Scholar Publishing Group
Distributed Processing System
https://doi.org/10.38007/DPS.2020.010106
ISSN 2790-0916 Vol. 1, Issue 1: 42-50

Task Allocation Mechanism Based on Ant Colony Algorithm in Distributed System

Sahill Kavitar[*]

Case Western Reserve University, USA

[*]*corresponding author*

Keywords: Ant Colony Algorithm, Task Allocation, Distributed System, Docker Container

Abstract: With the popularization and rapid development of the Internet, software applications have higher and higher requirements on concurrency and service quality, which promotes the continuous evolution of the Internet architecture. The rapidly increasing user scale and increasingly complex service system lead to the explosive growth of concurrent access traffic in the network. This paper mainly studies the application of task allocation mechanism based on ACA(ACA) in distributed system. In this paper, a task allocation system is constructed based on Master/Slave architecture, and a task allocation mechanism based on ACA is designed and implemented. By optimizing the resource allocation of workflow, the completion time of workflow is minimized. The simulation results show that ACA can improve the efficiency of task allocation.

1. Introduction

With the rapid development of computer and information technology and its deep integration with traditional technologies of various industries, a large number of complex systems characterized by distributed, open and intelligent have emerged. The distributed intelligent systems with the characteristics of distribution, connectivity, collaboration, openness, fault tolerance, independence and so on. DIS has attracted more and more attention from academia and business circles, and has become a new research hotspot [1-2]. At present, distributed Artificial Intelligence (DAI) provides an effective way for DIS implementation, and Multi-Agent systems (MAS) technology is one of the two important research branches in the field of DAI. It provides technical support for large-scale DIS analysis, design and implementation, and is widely used in many fields of DIS. Cooperation mode is an important part of multi-task cooperation mechanism. In view of the fact that a member of DIS with limited ability or resources has to negotiate and cooperate with other members of the system to form a cooperative alliance to complete the task that cannot be completed by itself [3-4]. Through collaboration alliance, tasks that cannot be completed by a single member can be

Copyright: © 2020 by the authors. This is an Open Access article distributed under the Creative Commons Attribution License (CC BY 4.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited (https://creativecommons.org/licenses/by/4.0/).

completed, and resource allocation can be adjusted to complete the established tasks with optimal resource allocation and efficiency, and maximum benefits can be obtained. Therefore, alliance is the main way to achieve multi-task cooperation in DIS. Therefore, in order to successfully realize multi-task cooperation in DIS, it is very necessary to establish corresponding cooperative alliance mechanism (including resource allocation mechanism, trust mechanism, decision evaluation mechanism, utility division mechanism, etc.) [5].

At present, the traditional optimization algorithms (such as analytical method and numerical analysis method) are often faced with the situation that the solution time is long, the solution accuracy is low, and even the solution cannot be solved when solving the multi-task resource allocation problem. The intelligent optimization algorithm overcomes the limitation of the traditional optimization algorithm in solving the problem with high condition requirements, and the search process of the algorithm does not depend on the specific information of the search problem, and has the characteristics of low computational complexity, which has been highly concerned by researchers at home and abroad [6]. For example, particle swarm Optimization (PSO) has simple coding, no mutation, crossover and other operations in genetic algorithm, low algorithm complexity, easy implementation, and need to adjust fewer parameters, so it has been widely used in many fields [7]. However, in the process of optimization and solution, PSO algorithm has a fast convergence rate in the early stage, and is prone to fall into local optimality in the later stage, which is not well solved [8].

In this paper, the task allocation mechanism of ACA is introduced into the task allocation of distributed system, which provides theoretical guidance for the research on the mechanism issues of resource allocation, decision evaluation and utility division of multi-task cooperation in complex DIS. It has important theoretical significance and application value.

2. Task Allocation Modules and Algorithms in Distributed Systems

2.1. Task Distribution System

In this paper, the whole system is divided into the following modules: Clients that accept user requests, RoseDeployer that deployer and manages containers based on scheduling solutions, Scheduler, Docker, and Resource Pool that has all Resource information of the service cluster.

This part adopts the Master/Slave architecture, RoseDeployer takes the Master and RoseHost takes the Slave. RoseDeployer is responsible for the management of all rosehosts in the cluster. RoseHost is responsible for the management of various resources on its host nodes, container start and stop, resource usage collection, etc. [9-10].

(1) RoseDeployer module

RoseDeployer is the core of the entire resource scheduling system. RoseDeployer manages all tasks and is responsible for the management of all rosehosts in the cluster, as well as the management of Dockers on Rosehosts. As shown in Figure 1, RoseDeployer consists of three parts: TaskManager, DockerManager and HostManager, which are responsible for managing sub-modules of different dimensions. The TaskManager accepts and manages tasks submitted by users, and deploys and schedules tasks accordingly. The DockerManager is responsible for managing all containers for the cluster, including container start, stop, and migration. HostManager manages all RoseHost hosts in the cluster [11-12]. RoseDeployer allocates resources to user tasks based on user requests and certain resource allocation and scheduling algorithms.

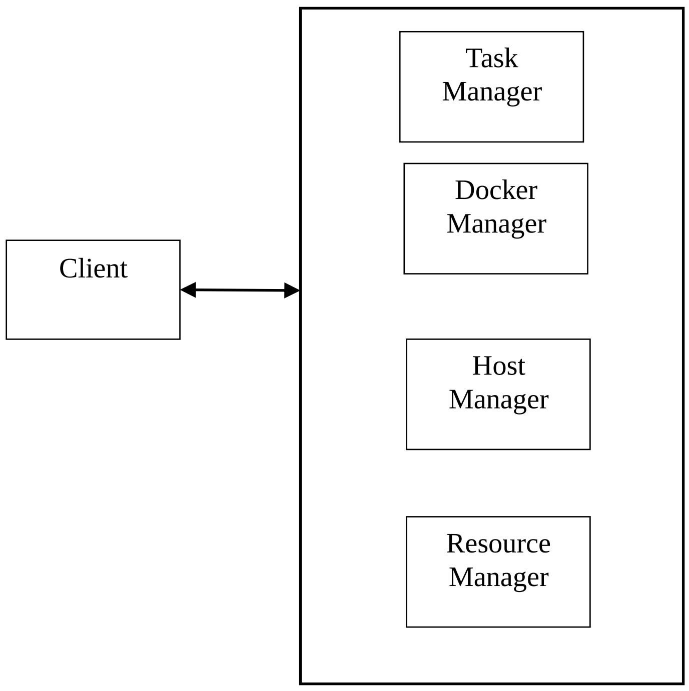

Figure 1. RoseDeployer module

RoseDeployer collects all the information of RoseHost in the cluster, including the total resource amount of RoseHost, used resource amount, remaining resource amount, and resource amount occupied by each Task [13-14].

RoseDeployer is responsible for:

Receives the scheduling scheme generated by the scheduler.

Execute corresponding tasks based on the scheduling scheme and deploy them to the corresponding RoseHost and Docker.

Maintain the resource pool and update data in the resource pool in a timely manner.

Manage RoseHost and containers of RoseHost in a cluster. Managing RoseHost includes maintaining the heartbeat of RoseHost and collecting the status of RoseHost. Managing containers of RoseHost includes starting, stopping, and migrating containers.

(2) RoseHost module

RoseHost manages a single node. RoseHost has the following responsibilities: RoseHost directly manages the resource allocation of this host, starting and stopping containers, and collecting resource usage. RoseHost manages containers on the host through Decker Daemon, including mirror pull, container start and stop, resource allocation, and resource isolation.

RoseHost consists of two parts, DockerManager and RoseHostInfo. The DockerManager accepts the commands from RoseDeployer and manages the container directly through the Docker Daemon on the host. RoseHostInfo manages the parts that are relevant to the host itself, such as the collection of resource usage. Figure 2 shows the RoseHost module diagram.

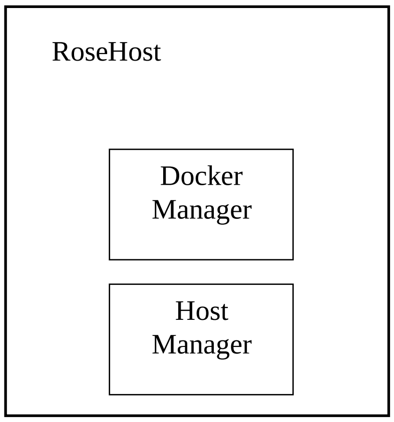

Figure 2. RoseHost module

2.2. ACA for Task Problems

The ant colony optimization algorithm designed in this paper consists of five key parts :(1) definition of heuristic function; (2) Definition of pheromone; (3) state transition probability; (4) The definition of pheromone update rules; (5) Local search strategy. In the design process of ant colony optimization algorithm, we call the assignment of task I to physical machine j a quest, which is represented by the symbol S(I, j), and on this basis, we discuss the design of the above five parts in detail.

(1) Definition of heuristic function

In this paper, a matrix η is used to store the value of the heuristic function. Each column of the matrix corresponds to a different task, and each row corresponds to a different physical machine. The variable ηij in the matrix η, <s:3> represents the elicitation function value of each inquiry S(I,j).</s:3> The value of ηij can provide an important basis for the state transition of ants. The larger the value of the heuristic function on a search, the better the search is, and the greater the probability of ants choosing this search. In the process of ant solving, the value of heuristic function can be divided into static calculation and dynamic calculation according to different requirements. Static calculation is to evaluate the value according to specific factors, and only need to calculate once. Static calculation can reduce the operation time of the algorithm, but it is difficult to find the optimal solution when dealing with the large-scale task allocation problem. Dynamic calculation of the solution can be got by has to take the next search heuristic function value, it needs to be in the process of solving each find inspiration function to calculate, so compared with the static calculation, the dynamic calculation of operation for a long time, but the dynamic calculation in solving large scale task allocation problem, High-quality solutions can still be found [15-16]. Therefore, this paper will use dynamic calculation to find out the heuristic function value of each quest.

In the task allocation problem studied in this paper, a heuristic for searching S(I, j) is directly related to the execution time and the earliest and latest end time of the task I on physical machine J. In traditional ACA, due to the positive feedback of ant colony, the heuristic information of an ant choosing a route is inversely proportional to the length of the route, that is, the longer the route, the smaller the heuristic information. Therefore, in this paper, the latest end time UETij of task I on physical machine j is calculated first when calculating the heuristic function value. And then find out all executable task I physical machine, the task I of the end time of the latest maximum Max ◌ finally, according to the absolute value of the difference and explore the S (I, j) heuristic function value, computation formula is as follows:

$$\eta_{ij} = \max_{s\in\varepsilon} UET_{i,s} - UET_{i,j} + 1 \qquad (1)$$

(2) Pheromone definition

After defining the heuristic function, it is necessary to define a suitable pheromone for ACA. In ACA, the pheromone value and the heuristic function value together constitute the solution construction process. In the process of ACA, pheromone and heuristic function jointly determine the selection probability of an ant. In the process of solving ACA, the pheromone value is not static, but a dynamic global variable, which directly reflects the size of the empirical information retained in each exploration in the iterative process, and also reflects the learning ability of ants [17]. In this case, pheromones are indicators of an ant's tendency to choose a quest, and how good the quest is. In this study, a matrix τ is used to store the pheromone values, whose columns represent each task and rows represent each physical machine. The value τij of the matrix represents the pheromone value of searching S(I,j).

(3) State transition probability

Each ant starts the process of building a solution with a set of all the tasks to be assigned and a list of candidate physical machines. It then assigns each task one by one to the most appropriate physical machine at the moment. The process of assigning tasks to ants is carried out by a state transition rule. This probability rule reflects how likely a quest is to be chosen by the ant. The state transition rules of ants are mainly determined by two factors, which are the value of pheromone on S(I,j) and the value of heuristic information on S(I,j). In the current study, there are various methods to combine these two factors to solve the state transition probability value. The state transition probability of an ant K choosing to assign task j to physical machine I is:

$$p_{i,j}^{k} = \begin{cases} \dfrac{\tau_{i,j}^{\alpha}\cdot\eta_{i,j}^{\beta}}{\sum_{s\in\varepsilon}(\tau_{i,j}^{\alpha}\times\eta_{i,j}^{\beta})} & i\in\Omega_b(j) \\ 0 \quad otherwise & \end{cases} \qquad (2)$$

In the equation, ωb (j) represents the set of tasks suitable for assignment to physical machine j.

(4) pheromone update rules

In the process of ant searching for the optimal solution, as the concentration of pheromone on the existing solution increases gradually, the influence of heuristic information in the state transition formula will be less and less, which will reduce the accuracy of obtaining the optimal solution. Therefore, in the process of ant colony solution construction, pheromones in the global optimal solution and the optimal solution of the current iteration need to be updated in order to avoid the algorithm falling into the local optimal solution due to too fast convergence speed. These characteristics are an embodiment of the learning function of ACA [18]. In ACA, ρ is generally used to reflect the rate of pheromone volatilization, the size of the pheromone volatilization rate ρ directly affects the convergence and global search ability of ACA. When the value of ρ is not 0, because of the volatile effect of pheromone, the pheromone on the solution which has not been traversed will approach to 0 in the large-scale task processing, which leads to the reduction of the global search ability of the algorithm. When the value of ρ is too large, the possibility of the ant choosing the traversed solution again will increase, which will also lead to the reduction of the global search ability of the algorithm. When the value of ρ decreases, the global search ability and randomness of the algorithm will be improved correspondingly, but the convergence speed of ants seeking the optimal solution will also be reduced.

After each iteration, in order to make the pheromone in the existing solution not have too much influence on the future solution, the following formula is used in this paper to volatilize the pheromone:

$$\tau_{i,j} = (1-\rho)\cdot\tau_{i,j} \qquad (3)$$

(5) Local search

A large number of studies on meta-heuristic algorithms show that the quality of solutions can be effectively improved by combining local search algorithms in the process of solving. Many experiments on ACA also prove this point, adding local search strategy to ACA can also significantly improve the quality of solutions. Therefore, this paper uses a hill-climbing algorithm as a local search strategy to improve the quality of the solution obtained at each iteration.

The hill-climbing algorithm is a cycle of continuous progress towards a better solution according to the neighboring peaks. When it reaches a highest peak, it terminates. At this time, there is no better solution than the "peak value" nearby. The algorithm is executed by starting at the current peak and comparing it to nearby peaks. If the compared peak is larger, it will be regarded as the highest peak, so as to reach the highest peak. It goes through several cycles until it finds the highest point. The hill-climbing algorithm can effectively avoid global traversal, and use heuristic information to select some nodes, which improves the overall efficiency of the algorithm.

3. Experimental Simulation

The simulation experiment designed in this paper is mainly to verify the feasibility of the negotiation model and algorithm in the task planning process of the proposed multi-agent system. The simulation computing platform is a common desktop computer, and multithread programming method is used to simulate the operation of the distributed system. The system configuration is shown in Table 1. The CPU is Intel Core I5-10400, the memory is 8G, the operating system is Windows10, and the planning program is implemented by Java language program. In order to verify the effectiveness of our experimental model and algorithm, we introduce the credit mechanism and task allocation strategy into the model, and divide the experiment into two parts. One part is to reflect the effectiveness of the model by comparing the time to complete the task with the increase of distributed system when the number of tasks is the same. The other part is to reflect the effectiveness of the model by comparing the time to complete tasks when the distributed system is the same and the number of tasks increases.

Table 1. System configuration information

Type	Configuration
CPU	Intel Core i5-10400
Memory	8G
Operating system	Windows10
Development of language	Java

4. Analysis of Experimental Simulation Results

In Experiment 1, the number of tasks was 500. As the number of agents changed, the comparison results of ACA and particle swarm optimization algorithm were shown in FIG. 3. It can be seen from the figure that when the number of tasks is unchanged, with the increase of the number of agents, the ant colony proposed in this paper takes less time to complete the task than the traditional particle swarm optimization algorithm, and the effect is more obvious with the increase of the number of distributed systems.

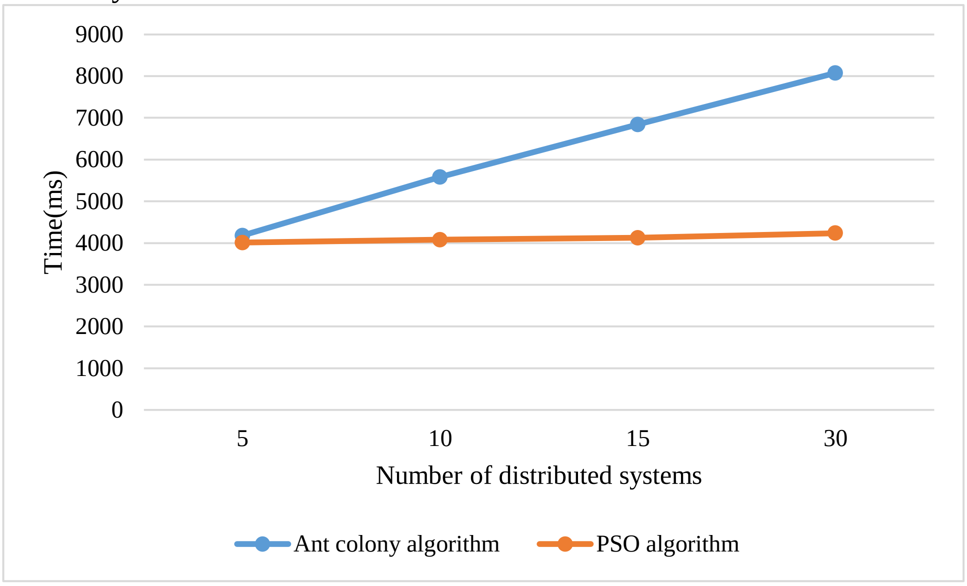

Figure 3. The distributed system counts the different running time comparison

In Experiment 2, all experimental environments and equipment are the same as in Experiment 1, and the number of distributed systems is set as 30. With the change of the number of tasks, the comparison results between ACA and particle swarm optimization are shown in FIG. 4. It can be seen from the figure that when the number of distributed systems is unchanged, with the increase of the number of tasks, the ACA proposed in this paper takes less time to complete the task than the traditional particle swarm optimization algorithm, and with the increase of the number of tasks, the effect is more obvious. This shows that the approach presented in this chapter is more suitable for large-scale task set environments.

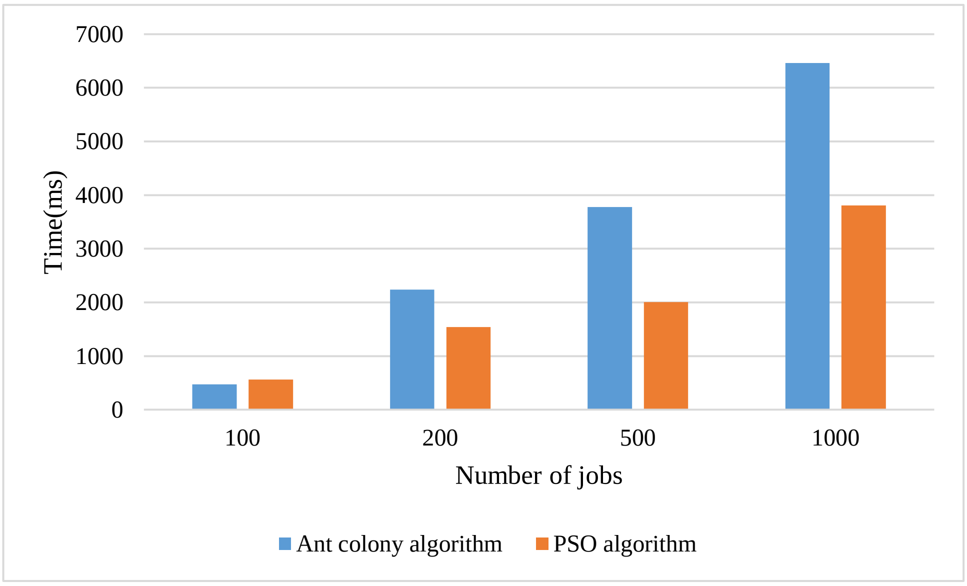

Figure 4. Comparison of different running times of task count

5. Conclusion

In this paper, a distributed system resource scheduling management system based on Docker container is designed and implemented. Through this system, users can realize efficient management of cluster and container applications. An ACA is designed and implemented to obtain the approximate optimal solution. Aiming at the characteristics of order and dependence among tasks in the workflow in this study, a heuristic information based on Ultimate End Time (UET) of tasks is designed for the ACA used. In addition, in order to prevent the algorithm from falling into the local optimal solution, a local search algorithm is designed to improve the quality of the current solution combined with the hill-climbing algorithm. Finally, the experiment proves that ACA can effectively improve the task allocation efficiency of workflow.

Funding

This article is not supported by any foundation.

Data Availability

Data sharing is not applicable to this article as no new data were created or analysed in this study.

Conflict of Interest

The author states that this article has no conflict of interest.

References

[1] Tsvetomira R, Anna D, Nancy L, et al. Costs of task allocation with local feedback: Effects of colony size and extra workers in social insects and other multi-agent systems. PLoS Computational Biology, 2017, 13(12):e1005904. https://doi.org/10.1371/journal.pcbi.1005904

[2] Rashidi S, Sharifian S. A hybrid heuristic queue based algorithm for task assignment in mobile cloud. Future Generation Computer Systems, 2017, 68(mar.):331-345. https://doi.org/10.1016/j.future.2016.10.014

[3] Samriya J K, Patel S C, Khurana M, et al. Intelligent SLA-Aware VM Allocation and Energy Minimization Approach with EPO Algorithm for Cloud Computing Environment. Mathematical Problems in Engineering, 2020,(6):1-13.

[4] Fernandez E, Gomez-Santillan C, Cruz-Reyes L, et al. Design and Solution of a Surrogate Model for Portfolio Optimization Based on Project Ranking. Scientific Programming, 2017, 2017(PT.2):1-10. https://doi.org/10.1155/2017/1083062

[5] Selvakumar A, Gunasekaran G. A Novel Approach of Load Balancing and Task Scheduling Using Ant Colony Optimization Algorithm. International Journal of Software Innovation, 2019, 7(2):9-20. https://doi.org/10.4018/IJSI.2019040102

[6] Singh H, Bhasin A, Kaveri P R. QoS based Efficient Resource Allocation and Scheduling in Cloud Computing. International journal of technology and human interaction, 2019, 15(4):13-29. https://doi.org/10.4018/IJTHI.2019100102

[7] Zitouni F, Harous S, Maamri R. A Distributed Approach to the Multi-Robot Task Allocation Problem Using the Consensus-Based Bundle Algorithm and Ant Colony System. IEEE Access,

2020, PP(99):1-1.

[8] Prongnuch S, Sitjongsataporn S, Wiangtong T. A Heuristic Approach for Scheduling in Heterogeneous Distributed Embedded Systems. International Journal of Intelligent Engineering and Systems, 2019, 13(1):135-145.

[9] Lemos M, Rabelo R, Mendes D, et al. An approach for provisioning virtual sensors in sensor clouds. International Journal of Network Management, 2019, 29(2):e2062. https://doi.org/10.1002/nem.2062

[10] Alhaqbani A, Kurdi H, Youcef-Toumi K. Fish-Inspired Task Allocation Algorithm for Multiple Unmanned Aerial Vehicles in Search and Rescue Missions. Remote Sensing, 2020, 13(1):27. https://doi.org/10.3390/rs13010027

[11] Josilo S, Dan G. Decentralized Algorithm for Randomized Task Allocation in Fog Computing Systems. IEEE/ACM Transactions on Networking, 2019, 27(1):85-97.

[12] Mishra S K, Puthal D, Sahoo B, et al. An adaptive task allocation technique for green cloud computing. The Journal of Supercomputing, 2018, 74(1):370-385.

[13] Shameer A P, Subhajini A C. Quality of Service Aware Resource Allocation Using Hybrid Opposition-Based Learning-Artificial Bee Colony Algorithm. Journal of Computational and Theoretical Nanoscience, 2019, 16(2):588-594. https://doi.org/10.1166/jctn.2019.7775

[14] Harrath Y, Bahlool R. Multi-Objective Genetic Algorithm for Tasks Allocation in Cloud Computing. International journal of cloud applications and computing, 2019, 9(3):37-57.

[15] Redishettywar K, Thekiya R J. An Enhanced Task Allocation Strategy In Cloud Environment. International Journal Of Computers & Technology, 2017, 16(6):6953-6961. https://doi.org/10.24297/ijct.v16i6.6304

[16] Nam C, Shell D A. Robots in the Huddle: Upfront Computation to Reduce Global Communication at Run Time in Multirobot Task Allocation. IEEE Transactions on Robotics, 2019, PP(99):1-17. https://doi.org/10.1109/TRO.2019.2937468

[17] Gupta P, Ghrera S P. Fault tolerant big bang-big crunch for task allocation in cloud infrastructure. International Journal of Advanced Intelligence Paradigms, 2018, 10(4):329. https://doi.org/10.1504/IJAIP.2018.092030

[18] Akter S, Dao T N, Yoon S. Time-constrained Task Allocation and Worker Routing in Mobile Crowd-Sensing using a Decomposition Technique and Deep Q-learning. IEEE Access, 2020, PP(99):1-1.