Analysis and Comparison of Monte Carlo Tree Search versus ACO Algorithms in Distribution of Resources and Environment

Boyu Xu
School of Software and Internet of Things Engineering (Mailu Campus), Jiangxi University of Finance and Economics, Nanchang, China
910880407@qq.com

Abstract. Swarm intelligence (CI) has been widely studied in the past decades. The most famous CI algorithm is ant colony algorithm (ACO), which is used to solve complex path search problems. AlphaZero program, through self-reinforcement learning of Monte Carlo tree algorithm from scratch, has achieved transcendental results in go, chess and general chess. By analyzing and comparing the Monte Carlo Tree Search (MCTS) and ACO algorithms on Distribution of resources and environment, the reasons for AlphaZero's success are revealed. It is not only because of deep neural networks and intensive learning, but also because the algorithm is essentially an evolutionary algorithm of CI emergence.

1. Introduction
The concept of Group Intelligence (CI) derives from Condorcet's jury theorem in 1785: If each member of a voting group has more than half the chance to make the right decision, the accuracy of most decisions in the group increases with the number of members.[1] In the second half of the 20th century, CI was applied to the field of machine learning, and a broader consideration was given to how to design aggregates of agents to meet the goals of the whole system. This is related to the incentive shaping of single agent, and has attracted the attention of many researchers in the field of game theory and engineering. However, CI algorithms, such as the well-known ant colony algorithm (ACO), focus on how to make swarm intelligence emerge and surpass individual intelligence, lack the mechanism of evolving individual intelligence, so they cannot become self-evolving artificial general intelligence (AGI) without significant expansion. [2] Recently, AlphaZero algorithm achieves superhuman performance in Go, Chess and Chess games by using deep convolution neural network and reinforcement learning in self-game. [3] However, the reasons for Alpha Zero's success have not been fully understood. By analyzing and experimenting with AlphaZero, we can feel that the logical thinking of swarm intelligence is implied in the algorithm. Starting from the development and logical thinking of CI, this paper applies AlphaZero algorithm to the game of Distribution of resources and environment, demonstrates the evolutionary ability of deep neural network, and then compares the Monte Carlo Tree Search (MCTS) with ACO to identify MCTS as a CI algorithm.
2. Exploration of Cluster Intelligence Logic in AlphaZero

2.1. Review of AlphaZero Core Concepts
From the point of view of actual game, AlphaZero uses MCTS algorithm to search for the best dropout. Because the search time is limited, it is impossible to exhaust all possible dropouts, so the strategy network is used to reduce the search width, and the value network is used to reduce the search depth. As a prior probability of sampling, the strategy network searches for the possible winners with a higher probability. [4] From the training point of view, the strategy network and the value network are trained by the strategy iteration algorithm of reinforcement learning. MCTS is equivalent to the strategy enhancement operator, because the search probability is better than the probability of the strategy network. The strategy network is trained with the search probability as a label. The self-game based on MCTS is equivalent to the strategy evaluation operator. The strategy refers to the search probability of MCTS, the evaluation is to use the search probability to play chess, which is used as the label training value network.

2.2. Evolution of Individuals through Reinforcement Learning
Once individuals have enough expressive power, the next question is how to make them evolve. In order for individuals to evolve continuously, we need to find the direction of evolution. In AlphaZero, it is through the individual's own experience to find the direction of evolution, that is, through reinforcement learning. The result is that individuals can continue to evolve and eventually surpass previous versions and human experts. [5]

The reason why the label generated by MCTS is better than that of policy network is that MCTS includes multiple simulations, in each simulation, policy network is used to give a prior probability, and value network is used to update action value. The strategy and value network in each simulation can be regarded as an individual, and the search probability will become accurate as the number of individuals increases. Therefore, MCTS can provide CI, which refers to the search probability and the winning or losing results obtained by playing chess with the search probability. In literature [24], self-game based on MCTS is regarded as a strategy evaluation operator in reinforcement learning, but its strategy refers to the search probability of MCTS, not the original strategy network, which is inconsistent with the original strategy iteration algorithm. Therefore, it is more appropriate to consider MCTS as CI algorithm. [6]

3. The Training Method Applied to Distribution of resources and environment
In the whole tree search space, MCTS uses random strategies to conduct a large number of simulations to evaluate the state value. As the number of simulations increases, the search tree increases more and the estimation of state value is more accurate. Tree search strategy is also improving in the search process. Gradually, tree search strategy converges to the optimal strategy, and state value estimation converges to the real state value.

Figure 1 illustrates four steps in an iterative step in MCTS search, as follows:

Selection: Starting from the root node of the tree, the child nodes are selected recursively according to the selection strategy until the leaf nodes of the search tree are reached. The tree search strategy in TSP problem is selected according to the confidence upper bound method (UCT) in all the sub-nodes.

Expansion: The leaf nodes are expanded, and all feasible cities after the current node are selected as the sub-nodes of the current node.

Simulation: When the leaf node is reached, the current path length $L$ is obtained by walking according to the default strategy until the end point is reached. The default strategy of simulation is to select the feasible city of the current node according to the proportional prior probability $P(s,a)$.

Backpropagation: After completing a simulation, the whole search tree is updated according to the current simulation results.
ACO algorithm simulates the behavior of ant colony in real natural environment, and solves the combinatorial optimization problems such as TSP well. When ant colonies search for food, they start with random strategy searches around their nests. Once ants find food, they move food back from the food source to the nest. In the process of carrying food, ants release chemical pheromones on their return journey. The amount of pheromones released depends on the quantity and quality of food they find. When the ants search later, they can judge the direction of food source according to the number of pheromones and find food more quickly. Ant colonies share information among multiple individuals through pheromones, which enables them to quickly find the shortest path from nest to food source, which is shown in figure 2.

When solving the optimal problem, each iteration step consists of the following two main steps: simulation: each ant completes a complete search according to the probability distribution according to the state transition probability matrix, and the probability of selecting each path is proportional to the state transition probability matrix.

Update: Once all ants complete their search, a global pheromone update is required.

The whole search process is iterated by the above steps until the termination state is reached.
In this paper, the two algorithms are applied to 30 chess games. In addition, in order to compare with the two methods, pure random search is added as a control. The three methods are used to optimize 10 game problems, and the final results are shown in Table 1. Compared with random search, both ACO and MCTS show good convergence. In the first 100 iterations, MCTS is slightly better than ACO, but the search stalls in the second half. One of the main reasons is that because MCTS search is a tree structure and ACO search is a network structure, ACO has stronger ability to optimize local area path.

Comparing the specific algorithms in each iteration step of ACO and MCTS, we can find that MCTS has similar mechanism with ACO. In each iteration step, each individual needs to search according to a specific strategy and update the information in real time according to the global group sharing strategy. The similarities between the two algorithms are as follows:

Simulation strategy: In ACO, the strategy of simulation is obtained according to the state transition probability matrix. In MCTS, the part of search tree is obtained according to UCT strategy, and the part of simulation adopts default simulation strategy.

Group information sharing: In ACO, all output results are updated to the global pheromone, which determines the state transition probability matrix. In MCTS, the simulation results are updated, which affects the next UCT strategy selected in the search tree.

Balanced Exploration and Utilization: In ACO, the simulated action selection is proportional to the probability distribution, and ensures exploration and utilization, which is influenced by the superparametric Q. In MCTS, UCT algorithm guarantees balanced exploration and utilization, and is influenced by superparametric Cp.

Table 1. ACO, MCTS and Random Search Results.

|            | Best Value | Average Value |
|------------|------------|---------------|
| MCTS       | 426.75     | 456.93        |
| ACO        | 450.74     | 463.13        |
| Random     | 731.93     | 852.11        |

4. Conclusion
In this paper, Alpha Zero's core algorithms, Monte Carlo Tree Search (MCTS) and ACO, are tested and implemented on Distribution of resources and environment. The reasons for Alpha Zero's success are revealed. This algorithm is not only due to deep neural networks and intensive learning, but also essentially an evolutionary CI algorithm. Compared with random search, its accuracy is twice as accurate, and it has irreplaceable advantages compared with ACO.

References
[1] Evangelisti S, Bendazzoli G L, Ansaloni R, et al. A full CI algorithm on the CRAY T3D. Application to the NH 3 molecule [J]. Chemical Physics Letters, 1995, 233 (4): 353-358.
[2] Holden N, Freitas A A. A Hybrid PSO/ACO Algorithm for Discovering Classification Rules in Data Mining [J]. Journal of Artificial Evolution & Applications, 2008, 2008: 2: 1-2: 11.
[3] Xie Z, Fu X Y, Yu J Y. AlphaGomoku: An AlphaGo-based Gomoku Artificial Intelligence using Curriculum Learning [J]. 2018.
[4] Silver D, Schrittwieser J, Simonyan K, et al. Mastering the game of Go without human knowledge [J]. Nature, 2017, 550 (7676): 354-359.
[5] Browne C B, Powley E, Whitehouse D, et al. A Survey of Monte Carlo Tree Search Methods [J]. IEEE Transactions on Computational Intelligence & AI in Games, 2012, 4 (1): 1-43.
[6] Rajendran C, Ziegler H. Ant-colony algorithms for permutation flow shop scheduling to minimize makespan/total flowtime of jobs [J]. European Journal of Operational Research, 2007, 155 (2): 426-438.