University of North Carolina Asheville
Journal of Undergraduate Research
Asheville, North Carolina
December 2022

SwarmBot

Christopher Blaha
Department of Computer Science
The University of North Carolina Asheville
One University Heights
Asheville, North Carolina 28804 USA
cblaha@unca.edu
Faculty Advisors: Dr Marietta Cameron, Dr Kenneth Bogert

Abstract

Swarm intelligence utilizes multiple agents that behave like a swarm of insects or other animals to optimize the skill of an AI program. Ants follow a pheromone to the optimal path, fireflies group towards the brightest bug, and cuckoo birds lay their eggs in other bird's nests to pass off as the other bird's babies. Those that survive pass on a genetic algorithm that dictates behavior towards the optimal path. Multiple AI agents using these behaviors simultaneously have an advantage over a single agent working alone. They can cover more ground when searching for the winning strategy. They also have a considerably faster performance compared to a single agent. This project utilizes the ant colony, firefly, and cuckoo bird swarm methods to compare performance in the game Othello.

1. Introduction:

The idea of swarm intelligence employs multiple agents to achieve a goal instead of a single agent. This project visits the game Othello. A gametree is a map of all possible moves in the game. With a state space of about 10^{28} , it is difficult to navigate the entire tree. An AI bot uses different methods to navigate a gametree to find the optimal path to a winning strategy. One such method is the Monte Carlo Tree Search (MCTS). A single agent simulates down the tree by picking nodes at random. The leaf node is evaluated. The agent back propagates up the tree and takes note if the simulation is a good strategy. Swarmbot uses multiple agents in parallel to mimic the behaviors of different swarms of animals. Specifically three behaviors are explored, the ant colony, the firefly, and the cuckoo bird.

2. Background:

Swarm intelligence takes on different varieties and mimics the behavior of animals and insects that have a swarm behavior "Ants, in spite of their individually limited capabilities, seem to be able to collaborate in solving problems that are out of reach for any single ant. We speak in this case of the emergence of a recognizable collective behavior in which the whole is more than the sum of the individual parts. The term swarm intelligence is also used, especially in connection with other insects such as bees. Moreover, in certain tasks that ants perform, such as the construction of a cemetery for dead ants, there seems to be no centralized control but, despite the short-sighted local vision of each single insect, a global coherence does emerge. This kind of phenomenon is typical of complex systems and one also speaks of self-organization in this context."[1] The book here talks about the ant colony, firefly, and Cuckoo bird methods. It also talks about the generic particle swarm method. These methods can be found in chapters 5, 6, and 7[1]. The ant colony will use pheromones as a guide towards the optimal path. The more ants go down a certain path, the stronger the pheromone will be. The fireflies will gravitate towards the brightest fireflies. The Cuckoo bird will lay eggs in other bird's nests and try to trick the other bird into taking care of them. The survivors will move on towards the goal and the ones that get discovered will have a probability of either getting destroyed, or the nest moves to another location. These swarms will also make use of genetic algorithms. " The GAs (genetic algorithms) are adaptive heuristic search methods based on principles of natural evolution and genetics. The GA encodes the decision variables of a problem into finite-length strings of alphabets. The strings, also referred to as chromosomes, are the candidate solutions to the search problem. A chromosome is composed of a sequence of genes from a certain alphabet. An alphabet could consist of continuous values, binary digits, integers, symbols, matrices, etc. The

representation of chromosomes depends on how the problem is structured in the GA."[2] The genetic algorithms applied to the agents in the swarm will provide additional heuristics to help the bot.

3. Description:

3.1 requirements:

This bot is made in C++ with the OpenMP library to handle the parallel threading. One thing that needs to be kept in mind is that I am adding more code and functionality to a bot that already uses a heavy amount of processing. Keeping the core of the program's gameplay and logic at its most efficient representation possible is an important baseline to start off. Two main implementations are the board representation and the gametree's structure.

This program uses a bitboard representation for the game's board functionality. This is the fastest board representation for an othello program to use. With the gametree's structure, every node has a varied number of children nodes representing the possible moves at that point in the game. This creates an N-ary tree that causes a lot of nested looping to search through and is not very efficient in this case. By making each node have a pointer to a child and a pointer to a sibling creates a binary tree structure where each node can search all its children by going to the initial child and then traversing that child's siblings. This gives the search one loop to use instead of multiple loops, giving an efficiency of 0(log n).

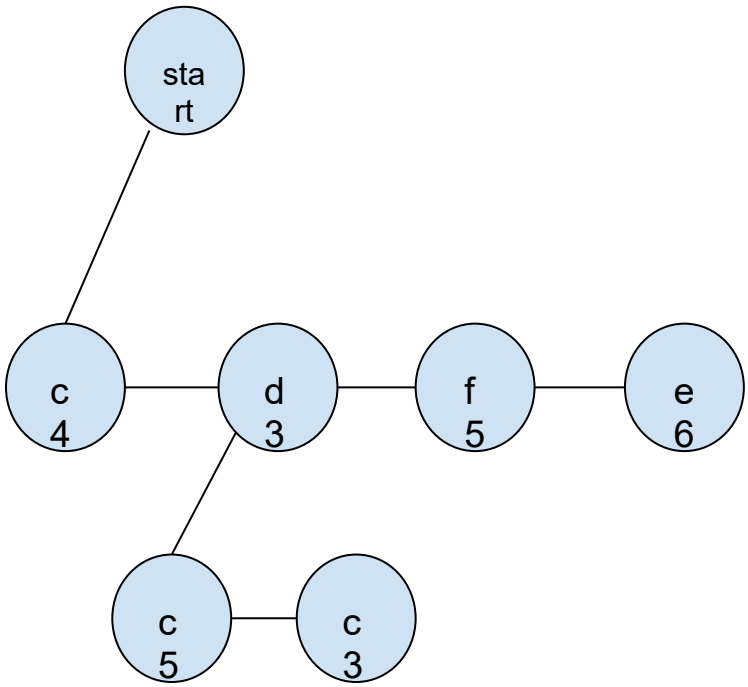

Figure 1: tree structure

3.2 Design:

Swarmbot starts the game with a representative agent. This agent regulates how many swarm agents to send down the gametree and decides what move to make during gameplay. At the basic level, Monte Carlo agents can be sent down in a normal Monte Carlo fashion. It will simulate n number of dives consisting of simulate, evaluate, and back propagate. As the agent traverses down the tree, a random decimal between 0 and 1 is generated. If that random number is greater than 0.5, the agent moves further down the tree. As the agents travel back up, each node traversed is marked with the number of dives (NI) and number of winning simulations (WI) found at that level. After the swarm agents are finished, the representative agent will look at each available move and determine which is the best move to take.

The next type of agent is the ant colony agent. These agents will traverse the gametree in the same fashion. In addition to the random traversal, the WI is used as a pheromone trail. If a WI is greater than 0, then WI/NI is added to the random number. This has the ant colony agents following a pheromone trail, but still has a chance to wander off trail periodically to look for more winning paths.

For the next two agent types, a genetic algorithm is needed. An array of 31 integers represents the moves made in a game by one player. 30 are filled in with random integers. The last one tells how many winning simulations that agent has made. If the integer at the ith move matches that of the GA's move at that index, then that path is given a boost so the agent favors the path dictated by the GA. After n simulations the agent with the higher number of winning simulations is mutated with lower winning GAs to produce an offspring. This gives the agents a survival of the fittest angle. This aspect is important for the next two agents. The firefly agent traverses the gametree with their GA being favored during simulation. The GA with the most wins represents the brightest firefly. Other fireflies mutate with the brightest one to produce offspring that will start to group together towards the optimal win. The cuckoo bird agents traverse the tree in a similar fashion. Each agent traverses to a leaf node. That represents a nest where the eggs have been switched with the cuckoo bird egg. A random decimal is generated to determine if the cuckoo bird has been discovered. A favorable GA will give that probability a boost to lower the chances of being caught. If the cuckoo bird is caught, another random decimal is created, also influenced by the bird's GA, to determine if the bird is destroyed, or gets to carry on its GA. All three of these swarm algorithms present a more targeted approach to the MCTS method.

One thing to point out on this program is that when you have multiple agents traversing a tree in parallel, You end up having agents tripping over each other. This ends up causing situations where one agent will scan a node in a simulation and switch to the other player to indicate on the tree who made that move. At the same time, another agent will hit that same node at the same time and switch the player back to the first player. This causes boards that are listing the wrong set of legal moves, and even cause some boards to be flipped. You end up with this dilemma where you can solve the problem by setting certain points where the agents have to wait their turn, but that comes at the expense of performance. I was able to get rid of the bug completely, but the agents were waiting in line for too long and took longer than a single agent takes to perform the same amount of work. This presents this interesting problem of where to place the critical points for the agents to do one at a time while not slowing down the program too much. I was not able to completely get rid of the bug, but it was slowed down. The bug becomes more prevalent with more agents doing more dives. You have to gauge how many agents and how many dives the bot can handle to stave off the bug.

3.3 Required Resources:

For software, I used a C++ IDE to program the bot. My computer with 8 cores on the processor gives me a swarm size of up to 8 agents. That is good enough to show improvements that are made with varying swarm sizes in comparison to the single monte carlo agent. Other Othello bots like eothello.com and an othello2018 game from Steam will be used to play against to collect metrics. The Google charts API will be used to graph collected information into tree graphs and performance graphs. These graphs will be put on crispspicestudios.com/swarmbot/swarmbotresults to interact with and view.

3.4 Testing

The Monte Carlo, Ant Colony, Firefly, and Cuckoo Bird swarms will be evaluated on three metrics. First will be the Gametree. Does the Gametree reflect the swarm's behavior? The Gametrees of the Ant Colony, Firefly, and Cuckoo Bird should show a targeted behavior towards the optimal path compared to Monte Carlo's random simulations. Second is performance. How fast does the swarm perform? 3000 simulations per move in parallel are timed during gameplay. These times are averaged and put on a graph showing average seconds with different size swarms of 2 to 8. Third is the win percentage. Do these improvements make a difference against an opponent? Win percentages are recorded.

4 Results

On the gametree behavior, figures 2.1 and 2.2 show the Monte Carlo agents at the top of the gametree and a few clicks down the tree. 2.3 shows the performance of Monte Carlo from 1 to 8 agents.

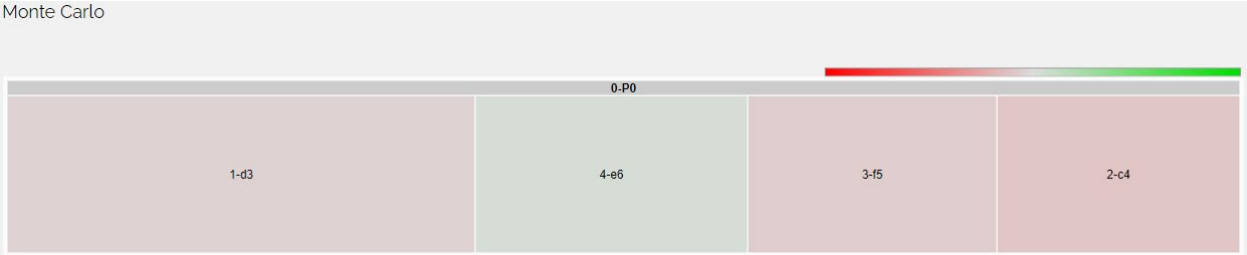

Figure 2.1: Monte Carlo tree graph

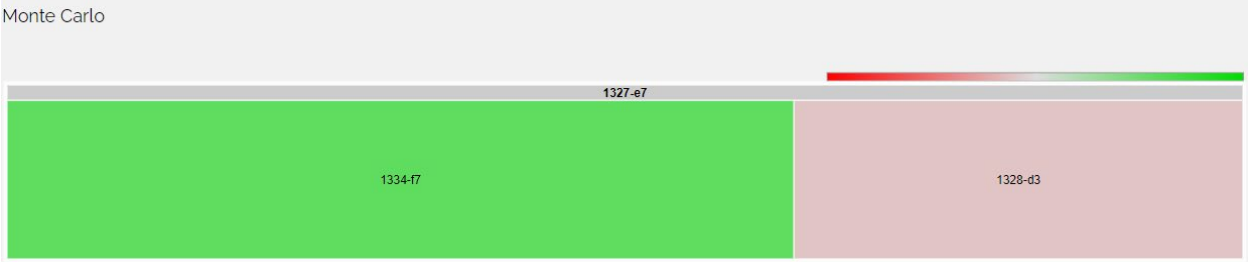

Figure 2.2: Monte Carlo tree graph further down

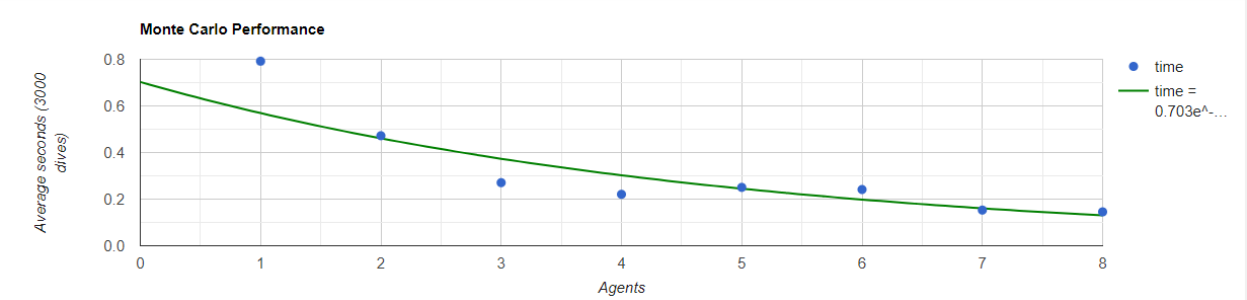

Figure 2.3: Monte Carlo performance

The top of the tree shows a spread out behavior that comes with simulating games at random. The colors indicating how many wins were found are faded, but become more pronounced as you travel down the tree. The performance graph shows improvements in time as more agents are added. Playing against the eOthello bot, the Monte Carlo bot shows about a 36% win rate. Figures 3.1, 3.2, and 3.3 show the Ant Colony

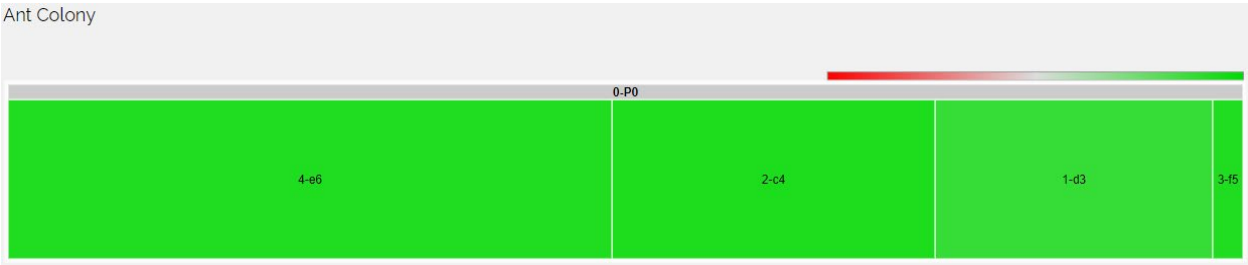

Figure 3.1: Ant Colony tree graph

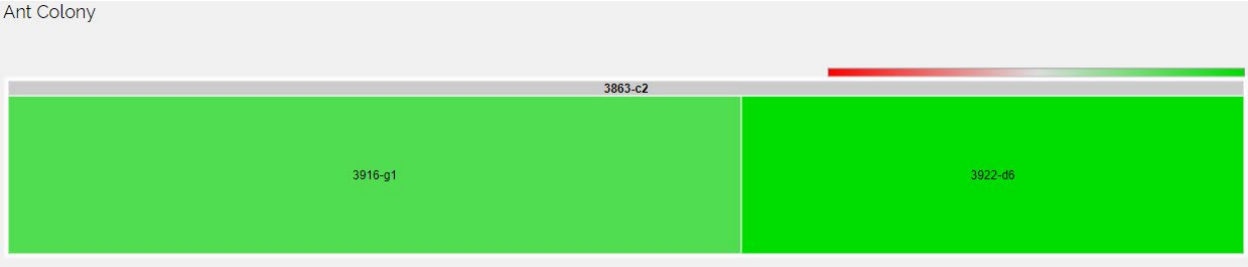

Figure 3.2: Ant Colony tree graph further down

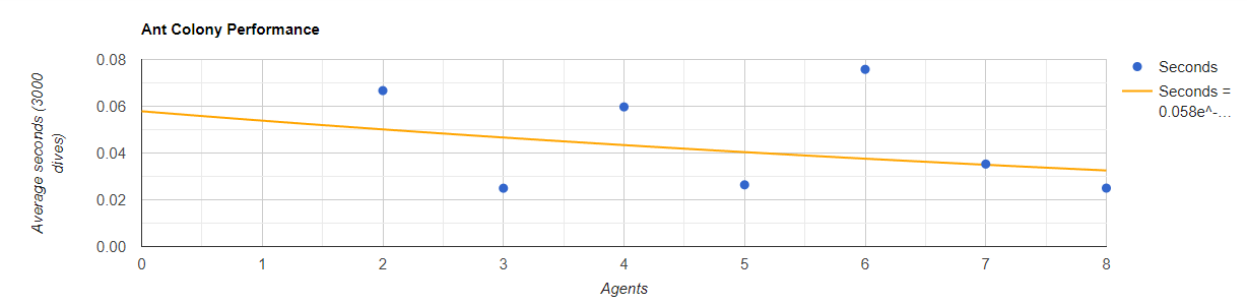

Figure 3.3: Ant Colony performance

The Ant Colony's tree graph shows a more targeted search on this tree structure. Solid green colors on the nodes indicate the greediness of this algorithm to focus on the winning paths and ignore the less optimal paths. This targeted behavior is consistent further down the tree. On the performance graph, you see considerable improvement in speed. Note that in figure 2.3, the speed is graphed in tenths of a second. Figure 3.3 is graphed in hundredths of a second, and even an Ant Colony swarm of 2 outperforms a Monte Carlo swarm of 8. When playing against the eOthello bot, The Ant colony swarm outperformed the rest of the swarms with a 61% win rate.

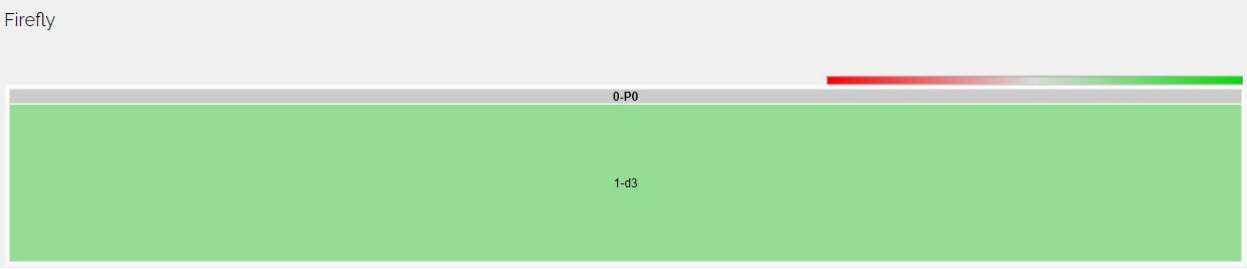

Figure 4.1: firefly tree graph

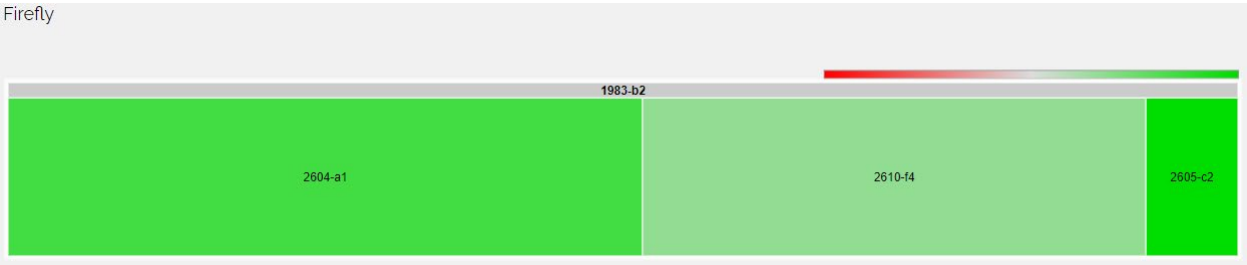

Figure 4.2: firefly tree graph

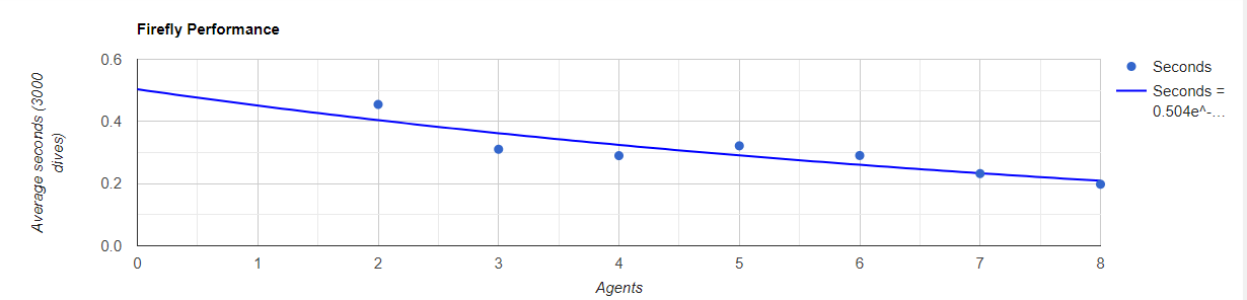

Figure 4.3: firefly performance graph

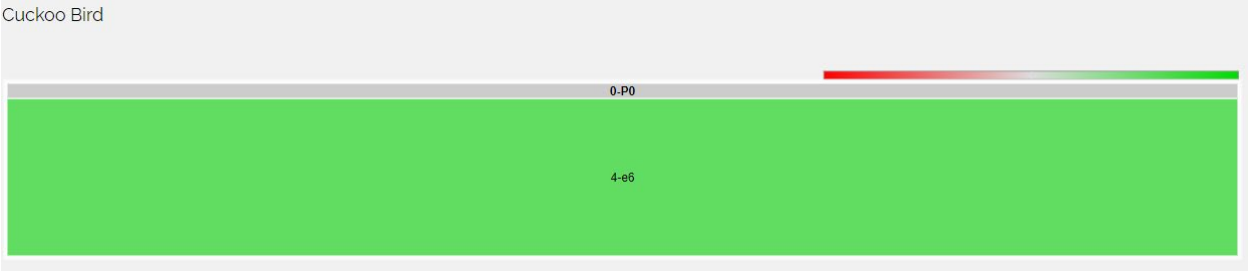

Figure 5.1: Cuckoo bird tree graph

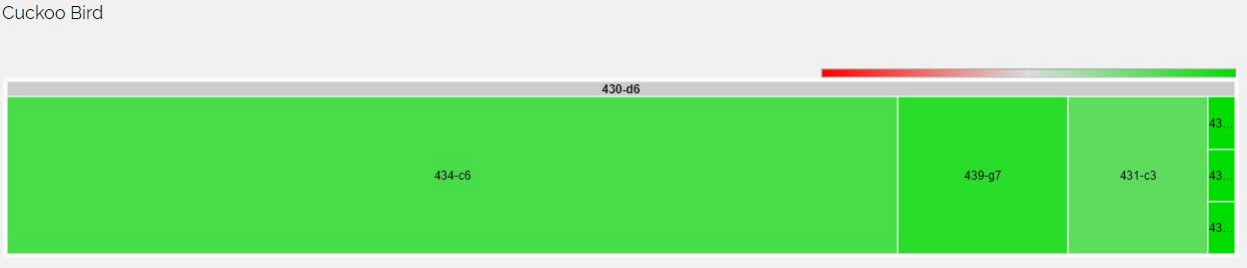

Figure 5.2: Cuckoo bird tree graph further down

Figure 5.3: Cuckoo bird performance graph

Both the firefly and cuckoo bird methods show similar performance in tree structure. You'll notice that these tree maps have only one node at the top. This is because this was in games where the agent played second. The first move was chosen for them. We see a similar start to the Monte Carlo agents as it starts out the same way. When the genetic algorithms mutate throughout the game, the agents start dictating behavior towards the optimal goal and grouping together in their searching. You see the same faded kind of colors, but the colors start getting more solidified further up the tree compared to the Monte Carlo agent. Performance wise both Firefly and Cuckoo Bird perform about the same speed as the Monte Carlo agent. On performance against the eOthello bot, both the Firefly and Cuckoo Bird swarms were close to each other with 41% for the Firefly and 43% for the Cuckoo Bird. These results can be found at www.crispspicestudios.com/swarmbot/swarmbotresults.html and of course the results will change as the bot performs more against different bots.

5 Conclusion/Future work

These three swarm methods improve the performance of a bot playing the game Othello. Using a well built othello bot for a baseline these swarms will allow the bot to make more targeted searches toward winning strategies. They perform faster compared to a single Monte Carlo Tree Search. For future work, I would like to implement more swarm types and have the representative agent be able to ensemble these different swarm types throughout the game. If Ant Colony swarm performs better in the beginning of the game, the agent will use more of those types. If Cuckoo Bird swarm performs better in mid game, the representative agent should choose more of those bots. I would like to have this bot on a website so it can be played against by the general public. I think this bot has the potential to be powerful

References:

[1]Chopard, B., & Tomassini, M. (2018). *An Introduction to Metaheuristics for Optimization (Natural Computing Series)* (1st ed. 2018 ed.). Springer.

[2]Peyman Abbaszadeh, Hamid Moradkhani, Hongxiang Yan,
Enhancing hydrologic data assimilation by evolutionary Particle Filter and Markov Chain Monte Carlo, Advances in Water Resources, Volume 111, 2018, Pages 192-204, ISSN 0309-1708,
https://doi.org/10.1016/j.advwatres.2017.11.011.

[3] a video on particle swarm optimization https://www.youtube.com/watch?v=JhgDMAm-imI

[4] Polikar R. (2012) Ensemble Learning. In: Zhang C., Ma Y. (eds) Ensemble Machine Learning. Springer, Boston, MA. https://doi.org/10.1007/978-1-4419-9326-7_1