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 together. 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, to 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 the algorithm succeeds under certain conditions. The turtle ant 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 a centralized controller [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-experimental analysis 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 several nests, located in dead tree branches, that are connected to each other in a circuit or network routing backbone [4,

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

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 describes 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. This is an important goal in computer routing networks, where convergence to a single path ensures in-order delivery of data packets [12]. 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, 13] 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 [13].

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 individual 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 networks in response to experimentally-induced edge breaks (Figure 1B) [13] (Figure 1B). We ask:

1. What algorithm do turtle ants at a node use to select which edge to traverse next, given only local information about pheromone levels on adjacent edges?

2. How well can the algorithm repair broken trails in simulated breaks in synthetic and real-world

network topologies?

   (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 one time-step of memory, allowing an ant to avoid the node it previously visited, improve algorithm performance relative to performance with no memory?

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. What is the theoretical basis for why the network repair algorithm succeeds in particular conditions, and what provable guarantees can we provide?

5. How well do other predictions from the algorithm match observations from field studies?

   (a) Do turtle ants form multiple alternative paths and then prune some of them over time?
   (b) How well do the time scales of repairing trails in simulation match time scales observed in field studies?

Related work

To our knowledge, this is the first computational-experimental 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 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]. While experimental work sometimes uses discrete mazes or Y-junctions to impose a graph structure, 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 can not form trails with nodes and edges at arbitrary locations; instead, they can only use the nodes and edges that are available to them.

Further, we assume only a single pheromone type; it appears that other ant species can leave two (positive or negative [24]) or three (trail, dead-end, food [25, 26, 27, 28, 29]) types of pheromone.

Ant colony optimization and reinforcement learning. 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 [30, 31, 32]. 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 [33], cunning ants [34], and iterated ants [35]. ACO, however, provides simulated ants more computational power than turtle 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 [36, 37], such as the Hamiltonian path problem [38] 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 [39], sending constant sized messages [40], or laying pheromone to mark an edge as explored [41]. 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.

Our work also relates to the general field of reinforcement learning. The q_{explore} parameter in our algorithm, which controls how often an ant explores edges with relatively low pheromone, is analogous to the ϵ parameter in popular ϵ-greedy reinforcement learning algorithms [42, 43, 44]. Classical reinforcement learning uses a single agent with no memory limits, whereas our algorithm is distributed and uses agents with limited memory and access only to local information.

Graph algorithms and edge-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 [45]. These abilities are all unlikely for turtle ants.

Distributed graph algorithms, where nodes are treated as fixed agents capable of passing messages to neighbors, have also been proposed to find shortest paths in a graph [46, 47], to construct minimum spanning trees [48, 49], and to approximate various NP-hard problems [50, 51]. 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 invites comparison to a Markov process. Edge-reinforced random walks [52], first introduced by Diaconis and others [53, 54], 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. Edge-reinforced random walks have been used to model the gliding behavior of myxobacteria [55]. We expand upon 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, as ant trail pheromone is volatile and decays [56, 26].

Results

Our goals are to find an algorithm that can both 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 two candidate distributed algorithms for network repair. We parameterize each algorithm using data from field experiments to determine how turtle ants choose which edge to traverse next from a node, given only local information about adjacent edges and their edge weights. Third, we analyze, both via simulation and theoretically, 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 edge, 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. There are N ants that walk on the graph, initialized to be distributed randomly across the nodes on the initial path.

Communication amongst 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 [57], turtle ants lay trail pheromone continuously as they walk [6, 13]. Though this has not been observed directly, we hypothesize that there are certain exceptions in which turtle ants discontinue laying pheromone (Methods). Each ant at a node can sense 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 may have sufficient memory 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 [58, 59, 60], 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, all edges (v_1, v_2) traversed increase their edge weight as:

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

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, 61]; thus $q_{\text{decay}} \in (0, 1)$, and at each time step, for each edge (v_1, v_2), its weight is updated as:

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

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 [13].

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 an alternative path from u to v that excludes the broken edge. 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 the algorithm concentrated pheromone on fewer paths. This measure is computed only in the simulations in which an alternative path was successfully found.

3. Path length: The length of the new path. Although turtle ants do not always find the globally shortest path [13], 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 has a constant amount of memory. It can remember the node it immediately previously visited, and it can remember whether it is returning from a dead end.

- 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.

Two candidate distributed algorithms

Below we introduce two biologically plausible algorithms (Algorithms 1–2) that 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, respectively, used to analyze ant trail formation in other species, such as Argentine ants [62, 63, 64] and pharaoh ants [65]. 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

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: 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 $1 - q_{\text{explore}}$.

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.

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 edge 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_{explore}^2 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: 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 and the ability to traverse and return from an edge on an explore step in one time-step. See Methods for full details.

Null model: We compared these two candidate distributed algorithms 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.

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

Our goal here is to determine which of the algorithms above, and which parameter values, best explain the observed sequence of edge choices made at a node junction by turtle ants.

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, 13]. 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, at 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 and pair 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 datasets (Figure S1). This suggests that across six colonies, there is 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 algorithm

closely matched the edge choices of turtle ants in the field, and it also performed well in simulations for network repair across six networks (Figure 3). The latter result is substantiated below.

Our first performance metric measures how well the ants succeed in finding an alternative path to repair the break (called the *success rate*). We simulated breaks under six planar network structures, each of which have an increasingly complex topology, with more edges, nodes, and possible paths in the graph. 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.

We analyzed both algorithms but only show results for RankEdge in the main text because Weighted rarely converged onto a single path (Figure S5), thus not satisfying our second performance metric; it also did not maintain trails in the absence of a break (Figure S6).

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". Instead 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 below).

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), where ants search the integer plane [39, 40, 41].

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 matches 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 [13].

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 [66] (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.

While global positioning systems often allow for centralized solutions to routing problems, new applications domains, such as for swarm robotics or molecular robots [67, 68, 69, 70] operating in close or remote environments, may necessitate distributed solutions.

Q2a. Converging onto a single consensus path (Figure 6). Our second performance metric measures how well foragers commit to a single alternative path (called the *path entropy*).

We find that the RankEdge algorithm consistently achieves a path entropy near 0, indicating that the final network converges to a single consensus path (Figure 6). 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 criteria. As mentioned above, the Weighted algorithm rarely converged onto a single path (Figure S5).

Q2b. Finding short paths (Table 2). 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. Further, the RankEdge found significantly shorter paths than the Unweighted random walk, which ignores all pheromone (Table 2). This 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 ($q_{\text{explore}} = 0.20, q_{\text{decay}} = 0.02$), 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 break-point on side B to the nest on side B.

Although uni-directional search has rarely been observed to occur in real turtle ant networks, we tested it here on the network repair problem because bi-directional search is often used computationally to improve the performance of search algorithms.

Q2d. The power of one time-step of memory (Figure S4). We find that providing simulated ants a single time-step of memory, compared to zero time-steps of memory, allows for a significant improvement in algorithm performance. The one time-step of memory allowed simulated ants to avoid backtracking, i.e., visiting the same node visited in the previous time-step (Methods). In contrast, with no memory, a simulated ant could keep going back and forth along the same edge.

In particular, one time-step of memory produced a success rate of 70% on the Full grid, compared to 0% with zero time-steps of memory (Figure S4). Thus, providing ants with a basic node-to-node sense of direction led to a significant improvement in performance.

Q3. Maintaining a trail in the absence of a break

Our goal here is to show that the same algorithm used to repair a path can also keep a path intact when it is not broken. This is an important consideration; if different algorithms were used to maintain versus repair the trail, then the turtle ants would need some signal to toggle between different algorithms depending on the context. We show that a single algorithm, RankEdge, is capable of maintaining trails and responding to breaks.

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 S6). 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. The theoretical basis for why the network repair algorithm succeeds

In the Supplement, we provide a probabilistic argument for why the maximum likelihood values for the RankEdge algorithm allow the ants to eventually find and commit to the alternative path on the Minimal graph with probability 1. Critically, to do this in reasonable time, the ants must not lay pheromone on the way back from the dead-end. If the ants do lay pheromone on the way back from the dead-end, then the expected time to put more pheromone on the edge leading to the new path rises dramatically.

This theoretical observation was confirmed by simulations that show that the RankEdge algorithm, using the observed maximum likelihood value of q_{explore}, and with ants not laying pheromone

on the way back from the dead-end, abandons the broken path in favor of the alternative trail with high probability (Figure 4A).

Q5. Matching other predictions from the algorithm with observed behavior of turtle ants

Below, we compare two predictions from our algorithm with data from field studies.

Q5a. Pruning as a general principle for discovering alternative paths. Early in our simulations, we 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 pruning* (Methods). We also quantified how the lengths of the remaining paths changed over time using a measure called *path length pruning* (Methods). We observed pruning towards shorter paths during the network repair process in all simulated networks (Table 3).

When tested experimentally in undisturbed colonies [13], it was found that turtle ants pruned nodes from existing trails from day to day, and never added nodes to existing trails [13]. Such pruning of nodes also led to global pruning of paths (Figure 7). 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 [71]. Thus, pruning may be a common biological strategy of network design when multiple possible topologies need to be explored in a distributed manner.

Q5b: Similarities in local behavior around a break (Figure S2). In our simulations, we observed that ants initially approach the dead-end, and then branch out and commit to a new route directed away from the dead-end. We computed the time it takes for this process to occur in simulation, and compared this with times to re-route from data on turtle ant responses to experimentally ruptured trails [13].

We found a close correspondence between the two (Figure S2), indicating that the algorithm derived here captures observed local behavior around a breakpoint. Simulated ants take slightly longer than turtle ants did. This may be because experiments were conducted on active trails on which there was likely already pheromone on alternative edges, directed away from the broken edge, laid by ants exploring away from the trail. In the simulation, ants found a new route by exploring edges on which there was previously no pheromone. In addition, simulations were made on a graph in which all nodes are alike, while in the vegetation used by turtle ants, node structure varies and the ants often choose nodes that are more quickly reinforced than others (Discussion).

Discussion

We described how turtle ants solve a fundamental computing problem: maintaining a trail network and finding alternative paths to route around broken links in the network. The RankEdge algorithm achieved the best performance trade-off between matching the observed behavior of turtle ants in the field (Figure 2–3) and solving the network repair problem according to our performance metrics, achieving higher success rates (Figure 4), better path entropy (Figure 6), and better ability to keep unbroken trails in tact (Figure S6) compared to Weighted. 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, the algorithm repaired trails on time scales that were similar to those observed in the field (Figure S2), and the algorithm used a pruning process to converge to alternative paths that also mimicked turtle ant responses (Table 3, Figure 7).

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. Second, to repair breaks it is essential to use bi-directional search and allow one time-step of memory. Observations show that when turtle ants encounter a break, ants from both sides of the break attempt to repair the trail [13]. When ants from both sides meet, they each encounter a trail that is already strongly reinforced and guided towards the other nest. In addition, one time-step of memory 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. Real ants encounter diverse situations analogous to breaks in the ongoing maintenance of trails. For example, when a food source is exhausted, there will be few ants on the temporary trail to that food source. Thus, when an ant moves toward the food source, it may encounter a node with no pheromone on any outgoing edge; this may also occur when the rate of flow of ants on a trail is extremely low. We find that a single algorithm can solve two diverse problems without requiring the additional complexity of a signal that distinguishes situations like these from a rupture in the trail.

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 [72, 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 [73]. Ants could use negative pheromone to signal to other ants not to select a certain edge, for example, towards a dead-end.

One disparity between our synthetic networks and the environment of real turtle ants is the physical structure of the edges. In the tropical forest, many edges are difficult to traverse, which may provide natural inhibition for selecting certain edges. 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. We studied pruning in the context of network repair, but our algorithm also may explain how from day to day, turtle ants constantly prune junctions from existing trails [13].

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 [67, 68, 69, 70]. Our work contributes to the growing intersection of distributed algorithms used by natural biological processes [74, 11].

Methods

Maximum likelihood estimation of parameter values

For each candidate algorithm (RankEdge, Weighted), 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 has been added to and has 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 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 dataset 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 datasets.

Unfortunately, it is not currently possible to measure or manipulate the exact pheromone levels on the branches in the canopy. However, as we show in the Supplement, the probability of each choice is most strongly determined by the most recent ant choices; the less recent choices and weights have little effect on how ants behave at the current time. Thus, even though we do not know the exact weights at the initial time, we can still perform accurate model fitting.

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 at the start of the simulation there were likely ants at nodes on both sides of the break. Each ant was randomly assigned to walk in search of one of the two nests. When an ant reached its destination nest, it then attempted to return to the other nest, and this repeated in a back-and-forth manner for T time-steps. The ants walk synchronously for T time-steps — a common assumption in distributed computing problems.

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 turtle ants have at least one time-step of memory. Our simulations include two 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 [75, 73]. 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 one time step of memory, each ant requires one bit of memory to remember if it is walking back from a dead end. The total memory requirements are still constant.

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 queuing increases the robustness of the algorithm. Certain edges in the routing backbone may need to be pruned when the network changes; however, if there is significant pheromone on such edges, pruning is difficult when many ants concurrently traverse the edge. Queueing helps avoid this scenario because ants are forced to select edges one at a time.

3. When turtle ants take an "explore step", they often traverse an edge for a short distance, and then return to the original node [13]. 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.

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), 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.

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". For RankEdge, this means commuting from one nest to the other by simply following the highest-weighted edges. We thus measure performance of the algorithm assuming $q_{\text{explore}} = 0$, and consider all paths that may be taken with positive probability under this constraint.

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 to commute between two nests with positive probability even if $q_{\text{explore}} = 0$.

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 maximal path. This is defined empirically by computing the percentage of the N simulations where a maximal 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 maximal path.
  - Let $M_1, M_2, \dots, M_n$ be the set of all n maximal paths in the final graph.
  - Let $p_1, p_2, \dots, p_n$ be the probabilities of taking each maximal path if an ant uses RankEdge 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 maximal paths in the final graph.

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. 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 pruning:** An information-theoretic measure of the degree to which ants explore multiple alternative paths before converging to one.
  - Let $S_t = S(\hat{p}_t)$ be the entropy over the distribution of chosen paths $\hat{p}$ at 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 pruning is then the maximum entropy minus the entropy at the end of the simulation: $S_{max} - S_T$.
- **Path length pruning**: A measure of ants of how well ants prune longer paths.
  - Suppose at time t the ants have taken chosen paths $p_1, p_2, \dots, p_n$ with frequencies $c_1, c_2, \dots, c_n$. Let $l(p_i)$ be the length of path p_i. We define the *weighted-mean chosen path length* at time t to be:

$$L_t = \frac{\sum_i c_i \cdot l(p_i)}{\sum_i c_i}$$

.

- 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. We then seek the set of parameter values that maximize robustness over the six networks.

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.

	Weighted	RankEdge	Unweighted
q_{explore}	0.05 ± 0.06	0.20 ± 0.04	N/A
q_{decay}	0.01 ± 0.10	0.02 ± 0.08	N/A
Log-likelihood	-345.05	-405.42	-557.37

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 datasets, each corresponding to field observations of one node (junction) on a given day. Both models are significantly more likely to explain the data than the null model (Unweighted).

	Optimal	Unweighted	RankEdge
Minimal graph	12.00	16.96 ± 07.60	12.00 ± 0.00
Simple graph	12.00	40.11 ± 36.11	12.98 ± 1.01
Medium graph	10.00	60.17 ± 55.33	10.95 ± 1.01
Full grid	13.00	220.65 ± 191.31	13.06 ± 0.34
Spanning grid	13.00	113.01 ± 108.50	14.56 ± 3.97

Table 2: **Average path length for each algorithm.** For each graph, we show the average path length of the final graph, generated using three different algorithms. The RankEdge algorithm performed much better than the null model (Unweighted) and close to the globally shortest path length (Optimal).

	Path pruning	Path length pruning
Simple	0.18 ± 0.26	0.19 ± 0.56
Medium	0.25 ± 0.29	0.68 ± 1.11
Full Grid	0.14 ± 0.20	0.40 ± 0.80
Spanning Grid	0.15 ± 0.18	1.42 ± 2.88
European roads	0.10 ± 0.16	0.46 ± 1.99

Table 3: **Path pruning**: For each network, we measured the average path pruning and path length pruning (Methods) observed using the RankEdge algorithm. We omit the Minimal graph, because there is only one possible path from one nest to the other, and thus no pruning is possible.

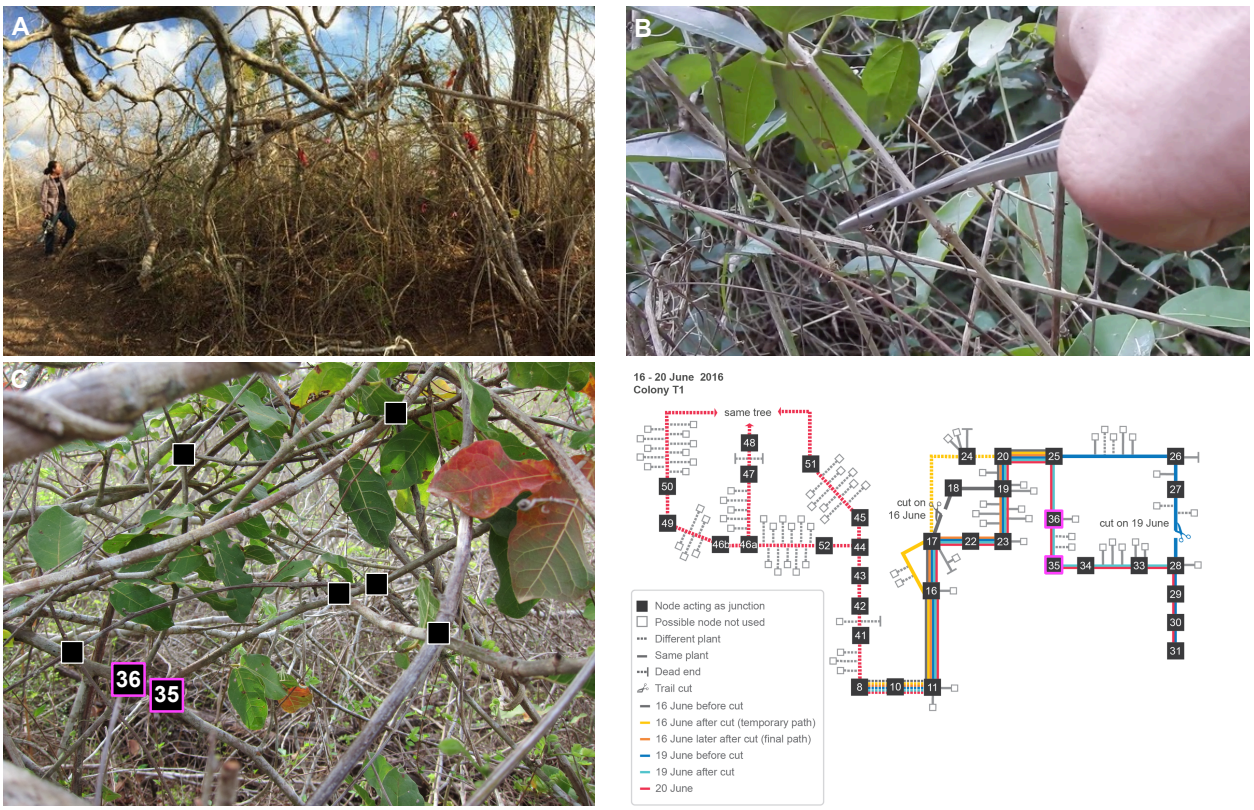

Figure 1: **Turtle ant habitat and trail network.** A) The photograph shows the highly tangled forest canopy in which turtle ants forage. B) Experiments were performed in which an edge in the path was cut, to observe how the ants respond and repair the break [13]. C) Modeling the trail network as a graph, with junctions as nodes and connecting branches and twigs as edges. The diagram on the right from [13] shows a detailed depiction of a large portion of the trail network; the photograph on the left shows nodes 35 and 36.

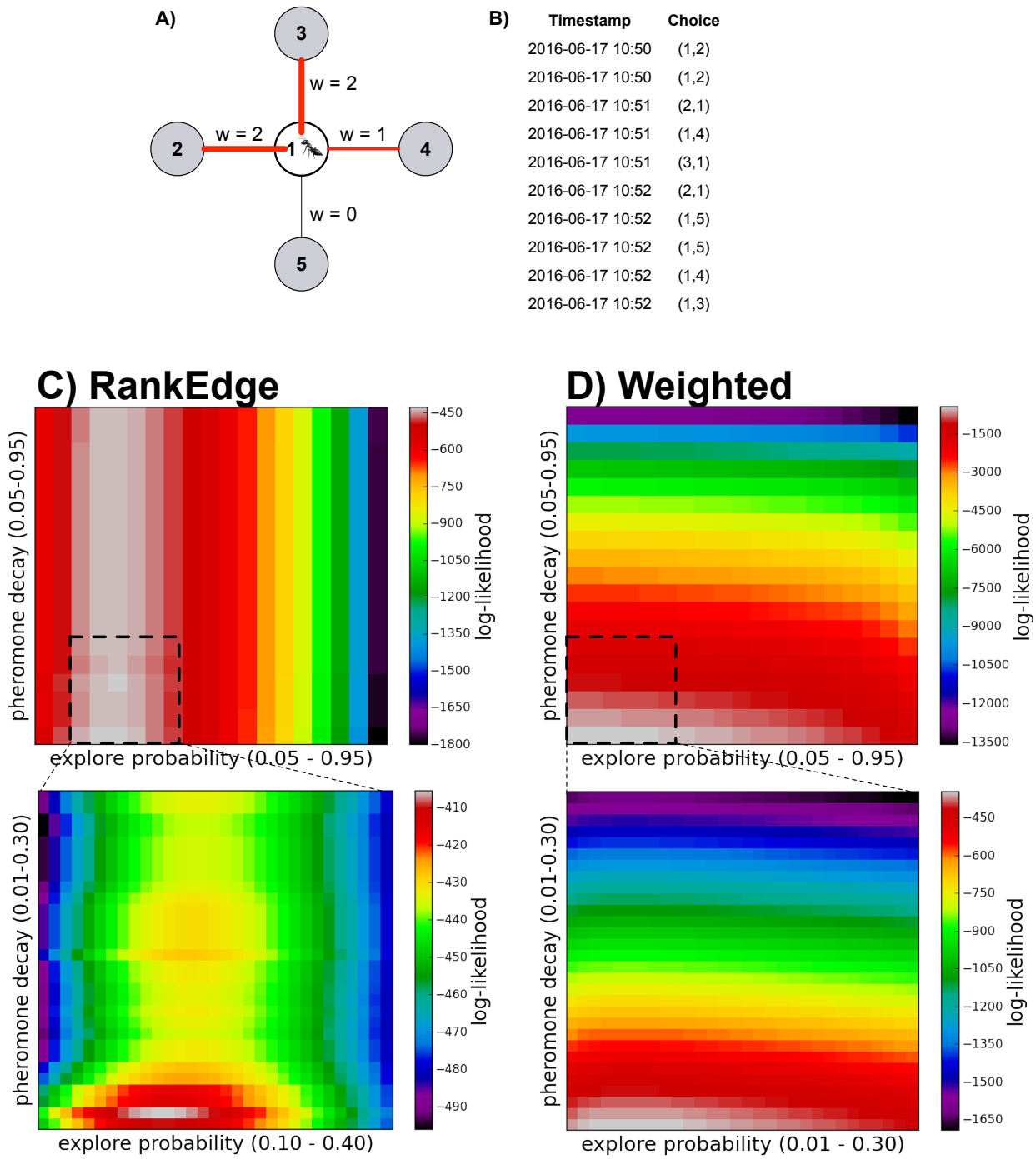

Figure 2: **Maximum likelihood computation.** A–B) Example node junction and edge choices for turtle ants. Under the RankEdge algorithm, the likelihood of choosing edges $1 \to 3$ or $1 \to 2$ is $(1 - q_{\text{explore}})(1/2)$; the likelihood of edge $1 \to 4$ is $q_{\text{explore}}(1 - q_{\text{explore}})$; and the likelihood of $1 \to 5$ is q^2_{explore}. 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.

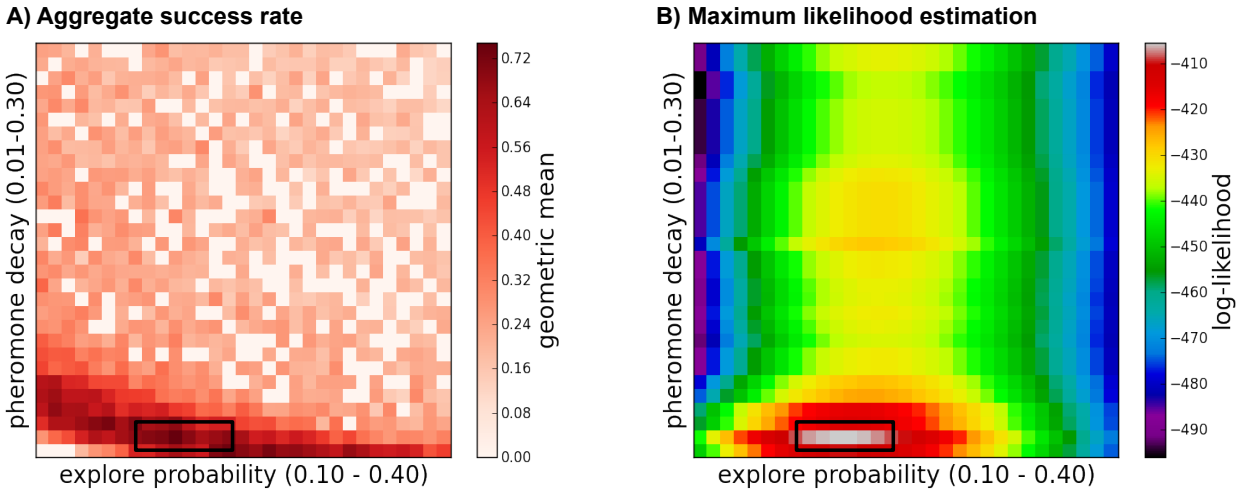

Figure 3: **The maximum likelihood parameters closely match the best simulation parameters.** A) The color of each square in the heatmap corresponds to the robustness (Methods) of the success rates for the RankEdge algorithm for each combination of q_{explore} (x-axis) and q_{decay} (y-axis) values. 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.

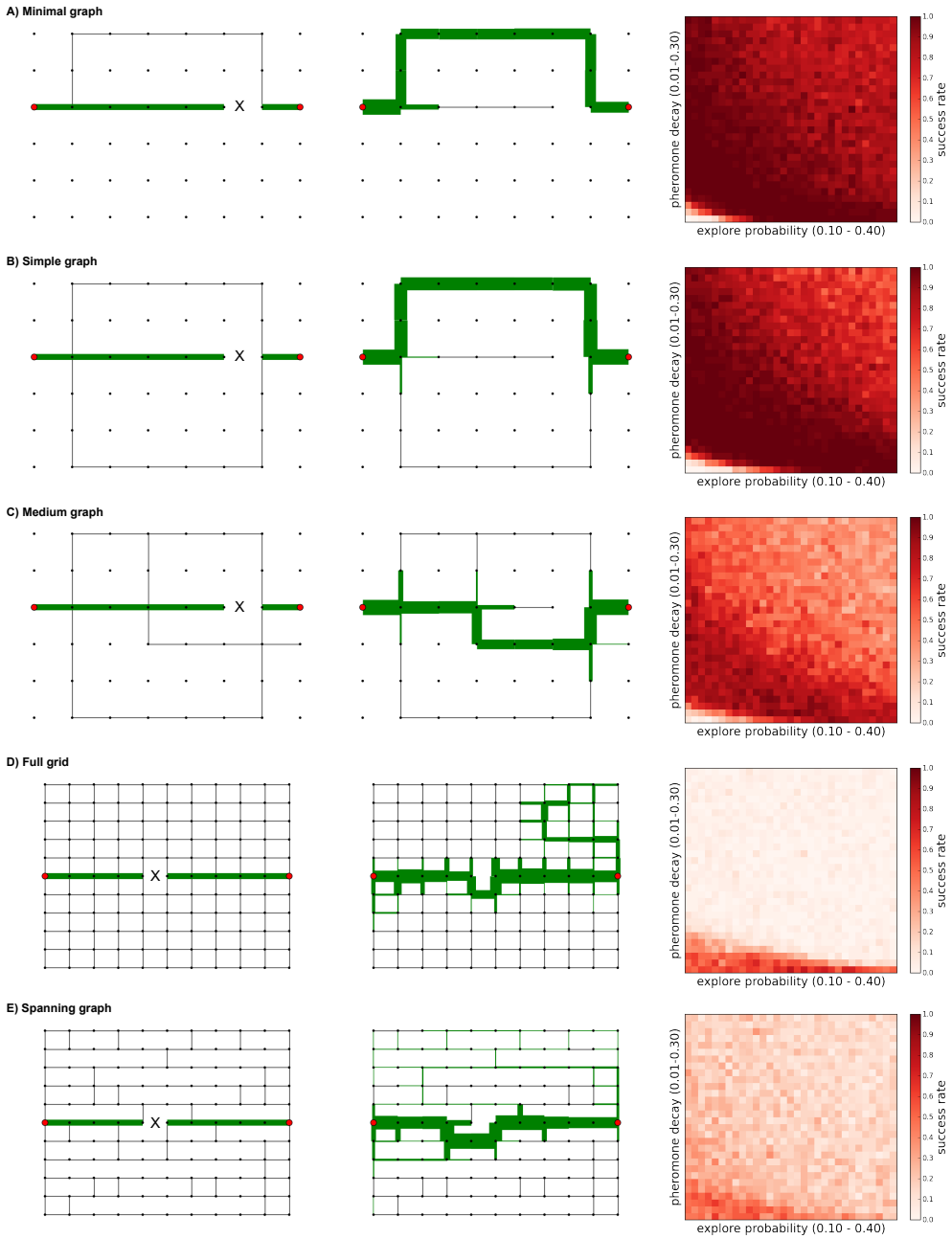

Figure 4: **Success rates for each network.** A–E) For each network we show the initial graph (left), an example of the final graph after running the RankEdge algorithm using the maximum likelihood parameters (middle), and the algorithm's success rate for each parameter combination (right). In the initial 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. In all cases where the algorithm succeeds, we observe a strong convergence to a single maximal path.

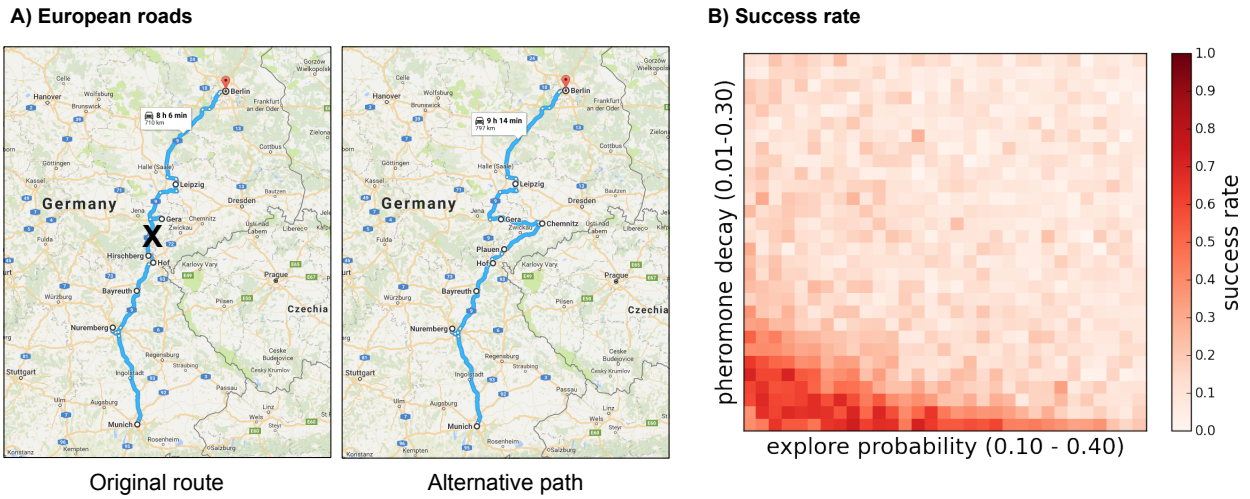

Figure 5: **Repairing road closures in the Europe road graph.** Analysis of how well the turtle ant algorithm translates to repair simulated breaks in a real-world transport network. A) An example path in the European E-road network connecting Munich to Berlin, Germany. The roads and junctions form a 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.

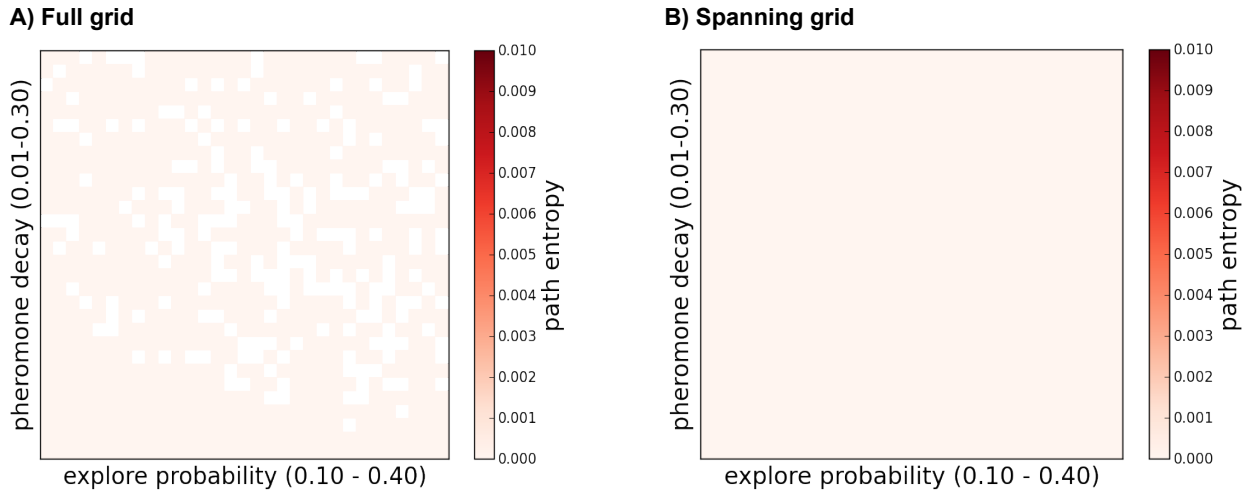

Figure 6: **Path entropy for the Full and Spanning grids.** RankEdge converges to a single path with an average path entropy near 0. Lower values are better, corresponding to lighter shades of red.

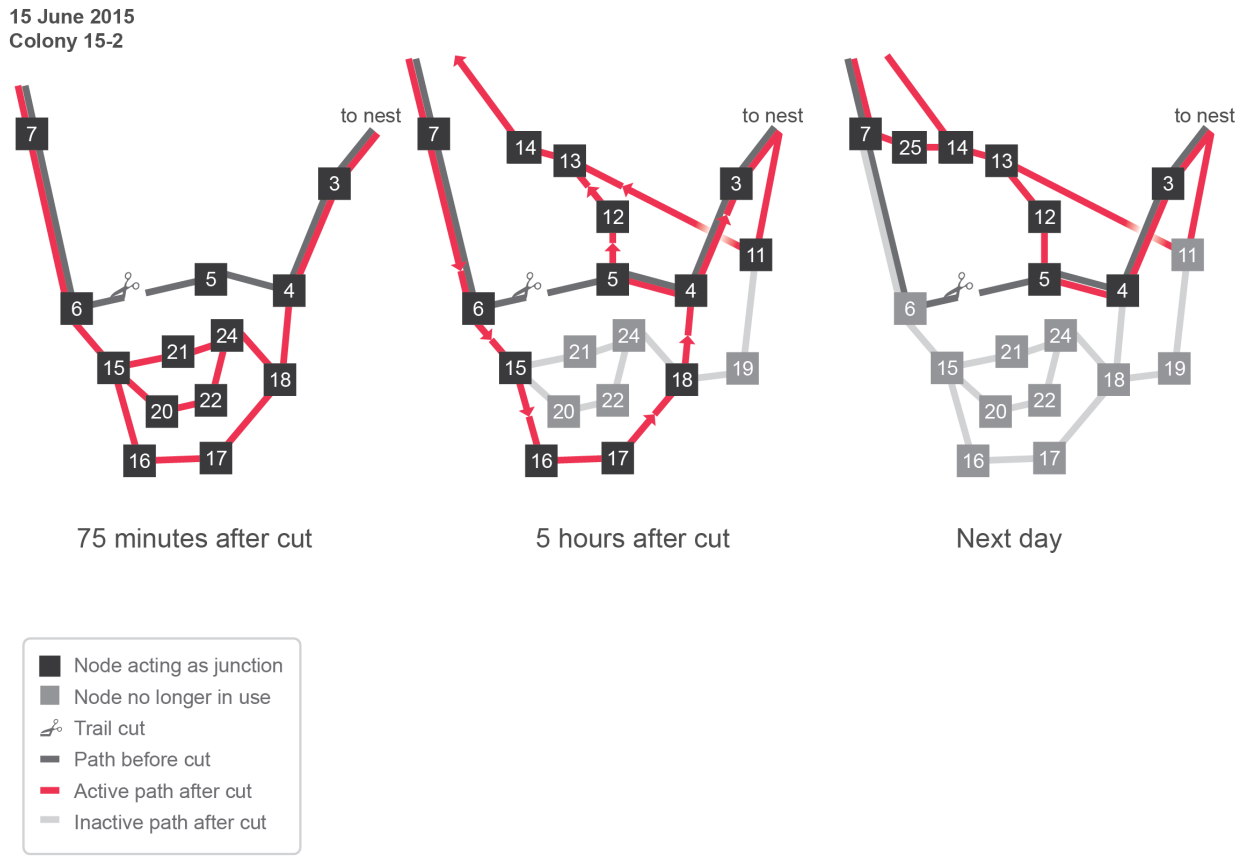

Figure 7: **Turtle ants prune paths.** The diagram from [13] shows the results of an experiment in which an edge was cut. Left: The initial trail is shown in grey. The edge connecting nodes 5 and 6 was cut. After 75 minutes, the turtle ants explored several new paths (red). Center: Five hours after the 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; Illia Ziamtsov, Javier How, Benjamin Cosman, Ailie Fraser, Will Hamilton, Sam Crow, and Alex Lang for helpful comments on the manuscript; and Semay Johnston for illustration of the trail networks in Figure 1C and Figure 7.

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

[2] Tero, A., Takagi, S., Saigusa, T., et al. "Rules for biologically inspired adaptive network design". *Science* 327.5964 (2010), pp. 439–442.

[3] Navlakha, S., Barth, A. L., and Bar-Joseph, Z. "PLOS Computational Biology: Decreasing-Rate Pruning Optimizes the Construction of Efficient and Robust Distributed Networks". *PLoS ONE* (2015). (Accessed on 07/13/2016).

[4] Latty, T., Ramsch, K., Ito, K., et al. "Structure and formation of ant transportation networks". *J R Soc Interface* 8.62 (2011), pp. 1298–1306.

[5] Brabazon, A., O'Neill, M., and McGarraghy, S. *Natural Computing Algorithms (Natural Computing Series)*. Springer, 2015.

[6] Gordon, D. M. "The dynamics of foraging trails in the tropical arboreal ant Cephalotes goniodontus". *PLoS ONE* 7.11 (2012), e50472.

[7] Bottinelli, A., Wilgenburg, E. van, Sumpter, D. J., and Latty, T. "Local cost minimization in ant transport networks: from small-scale data to large-scale trade-offs". *J R Soc Interface* 12.112 (2015).

[8] Lanan, M. C., Dornhaus, A., and Bronstein, J. L. "The function of polydomy: the ant Crematogaster torosa preferentially forms new nests near food sources and fortifies outstations". *Behavioral Ecology and Sociobiology* 65.5 (2011), pp. 959–968.

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

[10] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. *Introduction to algorithms*. Vol. 6. MIT press Cambridge, 2001.

[11] Gordon, D. M. "The evolution of the algorithms for collective behavior". *Cell Syst* 3.6 (2016), pp. 514–520.

[12] Gomez, C., Gilabert, F., Gomez, M. E., et al. "Deterministic versus adaptive routing in fat-trees". *Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International*. IEEE. 2007, pp. 1–8.

[13] Gordon, D. M. "Local regulation of trail networks of the arboreal turtle ant, Cephalotes goniodontus". In press at American Naturalist. 2017.

[14] Middleton, E. J. and Latty, T. "Resilience in social insect infrastructure systems". *Journal of The Royal Society Interface* 13.116 (2016), p. 20151022.

[15] Malkov, M., Yates, C., and Boov, K. *A stochastic model of ant trail following with two pheromones*. 2015. eprint: `arXiv:1508.06816`.

[16] Flanagan, T. P., Pinter-Wollman, N. M., Moses, M. E., and Gordon, D. M. "Fast and flexible: Argentine ants recruit from nearby trails". *PloS one* 8.8 (2013), e70888.

[17] Garnier, S., Guérécheau, A., Combe, M., et al. "Path selection and foraging efficiency in Argentine ant transport networks". *Behavioral Ecology and Sociobiology* 63.8 (2009), pp. 1167–1179.

[18] Dussutour, A., Fourcassie, V., Helbing, D., and Deneubourg, J.-L. "Optimal traffic organization in ants under crowded conditions". *Nature* 428.6978 (2004), pp. 70–73.

[19] Deneubourg, J.-L., Goss, S., Franks, N., and Pasteels, J. "The blind leading the blind: modeling chemically mediated army ant raid patterns". *Journal of insect behavior* 2.5 (1989), pp. 719–725.

[20] Cherix, D., Werner, P., Catzeflis, F, et al. "Spatial organisation of a polycalic system in Formica (Coptoformica) exsecta Nyl.(Hymenoptera: Formicidae)." *Mitteilungen der Schweizerischen Entomologischen Gesellschaft* 53.2/3 (1980), pp. 163–172.

[21] Deneubourg, J. L., Aron, S., Goss, S., et al. "Random behaviour, amplification processes and number of participants: how they contribute to the foraging properties of ants". *Physica D: Nonlinear Phenomena* 22.1 (1986), pp. 176–186.

[22] Franks, N. R. "Army ants: a collective intelligence". *American Scientist* 77 (1989), pp. 138–145.

[23] Reid, C. R., Lutz, M. J., Powell, S., et al. "Army ants dynamically adjust living bridges in response to a cost–benefit trade-off". *Proceedings of the National Academy of Sciences* 112.49 (2015), pp. 15113–15118.

[24] Hart, A. and Jackson, D. E. "U-turns on ant pheromone trails". *Current Biology* 16.2 (2006), R42.

[25] Jackson, D., Holcombe, M., and Ratnieks, F. "Coupled computational simulation and empirical research into the foraging system of Pharaohs ant (Monomorium pharaonis)". *Biosystems* 76.1 (2004), pp. 101–112.

[26] Jackson, D. E., Martin, S. J., Holcombe, M., and Ratnieks, F. L. "Longevity and detection of persistent foraging trails in Pharaoh's ants, Monomorium pharaonis (L.)" *Animal Behaviour* 71.2 (2006), pp. 351 –359.

[27] Robinson, E. J., Jackson, D. E., Holcombe, M., and Ratnieks, F. L. "Insect communication:no entrysignal in ant foraging". *Nature* 438.7067 (2005), pp. 442–442.

[28] Robinson, E. J., Green, K., Jenner, E., et al. "Decay rates of attractive and repellent pheromones in an ant foraging trail network". *Insectes sociaux* 55.3 (2008), pp. 246–251.

[29] Robinson, E. J., Ratnieks, F. L., and Holcombe, M. "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 (2008), pp. 250–258.

[30] Colorni, A., Dorigo, M., Maniezzo, V., et al. "Distributed optimization by ant colonies". *Proceedings of the first European conference on artificial life*. Vol. 142. Paris, France. 1991, pp. 134–142.

[31] Dorigo, M. and Blum, C. "Ant colony optimization theory: A survey". *Theoretical Computer Science* 344.2-3 (2005), pp. 243–278.

[32] López-Ibánez, M, Stützle, T, and Dorigo, M. "Ant Colony Optimization: A Component-Wise Overview" (2015).

[33] Gambardella, L. M., Montemanni, R., and Weyland, D. "Coupling ant colony systems with strong local searches". *European Journal of Operational Research* 220.3 (2012), pp. 831–843.

[34] Tsutsui, S. "Ant colony optimization with cunning ants". *Transactions of the Japanese Society for Artificial Intelligence* 22 (2007), pp. 29–36.

[35] Wiesemann, W. and Stützle, T. "Iterated ants: An experimental study for the quadratic assignment problem". *International Workshop on Ant Colony Optimization and Swarm Intelligence*. Springer. 2006, pp. 179–190.

[36] Fraigniaud, P., Ilcinkas, D., Peer, G., et al. "Graph exploration by a finite automaton". *Theoretical Computer Science* 345.2 (2005), pp. 331–344.

[37] Hanusse, N., Kavvadias, D., Kranakis, E., and Krizanc, D. "Memoryless search algorithms in a network with faulty advice". *Theoretical Computer Science* 402.2 (2008), pp. 190–198.

[38] Wagner, I. A., Lindenbaum, M., and Bruckstein, A. M. "Efficiently searching a graph by a smell-oriented vertex process". *Annals of Mathematics and Artificial Intelligence* 24.1-4 (1998), pp. 211–223.

[39] Feinerman, O., Korman, A., Lotker, Z., and Sereni, J.-S. "Collaborative Search on the Plane Without Communication". *Proceedings of the 2012 ACM Symposium on Principles of Distributed Computing*. PODC '12. Madeira, Portugal: ACM, 2012, pp. 77–86.

[40] Emek, Y., Langner, T., Stolz, D., et al. "Towards More Realistic ANTS". *2nd Workshop on Biological Distributed Algorithms (BDA)*. 2014.

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

[42] Wunder, M., Littman, M. L., and Babes, M. "Classes of multiagent q-learning dynamics with epsilon-greedy exploration". *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*. 2010, pp. 1167–1174.

[43] Tokic, M. and Palm, G. "Value-Difference Based Exploration: Adaptive Control between Epsilon-Greedy and Softmax." *KI*. Springer. 2011, pp. 335–346.

[44] Kuleshov, V. and Precup, D. "Algorithms for multi-armed bandit problems". *arXiv preprint arXiv:1402.6028* (2014).

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

[46] Chandy, K. M. and Misra, J. "Distributed computation on graphs: Shortest path algorithms". *Communications of the ACM* 25.11 (1982), pp. 833–837.

[47] Humblet, P. A. et al. "Another adaptive distributed shortest path algorithm". *IEEE transactions on communications* 39.6 (1991), pp. 995–1003.

[48] Perlman, R. "An algorithm for distributed computation of a spanningtree in an extended LAN". *ACM SIGCOMM Computer Communication Review*. Vol. 15. 4. ACM. 1985, pp. 44–53.

[49] Garay, J. A., Kutten, S., and Peleg, D. "A sublinear time distributed algorithm for minimum-weight spanning trees". *SIAM Journal on Computing* 27.1 (1998), pp. 302–316.

[50] Suomela, J. "Survey of local algorithms". *ACM Computing Surveys (CSUR)* 45.2 (2013), p. 24.

[51] Afek, Y., Alon, N., Bar-Joseph, Z., et al. "Beeping a maximal independent set". *Distributed computing* 26.4 (2013), pp. 195–208.

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

[53] Coppersmith, D. and Diaconis, P. "Random Walk with Reinforcement" (1986). Unpublished Manuscript.

[54] Diaconis, P. and Freedman, D. "de Finetti's theorem for Markov chains". *The Annals of Probability* (1980), pp. 115–130.

[55] Stevens, A. and Othmer, H. G. "Aggregation, blowup, and collapse: the ABC's of taxis in reinforced random walks". *SIAM Journal on Applied Mathematics* 57.4 (1997), pp. 1044–1081.

[56] Jeanson, R., Ratnieks, F. L., and Deneubourg, J.-L. "Pheromone trail decay rates on different substrates in the Pharaoh's ant, Monomorium pharaonis". *Physiological Entomology* 28.3 (2003), pp. 192–198.

[57] Aron, S, Pasteels, J. M., and Deneubourg, J. L. "Trail-laying behaviour during exploratory recruitment in the argentine ant, Iridomyrmex humilis (Mayr)". *Biology of Behaviour* 14 (1989), pp. 207–217.

[58] Pinter-Wollman, N., Wollman, R., Guetz, A., et al. "The effect of individual variation on the structure and function of interaction networks in harvester ants". *J R Soc Interface* 8.64 (2011), pp. 1562–1573.

[59] Mersch, D. P., Crespi, A., and Keller, L. "Tracking individuals shows spatial fidelity is a key regulator of ant social organization". *Science* 340.6136 (2013), pp. 1090–1093.

[60] Gordon, D. M. "The expandable network of ant exploration". *Animal Behaviour* 50.4 (1995), pp. 995–1007.

[61] Simon, T and Hefetz, A. "Trail-following responses ofTapinoma simrothi (Formicidae: Dolichoderinae) to pygidial gland extracts". *Insectes Sociaux* 38.1 (1991), pp. 17–25.

[62] Perna, A., Granovskiy, B., Garnier, S., et al. "Individual rules for trail pattern formation in Argentine ants (Linepithema humile)". *PLoS computational biology* 8.7 (2012), e1002592.

[63] Deneubourg, J.-L., Pasteels, J. M., and Verhaeghe, J.-C. "Probabilistic behaviour in ants: a strategy of errors?" *Journal of Theoretical Biology* 105.2 (1983), pp. 259–271.

[64] Fonio, E., Heyman, Y., Boczkowski, L., et al. "A locally-blazed ant trail achieves efficient collective navigation despite limited information". *eLife* 5 (2016), e20185.

[65] Sumpter, D. J. and Beekman, M. "From nonlinearity to optimality: pheromone trail foraging by ants". *Animal behaviour* 66.2 (2003), pp. 273–280.

[66] Kunegis, J. "KONECT – The Koblenz Network Collection". *Proc. Int. Conf. on World Wide Web Companion*. 2013, pp. 1343–1350.

[67] Lund, K., Manzo, A. J., Dabby, N., et al. "Molecular robots guided by prescriptive landscapes". *Nature* 465.7295 (2010), pp. 206–210.

[68] Brambilla, M., Ferrante, E., Birattari, M., and Dorigo, M. "Swarm robotics: a review from the swarm engineering perspective". *Swarm Intell* 7.1 (2013), pp. 1–41.

[69] Werfel, J., Petersen, K., and Nagpal, R. "Designing collective behavior in a termite-inspired robot construction team". *Science* 343.6172 (2014), pp. 754–758.

[70] Hecker, J. P. and Moses, M. E. "Beyond pheromones: evolving error-tolerant, flexible, and scalable ant-inspired robot swarms". *Swarm Intelligence* 9.1 (2015), pp. 43–70.

[71] Korn, C. and Augustin, H. G. "Mechanisms of Vessel Pruning and Regression". *Dev. Cell* 34.1 (2015), pp. 5–17.

[72] Prabhakar, B., Dektar, K. N., and Gordon, D. M. "The regulation of ant colony foraging activity without spatial information". *PLoS Comput Biol* 8.8 (2012), e1002670.

[73] Czaczkes, T. J., Grüter, C., and Ratnieks, F. L. "Trail pheromones: an integrative view of their role in social insect colony organization". *Annual review of entomology* 60 (2015), pp. 581–599.

[74] Navlakha, S. and Bar-Joseph, Z. "Distributed Information Processing in Biological and Computational Systems". *Commun. ACM* 58.1 (Dec. 2014), pp. 94–102.

[75] Czaczkes, T. J., Grüter, C., and Ratnieks, F. L. "Negative feedback in ants: crowding results in less trail pheromone deposition". *Journal of the Royal Society Interface* 10.81 (2013), p. 20121009.