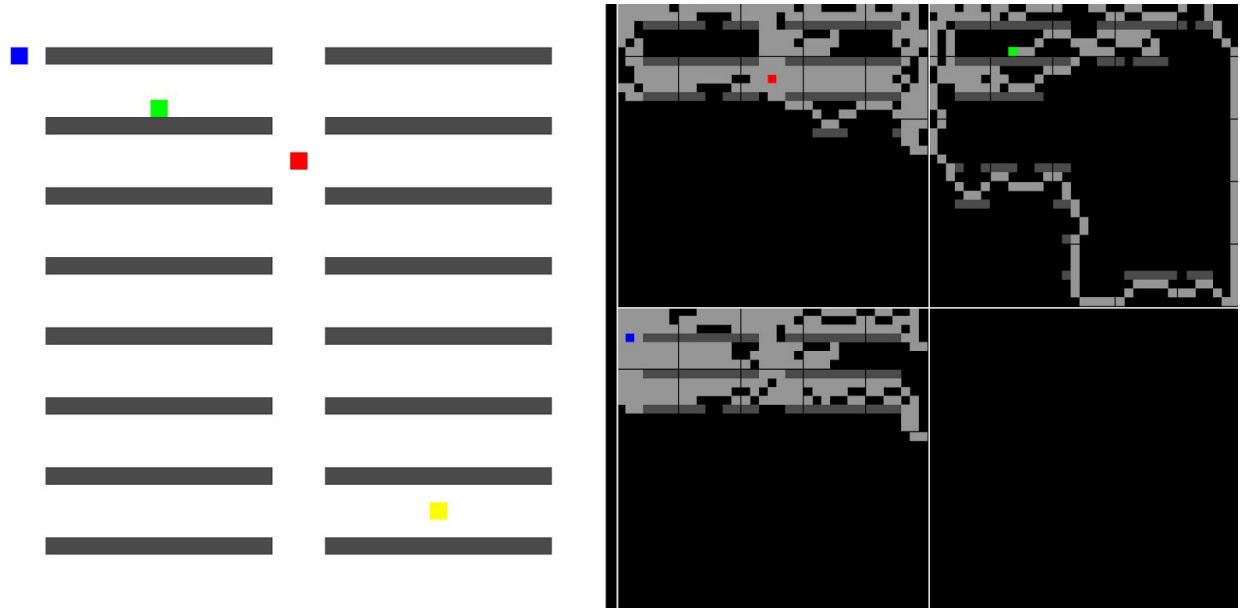
ME 498 / IB 496
Bioinspired Design



Game of Ants - AA Bots

Luis Rendon, Tor Shepherd, Faris Alquaddoomi
Amy Wissa

Submitted on May 5th, 2019

Table of Contents

Executive Summary

1. Understanding the Biological Solution (3)
    1.1. Overview of Assigned Organism (3)
    1.2. Summary of Organism Functions (4)
    1.3. Structure Behavior Function (7)
    1.4. Multi-Functional Solution Selection (10)
    1.5. Morphological Matrix- Draft (10)
    1.6. Morphological Matrix –Revised (10)
    1.7. Expert Report (12)
    1.8. “Humanization” of the Biological Solution (13)
2. Problem Identification and Understanding (14)
    2.1. Need Finding and User Definition (14)
        2.1.1. User Group & Need Finding Activities (14)
        2.1.2. Need finding and Benchmarking Results (15)
        2.1.3. User Persona (17)
    2.2. Problem Definition (18)
    2.3. Problem Specifications (19)
3. Conceptual Design (20)
    3.1. Ideation (20)
    3.2. Concept Evaluation (25)
    3.3. Detailed Final Concept (26)
    3.4. Concept Reflection (27)
4. Materialize: Design Embodiment (28)
    4.1. CFP Proposal (28)
    4.2. CFP Experimental Evaluation Plan (29)
    4.3. CFP Demonstration Results and Discussion (31)
5. Conclusions and Discussion (33)
6. References (34)
7. Appendices (35)

Executive Summary

Distributed systems are increasingly being applied to problems where consecutive task accomplishment systems fail or are inefficient. The traditional approach to designing a distributed system involves creating networks of robots fitted with expensive equipment suitable for long-range wireless communication. In contrast, our idea for a distributed system aims to be cost effective without sacrificing task accomplishment efficiency. We have designed a simulation of autonomous agents that follows simple decision-making rules for navigating their environment and searching in a complex space. The agents are capable of sharing information about the environment's layout through interactions with nearby agents. The simulated model is closely based on ant colonies where emergent behavior is shown in higher-level function without central oversight. Our system is intended for application in search and return tasks.

We explored the problem of package loss in warehouses and discovered that minimal or no efforts are made to recover lost items. Instead, warehouses cover the cost of lost items with credit and incur inventory shrinkage as a result. We decided to experiment with this problem by adapting our simulation to model our ant-inspired system.The first requirement of the system was the ability for agents to remember unknown parts of the warehouse as they explored and transmitted their memory of the explored space to nearby agents. The second requirement was for agents to collectively exhibit emergent behavior that lowered solve time when more agents were operating in the warehouse. We tested for the onset of emergence behavior by analyzing trends in the average time it takes a set number of agents to locate one lost package in the simulated warehouse.

The experiment consisted of agents programmed to seek one lost item, avoid obstacles, and be pulled towards unexplored areas of the simulated warehouse. We designated a fixed area for the lost item and fixed cells for the obstacles. We then ran 800 trials, varying between 1 and 9 agents in each trial, to accurately identify trends in solve step data for each set of agents. We exported data on the solve steps, agent number, map size, and number of agent interactions to excel and analyzed average solve step and number of interactions data in relation to a set number of agents. We also defined a metric - the series ratio - for identifying the onset of emergence in the simulation.

The results of our study show distinct trends relating solve steps and agent interactions to agent number. The number of average solve steps generally decreases with increasing agent count. The number of interactions between agents increases with increasing agent count and the correlation between solve steps and number of interactions also increases with increasing agent count. The standard deviation of solve times decreases with agent count. The series ratio metric indicates the onset of emergent behavior from 9 to 10 agents in a 35 x 35 cell map. We conclude that the number of agents necessary for exhibiting emergent behavior equals 10 for our conditions, but this value varies with map size and map layout.

1. Understanding the Biological Solution

1.1. Overview of Assigned Organism

Our team has researched ants to understand how certain ant functions may be applied to engineering problems. Ant society consists of a complex web of local interactions between ants specialized to accomplish specific tasks. The caste structure of an ant colony broadly includes reproductive males, reproductive females, and non-reproductive females. Reproductive females are usually referred to as queens and non-reproductive females may either be classified as workers, soldiers, or both. The specialization of function which individual female ants exhibit emerges from modifications to their development induced by growth hormones. The collective function of a brood is termed ‘emergent behavior’ because new methods of task accomplishment arise when ants scale local interactions.

We have explored several ant functions: one relating to individual ants and five relating to groups of ants. Individual ants are dynamically stable, enabling them to travel across rough terrain without toppling. The ants balance their center of mass by having three of their six legs always in contact with a surface during locomotion. The five other functions we explored relate to foraging processes, defense mechanisms, and ant communication. Worker ants optimize the delivery of food to their nests via local interactions that determine routes to more abundant food sources and establish efficient worker traffic flow in the process. Fire ants display emergent behavior when they conglomerate into multi-ant structures to ward off aggressors. All ants interpret signals sent from other ants and insects and read chemical signatures to distinguish useful plants and fungi. Ants also regulate their own communication for mutual benefit within their brood. These functions provide us with a set of principles to utilize when solving an engineering problem.

1.2. Summary of Organism Functions

1.2.1 The Regulation of Ant Colony Foraging Activity without Spatial Information

This paper was an analysis of an existing system, that system being the collective behavior exhibited by some insects, more specifically ants. The paper demonstrates a model of a Poisson process that closely predicts the foraging behavior of harvester ants according to real data collected in 2009 and 2010. Other papers have also reported on the dynamics of local interactions within social insects, creating models to explain the effect of colony size on the colony's behavior. This paper presents how foraging behavior is regulated by antennal contacts of returning and outgoing foraging ants. Harvester ants are able to direct the behavior of the entire colony using only local interactions. The ants accomplish this feat when incoming foragers communicate with outgoing foragers to continue searching. The more ants returning to the nest, the more food there is to be found, and so the more ants forage. The opposite is also observed to be true. When food is scarce, the ants take longer to return to the nest, and so the ants do not leave the nest as often to search for food. The article is well-written, logical, and assumptions seem correct. The article also addresses possible sources of error for their model due to unmeasurable effects of weather and tunnel configuration on foraging speed.

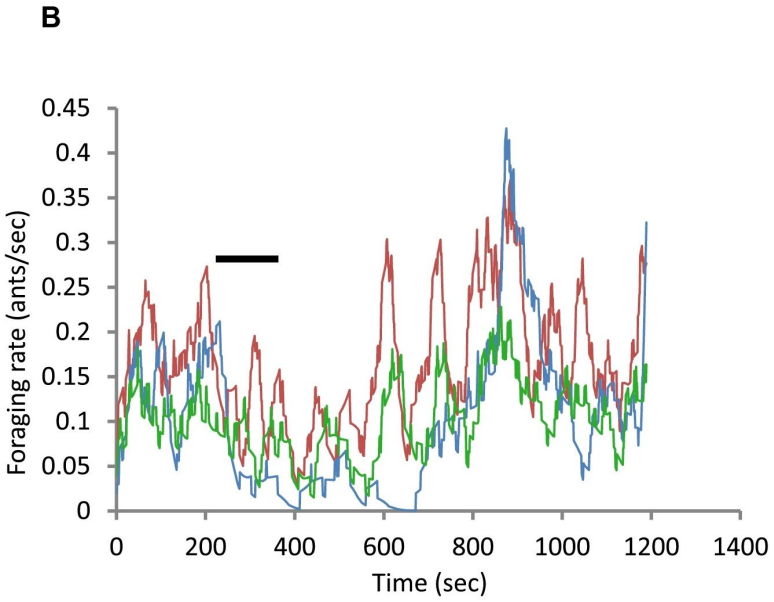


Figure 1. Plot of Foraging Rate vs Time.

In Figure 1, the red line corresponds to the observed rate of returning forager ants, the blue line is the observed rate of outgoing foragers, and the green line is the simulated rate of outgoing foragers. As the graph shows, the outgoing and returning rates of foraging ants closely follow one another. The period of 240 seconds to 420 seconds reflects when returning ants were removed from the experiment, causing an artificial drop in returning ants and a corresponding drop in outgoing ants.

1.2.2 Self-organized lane formation and optimized traffic flow in army ants

In this article, the authors investigate the emergent traffic behavior of groups of ants. By observing the behavior of *Eciton burchelli* as they travel in large pheromone paths, the researchers were able to formulate a model that simulates individual ants.

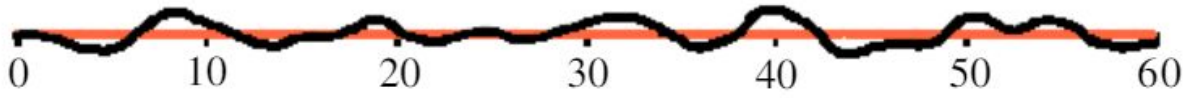


Figure 2. One simulated trajectory of an ant traveling on a pheromone path

By applying several decision-making control schemes to each ant agent, a network of ants is compiled. Without applying further control, a group of ants is able to collectively decide on a direction of trail to move in. Additionally, ants returning from the destination (moving the opposite direction as the rest) tend to occupy a center lane (of 3). Interestingly, this phenomenon manifests in humans as well, particularly on crowded sidewalks or similar. However, human lane-count scales with size of sidewalk, while *Eciton burchelli* always form 3 lanes.

1.2.3 Mechanics of fire ant aggregations

Fire ants, as this paper posits, can behave similarly to a liquid. Researchers explored properties of masses of worker ants, from their ability to spread to their resistance to loading. By forming a group of ants into a cluster, researchers were able to investigate them more similarly to a material, by deforming, dropping objects into, and pouring them. When a cluster is laid on a surface, it tends to expand similarly to viscous fluid, like honey.

1.2.4 Quantifying Dynamic Stability and Maneuverability in Legged Locomotion

This research paper was another analysis of an existing system, specifically the six-legged locomotion of ants and other arthropods. The paper is related to and references other papers on animal locomotion and dynamical systems. It's assumptions seem logical and correct. The paper attempts to define what stability means and how to quantify it. It also discusses some experiments quantifying the maneuverability of arthropods. The paper explains that a stable system is one that can return to a balanced position after perturbations. Ants use their six legs to keep themselves stable. During motion, three legs are kept in contact with the ground to maintain a favorable center of mass. Ants maintain their center of gravity by positioning their legs in a triangle so that their center of gravity is in the center. The paper was clear but a bit verbose.

1.2.5 The Interactions of Ants with their Biotic Environment

This article falls into the category of an analysis on ant-ant, ant-plant, ant-microbiota, and ant-fungi relationships. The article attempts to summarize how these relationships serve purposes ranging from symbiosis to parasitism. The context of the article represents a compilation of research findings on these relationships. The article contributes a broad review

of relationships between ants and their biotic environment and how biotic interactions are based on chemical signalling.

The degree of communication between ants within their own species and among other organisms occurs by the transfer of chemical signatures. Semiochemicals allow for the identification of ants within a specific caste structure and reinforce behaviors supporting a division of labor in ant society. Pheromone signals communicate worker tasks and sexual interaction. Pheromones most generally mediate cooperation and conflict; they are interpreted by ants in conjunction with semiochemical signatures, threat presence, and food supply.

Ant-ant communication is specifically mediated by a semiochemical class of molecules termed cuticular hydrocarbons (CHC). CHC chains are bound to the ant exoskeleton to waterproof the ant and present a permanently visible communication pathway. CHC chains serve protective functions and enable a discernable division of labor within a colony. Broods operates in mutual self-interest in response to queen CHC profiles and appeasement pheromones. CHCs also allow for parasitic behavior by masking the identity of ants with determined roles, enabling them to freeload off available resources.

Many species of ants forage for the purpose of gardening and generating a food supply. Certain seed coats present chemical cues to ants for ants to fertilize the seed and utilize the plant for nest structural stability and fungus fertilizer. Fungi are usually the end product of plant foraging and constitute the ant's food source. Ant-fungi and ant-plant relationships are symbiotic in that they benefit the growth and proliferation of all parties involved. The article is clear, asserting well-documented phenomena that provide correct, consensus views on ant interactions involving the major agents in their environment.

1.2.6 Genetic Constraints on Dishonesty and Caste Dimorphism in an Ant

This article falls into the category of an analysis of ant-ant chemiosignalling traits. The article considers a cost-benefit analysis of ant signalling and the value of honesty when generating a chemical cue for food, identification, or aggression. The article contributes an explanation of ant signal honesty within the context of natural selection, relating the hypothesis that signals undergo natural selection to benefit the sender by modifying the behavior of the receivers and that ants transfer reliable signals to their brood and queen - seemingly ignoring self-interest.

The article postulates that exaggerating signal strength is ineffective to generate more value for the sender per unit of signal strength invested. For example, a worker ant will not amplify its hunger signal to obtain more food if unnecessary for its survival. The article clearly explains this phenomenon from the standpoint of competitive advantage - an ant colony which permits overall effective cooperation can improve the queen's fecundity and therefore increase the proliferation of the brood. There is evidence supporting the correctness of this explanation. CHC chain length has been shown to affect queen fertility and worker benevolence towards the queen, indicating the presence of mutual self-interest within ant colonies. In principle, ants tune their signalling to reflect actual levels of need which preferences cooperation and emergent behavior.

1.3 Structure Behavior Function

1.3.1 Emergent Behavior

Ants are able to work together to complete objectives that benefit the entire colony without a central commander. Ants specialize to accomplish certain tasks. They can exchange information with others immediately around them to decide when their job is done and to move on to the next job. They communicate through various methods involving physical interactions and chemosignaling. From these interactions, they react to a set of simple rules without deviating for selfish reasons, so they are able to do what's best for the colony. The core principle are the chemical coatings on an ant's cuticle. Ants can transfer these chemicals either through antenna interactions or by leaving pheromone trails for other ants to follow.

1.3.2 Ability to Explore and Live in Diverse Environments

Ants explore places far from their nest to find as much food and resources as they can. They are able to reach these far distances because of their ability to navigate unfriendly environments. Ants can dig through many materials with their powerful jaws and muscles, and their six legs provide them with stability to carry relatively large objects. The core principle here is the ant's light exoskeleton and steady posture that allows their muscles to turn their focus away from supporting the ant and to manage other loads. Their small weight to muscle ratio gives them a great amount of strength relative to their body size. They also carry loads on a concentrated part of their body and create large moment arms to carry heavy things.

1.3.3 Optimization of traffic lanes from individual decision-making

It is not inaccurate to model ants as decision trees, yet collectively, they present behaviors that would suggest cooperative communication. In the case of *Eciton burchelli*, simple rules lead to highly efficient traffic patterns. The formation of efficient traffic lanes allows the society of ants to locate food without central leadership, increasing chances for survival. The mechanism is based on decisions made by individual ants. Ants leave pheromone trails, and when an ant discovers food, it retraces its steps repeatedly. Since ants are modeled as deciding to follow a trail probabilistically (based on concentration of trail), more ants will tend to follow and increase the concentration, causing a positive feedback loop.

1.3.4 Conglomeration into non-newtonian fluid-like material

While emergent behavior can result in traffic or navigation, it can also lead to liquid-like properties in a large mass of ants. By interacting and linking to each other with adhesive pads on their legs, fire ants can flow, drip, and perform other liquid functions. This can help with forming bridges, moving around as a group, and other evolutionary advantages. The ants do this by forming a "weave" which other ants can cluster around. The ants also improve the cohesiveness of the structure by packing smaller ants in gaps that are formed when larger ants link together.

1.3.5 Ant Communication with Biotic Environment Determines Actions

Individual ants must interpret signals within their biotic environment to choose what relationship they should establish with the agent emitting the signal. This choice constitutes a function of each individual ant to discern between symbiotic and parasitic agents. Ants and their associated agents utilize chemiosignalling, both passively and actively, by emitting hormones from glands distributed across body. Gland hormone emissions activate receptors on receiving ant antennae. Passive chemiosignalling refers to communication between agents that is interpreted from structures on the agents. Cuticular Hydrocarbons (CHCs) are the primary structures utilized in ant/ant and ant/plant passive chemiosignalling. Active chemiosignalling refers to communication akin to talking. Chemicals including pheromones, appeasement hormones, and growth hormones are the structures used to accomplish this communication. Formic acid is the best known example of active chemiosignalling and allows for ants to follow each others' trails when foraging or scouting.

1.3.6 Ant/Ant Signal Fidelity Balances Brood Health and Selfishness

When ants signal, they must choose between selfish benefit and inclusive benefit within their brood. Signal fidelity refers to the relative "honesty" of a signal - or how well it represents the ant's condition. Regulation of signal fidelity is a function of ants that allows them to cooperate within a brood without abusing or losing out on brood resources and benefits. Queen ants produce workers with undesirable fertility traits which they attempt to suppress using chemiosignalling. Queen emission of fertility hormones from their sex glands constitute an energy investment for the queen that may leave her weakened to brood aggression. Workers with high fecundity may attempt to assassinate the queen - aggressive behavior which queens also try to suppress by a different chemiosignalling pathway. Suppression of worker fertility and aggressiveness is done by releasing pheromones, but both behaviors constitute valuable features of the brood which must not be completely suppressed.

1.3.7 Summary of Functions, Structures, and Behaviors of Ants

The below table summarizes our exploration of ant functions and their associated structures and behaviors. Columns describe structures and behaviors related to each listed function. Specific ant species in which a function is described are also listed.

Table 1. Ant Function and Related Structural and Behavioral Components

Species	**Various Ants**	**Various Ants**	**Eciton Burchelli**	**Solenopsis geminata**	**Various Ants**	**Lasius Niger**
Function	Ants perform complicated tasks as a group.	Ants can reach far distances and explore wide-range of environments	Ants form optimized traffic by individual decision	Ant can agglomerate into a mass capable of accomplishing certain tasks.	Ants process cues from plants, fungi, and other ants to choose a symbiotic or parasitic relationship.	Signal fidelity is regulated to balance brood benefit versus individual benefit.
Behavior	Ants observe others immediately around them and respond with changes in behavior based on a set of simple rules.	Ants form living bridges with each other, walk through difficult terrain, and carry heavy objects.	Trails left by foraging ants reinforce the flow of ant traffic by depositing pheromones which signal a path for ants to follow.	Ants link their bodies on a large scale to create a fluid-like ball of ants. The ball can span gaps to form bridges.	Ant/ant signalling regulates caste roles and if they are carried out to the benefit of the brood. Ant/plant communication identifies seeds for use in fungi gardens (ant/fungi symbiosis) and nest support.	"Honesty" of ant signal changes in response to signalling by other ants to mediate aggressiveness, resource allocation, and fecundity. Signalling effect is discriminatory - based on caste.
Structure	Ants use their antennae and chemical signaling to communicate with other nearby ants. The core principle is that ants are unselfish and follow simple rules that lead to what's best for the colony.	Ants have six legs and strong muscles to keep themselves stable. The core principle lies in their light exoskeleton and overbuilt bodies that give them great relative strength to their small size	Pheromone trails left by individual ants. The core principle is the positive feedback loop created by ants where pheromone trails grow stronger when more ants use the path to lead to a large food source	The bodies of individual ants which act as scaffolding for larger structures. The core principle is the choice of ants to either join or cross the bridge based on the stability of the structure. If stability is limited, ants choose to join and if stability is assured, ants cross.	Ant/ant signalling relies on chemosignals like appeasement hormones and pheromone trails. Ant/plant communication relies on CHC profiles present on plant seed coats. Ant/fungi symbiosis recognizes fungi's value as nutrition and building material. The core principle is label interpretation and active labeling of ant's environment.	Queen pheromones and CHC profiles indicate ant identity. Worker ants lose fecundity in response to queen pheromones. Ants display symbiotic or parasitic behavior with other ants based on their unique CHC coating. The core principle is balancing the benefit (profit) of signal honesty with energy invested into the signal (cost).

1.4 Multi-Functional Solution Selection

1.4.1 Emergent Behavior: Optimizing Traffic Flow

Ants' emergent behavior will be the main inspiration for our design project. It is well-documented that ants live in large colonies and accomplish sophisticated tasks by working together. Individual ants are not very smart but they follow each others lead without any single ant actually leading. We would like to embody this principle of creating individual machines that together become more than the sum of their parts.

1.4.2 Ant Communication with Biotic Environment

Our design project must incorporate knowledge of how ants communicate with agents other than ants to ensure the survival of themselves and their colony. The decision to forage in a specific area, use a specific plant, or establish a particular type of relationship with another species of ant or insect determine the size and health of a brood. Emergent behavior may require biotic relations to initiate and persist, so exploring the function of ants to establish symbiotic or parasitic relationships via chemiosignalling is an important component of our project.

1.5/1.6 Revised Morphological Matrix

Table 2. Morphological Matrix Comparing Ant, Slime Mold, and Honey Bee Function

Organism	Ant	Slime Mold	Honey Bee	Core Principle
Function 1: Emergent Behavior	Emergent behavior in optimizing path to food source- Ants interact with their immediate neighbors using chemiosigniling and their antennas and respond according to a simple set of universal rules. The entire colony is able to change its behavior slowly according to what other ants are doing. In optimizing a path for food, ants leave pheromone trails released from special glands. Trails that are used often may lead to more abundant food sources, so the pheromone trail is reinforced. This created a positive feedback loop where the strongest pheromone trail leads to the most abundant food source.	The slime mold is a protoplasm that contains a network of tubular elements. The tubes service transportation systems for the cells of the protoplasm to shuttle nutrients and chemical signals. The tubes react to changes in internal flux so long, unused, tubes shrink. Similarly, short, fruitful tubes expand. Large tubes have greater flux, further increasing their size, and smaller tubes have less flux, causing them to dry up and shrink. This leads to a feedback cycle where tubes optimize themselves.	When a swarm of bees is looking for a new nesting site, scouts are sent out to search while the rest of the swarm stays behind with the queen. When a scout finds a suitable location, the scout "dances" to signal the swarm. The duration and specific movements in the dance relay various qualities of the location. A better location yields a fiercer and longer dance. Other bees may join the dance if they agree it is a suitable location for a nest. This leads to the best nest location being found with high accuracy, since longer and more intense dances attract more bees.	These organisms all use a feedback loop to optimize searches. They send chemical or visual signals to exchange information about which path is most desirable. Individuals do not exhibit complex behavior. Individuals follow any interpreted signal until the feedback loop leaves only the strongest signal and the most desirable path.

Function 2: Reacting with Biotic Environment	Ants use cuticular hydrocarbon coatings (CHCs) to represent ant caste and individual ant fertility. CHCs cover the surface of all insect exoskeletons and are even present on some seed coats. CHCs are recognized by the ant olfactory sense and may be transmitted via grooming, mouth-to-mouth feeding, or nest soil deposition. CHCs signal the identity of ants and other insects, allowing ants to take proper measures in reacting to their environment.	Slime molds sense food densities within their environment from the concentration of nutrient uptake in their cytoplasm. Once a slime mold has found food in their environment, they “remember” the location of the food sources. Parts of the slime membrane which stay in contact with their surface environment guide the membrane of the slime towards vein formation along nutrient-rich paths.	Honey bees also use CHCs to communicate eusocial roles between bees in a colony. CHC transmission is notably conveyed by honey produced by workers and the queen. Worker jelly is distinct and provides workers with their nutrients. Royal jelly causes suitable females to undergo mutation and become queens. Within a bee colony, the biotic environment which bees mainly interact with is other bees in their colony.	These organisms interact with their biotic environment by releasing chemicals or leaving trails to signal nutrients, identity, or fertility/aggression. The principle being utilized is labeling of the environment to facilitate interactions and reactions.

Experts:

Ant - Dr. Andy Suarez

Slime Mold - Dr. Simon Garnier

Honey Bee - Prof. Gene E. Robinson

1.7. Expert Report

Our group met with Dr. Andy Suarez and Dr. Gene Robinson to gain perspective on how ant emergent behavior and biotic interactions may be simulated.

Dr. Gene Robinson is the principal investigator at The Robinson Lab honey bee research group. We decided to interview Dr. Robinson because his research focuses on the study of social life in Western Honey Bees. We are interested in how ant emergent behavior is facilitated by social interactions between individuals, and bee social castes are a good parallel to ant castes. Dr. Robinson is the director of the Institute for Genomic Biology and the Bee Research Facility, so he is a highly qualified expert on bees and biology. After our interview, we gained new leads for finding useful models of collective behavior exhibited by eusocial insects. Dr. Robinson also shared his understanding of emergent behavior in insects that gave us a new perspective to how the function can be used to solve problems. He mentioned how emergency responses can be delayed due to slow relay of information by a central command. With this new knowledge we gained a new perspective to the type of problems we can solve, and we may make some changes to our design statement.

To gain further insight into the importance of job specificity and castes in ant emergent behavior, we spoke to Dr. Andy Suarez, Professor in the departments of Animal Biology and Entomology. Dr. Suarez is an expert on ants, and his current research focuses on energy storage of trap-jaw ants and the effect of body size variation. After bringing up the various models we found that researchers use, Dr. Suarez clarified for us that scientists are uncertain as to the exact mechanism used in nature, but researchers use three basic models: exhaustive work search, encounter frequency, and recruitment trail.

Exhaustive work search is the simplest decision-making scheme. Each ant wanders, searching for open job positions. If it starts by the nest, it looks for openings watching the nest. If it doesn't find a job, it works its way outward, selecting the next job function. The encounter frequency model means that ants make decisions based off how many antennae interactions they see. The recruitment trail scheme is most similar to a feedback loop in control theory. If ants encounter ants that have found food, for example, they will also seek the food source, amplifying the response. Researchers combine these three models and then tune relative gains to match closely what is observed.

Dr. Suarez agreed with our understanding of emergent behavior, but he also contributed a lot of valuable information. He agrees that emergent behavior can inspire more computationally efficient solutions to optimization problems. We will incorporate the three common models explained into our solution.

1.8. “Humanization” of the Biological Solution

The function of emergent behavior across ants, slime molds, and honey bees utilizes the core principle of positive feedback to reinforce actions that lead to desirable outcomes. Our team is focused on applying the principle of useful positive feedback. For instance, *how could we implement positive feedback systems in a solar panel plant to maximize energy harvested from the sun based only on local light intensity and light angle of incidence inputs?* The variability of sunlight intensity and the sunlight angle of incidence is analogous to the abundance and location of an ant food source. Sunlight supply varies with cloud cover and the solid angle of greatest intensity shifts locations. Available food sources similarly vary in abundance and location. This question passes the Grainger test because it asks how positive feedback loops could be applied to an interesting and valuable design question.

The function of biotic interactions relates to the core principle of labeling and interpreting labels in the environment to determine an appropriate response. A totally self-contained agent capable of analyzing and contributing information to another system can facilitate emergent function in collections of similar agents. *How could we design a self-contained robot which interacts with other robots in a warehouse to optimize the flow of commodities in a supply chain?* Analogies exist between supply chain flows and ant foraging lanes.This question passes the Grainger test because it references the principle of labeling and interpreting labels in local interactions to inform the design of a useful system.

(1) We can implement positive feedback systems in solar panel plants by having the panels ‘remember’ the intensity of sunlight from various directions and tilt to directions with the historically highest average intensities. This method reduces the need for panels to always track the motion of the sun and cuts down on energy costs associated with rotating the panels. (2) Another application of this system is in wind turbines that ‘remember’ the average direction of strongest wind flow. (3) Fully autonomous robot supply chains could be implemented by programming individual robots to appropriately address their local environment. If robotic workers could locally communicate their position, velocity, acceleration, and destination, they could modify their speed and pick-up/delivery pathway to increase the rate of supply flow.

The first and second application for feedback loop implementation in power generation are useful for increasing energy output and cutting down on energy costs associated with running the systems. This benefit justifies the approach and the method of feedback loop implementation is itself justified by its efficiency in optimizing ant foraging. The supply chain application is reasonable because it saves costs for the user by removing the need for central oversight. This application also removes human error from the equation because robots properly programmed with the environmental labeling/interpretation method will work towards optimizing every action in the supply chain. We intend to simulate a warehouse with ant-inspired agents capable of fulfilling roles in the warehouse in place of human workers. The agents will require no central oversight and operate on a pre-programmed set of rule.

2. Problem Identification and Understanding

2.1 Need Finding and User Definition

2.1.1. User Group & Need Finding Activities

Our user group will include manufacturing companies, shipping companies, retailers, or any other companies with large and complex supply chains that must be managed efficiently. Supply chains at large companies are managed from a central location, but an increase in performance and efficiency can be made if strategic decisions are made locally by nodes in the supply chain that interact. Decisions will be made quicker if each location can implement changes in strategy without waiting for a response from a central command. To figure out if our biological solution can be useful to these companies, we would like to interview representatives of supply-chain companies, and research these companies to learn where there is room for improvement.

To establish needs and wants, we will contact representatives of supply-chain companies to determine where emergent behavior can help them. We will describe our initial solution proposals and receive feedback. This will allow us to refine our concepts. We also want to ask them what their likes, dislikes, about their job are and what challenges they face on the job. This can help us create a user persona.

For our desk study, we plan to research different manufacturing companies to find out what their mission statements are, and what they are currently working on. This can give us insights into what our user group values and what their goals are. We can also gain an understanding of how our user solves problems, so we can come up with a product that our users will be excited to implement. By figuring out what our user values the most, we can also make important strategic decisions when it comes to adding functionality to our product.

2.1.2. Need finding and Bench Marking Results

Field Study Results

Workers in warehouses frequently experience difficulties with lost packages. We received a in-depth reply to a question we posed on reddit's *lossprevention* forum about how lost products were handled at major warehouses. Monetary losses due to lost products accrue over time and wrong shipments were mentioned as averaging about 12 per day at one particular distribution center. Product "Mis [sic] picks" also accounted for issues at warehouses where incorrect items were selected for outgoing shipments. This issue depletes warehouse stock. The reply mentioned that warehouse managers view blaming individual workers for lost product as futile because of the difficulty of pinpointing responsibility. Thresholds for lost products are set to repay the warehouse for depletion of stock, but that limit is usually set low - $700 maximum as mentioned in the reply.

We also researched *warehouse job reviews* to understand how workers dealt with package losses and tried to understand the problem in the context of how workers, warehouse managers, and customers were affected. The following empathy map summarizes our findings on challenges affected individuals face when dealing with lost products:

What was said	**What was thought**
Packages are often difficult to locate in crowded warehouses and post offices. When a package is sorted in the wrong location, it is impossible to determine if the package was lost or stolen until it is located. Blame for lost or stolen packages may be shifted to innocent workers, though usually that is not the case. It would be useful if lost packages could be routinely collected and sorted into their proper locations.	Active workers cannot be tasked with locating every lost package. Warehouse workers often think of requests to find lost packages as secondary to their roles in the warehouse. Clients who are affected by lost packages think it is the obligation of warehouses to locate the packages, but comprehensive systems for searching stocks for lost packages are usually not available.
What was felt	**What was done**
Dealing with lost packages is a problem for the worker, the warehouse, and the product recipient. Workers who lose shipments feel tension grow between themselves and warehouse managers and incur losses for the warehouse from lowered productivity and the material cost of the lost unit. The customers feel irritated and this affects the morale of the warehouse workers.	Lost packages may be replaced with identical products in an outgoing shipment. If a customer is sending a unique product through the warehouse, a search request can be made. However search requests for missing items may take long to fulfill and are often marked as unactionable when a certain period of time has passed. Credit is issued to the warehouse as insurance for lost items.

Takeaways from Desk Studies

We conducted desk studies to understand the internal operations of warehouses. Our project is focused on the application of emergent phenomenon and environmental interactions in the design of an automated supply chain warehouse. Warehouse types are categorized according to their function. Storage warehouses provide long-term holding sites for product. They are distinct from distribution centers and production warehouses in how they handle the flow of products in a supply chain. Distribution centers focus on the rapid inflow and outflow shipment of orders, usually from retailers to customers or retail stores. Production warehouses are involved in both the storage and manufacture of various products. Distribution centers present the greatest potential for loss of product and mistaken shipments given their fast-paced operation and high volume of orders processed, relative to storage and production warehouses.

Inventory loss is a universal issue in the supply chain industry, and may manifest in several forms. According to the 2016 National Security Survey, inventory shrinkage cost the U. S. retail economy $45.2 billion in 2015 and nearly half of businesses surveyed reported an increase in shrinkage over previous years (Hollinger 7). Shoplifting, theft, fraud, and administrative errors contribute to 92.8% of loss, but the remainder is due to miscellaneous error, including misplacement of product stock.

Currently, the industry lacks good tools to solve the issue. Professional supply chain websites recommend tips for minimizing loss, such as proper labeling, keeping good records of locations of different items, and training employees preventatively. Workers could also be hired or trained for the purpose of searching for lost inventory, but the running cost of salary usually outweighs the benefits.

Applying emergent behavior to recovery of lost inventory may improve warehouse efficiency by reducing loss of property. Our system would be designed as a distributed search algorithm to locate objects by dispatching independent, cheap robots. Centralized communication would be expensive and require time for installation and connection to each robot, so the robots would instead operate independently with no overarching network. Taking inspiration from ants, the robots would follow a simple decision tree to determine actions to take. They would follow an individual search path with basic obstacle avoidance until a lost object is found or another robot is met. If an object is found, the robot brings it back to the “nest”, a lost-and-found area. When it encounters another robot within the range of bluetooth connection, they exchange their travel history, allowing them to avoid where the other has already searched.

Information found from the bench and field studies corroborated, both affirming that a significant source of lost inventory is due to simple misplacement at warehouses and distribution centers. Currently, companies that receive large shipments and lose some items either “eat” the losses, or they receive some insurance credit for lost items, lending credence to our assumption that most of the time, the cost of solving the issue is prohibitive. Unfortunately, insurance will only cover losses that are over a certain threshold, which may be very high. Consequently, credit is often not given for lost items, so it is not a reliable solution and another must be found.

2.1.3. User Persona

Jesús Corona

"Jesús is a great manager and friend" - Waris, Jesús' Coworker

AMBITIOUS **PROACTIVE** **DETAILED**

"Order, activate, exceed expectations. Fess one up an' you're toast. As the manager of *Emergent Warehouse Solutions*, I strive for accuracy and excellence in my work."

BIO

Jesús works as a warehouse manager. His interest in supply chain operations and his passion for finding intuitive solutions to tedious problems propelled his career to stardom. A self-proclaimed "workaholic," Jesús prides himself on his track record of perfect job attendance for 9 years straight. He enjoys learning about new systems to implement in his warehouse. He always looks for ways to ensure his, and everyone else's, business is running smoothly.

DEMOGRAPHIC

Jesús, a 34 year old warehouse manager from Chicago, Illinois, makes a salary of $60,000.

PERSONALITY TRAITS

Jesús has a high attention to detail. He is always monitoring the warehouse floor, so he notices any disarray and wishes for a more organized warehouse. His coworkers would describe him as ambitious, proactive, and detailed.

MOTIVATIONS

Jesús has an intense loyalty to the company he works at. He started as a forklift operator when he turned 18 and has moved up the ladder since. His devotion to the company defines his career, and he is always looking for opportunities to prove himself as a valuable asset to the company.

GOALS AND FRUSTRATIONS

Jesús understands the issues each one of his operators faces because he's been in their shoes and because he makes a point of getting to know everyone personally. Their frustrations, such as items not being where they should be, are his.

- ❏ Jesús wants technology to augment his warehouse workers, not replace them
- ❏ Jesús wants to implement robots to maintain the bottom line of organization so his employees can focus on more complex tasks
- ❏ Jesús deals with loss of inventory regularly - his warehouse is large and deals with thousands of different items in a day

Image Source: https://imgaws.ehowcdn.com/340x221p/photos.demandstudios.com/getty/article/103/251/86504306.jpg

2.2. Problem Definition

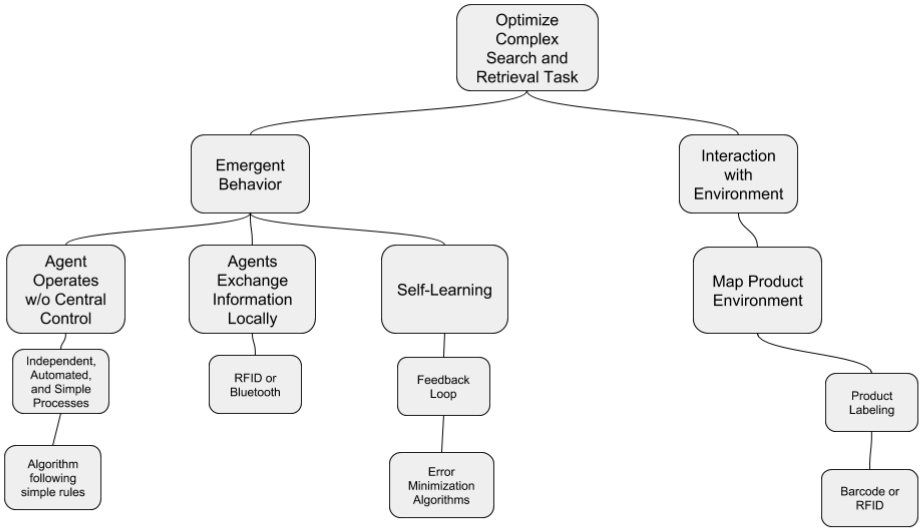


Figure 3. Objective Tree

Through our research on ants, we found they have an exceptional ability to complete complicated tasks without central oversight. This creates an adaptability and quickness that we wanted to emulate when humans attempt to solve complex tasks, specifically when searching for items the way ants search for food. After completing a market study, we found there are billions of dollars lost in lost items in warehouses and distribution centers, so our objective is to solve the task of searching for lost items in these spaces. Level one of our objective tree includes emergent behavior and interaction between the environment and our machine. Our solution prioritizes quickness. By utilizing multiple agents to distribute the task of searching, we can cover more ground in less time. Also, the agents must work together to share what they learn and dynamically optimize their paths to finding lost items. Emergent behavior in ants exhibits these behaviors to search for food, and our goal is similar to that of ants in finding the location of something unknown.

To augment the dynamic search our agents perform, it is also important they interact with their environment. Our team believes that if our product can mark locations already searched and communicate that to nearby agents, they will waste less time. We also believe it would be extremely useful for the agents to map the warehouse as they carry on their search. Indicated by our market study, many misplaced items are caused by distribution centers not sending entire orders. A store may order 100 items but only receive 98 and still be charged for all 100. To combat this, as the agents map the products at a distribution center, they can find if products don’t belong in certain places, returning misplaced items to where they belong.

2.3. Problem Specifications

Table 3. Design Criteria Metrics for Needs and Metric Importance Ranking

Design Criteria		
Need	**Metric**	**Importance (/10)**
Agent-Agent Local Communication	Connectivity	**10**
Agent Task Rate; Efficient Task Accomplishment Behavior	Task Accomplishment Time	**7**
Emergent Behavior Task Rate	Task Efficiency = Time Taken / Average Time	**8**
Self-Learning; Mistakes Avoided By Trying Alternative Process	Error = Mistakes Made / Attempts Taken	**9**
Reliable Product Identification by Code or Chip	Content Accuracy = True Content / Total Content	**8**
Agent Quickly Determines Product Location	Time to Find Product	**10**
Emergent Functionality; Compare to Functionality with Design Oversight	Task Time Emergent / Task time Oversight	**8**

3. Explore: Conceptual Design

3.1 Ideation

3.1.1 Ideation Chart

Table 4. Chosen Critical Functions and Implementations for the Design Statement

Function	Implementation 1	Implementation 2	Implementation 3	Implementation 4	Implementation 5
Self-Learning	Agent start-location optimization for searching warehouse	Memory of objects encountered - both lost items and obstacles	Obstacle avoidance for each obstacle encountered	Changes walk algorithm for package encounter	Agents exchange info on paths taken and obstacles found
No Central Oversight	Agents all operate on simple, predefined rules	Agents which underperform on solve time restart along with memory of obstacles	Agents always deliver lost package to fixed origin	Agents continuously loop through search and return functions for lost items	Agent search and return is modified by local interactions

3.1.2 Selection of Top 3 Implementations

Table 5. Post-It Note Method for Determining Function Implementations

Function	Implementation 1	Implementation 2	Implementation 3	Implementation 4	Implementation 5
Self-Learning	**Agent start-location optimization for searching warehouse**	**Memory of objects encountered - both lost items and obstacles**	Obstacle avoidance for each obstacle encountered	Changes walk algorithm for package encounter	**Agents exchange info on paths taken and obstacles found**
Luis					
Tor					
Faris					
Score:	**0**	**2**	-1	-1	**3**
No Central Oversight	**Agents all operate on simple, predefined rules**	Agents which underperform on solve time restart along with memory of obstacles	Agents always deliver lost package to fixed origin	**Agents continuously loop through search and return functions for lost items**	**Agent search and return is modified by local interactions**
Luis					
Tor					
Faris					
Score:	**2**	-1	-1	**1**	**2**

3.1.3 SWOT Method

Table 6. Self Learning

Implementation	Strengths	Weaknesses	Opportunity	Threats
(1) Start location Optimization	1. Easy to vary start location, so many variations can be tested	1. Difficult to implement. 2. Lack of baseline to compare our algorithm to	1. We find a secret metric that affects solve time.	1. Difficult to generalize for many warehouses 2. Start location may have no effect on solve time, so it can be ultimately waste of time.
(2) Object Memory	1. Simple to program for each agent; context independent 2. Easily scalable	1. Difficult to recognize and differentiate objects	1. Agents can navigate through obstacles	1. Memory storage may make each agent expensive
(3) Information exchange between agents	1. Enables agents to learn from unaccessed areas 2. Enables agents to quickly navigate new spaces	1. Unnecessary for navigating narrow/ordered spaces 2. Difficult to ensure agents always capable of transmitting signals	1. Agents converge on a solution collectively 2. lowers solve time	1. Exchanges may be difficult across long distance or through obstacle

Table 7. SWOT of No Central Oversight

Implementation	Strengths	Weaknesses	Opportunity	Threats
(1) Simple, predefined rules	1. Rules can be changed easily, so iterations are easy to make 2. Additional function can be added easily by adding new rules to program	1. Complex behavior may not be achieved through simple rules	1. Individual agents are easy to create and program	1. Agents may not differentiate enough if they follow the same rules/ they will all end up doing the same thing, eliminating need for multiple agents
(2) Agents continuously loop through search and return functions for lost items	1. No time is wasted on agents having to process external environment to determine next action	1. Same search and return algorithms are utilized, even if they are suboptimal	1. Allows for simplicity and identicality in the design of agents so individual agents cost less	1. Obstacles complicate search or return functions such that a loop causes agents to get stuck in the same location/region of the warehouse space
(3) Agent search and return is modified by local interactions	1. Improves efficiency of search and return functions as all agents search warehouse space	1. Local interactions are dependent on agents successfully interacting with objects - not assured in some warehouse spaces where spatial geometry is complex	1. Searches are dynamically improved as new information is encountered along the way by agents	1. Sub-optimal search/return patterns may be passed on by agents, actually increasing solve time

3.1.4 Morphological Analysis of Implementations

Table 8. Morphological Analysis Matrix

Function	Implementation 1	Implementation 2	Implementation 3
(A) No Central Oversight	Simple, predefined rules will be coded for each agent. These rules will describe every possible behavior.	Agents continuously loop through search and return functions for lost items.	Agent search and return is modified by local interactions. Each run the agent makes will give the agent more information to allow for a improved search and return.
(B) Self-Learning/Local Interactions	Using logistic regression or a similar statistical model, the start location of the agents will be optimized for least solve time.	Each agent will have a memory of the map that will update with every object encounter to allow for more efficient movement through obstacles.	Each agent will exchange information with others to share their knowledge of the map to search more efficiently.

Concept 1: One concept to consider includes the logistic regression model and a continuous loop of search and return functions for lost items. By using the same search and return functions, we can focus on optimizing the starting location of the agents to quickly explore the entire map and find lost items. A simulation can be run to figure out the optimal starting point of the agents for new warehouses before final deployment of the robots.

Concept 2: Another concept combines implementation 3A and B. Agents will exchange information on the location of obstacles around the map. This information will be used to dynamically search for lost items. Shorter paths can be mapped for agents by improved knowledge of where obstacles lie. This will make their return to the starting point quicker.

Concept 3: Our final concept combines nearly every implementation of table 6. Each agent will be given a set of simple, predefined functions that define its movements and interactions with other agents. Agents will also be equipped with a memory that allows them to map out the warehouse using their own findings and information gathered from other agents. They will do this using RFID technology to send information to all agents in the warehouse. Agents will receive this information to update their internal maps and change their paths accordingly. This concept combines implementation 1A, 2B, and 3A and B.

3.1.5 Sketches of Concepts 1-3

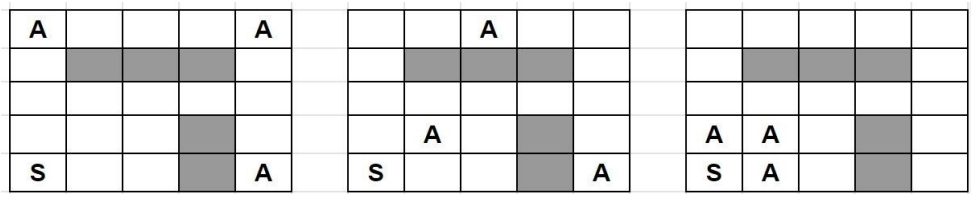


Figure 4. Concept 1 Sketch

Figure 4 shows how starting location will be changed for a distinct map. Starting positions will be entered into a logistic regression model as independent variables to find a relationship between starting position and solve time and the optimal location can be found.

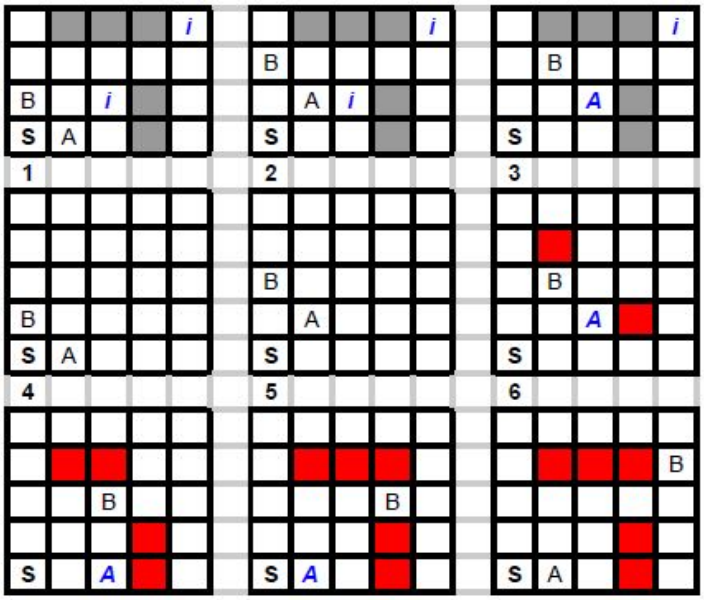


Figure 5. Concept 2 Sketch

Figure 5 illustrates concept 2. The top row of matrices shows the map with A and B being two agents, S being the return point, i being items, grey boxes indicating obstacles, and red boxes indicating obstacles found by agents. Six steps of the agents movement is shown in figure 5, showing how they update the map as they find obstacles.

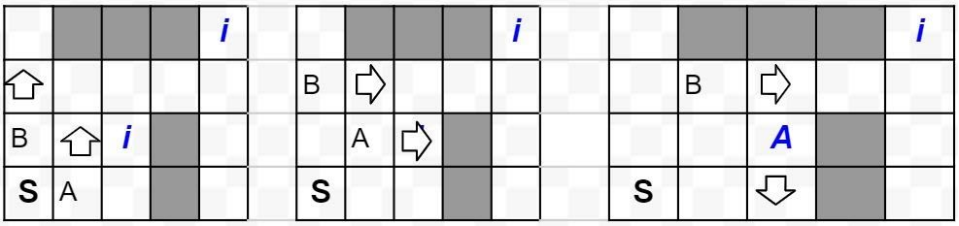


Figure 6. Concept 3 Sketch

Figure 6 attempts to show how agents will follow simple rules to move around the map. Each agent will be programmed to walk to a random, adjacent tile with weights on each tile. This will eliminate the risk that agents retrace their steps because unexplored tiles will be weighted more favorably. Concept 3 shares some function of concept 2 with regards to object mapping and information exchange.

3.2. Concept Evaluation

Metric Abbreviations Key

Connectivity	Task Accomplishment Time	Task Efficiency = Time Taken / Average Time	Error = Mistakes Made / Attempts Taken	Content Accuracy = True Content / Total Content	Time to Find Product	Task Time Emergent / Task time Oversight
Conn.	TAT	TET	Mistake Ratio	Accuracy	TFP	TTE

Table 9. Design Specification Matrix Including Rank (Weighted Importance) of Concepts

Design Criteria	Conn.		TAT		TET		Mistake Ratio		Accuracy		TFP		TTE		Sum	Rank
Concepts	0.245		0.082		0.122		0.061		0.204		0.041		0.245		1	-
(1) Lost Package Search Swarm - *Formica Emergens*	10	*2.5*	7	*0.6*	8	*1*	8	*0.5*	3	*0.6*	8	*0.3*	10	*2.5*	*7.878*	**1**
(2) Pallet Builder Swarm - *Aedificium Edificium Formica*	5	*1.2*	5	*0.4*	7	*0.9*	9	*0.5*	10	*2*	8	*0.3*	6	*1.5*	*6.876*	2
(3) Truck Unloading / Packing Ants - *Salsissimus Vir Vivens Formica*	3	*0.7*	8	*0.7*	5	*0.6*	10	*0.6*	10	*2*	10	*0.4*	3	*0.7*	*5.796*	4
(4) Ant Swarm Full Warehouse Processing System	10	*2.5*	10	*0.8*	1	*0.1*	5	*0.3*	1	*0.2*	1	*0*	8	*2*	*5.902*	3

Table 9 indicates the ranked concepts our team considered. The listed concepts were rated against the listed metrics by the following convention: Concepts likely generating better results for a metric are rated higher on a 1-10 scale. Concepts more heavily utilizing aspects of the listed metric are also rated higher on the same scale. For instance, concept (1) requires total connectivity between local agents in the swarm and therefore takes a rating of 10. Concept (4) is the least efficient of all concepts in terms of total time taken to complete its task (full automation of warehouse procedures), so it takes a rating of 1.

Table 9 also shows the final concept we intend to pursue. The ‘Lost Package Search Swarm’ - or *Formica Emergens* - design is built to locate lost packages in a warehouse.

3.3. Detailed Final Concept

The following image demonstrates a full-scale model of our simulation, indicating agents (A#), lost items (i), a fixed starting points (S), and agent behaviors. The return function path of the agents is shown as a sequence of arrows; this represents what the agent thinks is the shortest-line distance between their location and the fixed starting point. Notice that the agent plots the shortest-line distance to avoid only known obstacles. Other agents in the simulation increase the efficiency of the return function by communicating the location of obstacles. Agent A4 is shown to plot a return function through two unknown obstacle walls - causing it to waste time getting back to the fixed starting point.

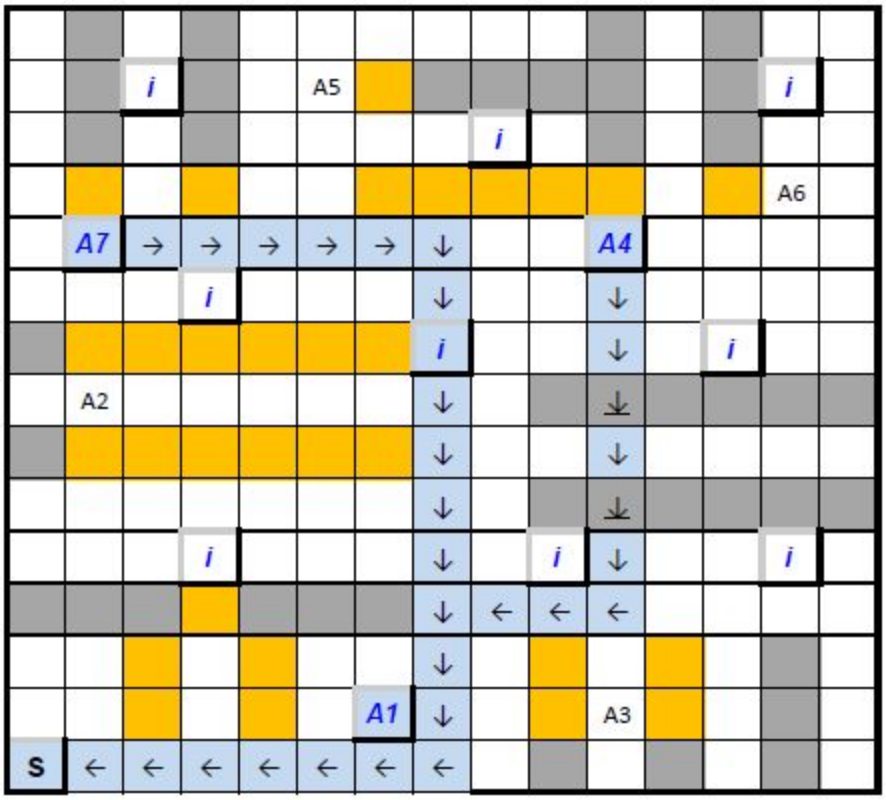


Figure 7. Map Indicating Agent Behaviors and Interactions within our Simulation

The major constraint involves scaling the simulation to incorporate more agents. We intend to fix the map size, obstacle position, and number of lost items so we can focus on varying the number of agents in our simulation. We expect that a certain number of agents will allow emergent behavior to effect agent search and return functions. The effectiveness of emergent behavior will be measured as a factor of solve time: the time it takes for all lost items to be returned to the fixed starting point.

3.4. Concept Reflection

Table 10. Four box reflection chart comparing ant bio-inspiration with warehouse agents

Problem Target		**Biological Source**
Operational Environment		*Operational Environment*
Warehouse/Distribution center	Similar	Ant colony and foraging paths
Highly unpredictable areas (warehouses)	Same	Highly unpredictable areas (forest)
Randomly spread targets (lost items)	Same	Randomly spread targets (food)
Climate controlled environment	Different	Natural environment
Functions		*Functions*
Find lost items in a warehouse	Similar	Forage for nutrients in a forest
Operate autonomously	Same	Operate autonomously
Navigate ordered lanes/obstacles	Similar	Navigate complex terrain
Share warehouse information (obstacle locations) with agents	Similar	Reinforce pheromone trails to nutrient-rich areas
Specifications		*Specifications*
Adaptable to work in all warehouses	Same	Adaptable to forage different paths
Inexpensive	N/A	N/A
Portable	Same	Swarm can translate nest site
Modular - can add additional agents	Same	New ants can be born into swarm
Learns about warehouse space through successive search-return iterations	Similar	Label biotic environment to remember certain trails/foraging targets
Criteria		*Criteria*
Translate in two degrees of freedom	Similar	Translate in 3D and rotate in 3D
Identify lost objects by tags/IDs	Similar	Identify nutrients by CHCs
Communicate to other agents via RFID	Similar	Communicate with other ants by CHCs
Grasp lost objects for retrieval/return	Similar	Lift up to six times bodyweight to carry nutrients to nest

4. Materialize: Design Embodiment

4.1. CFP Proposal

What is the minimum number of agents required to observe emergent behavior in our system? Our simulation attempts to answer this question by providing the environment and rules for emergent behavior to arise. The simulation will include implementation of the walk function, obstacle recognition/avoidance, item detection, and agent local interaction as communication of map data (known obstacles; empty spaces; explored space; found lost items). The simulation will not include the return function, since it seeks primarily to demonstrate emergent behavior through a prototype simulation.

We will use solve time to evaluate our critical function prototype. Adding more agents into the simulation should decrease the time it takes for all lost items to be retrieved (solve time). If a sudden drop in solve time is noticed after adding a certain number of agents to the simulation, it is possible that emergent behavior is influencing solve time. Two metrics will be used to assess the onset of emergent behavior: the emergence metric and the series ratio.

Table 11. Metrics of success for simulation

Agents	Solve Time	Emergence Metric	Series Ratio
1	1000		
2	500	500	
3	333	167	0.33
4	250	83	0.50
5	200	50	0.60
6	166	34	0.68
7	100	66	1.94
8	50	50	0.76
9	20	30	0.60
10	10	10	0.33

Table 11 shows the scenario where the number of agents is increased and solve time decreases non-linearly. The emergence metric quantifies the change in solve time between a set number of agents. For instance, solve time is 1000 for 1 agent and 500 for 2 agents, returning an emergence metric of 500 (1000 - 500). The series ratio allows us to characterize a sudden change in solve time and is the ratio between two consecutive emergence metrics. For instance, emergence metric is 500 for 2 agents and 167 for 3 agents, returning a series ratio of 0.33 (167/500). Series ratios between 0 and 1 indicate a positive, but weak effect on solve time

for additional agents. Series ratios about equal to: ((Solve Time N / # of agents) / Solve Time 1) indicate a simple linear effect on solve time for additional agents: e.g. adding 1 agent to 1 agent halves solve time from 1000 to 500. Series ratios above 1 indicate the addition of an agent strongly decreases solve time: e.g. 6 to 7 agents reduces solve time by a factor of 1.66 rather than the expected 0.86 (166 to 142 if solve time decrease is equal over all agents).

The purpose of our critical function prototype will be to establish a baseline for autonomy and emergent behavior for the agents. Specifically, we want to know what is the minimum number of agents required for the system to display emergent behavior. If we can build a system of agents that minimizes solve time by displaying emergent behavior, this indicates success.

4.2. CFP Experimental Evaluation Plan

The simulation we are constructing will return data on the parameters necessary to observe emergent behavior in our model. We define emergent behavior as the state of a system where interactions between local agents provides a benefit to their search function and decreases solve time as a result. Table 12 details our method for obtaining experimentally relevant data:

Table 12. Experimental Setup with Dependent and Independent Variables

	Map Size	**# of Agents**
Increase	35x35, 47x47, 69x69	3, 6, 9, 12, 15
Emergent Behavior Metrics: EM, SR	EM1,SR1 EM2,SR2 … EMn,SRn	EM1,SR1 EM2,SR2 … EMn,SRn

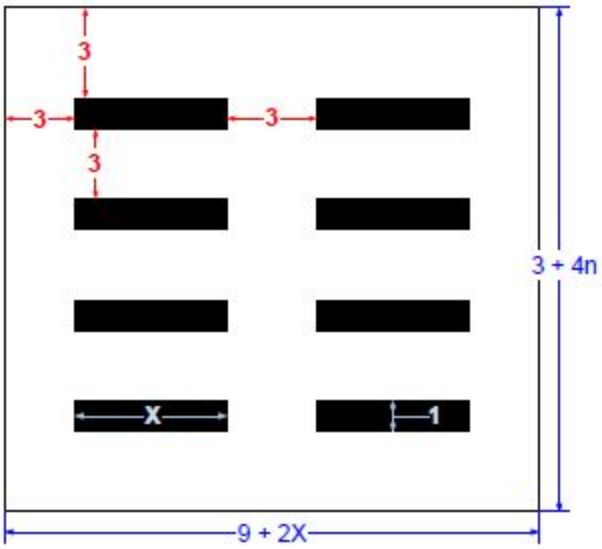


Figure 8. World Size Parameters for 'n' obstacles of 'X' length with 3 unit buffer spaces

We will run our simulation in iterations, varying the number of agents and the map size while keeping the layout of obstacles and items constant. For each map size, we will increase the map size by 12 tiles to preserve even lane sizes between obstacles. Agent count will increase by one per simulation and return one solve time for the entire swarm for each map size. In total there will be 15 experiments to allow for 1-5 agents to attempt each map.

EM refers to the emergence metric. The emergence metric is the first derivative of solve time obtained from the forward difference method for discrete intervals. EMn refer to the n'th emergence metric. The emergence metric tells us how much solve time varies between two consecutive sets of parameters. If a sudden jump in emergence metric is shown, it may indicate the onset of emergent behavior in our simulation.

SR refers to the series ratio. The series ratio is the ratio of two consecutive emergence metrics and returns a value between 0 and the maximum solve time for 1 agent. Series ratio is a normalized metric for directly determining the onset of emergence. Table 13 defines characteristics of the series ratio for all possible intervals in our experiment:

Table 13. Series Ratio Range Interpretations

0 < Series Ratios < 1	Series Ratio = 1	Series Ratio > 1
No Emergent Behavior The specific set of map size and agent number provides a benefit less than directly proportional to the set of parameters being tested.	*No Emergent Behavior* The specific set of map size and agent number provides a benefit directly proportional to the set of parameters being tested.	*Emergent Behavior* The specific set of map size and agent number provides a benefit more than directly proportional to the set of parameters being tested.

For series ratios above 1, we may conclude that emergent behavior is present in the system because solve time decreases faster than direct proportionality. Adding more agents can only decrease solve time, since each provides no negative effect on others' search function. Increasing map size can only increase solve time, because it adds more unexplored areas for agents to sort through.

4.3. CFP Demonstration Results and Discussion

The data returned from each iteration of our simulation includes: the number of agents, the number of interactions between agents, the map size, and the total solve steps to find the lost item. The correlation between the number of solve steps and the number of interactions for each set of agents increases with increasing agent count.

Table 14. Correlation Between Solve Steps and Number of Agent Interactions

Number Agents	**3 Agents**	**6 Agents**	**9 Agents**
Correlation Coefficient:	0.5600	0.7283	0.8329

Plotting the data for 3, 6, and 9 agents illustrates increasing correlation between these two parameters:

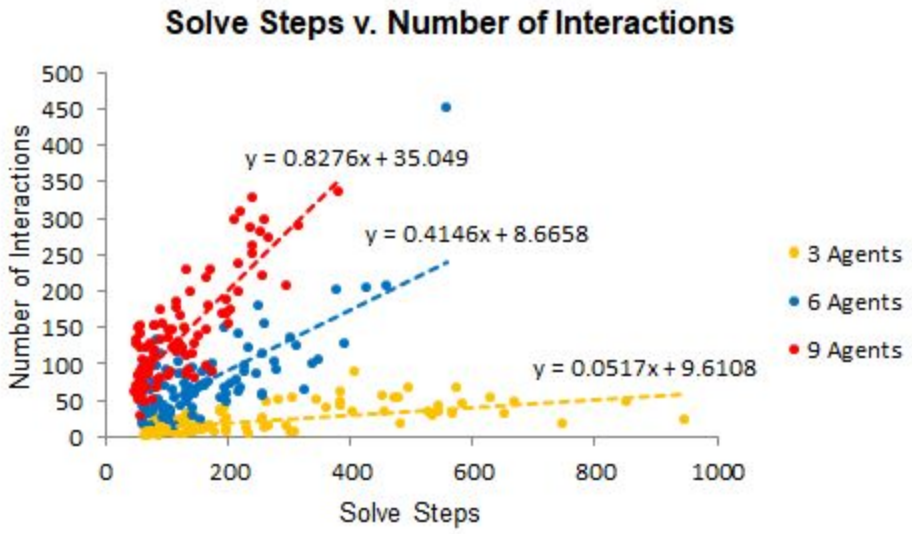


Figure 9. Scatter plot of agent number of interactions to solve steps with trendlines

The number of solve steps to agent count exhibits an exponential trend, with average solve time decreasing at a slowing rate for additional agents. The standard deviation of solve time also decreases for additional agents.

Table 15. Standard Deviation of Solve Time and Number of Agents

Number Agents	**3 Agents**	**6 Agents**	**9 Agents**
Standard Deviation:	198.99	98.09	72.20

This trends indicates that increasing the number of agents has the effect of converging on a solve time. The position of the mean solve time for each set of agents and the distribution of individual solve times for each trial illustrate the convergence phenomenon:

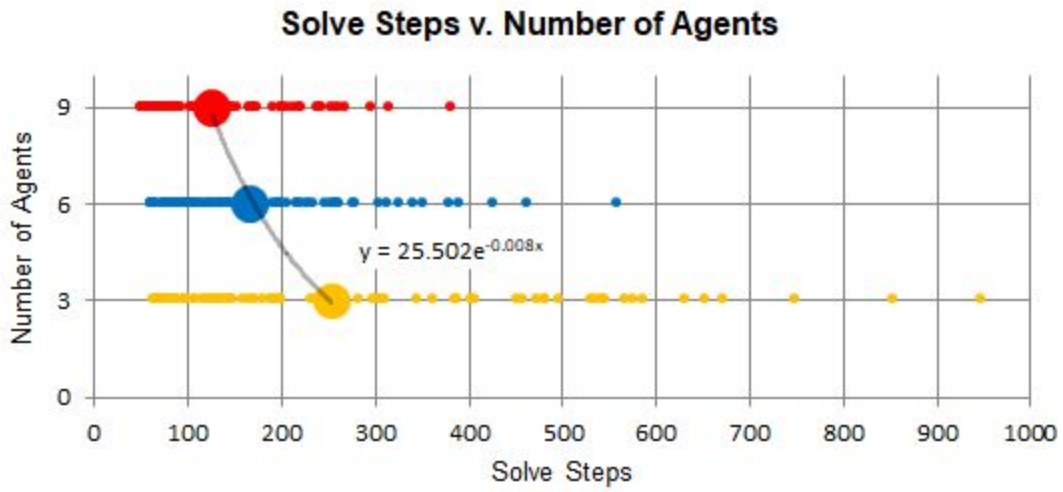


Figure 10. Total solve steps per trial per set of agents with trendline

Average solve time is expected to decrease with greater numbers of agents searching for the lost item and exchanging map information. Adding agents to the map increases the rate of unsearched blocks being searched. The increase in correlation between solve steps and number of interactions indicates some emergent behavior in the form of increased influence between agents; agents more likely influence each others' search behavior because they are more likely to interact and share map information.

Emergent behavior was demonstrated using the series ratio as a metric. The results of 50 trials, with 1 through 10 agents, returned 8 series ratios. The ratio from 9 to 10 agents indicates a significant drop in solve time:

Table 16. Metrics for 1-10 agents

Number of Agents	Average Solve Time	Emergence Metric	Series Ratio
1	945.02		
2	510.36	434.66	
3	265.52	244.84	0.5633
4	219.28	46.24	0.1889
5	199.42	19.86	0.4295
6	254.90	-55.48	-2.7936
7	134.78	120.12	-2.1651
8	138.06	-3.28	-0.0273
9	130.30	7.76	-2.3659
10	98.80	31.50	4.0593

According to our definition of emergent behavior, we conclude that the experiment was successful in identifying its onset at 10 agents on a 35 x 35 cell map.

5. Conclusions and Discussion

For our final design, we emulated a swarm of robot agents that explored a warehouse floor and found misplaced items. The purpose of the simulation was to examine and measure the emergent behavior multiple communicating agents would exhibit when searching for an item. Our design captured the essence of our biological inspiration by allowing simple agents to communicate with each other, using the exchanged information to dynamically improve their search. By allowing agents to store maps of the warehouse and share their map with others, the agents exhibited more complex behavior than the sum of their individual parts, mimicking an ant colony that is smarter than the sum of all the ants.

Our simulation incorporated the core principles of ant swarm biotic/local interactions and emergent behavior. The agents in our simulation exhibited local interactions by the trading of map data when they intersected paths. The agents collectively exhibited emergent behavior by each agent individually operating from a predefined set of rules and the agents altogether conferring a benefit to the swarm when they interacted locally. The specific benefit was an exponential decrease in average total solve steps for increasing numbers of agents in the simulation. If future steps were taken, we would like to see an improved center of mass function that would direct agents to unexplored territory. Our current center of mass function still operates with some randomness, which sometimes leads agents to get stuck at corners or at the center of the map. Additionally, we could add a return function and more lost items so the agents may return lost items to a starting point and immediately return to their search for more items.

The most challenging aspect of the bio-inspired design process was extracting the core principles from the organism's functions. The structures that allowed the functions were easy to identify, but translating these into more abstract principles that could be applied to other problems was sometimes difficult. The most interesting part of the bio-inspired design process was examining the ants and their abilities in detail. We learned a lot of creative and fascinating ways ants devised solutions to their problems. We believe bio-inspiration would be most useful in problem-based design. When there is a problem, you can reframe into biological terms and search the biological world to identify different solutions that can help you refresh during a brainstorming session.

6. References

Full, R. J. (2002). Quantifying Dynamic Stability and Maneuverability in Legged Locomotion. *Integrative and Comparative Biology*, 42(1), 149-157. doi:10.1093/icb/42.1.149

Couzin, Iain & Franks, Nigel. (2003). Self-organized lane formation and optimized traffic flow in army ants. Proceedings. Biological sciences / The Royal Society. 270. 139-46. 10.1098/rspb.2002.2210.

Tennenbaum, Michael, et al. “Mechanics of Fire Ant Aggregations.” *Nature News*, Nature Publishing Group, 26 Oct. 2015, www.nature.com/articles/nmat4450.

Prabhakar, B., Dektar, K. N., & Gordon, D. M. (2012). The Regulation of Ant Colony Foraging Activity without Spatial Information. *PLoS Computational Biology,8*(8). doi:10.1371/journal.pcbi.1002670

Chomicki, Guillaume, and Susanne S. Renner. “The Interactions of Ants with Their Biotic Environment.” *Proceedings of the Royal Society B: Biological Sciences*, vol. 284, no. 1850, 2017, p. 20170013., doi:10.1098/rspb.2017.0013.

Holman, Luke, et al. “Genetic Constraints on Dishonesty and Caste Dimorphism in an Ant.” *The American Naturalist*, vol. 181, no. 2, 2013, pp. 161–170., doi:10.1086/668828.

Hollinger, Richard. “THE 2016 NATIONAL RETAIL SECURITY SURVEY.” NRF, 2016, cdn.nrf.com/sites/default/files/2018-10/NRF_2016_NRSS_restricted-rev.pdf.

7. Appendices

Appendix A.

```
import pygame, numpy as np, math, csv

def print_map(list_of_lists):
  for row in list_of_lists:
        print(''.join(((' ')*(3-len(str(entry)))+str(entry)) for entry in row))
  print('')

def draw_map(list_of_lists, top_left, tile_length):
  for j in range(len(list_of_lists)):
        for i in range(len(list_of_lists[j])):
    pygame.draw.rect(screen, COLOR_SCHEME[list_of_lists[j][i]], [top_left[0] + (i * tile_length),
top_left[1] + (j * tile_length), tile_length, tile_length])
  pointlist = [top_left]
  pointlist.append([top_left[0] + (len(list_of_lists) * tile_length), top_left[1]])
  pointlist.append([top_left[0] + (len(list_of_lists) * tile_length), top_left[1] + (len(list_of_lists) *
tile_length)])
  pointlist.append([top_left[0], top_left[1] + (len(list_of_lists) * tile_length)])
  pygame.draw.aalines(screen, WHITE, True, pointlist)

def draw_mini_map(list_of_lists, top_left, map_length):
  tile_length = map_length // len(list_of_lists)
  for j in range(len(list_of_lists)):
        for i in range(len(list_of_lists[j])):
    pygame.draw.rect(screen, COLOR_SCHEME[list_of_lists[j][i]], [top_left[0] + (map_length * (i /
len(list_of_lists[j]))) // 1, top_left[1] + (map_length * (j / len(list_of_lists[j]))) // 1, tile_length,
tile_length])
  pointlist = [top_left]
  pointlist.append([top_left[0] + map_length, top_left[1]])
  pointlist.append([top_left[0] + map_length, top_left[1] + map_length])
  pointlist.append([top_left[0], top_left[1] + map_length])
  pygame.draw.aalines(screen, WHITE, True, pointlist)

def center_of_mass(space):
  numer_x = 0
  denom_x = 0
```

```
  numer_y = 0
  denom_y = 0

  for i in range(len(space)):
          for j in range(len(space[i])):
          if space[i][j] == -4:
          numer_x += j
       numer_y += i
          denom_x += 1
          denom_y += 1

  return [numer_x/denom_x, numer_y/denom_y]

def center_of_mass_undiscovered(space):
  numer_x = 0
  denom_x = 0
  numer_y = 0
  denom_y = 0

  for i in range(len(space)):
          for j in range(len(space[i])):
          if space[i][j] == -1:
          numer_x += j
          numer_y += i
          denom_x += 1
          denom_y += 1

  return [numer_x/denom_x, numer_y/denom_y]

def wall_weighted_COM(space, sides_discovered):
  weight = 100
  adjusted_space = []
  for i in range(weight):
    adjusted_space.append([sides_discovered[0]] * ((2 * weight) + len(space)))

  for i in range(len(space)):
          row = [sides_discovered[1]] * weight + space[i] + [sides_discovered[2]] * weight
          adjusted_space.append(row)
```

```
  for i in range(weight):
    adjusted_space.append([sides_discovered[3]] * ((2 * weight) + len(space)))

  [x_raw, y_raw] = center_of_mass_undiscovered(adjusted_space)
  #print_map(adjusted_space)
  #print([x_raw - weight, y_raw - weight])
  return [x_raw - weight, y_raw - weight]

def furthest_away(list_of_directions, curr_pos, repel_pos):
  furthest = 0
  furthest_distance = 0

  for i in list_of_directions:
          direction_vector = [repel_pos[0] - (curr_pos[0] + directions[i][0]), repel_pos[1] - (curr_pos[1]
+ directions[i][1])]
          distance = (direction_vector[0] ** 2) + (direction_vector[1] ** 2)
          if distance > furthest_distance:
          furthest = i

  return furthest

def closest_to(list_of_directions, curr_pos, attract_pos):
  closest = 0
  closest_distance = 10000

  for i in list_of_directions:
          direction_vector = [attract_pos[0] - (curr_pos[0] + directions[i][0]), attract_pos[1] -
(curr_pos[1] + directions[i][1])]
          distance = (direction_vector[0] ** 2) + (direction_vector[1] ** 2)
          if distance < closest_distance:
          closest = i
          closest_distance = distance
  return closest

class Agent:
  def __init__(self, start_pos, name_ID, map_size):
          #Initialize column, row, and unique name ID (positive int)
```

```
        self.x = start_pos[0]
        self.y = start_pos[1]
        self.id = name_ID
        self.prev_index = 0
        self.walls_discovered = [-1, -1, -1, -1]

        #ext is the border indices of the map
        self.ext = [map_size[0] - 1, map_size[1] - 1]

        #Initialize internal limited map to list with mapsize[1] (y dimension) number of of lists, each
of length mapsize[0] (x dimension)
        self.map = []
        for i in range(map_size[1]):
        self.map.append([-1] * map_size[0])

        #Label the starting location in the map the agent ID
        self.map[self.y][self.x] = self.id

        #Initialize agent to Searching state
        self.state = 0

        #Initialize trail of positions for retracing steps
        self.trail = []
   self.moves = []

 def grab_map(self, other_agent):
        if other_agent != self.id:
        for bot in robots:
        if bot.id == other_agent:
        for row_i in range(len(bot.map)):
        for entry_i in range(len(bot.map[row_i])):
          if self.map[row_i][entry_i] == -1:
        self.map[row_i][entry_i] = bot.map[row_i][entry_i]

 def move(self, global_map):
        next_index = 0
        next_pos = []
        global interactions
```

```
        if self.state == 0:
        local_options = list(range(8))

        if self.x == 0:
    local_options.remove(5)
    local_options.remove(6)
    local_options.remove(7)
        self.walls_discovered[1] = -4

        if self.x == self.ext[0]:
    local_options.remove(1)
        local_options.remove(2)
    local_options.remove(3)
        self.walls_discovered[2] = -4

        if self.y == 0:
        try:
      local_options.remove(3)
        except ValueError:
        pass
    local_options.remove(4)
        try:
        local_options.remove(5)
        except ValueError:
        pass
        self.walls_discovered[0] = -4

        if self.y == self.ext[1]:
        try:
      local_options.remove(7)
        except ValueError:
        pass
        local_options.remove(0)
        try:
      local_options.remove(1)
        except ValueError:
        pass
        self.walls_discovered[3] = -4
```

```
        local_second_choice = list(local_options) #Just in case there are no unexplored tiles
        local_wall_follow = list(local_options) #Just in case there are no unexplored tiles
        local_options_0 = list(local_options) #Create a copy of local_options to iterate through.
Previously, this led to some errors where the for loop would skip the next item if an item was
removed from it.

        for i in local_options_0:
        target_loc = global_map[self.y + directions[i][1]][self.x + directions[i][0]]
        map_loc = self.map[self.y + directions[i][1]][self.x + directions[i][0]]

        if i not in [self.prev_index - 1, self.prev_index, self.prev_index + 1]: #If diverges from path
      local_options.remove(i)
        if target_loc > 0:
        self.map[self.y + directions[i][1]][self.x + directions[i][0]] = -4
        self.grab_map(target_loc)
        interactions += 1
        local_second_choice.remove(i)
        local_wall_follow.remove(i)
        elif target_loc == -3:
        self.map[self.y + directions[i][1]][self.x + directions[i][0]] = target_loc
        local_second_choice.remove(i)
        local_wall_follow.remove(i)
        continue

        if map_loc == -4: #If already been there
      local_options.remove(i)
        if target_loc > 0:
        self.map[self.y + directions[i][1]][self.x + directions[i][0]] = -4
        self.grab_map(target_loc)
        local_second_choice.remove(i)
        local_wall_follow.remove(i)
        elif self.x == 0:
        if i != 4:
          local_wall_follow.remove(i)
            elif self.x == self.ext[0]:
        if i != 0:
          local_wall_follow.remove(i)
```

```
elif self.y == 0:
if i != 2:
 local_wall_follow.remove(i)
elif self.y == self.ext[1]:
if i != 6:
        local_wall_follow.remove(i)
elif global_map[self.y + directions[(i+1)%8][1]][self.x + directions[(i+1)%8][0]] != -3:
local_wall_follow.remove(i)

else: #If haven't been there
if target_loc == -3:
self.map[self.y + directions[i][1]][self.x + directions[i][0]] = target_loc
local_options.remove(i)
local_second_choice.remove(i)
local_wall_follow.remove(i)

elif target_loc == -2:
self.map[self.y + directions[i][1]][self.x + directions[i][0]] = target_loc
next_pos = [self.x + directions[i][0], self.y + directions[i][1]]
self.state = 1
break

elif target_loc > 0:
      self.map[self.y + directions[i][1]][self.x + directions[i][0]] = -4
self.grab_map(target_loc)
interactions += 1
local_options.remove(i)
local_second_choice.remove(i)
local_wall_follow.remove(i)

if self.state == 0:
if len(local_options) != 0:
next_index = np.random.choice(local_options)
next_pos = [self.x + directions[next_index][0], self.y + directions[next_index][1]]

elif len(local_second_choice) != 0:
#print("retracing steps: ", self.id)
#next_index = furthest_away(local_second_choice, [self.x, self.y],
```

```
wall_weighted_COM(self.map, self.walls_discovered))
            #next_index = closest_to(local_second_choice, [self.x, self.y],
wall_weighted_COM(self.map, self.walls_discovered))
        local_second_choice.append(closest_to(local_second_choice, [self.x, self.y],
center_of_mass_undiscovered(self.map)))

            next_index = np.random.choice(local_second_choice)

            next_pos = [self.x + directions[next_index][0], self.y + directions[next_index][1]]

            else:
            #print("stuck: ", self.id)
            next_pos = [self.x, self.y]

            self.moves.append(next_index)

            elif self.state == 1:
            #TODO: Add conditional for if self.trail[-1] has another robot in the way, don't move there.
            next_pos = self.trail.pop()

            #Vacate durrent position to 'explored'
            self.map[self.y][self.x] = -4

            #Set previous index to be current index
            self.prev_index = next_index

            #Add current position to the robot's trail
            self.trail.append([self.x, self.y])

            #Set cursor to next position
            self.x, self.y = next_pos

            #Write in id at new position
            self.map[self.y][self.x] = self.id

BLACK  = (  0,   0,   0)
RED    = (255,   0,   0)
GREEN  = (  0, 255,   0)
```

```
BLUE   = (  0,   0, 255)
ORANGE = (255, 255,   0)
CYAN   = (  0, 255, 255)
PURPLE = (255,   0, 255)
WHITE  = (255, 255, 255)
GREY_1 = ( 75,  75,  75)
GREY_2 = (150, 150, 150)

COLOR_SCHEME = {
  -4: GREY_2,
  -3: GREY_1,
  -2: ORANGE,
  -1: BLACK,
   0: WHITE,
   1: RED,
   2: GREEN,
   3: BLUE,
   4: CYAN
}

def closest_minimaps(number_of_agents):
  return ((math.sqrt(number_of_agents - 1) // 1) + 1) ** 2

directions = [[0, 1], [1, 1], [1, 0], [1, -1], [0, -1], [-1, -1], [-1, 0], [-1, 1]]

#FPS = 60

trials = []
#start_n_agents = int(input("Number of agents to start with: "))
#incr_n_agents = int(input("Number of agents to increment by: "))
#finish_n_agents = int(input("Number of agents to finish with: "))
#start_n_tile_side = int(input("Number of tiles/side to start with: "))
#incr_n_tile_side = int(input("Number of tiles/side to increment by: "))
#finish_n_tile_side = int(input("Number of tiles/side to finish with: "))
#n_trials = int(input("Number of trials for each: "))
#file_name = input("File name: ")

start_n_agents = 1
```

```
incr_n_agents = 1
finish_n_agents = 10

start_n_tile_side = 35
incr_n_tile_side = 12
finish_n_tile_side = 35

n_trials = 100
file_name = 'results_5_5_100trials'

for var_tiles in range(start_n_tile_side, finish_n_tile_side + incr_n_tile_side, incr_n_tile_side):
  for var_agents in range(start_n_agents, finish_n_agents + incr_n_agents, incr_n_agents):
          for trial in range(n_trials):
     trials.append([var_agents, var_tiles])

app_exit = False

MAP_PIX = 500
SCREEN_SIZE = (MAP_PIX * 2, MAP_PIX)

pygame.init()

screen = pygame.display.set_mode(SCREEN_SIZE)

pygame.display.set_caption("Graphic Emerge Simulation")

clock = pygame.time.Clock()

#---------------------------- Begin Sim Code ----------------------------
for trial_index in range(len(trials)):
  if app_exit:
          break

  n_agents, n_tile_side = trials[trial_index]

  #n_agents = int(input("Number of agents (must be square and even for now): "))
  #n_tile_side = int(input("Dimension of map (sidelength in # of tiles, must be factor of MAP_PIX
for now): "))
```

```
for i in range(n_agents):
        COLOR_SCHEME[i + 1] = COLOR_SCHEME[(i % 4) + 1]

minimap_pix = MAP_PIX // math.sqrt(closest_minimaps(n_agents))
map_tile_pix = MAP_PIX // n_tile_side
minimap_tile_pix = minimap_pix // n_tile_side
minimap_top_lefts = []
size = [n_tile_side, n_tile_side]

for j in range(int(math.sqrt(closest_minimaps(n_agents)))):
        for i in range(int(math.sqrt(closest_minimaps(n_agents)))):
   minimap_top_lefts.append([MAP_PIX + (minimap_pix * i), minimap_pix * j])

done = False
solve_steps = 0
interactions = 0

reference_map = []
for i in range(size[1]):
        reference_map.append([0] * size[0])

black_map = []
for i in range(size[1]):
        black_map.append([-1] * size[0])

empty_map = list(reference_map)

#Adding items and obstacles
n_obstacles = int((size[1] - 3) // 4)
obstacle_length = int((size[0] - 9) // 2)

#'Swear to god I brush my teeth / Never sleepin on some beef / Bet, turnin over a new leaf'

#Centered:
#reference_map[size[1] // 2][size[0] // 2] = -2
reference_map[size[1] - 6][size[0] - (4 + (obstacle_length // 2))] = -2
```

```
  #Adding obstacles
  for obstacle_row in range(n_obstacles):
          #Make row at y = 3 + (obstacle_row * 4)
          for obstacle_index in range(obstacle_length):
          reference_map[3 + (4 * obstacle_row)][3 + obstacle_index] = -3
          reference_map[3 + (4 * obstacle_row)][size[0] - (4 + obstacle_index)] = -3

  #reference_map[size[1] // 3][size[0] // 3] = -3
  #reference_map[size[1] // 3][(2 * size[0]) // 3] = -3
  #reference_map[(2 * size[1]) // 3][size[0] // 3] = -3
  #reference_map[(2 * size[1]) // 3][(2 * size[0]) // 3] = -3

  #robots = [Agent([2, 2], 1, size), Agent([2, size[1] - 3], 2, size), Agent([size[0] - 3, size[1] - 3], 3,
size), Agent([size[0] - 3, 2], 4, size)]
  robots = []
  for i in range(n_agents):
          robots.append(Agent([(3 * i) + 1, 1], i + 1, size))

  for bot in robots:
    reference_map[bot.y][bot.x] = bot.id

  screen.fill(BLACK)

  while not done:
          for event in pygame.event.get():
     if event.type == pygame.QUIT:
          done = True
          app_exit = True
          print('Sim cancelled. \nnumber of bots:', n_agents, '\ntiles per side:', n_tile_side, '\nsteps to
solve:', solve_steps)

          if event.type == pygame.KEYDOWN:
          if event.key == pygame.K_SPACE:
          pass

          solve_steps += 1

          for bot in robots:
```

```
            previous_position = [bot.x, bot.y]
        bot.move(reference_map)
        reference_map[previous_position[1]][previous_position[0]] = 0
            reference_map[bot.y][bot.x] = bot.id

            if bot.state == 1:
            done = True

            #draw_map(reference_map, [0,0], map_tile_pix)
     # draw_map(reference_map, [0,0], MAP_PIX)
            #for bot_i in range(len(robots)):
     #   draw_map(robots[bot_i].map, minimap_top_lefts[bot_i], minimap_tile_pix)
            #draw_mini_map(robots[bot_i].map, minimap_top_lefts[bot_i], minimap_pix)

            pygame.display.flip()

            #if trial_index == 0:
            # pygame.image.save(screen,'emerge-' + str(solve_steps) + '.jpg')
     # clock.tick(FPS)

            if solve_steps > 3000:
            done = True
      print('number of bots:', n_agents, '\ntiles per side:', n_tile_side, '\nsteps to solve:', solve_steps,
'\ninteractions:', interactions)
      trials[trial_index].append(solve_steps)
      trials[trial_index].append(interactions)
pygame.quit()

with open(file_name + '.csv', 'w', newline='') as csvfile:
            filewriter = csv.writer(csvfile, delimiter=',',
                        quotechar='|', quoting=csv.QUOTE_MINIMAL)
            filewriter.writerow(['number of agents n', 'map size m (m tiles x m tiles)', 'solve steps',
'number of interactions'])

            for trial in trials:
        filewriter.writerow(trial)
```