IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 18, 2019, accepted February 6, 2019, date of publication February 18, 2019, date of current version August 14, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2899926

Off-Line Time Aware Scheduling of Bag-of-Tasks on Heterogeneous Distributed System

HEJUN XUAN[1], SHIWEI WEI[2], YANLING LI[1,3], (Member, IEEE), AND HUAPING GUO[1,3]

[1]School of Computer and Information Technology, Xinyang Normal University, Xinyang 464000, China
[2]School of Computer and Technology, Guilin University of Aerospace Technology, Guilin 541000, China
[3]Henan Key Laboratory of Analysis and Application of Education Big Data, Xinyang Normal University, Xinyang 464000, China

Corresponding author: Hejun Xuan (xuanhejun0896@126.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 61572417, in part by the Science and Technology Department of Henan Province under Grant 182102210132 and Grant 182102210537, in part by the Innovation Team Support Plan of University Science and Technology of Henan Province under Grant 19IRTSTHN014, in part by the Guangxi Natural Science Foundation of China under Grant 2016GXNSFAA380226, in part by the Guangxi Young and Middle-aged Teachers' Basic Ability Improvement Foundation of China under Grant 2017KY0866, in part by the Internet of Things and Big Data Application Research Foundation of Guilin University of Aerospace Technology under Grant KJPT201809, and in part by the Nanhu Scholars Program for Young Scholars of XYNU.

ABSTRACT The resource allocation for bag-of-tasks in the heterogeneous distributed system is to distribute the tasks to proper processors such that the makespan is minimized. It is a well-known NP-hard problem, and is even more complex and challenging when the processors have off-line time. To tackle this challenging problem, first, we set up a mathematical model for this problem which minimizes the makespan of the bag-of-tasks with the off-line time segment of the processors. Second, to solve the model efficiently, we propose two new algorithms: a new scheduling algorithm referred to as sorting-allocation-pulling scheduling algorithm which first allocate the tasks to available time segment on proper processors and then pulls them to the formerly available time segment for the sake of minimizing the makespan, and an effective genetic algorithm with a novel local search operator and a well-designed modify operator. Finally, the numerical simulation experiments are conducted, and the two proposed algorithms are compared. The experimental results indicate the effectiveness of the proposed model and algorithms.

INDEX TERMS Distributed computing, task scheduling, bag-of-tasks, generic algorithm.

I. INTRODUCTION

Heterogeneous distributed system has emerged as a commonly system for handling large scale scientific and commercial problems in various fields, such as image processing, signal processing, pattern matching in text, and so on [2], [9], [12]–[14]. For the sake of improving the performance of the system, many task scheduling algorithms for heterogeneous or homogeneous distributed system have been proposed in the past decades, e.g., [30], [33], [41], [44]. In order to obtain high performance and fast scheduling, Wang *et al.* [41] proposed a multi-objective bi-level programming model for energy and locality aware multi-job scheduling in heterogeneous system. Sajid *et al.* developed two scheduling algorithms to schedule a BoT (bag-of-tasks, BoT) on heterogeneous system so as to minimize the makespan and the energy consumption [33]. A representative algorithm is proposed to ensure timing correctness and minimize energy consumption on processors with variable speeds [25]. A reliability cost, which is defined as the product of failure rate of processors and task processing time, is incorporated into scheduling algorithm for the tasks with precedence constraints on heterogeneous system [37]. Lee and Zomaya [23] classified the tasks into computation-intensive and data-intensive BoT and presented two task scheduling algorithms in Grid computing system respectively. Anglano *et al.* [3] proposed a scheduling algorithm with fault-aware strategy for BoT scheduling on desktop Grids. Legrand and Touati [24] analyzed the behavior of multiple noncooperative controllers for handling BoT scheduling problem. Anglano *et al.* [4] evaluated the performance of five knowledge-free task scheduling algorithms for scheduling multiple BoT in a desktop Grids computing system. In addition, the performance of several BoT scheduling solutions in large-scale distributed systems also has been studied [18]. For BoT, there are some other studies that aim to maximize throughput by establishing

The associate editor coordinating the review of this manuscript and approving it for publication was Aniruddha Datta.

linear programming or nonlinear programming [5], [6], and the works were focused on steady-state optimization problems and concentrated on numerous tasks including independent and similar tasks. Legrand *et al.* [5] researched a centralized and decentralized scheduling algorithm for task scheduling in heterogeneous system. To schedule concurrent BoT, Benoit *et al.* [6] developed the online and off-line scheduling algorithms. Celaya and Arronategui [7] designed a decentralized scheduling algorithm to minimize the maximum stretch among user-submitted tasks. Yang *et al.* [45] took the constraints of time, cost, and security into consideration, and designed a scheduling algorithm for data-intensive tasks. Oxley *et al.* [29] investigated both two problems: optimizing the makespan of the tasks under the constraints of energy, or minimizing energy consumption subject to makespan. However, it only studied the static resource allocation problem to optimize makespan and energy robustness for bag-of-tasks (BoT) on a heterogeneous computing system. Zhang *et al.* [47] established a multi-objective optimization model which minimizes makespan and resource cost. To solve the optimization model, a scheduling algorithm based on the ordinal optimization method is designed. However, the scheduling algorithm is inefficient when the task number or processing node number is large. For task scheduling models and algorithms, in order to make the problem simple, a fundamental assumption is that all processors which take participate in processing tasks are always available for processing tasks [10]. However, it might be unreasonable. For example, processor maintenance and breakdowns are often required or other constraints exits, which may result in the processers off-line for some time. In literature [16], the processor availability is defined as the ratio of the total available time to the total time during a given interval. In the existing works, Some researchers have investigated task scheduling algorithms with processor availability constraints [12], [19], [31], [32], [34]. Adiri *et al.* [1]. investigated the scheduling problem with availability constraints in a single machine system. For minimizing maximum lateness of the n jobs, Sheen *et al.* [36] studied the problem on homogeneous machines under machine availability and eligibility constraints. Kacem *et al.* [20] proposed a branch-and-bound method to solve the single-machine scheduling problem with machine availability constraints, and this work is extended to two-machine permutation flowshop scheduling problem with an availability constraint [42]. However, the basic assumption of this work is that the availability constraint is imposed only on the first machine. Vahedi-Nouri *et al.* [40] investigated the non-permutation flow-shop scheduling problem with the learning effects and machine availability constraints. To increase the availability and minimize the makespan of tasks, Zhao *et al.* [38] established an availability-aware scheduling model in heterogeneous systems and designed an optimization algorithm. As an extension work, Zhao *et al.* [39] developed a Hybrid Heuristic-Ant Colony Optimization (H2ACO) for multiclass tasks on heterogeneous distributed systems with availability constraint. H2ACO can make a good trade-off between availability and makespans of the tasks. Yuan et al. [46] proposed a quantum-behaved particle swarm optimization algorithm to optimize the availability-aware task Scheduling on heterogeneous systems. In heterogeneous wireless networks, Zhou *et al.* [49] designed a novel distributed availability-aware adaptive rate-allocation scheduling algorithm for multimedia tasks.

In this paper, we investigate an off-line time aware scheduling problem of bag-of-tasks (BoT) on heterogeneous distributed system. Different from the previous works, we does not define the availability as value. Since shutdown or maintenance often exits, there is off-line time in which processors cannot process the tasks, and the off-line time is known in advance. So, the time of a processor can be divided into off-line time segment and available time segment. The tasks can only be allocated to the available time segment and cannot be divided into several parts. In addition, the requirement for a task similar to that in literature [17] is taken into consideration. That is to say, for a specific task, it can only be processed in some specific processors. In our work, we investigate a resource allocation problem for bag-of-tasks in heterogeneous distributed system when the processors have off-line time. The major contributions of this study are summarized as follows:

- To minimize makespan of the tasks, we establish an optimization model which takes off-line time constraint of processors into consideration.
- A new scheduling algorithm referred to as sorting-allocation-pulling (SAP) scheduling algorithm is proposed, which first allocates the tasks in available time segment on proper processors and then pulls them to a suitable available time segment for the sake of minimizing the makespan.
- To solve the optimization model effectively, we design an effective generic algorithm with a novel local search and tailor-made operators.
- An analysis on the effectiveness of the two proposed algorithms on two different sized systems with different numbers of processors and tasks.

The rest of this paper is organized as follows. Section II gives the system and task description, and establishes the optimization model. The new scheduling algorithm of SAP is described in section III. To solve the optimization model effectively, we propose a generic algorithm with a novel local search and tailor-made operators in section IV. Section V presents simulation results to evaluate the algorithms. The paper is concluded with a summary and future work in Section VI.

II. PROBLEM FORMULATION

A. SYSTEM AND TASK DESCRIPTION

For the problem considered, the heterogeneous distributed system has $N + 1$ processors including a master processor and N slave processors. P_0 denotes the master processor, and the slave processors are denoted by $P = \{P_1, P_2, \cdots, P_N\}$.

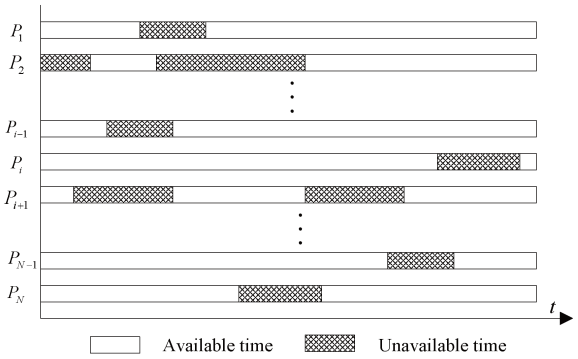


FIGURE 1. Available and off-line time of the processors in the heterogeneous distributed system.

Each slave processor $P_i(i = 1, 2, \ldots, N)$ is associated with a speed index w_i, which is the time taken to process a unit workload on processor P_i. Slave processor is the most basic processing unit in our research. Since some reasons, such as shutdown or termly maintenance requirements, slave processors have some off-line time. $[a_i^j, b_i^j](i = 1, 2, \ldots, N; j = 1, 2, \ldots, n_i)$ denotes the $j^{th}(j = 1, 2, \ldots, n_i)$ off-line time segment of processor $P_i(i = 1, 2, \ldots, N)$, and n_i is the number of off-line time segment for processor $P_i(i = 1, 2, \ldots, N)$. For the convenience of modeling, we use $[c_i^j, d_i^j](i = 1, 2, \ldots, N; j = 1, 2, \ldots, m_i)$ to denote $j^{th}(j = 1, 2, \ldots, n_i)$ available time segment of processor $P_i(i = 1, 2, \ldots, N)$, and m_i is the number of available time segment for processor $P_i(i = 1, 2, \ldots, N)$. To understand easily, the processors' time diagram is shown in Fig.1. As shown in Fig.1, there are some off-line time on each processor. That is to say, we cannot allocate some tasks on the off-line time.

In our study, the bag-of-tasks are investigated. That is to say, all the tasks are independent. The task set includes N_τ independent tasks and the $i^{th}(1 \le i \le N_\tau)$ task is denoted by τ_i. As in the previous work [17], these bag-of-tasks have different computing requirements, and we assume that each task can only be processed by some specific processors. Ω_i is a set of the processors that τ_i can allocated to, and τ_i^σ is the workload of the task. Following the previous studies [11], [22], we assume that the workload of task τ_i^σ is known after a task arrives according to the prediction mechanisms such as code profiling and statistical prediction. Following prior works [27], [32], [43], we assume that the bag-of-tasks are computation-intensive. That is to say, the time consuming of task data transmission does not influence much on the completion time and hence it can be negligible.

B. MATHEMATICAL MODELING

The task scheduling problem investigated in this paper is to schedule all the N_τ tasks to the proper available time segments of N processors in the heterogeneous distributed system with the purpose of minimizing the makespan of the tasks. Then, we will give the optimization modeling of the problem.

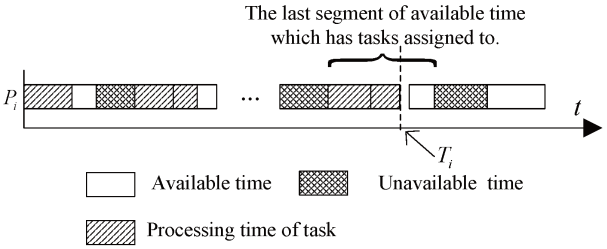


FIGURE 2. The processing time diagram of processor P_i.

$\Theta = (\theta_{ij}^k)_{N_\tau \times N \times m_j}$ is a binary matrix, where $\theta_{ij}^k = 1$ if and only if task τ_i is assigned to the k^{th} available time segment of processor P_j, otherwise $\theta_{ij}^k = 0$. Makespan is the latest processing finish time of processor of all the processors. If T_i denotes the processing finish time of processor P_i, the makespan T of the tasks can be denoted by

$$T = \max_{1 \le i \le N} \{T_i\} \tag{1}$$

From formula (1), we can see that the processing finish time of each processor should be calculated. For a specific processor P_i, its processing time diagram is shown in Fig.2.

In the processing time diagram of processor P_i, it shows that the processing finish time of processor P_i is determined by the last available time segment which has tasks assigned to. If l_i denote the last available time segment which has tasks assigned of processor P_i, and $\delta_i^{l_i}$ is a set of tasks assigned to l_i^{th} available time segment respectively, the processing finish time of processor P_i can be calculated by

$$T_i = c_i^{l_i} + w_i \sum_{q \in \delta_i^l} \tau_q^\sigma \tag{2}$$

Then, we can rewrite formula (1), as

$$\begin{aligned} T &= \max_{1 \le i \le N} \{T_i\} \\ &= \max_{1 \le i \le N} \left\{ c_i^{l_i} + w_i \sum_{q \in \delta_i^{l_i}} \tau_q^\sigma \right\} \end{aligned} \tag{3}$$

Since each processor $P_i(1 \le i \le N)$ has off-line time, the processing time of tasks that assigned to s-th segment should not be greater than the available time. If the task set assigned to s-th $(1 \le s \le m_i)$ available time segment on processor $P_i(1 \le i \le N)$ is denoted by δ_i^s, the following constraint should be satisfied

$$w_i \sum_{q \in \delta_i^s} \tau_q^\sigma \le d_i^s - c_i^s \tag{4}$$

The objective of scheduling bag-of-tasks in heterogeneous systems is to minimize the makespan T. It can be seen from Eq.(3) that the processing finish time of processor P_i only depends on the processing time of the last available time segment which has tasks assigned to. Thus the bag-of-tasks scheduling problem in heterogeneous distributed systems can

be modeled as determining the tasks distributed to processors so that the makespan is minimized, i.e. an optimization model for scheduling bag-of-tasks with the processors have off-line time considered in heterogeneous distributed systems can be set up as

$$\begin{cases} \min T = \min \left\{ \max\limits_{1\le i\le N} \left\{ c_i^{l_i} + w_i \sum\limits_{q\in\delta_i^{l_i}} \tau_q^{\sigma} \right\} \right\} \\ s.t.\ (a) \sum\limits_{i=1}^{N_\tau} \sum\limits_{j=1}^{N} \sum\limits_{k=1}^{m_j} \theta_{ij}^k = N_\tau; \\ \quad (b) \sum\limits_{j=1}^{N} \sum\limits_{k=1}^{m_j} \theta_{ij}^k = 1; \\ \quad (c)\, w_i \sum\limits_{q\in\delta_i^s} \tau_q^{\sigma} \le d_i^s - c_i^s; \\ \quad (d) \sum\limits_{P_j\in\Omega_i} \theta_{ij}^k \times \left(1 - \sum\limits_{P_j\in P\setminus\Omega_i} \theta_{ij}^k\right) = 1; \end{cases} \tag{5}$$

Constraint (a) expresses that all the tasks should be distributed to the processors and completed. Constraint (b) presents that a task can only assigned to one processor. Constraint (c) demonstrates that the processing time of the tasks assigned to the available time segment should be less or equal to the available time. Constraint (d) ensures that the task should be assigned to the processor which can process the task. To solve this global optimization model, an algorithm referred to as sorting-allocation-pulling (SAP) scheduling algorithm and generic algorithm with a local research strategy are proposed. The algorithm referred to as sorting-allocation-pulling (SAP) scheduling algorithm will be described in section III, and the generic algorithm proposed will be given in section IV.

III. PROPOSED SAP ALGORITHM

A. THE ALGORITHM FRAMEWORK

In this section, we consider the task scheduling problem with off-line time of the processors and propose the SAP algorithm. For the sake of understanding the algorithm macroscopically, the framework of the scheduling algorithm is presented before presenting the detailed steps of the algorithm. The sorting-allocation-pulling (SAP) scheduling algorithm is shown in Algorithm 1. Step 1 is to sort all the tasks $\tau_i(i = 1, \cdots, N_\tau)$ in an descending order according to the workload $\tau_i^{\sigma}(i = 1, \cdots, N_\tau)$ of tasks; Step 3 to step 8 allocate the tasks to available time segments on processors. For minimizing the makespan, step 10 to step 25 are used to pull the task to a available time segment before the segment which allocated to.

Algorithm 1 The algorithm framework of SAP

Input: Tasks $\tau_i(i = 1, \cdots, N_\tau)$, speed index w_i and available time segment $[c_i^j, d_i^j](j = 1, \cdots, m_i)$ of processor $P_i(i = 1, \cdots, N)$;
Output: a schedule scheme;

1 Put all the tasks into a task queue TQ, and sort them in an descending order according to the workload τ_i^{σ} of tasks $\tau_i(i = 1, \cdots, N_\tau)$;
2 **Allocation:**
3 **while** *TQ is not empty* **do**
4 Take out the first task in TQ, and denote it as τ_{head};
5 Select a processor P_{head} in Ω_{head} to make the current processing finish time is minimum;
6 Select an available time segment $[c_{head}^s, d_{head}^s]$ on processor P_{head} and assign task τ_{head} to it.
7 Update available time, $c_{head}^s = c_{head}^s + \tau_{head}^{\sigma} w_{head}$;
8 **end**
9 **Pull:**
10 **for** $i = 1$ *to* N **do**
11 **while** $l > 1$ **do**
12 $\%l$ is the last segment of available time segment which has tasks allocated to.
13 Put all the tasks which allocate to l^{th} segment into a task queue $subTQ_l$, and sort them in an descending order according to the workloads;
14 **while** *subTQ$_i$ is not empty* **do**
15 Take out the first task and denote it as $\tau_{subhead}, flag = 0, k = 1$;
16 **while** $k < l$ *and* $flag = 0$ **do**
17 **if** $\tau_{subhead}^{\sigma} w_i \le d_i^k - c_i^k$ **then**
18 $c_i^k = c_i^k + \tau_{subhead}^{\sigma} w_i$;
19 $flag = 1$;
20 **end**
21 $k = k + 1$;
22 **end**
23 **end**
24 $l = l - 1$;
25 **end**
26 **end**

B. DESCENDING ORDER

In the algorithm of SAP, we first sort the tasks in descending order according to the workloads of tasks. We will explain why we choose descending order as follows:

Case 1: As shown in Fig.3, suppose two tasks τ_i and τ_j are all allocated to processor P_k, and the workloads of tasks τ_i and τ_j satisfy $\tau_i^{\sigma} < \tau_j^{\sigma}$. In addition, both tasks τ_i and τ_j can executed in the first available time segment on processor P_k respectively, but τ_i and τ_j cannot executed in the first available time segment simultaneously. As shown in Fig.3(b), if τ_i executed before τ_j, we should allocate τ_i to the first available time segment and allocate the τ_j to the second segment. So, the makespan of the two tasks is $T_k^1 = c_k^2 + \tau_j^{\sigma} w_k$. However, if τ_j is executed before τ_i as shown in Fig3(c), the makespan of the two tasks is $T_k^2 = c_k^2 + \tau_i^{\sigma} w_k$. We have $\tau_i^{\sigma} < \tau_j^{\sigma}$, so $T_k^1 > T_k^2$.

Case 2: As shown in Fig.3(d) and Fig.3(e), tasks τ_i and τ_j can be executed in the first available time segment on processor P_k simultaneous. If τ_i executed before τ_j as shown in Fig.3(d), the makespan of the two tasks is

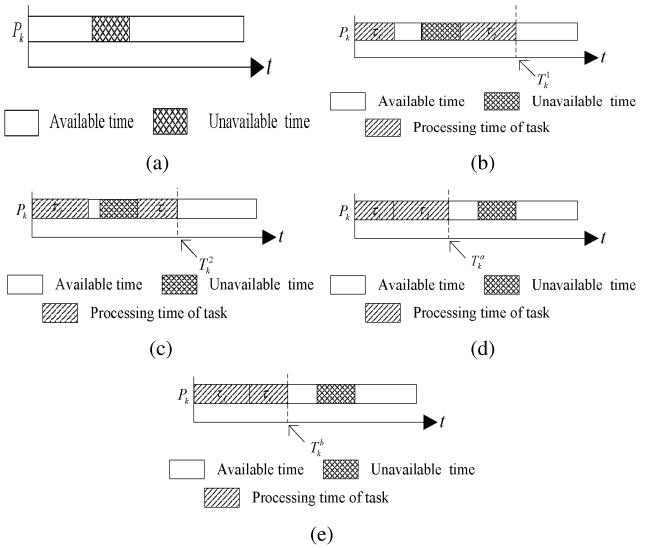


FIGURE 3. Influence of execute order on Makespan. (a) Time diagram of P_k. (b) τ_i executed before τ_j. (c) τ_j executed before τ_i. (d) τ_i executed before τ_j. (e) τ_j executed before τ_i.

$T_k^a = c_k^1 + (\tau_i^\sigma + \tau_j^\sigma)w_k$. Similarly, the makespan of the two tasks is $T_k^b = c_k^1 + (\tau_i^\sigma + \tau_j^\sigma)w_k$ when τ_j executed before τ_i as shown in Fig.3(e). So, we can obtain $T_k^a = T_k^b$.

From the above discussion, we can know that allocation order will effect the makespan of the tasks. If the larger task is allocated first, the makespan of tasks will equal to or shorter than that obtained by smaller workload task being allocated first.

C. PROCESSOR SELECTION IN ALLOCATION

When the task queue TQ is not empty, the first task in TQ is taken out and allocated to the processor. Since the objective is to minimize the makespan of the tasks, so we must allocate the task to a processor that can make the processing finish time is minimum. In our work, the processor that task τ_i should be allocated to is determine by

$$\Phi_i = \left\{P_j | P_j \in \Omega_i, d_j^k - c_j^k \geq \tau_i^\sigma w_j, 1 \leq k \leq m_j\right\} \quad (6)$$

$$np_{proper} = \arg\min_{np} \left\{\max_{np \in \Phi_i} \{T_{np}\}\right\} \quad (7)$$

where P_{np} is the candidate processor in processors set Φ_i that the task τ_i can be allocated to, and $P_{np_{proper}}$ is the proper processor determined. Eq.(6) is to find the set Φ_i that the task τ_i can be allocated to in Ω_i. The processors in Φ_i should satisfy two conditions: (1) the processors should be in the set Ω_i. (2) At least a available time segment, which can execute the task τ_i punctually, exists.

D. SEGMENT SELECTION IN ALLOCATION

After a proper processor determined, we should allocate task τ_i to an optimal available time segment on processor P_{np}. An excellent strategy that allocates task to an available time segment will help to minimize makespan of the tasks. We use Eq.(8) and Eq.(9) to determine which segment task τ_i should be allocated to.

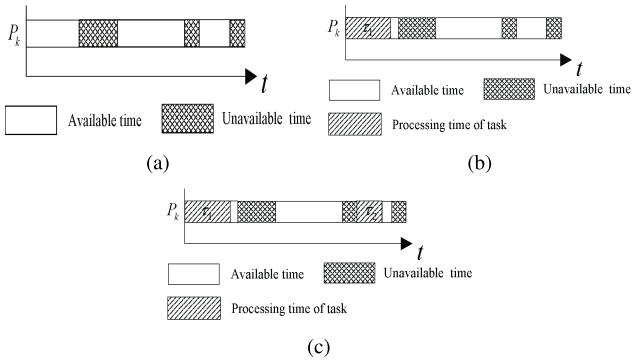


FIGURE 4. Segment determined. (a) Time diagram of P_k. (b) Allocate τ_1 to segment 1. (c) Allocate τ_2 to segment 3.

$$NS = \left\{ns | 1 \leq ns \leq m_{np}, d_{np}^{ns} - c_{np}^{ns} \geq \tau_i^\sigma w_{np}\right\} \quad (8)$$

$$ns_{proper} = \arg\min_{ns} \left\{\left(d_{np}^{ns} - c_{np}^{ns}\right) | ns \in NS\right\} \quad (9)$$

Eq.(8) is used to find some available time segments that can complete task τ_i in time on processor P_{np}. For the sake of decreasing time debris which cannot complete any task in time, the shortest time segment in NS is selected. There are two tasks τ_1 and τ_2 allocated on processor P_k, and $\tau_1^\sigma > \tau_2^\sigma$. τ_1 can completed in segment 1 and 2, and τ_2 can completed in segment 2 and 3. Because the tasks in task queue TQ are sorted in descending order according to workloads, τ_1 is allocated before τ_2. For τ_1, since $d_k^1 - c_k^1 < d_k^2 - c_k^2$, so τ_1 is allocated to segment 1 according to Eq.(9) as shown in Fig.4(b). Though segment 2 can complete τ_2 in time and segment 2 is before segment 3, we can see that τ_2 is allocated to segment 3 from Fig.4(c). This strategy can help to decrease time debris and increase the utilization of the available time segment.

E. THE STRATEGY OF PULLING

After the process of allocation, all the tasks are allocated to the available time segments on processors. However, some available time segments are exit because the strategy that described in section III-D. As shown in Fig.5(a), τ_i is allocated to $(j+1)^{th}$ available time segment of processor P_k, so the processing finish time of P_k is $T_k = c_k^{j+1} + \tau_i^\sigma w_k$. Since the j^{th} available time segment can complete task τ_i in time, we can pull task τ_i from $(j+1)^{th}$ available time segment to j^{th} available time segment as shown in Fig.5(b). So the processing finish time of P_k can be denoted as $T_k' = c_k^j + \tau_i^\sigma w_k$. $T_k' < T_k$ can be obtained intuitively. So, the strategy of pulling tasks to another available time segment can decrease the processing finish time of processors. For the sake of decreasing the processing time of the processors as much as possible, we should solve following two problems:(1)which task should be pulled to the objective segment? (2) which segment should be selected as the objective segment? These two issues will be tackled in section III-E1 and section III-E2.

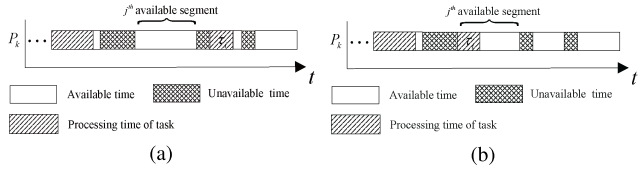


FIGURE 5. Pull tasks to another available time segment. (a) Time diagram of P_k after tasks are allocated. (b) Pull τ_i to j^{th} available time segment.

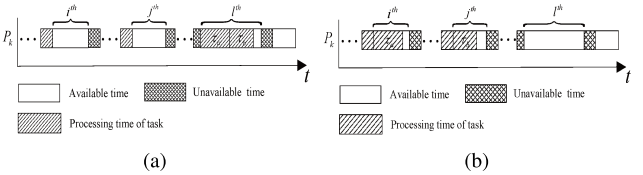


FIGURE 6. An example of pulling tasks to another available time segment. (a) Time diagram of P_k after tasks are allocated. (b) Pull τ_a and τ_b to i^{th}, j^{th} available time segment respectively.

1) SELECTION OF OBJECTIVE TASK

In this paper, we investigate the bag-of-tasks scheduling problem, and the objective of scheduling algorithm is to minimize makespan of the tasks. The makepan of the tasks is determined by the processing finish time of all processors in the heterogeneous computing system. From Eq.(2), we can see that the processing finish time $T_i(1 \le i \le N)$ depends on the tasks completed time in last available time segment l_i which has tasks allocated to. Suppose $SubQ_{l_i}$ is the tasks set which allocate to the last available time segment l. Eq.(10) is used to determine which task should be pulled to another available time segment.

$$nt_{pro} = \arg\max_{nt} \left\{ \tau_i^{\sigma} | \tau_i \in SubQ_{l_i} \right\} \tag{10}$$

2) SELECTION OF OBJECTIVE SEGMENT

To decrease the processing finish time of processors, in this section, we will determine which segment should be selected as the objective segment. For the sake of guaranting the task τ_{nt} completed in time, the segment ns_{pro} is selected according to Eq.(11).

$$ns_{pro} = \arg\min_{ns} \left\{ ns | d_k^{ns} - c_k^{ns} > \tau_{nt_{pro}} w_k \right\} \tag{11}$$

The strategy of pulling task to another segment can decrease processing finish time as much as possible. First, the task $\tau_{nt_{pro}}$ with the largest workload is selected according to Eq.(10), and an available time segment ns_{pro} is selected according to Eq.(11). If $ns_{pro} = \emptyset$, let $SubQ_{l_i} = SubQ_{l_i} \setminus \{\tau_{nt_{pro}}\}$, and then another task $\tau_{nt_{pro}}$ is selected according to Eq.(10). If $ns_{pro} \neq \emptyset$, We pull the task $\tau_{nt_{pro}}$ from segment l to the segment ns_{prop}.

An example is presented in Fig.6. Task τ_a and τ_b are allocated to the l_i^{th} available time segment, and $\tau_a^{\sigma} > \tau_b^{\sigma}$. First, τ_a and τ_b are put into $subQ_{l_i}$. According to Eq.(10), task τ_a is selected as the objective task. The i^{th} segment is selected as the objective segment according to Eq.(11). Then, we pull τ_a to i^{th} segment and update $c_k^i = c_k^i + \tau_a^{\sigma} w_k$. Since $ns_{pro} \neq \emptyset$, we can select task τ_b and j^{th} segment as the objective task and objective segment respectively. Then, task τ_b is pulled to j^{th} segment and update $c_k^j = c_k^j + \tau_b^{\sigma} w_k$. Let $l_i = l_i - 1$, a new round of pulling is conducted until the objective segment cannot be found.

IV. GA FOR BOT SCHEDULING

Task scheduling is an NP hard problem in the well-known hardest combinatorial optimization problems. GAs(Generic algorithms, GAs), which were invented by John Holland [15], have demonstrated their potential in solving many NP hard realistic application problems such as Control and Decision, image processing, and machine learning, etc [8], [21], [26], [35], [48]. In this paper, a GA is designed to solve the task scheduling optimization model proposed in section II.

Algorithm 2 Encoding and Population Initialization

Input: N, N_τ, population size Pop_{size};
Output: Initial population Pop;
for $i = 1$ *to* Pop_{size} **do**
  for $j = 1$ *to* N_τ **do**
    $flag\_allocated = 0$;
    while $flag\_allocated == 0$ **do**
      $p\Omega_j$ is a permutation of the elements in Ω_j;
      A random integer is generated in $[1, size(\Omega_j)]$, denoted as $k1$;
      $Pop(1, j, i) = p\Omega_j(k1)$;
      $flag\_segment = 0$;
      while $flag\_segment == 0$ **do**
        Let $n_p = Pop(1, j, i)$; pM be a permutation of elements $\{1, 2, \cdots, n_p\}$;
        A random integer is generated in $[1, m_{n_p}]$, denoted as $k2$;
        if $c_{n_p}^{k2} + \tau_j^{\sigma} w_{n_p} \le d_{n_p}^{k2}$ **then**
          $flag\_segment = 1$;
          $Pop(2, j, i) = pM(k2)$;
        end
      end
      if $flag\_segment == 1$ **then**
        $flag\_allocated = 1$;
        $c_{n_p}^{k2} = c_{n_p}^{k2} + \tau_j^{\sigma} w_{n_p}$;
      end
    end
  end
end

A. ENCODING AND POPULATION INITIALIZATION

Based on the characteristics of this optimization model for bag-of-tasks scheduling problem, the integer array encoding scheme is adopted. An array $C_{2\times N_\tau} = (c_{ij})_{2\times N_\tau}$ is used to represent a list of $2 \times N_\tau$ elements, called chromosome. For a specific task $\tau_j(1 \le j \le N_\tau)$, we have $c_{1i} = j, c_{2i} = k \Leftrightarrow \theta_{ij}^k = 1$. We can obtain the initial population Pop of generic

Algorithm 3 Mutation Operator

Input: Individual $C = (c_{ij})_{2\times N_\tau}$;
Output: Offspring $C' = (c'_{ij})_{2\times N_\tau}$;
$C' = C$;
Two random integers $i, j(i < j)$ are generated;
for $k = i$ *to* j **do**
  Let $n_p = i + j - k$;
  if $C(1, n_p) \in \Omega_i$ *and* $C(2, n_p) \leq m_{C(1,n_p)}$ **then**
    $C'(1, k) = C(1, n_p)$;
    $C'(2, k) = C(2, n_p)$;
  else
    Calculate np_{best} and ns_{best} by Eq.(12);
    $C'(1, k) = np_{best}$;
    $C'(2, k) = ns_{best}$;
  end
end

allocation according to the algorithm 2. It has an advantage that the individuals in initial population are all the feasible solutions.

B. MUTATION OPERATOR

Suppose that the chromosome $C = (c_{ij})_{2\times N_\tau}$ is chosen to take part in mutation, and the offspring $C' = (c'_{ij})_{2\times N_\tau}$ is obtained by the mutation as shown in algorithm 3.

$$\{np_{best}, ns_{best}\} = \arg\min_{np,ns} \left\{ \max_{1\leq ns\leq m_{np}} \{T_{np}\} \right\} \tag{12}$$

where $P_{np}(np \in \Omega_i)$ are the processors which have tasks allocated to and ns is the available time segment on processor P_{np}. Eq.(12) is used to search the best processor $P_{np_{best}}$ and available time segment ns_{better} on processor $P_{np_{best}}$ for tasks τ_i to make the maximum processing finish time of all processors in Ω_i is minimized.

C. LOCAL SEARCH

In this paper, a local search operator, which can accelerate the convergence and enhance the searching ability of the proposed algorithm, is designed. If the local search operator is applied to the chromosome $C = (c_{ij})_{2\times N_\tau}$, the offspring $C' = (c'_{ij})_{2\times N_\tau}$ is obtained by local search operator as shown in algorithm 4.

$$ns_{better} = \arg\min_{ns} \left\{ \max_{1\leq ns\leq m_{np}} \{T_{np}\} \right\} \tag{13}$$

where P_{np} is the processor that the tasks are allocated to. Eq.(13) is used to search a better available time segment ns_{better} in processor P_{np}.

D. MODIFIED OPERATOR

For the sake of accelerating the convergence of genetic algorithm and minimizing makespan of the tasks, a modified operator is designed. The pseudocode of the modified operator

Algorithm 4 Local Search Operator

Input: Individual $C = (c_{ij})_{2\times N_\tau}$;
Output: Offsprings $C' = (c'_{ij})_{2\times N_\tau}$;
$C' = C$;
for $i = 1$ *to* N_τ **do**
  Calculate ns_{better} by Eq.(13);
  if $ns_{better} \neq C(2, i)$ **then**
    $C(2, i) = ns_{better}$;
  end
end

Algorithm 5 Modified Operator

Input: Individual $C = (c_{ij})_{2\times N_\tau}$;
Output: Offsprings $C' = (c'_{ij})_{2\times N_\tau}$;
$C' = C$;
for $i = 1$ *to* N **do**
  for $j = m_i$ *to* 2 **do**
    Ψ_i^j is the set of tasks allocated to j^{th} available time segment of processor P_i, and the tasks are sorted in descending order according to workloads;
    for $k = size(\Psi_i)$ *to* 1 **do**
      if $\exists p < j$ *satisfy* $\tau^{\sigma}_{\Psi_i^j(k)} w_i < d^p - c^p$ **then**
        $C'(2, \Psi_i(k)) = p$;
      end
    end
  end
end

is shown in algorithm5. A chromosome $C = (c_{ij})_{2\times N_\tau}$ will modified as $C' = (c'_{ij})_{2\times N_\tau}$.

V. EXPERIMENTS AND ANALYSIS

As there is no algorithm available in the literature for scheduling bag of tasks in heterogeneous system with off-line time segment constraints. To demonstrate the effectiveness and efficiency of the proposed algorithms, we compare the proposed two algorithms with other two algorithms, which are proposed in literature [6] (briefly CBS3M_EDF_ROFF)and literature [39](briefly H2ACO). The two compared algorithms can be modified to suit for the problem of this work investigated. Several experiments on these two algorithms are conducted and the results are presented in this section. In section V-A, the parameters used in the algorithms will be given. Experimental results are presented in section V-B and section V-C. Finally, the experimental results are analyzed in section V-D.

A. PARAMETERS VALUE

1) TASKS PARAMETERS

In this paper, we investigate the bag-of-tasks(BoT) scheduling problem in heterogeneous distributed system. In small

simulation system, the workload ranges from 500 to 6000, and the tasks number N_τ varies between 50 and 500. Similarly, in large simulation system, N_τ varies between 1000 and 10000, and the workload ranges from 1000 to 12000. Six groups experiments are conducted in small and large systems, respectively. In addition, for each task $\tau_i(1 \leq i \leq N_\tau)$, a set Ω_i of the processors, which process task τ_i, is generated as follow: a random number n_r is generated in (0, 1], and set the cardinality of Ω_i as $n_{pro} = \lfloor n_r N \rfloor$. Then, n_{pro} processors are selected in P randomly and they form the set Ω_i, where P is the set of all processors.

2) SYSTEM PARAMETERS

For the heterogeneous simulation system, we adopt 20 and 200 heterogeneous slave processors in small simulation system and large simulation system respectively. The time consuming for unit workload $w_i(1 \leq i \leq N)$ of processor $P_i(1 \leq i \leq N)$ in the heterogeneous distributed system is referred to Shang [28]. The available time segment m_i on processor P_i is generated randomly in [5] and [20]. The length of $k^{th}(1 \leq k < m_i)$ available time segment on P_i is generated randomly in $\left[w_i \times mean_{1\leq i\leq N_\tau}\{\tau_i^\sigma\}, 3 \times w_i \times mean_{1\leq i\leq N_\tau}\{\tau_i^\sigma\}\right]$. If $k = m_i$, d_i^k can equal to $+\infty$, that is to say, the processor P_i can process tasks all the time in m_i^{th} available time segment.

3) GENETIC ALGORITHM PARAMETERS

In this paper, GA denotes the algorithm of genetic algorithm without local search operator and modified operator, and genetic algorithm with local search and modified operator denoted as GALM. Similarly, GAL and GAM are indicated as the genetic algorithm with local search and modified operator respectively. In the algorithm of GA, GAL, GAM and GALM, the following parameters are chosen: population size $Pop_{size} = 100$, crossover probability $p_c = 0.8$, mutation probability $p_m = 0.05$, elitist number $E = 5$ and maximum iterations $G_{max} = N_\tau$. Since the concept of neighborhoods of a individual is employed, neighbor size $T = 10$ in our experiments.

B. EXPERIMENTAL RESULTS IN SMALL SIMULATION SYSTEM

To evaluate the effectiveness of the proposed scheduling algorithm, in this sub-section, we present a performance evaluation study in small simulation system. First, we evaluate the makespans of the five algorithms (SAP, GA, GAL, GAM, GALM) and two compared algorithms (CBS3M_EDF_ROFF and H2ACO) for various task numbers (N_τ) and workloads in Fig.7(a) to Fig.7(f).

To evaluate the efficiency of GA, GAL, GAM and GALM, convergence results of the four algorithms are shown in Fig.8(a) to Fig.8(d). In these experiments, a specific number of task $N_\tau = 50$ is selected, and the maximum iterations are set to $G_{max} = 1000$ in every group of experiments.

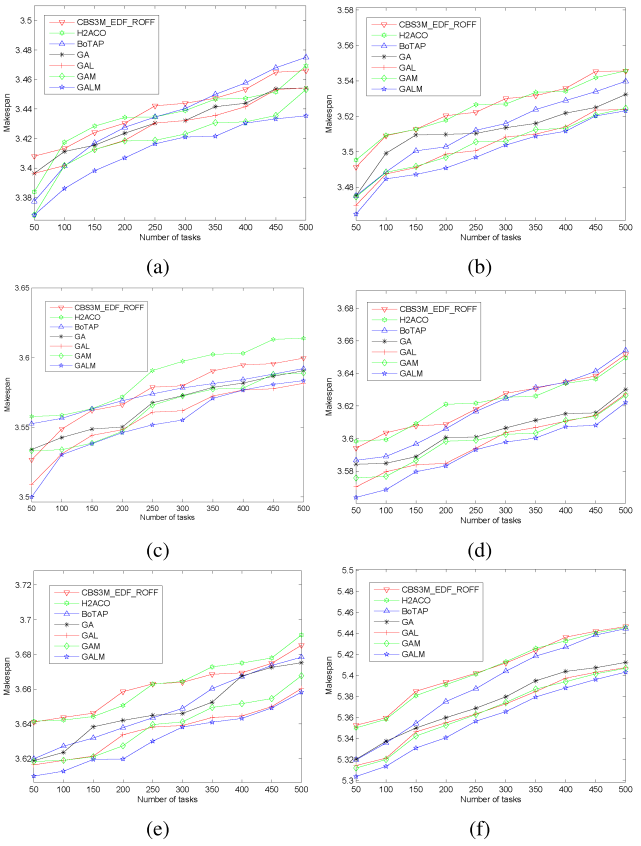


FIGURE 7. Makespans in small simulation system. (a) $500 \leq \tau_i^\sigma \leq 1000$. (b) $500 \leq \tau_i^\sigma \leq 2000$. (c) $500 \leq \tau_i^\sigma \leq 3000$. (d) $500 \leq \tau_i^\sigma \leq 4000$. (e) $500 \leq \tau_i^\sigma \leq 5000$. (f) $500 \leq \tau_i^\sigma \leq 6000$.

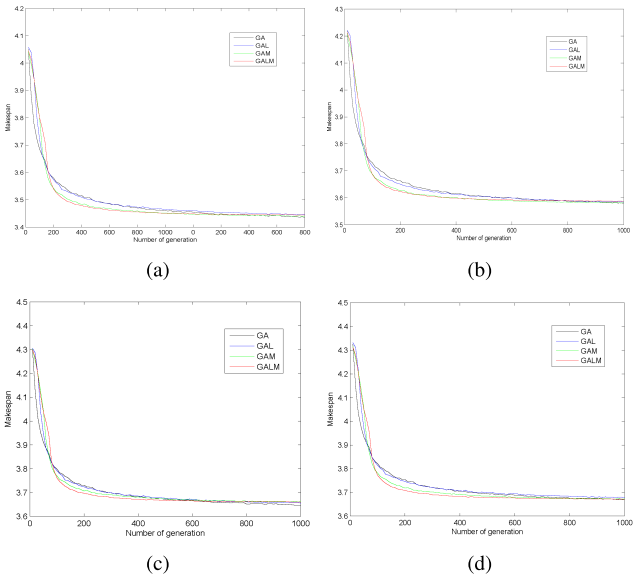


FIGURE 8. Convergence of the four algorithms in small simulation system. (a) $1000 \leq \tau_i^\sigma \leq 2000$. (b) $1000 \leq \tau_i^\sigma \leq 3000$. (c) $1000 \leq \tau_i^\sigma \leq 4000$. (d) $1000 \leq \tau_i^\sigma \leq 5000$.

To evaluate the robustness of the four algorithms (GA, GAL, GAM and GALM), every algorithm is executed

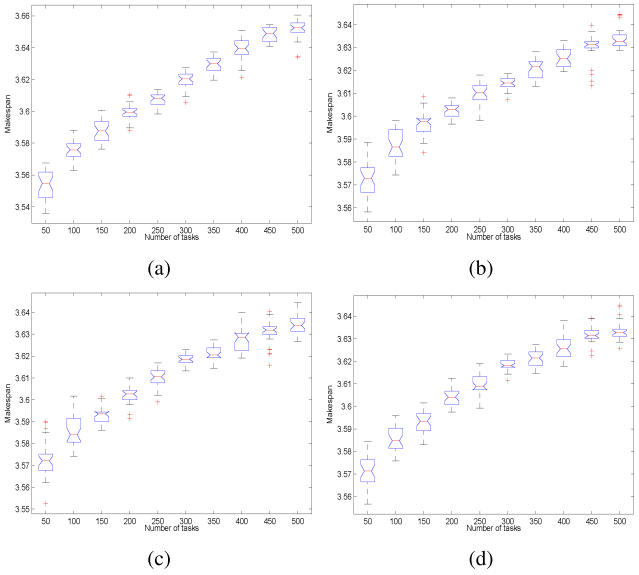


FIGURE 9. Robustness of the four algorithms in small simulation system. (a) The robustness of GA. (b) The robustness of GAL. (c) The robustness of GAM. (d) The robustness of GALM.

30 times independently. In these experiments, the workload is uniformly distributed in [1000 5000], and the number of tasks ranges from 50 to 500. The statistics results of the four algorithms are shown in Fig.12(a) to Fig.12(d) respectively using Box-whisker Plot.

C. EXPERIMENTAL RESULTS IN LARGE SIMULATION SYSTEM

Similar to the small simulation system, makespan, convergence and robustness of the proposed algorithms and two compared algorithms are evaluated with various task numbers (N_τ) and workload. The makespans obtained by the five algorithms in large simulation system are shown in Fig.10(a) to Fig.10(f).

To evaluate the efficiency of GA, GAL, GAM and GALM in large simulation system, convergence results of the four algorithms are shown in Fig.8(a) to Fig.8(d). In these experiments, a specific number of tasks $N_\tau = 1000$ is selected, and the maximum iterations $G_{max} = 10000$ in every group of experiments.

Similar to the small simulation system, robustness of the four algorithms (GA, GAL, GAM and GALM) in large simulation system is evaluated, every algorithm is executed 30 times independently. In these experiments, the workload is uniformly distributed in [1000 10000], and the number of tasks ranges from 1000 to 10000. The statistics results of the four algorithms are shown in Fig.12(a) to Fig.12(d) respectively using Box-whisker Plot.

D. EXPERIMENTAL ANALYSIS

The experiments are conducted in two simulation systems, i.e, the small simulation system and the large simulation system, with various numbers of tasks and workloads.

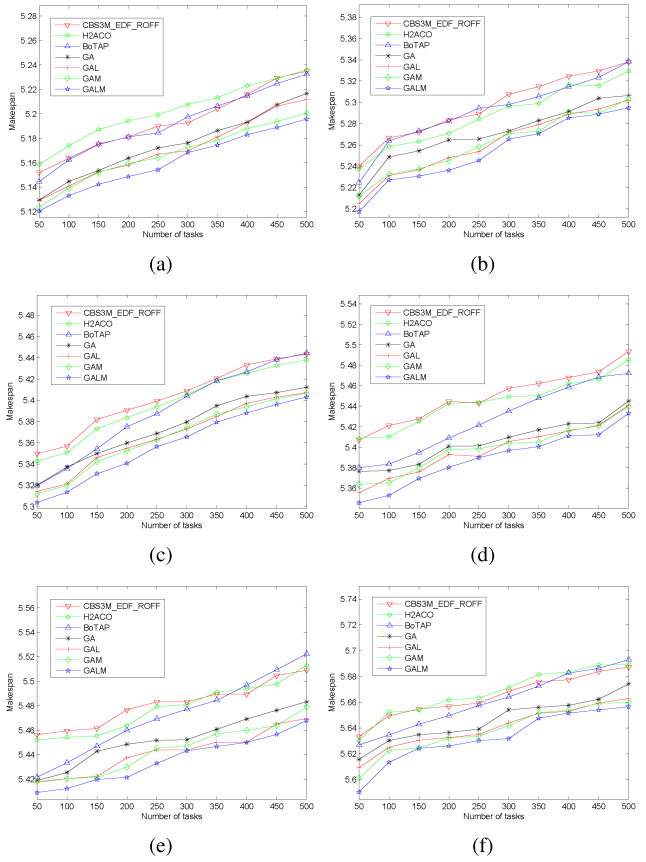


FIGURE 10. Makespans in large simulation system. (a) $1000 \le \tau_i^\sigma \le 2000$. (b) $1000 \le \tau_i^\sigma \le 4000$. (c) $1000 \le \tau_i^\sigma \le 6000$. (d) $1000 \le \tau_i^\sigma \le 8000$. (e) $1000 \le \tau_i^\sigma \le 10000$. (f) $1000 \le \tau_i^\sigma \le 12000$.

The makespans obtained by the proposed algorithms (SAP, GA, GAL, GAM, GALM) and compared algorithms (CBS3M_EDF_ROFF and H2ACO) are shown in Fig.7 and Fig.10. From Fig.7 and Fig.10, we can see that makespan obtained by proposed algorithms are smaller than that obtained by the compared algorithms. In addition, we can see that makespan obtained by SAP is shorter than those by GA, GAL, GAM and GALM when task number is small, and that makespan obtained by SAP is longer than those by other four algorithms when task number is large. Since the information of un-scheduled tasks has not been considered, the procedure of processor determination will result in a bad scheduling strategy in SAP algorithm. What is more, SAP algorithm first the tasks according to their workloads, and the task with a large workload will be scheduled preferentially, which also makes the scheduling algorithm with a low efficiency and easily trapping in a local optimal solution when task number is large. However, GA, GAL, GAM and GALM can obtain a better scheduling strategy according to all the information and the state of the processors. Since local search operator and modified operator are tailor-made, both of them are conducive to increasing the diversity of solutions and searching a local optimal solution in search space. So, GAL and GAM can convergent to a better solution than GA.

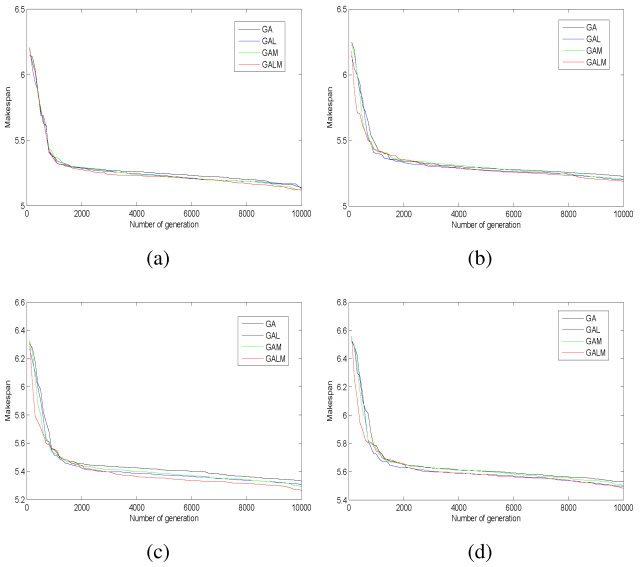


FIGURE 11. Convergence of the four algorithms in large simulation system. (a) $1000 \le \tau_i^\sigma \le 3000$. (b) $1000 \le \tau_i^\sigma \le 6000$. (c) $1000 \le \tau_i^\sigma \le 9000$. (d) $1000 \le \tau_i^\sigma \le 12000$.

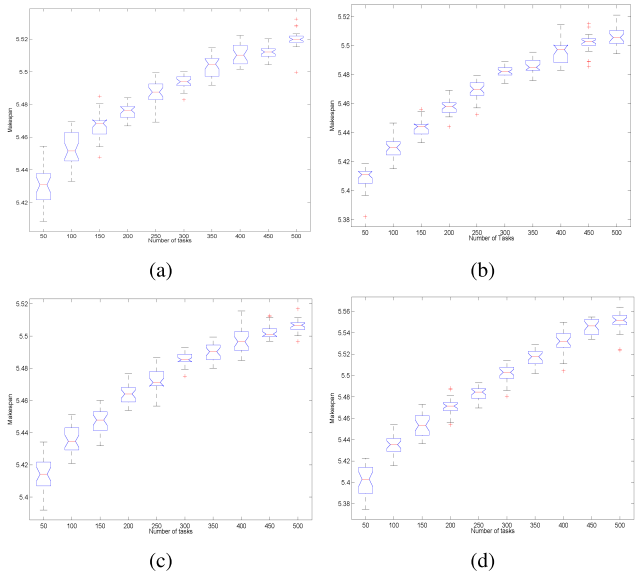


FIGURE 12. Robustness of the four algorithms in large simulation system. (a) The robustness of GA. (b) The robustness of GAL. (c) The robustness of GAM. (d) The robustness of GALM.

That is to say, makespan obtained by GAL and GAM both are shorter than that obtained by GA. GALM is a algorithm that local search and modified operator are added into GA. So, makespan obtained by GALM is shortest. However, GAL is hard to tell from GAM. Because local search operator and modified operator both are search a local optimal solution by changing scheduling scheme of a task. As shown in the Fig.7 and Fig.10, we can see that makespan obtained by GALM is shortest, and makespan obtained by GA is largest among the four algorithms(GA, GAL, GAM and GALM). Makespan obtained by GAL is shorter than that obtained by GAM in some cases. However, the opposite results can be obtained in other cases. In addition, we can see that the different between makespan obtained by SAP and GA are gradual increased with the num of task increased. Similarly, the same conclusion can be obtained in large similar system.

In addition, convergence of proposed four algorithms including GA, GAL, GAM and GALM are investigated in small simulation system and large simulation system. In this paper, we design a local search operator and a modify operator, and the two optimization algorithms referred as GAL and GAM. Local search operator and modified operator are conducive to increasing the diversity of the solutions and searching a local optimal solution in search space. On the one hand, local search operator can generate a better offspring than its parent individual by changing the value of a gene. On the other hand, modified operator can also decrease the processing finish time of a processor as much as possible. For a specific generation, the offsprings obtained by local search operator and modified operator will have better fitness than their parents. So, GAL and GAM can convergent to global optimal solution quickly. That is to say, GAL and GAM have a higher convergent speed than GA. However, we cannot tell good or bad for GAL and GAM. As we can see in the experimental results, GAL is better than GAM in some cases, and GAM is better than GAL in others case. Because local search operator and modified operator are both searching a local optimal solution by changing scheduling scheme of a task. GALM is a algorithm that comprise of GA, local search operator and modified operator. So, it can convergent to a global optimal solution as fast as possible. As shown in the Fig.7 and Fig.10, GALM has a highest convergent speed among the four algorithms (GA, GAL, GAM and GALM), and the convergent speed of GA is lowest.

What is more, the robustness of the four algorithms are investigated in small simulation system and large simulation system. Fig.9(a) to Fig.9(d) give the robustness of GA, GAL, GAM and GALM in small simulation system for various task number. The robustness of the four algorithms in large simulation system for various task number is shown in Fig.12(a) to Fig.12(d). From the figures, we can see that the four algorithms have a high robustness for various task number. With increasing of the tasks and processors, much more local optimal solutions exit. So, in small simulation system, the robustness of the algorithms are higher when the number of task is smaller, and the robustness will be decreased with the number of task increased. Similarly conclusion can be obtained in large simulation system.

VI. CONCLUSION

In this paper, we investigate a bag-of-tasks(BoT) scheduling problem with off-line time considered in heterogeneous distributed system. For the sake of minimizing the makespan of the tasks, we establish a mathematical optimization model with the off-line time constrained. we propose two new algorithms: a new scheduling algorithm referred to as sorting-allocation-pulling(SAP) scheduling algorithm and a genetic

algorithm. In the algorithm of SAP, we first allocate the bag-of-tasks to the available time segment. To reduce the makespan of the tasks, we design a strategy that pulls the tasks to another available time segment. In the genetic, local search operator and modified operator are tailor-made. In the experiments, two simulation systems are employed. Makespan obtained by the five algorithms with various number of tasks and workload are evaluated. In addition, convergence and robustness of GA, GAL, GAM and GALM are evaluated in the two simulation system. Experimental results show that the algorithms proposed are efficient. What is more, the four algorithms of GA, GAL, GAM, GALM can converge to the global optimal solution quickly and have a higher robustness.

REFERENCES

[1] I. Adiri, J. Bruno, E. Frostig, and A. H. G. R. Kan, "Single machine flow-time scheduling with a single breakdown," *Acta Inform.*, vol. 26, no. 7, pp. 679–696, 1989.

[2] M. I. Alghamdi, X. Jiang, J. Zhang, J. Zhang, M. Jiang, and X. Qin, "Towards two-phase scheduling of real-time applications in distributed systems," *J. Netw. Comput. Appl.*, vol. 84, pp. 109–117, Apr. 2017.

[3] C. Anglano, J. Brevik, M. Canonico, D. Nurmi, and R. Wolski, "Fault-aware scheduling for bag-of-tasks applications on desktop grids," in *Proc. 7th IEEE/ACM Int. Conf. Grid Comput.*, Sep. 2006, pp. 56–63.

[4] C. Anglano and M. Canonico, "Scheduling algorithms for multiple bag-of-task applications on desktop grids: A knowledge-free approach," in *Proc. IEEE Int. Symp. Parallel Distrib. Process. (IPDPS)*, Apr. 2008, pp. 1–8.

[5] O. Beaumont, L. Carter, J. Ferrante, A. Legrand, L. Marchal, and Y. Robert, "Centralized versus distributed schedulers for bag-of-tasks applications," *IEEE Trans. Parallel Distrib. Syst.*, vol. 19, no. 5, pp. 698–709, May 2008.

[6] A. Benoit, L. Marchal, J.-F. Pineau, Y. Robert, and F. Vivien, "Scheduling concurrent bag-of-tasks applications on heterogeneous platforms," *IEEE Trans. Comput.*, vol. 59, no. 2, pp. 202–217, Feb. 2010.

[7] J. Celaya and U. Arronategui, "Fair scheduling of bag-of-tasks applications on large-scale platforms," *Future Gener. Comput. Syst.*, vol. 49, pp. 28–44, Aug. 2015.

[8] R. Datta, S. Pradhan, and B. Bhattacharya, "Analysis and design optimization of a robotic gripper using multiobjective genetic algorithm," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 46, no. 1, pp. 16–26, Jan. 2016.

[9] M. D. de Assunção, A. da Silva Veith, and R. Buyya, "Distributed data stream processing and edge computing: A survey on resource elasticity and future directions," *J. Netw. Comput. Appl.*, vol. 103, pp. 1–17, Feb. 2018.

[10] Z. Ding, J. Liu, Y. Sun, C. Jiang, and M. Zhou, "A transaction and QoS-aware service selection approach based on genetic algorithm," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 45, no. 7, pp. 1035–1046, Jul. 2015.

[11] N. Doulamis, E. Varvarigos1, and T. Varvarigou, "Fair scheduling algorithms in grids," *IEEE Trans. Parallel Distrib. Syst.*, vol. 18, no. 11, pp. 1630–1648, Nov. 2007.

[12] M. Guo, Q. Guan, and W. Ke, "Optimal scheduling of VMs in queueing cloud computing systems with a heterogeneous workload," *IEEE Access*, vol. 6, pp. 15178–15191, 2018.

[13] S. Haider and B. Nazir, "Dynamic and adaptive fault tolerant scheduling with QoS consideration in computational grid," *IEEE Access*, vol. 5, pp. 7853–7873, 2017.

[14] H. Han, W. Bao, X. Zhu, X. Feng, and W. Zhou, "Fault-tolerant scheduling for hybrid real-time tasks based on CPB model in cloud," *IEEE Access*, vol. 6, pp. 18616–18629, 2018.

[15] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis With Applications to Biology, Control, and Artificial Intelligence*. Ann Arbor, MI, USA: Univ. of Michigan Press, 1975.

[16] H. C. Lau and C. Zhang, "Job scheduling with unfixed availability constraints," in *Proc. ACM 35th Meeting Decis. Sci. Inst. (DSI)*, 2004, pp. 4401–4406.

[17] M. Hu and B. Veeravalli, "Requirement-aware scheduling of bag-of-tasks applications on grids with dynamic resilience," *IEEE Trans. Comput.*, vol. 62, no. 10, pp. 2108–2114, Oct. 2013.

[18] A. Iosup, O. Sonmez, S. Anoep, and D. Epema, "The performance of bags-of-tasks in large-scale distributed systems," in *Proc. 17th Int. Symp. High Perform. Distrib. Comput.*, 2008, pp. 97–108.

[19] A. Janiak and W. Janiak, "Single-processor scheduling problem with dynamic models of task release dates," *IEEE Trans. Syst., Man, Cybern. A, Syst. Humans*, vol. 41, no. 2, pp. 264–271, Mar. 2011.

[20] I. Kacem, C. Sadfi, and A. El-Kamel, "Branch and bound and dynamic programming to minimize the total completion times on a single machine with availability constraints," in *Proc. IEEE Int. Conf. Syst., Man Cybern.*, vol. 2, Oct. 2005, pp. 1657–1662.

[21] G. Kumar and M. K. Rai, "An energy efficient and optimized load balanced localization method using CDS with one-hop neighbourhood and genetic algorithm in WSNs," *J. Netw. Comput. Appl.*, vol. 78, pp. 73–82, Jan. 2017.

[22] W. Y. Lee, S. J. Hong, and J. Kim, "On-line scheduling of scalable real-time tasks on multiprocessor systems," *J. Parallel Distrib. Comput.*, vol. 63, no. 12, pp. 1315–1324, 2003.

[23] Y. C. Lee and A. Y. Zomaya, "Practical scheduling of bag-of-tasks applications on grids with dynamic resilience," *IEEE Trans. Comput.*, vol. 56, no. 6, pp. 815–825, Jun. 2007.

[24] A. Legrand and C. Touati, "Non-cooperative scheduling of multiple bag-of-task applications," in *Proc. IEEE 26th Int. Conf. Comput. Commun. (INFOCOM)*, May 2007, pp. 427–435.

[25] J. Li, L. Shu, J.-J. Chen, and G. Li, "Energy-efficient scheduling in non-preemptive systems with real-time constraints," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 43, no. 2, pp. 332–344, Mar. 2013.

[26] C.-C. Lin, H.-H. Chin, and W.-B. Chen, "Balancing latency and cost in software-defined vehicular networks using genetic algorithm," *J. Netw. Comput. Appl.*, vol. 116, pp. 35–41, Aug. 2018.

[27] M. Maheswaran, S. Ali, H. J. Siegal, D. Hensgen, and R. F. Freund, "Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems," in *Proc. 8th Heterogeneous Comput. Workshop (HCW)*, Apr. 1999, pp. 30–44.

[28] S. Mingsheng, "Optimal algorithm for scheduling large divisible workload on heterogeneous system," *Appl. Math. Model.*, vol. 32, no. 9, pp. 1682–1695, 2008.

[29] M. A. Oxley *et al.*, "Makespan and energy robust stochastic static resource allocation of a bag-of-tasks to a heterogeneous computing system," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 10, pp. 2791–2805, Oct. 2015.

[30] C. Pang, J. Yan, and V. Vyatkin, "Time-complemented event-driven architecture for distributed automation systems," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 45, no. 8, pp. 1165–1177, Aug. 2015.

[31] X. Qi, T. Chen, and F. Tu, "Scheduling the maintenance on a single machine," *J. Oper. Res. Soc.*, vol. 50, no. 10, pp. 1071–1078, 1999.

[32] X. Qin and T. Xie, "An availability-aware task scheduling strategy for heterogeneous systems," *IEEE Trans. Comput.*, vol. 57, no. 2, pp. 188–199, Feb. 2008.

[33] M. Sajid, Z. Raza, and M. Shahid, "Energy-efficient scheduling algorithms for batch-of-tasks (BoT) applications on heterogeneous computing systems," *Concurrency Comput., Pract. Exper.*, vol. 28, no. 9, pp. 2644–2669, 2015.

[34] E. Sanlaville and G. Schmidt, "Machine scheduling with availability constraints," *Acta Inf.*, vol. 35, no. 9, pp. 795–811, 1998.

[35] G. Schmidt, "Scheduling with limited machine availability," *Eur. J. Oper. Res.*, vol. 121, no. 1, pp. 1–15, 2000.

[36] G.-J. Sheen and L.-W. Liao, "Scheduling machine-dependent jobs to minimize lateness on machines with identical speed under availability constraints," *Comput. Oper. Res.*, vol. 34, no. 8, pp. 2266–2278, 2007.

[37] S. Srinivasan and N. K. Jha, "Safety and reliability driven task allocation in distributed systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 10, no. 3, pp. 238–251, Mar. 1999.

[38] T. Zhao, K.-L. Li, Z. Xiao, and X. Qin, "A QoS scheduling scheme with availability constraint in distributed systems," in *Proc. 13th Int. Conf. Parallel Distrib. Comput., Appl. Technol. (PDCAT)*, Dec. 2012, pp. 481–486.

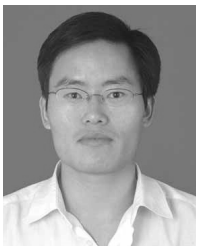
[39] T. Zhao, K.-L. Li, Z. Xiao, and X. Qin, "H2ACO: An optimization approach to scheduling tasks with availability constraint in heterogeneous systems," *J. Internet Technol.*, vol. 15, no. 1, pp. 115–124, 2014.

[40] B. Vahedi-Nouri, P. Fattahi, and R. Ramezanian, "Minimizing total flow time for the non-permutation flow shop scheduling problem with learning effects and availability constraints," *J. Manuf. Syst.*, vol. 32, no. 1, pp. 167–173, 2013.

[41] X. Wang, Y. Wang, and Y. Cui, "A new multi-objective bi-level programming model for energy and locality aware multi-job scheduling in cloud computing," *Future Gener. Comput. Syst.*, vol. 36, pp. 91–101, Jul. 2014.

[42] X. Wang and T. C. E. Cheng, "An approximation scheme for two-machine flowshop scheduling with setup times and an availability constraint," *Comput. Oper. Res.*, vol. 34, no. 10, pp. 2894–2901, 2007.
[43] L. Xiao, Y. Zhu, L. M. Ni, and Z. Xu, "Incentive-based scheduling for market-like computational grids," *IEEE Trans. Parallel Distrib. Syst.*, vol. 19, no. 7, pp. 903–913, Jul. 2008.
[44] H. Xuan, Y. Wang, and X. Wang, "Fault-tolerant scheduling algorithm with re-allocation for divisible loads on homogeneous distributed system," *IEEE Access*, vol. 6, pp. 73147–73157, 2018.
[45] Y. Yang, X. Peng, and X. Wan, "Security-aware data replica selection strategy for bag-of-tasks application in cloud computing," *J. High Speed Netw.*, vol. 21, no. 4, pp. 299–311, 2015.
[46] H. Yuan, Y. Wang, L. Chen, "An availability-aware task scheduling for heterogeneous systems using quantum-behaved particle swarm optimization," in *Proc. Int. Conf. Swarm Intell.* Berlin, Germany: Springer, 2010, pp. 120–127.
[47] F. Zhang, J. Cao, K. Li, S. U. Khan, and K. Hwang, "Multi-objective scheduling of many tasks in cloud platforms," *Future Generat. Comput. Syst.*, vol. 37, pp. 309–320, Jul. 2014.
[48] Y. Zhang and Y. Zhou, "Distributed coordination control of traffic network flow using adaptive genetic algorithm based on cloud computing," *J. Netw. Comput. Appl.*, vol. 119, pp. 110–120, Oct. 2018.
[49] L. Zhou, H. Wang, S. Lian, Y. Zhang, A. Vasilakos, and W. Jing, "Availability-aware multimedia scheduling in heterogeneous wireless networks," *IEEE Trans. Veh. Technol.*, vol. 60, no. 3, pp. 1161–1170, Mar. 2011.

HEJUN XUAN received the B.Sc. degree in computer science and technology from Xinyang Normal University, China, in 2012, and the Ph.D. degree in computer software and theory from Xidian University, China, in 2018. He is currently with the School of Computer and Information Technology, Xinyang Normal University. His research interests include cloud/grid/cluster computing, and scheduling in parallel and distributed systems.



SHIWEI WEI received the B.Sc. and M.Sc. degrees in computer science and technology from the Guilin University of Electronic Technology, China, in 2004 and 2007, respectively. He is currently an Associate Professor in computer and technology with the Guilin University of Aerospace Technology. His research interests include cloud computing and machine learning.

YANLING LI received the Ph.D. degree in computer science and technology from the Huazhong University of Science and Technology, China. She is currently a Professor with the Computer and Information Technology, Xinyang Normal University. Her research interests include machine learning and image processing.

HUAPING GUO received the Ph.D. degree in computer science and technology from Zhengzhou University, China. He is currently an Assistant Professor with the Computer and Information Technology, Xinyang Normal University. His research interests include machine learning and image processing.