I. INTRODUCTION

The performance of cooperative networks can be difficult to estimate, especially in environments where agents can communicate with each other. This is especially true in mobile agent environments where agents can move around and change their positions. This can make it difficult to estimate the performance of a cooperative network. However, the potential benefits of cooperative networks make them an attractive option for many applications.

Abstract — Searching in large, distributed, and dynamic environments for information or services can be a complex task. In this paper, we consider how agents in cooperative networks can be used to enhance the performance of algorithms that search for information. We present a new algorithm that uses cooperative agents to search for information in a distributed environment. The algorithm is designed to be efficient and scalable, making it well suited for use in large-scale search scenarios.

D. Algorithms

The algorithm presented in this paper is based on a search for information using agents that are placed in a cooperative network. The network is formed by connecting the agents together in a cooperative manner. This allows the agents to share information and work together to find the information they are searching for. The algorithm uses a tree-based search approach, which allows the agents to search for information in a structured way. This makes it easy to add new agents to the search, and it allows the agents to find the information they are looking for quickly.

E. Applications

The algorithm presented in this paper has a number of potential applications. It can be used in a variety of environments, including search engines, social networks, and online communities. It can also be used in a variety of domains, such as e-commerce, social networking, and healthcare.

F. Conclusion

In conclusion, the algorithm presented in this paper provides a powerful tool for searching information in large, distributed, and dynamic environments. It is a scalable and efficient approach that can be used to enhance the performance of algorithms that search for information. The algorithm is designed to be easy to use and can be applied to a wide range of applications. The results of the algorithm are promising, and it is hoped that it will be used in a variety of real-world scenarios.
II. PROBLEM DEFINITION

Sophisticated models will be required in the future, as a search for possible (and more) optimal solutions simplifies and modifies the objectives and constraints. The goal of the optimization process is to find a solution that is optimal in the context of the problem. This process involves identifying a set of candidate solutions and evaluating their performance. The evaluation criteria are based on a combination of factors, such as cost, efficiency, and reliability. The optimization process is iterative, and the solutions are refined until an optimal solution is found.
III. METHOD

**Figure 1:** Possible move directions for agents in all 8 orientations.

where

\[(i)^{t+1} - 1]_{(z+1)\times (z+1)} = (z + i)^t\]

and

\[(i)^{t+1} - 1]_{(z+1)\times (z+1)} = (z + i)^t\]

Then we get

\[z + i \text{ step if } z + i \text{ is not a cell} (x,y) \text{ that is a -state at time } t \text{ and it is on a path that moves to the next cell (x,y) in the direction of other actions.}

It is possible that other actions may have steps in time, but no steps at all. Thus, we have

\[\{1, 2, 3, 4\} = \{1, 2, 3, 4\} \text{ where in the continuous space}

\[\{1, 2, 3, 4\} \in \{(1, 2, 3, 4)\} \text{ the time step is zero.}

The essential step is that, at every step, the agent can move in any direction as long as it is not a -state.
As described earlier, we consider three situations:

1. **Learning Algorithm**: The learning of the neural networks is modeled to simulate the learning process of the agents. The learning process involves adjusting the weights of the neural networks to minimize the error between the predicted outcomes and the actual outcomes. The learning rate is adjusted based on the performance of the agents. This process is repeated for a fixed number of iterations or until a certain level of accuracy is achieved.

2. **Greedy Search**: This is a local-search type of algorithm where the best possible move is chosen at each step. The advantage of this method is its simplicity and efficiency in finding a good solution. However, it can get stuck in local optima and may not always find the global optimum.

3. **Centralized Learning**: In this approach, all agents share their information and make decisions collectively. This method is more complex and computationally intensive but can lead to better overall performance.

**Simulation Results**

- **Greedy Search**: This method is effective for finding quick solutions but may not always yield the optimal result.
- **Centralized Learning**: This approach is more robust and can find better solutions, albeit at a higher computational cost.
- **Comparison**: The centralized learning approach generally outperforms the greedy search, especially in complex environments or when the number of agents increases.

**Conclusion**

Centralized learning is more effective in situations where the environment is complex and the interactions between agents are crucial. Greedy search is more suitable for simple environments or when computational resources are limited. The choice between the two methods depends on the specific requirements of the application.
In this paper, we have compared the performance of several clustering algorithms to learn agent coordinates. However, it appears that OCL needs about 30 steps of learning before the time needed to reach 98% certainty. However, the Greedy Search also outperforms the LCA algorithm. This is consistent with Figure 4, which shows the OCL algorithm is more efficient than the DL algorithm. Instead of using a 20 x 20 environment, it is assumed that the Greedy algorithm is equal to the actual search path.

In addition, Figures 5 and 6 show the actual search path when PLC. The parameters for these figures are given in Figure 3.

Finally, Figures 5 and 6 show the actual search paths. The parameters for these figures are given in Figure 3.

The number of search steps needed to reach 98% certainty is shown in Figure 4. The parameters for these figures are given in Figure 3.