Pathfinding Using Swarm Simulation

Abstract

Pathfinding, the process of identifying and optimizing routes through various environments, is a ubiquitous mechanism, appearing in numerous fields, including the control of autonomous robots. In the case of controlling several such autonomous robotic agents, it may be applicable to distribute the computation of this process among the agents by utilizing principles of swarm intelligence. The researcher programmed a computer simulation of up to 2000 autonomous agents moving within an environment. Each agent does not know its location and can only sense its immediate surroundings. By communicating with others nearby and relaying messages to one another, agents demonstrated capabilities for identifying and optimizing efficient routes between target locations in the environment. In situations where multiple routes were possible, the agents usually preferred shorter routes, and any specific route around obstacles took shape to minimize the path length. Using only detection of obstacle surfaces directly ahead of them, agents were able to be programmed to avoid colliding with obstacles, instead rotating smoothly to evade them. Overall, this particular method of pathfinding requires minimal computational capability to be present in each individual, instead spreading the task across several such agents. Such a method as this for distributed pathfinding can be especially useful in navigating unknown environments, as the entire process occurs without any single agent requiring large-scale knowledge of the environment. There may also be applications in situations where communication of all agents with a central authority is difficult, as with this method, no such central authority exists.

Introduction

Pathfinding, the process of identifying and optimizing routes through various environments, is a ubiquitous mechanism, appearing in numerous fields, including the control of autonomous robots. This project investigates the effectiveness of using simulated swarms of agents for the purpose of solving pathfinding problems. While traditional, algorithmic solutions for finding optimal paths often take much longer to run as the complexity of the environment increases, swarm simulation pathfinding offers a more heuristic approach that may have the potential to scale better while still providing good solutions to pathfinding problems. Another advantage of this approach is that the agents do not need to have any advance knowledge of the environment they will be navigating, nor do they individually need to have much computation power. This would be especially useful if translating this approach to use autonomous drones as agents conducting, for example, a search-and-rescue mission. Simulated pathfinding still has its applications, however: when planning the construction of new roads, simulated pathfinding can help to identify routes for these roads to take through and around difficult terrain and obstacles.

Literature Review

The researcher read over the ISEF rules for pre-college science research (“International,” 2023).

A study presented at the International Conference on Robotics and Artificial Intelligence tested the prospects of using simulated Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO) for path planning. In PSO, particles represent potential solutions to a pathfinding problem, and particles move within an abstract search space based on local and global properties. In ACO, agents move around a space and communicate by leaving simulated pheromone trails. The study found a hybrid algorithm of PSO and ACO to be effective in solving pathfinding problems efficiently (Mathew, et al., 2021).

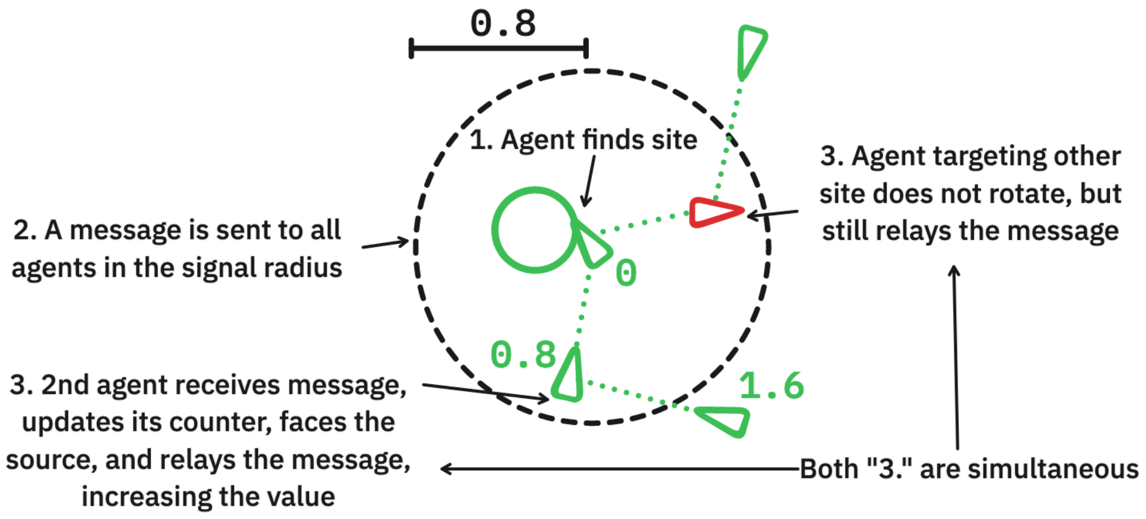
Decentralized pathfinding through swarm intelligence principles can have applications in the control of groups of unmanned vehicles for use in navigating dangerous situations such as earthquakes and wildfires. In these situations, connection of all individual drones to a central control hub may be unreliable, increasing reliance on local communication with nearby drones. This amplifies the utility of distributing computation among drones using swarm intelligence principles (Pyke & Stark, 2021).

“Boids” refers to a control model for the behavior of groups of autonomous agents such as flocks of birds or schools of fish. This model incorporates three primary types of interactions between agents: separation, i.e., agents attempt to maintain adequate distance between themselves and others surrounding them; alignment, i.e, agents attempt to face in a similar direction to others around them; and cohesion, i.e., agents will attempt to move towards the average position of those around them, so as to remain in a close group. These principles of swarm behavior have applications in the context of swarm robotics and autonomous control (Reynolds, 1987; Wong, 2008).

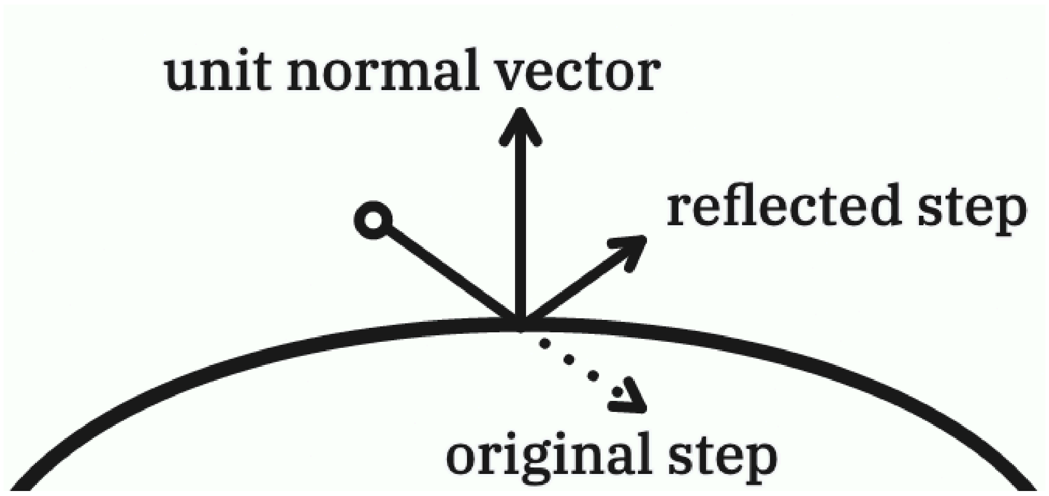
“Dijkstra’s Algorithm” is an algorithm for identifying optimal paths between nodes of weighted graphs. The algorithm functions by beginning at one target node and “exploring” the graph by traversing to adjacent unexplored nodes, annotating these nodes with values based upon the explored node from which the algorithm came and upon the weight of the edge that was travelled. This process is repeated iteratively until the other target node is reached, at which point the annotated values of the other nodes are used to identify a path between the target nodes that optimally minimizes the total weight of all edges travelled along the path (Dijkstra, 1959).

Methodology

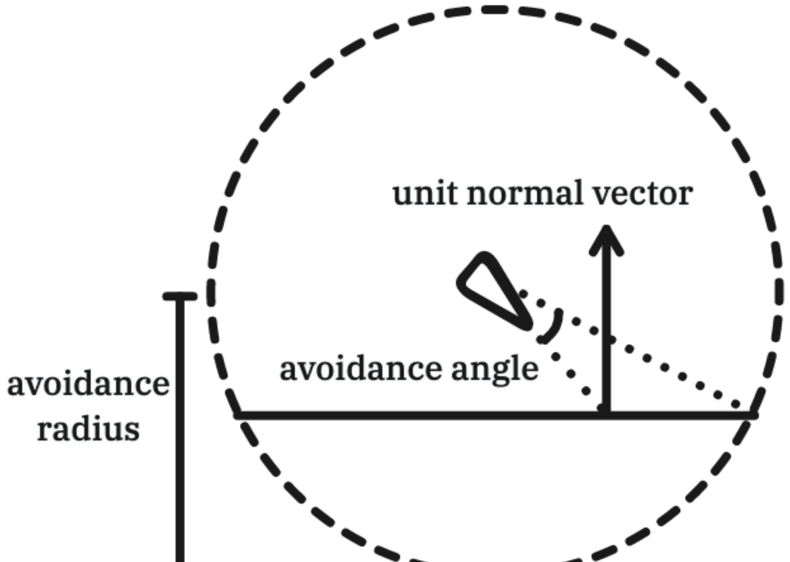
The researcher programmed a computer-based simulation of many “agents,” each with a limited set of individual capabilities with regard to movement and logic, in which there exist multiple “sites” at certain locations, between which the particle swarms are tasked with finding routes that navigate around various obstacles present in the scenario. Each individual agent keeps track of a maximal distance from each site in the scenario. These distance counters are incremented at each frame of the simulation by a quantity equal to the distance travelled by the corresponding agent during that frame. Such an increment assumes a worst-case scenario in which the agent stepped farther from every site in the scene. This assumption is made with the objective of ensuring that no agent ever underestimates its distance from any site, but rather only overestimates it. When an agent encounters a site, it resets its distance counter corresponding to that site to 0, and transmits a message to all other agents within a certain radius. This message communicates the ID of the site, and a distance equal to the radius of communication. This ensures that all agents that receive the message are no farther from the site than the distance contained in the message. When an agent receives such a message, if it informs of a distance that is smaller than that which the agent already has in its counter for that site, the agent will update its counter to the new distance, and relay a new message to all other agents within range. This new message will contain the distance from the original message plus the maximum communication range, again ensuring that all distance counters contain only overestimates. Agents keep track of what site they are currently targeting, and when an agent receives a message about its target site that contains a distance shorter than the one present in the agent’s memory, the agent will turn to face the direction from which it received the message. Agents also have a random chance to spawn as “scouts” that do not seek sites at all, only relaying messages.


0.8
1. Agent finds site
2. A message is sent to all agents in the signal radius
3. Agent targeting other site does not rotate, but still relays the message
0
0.8
1.6
3. 2nd agent receives message, updates its counter, faces the source, and relays the message, increasing the value
Both "3." are simultaneous

Obstacles in pathfinding environments are constructed as generic objects defining methods to provide the obstacle's bounding box, to check if a given coordinate lies inside the obstacle, and to provide all intersections of a given ray with the boundary of the obstacle, along with the surface unit normal vector at each of said intersections. The program defines these methods for three types of obstacles: circles, triangles, and coordinate-aligned rectangles. The rendering of the obstacles is done by using the first method to choose what pixels to check, and using the second method to conduct a check of which pixels to color. To ensure that agents are unable to move through obstacles, each simulation frame, all agents check whether the step they take that frame would cause them to end inside any obstacle. In the event that a given agent finds that it shall collide with one or more obstacles, it uses the obstacles' provided raycast methods to find points of collision with each obstacle. The agent then selects the obstacle with which it shall collide first, and uses it to conduct the following collision computation. The point of collision is used to separate the portion of the agent's step occurring after the collision from that occurring before. The portion occurring before the collision is maintained, while the portion occurring after the collision is reflected according to the collision normal vector, as calculated by a dot product.



Each agent is granted the capability to detect obstacle surfaces within a short distance in the agent's heading direction. If a wall is detected within a close enough range, the agent will use the normal vector of the detected surface to approximate the wall as a straight line and determine an angle by which to rotate in order to place the new point of intersection between the straight approximation of the wall and the agent heading precisely at the boundary of the obstacle avoidance range. Steering using this angle would cause agents to asymptotically approach detected walls. To instead cause agents to curve away from walls, a constant tolerance is added to this steer angle, and the result is used to rotate the agent away from the obstacle wall.

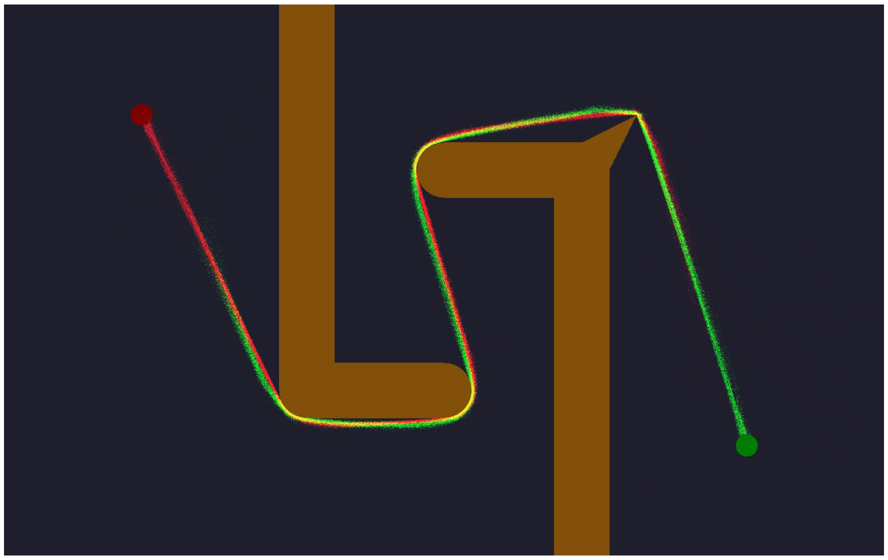


In order to better visualize the large-scale behavior of the agents and the routes they tended to overall prefer, a special render mode was programmed for the simulation, in which individual agents are not rendered, but agents seeking sites leave faint trails in the rendered pixel buffer in the color of the targeted site. These trails fade over time at a constant rate. The trail left by one agent is too faint to be seen, but many agents traveling through the same regions will combine in trail intensity, highlighting the routes most often taken by the most agents.

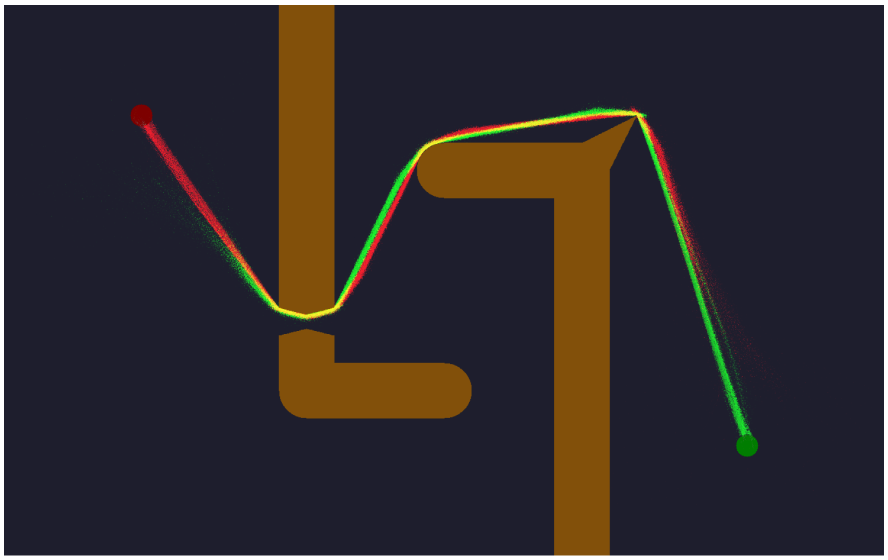
The researcher created multiple different environments in order to assess the manners in which the algorithms described affect the collective behavior of the agents and their ability to create optimal and/or efficient routes between sites. Data was collected detailing each time an agent travelled a path that was shorter than the shortest path seen previously. The timestamps and path lengths of these events were recorded.

Results

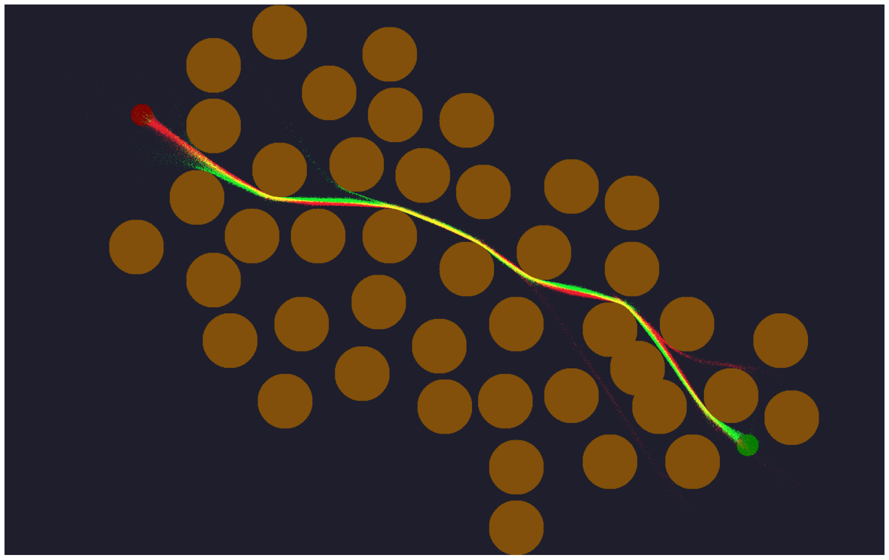
The following images from the simulation display the trails discussed in “Methodology,” as opposed to the agents themselves, so as to portray the overall collective behavior of the agents.



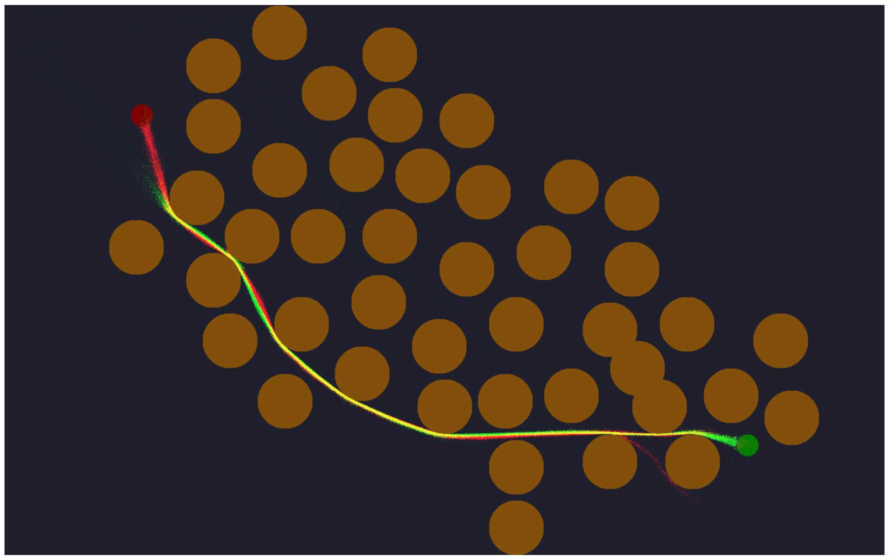
Environment 1: The agents identify the only route between the sites, traversing it in an optimal path.



Environment 1, modified: A gap has been introduced in a wall, creating a new, shorter route. The agents identify and travel the new route, again optimizing the length of the path.

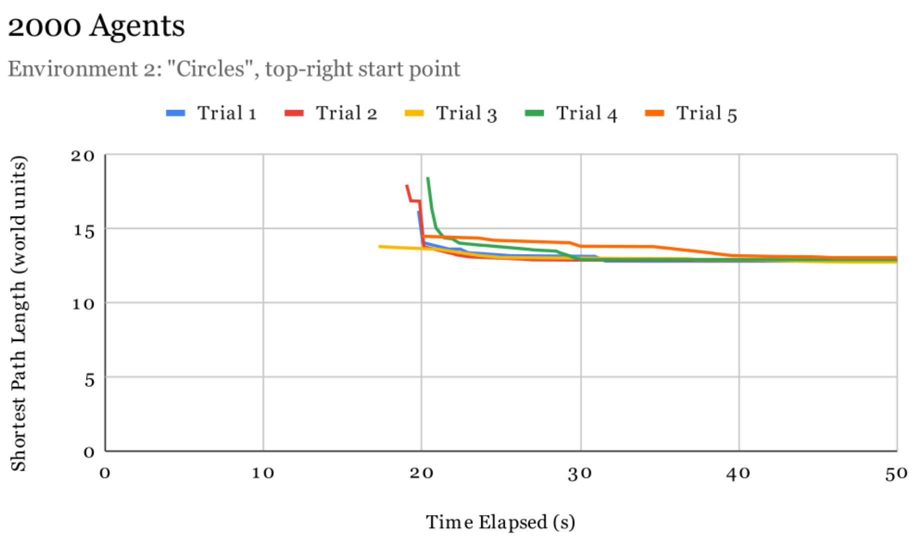


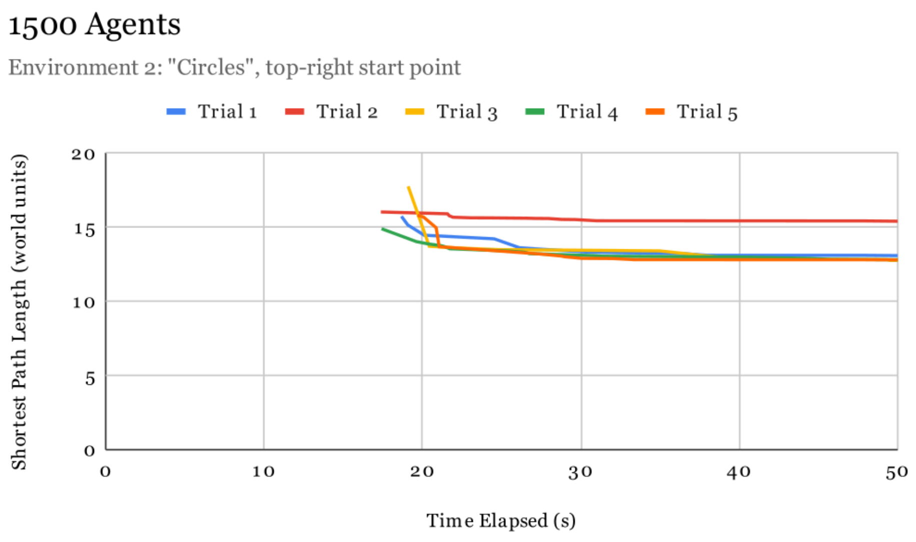
Environment 2: Having begun in the upper-right corner of the scene, the agents identify an efficient route around the obstacles.



Environment 2: Having begun in the lower-center region of the scene, the agents identify a different route than before that is still relatively effective.

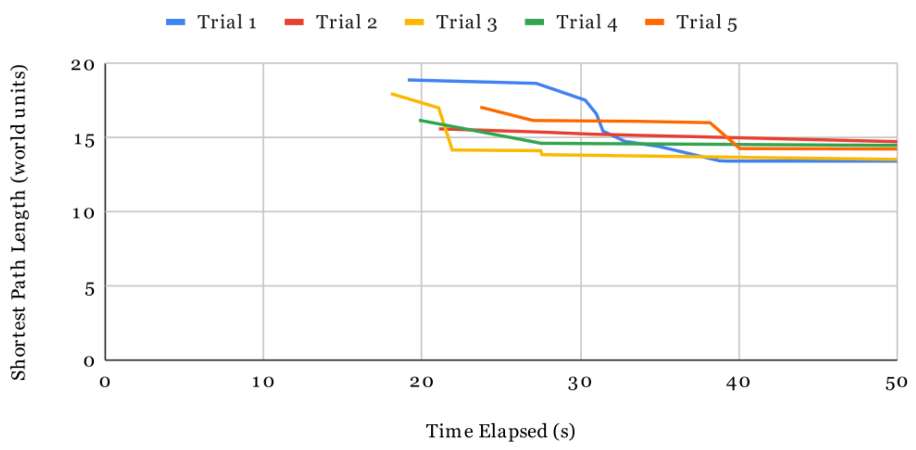
The following graphs display the convergence of the agents upon optimal paths within the aforementioned Environment 2 (“Circles”) when different quantities of agents are present. All environments are a measure of 16 world units across by 10 world units tall.





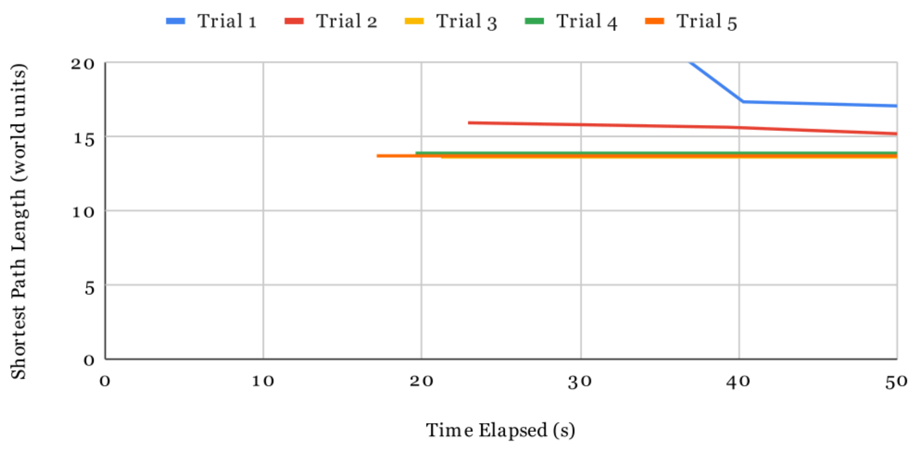
1000 Agents

Environment 2: "Circles", top-right start point



500 Agents

Environment 2: "Circles", top-right start point



Discussion

The simulated swarms of agents demonstrated capability to create efficient routes between target locations. When faced with multiple possible routes, they tended to prefer shorter ones. In traversing a given route, the agents tend towards the shortest way to travel that route, straightening segments of it to optimize path length. However, the agents struggled to identify certain routes that require navigating through very small gaps between obstacles. This problem was often amplified by the tendency of the obstacle avoidance system to cause agents to travel approximately parallel to walls and be therefore less likely to travel through sudden openings. This is not, however, an inherent fault of the pathfinding system, and may be remedied by additional sensing capabilities for each agent. Also, the algorithm took much longer to identify an effective route when fewer agents were present, on account of the fact that the algorithm relies on agents being within communication distance of each other. This is clearly evidenced by the convergence data, which shows that 500 agents were only in select cases even able to solve the pathfinding problem at all. This problem may be able to be addressed by agents communicating at multiple ranges, so as to increase potential for discovering routes without sacrificing path granularity. This method was not able to be tested in this project due to computational limitations. Additionally, the convergence data for agent counts less than 2000 lacks consistency, indicating that with smaller agent counts, the algorithm relies considerably upon random chance for the small number of agents to be able to identify an effective route.

Limitations

The overall potential of the simulation was limited by the computational power available to the researcher. This resulted in limits upon what capabilities the simulated agents could possess while allowing the simulation to run at an acceptable speed. Additionally, the simulation takes place in only two dimensions, so it may not account for elevation changes in the environment or the complexities of 3D pathfinding in the context of, for example, aerial drones.

Future Studies

Additional studies of the algorithm may be performed using more powerful computational tools. The simulation could also be modified to run on a graphics processor to further increase performance. This would permit testing larger communication ranges. This extra computation power may also enable the addition of aspects to the simulation to increase applicability, such as allowing agents to see in a range around them and/or ahead of them, to increase efficiency in identifying routes. Other studies on this topic may also address prospects of additional information being communicated between agents, rather than just distances to sites.

References

International rules for pre-college science research. (2023). Society for Science. Retrieved August 16, 2023, from https://sspcdn.blob.core.windows.net/files/Documents/SEP/ISEF/2024/Rules/Book.pdf

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1), 269–271. https://doi.org/10.1007/BF01386390

Mathew, A., Paul, A., Rojan, A., Thomas, A. (2021). Implementation of swarm intelligence algorithms for path planning (1831; *Journal of Physics: Conference Series*). Institute of Physics. https://iopscience.iop.org/article/10.1088/1742-6596/1831/1/012008/pdf

Patel, A. (2023). Applications. Amit's Thoughts on Pathfinding; Stanford University. Retrieved September 15, 2023, from https://theory.stanford.edu/~amitp/GameProgramming/Applications.html

PEC. (2018, July 3). How to reduce eye strain when working with computers. Piedmont Eye Care. https://charlotteoptometry.com/how-to-reduce-eye-strain-when-working-with-computers

Pixels Library Documentation. (2023, June 13). Docs.rs. https://docs.rs/pixels/0.13.0

Pyke, L. M., & Stark, C. R. (2021). Dynamic pathfinding for a swarm intelligence based UAV control model using particle swarm optimisation. Frontiers in Applied Mathematics and Statistics, 7. https://www.frontiersin.org/articles/10.3389/fams.2021.744955

Rand Library Documentation. (2022, February 14). Docs.rs. https://docs.rs/rand/0.8.5

Reynolds, C. W. (1987). Flocks, herds and schools: A distributed behavioral model. ACM SIGGRAPH Computer Graphics, 21(4), 25–34. https://doi.org/10.1145/37402.37406

Ron Library Documentation. (2023, August 17). Docs.rs. https://docs.rs/ron/0.8.1

Rust Standard Library Documentation. (2023, December 21). Rust Programming Language. https://doc.rust-lang.org/std/

Serde Library Documentation. (2023, October 12). Docs.rs. https://docs.rs/serde/1.0.189

Winit Library Documentation. (2023, May 14). Docs.rs. https://docs.rs/winit/0.28.6

Wong, T. (2008, September). Boids. https://cs.stanford.edu/people/eroberts/courses/soco/projects/2008-09/modeling-natural-systems/boids.html