International Journal of Computer Applications Technology and Research
Volume 13–Issue 10, 79 – 83, 2024, ISSN:-2319–8656
DOI:10.7753/IJCATR1310.1007

Research on Edge Task Offloading Problem Based n Dual Fitness Genetic Algorithm

Chengyu Hou
College of Communication Engineering,
Chengdu University of Information Technology,
Chengdu, China

Wenzao Li*
College of Communication Engineering,
Chengdu University of Information Technology,
Chengdu, China

Sai Yao
College of Communication Engineering,
Chengdu University of Information Technology,
Chengdu, China

Abstract: With the widespread adoption of Internet of Things (IoT) technology across various sectors, leading to a substantial increase in terminal devices and task data volumes. Efficient task scheduling has thus become a critical challenge in cloud computing. To address this issue, this paper introduces a novel Dual Adaptive Genetic Algorithm (DAGA), building upon the original Adaptive Genetic Algorithm (AGA) but tailored for the evolving characteristics of cloud environments. DAGA not only prioritizes minimizing total task completion time but also emphasizes achieving balanced average task completion times. The study includes simulations in a cloud computing environment using Matlab, where DAGA is compared against AGA. Test parameters are carefully set and adjusted to evaluate and contrast the original and enhanced algorithms. Through rigorous testing, DAGA demonstrates superior performance over AGA in terms of both total job completion time and average task completion time.

Keywords: Cloud computing; Genetic algorithm; Dual fitness; Task offloading

1. INTRODUCTION

With the rapid advancement of IoT technology, its integration into industries such as healthcare, manufacturing, and smart cities has led to an exponential increase in the number of connected devices and the volume of data generated [1]. This surge in IoT applications has placed immense demands on cloud computing infrastructures, where managing and offloading tasks efficiently has become a crucial challenge [2]. The increasing complexity of task data, the dynamic nature of workloads, and the need for real-time processing have intensified the need for more sophisticated offloading strategies. Current cloud computing environments require algorithms that can not only handle large-scale data but also optimize the overall system performance. Motivated by the limitations of traditional methods in meeting these demands, this paper explores enhanced approaches for task offloading in cloud computing, with the goal of improving both efficiency and resource utilization [3].

In modern cloud computing environments, task offloading faces several challenges. These include the fluctuating nature of workloads, the need to minimize total task completion time, and the challenge of balancing load across computing resources to prevent bottlenecks. Existing approaches like the AGA have shown promise but often fall short in accommodating the diverse and dynamic conditions of cloud systems [4]. The AGA typically focuses on optimizing total completion time, but it overlooks other critical factors such as the average completion time, which can lead to inefficiencies. To address these challenges, this paper proposes the DAGA. The DAGA enhances the traditional AGA by introducing a second fitness function that not only considers the total job completion time but also incorporates the average task completion time as a key factor, leading to more balanced workload distribution and improved overall performance.

This paper makes several key contributions to the field of cloud computing task offloading. First, it introduces the DAGA algorithm, which improves upon existing genetic algorithms by integrating a dual fitness function to better balance total and average completion times. Second, it provides a comprehensive comparison between the traditional AGA and the improved DAGA through extensive Matlab simulations, demonstrating significant improvements in performance. Lastly, the paper outlines a clear and practical framework for implementing DAGA in cloud environments. The structure of the paper is as follows: Section 2 reviews related work on task offloading algorithms, Section 3 presents the proposed DAGA method, Section 4 details the experimental setup and results, and Section 5 concludes with a discussion of the findings and potential future work.

2. RELATED WORK

This section describes the current state of research: Firstly, Karishma et al. improved the performance of Genetic Algorithm (GA) by introducing new crossover and mutation operators, and enhanced the functionality of traditional Particle Swarm Optimization (PSO) by combining it with GA [5]. Furthermore, Li et al. proposed an improved ga-encoding double chromosome with a conflict mediation mechanism to genetically manipulate populations while satisfying these limitations. The proposed GA ensures universal excellence in population evolution while significantly improving population optimization and convergence performance [6]. Last but not list, the strategy proposed by Deepak et al. focuses primarily on minimizing task execution time by treating it as a fitness function when GA is implemented. Reinforcement learning is integrated with the proposed algorithm to improve its performance while finding the optimal resource allocation [7].

3. SYSTEM MODEL

Mission alignment in the cloud environment is a multi-goal optimization issue that has proven to be a NP-hard problem [8]. There are two main aspects: from the reader's point of view, the completion time of the work should be minimized. The completion time should be within the tolerance period. Therefore, the completion time of the work should be minimized. In the cloud environment, mission offloading is a three-tier structural model, i.e., task request, resource management, and task execution. At the first level, users interact with the system; At the middle level, resource management plays an important role in job separation, task allocation and resource management. At the last level, the task is executed as an ordered predetermined sequence. Resource management is the key to task allocation. The specific framework is shown in Figure 1:

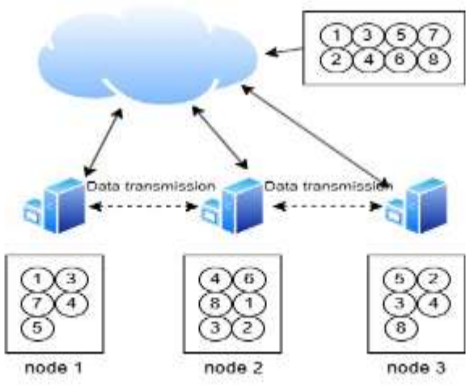


Figure 1: System flow framework diagram

In the figure, different numbers represent different tasks, and the tasks contained in the node represent calculations at that node.

3.1 Genetic algorithms

The genetic algorithm mimics the phenomena of multiplication, crossover, and genetic mutation in the natural world and natural inheritance processes [9]. It will each possible solution is treated as an individual in a group (all possible solutions), and each individual is encoded in the form of a string of characters, and each individual is evaluated according to a predetermined target function. Give an adaptation value. In the beginning, a number of individuals are born at random, and according to the adaptability of these individuals, genetic operators are used to manipulate these individuals to obtain a new group of individuals that inherit some excellent traits from the previous generation. As a result, it is clearly superior to the previous generation, and this is gradually moving towards a better understanding. The legacy algorithm simultaneously searches for different regions of the parameter space at the same time in each generation, and then concentrates the attention to the one with the highest medium-term value of the solution space partly, the possibility of finding the best solution for the whole situation is greatly increased [10]. The flow of the genetic algorithm is shown in Figure 2:

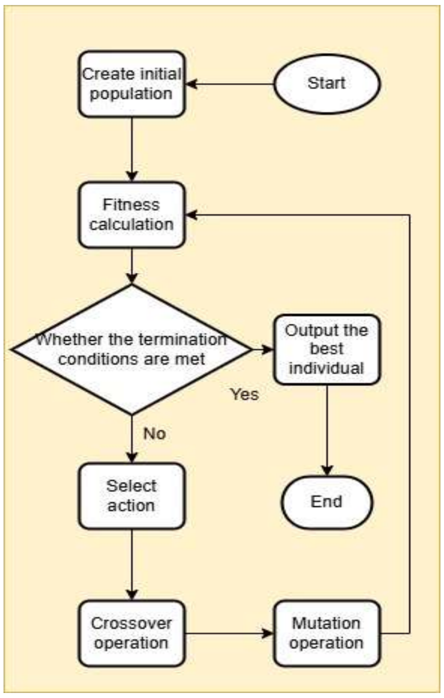

Create initial population
Start
Fitness calculation
Whether the termination conditions are met
Output the best individual
Yes
No
Select action
End
Crossover operation
Mutation operation

Figure 2: Genetic algorithm flow diagram

The objective of this paper is to complete the tasks in a cloud environment in a short time and to complete them evenly. It doesn't take too long to grow up either. Taking into account the average time spent on the task is conducive to the convergence and development of the algorithm, so as to find the best solution [11]. Therefore, the text defines two adaptation functions:

$$fitness_1 = \frac{1}{T_i} \tag{1}$$

where $1 < i < population$, T_i is the time when the i-th individual completes the overall task.

$$fitness_2 = time(i) = \frac{\sum_{t=1}^{N} Time(t,i)}{N} \tag{2}$$

where $1 < i < population$, $Time(t,i)$ represents the time taken to complete the t-th task in individual i.

With the double adaptation function, a measurement criterion is required before selecting a chromosome. In this paper, the parameters c_1 and c_2 are cited to show the probabilities of choosing p_1 and p_2, respectively, where $c_1 > 0$, $c_2 < 1$, $c_1 + c_2 = 1$. Randomly select one as the probability of the selected individual. As a result of the above operations, there are individuals in the population with a short total completion time and a short average completion time. The selection probabilities are shown in equations (3) and (4):

$$P_1(i) = \frac{fitness_1(i)}{\sum_{i=1}^{Population} fitness_1(i)} \quad (3)$$

$$P_2(i) = \frac{fitness_2(i)}{\sum_{i=1}^{Population} fitness_2(i)} \quad (4)$$

where $fitness$ indicates the individual's suitability in choosing the fitness function $P(i)$; $Population$ indicates population size. Finally, a round-robin strategy is used to make individual selections. As mentioned above, the greater the individual adaptation value, the greater the likelihood that the individual will be selected. In order to implement this algorithm, the program uses the $rand$ function to automatically obtain a number k , where $0 < k < 1$. If the number of machines k is full of $P_1 + P_2 + \cdots + P_{i-1} < k \le P_1 + P_2 + \cdots + P_i$. Then the first i-individual will be selected.

The method of this algorithm is based on the random exchange of gene pairs. If the two individuals X_i and X_j are crossed, one or more pairs of the allied genes on X_i and X_j are swapped by randomly swapping the genes. The variation operator uses reverse variation, which is to randomly select a gene for chromosomes to perform a first-place inversion. The self-adapting crossover and variation probabilities used in the experiment are as follows:

$$P_c = \begin{cases} \frac{k_1(f_{max} - f')}{(f_{max} - \overline{f})}, & f \ge \overline{f} \\ k_2, & f < \overline{f} \end{cases} \quad (5)$$

$$P_m = \begin{cases} \frac{k_3(f_{max} - f')}{(f_{max} - \overline{f})}, & f \ge \overline{f} \\ k_4, & f < \overline{f} \end{cases} \quad (6)$$

where $k_1 + k_2 + k_3 + k_4 \le 1$, f_{max} is the maximum fitness values in the population, $\overline{f}$ is the average fitness value of each generation, f' is the greater fitness value of the two individuals to be crossed, and f is the fitness value of the individuals to be different.

3.2 Task offloading

For the purpose of analysing the total completion time of the assignments, the number of tasks in the overall definition of this document is , and since the computing capacity of each node is fixed, the execution time of the tasks can be estimated after the tasks are loaded. In this paper, the moment matrix is used to represent the number of task on the i-th on the i-th node [12]. Therefore, the time spent for the n-th task can be calculated from the decoded sequence and matrix, and the total time to complete the work is shown in Equation 7. If a single point fails to be achieved in the process, the average completion time of the operation can be expressed as Equation 8 :

$$Time(t) = \max_{i=1} \sum_{j=1}^{k} TaskTime(j,i) \quad (7)$$

$$\overline{Time} = \frac{\sum_{t=1}^{N} Time(t)}{N} \quad (8)$$

where $TaskTime(j,i)$ represents the time cost of executing $task\ i$ at node i , and $\overline{Time}$ represents the average completion time.

4. SIMULATION AND DISCUSSION

In order to evaluate the convergence of the legacy algorithm proposed in this paper, this chapter mainly tests the MATLAB simulation cloud environment task tuning problem. DAGA and AGA were compared in the same environment, and the test results were analyzed.

4.1 Experimental scheme and parameter setting

Suppose the number of physical nodes is 50, and the size of each block is 128M [12]. In order to verify the universality of the algorithm, the empirical data is modeled on the size of the data set up by the weekly submission. The total number of jobs is 30, and the number of tasks in a single operation varies from 1 to 100, as shown in Table 1. Matrix are randomly generated by the system. The specific parameter settings are shown in Table 2 [14]. If it is executed 200 times, if there is no obvious time change for 50 generations, the algorithm is considered to be approximately convergent and the algorithm is terminated. In addition, if the algebra of evolution exceeds the maximum algebra set, the algorithm is terminated.

Table 1 Job information

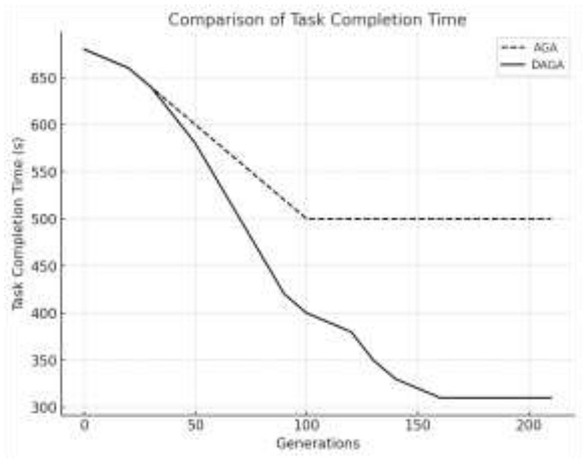
Number of jobs	Task number	Percentage
15	1-5	50%
8	6-20	27%
4	21-52	13%
3	53-100	10%

Algorithm	Parameter	Size
AGA	Population	100
	k_1	0.35
	k_2	0.85
	k_3	0.06
	k_4	0.08
DAGA	Population	100
	k_1	0.35
	k_2	0.85
	k_3	0.06
	k_4	0.08
	c_1	0.6
	c_2	0.4

Table 2 Parameter setting table

4.2 Experimental results and analysis

In order to verify the effectiveness of the algorithm, the ability of task offloading under the same task density is discussed. To reduce initialization errors, each experiment was performed 5 times under the same conditions. For the same number of tasks, the convergence results of different algorithms are shown in Figure 3 :



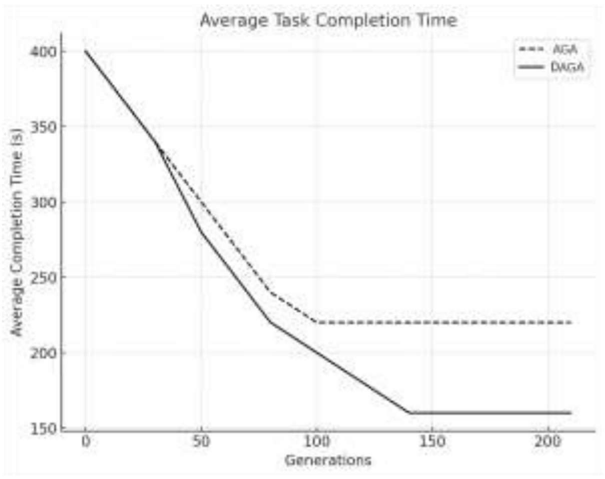


Figure 3: Comparison of Task Completion Time

Generally speaking, the total completion time of the operation, the average completion time, interacts in the process of evolution, and finally the experiment returns to a comprehensive task adjustment sequence that takes into account the above two aspects. However, DAGA is slightly slower to converge than AGA. AGA converges left and right in the 100th generation, while DAGA converges left and right in the 160th generation. This is due to more factors than consideration. This strategy of getting less time to complete work in a small iteration time is acceptable. Therefore, it can be considered that the dual-fitness arithmetic proposed in this paper that comprehensively considers the total completion time of the operation and the average completion time of the operation still has good convergence.

5. CONCLUSION

The advent of cloud computing has revolutionized the landscape of on-demand parallel processing for large-scale data, presenting an exhilarating opportunity to tackle complex computational tasks with unprecedented efficiency. However, a significant challenge lies in optimizing the completion time of individual assignments while ensuring these durations remain reasonably brief. This balance is crucial to maintain system responsiveness and resource utilization efficiency. In this context, genetic algorithms emerge as a powerful tool, leveraging their adaptive search capabilities to evolve a tuning sequence for task execution that minimizes both total and average operational times. This paper proposes a task tuning algorithm based on the legacy arithmetic to find a satisfactory solution for the operational tuning. The goal is to ensure that the overall completion time is short and the average work time per task remains reasonable.

Whole-of-the-road adaptive control strategies that can be considered in the future. For example, in order to maintain population diversity, population size should be larger in the early stages of evolution and smaller in the later stages of evolution; Individuals with good performance should perform more cross-manipulations, while individuals with low fitness should perform more variation manipulations to better match the fitness function. Considering the unique nature of the cloud environment, the time complexity should not be too high.

6. ACKNOWLEDGEMENTS

We would like to express our heartfelt gratitude to all those who have contributed to this research. Firstly, we would like to thank our supervisor for his/her valuable guidance and insightful comments throughout the entire research process. We also extend our sincere appreciation to all the participants who have generously devoted their time and effort to provide us with valuable data and insights. Finally, we would like to thank our families and friends for their unwavering support and encouragement. This research would not have been possible without their support. Also, we thanks for Sichuan Province Science and Technology Department, Sichuan Province major science and technology project, (No. 24JBGS0050). Sichuan Province Philosophy and Social Science Research Project, (No. SC23TJ006). Meteorological Information and Signal Processing Key Laboratory of Sichuan Higher Education Institutes of Chengdu University of Information Technology, the fund of the Scientific and Technological Activities for Overseas Students of Sichuan Province (2022) and Funded by the Sichuan Provincial Department of Human Resources and Social Welfare “Researches on Key issues of Edge Computing Server Deployment and Computing task Offloading”. Network and Data Security Key Laboratory of Sichuan Province, UESTC (No. NDS2024-3).

7. REFERENCES

[1] Radhakrishnan,,Indu,Jadon,,Shruti,Honnavalli,,Prasad,& B..(2024).Efficiency and Security Evaluation of Lightweight Cryptographic Algorithms for Resource-Constrained IoT Devices.SENSORS,24(12),4008.

[2] Zhu,,& Fengxia.(2024).Cloud computing load balancing based on improved genetic algorithm.INTERNATIONAL JOURNAL OF GLOBAL ENERGY ISSUES,46(3-4),191-207.

[3] Behera,,Ipsita,Sobhanayak,,& Srichandan.(2024).Task offloading optimization in heterogeneous cloud computing environments: A hybrid GA-GWO approach.JOURNAL OF PARALLEL AND DISTRIBUTED COMPUTING,183.

[4] Li,,Jianxia,Liu,,Ruochen,Wang,,& Ruinan.(2024).Handling dynamic capacitated vehicle routing problems based on adaptive genetic algorithm with elastic strategy.SWARM AND EVOLUTIONARY COMPUTATION,86.

[5] Karishma,Kumar,,& Harendra.(2024).A novel hybrid model for task offloading based on particle swarm optimization and genetic algorithms.MATHEMATICS IN ENGINEERING,6(4),559-606.

[6] Li,,Jiaxuan,Yang,,Xuerong,Yang,,Yajun,Liu,,& Xianglin.(2024).Cooperative mapping task assignment of heterogeneous multi-UAV using an improved genetic algorithm.KNOWLEDGE-BASED SYSTEMS,296.

[7] Deepak,,& B.B..(2024).Genetic algorithm with reinforcement learning for optimal allocation of resources in task offloading.International Journal of Cloud Computing,13(3),285-304.

[8] Antkiewicz,,Michal,Myszkowski,,Pawel,& B..(2024).Balancing Pareto Front exploration of Non-dominated Tournament Genetic Algorithm (B-NTGA) in solving multi-objective NP-hard problems with constraints.INFORMATION SCIENCES,667.

[9] Wei,,Huixian,Liu,,& Jia.(2021).Computer Mathematical Modeling Based on the Improved Genetic Algorithm and Mobile Computing.WIRELESS COMMUNICATIONS & MOBILE COMPUTING,2021.

[10] Alhijawi,,Bushra,Awajan,,& Arafat.(2024).Genetic algorithms: theory, genetic operators, solutions, and applications.EVOLUTIONARY INTELLIGENCE,17(3),1245-1256.

[11] Kamalinia,,Amin,Ghaffari,,& Ali.(2017).Hybrid Task offloading Method for Cloud Computing by Genetic and DE Algorithms.WIRELESS PERSONAL COMMUNICATIONS,97(4),6301-6323.

[12] Zhao,,Qian,Lin,,Yuji,Wang,,Fengxingyu,Meng,,& Deyu.(2024).Adaptive weighting function for weighted nuclear norm based matrix/tensor completion.INTERNATIONAL JOURNAL OF MACHINE LEARNING AND CYBERNETICS,15(2),697-718.

[13] Chen,,Ming,Qi,,Ping,Chu,,Yangyang,Wang,,Bo,Wang,,F ucheng,Cao,,& Jie.(2024).Genetic algorithm with skew mutation for heterogeneous resource-aware task offloading in edge-cloud computing.HELIYON,10(12),e32399.

[14] Zhang,,& Xiuyan.(2023).A Hybrid Method Based on Gravitational Search and Genetic Algorithms for Task offloading in Cloud Computing.INTERNATIONAL JOURNAL OF ADVANCED COMPUTER SCIENCE AND APPLICATIONS,14(6),30-36.

[15] Xidias,,E.,Moulianitis,,V,Azariadis,,& P..(2021).Optimal robot task offloading based on adaptive neuro-fuzzy system and genetic algorithms.INTERNATIONAL JOURNAL OF ADVANCED MANUFACTURING TECHNOLOGY,115(3),927-939.