Models of Adaptive Navigation, Inspired by Ant Transport Strategy in the Presence of Obstacles

Elizabeth E. Esterly[1], Helen McCreery[2], Radhika Nagpal[3]

[1]Department of Computer Science, University of New Mexico
[2]Department of Integrative Biology, Michigan State University
[3]Department of Computer Science, Harvard University
elizabethesterly@gmail.com, hmccreery@gmail.com, rad@eecs.harvard.edu

***Abstract*— Cooperative transport is an impressive example of collective behavior in ants, where groups of ants work together to move heavy food objects back to their nest over heterogeneous terrain. This behavior also serves as a model for bio-inspired robotics. While many studies have considered the mechanisms by which ants transport objects in simple settings, few have looked at how they deal with obstacles and heterogeneous terrain. A recent study on *Paratrechina longicornis* (crazy ants) demonstrated that groups of these ants implement a stochastic, adaptive, and robust cooperative transport strategy that allows them to succeed at navigating challenging obstacles that require moving away from their goal. In this paper, we use group-level computational models to investigate the significance and implications of this biological strategy. We develop an algorithm that reproduces important elements of the strategy, and compare it to several benchmark algorithms for a range of obstacle sizes and shapes. Our results show that, for smaller obstacles, the ant-inspired adaptive stochastic strategy is adept at efficient navigation, due to its ability to match the level of randomness it uses to unknown object size and shape. We also find some unexpected differences between our results and the original ant transport behavior, suggesting new biological experiments.**

I. Introduction

In an impressive example of emergent behavior, some ant species form groups to collectively carry large food objects home to their nests. While many ant species are uncoordinated, some can cooperatively transport objects many thousands of times the mass of an individual; centimeter-scaled weaver ants, for example, are able to cooperatively carry dead birds and reptiles up trees to their arboreal nests [1]. The ecology and mechanisms of cooperative transport have been an active area of research in both biology and computer science, where ant behavior has served as bio-inspiration for robotic cooperative transport [2]. Experimental studies of ant cooperative transport have improved our understanding of the mechanisms that allow for the recruitment of a sufficient number of ants at a heavy object, and that enable these individuals to coordinate their efforts and move successfully in the direction of their nest in flat terrains [3], [4], [5].

Natural environments pose a complex challenge to groups of ants engaged in cooperative transport, as ants must navigate to their nests across heterogeneous terrain. At an ant's scale, even small features such as leaves and twigs complicate navigation. Yet the mechanism by which ants are able to navigate these obstacles while engaged in cooperative transport has not been well-studied. Obstacle navigation requires making a series of decisions that build on one another, rather than a discrete decision, and is thus akin to maze navigation, requiring problem solving. There is a tradeoff between simple and robust obstacle navigation strategies; simple strategies that only require limited information will succeed with a narrower range of obstacle types than more complex approaches. The strategy complexity comes at a cost, requiring more difficult sensing, memory, and communication by the ant group.

A recent set of experiments by McCreery et al.[6] looked at the obstacle-navigation strategy of groups of crazy ants, *Paratrechina longicornis*, a species that is highly effective at group transport. The ants were presented with multiple challenges, including a concave obstacle; concave obstacles pose a special challenge as they require the ants to move away from the direction of their nest. These experiments suggest that this ant species employs a stochastic but time-adaptive strategy that enables successful navigation of concave obstacles. McCreery et al. [6] concluded that these groups start with a relatively simple strategy, using nest direction information, and incorporate increasing levels of stochastic behavior into their strategy over time, moving farther away from their goal the longer they are stuck.

In this paper, we use group-level computational models to investigate the significance of this biological strategy. We present an algorithmic description of the ant strategy described by McCreery et al., and we develop a simulation environment to test this algorithm with new obstacle scenarios beyond the original ant experiments. We compare the effectiveness and versatility of the ant-inspired algorithm to other algorithms, such as stochastic algorithms that do not adapt with time, as well as deterministic algorithms from robotics for single robot navigation that require more complex reasoning. Our results show that, for smaller obstacles, an adaptive stochastic strategy is flexible to accommodating changes to obstacle size and shape due to its time-varying behavior, which allows the algorithm to match the level of randomness it uses to unknown object size and shape. We also find some unexpected differences between our results and the original ant transport

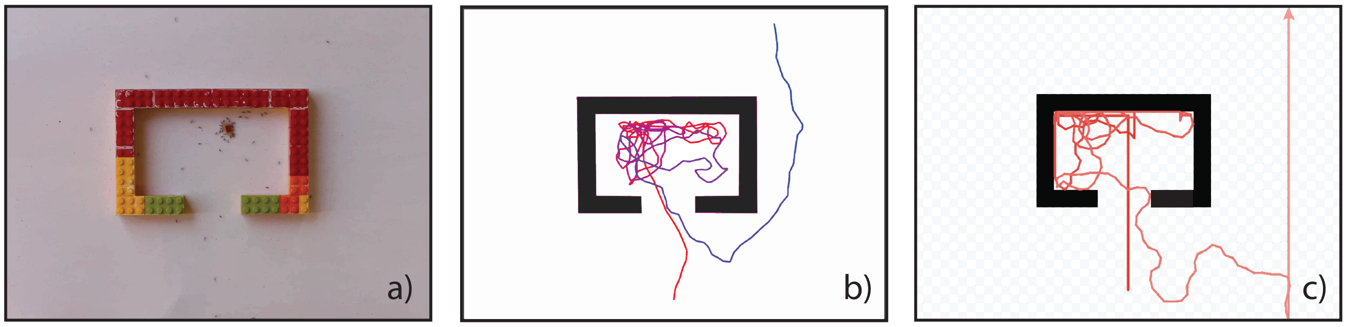

Fig. 1. Comparison of arenas. a) Experimental video still from McCreery et al. [6]; b) Sample trajectory taken by *P. longicornis*; c) Sample trajectory generated by the Adaptive Stochastic algorithm.

behavior, suggesting new areas for biological experiments. This work improves our understanding of the implications of the observed biological navigation strategy used by ants, suggests new algorithmic possibilities for navigation in simple robots that can work in the presence of obstacles, and suggests future possibilities for collective robotics.

II. Models

We begin this section by briefly summarizing the results of McCreery et al [6]. Next, we discuss how we translated their experimental setup into a computational model. Finally, we express several elements of the ants' behavior identified in McCreery et al. as algorithmic components and construct algorithms from combinations of these components.

A. McCreery et al. (2016)

McCreery et al. [6] compared the trajectories of groups of ants transporting bait in the presence of three shapes of obstacles that increase in complexity. The simplest of these obstacles is a straight, short wall. The only information required to successfully navigate the wall is the direction of the goal. The cul-de-sac obstacle is more complex than the wall, and forces the ants to move away from the goal to succeed (Fig. 1a). McCreery et al. also obstructed groups with an impossible trap (a cul-de-sac without an opening) to see if transporting ants can detect that no solution exists.

P. longicornis groups appear to implement a stochastic, adaptive strategy when navigating all three obstacles, incorporating the following behavioral elements: move toward goal, symmetry breaking, perimeter following, spontaneous direction changes, and move away from goal [6]. Importantly, the behavior of these groups changed over time; the longer a group was stuck, the more they moved away from their goal. This strategy allowed them to rapidly solve the simple wall and still succeed at the cul-de-sac. Although McCreery et al. studied the behavior of ants acting collectively, they evaluated the strategy of the ant group by recording the trajectory of the bait only, as if it were a single entity.

B. Simulation Model

In this paper, we focus primarily on the cul-de-sac experiments. The experiment arena dimensions are matched to the area of the cul-de-sac experiment video (432mm X 280 mm, A3 paper size) and we reproduce the shape, size, and position of the obstacle in [6] by placing it in the center of the arena (height 88 mm, width 160 mm, arms 40 mm each) (Fig. 1c). The simulation is implemented in NetLogo 5.2 [12]; space is modeled as continuous and time is modeled in discrete steps.

In the experimental study, the navigation strategy is studied at the group level by recording and analyzing the trajectory of the transported bait. The ants maintain a mostly steady speed while cooperatively transporting their target object and rarely stall [6]. Following the study choices, our group-level model abstracts the ants and the bait together as a single agent that travels at a constant speed of 8 mm per time step. Henceforth we refer to 1 step as having a length of 8 mm. The agent is considered to have reached the goal when it encounters the northern border of the simulation.

C. Algorithms

We define three classes of algorithms based on strategy: **Deterministic**, **Stochastic**, and **Hybrid**. We present both verbal and finite state automata descriptions (Fig. 2) of the algorithms contained in these classes. The **Deterministic** strategies are inspired by a class of robot navigation algorithms, ironically called "Bug" algorithms, that navigate unknown obstacles using simple sensing strategies and without building maps [7], [8]. We acknowledge that the Bug0 and BugD algorithms contained within this class do make a single stochastic choice upon encountering an obstacle; however, their behavior is deterministic otherwise with respect to the obstacle. The **Stochastic** strategy, *Infinite Stochastic*, employs a correlated random walk. The **Hybrid** strategies allow switching between deterministic and stochastic behaviors, as has been observed in ants [6]. While each algorithm functions in its own right, we also treat the algorithms *Go North*, *Bug0*, and *Correlated Random Walk* as components in other finite state automata for simplicity.

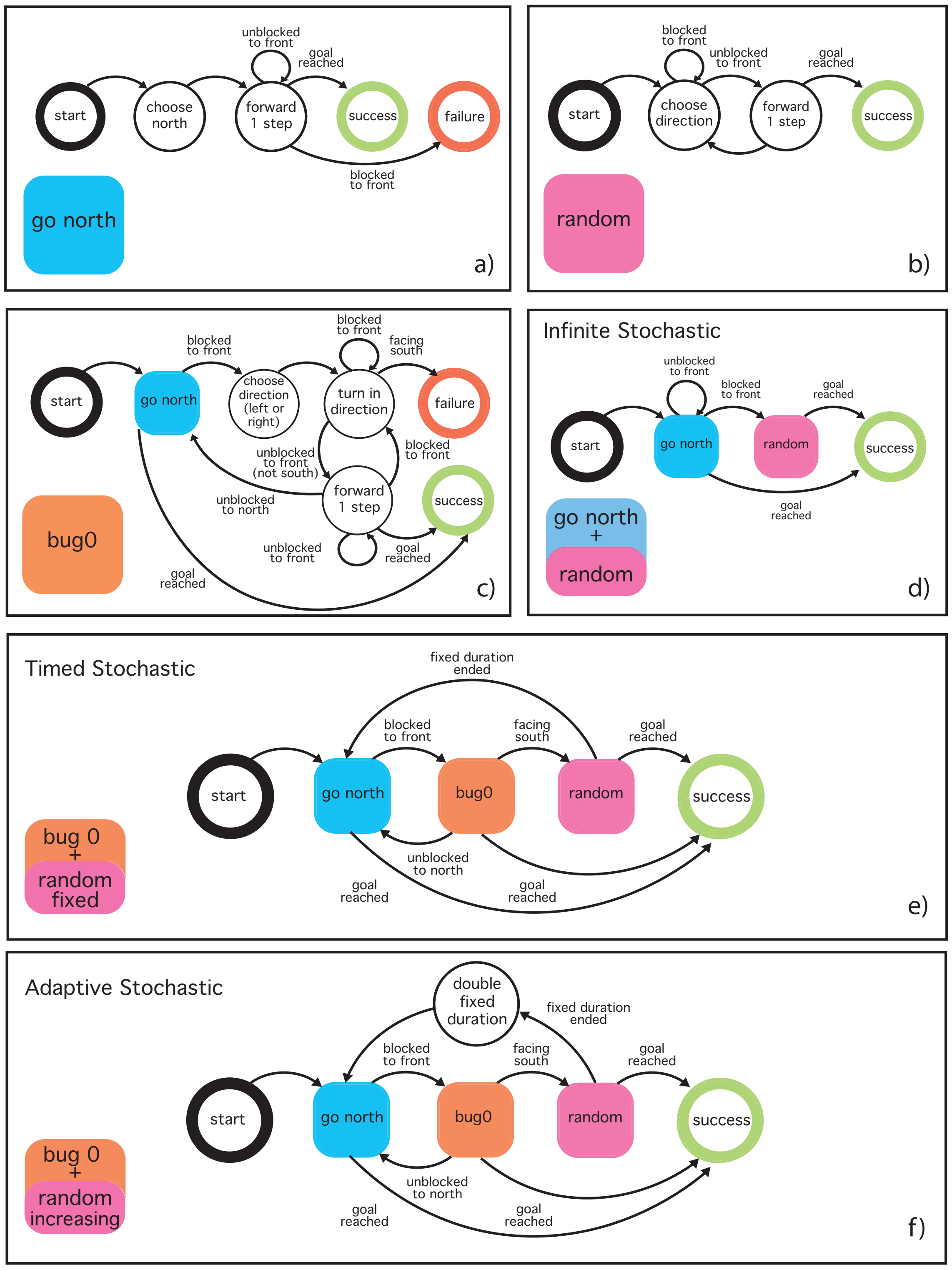

Fig. 2. Finite State Automata descriptions of our algorithm components and combination algorithms. We denote the Correlated Random Walk component here simply as Random. From top to bottom, left to right: *Go North*, *Random*, *Bug0*, *Infinite Stochastic: Go North + Random*, *Timed Stochastic: Bug0 + Random-Fixed*, and *Adaptive Stochastic: Bug0 + Random-Increasing*.

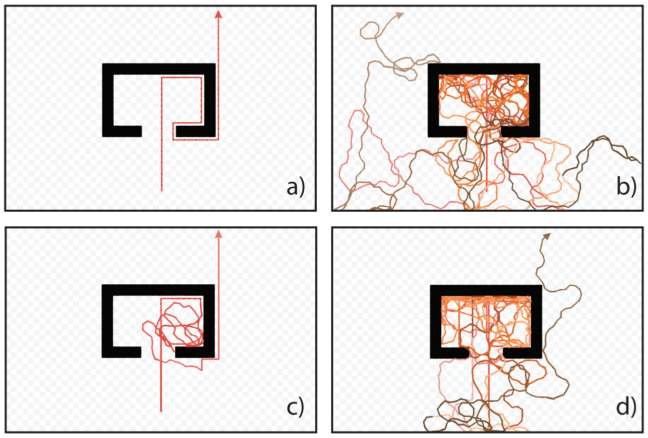

Fig. 3. Sample simulated trajectories generated by different algorithms. a) BugD; b) Infinite Stochastic; c) Timed Stochastic; d) Adaptive Stochastic.

Deterministic

1) Go North: In our simulation, the goal is defined as the northern border and models nest direction for real ants. Go North simulates the tendency of ants to move directly towards their goal. An agent performing the Go North algorithm sets its heading to north and advances 1 step per time step until it reaches the goal (success) or is blocked (failure). Go North is mainly used as a component to other algorithms. *Failure Condition:* The agent is blocked to the north by an obstacle. Unable to take any additional action, it remains stuck.

2) Bug0: At a high level, Bug0 models the behavior of perimeter-following when the simulated agent/ant group is obstructed by an obstacle. The Bug0 algorithm, introduced by Lumelsky and Stephanov [7], is known to work for navigating simple convex obstacles. The agent performs Go North until its failure condition is reached, i.e., it is blocked by an obstacle. At this point, it chooses randomly from one of either left or right, and turns in this direction 90° once per time step until it is unblocked. It then advances in this new direction 1 step per time step. If the agent is blocked, it again executes the series of 90° turns until it is able to advance. This sequence of actions results in basic perimeter-following behavior. While perimeter-following, the agent checks each time step if north is unblocked; if so, it switches back to the Go North algorithm. *Failure Condition:* The agent is forced to turn south, then becomes stuck in a looping behavior. It steps to the south. On the subsequent time step, it detects that the northern direction is open. Thus it moves north. Blocked again to all other directions, it turns until facing south. The cycle repeats infinitely and the algorithm fails.

3) BugD: The BugD algorithm, introduced by Kamon and Rivlin [8], is able to escape the failure condition of Bug0 by introducing an additional agent capability–the ability to measure and compare distances to the goal. The agent memorizes the distance to the goal upon first encountering an obstacle to the north and will not attempt to go north, even if unobstructed, unless it is already further north than the original distance. *Failure Condition*: As shown in [8], BugD does not have a failure condition and can navigate any set of polygonal closed-boundary obstacles. The cul-de-sac is a classic obstacle shape where Bug0 fails and BugD succeeds; this distinction inspired its use in the biological study [6].

Stochastic

1) Correlated Random Walk: On each time step, an agent chooses an integer in the interval $[-60°, 60°]$ uniformly at random and adds that value to its current heading. It then takes 1 step in the direction of its heading. We choose the interval $[-60°, 60°]$ based on visual inspection of simulated trajectories generated by our agent model and a blind test against ant experimental trajectories from [6]. *Failure Condition*: A correlated random walk is guaranteed to visit every point on a plane, given infinite time. It does not have a failure condition.

2) Infinite Stochastic: The agent performs Go North until blocked, then switches to Correlated Random Walk. *Failure Condition*: Infinite Stochastic does not have a failure condition.

Hybrid

We consider two Hybrid algorithms, *Timed Stochastic* and *Adaptive Stochastic*, that have the same underlying structure; they begin with the simplest component behavior and add complexity as simpler strategies fail. Both algorithms begin in Go North. Upon encountering an obstacle, Go North fails, which prompts the agent to switch to Bug0. Triggering the failure condition of Bug0 causes the agent to transition to the Correlated Random Walk phase for a specific duration. The two algorithms determine this duration differently:

1) Timed Stochastic–Bug0+Random-Fixed: The agent performs a Correlated Random Walk for a fixed, pre-determined number of time steps. After the fixed duration ends, the phase terminates, the agent switches to Go North, and subsequently executes the chain of algorithm components again, switching to Bug0 if it encounters an obstacle, and then again to the Correlated Random Walk phase for the same fixed duration.

2) Adaptive Stochastic–Bug0+Random-Increasing: The algorithm begins its first Correlated Random Walk phase with a duration of 5 time steps. The duration of the Correlated Random Walk phase doubles each time the algorithm encounters the failure condition of Bug0. The next time the algorithm switches to this phase, it will execute for 10 time steps; the next, 20, then 40, etc. The Adaptive Stochastic strategy attempts to capture all of the behavioral elements observed by McCreery et. al in the natural system, including goal-directed travel when unobstructed, perimeter-following along wall obstacles, and increasing stochastic trajectories when forced to travel in a direction opposite from the goal.

III. Experiments and Results

In this section, we present a series of simulation experiments that compare the performance of the different algorithms on multiple cul-de-sac inspired obstacles, looking first at varying obstacle size and then at varying obstacle shape. We define our performance metric as path length. Since our simulated agent moves at a constant speed per time step, path length is equivalent to the number of time steps taken by

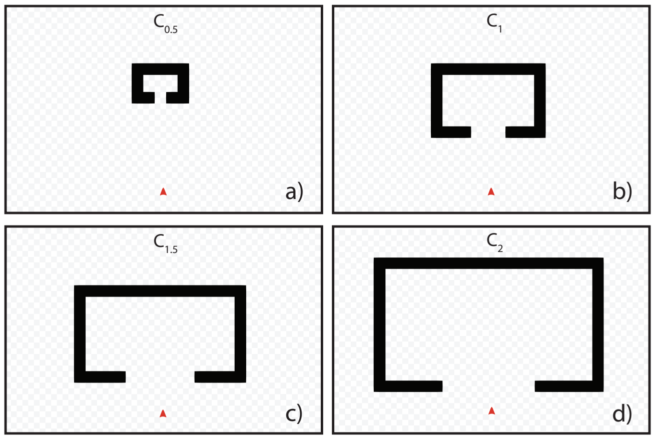

Fig. 4. Cul-de-sac sizes. a) $C_{0.5}$; b) C_1; c) $C_{1.5}$; d) C_2.

the agent to reach the northern goal; a shorter path length denotes better performance. The simulation arena is created in NetLogo 5.2 [12]. Each experiment consists of 10,000 runs of the simulation and is executed via the NetLogo Behavior Space. To test whether our algorithms statistically differ in performance, we use analysis of variance (ANOVA) on log-transformed data where appropriate, including Tukey HSD post-hoc comparisons. In some cases, data could not be transformed to meet the assumptions of ANOVA, so we use non-parametric Kruskal-Wallis tests, including Dunn post-hoc comparisons with a Bonferroni adjustment. All statistical analyses are performed in R 3.4.1 [13].

The Deterministic algorithms Bug0 and BugD also act as bookend comparators in terms of performance. Bug0 always fails in the cul-de-sac with infinite path length, and BugD always succeeds by following the perimeter of the cul-de-sac around the arm and then going straight north when it reaches the end of the arm (path length = 73, Fig. 3a). The purely stochastic approach, Infinite Stochastic, achieves a performance in between, always succeeding but with inefficiency and variability in path length (sample trajectory Fig. 3b).

Where do our hybrid algorithms lie in the performance space? We hypothesize that the Timed Stochastic algorithm will succeed at navigating the cul-de-sac obstacle with lower mean path length and less variance than Infinite Stochastic, if the fixed duration for Timed Stochastic's random walk phase is optimized to the obstacle size. In particular, the Timed strategy should increase efficiency by preventing the agent from random walking for too long after having successfully exited the cul-de-sac obstacle. However, if the obstacle size is unknown, then a Timed Stochastic strategy cannot choose an optimized random walk duration. The Adaptive Stochastic strategy solves this problem by searching for a good parameter; it starts with a small duration and increases it the longer the agent remains stuck. We also hypothesize that searching for a workable value for the random walk duration will provide an advantage over simply randomly walking through the space, and thus Adaptive Stochastic will exhibit a lower mean path length and variance than Infinite Stochastic. Given that the Adaptive Stochastic strategy does not have a parameter tuned to each obstacle, we do not expect it to outperform the optimized Timed Stochastic strategy in terms of lower mean path length and variance. Instead, we look at how high a price is paid by Adaptive Stochastic for searching for the parameter, and compare the flexibility of the two strategies.

A. *VARYING CUL-DE-SAC SIZE*

We denote the original cul-de-sac obstacle as C_1, and create variant cul-de-sac obstacles $C_{0.5}$, $C_{1.5}$ and C_2 with 0.5×, 1.5×, and 2× each of the wall lengths of C_1, respectively (Fig. 4). The area of each cul-de-sac scales as the square of the increase or decrease in wall lengths; in other words, doubling the wall length all around (as we did to create C_2) quadruples the area of the resulting cul-de-sac. To determine the optimized value for the fixed-duration random walk used by Timed Stochastic, we perform a parameter sweep across all cul-de-sac size variants. The optimized values for $C_{0.5}$, C_1, $C_{1.5}$ and C_2 are 25, 100, 150, and 250 time steps, respectively (Fig. 5). In the case of C_2, the parameter setting of 200 appears to have better median performance than our chosen value of 250 when simply observing the figure. But the mean path length and variance generated by the parameter setting 250 was lower than that of 200 ($p = 200$: mean path length $= 1488.55$, $\sigma = 1272.67$; $p = 250$: mean path length $= 1456.36$, $\sigma = 1182.65$). Since we favor consistent performance, we choose to use 250 as the optimized parameter for C_2. For the remaining size cul-de-sac sizes, the median and mean are in agreement for the optimized parameter value.

We test three strategies on each of $C_{0.5}, C_1, C_{1.5}$, and C_2: Timed Stochastic with optimized parameter per size, Adaptive Stochastic, and Infinite Stochastic. Statistical tests showed significant differences in algorithm performance among all cul-de-sac sizes ($C_{0.5}$: H = 9679, df = 2, p < 0.0001; C_1: F = 2157, df = 2, p < 0.0001: $C_{1.5}$: F = 1153, df = 2, p < 0.0001; C_2: F = 646, df = 2, p < 0.0001). All post-hoc comparisons in every size of cul-de-sac showed significant differences between all pairs of algorithms (all p values < 0.0001). The Timed Stochastic algorithm is the strongest performer of the three strategies tested, exhibiting the shortest mean path length and lowest variance of all three algorithms across each cul-de-sac size variant ($C_{0.5}$: *Timed*– mean path length $= 120.34$, $\sigma = 65.91$; *Adaptive*– mean path length $= 147.41$, $\sigma = 98.06$; *Infinite*– mean path length $= 408.81$, $\sigma = 339.29$. C_1: *Timed*– mean path length $= 352.47$, $\sigma = 230.65$; *Adaptive*– mean path length $= 544.68$, $\sigma = 391.45$; *Infinite*– mean path length $= 656.24$, $\sigma = 473.23$. $C_{1.5}$: *Timed*–mean path length $= 669.80$, $\sigma = 508.69$; *Adaptive*– mean path length $= 1010.12$, $\sigma = 721.34$; *Infinite*–mean path length $= 986.14$, $\sigma = 754.75$. C_2: *Timed*–mean path length $= 1456.36$, $\sigma = 1182.65$; *Adaptive*– mean path length $= 2051.581$, $\sigma = 1556.99$; *Infinite*–mean path length $= 1925.78$, $\sigma = 1584.20$). The performance of Timed Stochastic supports our first hypothesis: there is a clear advantage to fixing the duration of the random walk phase

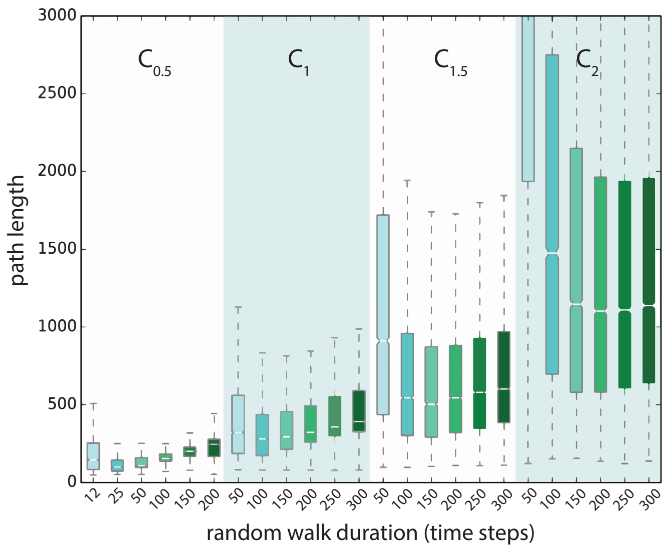

Fig. 5. Parameter sweep across $C_{0.5}, C_1, C_{1.5}$, C_2. The optimized values for the random walk phase duration are 25, 100, 150, and 250 time steps, respectively. This box plot and all following use these conventions: the box contains the Interquartile Range (IQR), Q1 to Q3, with median Q2 represented as a line. Whiskers denote $\pm$ 1.5×IQR. Fliers are omitted for clarity.

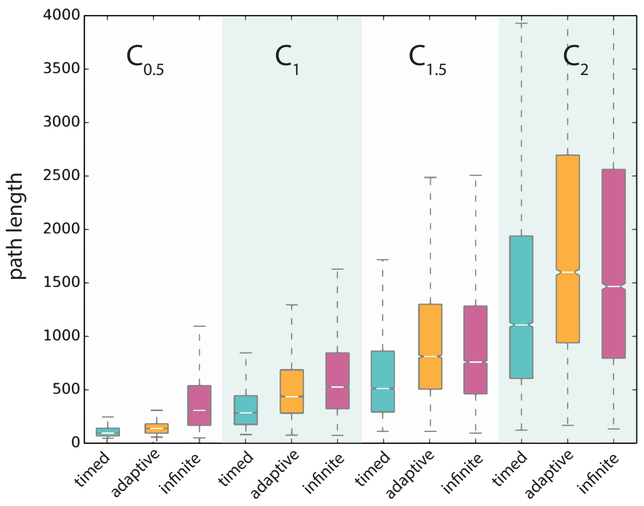

Fig. 6. Performance of the Timed Stochastic, Adaptive Stochastic, and Infinite Stochastic algorithms across cul-de-sac variants $C_{0.5}$, C_1, $C_{1.5}$, and C_2.

and optimizing it for obstacle size, in terms of both improved efficiency and lower variance. The advantage persists across obstacle scale, but is especially strong for smaller obstacle sizes. However, the strategy is required to know obstacle size ahead of time to set the right duration.

The Adaptive Stochastic algorithm uses a single general strategy, regardless of obstacle size. Even without an optimized parameter value, Adaptive Stochastic exhibits competitive performance to Timed Stochastic across $C_{0.5}$ and C_1. The algorithm is also more consistent in performance than Infinite Stochastic, having lower variance than the latter. But the Adaptive Stochastic strategy has diminishing returns. For the larger obstacle sizes, $C_{1.5}$ and C_2, the algorithm falls in efficiency; indeed, at these sizes, the simulated agent is better off performing a random walk, as Infinite Stochastic does. One possible explanation for this is that at larger scales, Adaptive Stochastic must complete multiple algorithm cycles in order to search for a workable value, and the algorithm incurs a heavy price by restarting random walks too many times. Another artificial factor contributing to efficiency is that the arena is bounded, and the area outside the cul-de-sac is reduced as the obstacle scales up. The reduction in area in turn reduces the price Infinite Stochastic pays for doing a long random walk after having escaped the cul-de-sac. Regardless, Adaptive Stochastic seems to exhibit the most advantage for smaller obstacles. Furthermore, it is likely that ants performing cooperative transport in a natural environment encounter smaller complex obstacles, such as leaves and twigs, with greater frequency than larger ones. In this case, trading off efficiency at navigating larger obstacles for quicker success at smaller ones, as Adaptive Stochastic does, would be advantageous. Overall, we see mixed results in support of our second hypothesis. Adaptive Stochastic does not pay a high cost for searching for an optimal correlated random walk duration at smaller sizes, and exhibits notable flexibility. At these sizes, there is also a clear advantage in terms of efficiency over simply random walking in terms of both shorter mean path length and lower variance. However, the advantage decreases with scale.

B. VARYING CUL-DE-SAC SHAPE

To further examine the effect of Adaptive Stochastic's adaptive parameter on performance, we look at varying obstacle shape by slicing the basic cul-de-sac C_1 in half. We call the resulting cul-de-sac C_{1S}. We compare the performance of Adaptive Stochastic and Timed Stochastic using the optimized parameter for C_1 (C_1-opt.) on C_{1S}.

In this scenario, both algorithms must break symmetry and choose either the left or the right direction upon encountering the back wall of the C_{1S} obstacle. A choice of left leads to an easy exit; conversely, a choice of right leads the agent into the right corner, triggers the failure condition of Bug0, and results in a switch to the random walk phase. We expect that when this happens, starting with a smaller parameter value, as Adaptive Stochastic does, should provide an advantage. While Timed Stochastic must random walk for 100 time steps after encountering the back right corner of the obstacle, under the same conditions, Adaptive Stochastic will Go North after random walking for just 5 time steps, and then has a chance to encounter the back wall and choose left to successfully navigate the obstacle with minimal path length from that point. On the other hand, if Adaptive Stochastic repeatedly chooses right, it needs to complete multiple algorithm cycles to increase its random walk duration until it can successfully navigate the

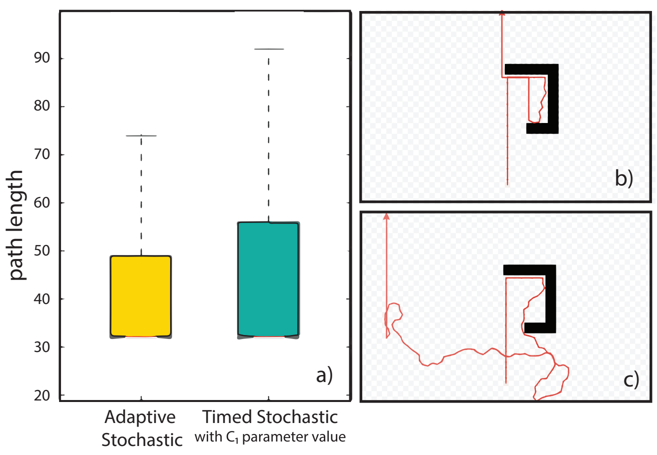

Fig. 7. Performance of Adaptive Stochastic and Timed Stochastic on variant cul-de-sac obstacle shape C_{1-S}. a) Adaptive Stochastic is the stronger performer. b) Sample trajectory taken by Adaptive Stochastic. The variable parameter allows it to recover quickly from a wrong choice of direction. c) Sample trajectory taken by Timed Stochastic.

right arm. Our results show a statistically significant difference in performance between the two algorithms (Kruskal-Willis H-test; H = 51.1, df = 1, p < 0.0001). The Adaptive Stochastic algorithm also exhibits lower mean path length and variance (*Adaptive*–mean path length $= 44.97$, $\sigma = 18.98$; *Timed (C_1-opt.)*–mean path length $= 63.25$, $\sigma = 51.58$). The median values for the two algorithms, however, are identical (median path length = 32), due to the fact that both algorithms have an equal chance of choosing left upon first encountering the back wall and reaching the goal with minimum path length (*Adaptive* and *Timed (C_1-opt.)*–minimum path length = 32). This optimal path length dominates Q1 of the data (Fig. 7).

On this obstacle variant, Adaptive Stochastic benefits from its flexibility; it can recover from choosing right during the shorter duration random walks. Overall, we see further evidence to support our second hypothesis: the flexible parameter employed by Adaptive Stochastic drives it to a better overall performance than Timed Stochastic when the target obstacle's shape is varied because it is able to adapt to the circumstance it is faced with.

C. TRAJECTORY COMPARISON

The Adaptive Stochastic algorithm was designed to mimic the behavior of the crazy ants described by McCreery et al., and to test the flexibility and generality implications of the behavior that the authors suggested. However, some aspects of the Adaptive Stochastic algorithm are not directly derivable from experimental observation – especially aspects of the random walk, such as the angular correlation or how durations of random walks are adaptively increased. A visual comparison reveals a clear qualitative similarity between sample trajectories generated by *P. longicornis* and those generated by the Adaptive Stochastic algorithm when we look inside the cul-de-sac (e.g., see Fig. 1b as compared with Fig.1c and Fig 3d). The move towards goal, symmetry-breaking, and perimeter following behaviors exhibited by the ants are also well-replicated by the Adaptive Stochastic algorithm. However, outside the cul-de-sac, we observed some clear and unexplained differences in trajectory behavior. In particular, the ant groups in McCreery et. al rarely re-enter the obstacle after having exited, and seem to have a more smooth and direct path towards the goal after exiting. In contrast, our simulated trajectories frequently re-enter the cul-de-sac. To investigate further, we present the number of cul-de-sac re-entries as a quantitative metric and compare the number of re-entries taken by the crazy ants and the Adaptive Stochastic algorithm.

We define a re-entry as having two stages–an exit and a re-entrance. The agent is considered to have performed an exit if it has passed the southernmost y-coordinate of the obstacle. A re-entry has occurred if it the agent is in the exit stage and passes the northernmost coordinate of the southern wall of the obstacle. Working from the experiment videos from McCreery et al., we manually counted the number of times that the ants re-entered the cul-de-sac. Re-entries are rare and were observed in roughly 16% of experiments (3/19). In just one of these three experiments did the ant group make multiple, clear re-entries; in the other two experiments, the ants barely passed the lip of the obstacle.

Our Adaptive Stochastic algorithm, on the other hand, causes the agent to re-enter the cul-de-sac at least once in 45% of trials (9/20). It is not immediately obvious why this is the case. To investigate, we first examined the relationship between the correlated random walk angle (θ) and re-entry frequency. The intuition behind this is that the value for θ determines the width of loops in the random trajectory; a large θ allows the trajectory to turn by a similarly large amount each step. Therefore, a random walk that exits the cul-de-sac could end up quickly looping back into the narrow opening. Based on the looping of the original ant trajectories inside the cul-de-sac, we approximated the value for the correlated random walk θ as 60 degrees. By simulating different values, we found a direct correspondence between θ and the number of re-entries (Fig. 8a). A correlated walk with $[-20°, 20°]$ degree interval, for example, substantially reduces the number of re-entries. However, using small values for θ produces long, jagged simulated trajectories that are unlike those of the ant groups, and thus do not model the type of random walk performed by the ants in the experiment video trajectories (compare Fig. 1b to Fig. 8b-c).

An alternative possibility is that the ants employ a more complex algorithm that changes once they have exited the cul-de-sac. It is known that *P. longicornis* transport groups tend to have many escort ants that do not engage in transport of the bait, but seem to follow the object [4]. There has also been recent evidence that in some cases, ants may lay short pheromone trails to guide a transport group [9]. One hypothesis is that these escort ants may act as informed individuals and play some role in guiding the transport once it has exited. While it is currently unknown how groups of ants avoid re-entry, what is clear is that the Adaptive Stochastic

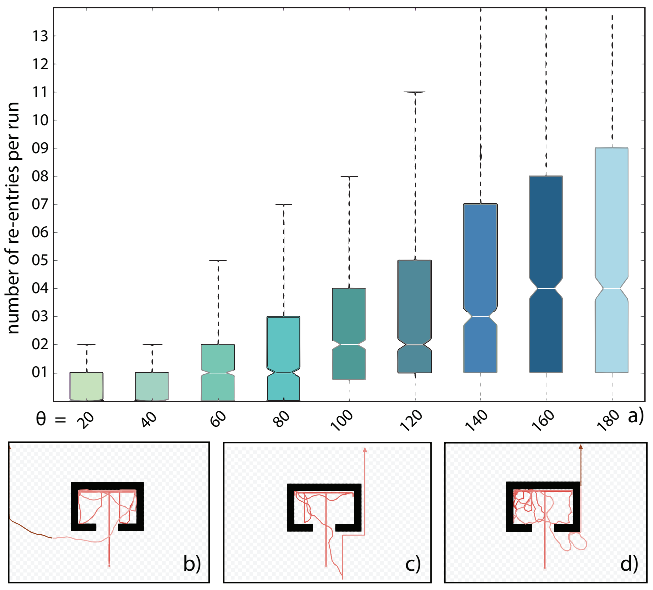

Fig. 8. Relationship between θ, re-entries, and generated simulated trajectories. a) Re-entries per run observed for increasing values of θ. b) Trajectory generated by $\theta = 20$ degrees. c) Trajectory generated by $\theta = 40$ degrees. d) Trajectory generated by $\theta = 60$ degrees.

algorithm by itself does not capture the full behavior of the cooperatively transporting ants.

IV. Conclusion

In this paper, we presented a computational model of an adaptive navigation strategy, based on the experimental study by McCreery et al. of *Paratrachina longicornis* (crazy ants) cooperatively transporting bait in the presence of obstacles. The crazy ants successfully navigated multiple obstacle types and demonstrated a range of component behaviors, including move towards goal, symmetry breaking, perimeter following, and random walks that increased over time. From the experimental descriptions of these component behaviors, we developed the Adaptive Stochastic algorithm to model the presumed ant behavior, and compared the algorithm's efficiency and flexibility to that of two other strategies. We showed that for smaller obstacle sizes, compared to pure random walks or even random walks with fixed duration, the Adaptive Stochastic algorithm is the most robust to varying object sizes and shapes; with no parameter tuning, it achieves efficient performance competitive with an optimized stochastic algorithm. The Adaptive Stochastic algorithm also models the ants' movement inside the cul-de-sac trap well when a qualitative comparison of trajectories is performed. Yet, in a surprising result, the trajectories outside the cul-de-sac are markedly different. Our results suggest that a single algorithm may not be able to reproduce the entire range of behaviors observed in McCreery et al.

Our computational study suggests several directions for future investigations. Experiments can be conducted with *P. Longicornis* on larger cul-de-sacs, such as our C_2 variant, to investigate if the ants employ a different strategy when the scale of the obstacle is increased dramatically. The same experiments can also be extended to ant species that are strong at cooperative transport [4], [10], to determine if other strategy variations exist; if so, they can modeled using our framework for simulation and algorithmic comparison. Most importantly, the computational study shows that the Adaptive Stochastic algorithm is not a complete description of the crazy ants' behavior, and more experimental observations are needed to understand the behavior outside of the cul-de-sac obstacle. Our computational model also suggests new directions for the development of algorithms for robots navigating in the presence of obstacles [10], [11]: for single robots who cannot measure distance accurately to the source, or, in future work, translating group-level behavior to individual behaviors for collective robotics. The Adaptive Stochastic algorithm pays a cost in efficiency compared to BugD, but relies on simple direction sensing and adaptive random walks to achieve reasonable efficiency.

V. Acknowledgements

This work was supported by the Wyss Institute for Biologically Inspired Engineering.

References

[1] Wojtusiak J, Godzinska E, Dejean A (1995). Capture and retrieval of very large prey by workers of the African weaver ant, Oecophylla longinoda (Latreille 1802). Trop Zool 8:309-318.
[2] Camazine, S, Deneubourg, JL, Franks, NR, Sneyd, J, Theraulaz, G, Bonabeau, E (2001). Self-organization in biological systems. Princeton: Princeton University Press.
[3] McCreery HF, Breed MD (2014). Cooperative transport in ants: a review of proximate mechanisms. Insectes Sociaux 61:99-110. doi: 10.1007/s00040-013-0333-3
[4] Czaczkes TJ, Ratnieks FLW (2013). Cooperative transport in ants (Hymenoptera: Formicidae) and elsewhere. Myrmecological News 18: 1-11.
[5] Gelblum A, Pinkoviezky I, Fonio E, Ghosh A, Gov N, Feinerman O (2015). Nature Communications 6, Article number: 7729.
[6] McCreery HF, Dix ZA, Breed MD, Nagpal R (2016). Collective strategy for obstacle navigation during cooperative transport by ants. Journal of Experimental Biology 219: 3366–3375. doi: 10.1242/jeb.143818
[7] Lumelsky VJ , Stepanov AA (1987). Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape. Algorithmica 2 (1), 403-430.
[8] Kamon I, Rivlin E (1997). Sensory-based motion planning with global proofs. IEEE transactions on Robotics and Automation 13 (6), 814-822.
[9] Fonio E, Heyman Y, Boczkowski L, Gelbum A, Kosowski A, Korman A (2016). A locally-blazed ant trail achieves efficient collective navigation despite limited information. eLife 5:e20185. doi: 10.7554/eLife.20185
[10] Berman S, Lindsey Q, Sakar MS, Kumar V, Pratt S (2011). Experimental Study and Modeling of Group Retrieval in Ants as an Approach to Collective Transport in Swarm Robotic Systems. Proceedings of the IEEE, Vol. 99, No. 9, September 2011.
[11] Rubenstein M, Cabrera A, Werfel J, Habibi G, McLurkin J, Nagpal R (2013). Collective transport of complex objects by simple robots: theory and experiments. Intl. Conf. on Autonomous agents and multiagent systems (AAMAS'13).
[12] Wilensky U (1999). NetLogo. http://ccl.northwestern.edu/netlogo/. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.
[13] R Core Team (2017). R: A language and environment for statistical computing. https://www.r-project.org/ . R Foundation for Statistical Computing, Vienna, Austria.