ± 1.58	0.04 ± 0.07
MaxWeighted	0.44 ± 0.78	1.04 ± 1.23	0.68 ± 1.43	1.36 ± 2.93	0.38 ± 0.46
Deneubourg	0.43 ± 0.56	1.60 ± 0.96	0.55 ± 1.45	1.40 ± 2.36	1.47 ± 2.95

Table 5: **Path elimination**: For each algorithm, we show the average reduction in entropy over chosen paths over time (Methods). We omit the Minimal graph, because there is only one possible path from one nest to the other, and thus no path elimination is possible.

Algorithm	Simple	Medium	Full Grid	Spanning Grid	European Roads
Weighted	0.94 ± 0.58	0.48 ± 0.32	0.00 ± 0.00	0.001 ± 0.002	0.14 ± 0.30
RankEdge	0.30 ± 0.58	0.39 ± 0.58	0.04 ± 0.14	0.97 ± 2.59	0.35 ± 0.50
MaxEdgeA	0.30 ± 0.27	0.28 ± 0.30	0.11 ± 0.17	0.15 ± 0.20	0.05 ± 0.10
MaxEdgeB	0.14 ± 0.23	0.15 ± 0.23	0.06 ± 0.10	0.11 ± 0.16	0.08 ± 0.14
MaxEdgeC	0.31 ± 0.29	0.50 ± 0.36	0.12 ± 0.17	0.13 ± 0.21	0.08 ± 0.13
MaxWeighted	0.29 ± 0.30	0.39 ± 0.33	0.15 ± 0.23	0.19 ± 0.24	0.17 ± 0.23
Deneubourg	0.46 ± 0.17	0.42 ± 0.28	0.001 ± 0.003	0.04 ± 0.09	0.02 ± 0.09

Table 6: **Path length pruning**: For each algorithm we show the average reduction in lengths of chosen paths over time (Methods) observed. We omit the Minimal graph, because there is only one possible path from one nest to the other, and thus no pruning is possible.

Figure Legends

Figure 1: **Turtle ant habitat and trail network.** A) The photograph shows the highly tangled forest canopy in which turtle ants forage. B) Experiments were performed in which an edge in the path was cut, to observe how the ants respond and repair the break [12]. C) Modeling the trail network as a graph, with junctions as nodes and connecting branches and twigs as edges. The diagram on the right from [12] shows a detailed depiction of a large portion of the trail network. Each days path is shown in a different color (see legend), and additional repair paths are shown in a distinct color. Solid lines connect two nodes that are on the same plant (e.g. node 36 and node A are on the same plant). Dashed lines connect two nodes that are on a different plant (e.g. nodes B and C are on different plants).

Figure 2: **Maximum likelihood computation.** A–B) Example node junction and edge choices for turtle ants. All ants arrive at node 1 from a different node that is not shown. In the example, we assume pheromone has been deposited at previous time-points, and we now compute the likelihood of the next ant choice. Under the RankEdge algorithm, the likelihood of choosing edges $1 \to 3$ or $1 \to 2$ is $(1 - q_{\text{explore}})(1/2)$; the likelihood of edge $1 \to 4$ is $q_{\text{explore}}(1 - q_{\text{explore}})$; and the likelihood of $1 \to 5$ is q_{explore}^2. Under the Weighted algorithm, the likelihood of choosing edge $1 \to 5$ is q_{explore}; the likelihood of edge $1 \to 4$ is $(1 - q_{\text{explore}})(1/(1 + 2 + 2))$; and the likelihood of edges $1 \to 2$ or $1 \to 3$ is $(1 - q_{\text{explore}})(2/(1 + 2 + 2)$. Under the Unweighted algorithm, the edge weights are disregarded, and the likelihood of taking any one of the four edges is $(1/4)$. C–D) For each combination of q_{explore} (x-axis) and q_{decay} (y-axis) values, we determined the pair's likelihood of producing the choices observed in turtle ants. Each heatmap shows the likelihood for each algorithm with a zoom-in below around the highest likelihood region. The optimal parameter values for each algorithm, depicted in white, are shown in Table 1.

Figure 3: **The maximum likelihood parameters closely match the best simulation parameters:** A) The color of each square in the heatmap corresponds to the robustness (Methods) of the success rates for the RankEdge algorithm for each combination of q_{explore} (x-axis) and q_{decay} (y-axis) values. Results are aggregated over the six simulated and real-world networks presented in Figures 4 and 5. B) The maximum likelihood parameter estimates for RankEdge from observations of turtle ants. The black rectangle in both panels shows that the parameter values that best explain the turtle ants' behavior also perform best for solving the network repair problem.

Figure 4: **Success rates for each network.** A–E) For each network we show the initial graph (left), an example of the final graph after running the RankEdge algorithm using the maximum likelihood parameters (middle), and the algorithm's success rate for each parameter combination (right). In each panel, black dots indicate nodes in the network, and solid lines indicate edges that may be traversed. If two adjacent nodes are not connected by an edge, there is a space between them. In the initial graphs, the 'X' marks the edge that is broken. The x-axis of the heatmap (right column) shows q_{explore}, and the y-axis shows q_{decay} under the range close to the MLE parameters. Darker shades of red are indicate success rates closer to 1, and thus are better.

Figure 5: **Repairing road closures in the Europe road graph.** Analysis of how well the turtle ant algorithm translates to repair simulated breaks in a real-world transport network. A) An example of a path in the European E-road network connecting Munich to Berlin, Germany. The roads and junctions form a graph. On the left, the black 'X' shows a road that has been broken or closed along the path. On the right, we show an alternative path that avoids the broken road. B) The success rate of the turtle ant algorithm (RankEdge) applied to this network. Map data: Google, DigitalGlobe.

Figure 6: **Poor path entropy for Weighted.** The initial (left) and final (right) networks for the (A) Full grid and (B) Spanning grid. In both cases, the MLE parameter values ($q_{\text{explore}} = 0.05, q_{\text{decay}} = 0.01$) for Weighted did not find a low path entropy solution.

Figure 7: **Analysis in the absence of a break.** A) Initial Spanning grid, with no break. B) The final network produced using Weighted, which does not find a low entropy solution. C) The final graph using RankEdge, which finds a low path entropy solution.

Figure 8: **Turtle ants prune paths.** The diagram from [12] shows the results of an experiment in which an edge was cut. Left: The initial trail is shown in grey. The edge connecting nodes 5 and 6 was cut. After 75 minutes, the turtle ants explored several new paths (red). Center: Five hours after the cut, some of the red paths were pruned (transparent grey). Ants traveling down from node 7 took one trail, consisting of nodes 6, 15, 16, 17, 18, 4, and 3, because they could not use 12 in this direction. Ants traveling in the other direction took another trail, consisting of nodes 3, 4, 5, 12, 13, and 14, or the trail consisting of 11, 13, and 14. Right: The next day, there was additional pruning. Because node 12 could now be used in both directions, ants traveled both ways on the indicated trail.

Acknowledgments

The authors thank Jason Schweinsberg for guiding us in the theoretical proof. We are grateful to Illia Ziamtsov, Javier How, Benjamin Cosman, Ailie Fraser, Will Hamilton, Sam Crow, Alex Lang, and the anonymous reviewers for helpful comments on the manuscript.

Source code and datasets

All source code for the algorithm and all datasets for the ant choices are available at our Github repository: `http://github.com/arjunc12/Ants`.

References

[1] N. A. Lynch. *Distributed Algorithms*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1996.

[2] A. Tero, S. Takagi, T. Saigusa, K. Ito, D. P. Bebber, M. D. Fricker, K. Yumiki, R. Kobayashi, and T. Nakagaki. Rules for biologically inspired adaptive network design. *Science*, 327(5964):439–442, Jan 2010.

[3] Saket Navlakha, Alison L. Barth, and Ziv Bar-Joseph. Plos computational biology: Decreasing-rate pruning optimizes the construction of efficient and robust distributed networks. *PLoS ONE*, July 2015. (Accessed on 07/13/2016).

[4] T. Latty, K. Ramsch, K. Ito, T. Nakagaki, D. J. Sumpter, M. Middendorf, and M. Beekman. Structure and formation of ant transportation networks. *J R Soc Interface*, 8(62):1298–1306, Sep 2011.

[5] Anthony Brabazon, Michael O'Neill, and Sen McGarraghy. *Natural Computing Algorithms (Natural Computing Series)*. Springer, 2015.

[6] D. M. Gordon. The dynamics of foraging trails in the tropical arboreal ant Cephalotes goniodontus. *PLoS ONE*, 7(11):e50472, 2012.

[7] A. Bottinelli, E. van Wilgenburg, D. J. Sumpter, and T. Latty. Local cost minimization in ant transport networks: from small-scale data to large-scale trade-offs. *J R Soc Interface*, 12(112), Nov 2015.

[8] Michele C Lanan, Anna Dornhaus, and Judith L Bronstein. The function of polydomy: the ant crematogaster torosa preferentially forms new nests near food sources and fortifies outstations. *Behavioral Ecology and Sociobiology*, 65(5):959–968, 2011.

[9] Mark Newman. *Networks: An Introduction*. Oxford University Press, Inc., New York, NY, USA, 2010.

[10] Thomas H.. Cormen, Charles Eric Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*, volume 6. MIT press Cambridge, 2001.

[11] D. M. Gordon. The evolution of the algorithms for collective behavior. *Cell Syst*, 3(6):514–520, Dec 2016.

[12] D. M. Gordon. Local Regulation of Trail Networks of the Arboreal Turtle Ant, Cephalotes goniodontus. *Am. Nat.*, 190(6):E156–E169, 12 2017.

[13] Crispín Gomez, Francisco Gilabert, María Engracia Gomez, Pedro López, and José Duato. Deterministic versus adaptive routing in fat-trees. In *Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International*, pages 1–8. IEEE, 2007.

[14] Eliza JT Middleton and Tanya Latty. Resilience in social insect infrastructure systems. *Journal of The Royal Society Interface*, 13(116):20151022, 2016.

[15] Miriam Malíčková, Christian Yates, and Katarína Bodová. A stochastic model of ant trail following with two pheromones. *arXiv:1508.06816*, 2015.

[16] Tatiana P Flanagan, Noa M Pinter-Wollman, Melanie E Moses, and Deborah M Gordon. Fast and flexible: Argentine ants recruit from nearby trails. *PloS one*, 8(8):e70888, 2013.

[17] Simon Garnier, Aurélie Guérécheau, Maud Combe, Vincent Fourcassié, and Guy Theraulaz. Path selection and foraging efficiency in argentine ant transport networks. *Behavioral Ecology and Sociobiology*, 63(8):1167–1179, 2009.

[18] Audrey Dussutour, Vincent Fourcassie, Dirk Helbing, and Jean-Louis Deneubourg. Optimal traffic organization in ants under crowded conditions. *Nature*, 428(6978):70–73, 2004.

[19] Jean-Louis Deneubourg, Simon Goss, Nigel Franks, and JM Pasteels. The blind leading the blind: modeling chemically mediated army ant raid patterns. *Journal of insect behavior*, 2(5):719–725, 1989.

[20] Daniel Cherix, Ph Werner, F Catzeflis, et al. Spatial organisation of a polycalic system in formica (coptoformica) exsecta nyl.(hymenoptera: Formicidae). *Mitteilungen der Schweizerischen Entomologischen Gesellschaft*, 53(2/3):163–172, 1980.

[21] Jean L Deneubourg, Serge Aron, SAPJM Goss, JM Pasteels, and Guido Duerinck. Random behaviour, amplification processes and number of participants: how they contribute to the foraging properties of ants. *Physica D: Nonlinear Phenomena*, 22(1):176–186, 1986.

[22] Nigel R Franks. Army ants: a collective intelligence. *American Scientist*, 77:138–145, 1989.

[23] Chris R Reid, Matthew J Lutz, Scott Powell, Albert B Kao, Iain D Couzin, and Simon Garnier. Army ants dynamically adjust living bridges in response to a cost–benefit trade-off. *Proceedings of the National Academy of Sciences*, 112(49):15113–15118, 2015.

[24] Duncan Jackson, Mike Holcombe, and Francis Ratnieks. Coupled computational simulation and empirical research into the foraging system of pharaohs ant (monomorium pharaonis). *Biosystems*, 76(1):101–112, 2004.

[25] Duncan E. Jackson, Stephen J. Martin, Mike Holcombe, and Francis L.W. Ratnieks. Longevity and detection of persistent foraging trails in pharaoh's ants, monomorium pharaonis (l.). *Animal Behaviour*, 71(2):351 – 359, 2006.

[26] Elva JH Robinson, Duncan E Jackson, Mike Holcombe, and Francis LW Ratnieks. Insect communication:no entrysignal in ant foraging. *Nature*, 438(7067):442–442, 2005.

[27] E JH Robinson, KE Green, EA Jenner, M Holcombe, and F LW Ratnieks. Decay rates of attractive and repellent pheromones in an ant foraging trail network. *Insectes sociaux*, 55(3):246–251, 2008.

[28] Elva JH Robinson, Francis LW Ratnieks, and M Holcombe. An agent-based model to investigate the roles of attractive and repellent pheromones in ant decision making during foraging. *Journal of Theoretical Biology*, 255(2):250–258, 2008.

[29] Alberto Colorni, Marco Dorigo, Vittorio Maniezzo, et al. Distributed optimization by ant colonies. In *Proceedings of the first European conference on artificial life*, volume 142, pages 134–142. Paris, France, 1991.

[30] M. Dorigo and C. Blum. Ant colony optimization theory: A survey. *Theoretical Computer Science*, 344(2-3):243–278, nov 2005.

[31] Manuel López-Ibáñez, Thomas Stützle, and Marco Dorigo. *Ant Colony Optimization: A Component-Wise Overview*, pages 1–37. Springer International Publishing, Cham, 2016.

[32] Luca Maria Gambardella, Roberto Montemanni, and Dennis Weyland. Coupling ant colony systems with strong local searches. *European Journal of Operational Research*, 220(3):831–843, 2012.

[33] Shigeyoshi Tsutsui. Ant colony optimization with cunning ants. *Transactions of the Japanese Society for Artificial Intelligence*, 22:29–36, 2007.

[34] Wolfram Wiesemann and Thomas Stützle. Iterated ants: An experimental study for the quadratic assignment problem. In *International Workshop on Ant Colony Optimization and Swarm Intelligence*, pages 179–190. Springer, 2006.

[35] Pierre Fraigniaud, David Ilcinkas, Guy Peer, Andrzej Pelc, and David Peleg. Graph exploration by a finite automaton. *Theoretical Computer Science*, 345(2):331–344, 2005.

[36] Nicolas Hanusse, Dimitris Kavvadias, Evangelos Kranakis, and Danny Krizanc. Memoryless search algorithms in a network with faulty advice. *Theoretical Computer Science*, 402(2):190–198, 2008.

[37] Israel A Wagner, Michael Lindenbaum, and Alfred M Bruckstein. Efficiently searching a graph by a smell-oriented vertex process. *Annals of Mathematics and Artificial Intelligence*, 24(1-4):211–223, 1998.

[38] O. Feinerman, A. Korman, Z. Lotker, and J.-S. Sereni. Collaborative search on the plane without communication. In *Proceedings of the 2012 ACM Symposium on Principles of Distributed Computing*, PODC '12, pages 77–86, New York, NY, USA, 2012. ACM.

[39] Y. Emek, T. Langner, D. Stolz, J. Uitto, and R. Wattenhofer. Towards More Realistic ANTS. In *2nd Workshop on Biological Distributed Algorithms (BDA)*, October 2014.

[40] C. Lenzen and T. Radeva. The power of pheromones in ant foraging. In *1st Workshop on Biological Distributed Algorithms (BDA)*, 2013.

[41] J. Kleinberg and E. Tardos. *Algorithm Design*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005.

[42] K Mani Chandy and Jayadev Misra. Distributed computation on graphs: Shortest path algorithms. *Communications of the ACM*, 25(11):833–837, 1982.

[43] Pierre A Humblet et al. Another adaptive distributed shortest path algorithm. *IEEE transactions on communications*, 39(6):995–1003, 1991.

[44] Radia Perlman. An algorithm for distributed computation of a spanningtree in an extended lan. In *ACM SIGCOMM Computer Communication Review*, volume 15, pages 44–53. ACM, 1985.

[45] Juan A Garay, Shay Kutten, and David Peleg. A sublinear time distributed algorithm for minimum-weight spanning trees. *SIAM Journal on Computing*, 27(1):302–316, 1998.

[46] Jukka Suomela. Survey of local algorithms. *ACM Computing Surveys (CSUR)*, 45(2):24, 2013.

[47] Yehuda Afek, Noga Alon, Ziv Bar-Joseph, Alejandro Cornejo, Bernhard Haeupler, and Fabian Kuhn. Beeping a maximal independent set. *Distributed computing*, 26(4):195–208, 2013.

[48] F. Merkl and S.W.W. Rolles. Linearly edge-reinforced random walks. In *Institute of Mathematical Statistics Lecture Notes - Monograph Series*, pages 66–77. Institute of Mathematical Statistics, 2006.

[49] Persi Diaconis and David Freedman. de finetti's theorem for markov chains. *The Annals of Probability*, pages 115–130, 1980.

[50] Burgess Davis. Reinforced random walk. *Probability Theory and Related Fields*, 84(2):203–229, Jun 1990.

[51] Angela Stevens and Hans G Othmer. Aggregation, blowup, and collapse: the abc's of taxis in reinforced random walks. *SIAM Journal on Applied Mathematics*, 57(4):1044–1081, 1997.

[52] S Aron, Jacques M. Pasteels, and Jean Louis Deneubourg. Trail-laying behaviour during exploratory recruitment in the argentine ant, Iridomyrmex humilis (Mayr). *Biology of Behaviour*, 14:207–217, 1989.

[53] N. Pinter-Wollman, R. Wollman, A. Guetz, S. Holmes, and D. M. Gordon. The effect of individual variation on the structure and function of interaction networks in harvester ants. *J R Soc Interface*, 8(64):1562–1573, Nov 2011.

[54] Danielle P Mersch, Alessandro Crespi, and Laurent Keller. Tracking individuals shows spatial fidelity is a key regulator of ant social organization. *Science*, 340(6136):1090–1093, 2013.

[55] Deborah M Gordon. The expandable network of ant exploration. *Animal Behaviour*, 50(4):995–1007, 1995.

[56] Raphaël Jeanson, Francis LW Ratnieks, and Jean-Louis Deneubourg. Pheromone trail decay rates on different substrates in the pharaoh's ant, monomorium pharaonis. *Physiological Entomology*, 28(3):192–198, 2003.

[57] T Simon and A Hefetz. Trail-following responses oftapinoma simrothi (formicidae: Dolichoderinae) to pygidial gland extracts. *Insectes Sociaux*, 38(1):17–25, 1991.

[58] Andrea Perna, Boris Granovskiy, Simon Garnier, Stamatios C Nicolis, Marjorie Labédan, Guy Theraulaz, Vincent Fourcassié, and David JT Sumpter. Individual rules for trail pattern formation in argentine ants (linepithema humile). *PLoS computational biology*, 8(7):e1002592, 2012.

[59] Jean-Louis Deneubourg, Jacques M Pasteels, and Jean-Claude Verhaeghe. Probabilistic behaviour in ants: a strategy of errors? *Journal of Theoretical Biology*, 105(2):259–271, 1983.

[60] Ehud Fonio, Yael Heyman, Lucas Boczkowski, Aviram Gelblum, Adrian Kosowski, Amos Korman, and Ofer Feinerman. A locally-blazed ant trail achieves efficient collective navigation despite limited information. *eLife*, 5:e20185, 2016.

[61] David JT Sumpter and Madeleine Beekman. From nonlinearity to optimality: pheromone trail foraging by ants. *Animal behaviour*, 66(2):273–280, 2003.

[62] Jrme Kunegis. KONECT – The Koblenz Network Collection. In *Proc. Int. Conf. on World Wide Web Companion*, pages 1343–1350, 2013.

[63] K. Lund, A. J. Manzo, N. Dabby, N. Michelotti, A. Johnson-Buck, J. Nangreave, S. Taylor, R. Pei, M. N. Stojanovic, N. G. Walter, E. Winfree, and H. Yan. Molecular robots guided by prescriptive landscapes. *Nature*, 465(7295):206–210, May 2010.

[64] M. Brambilla, E. Ferrante, M. Birattari, and M. Dorigo. Swarm robotics: a review from the swarm engineering perspective. *Swarm Intell*, 7(1):1–41, jan 2013.

[65] J. Werfel, K. Petersen, and R. Nagpal. Designing collective behavior in a termite-inspired robot construction team. *Science*, 343(6172):754–758, Feb 2014.

[66] Joshua P Hecker and Melanie E Moses. Beyond pheromones: evolving error-tolerant, flexible, and scalable ant-inspired robot swarms. *Swarm Intelligence*, 9(1):43–70, 2015.

[67] C. Korn and H. G. Augustin. Mechanisms of Vessel Pruning and Regression. *Dev. Cell*, 34(1):5–17, Jul 2015.

[68] Balaji Prabhakar, Katherine N Dektar, and Deborah M Gordon. The regulation of ant colony foraging activity without spatial information. *PLoS Comput Biol*, 8(8):e1002670, 2012.

[69] Tomer J Czaczkes, Christoph Grüter, and Francis LW Ratnieks. Trail pheromones: an integrative view of their role in social insect colony organization. *Annual review of entomology*, 60:581–599, 2015.

[70] Saket Navlakha and Ziv Bar-Joseph. Distributed information processing in biological and computational systems. *Commun. ACM*, 58(1):94–102, December 2014.

[71] Tomer J Czaczkes, Christoph Grüter, and Francis LW Ratnieks. Negative feedback in ants: crowding results in less trail pheromone deposition. *Journal of the Royal Society Interface*, 10(81):20121009, 2013.

Author Contributions

AC and SN performed the computational experiments. DMG performed the field experiments. All authors wrote and reviewed the manuscript.

Competing Interests

The authors declare no competing interests.