Journal of Computing and Electronic Information Management
ISSN: 2413-1660 | Vol. 21, No. 2, 2026

Graph Convolutional Network-Based Multi-Agent Collaborative Task Allocation Optimization Algorithm

Hao Yang

Shanghai Lujing Education Technology Co., Ltd., Shanghai, China

Abstract: Due to the complex interdependencies between tasks, the dynamic state changes of the tasks, and the lack of flexibility of the traditional static matching method in the collaborative task allocation with multi-agent, this study aims to explore an optimization algorithm for collaborative task allocation driven by a graph convolutional network. The paper details the construction of multi-agent task graphs, the design of node features and edge weights as well as the graph convolutional feature aggregation mechanism. It also introduces updating of dynamic collaborative relationship, construction of allocation utility function and collaborative optimization process. The results of comparative and ablation experiments in a two-dimensional simulation platform show that the algorithm has a 97.4% task completion ratio and an average utility of 86.7 with an excellent allocation efficiency and convergence stability.

Keywords: Graph Convolutional Network; Multi-agent; Task Allocation; Cooperative Optimization; Utility Function.

1. Introduction

In many applications, like unmanned swarm inspection, intelligent warehouse scheduling and emergency collaborative response, multi-agent systems are required to perform complex tasks, and the task allocation performance directly affects the response speed, resource utilization, and collaborative stability of a multi-agent system (Li et al. 2021). Static cost matrices are frequently used in traditional approaches for task allocation, which are not able to cover differences among agents capabilities, tasks temporal relationships, communication states changing and multi-hop collaborative relationships. This means that there may be an unbalance and frequent retuning in the event of a sudden task or change in the node state. Agents, tasks and their relationships can be represented as a GCN (Graph Convolutional Networks). They enhance the relational representation ability for task assignment optimization by using neighborhood aggregation to learn latent collaborative features between each node (Chen et al. , 2021). Built upon this, the study proposes a multi-agent task graph (MATG) and the design of node features and a dynamic edge weight mechanism, and then uses a utility function to make collaborative assignment decisions. The study demonstrates and validates the algorithm's enhancements on assignment efficiency, task completion rates, and stability through comparative experiments and ablation studies. Ablation studies and convergence analysis validate the algorithm's improvements on assignment efficiency, task completion rates, and stability (Li et al. , 2022).

2. Graph Convolutional Network-Based Task Association Modeling Methods

2.1. Construction of Multi-Agent Task Graphs

In multi-agent collaboration, the task allocation needs to be uniformly mapped to a task graph which includes agents, tasks and collaborative relationships. Agent nodes include the information of position coordinates, remaining energy, execution capability and communication status, and task nodes include spatial location, priority, resource requirement, execution duration and deadline constraints (Fawaz et al. , 2022). A hybrid graph of execution and collaboration relations and constraint relations between agents and tasks is established according to reachability distance, capability matching, time window compatibility and inter-task dependency among agents and tasks. The graph should have weighted links, using distance decay, capability differences, and task urgency to determine the strength of the connection between agents that are involved in the execution of the task (Goeckner et al. , 2024). This modeling method offers systematic information to be inputted to a later GCN neighborhood aggregation method, and changes the task assignment from standalone matching towards relationship-based collaborative optimization. The agents, task objects and their relationships in multi-agent collaborative task assignment can be uniformly represented by using weighted graph structure, the specific construction process is shown in figure 1.

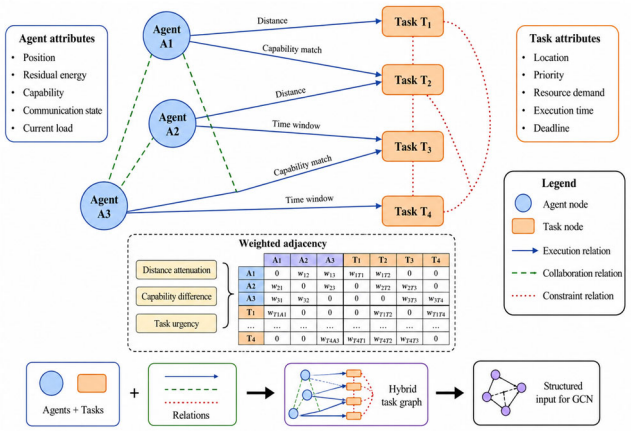

Figure 1. Schematic diagram of multi-agent task graph construction

2.2. Design of Node Features and Edge Weights

The node features should not only include agent execution states but also include task constraint information, to avoid making the status of distance or idle as the basis for

assignment (Tang et al. , 2021). On the agent side, attributes like current location, available energy, carrying capacity, execution speed, communication quality and past completion rate can be selected, and on the task side, attributes like spatial coordinates, priority, number of resources required, estimated execution time, deadline and collaboration requirements can be selected. Then these are normalized to create a single feature vector (Duan et al. , 2024). The matching strength between agents and tasks is described by means of weights on the edges. They should consider all the above in depth and come up with the following calculation formula:

$$\omega_{ij} = \alpha C_{ij} + \beta E_{ij} + \gamma T_{ij} + \delta P_j \quad (1)$$

whereω_{ij}represents the edge weight between agent$i$and task$j$;C_{ij}denotes the capability matching score;E_{ij}represents the remaining energy or resource compatibility;T_{ij}indicates time-window reachability;$P_j$denotes task priority; andα,β,γ, and$\delta$are weight coefficients used to adjust the influence of different factors on the assignment decision.

2.3 Graph Convolutional Feature Aggregation Mechanism

For the weighted task graph, the GCN should first normalize the neighbor relationships in the graph to reduce the influence of highly connected nodes on the aggregation result, and then for each node (agent node), they should obtain the information about the tasks they are connected to and the neighbor nodes they are connected to based on the weight of the edge (Wang et al. , 2023). The adjacency matrix is operated on in the following manner:

$$\hat{A} = \widetilde{D}^{-\frac{1}{2}} \tilde{A} \widetilde{D}^{-\frac{1}{2}} \quad (2)$$

where $\hat{A} = A + I$ denotes the adjacency matrix after incorporating self-connections; $\widetilde{D}$ is the degree matrix corresponding to$\tilde{A}$; and$\hat{A}$is the normalized result. Then, the node features and structural weights are jointly fed into the convolutional layer to update the node embedding representation:

$$H^{(l+1)} = \sigma\left(\hat{A} H^{(l)} W^{(l)}\right) \quad (3)$$

where$H^{(l)}$denotes the node features of the$l$layer,$W^{(l)}$is the trainable weight matrix, and σ is the nonlinear activation function. After multi-layer propagation, the node representations can simultaneously incorporate both local matching relationships and multi-hop collaborative dependencies.

3. Design of a GCN-Based Cooperative Task Assignment Optimization Algorithm

3.1. Dynamic Update of Collaborative Relationships

The agent's location, energy, payload and communication links all dynamically change during the execution of the task. It can simply trigger the repeated task distribution to heavily loaded nodes, the job distribution to nodes far away, and the lack of timely job distribution of burst tasks in the case that the original task graph is static (Rokhforoz et al. , 2023). Changes to collaborative relationships should be based on feedback in execution. Edge weights between agents and tasks should be updated when a task is finished, postponed, the energy of an agent is below a threshold or the quality of communication is degraded (Zhang et al. , 2025). New edges should be created for new tasks, based on spatial distance, capability requirement and time window; and the edges should be weakened or removed when the tasks have failed or the agents are unreachable. The graph structure is then updated and re-input to the GCN, so that node embeddings can capture the current state of collaboration, and avoid being stuck on the initial matching result (Hong et al. , 2024).

3.2. Construction of the Assignment Utility Function

The utility function must convert the node embeddings output by the GCN into comparable allocation scores, incorporating task utility, execution costs, and collaboration compatibility into the optimization objective (Gao et al., 2022). The utility of a single agent-task match can be expressed as:

$$s_{ij} = \lambda_1 q_j + \lambda_2 h_i^T h_j - \lambda_3 d_{ij} - \lambda_4 e_{ij} \quad (4)$$

where$s_{ij}$is the benefit score for agent$i$executing task$j$;$q_j$is the task priority benefit; h_i and h_j are the agent and task embeddings extracted by the GCN; d_{ij} is the distance cost;$e_{ij}$is the energy consumption cost; and$\lambda_1$through$\lambda_4$are adjustment coefficients. The overall assignment objective is further expressed as:

$$max\ R = \sum_{i=1}^{m} \sum_{j=1}^{n} x_{ij} s_{ij} - \eta B \quad (5)$$

where$R$is the total reward;$x_{ij} \in \{0,1\}$indicates whether a task is assigned to an agent;$m$and$n$denote the number of agents and tasks, respectively; BBB is the load imbalance penalty term; andη is the penalty coefficient (Mo et al., 2022).

3.3. Design of the Collaborative Optimization Process

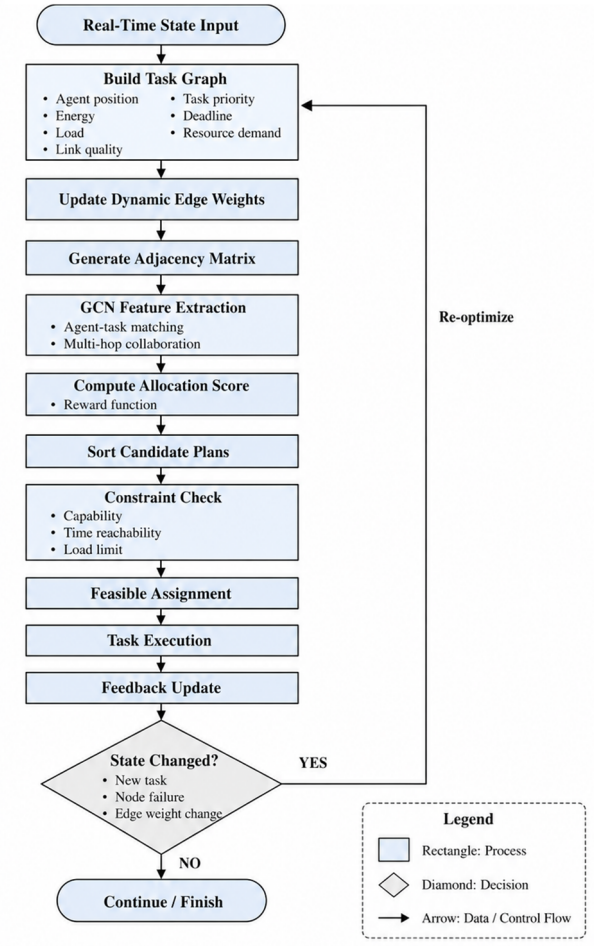

Figure 2. Flowchart of GCN-based Collaborative Task Assignment Optimization

Collaborative optimization process starts with real-time state acquisition, while writing down agent location, energy, payload and communication quality, task priority, task deadlines and resource requirements into the task graph. It then creates a current adjacency matrix, with dynamic edge weights. The graph structure is fed into the GCN, where node embeddings capture relationships between agents and tasks that are matched and between different tasks that collaborate via multi-hop paths, and a utility function is used to compute the scores for candidate assignments (Galliera et al. , 2024). In the assignment phase, constraint verification is done in the order of usefulness, discarding solutions that cannot be given sufficient capacity, time-infeasible routes, and payloads that are too large, and keeping task combinations that meet collaboration conditions. Feedback is received while performing the task and when new nodes are added or nodes fail or their weight changes substantially, the graph is updated, and the utility values recalculated, so that the results of the assignment can be adjusted iteratively based on the state of the environment (Ahmed et al. 2024). The joint optimization process of " " allows for the dynamically adjusting task distribution by state based input, graph structure modification, GCN features extraction, benefits ranking, constraints check and feedback re-optimization process. This is shown in the specific process in Figure 2.

4. Experimental Results and Analysis

4.1. Experimental Setup

All experiments were performed using an Intel i7-12700F CPU, 32 GB RAM, and NVIDIA RTX 3060 GPU in a Python 3.10 and PyTorch 2.1 environment. The simulation area was defined as a two-dimensional task space (1000 m × 1000 m), having 20 agents and 80 task nodes, and with a communication radius of 180 m. Two graph convolutions, the hidden dimension of 64, the learning rate of 0.001, 300 training epochs, and a batch size of 32 were used for the GCN model. The initial position of agents and tasks were randomly generated from a uniform distribution. The initial energy was sampled as 80-120 normalized energy units, the task priorities were sampled between 1 and 5, and the resource requirements, execution times and deadlines were sampled hierarchically based on task priority. Two types of costs, distance and energy, are used as distance costs are calculated using Euclidean distance and energy costs are calculated according to the travel distance, task load and execution time. Time window overlaps, capacity shortages, communication unreachability and load overages cause task conflicts. The comparison algorithms consist of Hungarian algorithm, auction algorithm and genetic algorithm. All experiments in each set are repeated 30 times and averaged to get the results as a single data source for later efficiency comparison, ablation, and convergence performance evaluation.

4.2. Comparison of Allocation Efficiency

In order to verify the efficiency improvement of the GCN-based collaborative algorithm for multiple agents task allocation, the Hungarian algorithm and auction algorithm, GCN algorithm and the genetic algorithm were compared in the same task scale, the same number of agents and the same constraints. Average allocation time, task completion rate, average completion time and total path cost were used as evaluation metrics and the results are presented in Table 1.

Table 1. Comparison of Efficiency Among Different Task Assignment Algorithms

Algorithm	Average Allocation Time/ms	Task Completion Rate/%	Average Completion Time/min	Total Path Cost / m
Hungarian Algorithm	34.7	89.6	42.8	13,246
Auction Algorithm	28.5	91.3	40.6	12,682
Genetic Algorithm	56.9	94.1	38.4	11,875
GCN Collaborative Algorithm	21.8	97.4	35.2	10,893

The average allocation time of the GCN collaborative algorithm is 21.8ms, as shown in Table 1, which is less than the Hungarian algorithm 34.7ms, auction algorithm 28.5ms, genetic algorithm 56.9ms. This suggests that graph convolutional feature aggregation can help to lower the number of unnecessary matching searches. For the completion rate of tasks, the proposed algorithm performs at 97.4% which is 3.3% and 7.8% higher than the genetic algorithm and the Hungarian algorithm, respectively. In addition to improving the matching efficiency, the algorithm also optimizes the cost of the task execution path and the overall completion time, which are reduced from 13,246 m in the Hungarian algorithm to 10, 893 m, and from 36.6 min to 35.2 min.

4.3. Analysis of Ablation Experiment Results

Table 2. Results of Ablation Experiments for the GCN Collaborative Algorithm

Experimental Model	Task Completion Rate/%	Average Payoff	Number of Conflicts/times	Convergence Iterations
Complete GCN Collaborative Algorithm	97.4	86.7	3.1	138
Edge Weight Removal Mechanism	93.8	79.5	6.4	176
Remove dynamic updates	92.6	77.9	7.2	191
Remove collaboration revenue items	94.3	81.2	5.8	165
MLP Replaces GCN	90.7	74.6	8.9	214

To better understand the respective influence of each of the above components on the performance of the algorithm, we replaced the edge weight mechanism, the dynamic update mechanism, and the collaboration benefit term of the overall GCN collaborative algorithm one by one, respectively, and used an MLP as a control. Experiments were contrasted with respect to four factors: completion rate of the task, average benefit value, number of conflicts and number of convergence

iterations. The results obtained are presented in Table 2.

The complete GCN collaboration algorithm outperformed the other algorithms in terms of task completion rate as shown in Table 2, with 97.4% task completion rate, average benefit value of 86.7, only 3.1 conflicts and converged in 138 iterations. Task completion rate decreased to 92.6% and the number of convergence iterations increased to 191 after removing dynamic updates, which showed that static graph structures are not able to update quickly enough to keep up with changes in task states. The task completion rate is further reduced to 90.7%, while the number of conflicts increased to 8.9, when GCN is removed and MLP is used, demonstrating the capability of graph neighborhood aggregation is strongly diminished when it is removed.

4.4. Analysis of Stable Convergence Performance

This paper further plots the convergence curves of the return of the different algorithms on the Y axis and the number of iterations on the X axis with average return of the algorithms on the y-axis to further evaluate the stability of the different task allocation algorithms in the iteration, and the results are shown in figure 3. Figure 3 illustrates how the GCN collaborative algorithm quickly went from 55.2 to 83.1 returns in the first 100 rounds to 86.2 returns at round 150. It then oscillated between 86.6 and 86.8, eventually settling at 86.7 in round 300 showing a rapid convergence rate, and good stability in the later rounds. By contrast, the genetic algorithm reached a utility of 80.9 after 300 rounds, 5.8 points less than the utility of the GCN collaborative algorithm; the auction algorithm and the Hungarian algorithm finally reached utilities of 75.6 and 72.0, respectively, with a greater difference compared to the algorithm proposed in this paper. Moreover, after 150 rounds, the GCN curve almost reached a plateau, whereas other algorithms still had a steady increasing trend. This means that graph convolutional aggregation is able to learn and capture the agent-task matching relationship earlier and, based on the dynamic weight updating of the edges, alleviate the volatility of allocation and enhance the convergence quality of collaborative task allocation.

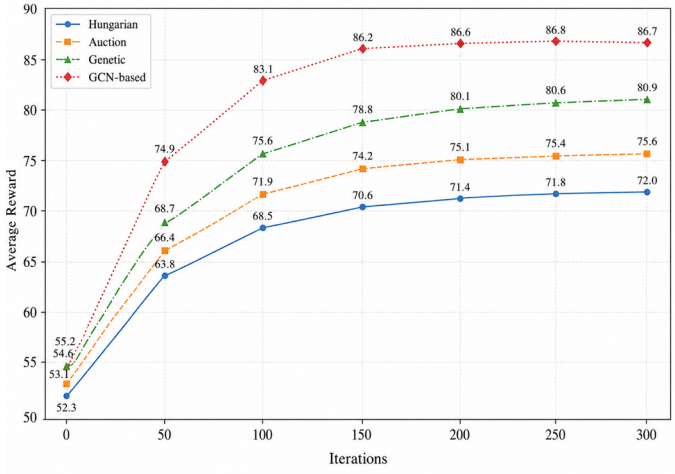

Figure 3. Comparison of convergence curves for different algorithms

5. Conclusion

This multi-agent collaborative task allocation approach via graph convolutional networks integrates agent states, task constraints and collaborative dependencies into a single graph structure. It makes use of node characteristics and dynamic edge weights, and aggregates neighbor's characteristics to better capture the characteristics of complex tasks relationships. Execution efficiency and collaborative stability are obtained with the inclusion of factors like priority, distance, energy consumption, and load balancing in the utility function. The experimental results demonstrate that the GCN based collaborative algorithm can accomplish tasks with 97.4% completion rate, and its final average utility settles at 86.7, which surpasses traditional matching algorithm and heuristic algorithm. In the future, one could imagine extending the algorithm to communication-constrained environments and sudden changes in task reallocation, by adding reinforcement learning and online transfer mechanisms to improve the algorithm's adaptability in real-world dynamic systems and make it more useful for heterogeneous agents.

References

[1] Li, Y., Ren, S., Wu, P., Chen, S., Feng, C., & Zhang, Q. (2021). Learning distilled collaboration graph for multi-agent perception. Advances in Neural Information Processing Systems, 34, 29541-29552.

[2] Chen, S., Dong, J., Ha, P., Li, Y., & Labi, S. (2021). Graph neural network and reinforcement learning for multi-agent cooperative control of connected autonomous vehicles. Computer-Aided Civil and Infrastructure Engineering, 36(7), 838-857. https://doi.org/10.1111/mice.12673

[3] Li, M., Chen, S., Shen, Y., Liu, G., Zhou, X., & Wu, X. (2022). Online multi-agent forecasting with interpretable collaborative graph neural networks. IEEE Transactions on Neural Networks and Learning Systems, 35(4), 4768-4782.

https://doi.org/10.1109/TNNLS.2022.3187316

[4] Fawaz, H., Lesca, J., Quang, P. T. A., & Tran, P. (2022). Graph convolutional reinforcement learning for collaborative queuing agents. IEEE Transactions on Network and Service Management, 20(2), 1363-1377.

https://doi.org/10.1109/TNSM.2022.3225981

[5] Goeckner, A., Sui, Y., Martinet, N., Fang, Z., Zhang, Y., & Duan, W. (2024). Graph neural network-based multi-agent reinforcement learning for resilient distributed coordination of multi-robot systems. In Proceedings of the 2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (pp. 5732-5739). IEEE.

[6] Tang, B., Zhong, Y., Neumann, U., Wang, G., Chen, Y., & Zhang, L. (2021). Collaborative uncertainty in multi-agent trajectory forecasting. Advances in Neural Information Processing Systems, 34, 6328-6340.

[7] Duan, W., Lu, J., & Xuan, J. (2024). Group-aware coordination graph for multi-agent reinforcement learning. arXiv, arXiv:2404.10976. https://doi.org/10.48550/arXiv.2404.10976

[8] Wang, Y., Wu, J., Hua, X., Wang, C., & Zhang, Y. (2023). Air-ground spatial crowdsourcing with UAV carriers via geometric graph convolutional multi-agent deep reinforcement learning. In Proceedings of the 2023 IEEE 39th International Conference on Data Engineering (ICDE) (pp. 1790-1802). IEEE. https://doi.org/10.1109/ICDE55515.2023.00148

[9] Rokhforoz, P., Montazeri, M., & Fink, O. (2023). Multi-agent reinforcement learning with graph convolutional neural networks for optimal bidding strategies of generation units in electricity markets. Expert Systems with Applications, 225, 120010. https://doi.org/10.1016/j.eswa.2023.120010

[10] Bocheng, Z., Huo, M., Li, Z., Wang, X., & Liu, Y. (2025). Graph-based multi-agent reinforcement learning for

collaborative search and tracking of multiple UAVs. Chinese Journal of Aeronautics, 38(3), 103214.

https://doi.org/10.1016/j.cja.2024.08.011

[11] Hong, S., Liu, Y., Li, Z., Wang, T., & Chen, Q. (2024). Multi-agent collaborative perception via motion-aware robust communication network. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 15301-15310). IEEE.

[12] Gao, X., Li, X., Liu, Q., Wang, H., Chen, Y., & Zhang, Y. (2022). Multi-agent decision-making modes in uncertain interactive traffic scenarios via graph convolution-based deep reinforcement learning. Sensors, 22(12), 4586. https://doi.org/10.3390/s22124586

[13] Mo, X., Huang, Z., Xing, Y., Liu, W., & Lv, Y. (2022). Multi-agent trajectory prediction with heterogeneous edge-enhanced graph attention network. IEEE Transactions on Intelligent Transportation Systems, 23(7), 9554-9567.

https://doi.org/10.1109/TITS.2021.3121249

[14] Galliera, R., Venable, K. B., Bassani, M., Piro, G., & Boggia, G. (2024). Collaborative information dissemination with graph-based multi-agent reinforcement learning. In International Conference on Algorithmic Decision Theory (pp. 160-173). Springer Nature Switzerland.

[15] Ahmed, A. N., Mercelis, S., & Anwar, A. (2024). CollabGAT: Collaborative perception using graph attention network. IEEE Access, 12, 142380-142393.

https://doi.org/10.1109/ACCESS.2024.3412803