Journal of Computers Vol. 31 No. 3, 2020, pp. 40-57
doi:10.3966/199115992020063103004

Multi-objective Resource Allocation of Bag-of-Tasks in Heterogeneous Computing System

Shiwei Wei[1], Hejun Xuan[2*]

[1] School of Computer Science and Technology, Guilin University of Aerospace Technology Guilin, 541004, Guangxi, China
swwei_2001@163.com

[2] School of Computer and Information Technology, Xinyang Normal University, Xinyang, 464000, Henan, China
xuanhejun0896@126.com

Received 17 September 2018; Revised 17 March 2019; Accepted 9 April 2019

Abstract. With the development of new computer technologies and cloud computing, more and more heterogeneous computing systems are becoming an important platform to process large-scale applications in many fields. How to effectively utilize various resources in heterogeneous computing system to process a large amount of independent tasks (also called bag-of-tasks) is a challenging problem, especially when considering more than one objectives of tasks. In this paper, we investigate a bi-objective resource allocation problem for bag-of-tasks in heterogeneous computing system. Since system cost and makespan of workloads are the major issue that need to be considered by the data center, we establish a bi-objective optimization model which minimizes the cost of system and the makespan of the bag-of-tasks, and then propose a novel effective bi-objective genetic algorithm based on MOEA/D to efficiently solve the bi-objective model. Finally, numerical simulation experiments are conducted, and experimental results verify the effectiveness of the proposed model and algorithm.

Keywords: bi-objective optimization, BoT, genetic algorithm, heterogeneous computing, MOEA/D

1 Introduction

In the past several decades, heterogeneous computing system has emerged as an efficient platform to process large-scale applications in various fields, such as biological information, geography, astronomy, and physics [1-3]. Numerous users around the world, such as academic institution, government agency or commercial enterprise, can submit numerous computing tasks and conveniently conduct them on the heterogeneous computing system. Therefore, how to make all the tasks from different users be done more quickly and more efficiently by optimally scheduling of resources is a key problem which need to be studied. What is more, the dynamic and heterogeneity of the heterogeneous computing system also increase the difficulty of solving the scheduling of resource problem. In this paper, we investigate the bag-of-tasks (BoT) scheduling problem on heterogeneous computing system. BoT, which consists of numerous independent tasks, is a typically "embarrassingly parallel" type of task and can be processed in parallel without communications [4]. It is a good candidate to execute such tasks on heterogeneous distributed computing systems, for BoT can be executed on multiple processors simultaneously.

A large number of scheduling algorithms have been proposed in the past several decades. Since this scheduling problem is an NP-hard problem [5], the majority of proposed scheduling algorithms, such as Well-known Max-Min and Min-Min algorithms, are heuristic algorithms. In Max-Min algorithm, the largest task among all unscheduled tasks is scheduled preferentially, and it should be allocated to the

* Corresponding Author

processor which can complete its current tasks within minimal time.

The main difference between Max-Min algorithm and Min-Min algorithm is that the task with the shortest completion time (the smallest task) is scheduled first in Min-Min scheduling algorithm, while the task with the longest completion time (the largest task) scheduled first in Max-Min algorithm. Maheswaran, et al. proposed a *Sufferage* algorithm in literature [6]. The *sufferage* value of a task is defined, and the task with the largest *sufferage* value among all unscheduled tasks is always scheduled preferentially. These methods including recently proposed algorithms [7-9] are all knowledge-based algorithms. If the prediction information is available, it is simple and efficient to deal with resource scheduling problem by using these methods. However, most of time such as prediction information may be difficult to be determined in practice. Therefore, some knowledge-free scheduling algorithms have been proposed, such as RR [10]. After controlling list scheduling for the tasks in the unscheduled queue, RR makes replicas for the running tasks in a round-robin fashion. Cirne, et al. proposed WQR algorithm [11]. WQR algorithm allocates tasks to processors as soon as these processors become available. WQR starts to create replicas of the running tasks after all tasks were scheduled. The prediction information of underlying resources is unnecessarily required for knowledge-free scheduling algorithm. Even the prediction information is unavailable, knowledge-free scheduling algorithms are still efficient to be performed well.

For scheduling grid tasks, an on-line task scheduling algorithm, which called prudent algorithm with replication (PAR), is proposed. PAR can make an efficient scheduling strategy even when the performance predictions are imprecise or inaccurate. What is more, PAR develops a duplication strategy to deal with enormous increasement of task scheduling when the number of the tasks increase. Lee and Zomaya classified the tasks into computation-intensive and data-intensive BOT [12] and presented two task scheduling algorithms in Grid computing system. A scheduling algorithm with fault-aware strategy was proposed for BoT task scheduling on desktop Grids [13]. Avinab [14] proposed an energy-aware fault-tolerant dynamic task scheduling scheme for virtualized cloud data centers. Abdi also proposed an approach named GRASP-FC [15] for obtaining an approximate optimal solution of BoT scheduling in the federation to minimize financial cost including fees for running VMs and fees for data transfer, and fulfill deadline and resource constraints in the clouds. In addition, the performance of BoT in large-scale distributed systems also has been studied [16-17]. A classification of scheduling strategy has been proposed, and a workload model for BoT was established. These approaches aforementioned always have the goal of minimizing the makespan of the tasks. However, there are some other studies that aim to maximize throughput by establishing linear programs or nonlinear programs [15-18], and these works focus on steady-state optimization problems and are concentrated in numerous bag-of-task. Legrand et al. proposed a centralized and decentralized scheduling algorithm [18]. To schedule concurrent bag-of-tasks, Benoit et al. [19] presented an online and off-line scheduling algorithm. Long presented an approach to schedule jobs whose performance are unknown before execution with deadlines on the cloud [20]. A more realistic and probabilistic task model was proposed, and intelligent heuristic scheduling algorithm was designed [21]. Gu et al. [22] proposed a TVSA algorithm to minimize the makespan for studing the large-scale mixed job shop scheduling problem with general number of jobs on each route. Celaya, et al. designed a decentralized scheduling algorithm which minimizes the maximum stretch among user-submitted tasks [23]. Yang et al. [24] designed a scheduling algorithm for data-intensive tasks, where the processing time, system cost and security are all taken into consideration. Oxley et al. investigated both two problems: optimizing the makespan of the tasks under the constraints of energy, or minimizing energy consumption subject to makespan [25]. However, this study is only suitable to the static resource allocation problem, the purpose of which is to optimize makespan and energy Robust stochastic for bag-of-tasks on a heterogeneous computing system. For the sake of minimizing the makespan and the energy consumption, Sajid et al. [26] proposed two energy-aware task scheduling algorithms to schedule numerous independent tasks on heterogeneous computing system. In order to process various tasks including bags of independent gangs and bags-of-tasks, a hybrid task scheduling algorithm was proposed [27]. A multi-objective optimization model, which minimizes makespan and resource cost, was established [5, 28]. To solve the optimization model, a scheduling algorithm based on the ordinal optimization method was designed. However, the scheduling algorithm is inefficient when the task number or processing node number is large.

For users, they want their submitted tasks in the heterogeneous system to be finished in shorter time. But service providers prefer to spend less system cost on the tasks if they can meet users' demand. To

this end, we design a bi-objective optimization model (BOOM for short) to handle bag-of-tasks on distributed computing systems, where both processing time of the tasks (makespan) and the system cost are taken into consideration. The major contributions of our work are summarized as follows:

- Not only the makespan but also the system running cost is taken into account and a bi-objective optimization model (BOOM) for dealing with bag-of-tasks problem is established in heterogeneous distributed system.
- For the sake of solving the bi-objective model efficiently, a novel bi-objective genetic algorithm based on MOEA/D is proposed, which is an effective and efficient algorithm to tackle the BOOM.
- Comprehensive experiments are conducted to verify the performance of the proposed model and the corresponding algorithm. Experimental results indicate that BOOM and the bi-objective genetic algorithm outperform other algorithms. The obtained Pareto front are both minimized in terms of the system cost and the makespan of tasks, which can supply many solutions for users to select.

The rest of this paper is organized as follows: Section 2 introduces some basic concepts of multi-objective optimization; Section 3 descripts the bi-objective optimization model in details; Section 4 presents a novel genetic algorithm based on MOEA/D to solve the bi-objective optimization model; Section 5 evaluates the performance of the proposal by comparing with some classic algorithms on comprehensive experiments; And finally, conclusions are drawn in Section 6.

2 Related Concepts of Multi-objective Optimization

A multi-objective optimization problem (MOP) has been widely used in engineer application and scientific research, and it can be described as

$$\begin{cases} \min\ y = F(x) = (f_1(x), f_2(x), \cdots, f_m(x))^T, \\ s.t. \\ \quad x \in \Omega, \end{cases} \tag{1}$$

where Ω is the feasible region for the multi-objective optimization problem. $f_i(x)(i = 1,2,\cdots,m)$ are the objective functions. In general, the objectives are conflicting to each other in a multi-objective problem, which means that no solution in feasible region can minimize all the objective functions simultaneously. Therefore, multi-objective optimization algorithms are generally used to find a set of Pareto optimal solutions in the feasible region for MOP problem.

Definition 1. For the minimization problem of each objective, two solutions $x_a, x_b \in x_f$ (x_f is the feasible region for the multi-objective optimization problem), x_b is **dominated** by the solution x_a (denoted as $x_a \prec x_b$) if and only if formula (2) is satisfied:

$$\begin{cases} f_i(x_a) \le f_i(x_b) & \forall i = 1,2,\cdots,m, \\ f_j(x_a) < f_j(x_b) & \exists j = 1,2,\cdots,m. \end{cases} \tag{2}$$

Definition 2. The solution $x*$ is a **Pareto optimal solution** if and only if formula (3) is satisfied:

$$\neg\exists x \in x_f : x \prec x*. \tag{3}$$

Definition 3. All the Pareto optimal solutions constitute a **Pareto optimal solution set**, which is defined as:

$$P* \triangleq \{x* \mid \neg\exists x \in x_f : x \prec x*\}. \tag{4}$$

Definition 4. Pareto front is a surface which consists of all the objective function vectors determined by solutions in Pareto optimal solution set, and it can be defined as:

$$PF* \triangleq \{F(x*) = (f_1(x*), f_2(x*), \cdots, f_m(x*))^T \mid x* \in P*\}. \tag{5}$$

3 A New Multi-Objective Programming Model

3.1 System and Task Description

In our work, suppose that the heterogeneous computing system has N virtual clusters, and they are denoted by $V=(V_1,V_2,\cdots,V_N)$. For each virtual cluster $V_i(1\le i\le N)$, it contains n_i homogeneous virtual machines. The j^{th} virtual machines on i^{th} virtual cluster is denoted by $v_{ij}(1\le i\le N,1\le j\le n_i)$, and $P(v_{ij})$ $(1\le i\le N,1\le j\le n_i)$ denotes the property information of virtual node v_{ij}. In general, the heterogeneous computing system includes virtual machines which are the basic processing unit and a task dispatcher node whose duty is to distribute tasks to the suitable virtual nodes. For better understanding the function of them, the system model is shown in Fig. 1.

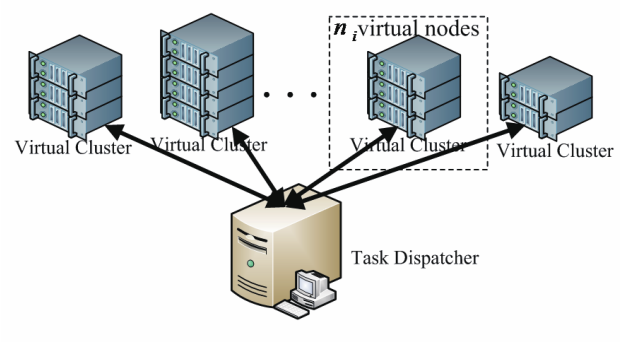

Fig. 1. System Model

For the bag-of-tasks problem, let's assume that there are N_τ independent tasks, and the k^{th} $(1\le k\le N_\tau)$ task is denoted by the tuple $\tau_k=(r_k,\delta_k)$, where r_k is the k^{th} task's lowest requirement for the system while δ_k is the workload of the k^{th} task. According to the previous literature [29] and [30], when a task arrives, it's workload δ_k is known, or can be determined by the prediction mechanisms such as code profiling and statistical prediction.

3.2 Problem Description and Mathematical Model

In the following, our proposals, a bi-objective optimization model (BOOM for short) and a task scheduling algorithm, are descripted in details. The duty of the task dispatcher node in the heterogeneous computing system is to distribute tasks to the suitable nodes in advance, while the task scheduling algorithm, which aims to minimize the system cost and processing time (makespan) simultaneously, is proposed for solving BOOM model. Following gives the formula about the system cost model and the makespan of tasks. If the task $\tau_k(1\le k\le N_\tau)$ is distributed to the virtual machine v_{ij}, the processing time t_{ijk} can be calculated by Equation 6, while the system cost c_{ijk} can be obtained by Equation 7.

$$t_{ijk}=\delta_k w_{ik}, \tag{6}$$

$$c_{ijk}=\delta_k u_{ik}, \tag{7}$$

where w_{ik} and u_{ik} are the time consuming and system cost per unit of task τ_k on virtual cluster V_i respectively.

Let θ_{ijk} denote whether the task τ_k is distributed to the virtual node v_{ij} or not. If the task τ_k is distributed to the virtual node v_{ij}, $\theta_{ijk}=1$; otherwise $\theta_{ijk}=0$. T_{ij} denotes the processing finish time of virtual node V_{ij}, while C_{ij} is the system cost of virtual node V_{ij}. T_{ij} and C_{ij} are determined by Equation 8 and Equation 9, respectively.

$$T_{ij} = \sum_{k=1}^{N_r} (\theta_{ijk} t_{ijk}) = \sum_{k=1}^{N_r} (\theta_{ijk} \delta_k w_{ik}); \tag{8}$$

$$C_{ij} = \sum_{k=1}^{N_r} (\theta_{ijk} c_{ijk}) = \sum_{k=1}^{N_r} (\theta_{ijk} \delta_k u_{ik}). \tag{9}$$

Makespan of tasks is the max of lasting time during which all of tasks have already completed on the virtual machines, and it is determined by Equation 10. The system cost of tasks is determined by Equation 11.

$$T = \max_{i \le i \le N, 1 \le j \le n_i} \{T_{ij}\} = \max_{i \le i \le N, 1 \le j \le n_i} \left\{ \sum_{k=1}^{N_r} (\theta_{ijk} \delta_k w_{ik}) \right\}; \tag{10}$$

$$C = \sum_{i=1}^{N} \sum_{j=1}^{n_i} C_{ij} = \sum_{i=1}^{N} \sum_{j=1}^{n_i} \sum_{k=1}^{N_r} (\theta_{ijk} \delta_k u_{ik}). \tag{11}$$

The bi-objective for scheduling tasks in heterogeneous computing systems is to minimize both the makespan T and the total system cost C simultaneously. That is to say, when starting to distribute tasks to various virtual machines, not only the makespan but also the total system cost must be taken into account. A bi-objective optimization model for scheduling bag-of-tasks in heterogeneous computing systems is established as shown in Equation 12.

$$\begin{cases} \min T = \min \left\{ \max_{i \le i \le N, 1 \le j \le n_i} \left\{ \sum_{k=1}^{N_r} (\theta_{ijk} \delta_k w_{ik}) \right\} \right\}, \\ \min C = \min \left\{ \sum_{i=1}^{N} \sum_{j=1}^{n_i} \sum_{k=1}^{N_r} (\theta_{ijk} \delta_k u_{ik}) \right\}. \\ s.t. \\ \quad (1) \ \sum_{i=1}^{N} \sum_{j=1}^{n_i} \sum_{k=1}^{N_r} \theta_{ijk} = 1; \\ \quad (2) \ P(v_{ij}) \ge r_k. \end{cases} \tag{12}$$

There are two objective functions in this optimization model. One is the processing time function and the other is the system cost function. Constraint (1) indicates that all the tasks must be distributed to the virtual machines. Constraint (2) requires that each of virtual nodes, which has been selected to carry out some corresponding tasks, must satisfy the lowest requirement of these tasks. To solve this bi-objective optimization model, an effective genetic algorithm based on MOEA/D is proposed, which will be illustrated in next section.

4 An Effective Genetic Algorithm Based on MOEA/D

To deal with Multi-objective optimization problem, many multi-objective evolutionary algorithms (MOEAs) have been proposed in the past several decades, such as NSGA-II (non-dominated sorting genetic algorithm) [31], and MOEA/D (Multi-objective Evolutionary Algorithm Based on Decomposition) [32]. MOEA/D decomposes the multi-objective optimization problem into several scalar sub-problems, and then optimizes the scalar sub-problems simultaneously by using neighborhood information and single objective global optimize techniques. MOEA/D generally has a better performance than the other popular multi-objective evolutionary algorithms such as NSGA-II. Therefore, to solve the above bi-objective optimization model effectively and efficiently, a new genetic algorithm based on MOEA/D is proposed.

4.1 Encoding and Decoding

Encoding of chromosomes is much more significant, because it is the first step when we utilize genetic algorithm to solve multi-objective optimization problem. A suitable encoding scheme, which expresses the solutions in problem domain by chromosomes, can make the search easier and more efficiently by limiting the search space, and it also helps to simplify the process of solving the complicated problem in practice. Based on characterizes of this bi-objective scheduling for BoT scheduling problem, the encoding scheme of integer is adopted in our algorithm, since it is easier and more intuitive.

A sequence $S=(s_1,s_2,\cdots,N_\tau)$, called chromosome, consists of a list of N_τ elements, where the k^{th} ($1 \le k \le N_\tau$) element s_k of the list is presented by an integer ranging from 1 to $\sum_{i=1}^{N} n_i$. If the task k is distributed to the virtual node v_{ij}, the k^{th} element s_k in the chromosome is calculated by Equation 13:

$$s_k = \sum_{p=1}^{i-1} n_p + j . \tag{13}$$

The procedure which translates the chromosome into the distribution scheme is called decoding. For a specific chromosome $S=(s_1,s_2,\cdots,N_\tau)$, the corresponding distribution sequence can be obtained by decoding. An effective decoding scheme will be given below. If task τ_k is distributed to the virtual node v_{ij}, the i and j can be calculated by Equation 14 and Equation 15 as follows:

$$i = \left\{ p \left| \sum_{q=1}^{p} n_q \le s_k < \sum_{q=1}^{p+1} n_q \right. \right\}; \tag{14}$$

$$j = s_k - \sum_{q=1}^{i} n_q . \tag{15}$$

In order to make encoding and decoding simpler and easier, we use a fixed sequence of positive integers as reference sequence to identify each virtual cluster in the heterogeneous system, and perform the same operations for each virtual machine.

4.2 Crossover Operator

Crossover is an important operator in genetic algorithm, which can generate new offsprings by combining two parents. In general, some offsprings obtained by crossover operation may be better than both of their parents if they take the best characteristics from each of the parents. In our work, three crossover operators are designed, which are multiple-point crossover, differential evolutionary (DE) and Particle Swarm Optimization (PSO). The specific steps for using these crossover operators are shown in Algorithm 1.

Algorithm 1. Crossover Operator

Input: A chromosome C, some parameters such as p_c, p_l, c_1, c_2, m_1 and m_2;
Output: The set of offsprings O;

```
if rand() < p_c then
  if rand() < p_1 then
   An individual in the neighborhood of individual C is randomly
      selected, denoted as C^1;
  else
   An individual not in the neighborhood of individual C is randomly
      selected, denoted as C^n;
  end
  Perform multi-point crossover operations on C and C^1 according to
   the crossover operator;
  Differential evolutionary is implemented by Eq.(16) and Eq.(17);
  Particle Swarm Optimization is implemented by Eq.(19), Eq.(20),
   Eq.(21) and Eq.(22);
```

```
end
Put all the offsprings into the set O;
return O;
```

$$x_{ci}^{t+1} = x_i^t + F \times (g_c - f(x_i^t)), \tag{16}$$

$$x_{mi}^{t+1} = x_i^t + F \times (g_m - f(x_i^t)), \tag{17}$$

where x_i^t is the i^{th} $(1 \le i \le Popsize)$ individual in the t^{th} generation population. x_{ci}^{t+1} and x_{mi}^{t+1} are both offsprings of x_i^t, which are generated by different equations. g_c is the individual with the minimum system cost among all individuals in all generation population up to now, and g_m is the individual with the minimum makespan among all individuals. F is a constant and $f(x_i^t)$ is an individual which is determined by the following equations:

$$f(x_i^t) = \begin{cases} L(P) & rand() < p_l \\ nL(P) & rand() \ge p_l \end{cases}. \tag{18}$$

where P is the population, L is a subset of population P and it consists of the neighborhood of individual x_i^t, $nL = P \setminus \mathrm{L}$, p_l is a constant and satisfies $0 < p_l < 1$. That is to say, if $rand() < p_l$, $f(x_i^t)$ is a random individual in L, otherwise, $f(x_i^t)$ is a random individual in nL.

$$v_{ci}^{t+1} = v_{ci}^t + c_1 \times rand() \times (g_c - x_i^t) + c_2 \times rand() \times (l_c^t - x_i^t), \tag{19}$$

$$v_{mi}^{t+1} = v_{mi}^t + m_1 \times rand() \times (g_m - x_i^t) + m_2 \times rand() \times (l_m^t - x_i^t), \tag{20}$$

$$x_{ci}^{t+1} = x_i^t + v_{ci}^{t+1}, \tag{21}$$

$$x_{mi}^{t+1} = x_i^t + v_{mi}^{t+1}, \tag{22}$$

where c_1, c_2, m_1, m_2 are four constants and *rand()* is a random function which can randomly generate float numbers varying from *0* to *1;* v_{ci}^t and v_{mi}^t indicate the moving speed of x_c^t; l_c^t represents the individual with the minimum system cost in the t^{th} generation, while l_m^t represents the one with minimum makespan.

4.3 Mutation Operator

Mutation is a genetic operator used to maintain genetic diversity from one generation of a population of genetic algorithm chromosomes to the next. A better mutation operator may produce some novel offspring individuals, which may change entirely from the previous individual, so it can maintain or increase the diversity of population. In addition, mutation operator is also helpful for population to escape from the local optimal solution. Both theory and empirical experiments have proved that a mutation operator well designed can make the genetic algorithm generate a better solution set [31-33]. The design scheme of the mutation operator used in our paper is given below. Suppose that the chromosome $O = (o_1, o_2, \cdots, o_{N_\tau})$ is selected to take part in mutation operation, and the offspring $O' = (o'_1, o'_2, \cdots, o'_{N_\tau})$ is determined by the mutation (shown in Algorithm 2).

Algorithm 2. Mutation Operator

Input: An chromosome O;
Output: The offsprings O';

```
Let O'=O;
for i=1 to N_τ do
    if rand() < p_m then
```

```
        v = randi([1, ∑_{j=1}^{N} n_j]);
        if v ≥ r_i then
            O'[i] = v;
        end
    end
end
```

4.4 Fitness Function

Fitness function is an indicator used to evaluate the quality of each individual in one generation of a population. In general, a fitness function is derived from or related to the objective optimization model. In this paper, since a bi-objective optimization model is established, two fitness functions are built to measure the quality of individuals. In addition, these two fitness functions are consistent with the two objective in the bi-objective optimization model, which are described in Algorithm 3.

Algorithm 3. Fitness Function

Input: An individual $S = (s_1, s_2, \cdots, s_{N_r})$;

Output: The fitness function $G(S) = (G_1(S), G_2(S))$;

1. Generate the distribution sequence $H = (h_1, h_2, \cdots, h_{N_r})$ by decoding the individual ***S*** according to Eq.(14) and Eq.(15);
2. Calculate the makespan ***T*** and system cost ***C*** by Eq.(10) and Eq.(11);
3. **Let** $G_1(S) = T$ and $G_2(S) = C$;
4. **return** $G(S)$;

4.5 Local Search

Local search is also an important operator in genetic algorithm which can help to jump out the local optima. In this paper, in order to accelerate the convergence and enhance the searching ability of the proposed algorithm, a local search operator is specially designed and used in our algorithm which is presented in Algorithm 4.

Algorithm 4. Local Search

Input: An individual $S = (s_1, s_2, \cdots, s_{N_r})$;

Output: A new offsprings $S' = (s'_1, s'_2, \cdots, s'_{N_r})$;

```
Let S' = S;
i and j are generated randomly;
if s'_j ≥ r_i then
    s'_i = s_j;
else
    do
        v = randi([1, ∑_{j=1}^{N} n_j]);
    while v ≥ r_i
    s'_i = r_i;
end
if s'_i ≥ r_j then
    s'_j = s_i;
```

else

$s'_j = randi(r_j, N_\tau)$;

do

$v = randi([1, \sum_{j=1}^{N} n_j])$;

while $v \geq r_j$

$s'_j = r_j$;

end

if S' is dominated by S **then**

$S' = S$;

end

return S';

4.6 A Novel Genetic Algorithm Based on MOEA/D

A novel genetic algorithm based on MOEA/D is proposed in this paper, which can minimize system cost and makespan simultaneously for BoT scheduling problem in distributed system. Main steps of this novel genetic algorithm is shown in Algorithm 5 below.

Algorithm 5. A Novel Genetic Algorithm Based on MOEA/D

1. **Initialization:**
2. **Set** Epa = ϕ, U = 1,2,···, q;
3. Generate initial population $p_1, p_2, \cdots, p_q$, where q is the number of the population;
4. Generate uniformly distributed weight vectors $\lambda_1, \lambda_2, ..., \lambda_q$, where q is the number of the weight vectors;
5. Calculate the Euclidean distance between vectors p_i and p_j, where $1 \leq i \neq j \leq q$. In addition, T closest vectors of p_i should be calculated out. *NB(i)* is used to store the indexes of the T closest vectors of individual $p_i (i = 1,2,\cdots,q)$;
6. Calculate the fitness value $G_1(p_i)$ and $G_2(p_i)$ of the individual $p_i (i = 1,2,\cdots,q)$;
7. **Update:**
8. **for** *i=1* to *q* **do**
9. **Crossover:** Randomly select an index *k* from *NB(i)* or *U- NB(i)*, and then two new solutions *x* and *y* are generated from p_i and p_k by using the crossover operator designed according to Algorithm 1;
10. **Mutation:** Mutation is executed on *x* and *y* by using mutation operator designed according to Algorithm 2, and two offsprings *x'* and *y'* are determined;
11. **Local Search:** Local search operator is applied on the *x'* and *y'*, and then two offsprings *x"* and *y"* are generated by algorithm 4, finally their fitness values $G(x")$ and $G(y")$ are calculated out.
12. **Update of neighboring solutions:**
13. For each index $j \in NB(i)$, if $g^{ws}(x" | \lambda_j) \leq g^{ws}(p_j | \lambda_j)$, then set $p_j = x"$; If $g^{ws}(y" | \lambda_j) \leq g^{ws}(p_j | \lambda_j)$, then set $p_j = y"$;
14. **Update of** *Epa*:

15. Remove all the vectors dominated by $G(x")$ and $G(y")$ from *Epa*. Add $G(x")$ to *Epa* if no vector in *Epa* dominates $G(x")$. Add $G(y")$ to *Epa* if no vector in Epa dominates $G(y")$;
16. **end**
17. **Stopping Criteria:**
18. If the stopping criteria is satisfied, then stop and output Epa. Otherwise, go to Step 8;

5 Experiments and Analysis

Comprehensive experiments are conducted to evaluate the effectiveness and efficiency of the proposed algorithm. In section 5.1, we give all parameters used in the algorithms. In section 5.2, experimental results and analysis about them are represented.

5.1 Parameters Value

In our experiments, we build 15 heterogeneous virtual clusters in the distributed system, and each of cluster has 3~5 homogeneous virtual machines. The parameters $w_k (1 \le k \le N)$ of the heterogeneous computing system are refer to literature [34]. Since different virtual machine has different processing speed for different task, in our experiments, $w_k = r_i w_k$ is selected, where w_k is the time consuming for unit task of virtual cluster V_k. Similarly, $c_{ik} = a_k r_i w_k + b_k$, where a_k and b_k are generated refer to literature [34]. Parameters β and κ are set as : $\beta = 25$ and $\kappa = 100$. In the proposed genetic algorithm based on MOEA/D, the following parameters are used: population size *pop_size = 100*, crossover probability $p_c = 0.8$, mutation probability $p_m = 0.1$, elitist number $E = 5$ and stop criterion $g_{max} = 2000$. The number of neighbors of each weight vector in MOEA/D is $nT = 10$. In addition, the details about the hardware configuration used in the experiments are shown below:

- Processor: Intel(R) Core(TM)i7
- CPU speed: 2.93GHz
- RAM capacity: 8GB

5.2 Experimental Results and Analysis

Comparison experiment. In the first experiment, the number of task ranges from 50 to 500 and the workload changes in [100, 500]. Bi-GA represents the bi-objective genetic algorithm based on MOEA/D and Bi-GAL denotes the Bi-GA algorithm with a local search operator. To verify the outperformance of Bi-GA and Bi-GAL, we compare them with six well known scheduling algorithms: *Max-Min*, *Min-Min*, *WQR*, *RR*, *NSGA* and *MOS*. Since *Max-Min*, *Min-Min, WQR*, *RR* and *MOS* only aim at minimizing the makespan of the tasks and not taking system cost into consideration, only their makespans are compared when the task number varies from 50 to 500 and workloads are fixed. Experimental results are shown in Fig. 2(a), Fig. 2(b), Fig. 2(c) and Fig. 2(d) where their workload ranges from 100-200, 200-300, 300-400 and 400-500, respectively.

Performance evaluation. To evaluate the performance of the genetic algorithm based MOEA/D, the Pareto front are given in Fig. 3 to Fig. 6 with the task number ranges from 50 to 500 and the workload changes in 100-500. In addition, the following two metrics are utilized to evaluate the Pareto solutions:

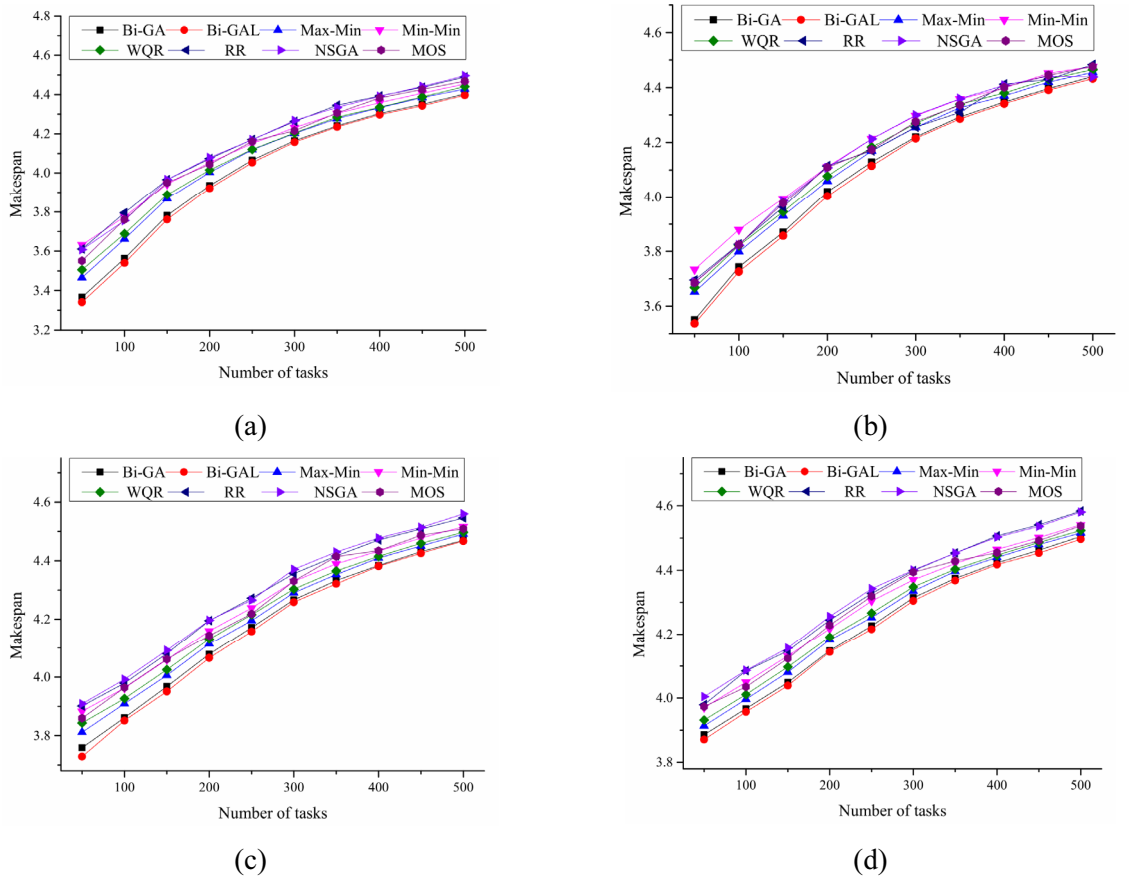

Fig. 2. The makespan changes with the load number ranging from 50 to 500

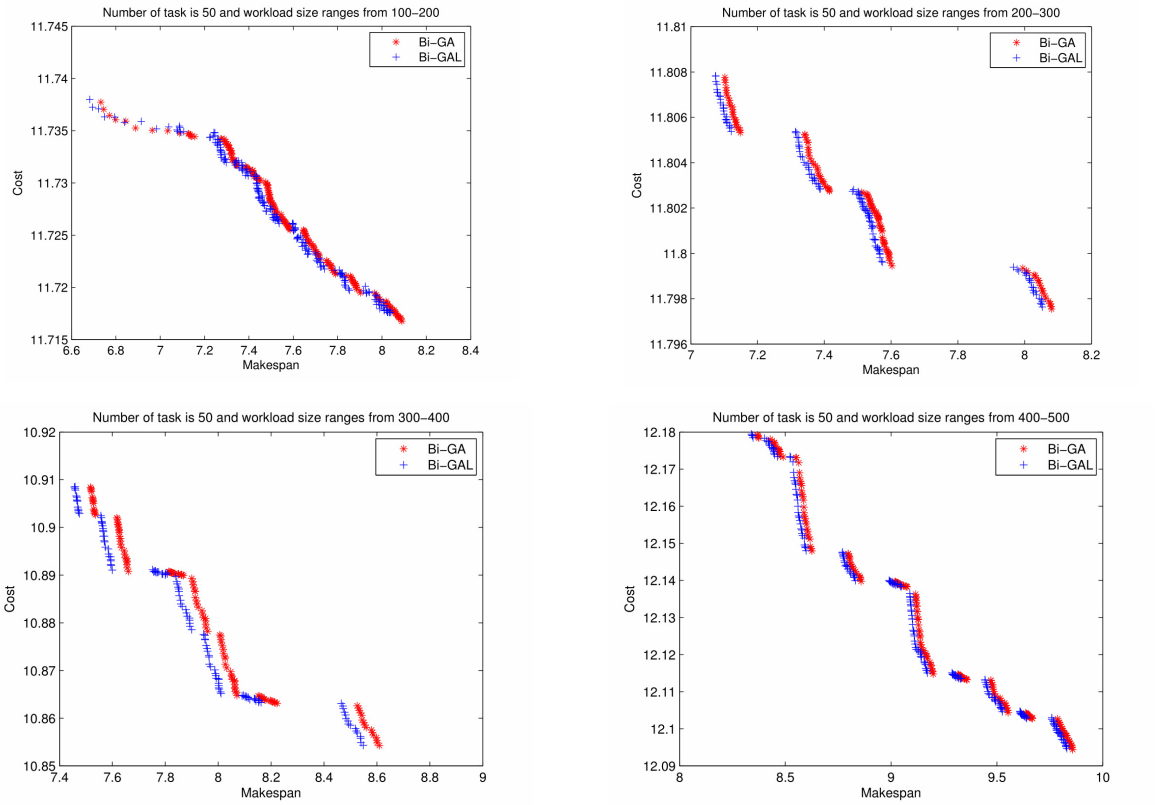

Fig. 3. Pareto fronts of Bi-GA and Bi-GAL when task number is equals to 50

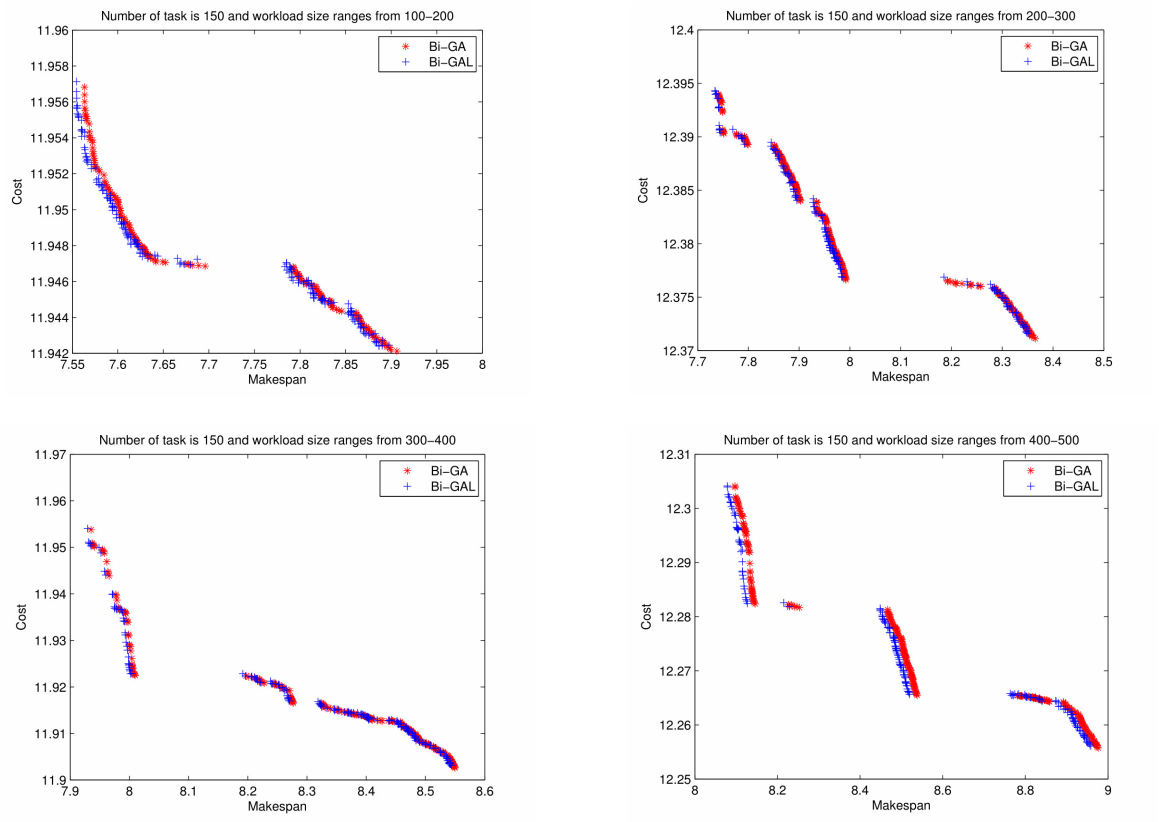

Fig. 4. Pareto fronts of Bi-GA and Bi-GAL when task number is equals to 150

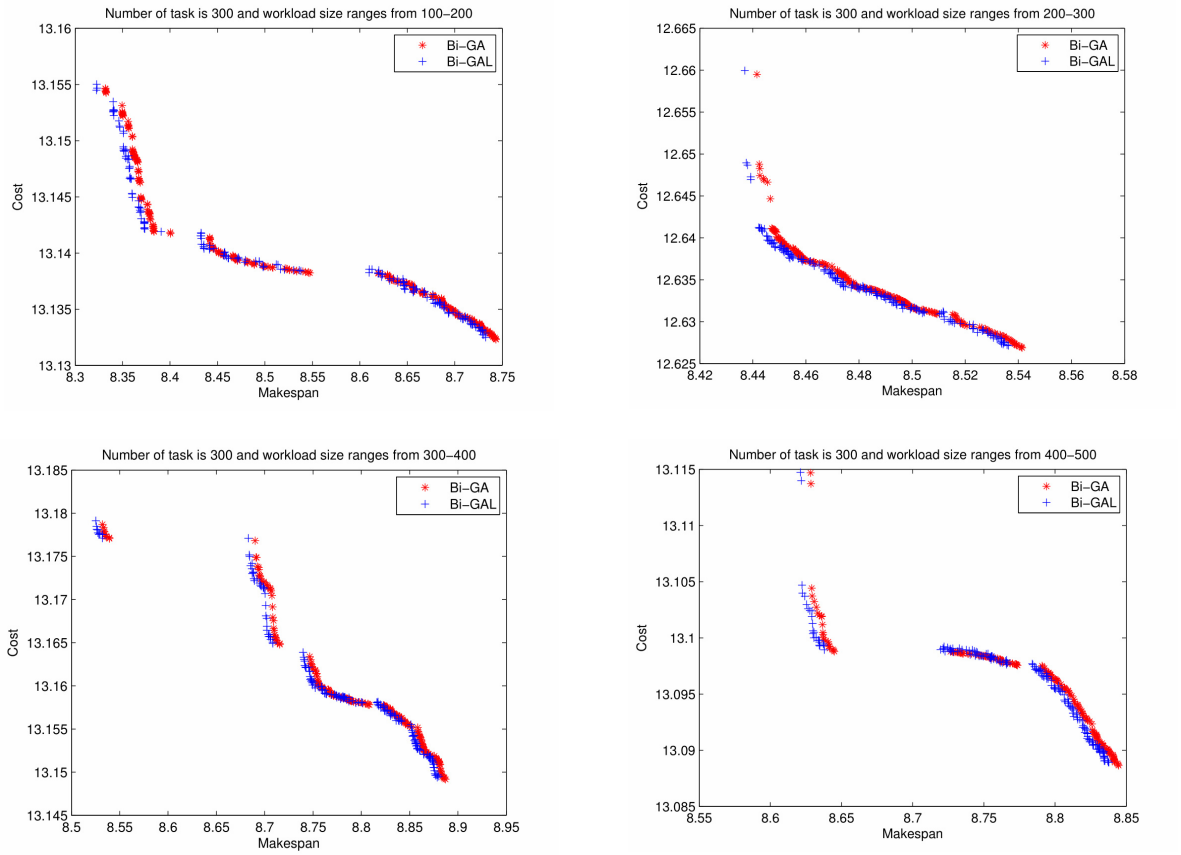

Fig. 5. Pareto fronts of Bi-GA and Bi-GAL when task number is equals to 300

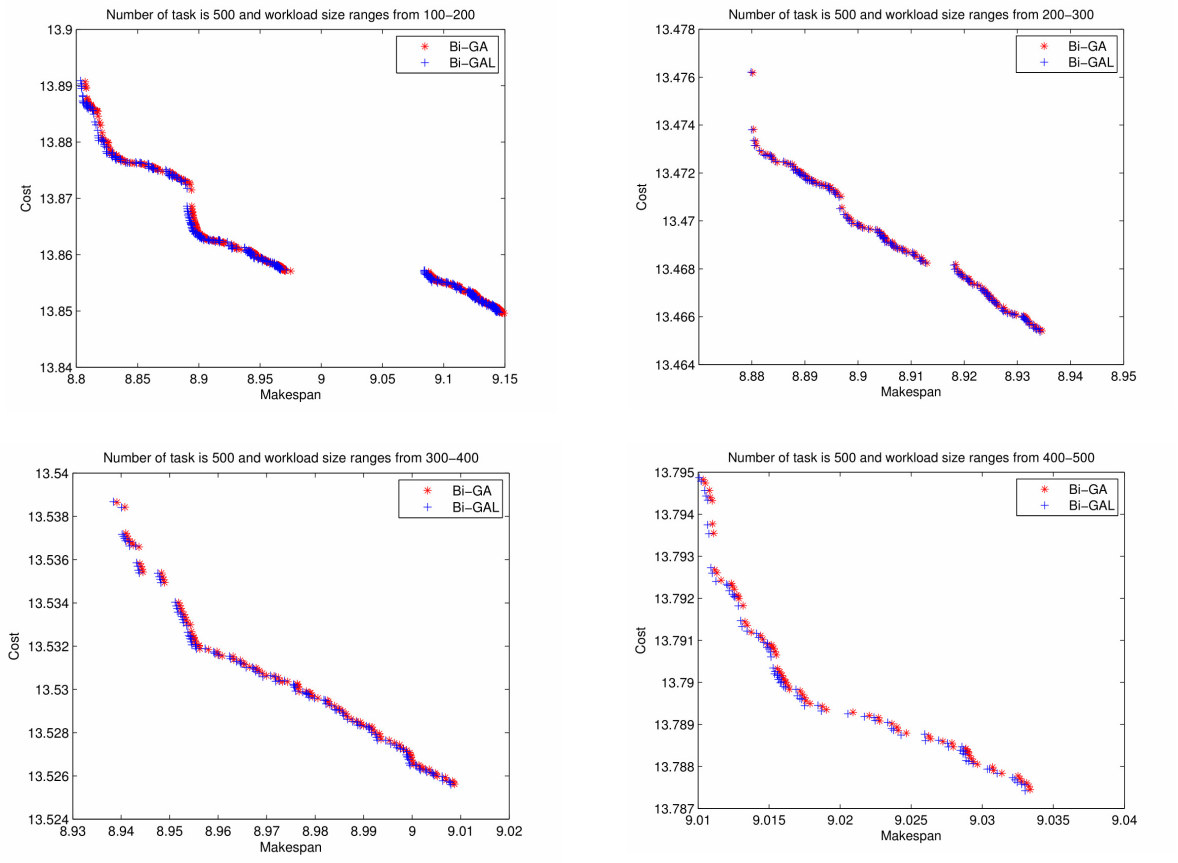

Fig. 6. Pareto fronts of Bi-GA and Bi-GAL when task number is equals to 500

- *Spacing Index(SI)*: defined by Equation 23 below:

$$\begin{cases} SI(A) = \sqrt{\dfrac{1}{|PF^*|-1}\sum_{z\in PF^*}(\bar{d}-d(z))^2}, \\ \bar{d} = \dfrac{1}{|PF^*|}\sum_{z\in PF^*} d(z), \\ d(z) = \min\left\{\|z-z'\| \,\middle|\, z\neq z', z'\in PF^*\right\}. \end{cases} \tag{23}$$

Spacing Index is used to evaluate the uniformity of the pareto solution. The smaller the value of SI is, the better the performance of the solution becomes.

- *Hypervolume Index(HI)*: which is used to test the uniformity, convergence and diversity of the solutions, and defined by the following equation:

$$HI(PF^*) = \bigcup_{z\in PF^*} vol(z), \tag{24}$$

where $vol(z)$ is the hypervolume of area which is surrounded by z and the reference point $r = (r_1, r_2, \cdots, r_m)$, m is the dimensionality of the objective space.

Table 1 and Table 2 give the evaluation results of the pareto optimal solution obtained by algorithm *Bi-GA* and *Bi-GAL*. Since the values of *SI* and *HI* index are large, the conversion formula $\ln(\sqrt{SI})$ and $\ln(\sqrt{HI})$ are used in Table 1 and Table 2.

Table 1. Evaluation index of SI

Task number	50							
Workload	100-200		200-300		300-400		400-500	
Algo.	GA	GAL	GA	GAL	GA	GAL	GA	GAL
Value	2.7804	2.7733	1.9816	2.0450	2.7606	2.6986	4.0336	4.0484
Task number	100							
Workload	100-200		200-300		300-400		400-500	
Algo.	GA	GAL	GA	GAL	GA	GAL	GA	GAL
Value	3.1554	3.2074	2.2841	2.4226	3.0297	3.0194	2.9469	3.0722
Task number	150							
Workload	100-200		200-300		300-400		400-500	
Algo.	GA	GAL	GA	GAL	GA	GAL	GA	GAL
Value	2.2193	2.4041	2.0355	3.3508	3.6969	3.8271	3.1880	3.7457
Task number	200							
Workload	100-200		200-300		300-400		400-500	
Algo.	GA	GAL	GA	GAL	GA	GAL	GA	GAL
Value	4.0251	4.1567	4.9834	4.9932	3.5908	4.0109	4.6008	4.7221
Task number	250							
Workload	100-200		200-300		300-400		400-500	
Algo.	GA	GAL	GA	GAL	GA	GAL	GA	GAL
Value	3.7260	3.7649	3.5852	4.1990	3.7916	3.8233	3.3078	3.5469
Task number	300							
Workload	100-200		200-300		300-400		400-500	
Algo.	GA	GAL	GA	GAL	GA	GAL	GA	GAL
Value	3.1248	3.8371	5.4457	5.5738	4.4964	4.5499	4.2548	4.0863
Task number	350							
Workload	100-200		200-300		300-400		400-500	
Algo.	GA	GAL	GA	GAL	GA	GAL	GA	GAL
Value	4.5996	5.1783	5.5081	5.9893	3.7826	4.6290	3.8120	3.8720
Task number	400							
Workload	100-200		200-300		300-400		400-500	
Algo.	GA	GAL	GA	GAL	GA	GAL	GA	GAL
Value	6.0731	6.0769	5.2802	5.7541	3.3258	3.6360	6.9188	6.9198
Task number	450							
Workload	100-200		200-300		300-400		400-500	
Algo.	GA	GAL	GA	GAL	GA	GAL	GA	GAL
Value	2.9023	3.1809	5.3311	5.3432	3.9073	3.9967	3.6895	3.6708
Task number	500							
Workload	100-200		200-300		300-400		400-500	
Algo.	GA	GAL	GA	GAL	GA	GAL	GA	GAL
Value	4.7801	4.7337	5.0011	5.0198	3.4776	3.6592	3.7396	3.8582

Table 2. Evaluation index of HI

Task number	50							
Workload	100-200		200-300		300-400		400-500	
Algo.	GA	GAL	GA	GAL	GA	GAL	GA	GAL
Value	9.9096	9.8851	9.9422	9.9283	9.7468	9.7164	10.9964	10.9829
Task number	100							
Workload	100-200		200-300		300-400		400-500	
Algo.	GA	GAL	GA	GAL	GA	GAL	GA	GAL
Value	10.1947	10.1812	10.2216	10.2080	10.0237	10.0172	9.7456	9.7454
Task number	150							
Workload	100-200		200-300		300-400		400-500	
Algo.	GA	GAL	GA	GAL	GA	GAL	GA	GAL
Value	9.9300	9.9258	10.3761	10.3693	10.2437	10.2409	10.6307	10.6217

Table 2. (continue)

Task number					50			
Task number					200			
Workload	100-200		200-300		300-400		400-500	
Algo.	GA	GAL	GA	GAL	GA	GAL	GA	GAL
Value	10.4833	10.4794	10.8051	10.7986	10.5072	10.5046	10.3390	10.3339
Task number					250			
Workload	100-200		200-300		300-400		400-500	
Algo.	GA	GAL	GA	GAL	GA	GAL	GA	GAL
Value	10.5227	10.5221	10.9898	10.9835	10.5572	10.5564	10.7861	10.7823
Task number					300			
Workload	100-200		200-300		300-400		400-500	
Algo.	GA	GAL	GA	GAL	GA	GAL	GA	GAL
Value	10.9464	10.9411	10.5992	10.5969	11.0310	11.0277	10.9780	10.9746
Task number					350			
Workload	100-200		200-300		300-400		400-500	
Algo.	GA	GAL	GA	GAL	GA	GAL	GA	GAL
Value	11.2486	11.2451	11.2510	11.2465	11.3851	11.3776	11.2921	11.2902
Task number					400			
Workload	100-200		200-300		300-400		400-500	
Algo.	GA	GAL	GA	GAL	GA	GAL	GA	GAL
Value	11.0667	11.0655	11.3919	11.3889	11.0777	11.0772	11.5094	11.5045
Task number					450			
Workload	100-200		200-300		300-400		400-500	
Algo.	GA	GAL	GA	GAL	GA	GAL	GA	GAL
Value	11.1698	11.1684	11.5349	11.5335	11.2545	11.2534	11.1934	11.1932
Task number					500			
Workload	100-200		200-300		300-400		400-500	
Algo.	GA	GAL	GA	GAL	GA	GAL	GA	GAL
Value	11.5158	11.5140	11.2052	11.2051	11.2734	11.2731	11.4140	11.4139

From Fig. 2, we can see that the makespans obtained by the proposed algorithm are much less than the ones obtained by *Max-Min*, *Min-Min*, *WQR*, *RR*, *NSGA* and *MOS* with various task number and workload. *Max-Min* algorithm always first schedules the largest task among all unscheduled tasks, and then allocates it to the processor with the earliest completion time (makespan). *Max-Min* scheduling algorithm may bring out a result that unbalanced load can occur on processors. The process of *Min-Min* algorithm is similar to the *Max-Min*'s. The main difference between them is that the task with the longest completion time is scheduled first in *Max-Min* scheduling algorithm, while the task with the shortest completion time is scheduled first in *Min-Min* scheduling algorithm. The *Min-Min* algorithm always first schedule the task with a shortest completion time, but the specific task with larger completion time has not been taken into consideration, which results in a vital effect on the makespan. So, the *Min-Min* algorithm has a larger makespan when the task workload changes significantly. *WQR* and *RR* are heuristic algorithms which can obtain a solution quickly, however it usually is an approximate solution but not an optimal one. Therefore, the makespans are larger than that of our algorithms as shown in Fig. 2. For algorithm *NSGA* and *MOS,* although they can got the theoretical optimal solution for given problems, in practices, a suboptimal solution is always generated in a limited time when the scale of the problem is large (i.e., the number of tasks and resource is huge). Because the search pace is so large that the algorithms are easy to fall into the local optimal solution. Therefore, the results obtained by *NSGA* and *MOS* are not better than that of our proposed algorithms. In our research, a local research operator is designed. From Fig. 2, we can see that the makespan of *Bi-GAL* is smaller than that of *Bi-GA*.

As can be seen from Fig. 3 to Fig. 6, the Pareto fronts (PF) determined by *Bi-GA* for different workload are wide spread and uniformly distributed, which indicates that the proposed algorithm *Bi-GA* is able to find various kinds of Pareto optimal solutions. Therefore, it is able to satisfy the requirements of decision makers. Moreover, in order to evaluate the PF obtained by *Bi-GA* and *Bi-GAL*, spacing index (SI) and hypervolume index (HI) are calculated and shown in Table 1 and Table 2, from which we can see that the *Bi-GAL* generates a better PF when solving the Bi-objective optimization model proposed in

this paper.

6 Conclusion

This paper investigates the multi-objective resource allocation problem for bag-of-tasks in heterogeneous computing system. A new bi-objective optimization model is established to minimize the cost of system and the makespan of the tasks simultaneously, For the sake of solving the bi-objective model efficiently, a novel effective bi-objective genetic algorithm based on MOEA/D is proposed, which designs two fitness functions to measure the quality of individuals and develops three crossover operators including multiple-point crossover, differential evolutionary (DE) and Particle Swarm Optimization (PSO) to improve search ability. Comprehensive simulation experiments are conducted to verify the performance of the proposed bi-objective optimization model and algorithm, In particular, two metrics of SI and HI are utilized to measure the solutions determined by the algorithm. The experimental results demonstrate that the proposed algorithm can generate better solutions than other compared algorithms, i.e., the pareto optimal solution set has a better convergence, diversity and uniformity.

In order to further verify the performance of our proposed model and algorithm, they should be utilized to deal with the large-scale task scheduling problem with multi-objectives in heterogeneous computing system in practice, which is the work direction we will study next. In addition, with the increasing of the number of tasks and available resource in system, the search space becomes more and more large, which results in a longer runtime required to obtain a better solution for the given problem. In order to getting a solution effectively and efficiently simultaneously, new strategies need to be developed to reduce the time complexity of the proposed model and algorithm, which is also our future work to be study.

Acknowledgements

This work was supported in part by the Guangxi Natural Science Foundation of China (No. 2016GXNSFAA380226), and by Guangxi Young and Middle-aged Teachers' Basic Ability Improvement Foundation of China (No. 2017KY0866).

References

[1] F. Pop, A. Iosup, R. Prodan, HPS-HDS: high performance scheduling for heterogeneous distributed systems, Future Generation Computer Systems 78(2018) 242-244.

[2] L.F. Bittencourt, A. Goldman, E.R.M. Madeira, Scheduling in distributed systems: A cloud computing perspective, Computer Science Review 30(2018) 31-54.

[3] J.S. Otto, M.A. Sanchez, D.R. Choffnes, On blind mice and the elephant: understanding the network impact of a large distributed system, ACM SIGCOMM Computer Communication Review. ACM 41(4)(2011) 110-121.

[4] M. Hu, B. Veeravalli, Requirement-aware scheduling of bag-of-tasks applications on grids with dynamic resilience, IEEE Transactions on Computers 62(10)(2013) 2108-2114.

[5] F. Zhang, J. Cao, K. Li, Multi-objective scheduling of many tasks in cloud platforms, Future Generation Computer Systems 37(2014) 309-320.

[6] D. Hensgen, M. Maheswaran, S. Ali, Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems, in: Proc. 1999 Heterogeneous Computing Workshop, 1999.

[7] M. Hu, J. Luo, Y. Wang, Adaptive scheduling of task graphs with dynamic resilience, IEEE Transactions on Computers 66(1)(2017) 17-23.

[8] I.A. Moschakis, H.D. Karatza, A meta-heuristic optimization approach to the scheduling of bag-of-tasks applications on heterogeneous clouds with multi-level arrivals and critical jobs, Simulation Modelling Practice and Theory (57)(2015) 1-25.

[9] N. Kumar, N. Chilamkurti, S. Zeadally, Achieving Quality of Service (QoS) Using Resource Allocation and Adaptive Scheduling in Cloud Computing with Grid Support, Computer Journal 57(2)(2018) 281-290.

[10] N. Fujimoto, K. Hagihara, Near-optimal dynamic task scheduling of independent coarse-grained tasks onto a computational grid, in: Proc. 2003 International Conference on Parallel Processing, 2003.

[11] W. Cirne, F. Brasileiro, D. Paranhos, On the efficacy, efficiency and emergent behavior of task replication in large distributed systems, Parallel Computing 33(3)(2007) 213-234.

[12] Y.C. Lee, A.Y. Zomaya, Practical scheduling of bag-of-tasks applications on grids with dynamic resilience, IEEE Transactions on Computers 56(6)(2007) 815-825.

[13] C. Anglano, J. Brevik, M. Canonico, Fault-aware scheduling for bag-of-tasks applications on desktop grids, in: Proc. 2006 IEEE/ACM International Conference on Grid Computing, 2006.

[14] A. Marahatta, Y. Wang, F. Zhang, Energy-aware fault-tolerant dynamic task scheduling scheme for virtualized cloud data centers, Mobile Networks and Applications (2018) 1-15. doi:10.1007/s11036-018-1062-7

[15] S. Abdi, L. PourKarimi, M. Ahmadi, Cost minimization for bag-of-tasks workflows in a federation of clouds, The Journal of Supercomputing 74(6)(2018) 2801-2822.

[16] A. Iosup, O. Sonmez, S. Anoep, The performance of bags-of-tasks in large-scale distributed systems, in: Proc. 2008 International Symposium on High Performance Distributed Computing, 2008.

[17] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G.S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viegas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng, TensorFlow: large-scale machine learning on heterogeneous distributed systems. <https://arxiv.org/abs/1603.04467>, 2016.

[18] O. Beaumont, L. Carter, J. Ferrante, Centralized versus distributed schedulers for bag-of-tasks applications, IEEE Transactions on Parallel & Distributed Systems 19(5)(2008) 698-709.

[19] A. Benoit, L. Marchal, J.F. Pineau, Scheduling concurrent bag-of-tasks applications on heterogeneous platforms, IEEE Transactions on Computers 59(2)(2010) 202-217.

[20] L. Thai, B. Varghese, A. Barker, Minimising the execution of unknown bag-of-task jobs with deadlines on the cloud, in: Proc. 2016 ACM International Workshop on Data-intensive Distributed Computing, 2016.

[21] H. Casanova, M. Gallet, F. Vivien, Non-clairvoyant Scheduling of Multiple Bag-of-Tasks Applications, in: Proc. 2010 International Euro-par Conference on Parallel Processing, 2010.

[22] M. Gu, X. Lu, J. Gu, An asymptotically optimal algorithm for large-scale mixed job shop scheduling to minimize the makespan, Journal of Combinatorial Optimization 33(2)(2017) 473-495.

[23] J. Celaya, U. Arronategui, Fair scheduling of bag-of-tasks applications on large-scale platforms, Future Generation Computer Systems 49(2015) 28-44.

[24] Y. Yang, X. Peng, X. Wan, Security-aware data replica selection strategy for Bag-of-Tasks application in cloud computing, Journal of High Speed Networks 21(4)(2015) 299-311.

[25] M.A. Oxley, S. Pasricha, A.A. Maciejewski, Makespan and energy robust stochastic static resource allocation of a bag-of-tasks to a heterogeneous computing system, IEEE Transactions on Parallel and Distributed Systems 26(10)(2015) 2791-

2805.

[26] M. Sajid, R. Zahid, S. Mohammad, Energy-efficient scheduling algorithms for batch-of-tasks (BoT) applications on heterogeneous computing systems, Concurrency and Computation: Practice and Experience 28(9)(2016) 2644-2669.

[27] Z.C. Papazachos, H.D. Karatza, Scheduling bags of tasks and gangs in a distributed system, in: Proc. 2015 International Conference on Computer, Information and Telecommunication Systems (CITS), 2015.

[28] F. Zhang, J. Cao, K. Li, Multi-objective scheduling of many tasks in cloud platforms, Future Generation Computer Systems 37 (2014) 309-320.

[29] W.Y. Lee, S.J. Hong, J. Kim, On-line scheduling of scalable real-time tasks on multiprocessor systems, Journal of Parallel and Distributed Computing 63(12)(2003) 1315-1324.

[30] N.D. Doulamis, A.D. Doulamis, E.A. Varvarigos, Fair scheduling algorithms in grids, IEEE Transactions on Parallel and Distributed Systems 18(11)(2007) 1630-1648.

[31] K. Deb, S. Agrawal, A. Pratap, A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II, in: Proc. 2000 International Conference on Parallel Problem Solving from Nature, 2000.

[32] Q. Zhang, H. Li, MOEA/D: A multiobjective evolutionary algorithm based on decomposition, IEEE Transactions on Evolutionary Computation 11(6)(2007) 712-731.

[33] N. Srinivas, K. Deb, Muiltiobjective optimization using nondominated sorting in genetic algorithms, Evolutionary Computation 2(3)(1994) 221-248.

[34] M. Shang, Optimal algorithm for scheduling large divisible workload on heterogeneous system, Applied Mathematical Modelling 32(9)(2008) 1682-1695.