Task Allocation in MultiAgent Robotic Systems in a Warehouse

Francis Bermillo, Kartik Gupta, Niraj Basnet

***Abstract*—In this paper we propose a distributed approach in warehouse task allocation in which the multiagent system use an evolutionary algorithm to exploit the best policies and efficiently select tasks to complete in a warehouse simulation. We use Cooperative Co-Evolutionary Algorithm with Hall of Fame and Difference Evaluation to train the multiagent system and compare its performance with a centralized approach commonly used in most autonomous warehouses. We show that the performance of the distributed system greatly outperforms the centralized approach in allocating tasks.**

I. Introduction

E-COMMERCE industry is a time critical domain with warehouses as their backbone. Improving warehouse efficiency trickles down to the whole supply chain and logistics resulting in reduced delivery times and costs. Today, warehouses are expanding and growing larger in size leading to companies towards purchasing autonomous robots to develop an efficient system for warehouse tasks, such as sorting, packing, and delivery, to reduce overhead costs and boost efficiency.

In this domain, a task refers to the fetching of the desired object from its known source position and taking it to the target location, which is usually the packaging station. While the tasks are Single-Robot(SR) tasks, at any moment of time there are multiple tasks to be completed during the same time. The challenge that these robotic warehouses come across is optimizing multiagent robots task allocation, coordination, and path planning to create an efficient ecosystem, in which this domain is still being actively researched due to the problem being NP-hard.

In this paper, we focus on the problem of task-allocation which refers to the distribution of tasks amongst the robots across time to achieve maximum system performance, which mostly refers to the reduction in the total costs incurred. This cost can be measured in terms of time spent, distance traveled, the power consumed, etc. As the e-commerce industry is time-critical, we consider the total time spent as the cost to be minimized.

Current autonomous warehouses use centralized methods to distribute the tasks amongst the warehouse robots. An issue with having a centralized planner is the robustness of the system. Centralized systems have tight coupling due to one central node handling information for all other nodes in the system. Because of this, any error in the central node will propagate out to the child nodes causing the whole system to be in an undesired state. Thus, these types of approaches need careful implementation to prevent the system from failure. And because one central node handles all the processing of information, the implementation of this node can become very complex as one has to think of the best strategy to allocate tasks while keeping a record of each node state.

Distributed systems overcome these challenges by creating a loosely coupled architecture leading to a more robust system for multiagents. Even if one of the agents fails, other agents can still function on their own, hence promoting scalability and flexibility in large complex warehouse systems. However, distributed or decentralized approaches can produce highly sub-optimal solutions as it is not guaranteed that an optimal local solution would sum to an optimal global solution. Recent approaches in distributed systems have been widely applied to multiagent task allocation problems. One in particular, complements a distributed system with auction-based method known as a market based approach[1][2][3]. Another researched approach invokes the framework of token-based task assignment[4]. However, both these approaches are time-intensive which limits their usability[5].

Our approach is to give each agent the option to select a task that will maximize the reward of the whole system based on their current situation (i.e. the agent selects an item in an item pick list that it is closest to in order to complete the task faster). We will apply the cooperative co-evolutionary learning algorithm, specifically the Hall-of-Fame with differential rewards, to develop policies by the agents. One key drawback of this method is the high computational cost. We plan to take advantage of reinforcement learning to initialize the policies of the agents to speed up the initial stages of the coevolutionary learning algorithm.

By having each agent learn the best strategy when choosing a task, based on their locations with respect to the locations of the items and item drop off bins, our work contributes to minimizing the time to complete a collection of tasks in a warehouse setting.

II. Background

A. MultiAgent Task Allocation

Centralized controllers are currently the most popular platform for autonomous warehouse systems. These controllers command agents with their tasks. Agents are not aware of what tasks they are assigned and execute the task without any coordination with other robots. These types of centralized methods use dynamic scheduling to create an optimal and self-sufficient environment. These are usually the popular solution for smaller systems, easily decomposable tasks, and readily available global state configuration. Though these methods produce highly efficient task allocation, they are known to

fail with an increase in complexity, the number of agents, management of multiple communication channels. They lack responsiveness to changes in the environment and create bottlenecks in multiagent systems.

Amongst the distributed systems, a popular approach is the auction-based method[1][2][3], also known as a market based approach. The idea is that all the agents act independently and each agent maximizes their reward by trading tasks with other agents through negotiations. MURDOCH[6] is an approach that uses this concept and dynamically distributes the tasks using a publisher/subscriber communication method for a diverse team. The tasks are dynamically allocated using a sequence of first-price single-round auctions in a greedy fashion. Another approach that uses this concept is the TraderBots[7] architecture. In [7], self-interested agents maximize individual profit such that each agents profit results in a globally efficient result.

Token-based task assignment is another framework quite popular in task assignment. Tokens represent tasks to be executed and are exchanged between the agents such that the decision to accept token by each agent is solely based on maximizing its utility[4]. The agent acts upon certain threshold taking account of the availability of tokens, resource constraints, and the team composition. It is widely used in large-scale applications with extremely low computation and communication requirements, but cannot guarantee an optimal solution. Normally, the agents bid only on a single unallocated task in each round. But this can result in highly suboptimal allocation due to the possibility of synergies between multiple tasks. To address this, combinatorial auction allows agents to bid on bundles of tasks. However, it still poses a significant challenge of finding appropriate bundles given the large number of tasks and limited time to allocate them[5].

B. Cooperative Co-Evolutionary Learning Algorithm with Hall of Fame and Difference Evaluation

Coevolution is a natural way to evolve a group of agents that must cooperate to achieve a system objective. A recent approach to achieving coordination within a team is the use of Cooperative Coevolutionary Algorithms or CCEAs. This method involves simultaneously evolving multiple populations and evaluating individual agent fitness based on how well they interact with other agents within the system[8]. This algorithm is an approach that comes naturally to situations where the agents fail or succeed based on their teamwork. However, this is challenged with the credit assignment problem. Using a fitness function known as the Hall of Fame and Difference Rewards help overcome this difficulty[9].

The concept of the Hall of Fame saves the top individuals from each population for later generations[10]. This concept is extended to CCEAs by keeping the best agent in each population and groups them to form a "Hall of Fame." This ensures that desired genotypes are kept even if they poorly perform due to stochastic events[9].

CCEA with hall of fame can be combined with difference evaluations to optimize system performance. At the end of each generation, the team that performs the best is compared against the hall of fame members. If that team performed better than all of the hall of fame members, then that team is added to the hall of fame. The CCEA using hall of fame and difference evaluations includes shaped fitness functions to notify agents their individual contributions to the team and as well as forming a bias fitness function by estimating optimal collaborators based on the hall of fame[9].

```
Initialize N populations of k neural networks
foreach Generation do
    foreach Population do
        produce k successor solutions
        mutate successor solutions
    end
    for i = 1 → 2k · m do
        randomly select one agent from each population
        add agents to team T_i
        simulate T_i in domain
        assign fitness to each agent with Eq. 10
    end
    foreach Team T_i do
        if G(z|T_i) > G(z|HOF_best) then
            add T_i to HOF
        end
    end
    foreach Population do
        select k networks using ε-greedy
    end
end
```

Fig. 1. Pseudocode for CCEA with Hall of Fame + Difference Rewards[9]

III. METHODOLOGY

In this section we describe the model of the warehouse and the agents used for our experiments, followed by a discussion on the method used.

A. Warehouse Model

We model our warehouse as a grid world where there are p type of products that are stored at separate predefined places in the warehouse along with m fixed dropoff points distributed across the warehouse. Each item location p and dropoff location m occupies one grid in the world.

To mimic a warehouse operation, we simulate each drop off point as an order t with a fixed number of items and the quantity of each item in the list. We denote the requirement of dropoff bins i as $t_i = q_1, .., q_x, .., q_p$ where q_x denotes the required quantity of item type x at dropoff point i. Thus, the task list is represented as $T = t_1, .., t_m$.

The list of tasks T is allocated to a team of n homogeneous robots which are required to complete the collection of tasks in minimum possible time. These robots are designed as a multiagent system with each agent equipped with the knowledge of drop off bins and item locations as well as the task list T. All agents start at a fixed starting position at the center of the grid world.

At the start of an episode, each agent selects one particular task from T. These selected tasks are removed from the task list T, which is maintained by a central node and communicated to each agent when requested. After an agent completes a task, the agent can a new task from the task list and this process continues for all agents until there are no remaining tasks in the task list.

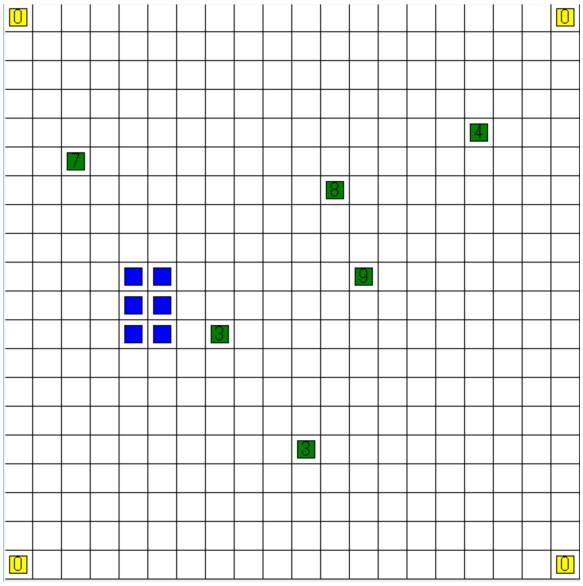

Fig. 2. A sample of the warehouse model with product types, p = 6 (in yellow); dropoff points, m = 4 (in green); number of agents = 6 (in blue) and gridworld of size 20x20

B. Learning Methodology

We utilize the CCEA algorithm with Hall of Fame + Difference Reward learning framework for our problem. We fix the task list to contain a fixed number of total tasks and randomly select a distribution of this fixed total for each simulation. Each policy in the agents' population is maintained as a neural network that takes as input the locations of the dropoff bins, the item locations, and the current updated task list. We use the algorithm found in [9] to model our algorithm.

IV. Results

In the final result, we show that the performance of the Hall of Fame + Difference Rewards give a significant boost compared to a centralized system when doing task allocation in a warehouse environment.

A. Experimental Setup

In our experiment, we created a grid world of size 1000x1000 with 6 product pickup locations and 2 dropoff points. To scale the tasks with the increasing number of agents,the total number of tasks was determined by multiplying 10 to the number of agents in the environment. The maximum time steps allowed to complete the task list was set to 50000 and evolution was limited to 2000 generations for performance evaluation. For each simulation the global reward assigned to the system was (50000 - total time taken). During co-evolutionary learning, the network mutation was accomplished by adding values drawn from a Gaussian distribution to network weights. The standard deviation with which network weights were mutated was initialized as 0.7 and finally decreased down to 0.1 towards final generations. Mutation was also carried out on 70 percent of total weights at first and later decreased to 30 percent.The selection of next generation policies was carried out using epsilon-greedy approach with epsilon initialized as 0.2, and later decayed to 0 to end exploration. All of these parameter changes were linearly varied throughout the overall span of generations. We first simulated with 10 agents then increased it to 30. Each agent had a population size of 15 policies for both the cases.

The task list was generated randomly at the start of the simulation and the same was used for entire generations. As mentioned above, we ignore path planning and collisions in this experiment and we solely focus on task allocation. We run 5 simulations of this setup to get a confidence interval of our performance.

B. Analysis

To compare our approach with the prevailing centralized approaches, we have used centralized sequential task allocation as our primary baseline to assess the performance of our approach. In sequential task allocation, a central node allocates task to robots in a sequential manner. As soon as any robot completes the task, it gets the next task in the list that is updated by the central node. This continues until all tasks from the list are exhausted.

As is seen in fig 3, our implementation of CCEA with HOF+DF outperforms the centralized approach. Our approach outperforms the centralized approach after training over only 500 and 550 generations for cases with 10 agents and 30 agents respectively. Also, in our simulations we observed that each agent learns to specialize to perform task with one sector of the warehouse, and moves to the next best sector when the task there is completed.

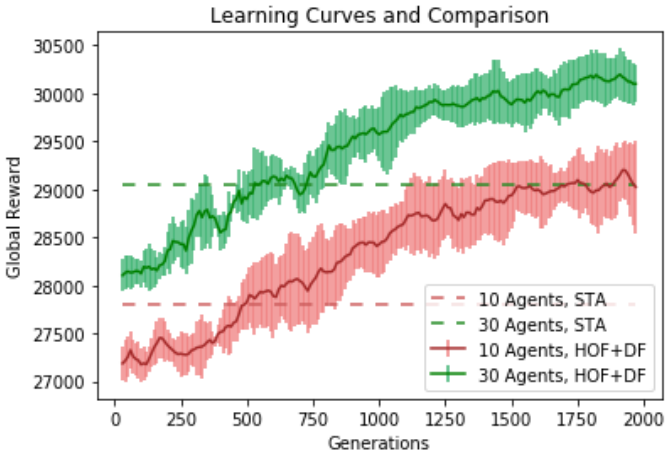

Fig. 3. The graph shows the performance learning curve for the CCEA with HOF+DF, for two cases - 30 agents and 10 agents, as per the task space explained in IV(A). The bold lines show the moving average of the global reward over 50 generations and the lighter region indicates the standard deviation observed over 5 simulations. The performance is compared with the centralized algorithm, Sequential Task Allocation(STA).

V. Conclusion

In this paper, we had shown our contribution of minimizing the time to complete a collection of tasks in a warehouse environment. Using the distributed approach, task allocation can significantly improve over current centralized methods used

in automated warehouses using Cooperative Co-Evolutionary Algorithms with Hall of Fame and Difference Evaluation. This approach helped select better policies for finding time efficient distribution of tasks among agents.

Although improving task allocation shows promise in boosting warehouse efficiency, that alone will not be enough to improve the efficiency of the whole warehouse ecosystem. In our future work, we plan to tackle another part of this ecosystem by incorporating collision free path planning with our task allocation algorithm. In addition, because warehouses are large in size, we plan to also test for scalability by simulating in larger grid worlds and higher number of items and agents.

REFERENCES

[1] Robert Zlot, Anthony Stentz, M Bernardine Dias, and Scott Thayer. Multi-robot exploration controlled by a market economy. In *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, volume 3, pages 3016–3023. IEEE, 2002.

[2] Lovekesh Vig and Julie A Adams. Coalition formation: From software agents to robots. *Journal of Intelligent and Robotic Systems*, 50(1):85–118, 2007.

[3] Esben H Ostergaard, Maja J Mataric, and Gaurav S Sukhatme. Distributed multi-robot task allocation for emergency handling. In *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, volume 2, pages 821–826. IEEE, 2001.

[4] Paul Scerri, Alessandro Farinelli, Steven Okamoto, and Milind Tambe. Allocating tasks in extreme teams. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 727–734. ACM, 2005.

[5] Marc Berhault, He Huang, Pinar Keskinocak, Sven Koenig, Wedad Elmaghraby, Paul M Griffin, and Anton J Kleywegt. Robot exploration with combinatorial auctions. In *IROS*, pages 1957–1962, 2003.

[6] Brian P Gerkey and Maja J Mataric. Sold!: Auction methods for multirobot coordination. *IEEE transactions on robotics and automation*, 18(5):758–768, 2002.

[7] M Bernardine Dias. Traderbots: A new paradigm for robust and efficient multirobot coordination in dynamic environments. *Robotics Institute*, page 153, 2004.

[8] Mitchell A Potter and Kenneth A De Jong. Evolving neural networks with collaborative species. In *Summer Computer Simulation Conference*, pages 340–345. SOCIETY FOR COMPUTER SIMULATION, ETC, 1995.

[9] Mitchell Colby and Kagan Tumer. Shaping fitness functions for coevolving cooperative multiagent systems. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems*, volume 1, pages 425–432. AAMAS, 2012.

[10] Christopher D Rosin and Richard K Belew. New methods for competitive coevolution. *Evolutionary computation*, 5(1):1–29, 1997.

VI. TEAM PERFORMANCE

Part	Francis	Kartik	Niraj
Organisation	33.3%	33.3%	33.3%
Technical Contribution	25%	50%	25%
Coding	25%	25%	50%
Writing	50%	25%	25%