Hindawi
Wireless Communications and Mobile Computing
Volume 2018, Article ID 2734219, 8 pages
https://doi.org/10.1155/2018/2734219

Research Article

Immune Scheduling Network Based Method for Task Scheduling in Decentralized Fog Computing

Yabin Wang, Chenghao Guo, and Jin Yu

Science and Technology on Information Systems Engineering Laboratory, Nanjing 21007, China

Correspondence should be addressed to Yabin Wang; 517129269@qq.com

Received 2 May 2018; Accepted 15 July 2018; Published 2 September 2018

Academic Editor: Fuhong Lin

Fog computing has changed the distributed computing rapidly by including the smart devices widely distributed at the network edges. It is able to provide less latency and is more capable of decreasing traffic jam in the network. However, it will bring more difficulties for resource managing and task scheduling especially in a decentralized ad hoc network. In this paper, we propose a method that takes advantages of the immune mechanism to schedule tasks in a decentralized way for fog computing. By using forward propagation and backward propagation in the ad hoc network, the power of distributed schedulers is used to generate the optimized scheduler strategies to deal with computing nodes overloaded and achieve the optimal task finishing time reducing. The experiment results show that our approach can beat similar methods.

1. Introduction

With the number of smart wearable devices increasing and in-vehicle systems continuing improving, more and more users depend on mobile computing to deal with affairs in everyday life. Many mobile applications rely on offloading remote resources to complete their tasks. The remote resources mainly consist of large-scale computing clusters within a data center. These cloud computing clusters are also used for storing data.

User applications access the cloud center by using access points to exchange data between the data center and the user applications by using the core network. The computing capability of the edge network makes the access points being able to provide computing and storage services [1]. Edge computing nodes communicate with cloudlets. Cloudlets can also exchange information including control/management data (e.g., computing nodes and applications state) with each other to manage data and processes. Data storage and analysis is one hop away from the place where data are produced and consumed. This infrastructure can benefit different types of user applications. These application types are as follows:

(i) Applications that need to be run in low latency, e.g., surveillance applications and tactical applications will benefit from being only one hop away to the cloudlet.

(ii) If applications relying on the cloud to analyze large amount of data deploy themselves in a fog and the analyzing procedure is performed in the cloudlet one hop away, shorter time of delay and response time can be achieved. Less data traffic can also be achieved this way.

(iii) Large amount of unprocessed raw data collected by many edge devices need not to be stored in the remote cloud for long term. Data can be filtered, analyzed, and aggregated to generate knowledge and less amount of data need to be stored. The knowledge can be also used for other edge devices. The fog can reduce the network traffic from edge to the remote cloud in the above both cases.

Fog computing is able to provide less latency and is more capable in decreasing traffic jam in the network. But this will bring more difficulties for resource managing and task scheduling. More new challenges need to be overcome. For example, if a fog node is overloaded, on which node and when to respond to corresponding requests processed by the overloaded nodes should be determined. Generated strategies must consider the mobility of data and computing nodes [2]. This is even more difficult when the decisions are made in a fully decentralized ad hoc network in fog

computing [3]. However, most of the existing scheduling methods depend on a centralized controller to perform task scheduling which is not suitable for the ad hoc scenario [4]. In this paper, we first introduced the immune mechanism which has the characteristics of self-organization, cooperation, and robustness to reschedule tasks to deal with the problem of overload in fog computing.

The rest of the paper is organized as follows. The related work to this paper is introduced in Section 2. Proposed approach is introduced in Section 3. Experiments and the results are discussed in Section 4 and followed by the conclusion in Section 5.

2. Related Work

Previous researches about scheduling in a distributed scenario can be classified into 2 classes as follows [5].

(i) Deterministic methods: these methods use some characteristics of a problem to solve it.

(ii) Nondeterministic methods: random search methods techniques are used.

The deterministic methods include list-scheduling algorithms and clustering algorithms [6–8]. List-scheduling algorithm first creates one priority list of tasks. Then one task is selected from the list and assigned to one node based on some heuristics. List-scheduling algorithms can be further classified into dynamic and static list-scheduling according to whether the initial priority list changes during the executing of the algorithm. The priority list will not be changed in the executing of static list scheduling. The drawbacks of list-scheduling are that they will not achieve consistent results for different problems. The reason is that some specific heuristics are not persistent to problems with different properties. The most popular list-scheduling methods include Modified Critical Path (MCP) [8], Mapping Heuristic (MH) [9], and Dynamic Critical Path (DCP) [10]. Clustering algorithms group tasks with high data dependency together. The group is called a cluster task. The group of task will be assigned to computing nodes that are adjacent to each other. By this, the makespan can be reduced. Clustering methods first assume that unbounded number of computing nodes can be used and then the number decreases to the realistic number [11]. The most popular cluster methods are Dominant Sequence Clustering (DSC) [12].

The nondeterministic includes population-based algorithms such as PMC_GA [13]. The method takes longer time to find scheduling solutions compared with the above methods. The advantage of PMC_GA is to be suitable for wider range of problems compared with the above-mentioned methods.

3. Proposed Approach

Our approach effectively uses the ability of geographically distributed computing nodes to make schedule decisions. The scheduler is distributed in each computing node. Each scheduler will generate scheduling polices. Each scheduler will corporate with each other to generate better scheduling policies.

The scheduler in the computing node consists of an immune scheduling network. The network uses immune mechanism to balance the scalability and scheduling quality. The network is a distributed cooperative scheduling framework, and the local scheduler in each computing node makes scheduling decisions independently. The local scheduler collaborates with one another and synchronizes information. Tasks are allocated to local schedulers, and local schedulers communicate with each other regularly, exchanging information of each computing node's network and computing load. The framework avoids conflict scheduling decisions caused by fully decentralized scheduling and avoids centralized bottleneck and single point failure of schedulers.

First we will describe the concepts in the immune mechanism and the immune scheduling network and then describe the detailed procedure of our approach.

3.1. Immune Mechanism. The immune system is one of the most important systems in our body. It is able to protect our body from the pathogens and abnormal behaviour of some elements within our body [4]. One of the most important constituents of the immune system is lymphocytes. Lymphocytes are also called white blood cells. These cells are generated in the bone marrow.

The natural immune system protects our body from the harmful foreign elements and abnormal behaviour of our body. It is one of the most important organs in our body. One of the most important constituents of our immune system is blood cells. Blood cells are created in the bone marrow. They exist in the blood and lymph system and perform immunological functions. The main population of lymphocytes consists of B and T cells [14].

B cells and T cells have receptors. Receptors of B cells will recognize pathogens and then the B cell will clone and differentiate itself to be proliferated. Among the cloned cells, some are antibodies. Antibodies are responsible for combating with antigens. Some of the cloned cells are responsible for memorizing the type of attacks so that it will get a faster reaction when facing the same attack for the second time. The affinity between each antibody and each antigen will be calculated. The affinity will be improved by mutating antibodies. Better antibodies for combating with specific antigens can be selected by choosing antibodies with higher affinities. The essential parts of immune mechanism are the cloning, mutation, and selection.

The immune network theory [15] is another important concept. The core idea in immune network theory is that antibodies will stimulate and recognize each other. An antibody will be stimulated and cloned when recognizing another antibody. The recognized antibody will be suppressed. This procedure will avoid producing inordinate antibodies and make the whole immune system stable. The immune network, clone, and selection lead to diversity of population and improved searching for optimized solutions.

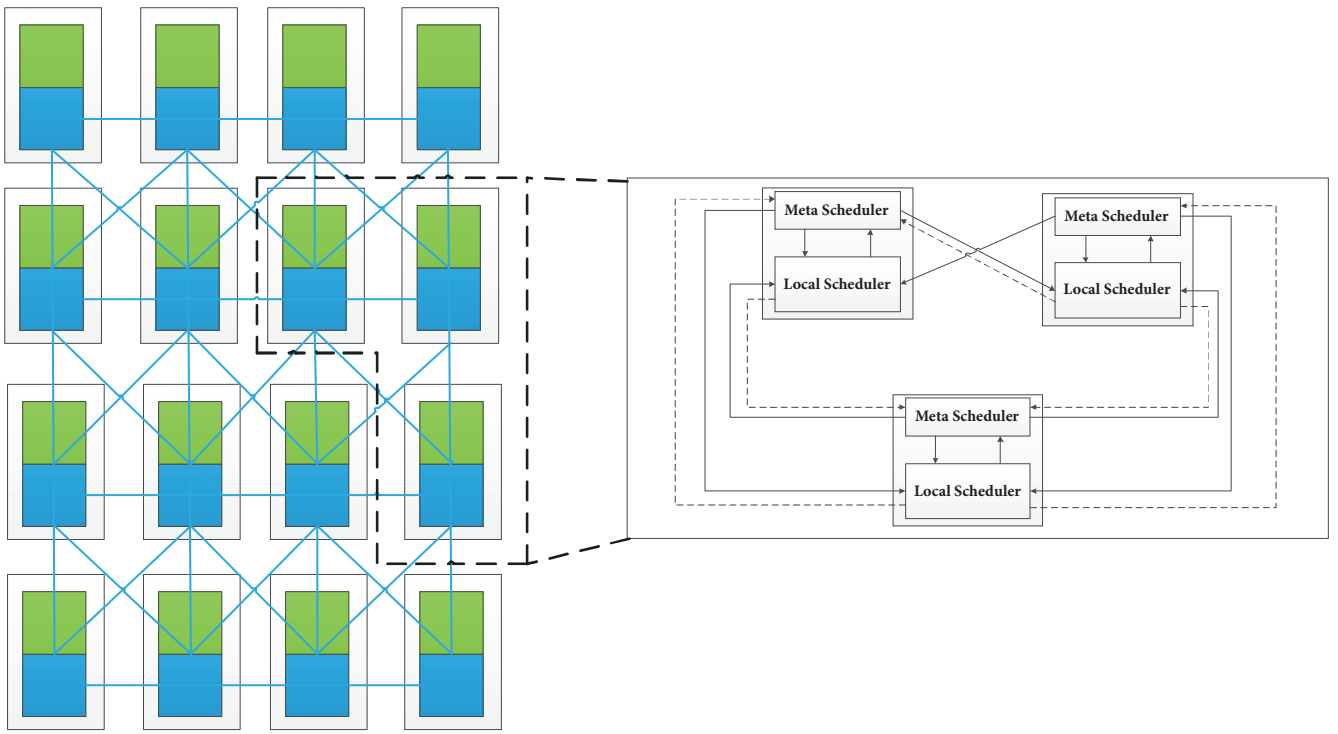

Figure 1: The architecture of immune scheduling network.

3.2. Immune Scheduling Network. The immune scheduling network as shown in Figure 1 consists of meta-schedulers and local schedulers. The network uses the immune network to implement the scheduling. The immune network is based on the interaction between antibodies and the communication between different kinds of immune cells. The immune network regulates the mutual promotion and inhibition between the antibodies to match the antigen and the antibody.

The main function of B cells is to produce antibodies. Immune networks rely mainly on antibodies to attack invasive antigens to protect organisms. The B cell is equivalent to the local scheduler in the immune scheduling network. The local scheduler exists in every computing node, which produces the equivalent of the scheduling strategy, and the antigen is equivalent to the problem that the scheduling policy needs to solve such as minimizing the execution time and minimizing the network traffic. The antibody is a scheduling strategy relative to the problem to be solved.

Meta-scheduler collects the tasks of the surrounding nodes to occupy the resource information (information including CPU, memory, bandwidth, hard disk storage, etc.), and broadcasts the information to adjacent nodes, and the adjacent nodes learn continuously to obtain more and more nonadjacent computing nodes.

Local scheduler learns the information of meta-scheduler and generates antibody according to immune mechanism (scheduling strategy), and the scheduling strategy of different nodes will choose the best solution.

3.3. Detailed Procedure. When some computing nodes suffer from overloading or damaging, two steps are implemented.

3.3.1. Nonspecific Immunization Stage. This stage is an emergency treatment stage. The nodes around the overloaded nodes quickly evaluate their own resource status to determine whether their resource can execute the tasks. If the evaluation result is yes, the tasks will be assigned to a qualified node randomly by the task requester fast. After this stage the nodes around overloaded node may also suffer from overload. Then the second procedure needs to be performed to balance the amount of tasks of the nodes in the network.

3.3.2. The Specific Immunization Phase. The schedulers working as immune cells need to identify the characteristics of the scheduled tasks working as antigen. The scheduler needs to generate a targeted scheduling strategy based on the type of task in the overloaded node. Different types of task correspond to different optimization target functions (the affinity function of antibodies and antigens). The target function includes minimizing the execution time.

After determining the objective function, the information of overloaded node needs to be broadcast nodes by nodes to inform the nodes not adjacent to the overloaded node, as nodes can only sense the nodes around them. The informed nodes may activate the local scheduler to generate scheduler strategies. Schedulers need to use the immune learning methods to generate some antibodies according to the resources of adjacent nodes. The strategies generated by different schedulers will be compared and the best one will be selected as the final scheduling strategy. Immune learning is complex if all the schedulers in the network all perform immune learning; it will cost too much time. In our approach, we will use path selection to select path that the overloaded

information being broadcast so that only a subset of scheduler needs to be activated.

This kind of distributed scheduling will cause a few nodes to be dispatched frequently, causing overheating [16]. Immune suppression mechanism can avoid this situation.

3.4. Forward Propagation and Backward Propagation. The propagation process is composed of forward propagation process and back propagation process. In the forward propagation process, the activation signal is transported nodes by nodes. If the scheduler is activated, the scheduler will use the immune mechanism to generate antibodies. The key to forward propagation is to find a scheduler activating path that can generate the optimized antibodies. In the backward propagation process, all the antibodies generated by activated schedulers will be selected, cloned, mutated, and compared to generate the final antibodies.

In order to find the optimized scheduler activated path, our approach uses the objective function to evaluate the local and surrounding resources and assign a score S_1 to it. S_1 will be distributed to the surrounding nodes. The surrounding nodes will use the same way for evaluation to get S_2. $|S_1 - S_2|$ will be used as input to the local activation function. The activation function determines whether the scheduler is activated or not.

The aim of the activation function is to evaluate whether the node and its surrounding nodes are eligible to be added to a list of qualified antibodies. The idea of the activation function is that if S_1 is high, scheduler with higher scores than S_1 should be activated. In order to get more appropriate schedulers, schedulers with approximate but less score are still activated. If S_1 is low, the situation is the opposite. In order to deal with these two situations, the activation function must be designed carefully.

In computational network, the activation function is the heart of the neural network [17]. Each node in the neural network receives the input value and passes the input value to the next layer. The input node directly transfers the value of the input attribute to the next layer (hidden layer or output layer). In the neural network, there is a functional relationship between the input and output of the hidden layer and the output layer node, which is called the activation function.

The output of each node in the neural network is based on the definition of the relevant activation function. The activation function is sometimes called the processing unit function or the compression function. It is used to input a set of inputs on the input arc. The activation function is also called the ignition rule, which makes it connect with the work of the human brain. When a neuron's input is large enough, it will ignite, that is, to send electrical signals from its axon (output connection). Similarly, in artificial neural networks, the output rule is generated when the input exceeds a certain standard, which is the idea of ignition rules. When dealing with only two value outputs, the output is either 0 or 1 depending on whether the neuron should be ignited [18]. In our research, the activation function will determine whether a node will create antibodies. Bekir [19] studied different action activation functions and found the best function Tanh which is used in our approach. The function is represented as

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{1}$$

Figure 2 shows an example of finding a scheduling activating path in forward propagation. There are 11 computing nodes composing an ad hoc fog network. Each box represents a node. The green rectangle inside a box represents a task running in the node. A blue box indicates that the computing node is activated. Now node A is overloaded. The surrounding nodes B, C, D, and J cannot process all the tasks in A, so the specific immunization phase has to start. D and J get higher scores after evaluation using objective function. D and J are activated (a). The objective function will be described in detail in the following section. E gets higher scores than D so E is activated (b). G and F get lower scores than E. I and K get similar high scores than J. G and F are not activated and I and K are activated (c). L has the same high scores as I does, so L is activated.

In the forward propagation procedure, each activated scheduler uses the immune mechanism described in Section 3.3 to generate the best antibodies. The generated antibodies will be transported back to the overloaded computing node in backward propagation. During the transportation the generated antibodies will be further compared and the best ones remained. Figure 3 shows an example of backward propagation. When antibodies are transformed from I to J, the generated antibodies will be compared using the objective function. The antibodies with the highest scores will remain.

3.5. Objective Function. Evaluating an antibody involves evaluating the computing capability of each node, bandwidth of communication between pairs of nodes, and other properties, e.g., communication methods of protocols between processors. Some researches [4] considered the environment homogenous. In the real fog environment, the nodes may have different types of processors and communication bandwidth. In our research, heterogeneous environment, processors, and message transferring ways are assumed. We also assumed that the nodes will suffer from failure. The notation is described as follows. P indicates processors. p_i indicates the ith processor. P_c indicates the number of processors.

The aim of scheduling tasks is to minimize the makespan. The aim can be formulated as

$$f : t_s \longrightarrow N_c \tag{2}$$

The function creates a map between a task and a computing node. The notations are denoted by

$$T_A = \left\{ t_j \in T \mid f\left(t_j\right) = n_i \right\} \tag{3}$$

T_A indicates the whole task set that has been assigned to the computing node n_i. The completion time of the whole task in node i is presented as

$$c_{i=} \left\{ rft\left(t_j\right) \right\} \quad where\ t_j \in T_A \tag{4}$$

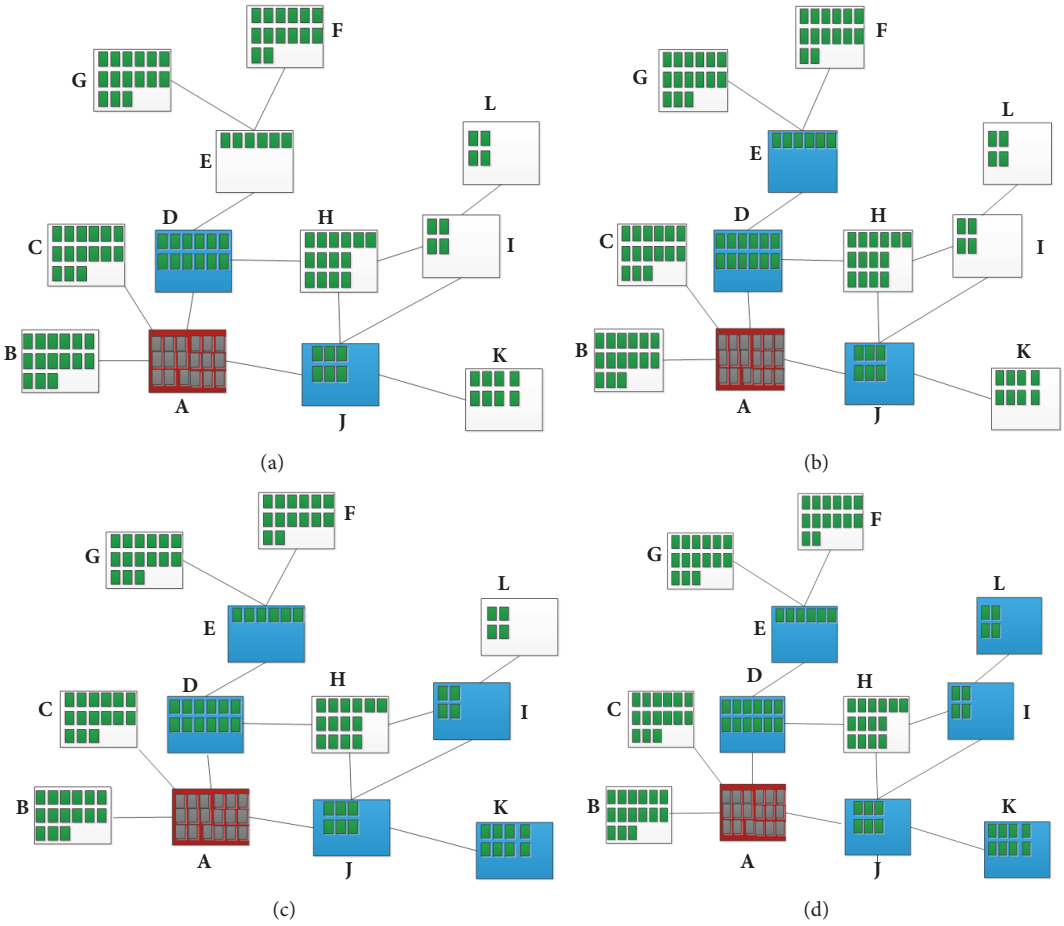

Figure 2: An example of selecting path in forward propagation.

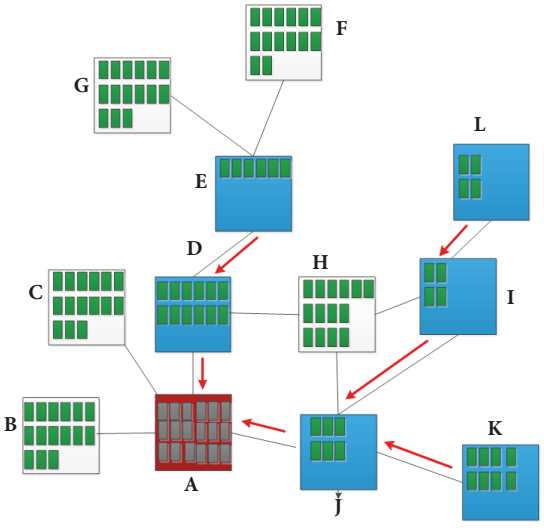

Figure 3: An example of backward propagation.

t_j is the last executed task that is assigned to the computing node *i*. $rft(t_j)$ is the realistic finish time of task t_j.

$$rft\left(t_j\right) = \left\{rst\left(t_j\right) + et\left(t_j, f\left(t_j\right)\right)\right\} \tag{5}$$

$et(t_j, n_i)$ is the time of task t_j executing on node i. *rst* is the realistic starting time of a task. A task is ready if all its predecessor tasks have been executed and the data the task required have been ready. At this moment, the task started. In this paper, the time is called the first start time (*fst*). *fst* is formally represented as

$$\begin{aligned} fst\left(t_i\right) = \max \{&rft\left(t_j\right) + w_{ij} \\ &* C\cos\left(f\left(t_i\right), f\left(t_j\right)\right) \mid t_j \in pre\left(t_i\right)\} \end{aligned} \tag{6}$$

In this equation, C cos is represented as

$$C\cos\left(n_i, n_j\right) = \begin{cases} 0, & if\ i = j \\ r_{ij}, & if\ i \neq j \end{cases} \tag{7}$$

rij is the communication cost between computing nodes i and j. When calculating the C cos, the distance between f(ti) and f(tj) is also considered.

rst(tj) indicates the time the task is ready to be started and there is no other task with higher priority to be processed. rst(tj) is represented as

$$rst\left(t_j\right) = \left\{fst\left(t_j\right) + waittime\left(t_j\right)\right\} \tag{8}$$

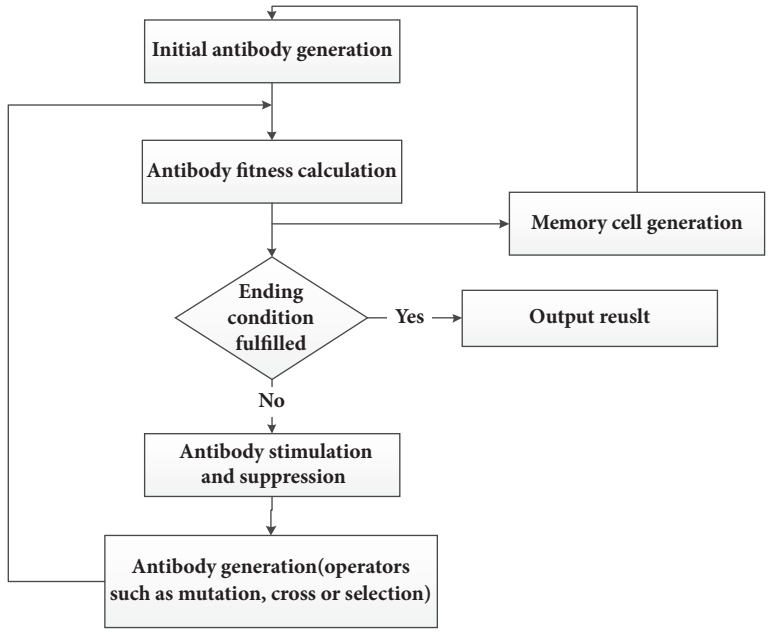

Figure 4: The detailed procedure.

Waittime(Tj) is the time that task Tj which is ready waits for task with higher priority that is ready to be finished. The makespan is represented as

$$C\max = \max\left(c_i\right) \quad i = 0, 1, \ldots n_i \tag{9}$$

The objective function of our approach is

$$obj = \frac{cm - \min\left(cm\right)}{\max\left(cm\right) - \min\left(cm\right)} \tag{10}$$

In this function, cm = C max. min (cm) and max (cm) are the minimum and maximum value of all the solutions.

It is important to note that the time cost of each the task running on each node and the time cost of communication between nodes need to be tested and recorded beforehand. The time cost will be input of our objective function.

The algorithm started with a set of population generated at random. The population will be further improved in several iterations. Each antibody represents a candidate for the solutions of the scheduling. The algorithm encodes the antibody as a string consisting of integers. The indexes of the integers in the string indicate the index number of tasks. The value of an integer indicates the computing node that the corresponding task will be assigned to.

3.6. Antibody Generating. The basic idea of generating antibodies is as follows:

(1) Using the basic operation of genetic algorithm and the way of population evolution to generate the optimal antibody

(2) Based on the mechanism of lymphocyte interaction in the immune network to regulate antibody affinity, to suppress similar antibodies

(3) Using immune memory to accelerate the convergence of the algorithm

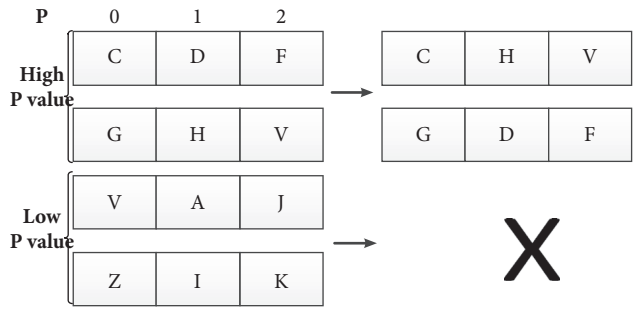

Figure 5: Antibody stimulation and suppression.

The detailed procedure is shown in Figure 4.

Initial antibody generation: it generates a random population. The population of antibody determines the order of executing the tasks. During the execution, the makespan can be calculated. After this is the clonal selection procedure. The clonal selection results in the search in the solution space and gets the best solution from the whole space. The key is the tradeoff between exploration and exploitation [4]. The clonal selection procedure includes antibody fitness calculation, antibody stimulation and suppression, and antibody generation.

As shown in Figure 5, an antibody is represented as an array where the index of each element represents a task index. The elements in the array represent the node to which the corresponding task needs to be assigned to. In Figure 5, numbers 0, 1, and 2 tasks are dispatched to C, D, and F nodes, respectively.

Antibody fitness calculation: it calculates fitness of each antibody using the objective function. The antibody with the highest fitness is retained. After several iterations of stimulation and inhibition, the best antibodies are retained.

The immune-remove phase follows the clonal selection. The elitism is provided by the immune phase. To make the next population more diverse, the elitism and random population insertion is performed. The algorithm will be iterated for K times. K is set in the begging. The output of the algorithm is the best antibody of all the populations and the order of executing the task.

Memory cell generation: after the fitness of each antibody calculated, the maximum fitness of the total population will be calculated. If the maximum fitness value is greater than the maximum fitness value provided by the immune network, the corresponding individual is added to the antibody memory table as a new good antibody. The maximum fitness value will be also added to the antibody adaptation table; otherwise, the immune memory process started. According to the historical antibody table, the immune memory uses the chaos multiplication to find the individual with highest fitness value to work as the better immune antibody [20].

Antibody stimulation and suppression: many highly frequent mutations will destroy the affinity between antibodies and antigens. It is necessary to selectively increase the number of high affinity antibodies to solve this problem. The replication operation of the next generation is carried out according to the replication probability calculated in the antibody production operator.

In a population, the expected reproductive probability of each individual is determined by both the affinity between antibody and antigen and the antibody concentration.

$$P = \alpha \frac{A_v}{\sum A_v} + (1 - \alpha) \frac{C_v}{\sum C_v} \quad (11)$$

α is a constant. Av indicates the fitness of an individual. Cv indicates the concentration of the individual. The higher the Av and Cv are, the greater the expected reproductive rate will be. A larger value of α encourages the generation of individuals with high fitness and suppresses the individuals with high concentration as shown in Figure 5.

Antibody generation: different operators have different global search capability [21]. Qiuzhen Lin [22] proposed a new operator which gained a wider diversity by including a suitable parent selection strategy. This operator is used to make the scheduler strategies more diverse to make the more optimized antibody to be generated in the earlier iterations. In order to generate more optimized antibodies, each antibody has to be cloned for Cc times. Cc is a positive integer value and works as one of the inputs of our approach. Each cloned antibody will be mutated. Random elements in the antibody will be selected and on which the operator described above needs to be performed. After cloning the fitness of each clone is calculated. The clone with the highest fitness is added to the antibody population.

4. Experiment and Result

In our experiment, proposed approach is compared with well-known scheduling algorithms including MCP [8], DSC [9], MD [8], DCP [10], and PMC_GA. A popular parameters setting method used in [4, 11] is used in our experiment. The parameter setting is described in Table 1.

Table 1: Parameter setting.

Parameter	value
Number of antibodies	400
Number of clones	50
Selection rate	0.25

Table 2: Experiment results for application with 8 tasks.

Approaches	MCP	DSC	MD	DCP	PMC_GA	Proposed
Number of nodes	8	8	8	8	8	8
Finish time (s)	26	23	29	27	20	13
Number of iterations	52	47	54	50	39	28

Table 3: Experiment results of application with 18 tasks.

Approaches	MCP	DSC	MD	DCP	PMC_GA	Proposed
Number of nodes	8	8	8	8	8	8
Finish time (s)	530	460	461	443	443	320
Number of iterations	120	106	105	100	101	88

Table 4: Experiment results of application with 50 tasks.

Approaches	PMC_GA	Proposed
Number of nodes	8	8
Finish time (s)	719	692

Two kinds of Gaussian Elimination application [11] are used for scheduling, one containing 9 tasks and the other containing 18 tasks. In order to compare proposed approach with other approaches a simulation experiment is conducted. In the experiments random node is selected to be made overloaded. Then the scheduling procedure is triggered. The finish times of all the tasks are recorded and compared. The results for application with 8 tasks are shown in Table 2. It is evident that our approach can achieve smaller finish time including the time of scheduling tasks.

Table 3 shows the results of application with 18 tasks. It is obvious that our approach can achieve lower finish time and smaller number of iterations.

In order to evaluate the performance of our approach on more complex application with 50 tasks, different conditions are considered based on some popular used task graph database [23]. The communication cost of tasks is added to make the simulation more realistic. The best approach of all the traditional ones is compared with our approach. The termination condition is set according to the study of Vahid Majid Nezhad et al. [22]. The termination condition is that the fitness remains unchanged in 10 iterations. The results are shown in Table 4. It is evident that our approach is still better than PMC_GA in more complex applications.

5. Conclusions

In this paper, a decentralized immune scheduler network based method is proposed to assign tasks in an ad hoc network for fog computing. This method takes advantage of

the immune mechanism to scheduler tasks in a decentralized way in an ad hoc network for fog computing. In the proposed method, by using propagation and backward propagation in the ad hoc network, the power of distributed schedulers is used to generate the optimized scheduler strategies to deal with computing node overloaded and achieve the optimal task finishing time decrease. The experiment results show that our approach can beat similar methods.

Data Availability

The experimental data used to support the findings of this study have not been made available because the data are related to our lab's commercial secrets.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

The study is funded by the Pre-Research on the Equipment of the Army Information System Project [no. 3511080401].

References

[1] L. F. Bittencourt, J. Diaz-Montes, R. Buyya, O. F. Rana, and M. Parashar, "Mobility-Aware Application Scheduling in Fog Computing," *IEEE Cloud Computing*, vol. 4, no. 2, pp. 26–35, 2017.

[2] C. Mouradian et al., "A Comprehensive Survey on Fog Computing: State-of-the-art and Research Challenges," *IEEE Communications Surveys & Tutorials*, vol. 99, p. 1, 2017.

[3] K. A. Hummel and G. Jelleschitz, "A robust decentralized job scheduling approach for mobile peers in ad-hoc grids," in *Proceedings of the 7th IEEE International Symposium on Cluster Computing and the Grid, CCGrid 2007*, pp. 461–468, May 2007.

[4] M. Sanei and N. M. Charkari, "Hybrid heuristic-based artificial immune system for task scheduling," *International Journal of Distributed & Parallel Systems*, vol. 2, no. 6, 2011.

[5] Y.-K. Kwok and I. Ahmad, "Static scheduling algorithms for allocating directed task graphs to multiprocessors," *ACM Computing Surveys*, vol. 31, no. 4, pp. 406–471, 1999.

[6] B. Kruatrachue and T. Lewis, "Grain size determination for parallel processing," *IEEE Software*, vol. 5, no. 1, pp. 23–32, 1988.

[7] G. C. Sih and E. A. Lee, "Compile-time scheduling heuristic for interconnection-constrained heterogeneous processor architectures," *IEEE Transactions on Parallel and Distributed Systems*, vol. 4, no. 2, pp. 175–187, 1993.

[8] M. Wu and D. D. Gajski, "Hypertool: a programming aid for message-passing systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 1, no. 3, pp. 330–343, 1990.

[9] H. El-Rewini and T. G. Lewis, "Scheduling parallel program tasks onto arbitrary target machines," *Journal of Parallel and Distributed Computing*, vol. 9, no. 2, pp. 138–153, 1990.

[10] Y. Kwok and I. Ahmad, "Dynamic critical-path scheduling: an effective technique for allocating task graphs to multiprocessors," *IEEE Transactions on Parallel and Distributed Systems*, vol. 7, no. 5, pp. 506–521, 1996.

[11] R. Hwang, M. Gen, and H. Katayamaa, "A comparison of multiprocessor task scheduling algorithms with communication costs," *Computers & Operations Research*, vol. 35, no. 3, pp. 976–993, 2008.

[12] T. Yang and A. Gerasoulis, "DSC: scheduling parallel tasks on an unbounded number of processors," *IEEE Transactions on Parallel and Distributed Systems*, vol. 5, no. 9, pp. 951–967, 1994.

[13] Y. C. Lee and A. Y. Zomaya, "An Artificial Immune System for Heterogeneous Multiprocessor Scheduling with Task Duplication," in *Proceedings of the 2007 IEEE International Parallel and Distributed Processing Symposium*, pp. 1–8, Long Beach, CA, USA, March 2007.

[14] D. Dasgupta and F. Nino, "Immunological computation : theory and applications," in *Longman*, p. 140, 2008.

[15] N. K. Jerne, "Towards a network theory of the immune system," *Annales Dimmunologie*, vol. 125, no. (1-2), p. 373, 1974.

[16] Y. Huang, N. Bessis, P. Norrington, P. Kuonen, and B. Hirsbrunner, "Exploring decentralized dynamic scheduling for grids and clouds using the community-aware scheduling algorithm," *Future Generation Computer Systems*, vol. 29, no. 1, pp. 402–415, 2013.

[17] P. Ramachandran, B. Zoph, and Q. V. Le, *Searching for Activation Functions*, 2017.

[18] I. H. Witten and E. Frank, *Data Mining. Practical Machine Learning Tools & Techniques with Java Implementations*, vol. 13, 2005.

[19] B. Karlik and A. V. Olgac, *Performance Analysis of Various Activation Functions in Generalized MLP Architectures of Neural Networks*, 42, Cambridge University Press, 2010.

[20] D. E. Goldberg, "Genetic algorithms in search, optimization, and machine learning," *Choice Reviews Online*, vol. 27, no. 02, pp. 27-0936–27-0936, 1989.

[21] W. Gong, Á. Fialho, Z. Cai, and H. Li, "Adaptive strategy selection in differential evolution for numerical optimization: an empirical study," *Information Sciences*, vol. 181, no. 24, pp. 5364–5386, 2011.

[22] Q. Lin et al., "A novel hybrid multi-objective immune algorithm with adaptive differential evolution," *Computers & Operations Research*, vol. 62, pp. 95–111, 2015.

[23] "Standard Task Graph Set," http://www.kasahara.elec.waseda.ac.jp/schedule.