LPT-One and BFD-One Search Algorithms for Load Balance and Bin-Packing of Cloud Computing

Guangyao Zhou[1], Wenhong Tian[1,*], Rajkumar Buyya[2,1]

[1]School of Information and Software Engineering, University of Electronic Science and Technology of China, China
[2] School of Computing and Information Systems, The University of Melbourne, Australia
Guangyaozhou_uestc@163.com, 459465455@qq.com, rbuyya@unimelb.edu.au

2023 IEEE 9th International Conference on Cloud Computing and Intelligent Systems (CCIS) | 979-8-3503-0442-8/23/$31.00 ©2023 IEEE | DOI: 10.1109/CCIS59572.2023.10263040

Abstract: Cloud computing contains large-scale tasks and resources. Currently, the local search is a considerable choice in ensuring both computational complexity and optimization. Based on our previous research on multi-route search algorithm to reduce makespan, we apply BFDO and LPTO algorithms to address load balancing and bin-packing problems. Through abundant experiments, we validate the superiority of our proposed BFDO and LPTO.

Keywords: Cloud; Scheduling; BFD-OneStep; Load Balancing; Bin-Packing; LPT-OneStep,

1 Introduction

The development of distributed systems increases the requirement of resource management; hence the demands of flexibility, reliability and robustness are emerging [1]. Pay-as-you-go manner is commonly used to ensure high-performance of Cloud [2]. The research on resource scheduling and task allocation will play a crucial role in promoting cloud and promoting the improvement of actual computing tasks [3].

Load balancing and bin-packing are two prevalent ways to optimize the resources of Cloud. In our previous study [4], we proposed multi-search-route algorithms to solve minimizing makespan problem and provided theoretical proof of their approximations. On the basis of our previous study [4], this paper applies our proposed multi-route local search algorithms in solving load balancing and bin-packing problems. Additionally, this paper executes extensive experiments in load balancing and bin-packing problems to validate the superiority of LPTO and BFDO.

The main contributions can be listed as follows:

(1) Applying LPT-One and BFD-One search algorithms to solve load balancing problem.

(2) Applying LPT-One and BFD-One search algorithms to solve bin-packing problem.

(3) Executing abundant experiments in load balancing and bin-packing problems to show the outstanding behavior of our algorithms.

2 Related Work

Multi-phase [5], VMs migration [6], queuing model [7] are some common methods to improve the resource management of Cloud except for scheduling algorithm. However, optimization-based scheduling algorithm is still the foundation of these methods.

In scheduling algorithms, branch and bound method (BB), dynamic programming (DP), linear programming (LP) etc. are included in exact algorithms.

Deep learning methods include DREP [8] and DLSC [9]. Reinforcement learning methods include QEEC [10] and ADEC [11]. Deep reinforcement learning has some examples that DRM_Plus [12] and DQTS [13]. These constitute machine learning algorithms in scheduling. However, due to the training characteristics of machine learning, its feasibility in real scenarios with large-scale tasks and resources is very limited, which requires a lot of measured data as support

Except for our proposed multi-route local search algorithms [4], Neighborhood search [14], Crow Search [15] and Tabu Search [16] are some typical local search algorithms. Due to the limitations of local convergence points, ordinary local search algorithms have insufficient optimization for resource management.

Hybrid algorithms, such as HGA-ACO [17], DAAGA [18], PSO-ACS [19] and FACO [20] are also applied in scheduling of distributed systems.

3 Problems Formulation

It can be set the maximum number of instructions that a CPU can process per time slot as the capacity of CPU $MaxL_j^{CPU}$ where the unit of CPU capacity means (number of instructions) / (unit time). Taking load of CPU that L_i^{CPU} as instance, the volume of tasks is V_{ij}^{CPU} which means the i-th task T_i requires V_{ij}^{CPU} (unit of CPU capacity) capacity of the j-th resource R_j when T_i is allocated to R_j where $L_j, V_{ij}^{CPU} \in \square^+$. Therefore, the parameter of task T_i can be substituted as $P_i = \langle V_{i1}^{CPU}, V_{i2}^{CPU}, ..., V_{iM}^{CPU} \rangle$. We consider the balance of the total load of resources; thus, we need to make additional assumptions about the features of tasks and resources as follows.

(1) Tasks occupy CPU in a resource simultaneously as the instantaneous load of resources will usually affect the task response of the whole cloud;

(2) The CPU usage of tasks for resources satisfies the

979-8-3503-0442-8/23/$31.00 ©2023 IEEE

superposition relationship.

Therefore, for load balancing, dual-objectives, i.e., minimizing the total load and maximizing degree of load balancing require considerations shown as follows assuming $\mu_{load}^{CPU} = \frac{1}{M}\min\sum_{i=1}^{N} V_{ij}^{CPU}$.

$$\begin{cases} \min \omega_{standard-deviation} = \sqrt{\dfrac{\sum_{j=1}^{M}\left(L_j^{CPU} - \mu_{load}^{CPU}\right)^2}{M}} \\ \min \omega_{total-load} = \sum_{j=1}^{M} L_j^{CPU} \end{cases} \quad (1)$$

For bin-packing problem, the properties of tasks and resources are analogous to the problem of load balancing except that the number of resources is variable. Then, the problem of bin-packing is:

$$\min \omega_{resources-number} = \min \sum_{j=1}^{M} \max x_{ij} \mid_{i=1,2,\ldots,N} \quad (2)$$

The constraints are:

$$\begin{cases} \sum_{j=1}^{M} x_{ij} = 1, x_{ij} \in \{0,1\}, L_j^{CPU} \le MaxL^{CPU} \\ \forall i \in \{1,2,\ldots,N\}, \forall j \in \{1,2,\ldots,M\} \end{cases} \quad (3)$$

4 Methodology

4.1 General Local Search Algorithm

A general algorithm of multi-routes search algorithm can be seen in Algorithm 1. Changing the search path in Algorithm 1 will generate different algorithms.

4.2 Specified basic local search route

Primarily, several basic local search routes are presented.

Algorithm 1: General Local Search Algorithm based on the Neighbors of Dual Resources

Input Tasks and Resources
Initially Allocate tasks to resources and gain the general aspect of resources by $A_j = h_j\left(E_{ij}|_{T_i \in TS_j}\right)$
while Yes_adjustment **do**
  Sort resources by their value A_j
  for *j in resources* **do**
    $or\_j = j, \beta = f(A_1, A_2, \ldots, A_M)$
    for *k in* [0, *j*) **do**
      Use **specified local search route** to adjust tasks belonging to TS_j and TS_k then gain TS'_j and TS'_k
      $c = max(j,k), b = min(j,k)$
      if $\beta > f\left(A_1, \ldots, A'_b, \ldots, A'_c, \ldots, A_M\right)$ **then**
        $pre\_j = k$
        $\beta = f\left(A_1, \ldots, A'_b, \ldots, A'_c, \ldots, A_M\right)$
        Yes_adjustment
    if Yes_adjustment **then**
      Update resources as $TS_{or\_j} = TS'_{or\_j}$, $TS_{pre\_j} = TS'_{pre\_j}$
      Break (for repeat of *j*)

4.2.1 One-Step search route

For load balancing and bin-packing, a sort-based algorithm for searching the optimal solution in One-Step neighborhood can be used to reduce the computational complexity seen in Algorithm 2.

Algorithm 2: One-step search route of R_j for load balancing, makespan and bin packing

Input tasks and resources, Set $\gamma = \mu_{load}^{CPU}$ for load balancing problem or $\gamma = MaxL^{CPU}$ for bin packing problem. $C = A_j - \gamma$
$B_1 = \arg\min_{T_i}\left(\left|E_{ij} - C\right|_{T_i \in TS_j}, \left|E_{ik} + C\right|_{T_i \in TS_k}\right)$
$vl_1 = \min\left(\left|E_{ij} - C\right|_{T_i \in TS_j}, \left|E_{ik} + C\right|_{T_i \in TS_k}\right)$
sort $\left\{E_{ij}|_{T_i \in TS_j}, E_{ik} + C|_{T_i \in TS_k}\right\} \rightarrow \{G_{\eta_1}, G_{\eta_2}, \ldots\}$
$B_2 = \arg\min_{T_{\eta_i}}\left|G_{\eta_i} - G_{\eta_{i+1}}\right|, vl_2 = \min\left|G_{\eta_i} - G_{\eta_{i+1}}\right|$ where $\{T_{\eta_i}, T_{\eta_{i+1}}\} \not\subset TS_j \wedge \{T_{\eta_i}, T_{\eta_{i+1}}\} \cap TS_j \neq \emptyset$
if $B_1 = \emptyset$ *and* $B_2 = \emptyset$ **then**
  Return No_adjustment
else
  Update resources by B_i where $vl_i = \min(vl_1, vl_2)$
  Return Yes_adjustment

4.2.2 LPT Search Route

LPT algorithm is an approximate algorithm. Applying in Algorithm 1, it can play a role of search path. The LPT route is shown in Algorithm 3.

Algorithm 3: LPT search route

Input Tasks of two Resources
for *tasks from largest to smallest* **do**
  Put the task in the smaller resource and update the value of the selected resource += value of the task

4.2.3 BFD Search Route

Similar to LPT, BFD can also act as the search path. Its algorithm when used as search path in load balancing and bin-packing problems can be seen in Algorithm 4.

Algorithm 4: BFD search route

Input Tasks and Resources
set $\gamma = average = \frac{\sum E_{ij}}{M}$ for balancing problem, or set $\gamma = MaxL^{CPU}$ for bin packing problem.
for *i in tasks from largest to smallest* **do**
  Put the task in the resource α **s.t.**
  $\arg\min_{\alpha}\left|A_\alpha + E_{ij} - \gamma\right|$ for balancing problem, or
  $\arg\min_{\alpha}\left(A_\alpha + E_{ij} - \gamma\right)$ where $\left(A_\alpha + E_{ij} - \gamma\right) \le 0$ for ban packing problem.
  Update resource α

4.3 Combination of Multi-routes

Local Search Algorithm can be enhanced through the combinations of these search paths to construct different multi-route local search algorithms. Other strategies using the three routes to gain optimization solutions include LPTO+BFDO, BFDO+LPTO, and LPT-BFD-One.

(1) LPTO: Adjust tasks $(TS_j, TS_k) \overrightarrow{_{LPTS}} (TS'_j, TS'_k)$ and $(TS'_j, TS'_k) \overrightarrow{_{OneS}} (TS''_j, TS''_k)$; use (TS''_j, TS''_k) as the LPT-One-based neighbor of (TS_j, TS_k).

(2) BFDO: Adjust tasks $(TS_j, TS_k) \underset{BFDS}{\rightarrow} (TS_j^{'}, TS_k^{'})$ and $(TS_j^{'}, TS_k^{'}) \underset{Ones}{\rightarrow} (TS_j^{''}, TS_k^{''})$; use $(TS_j^{''}, TS_k^{''})$ as the BFD-One-based neighbor of (TS_j, TS_k).

(3) LPTO+BFDO: Use the output of LPT-One as the initial input of BFD-One.

(4) BFDO+LPTO: Use the output of BFD-One as the initial input of LPT-One.

(5) LPT-BFD-One: Simultaneously use LPT path, BFD path and OneStep path to update solutions.

5 Experimental Evaluation

In this section, we execute experiments by comparing them with several baselines to demonstrate the superiorities of the proposed algorithms in the scenarios of load balancing and bin-packing. For problems of load balancing and bin-packing in experiments, we simulate that tasks occupy CPU in a resource simultaneously and the CPU occupation L_j^{CPU} is the sum of V_{ij}^{CPU} of all tasks in the corresponding resource. In problems of load balancing, the numbers of resources are fixed. Additionally in bin-packing, the load constraint is $L_j^{CPU} \le 300$ (unit of CPU capacity) and the number of resources is variable.

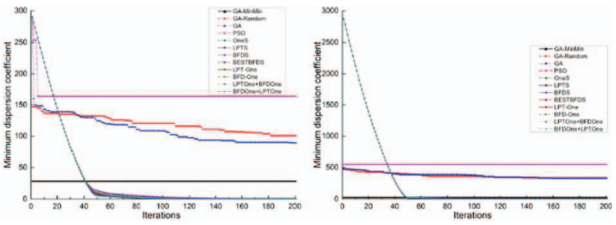
For comprehensive observation, diverse indexes are exhibited to evaluate algorithms' performances, which contain average standard deviation, the total load, Pareto scatter, the probabilities of minimum standard deviation, iterative process of minimum dispersion coefficient.

5.1 Experiment Setup

The comparison algorithms include genetic algorithm, ant colony optimization, particle swarm optimization, FF, FFD, etc. The parameters of tasks are randomly generated.

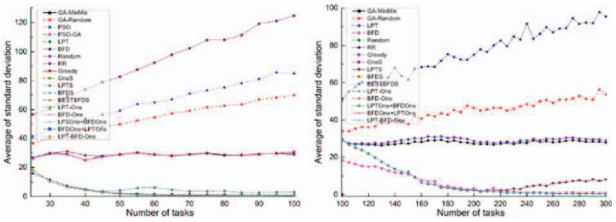
5.2 Result and Discussion

5.2.1 Problem of Load Balancing



(a) $M = 100, N = 1000$ (b) $M = 100, N = 10000$

Figure 1 Iterative processes of dispersion coefficient

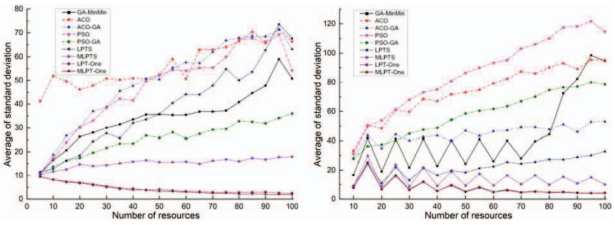


(a) $M = 20, N \in [25,100]$ (b) $M = 100, N \in [100,300]$

Figure 2 The average standard deviations

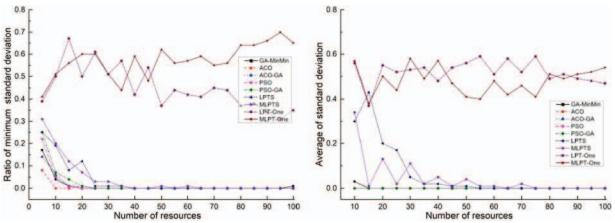
For this case, first group of experiments as Figure 1 for homogenous resources is the iterative processes of dispersion coefficient for each algorithm with property of searching. We use convergence values to replenish iterations. From Figure 1, the proposed algorithms can achieve a convergence state whose dispersion coefficients are significantly close to 0, which the meta-heuristic algorithms can hardly achieve through a few iterations.

Second group of experiments for homogenous resources is to observe the average standard deviation as Figure 2. Each point of Figure 2 presents the average results of 100 cases originated from random generation by simulation environment, in which $V_{ij}^{CPU} \in [1,100]$ (unit of CPU capacity). In Figure 2, the proposed algorithms have demonstrated better performance than other algorithms especially LPT-One, BFD-One, LPTO+BFDO, BFDO+LPTO and LPT-BFD-One have higher ratios to gain the minimum standard deviation in load balancing problem.



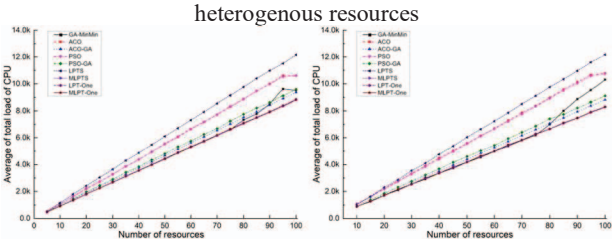
(a) $M = 5, N \in [5,100]$ (b) $M = 10, N \in [10,100]$

Figure 3 The average standard deviations for heterogeneous resources



(a) $M = 5, N \in [5,100]$ (b) $M = 10, N \in [10,100]$

Figure 4 The ratio of minimum standard deviation for heterogenous resources

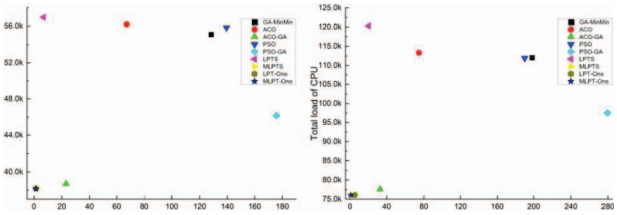


(a) $M = 5, N \in [5,100]$ (b) $M = 10, N \in [10,100]$

Figure 5 The average of total load for heterogenous resources

For heterogeneous resources, we also choose meta-heuristic algorithms and hybrid algorithms as baselines to observe the performances of the proposed algorithms. In experiments, the CPU capacity occupied by each task on any resource is a randomly generated in [75,100] (unit of CPU capacity) and each point also present the average standard deviation of 100 cases. Afterwards, the average of standard deviation is as Figure 3, the probability achieving the minimum

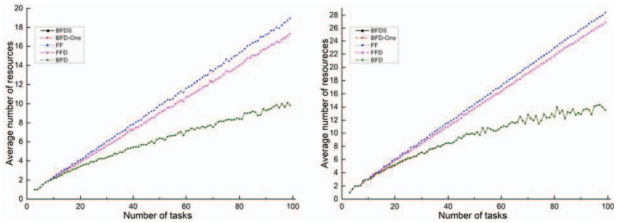
standard deviation of each algorithm as Figure 4 and the average of total load of CPU as Figure 5 severally. In these results, MLPT-One obtains lower average standard deviations, higher probabilities of minimum standard deviation and lower total load of CPU obviously, which validates the advantages of multi-routes to address dual-objectives, i.e., minimizing standard deviation and minimizing total load. Then, we plot the Perato Scatter in two cases as Figure6. From Figure6, MLPT-One obtain the best Perato solutions.



(a) $M=100, N=500$ (b) $M=100, N=1000$

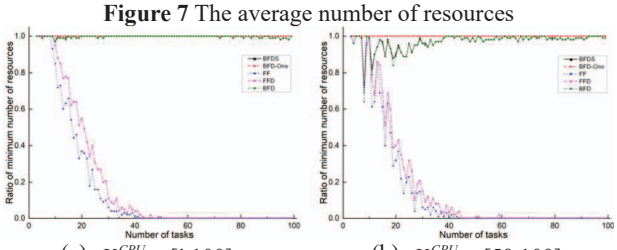
Figure 6 Pareto scatter of CPU load for heterogeneous resources

5.2.2 Problem of Bin-packing



(a) $V_{ij}^{CPU} \in [1,100]$ (b) $V_{ij}^{CPU} \in [50,100]$

Figure 7 The average number of resources



(a) $V_{ij}^{CPU} \in [1,100]$ (b) $V_{ij}^{CPU} \in [50,100]$

Figure 8 The probability of achieving the minimum number of resources

For the problem of minimizing packing number of homogenous resources, we set $MaxL_j^{CPU}=300$ (unit of CPU capacity) and carry out multi group experiments with various ranges of parameters that $V_{ij}^{CPU} \in [1,100]$ and $V_{ij}^{CPU} \in [50,100]$. We mainly demonstrate the average of packing number in Figure 7 and the probability of achieving the minimum number of resources in Figure 8. The curves of BFDS, BFD-One and BFD are almost coincident and better than FF and FFD in Figure 7, while in Figure 8, BFD-One has higher and more stable probabilities to achieve the minimum number of resources than BFD and BFDS.

6 Conclusion

This paper is based on our previous study of multi-route local search algorithm. We leverage LPTO, BFDO and their variants in solving load balancing and bin-packing problems. Abundant simulation experiments have demonstrated the superiorities of our algorithms for problems of load balancing and bin-packing for homogenous resources, as well as the problem of load balancing for heterogenous resources. This demonstrates our previously proposed algorithms are feasible to not only minimizing makespan but also load balancing and bin-packing problems.

Along this research direction, A future direction is to explore more application scenarios and problems. The theoretical exploration of algorithms is also meaningful and necessary.

Acknowledgements

The funds of this paper include 61672136 and 61828202 of National Natural Science Foundation of China.

References

[1] M. Adhikari, T. Amgoth, S. N. Srirama, A survey on scheduling strategies for workflows in cloud environment and emerging trends, ACM Comput. Surv. 52 (4) (2019) 68:1–68:36.

[2] L. Zhang, L. Zhou, A. Salah, Efficient scientific workflow scheduling for deadline-constrained parallel tasks in cloud computing environments, Inf. Sci. 531 (2020) 31–46.

[3] P. Cong, G. Xu, T. Wei, K. Li, A survey of profit optimization techniques for cloud providers, ACM Comput. Surv. 53 (2) (2020) 26:1–26:35.

[4] G. Zhou, W. Tian, R. Buyya, Multi-search-routes-based methods for minimizing makespan of homogeneous and heterogeneous resources in cloud computing, Future Gener. Comput. Syst. 141 (2023) 414–432.

[5] S. Guo, J. Liu, Y. Yang, B. Xiao, Z. Li, Energy-efficient dynamic computation offloading and cooperative task scheduling in mobile cloud computing, IEEE Trans. Mob. Comput. 18 (2) (2019) 319–333.

[6] S. C. A, C. Sudhakar, T. Ramesh, Energy efficient VM scheduling and routing in multi-tenant cloud data center, Sustain. Comput. Informatics Syst. 22 (2019) 139–151.

[7] J. Mei, K. Li, Z. Tong, Q. Li, K. Li, Profit maximization for cloud brokers in cloud computing, IEEE Trans. Parallel Distributed Syst. 30 (1) (2019) 190–203.

[8] C. Bitsakos, I. Konstantinou, N. Koziris, DERP: A deep reinforcement learning cloud system for elastic resource provisioning, in: 2018 IEEE International Conference on Cloud Computing Technology and Science, CloudCom 2018, Nicosia, Cyprus, December 10-13, 2018, IEEE Computer Society, 2018, pp. 21–29.

[9] S. S. Haytamy, F. A. Omara, A deep learning based framework for optimizing cloud consumer qos-based service composition, Computing 102 (5) (2020) 1117–1137.

[10] D. Ding, X. Fan, Y. Zhao, K. Kang, Q. Yin, J. Zeng, Q-learning based dynamic task scheduling for energy-efficient cloud computing, Future Gener. Comput. Syst. 108 (2020) 361–371.

[11] S. M. R. Nouri, H. Li, S. Venugopal, W. Guo, M. He, W. Tian, Autonomic decentralized elasticity based on a reinforcement learning controller for cloud applications, Future Gener. Comput. Syst. 94 (2019) 765–780.

[12] W. Guo, W. Tian, Y. Ye, L. Xu, K. Wu, Cloud resource scheduling with deep reinforcement learning and imitation learning, IEEE Internet Things J. 8 (5) (2021) 3576–3586.

[13] Z. Tong, H. Chen, X. Deng, K. Li, K. Li, A scheduling scheme in the cloud computing environment using deep Q-learning, Inf. Sci. 512 (2020) 1170–1191.

[14] C. Li, Y. Zhang, Y. Luo, Neighborhood search-based job scheduling for iot big data real-time processing in distributed edge-cloud computing environment, J. Supercomput. 77 (2) (2021) 1853–1878.

[15] K. R. P. Kumar, K. Kousalya, Amelioration of task scheduling in cloud computing using crow search algorithm, Neural Comput. Appl. 32 (10) (2020) 5901–5907.

[16] M. Diallo, A. Quintero, S. Pierre, A tabu search approach for a virtual networks splitting strategy across multiple cloud providers, Int. J. Metaheuristics 7 (3) (2020) 197–238.

[17] A. M. S. Kumar, M. Venkatesan, Multi-objective task scheduling using hybrid genetic-ant colony optimization algorithm in cloud environment, Wirel. Pers. Commun. 107 (4) (2019) 1835–1848.

[18] Y. Yang, B. Yang, S. Wang, F. Liu, Y. Wang, X. Shu, A dynamic ant-colony genetic algorithm for cloud service composition optimization, The International Journal of Advanced Manufacturing Technology 102 (1-4) (2019) 355–368.

[19] M. M, J. T, Combined particle swarm optimization and ant colony system for energy efficient cloud data centers, Concurr. Comput. Pract. Exp.33 (10) (2021).

[20] A. Ragmani, A. Elomri, N. Abghour, K. Moussaid, M. Rida, FACO: a hybrid fuzzy ant colony optimization algorithm for virtual machine scheduling in high-performance cloud computing, J. Ambient Intell. Humaniz.Comput. 11 (10) (2020) 3975–3987.