Journal of Computers Vol. 29 No. 4, 2018, pp. 217-229
doi:10.3966/199115992018082904017

Task Scheduling Model and Multi-objective Optimization Genetic Algorithm Considering Quality of Service

Shanshan Hao, Yuping Wang*, Hejun Xuan

School of Computer Science and Technology, Xidian University, Xi'an, 710071, Shaanxi, China
sshanhao@163.com, ywang@xidian.edu.cn, xuanhejun0896@126.com

Received 11 April 2017; Revised 13 August 2017; Accepted 6 October 2017

Abstract. Divisible task scheduling is a famous NP-hard problem. Most existing works aim at minimizing the completion time, i.e., minimizing the makespan as their single target. However, in practice, there are usually more than one objectives needed to be optimized. In this paper, we introduce the quality of service, and use it as another optimization objective except for optimizing the makespan. As a result, we set up a new optimization model: multi-objective optimization model, which is more practical and more reasonable. To resolve this multi-objective optimization model efficiently, a novel genetic algorithm based on MOEA/D is proposed, in which two new crossover operators and a specific-designed mutation operator are put forward. To demonstrate the effectiveness and efficiency of the proposed model and algorithm, a series of experiments are conducted, and the experimental results indicate the effectiveness and efficiency of the proposed model and algorithms.

Keywords: bi-objectives, divisible task scheduling, genetic algorithm, quality of service

1 Introduction

With the advent of the era of big data, how to effectively use the parallel and distributed system for large data processing has gradually become a hot issue of scientific research. Among them, the task scheduling in distributed system [1-2] is a typical method to realize the efficient processing of large data. Divisible load is a relatively special task, it can be divided into any size of subtasks, and the subtasks are independent respectively. In different network topologies, the model and algorithm based on the theory of divisible task scheduling have been studied extensively, including bus topology [7-8, 21], tree topology [4, 22], b-Ary Tree topology [23], and so on. How to establish a reasonable and efficient task scheduling model for large data processing system and design an efficient task allocation scheme is a difficult problem in the research field of task scheduling. An increasing number of literatures have studied the optimal solution of the scheduling scheme under various network topologies. In paper [3-5], the optimal solution of task completion time is given for linear network, tree type network and star network topology respectively. Without taking into account the computational start-up overhead of the processor and the start-up of the network communication, the study of the literature [6] proves that the optimal scheduling sequence is only related to the communication rate between the processors. The paper [7-8] have proved that to have the shortest completion time in the bus network with constant startup overhead, the load should be scheduled by the sequence of the calculation rate of decline. For an arbitrary computing start-up overhead and network communication startup overhead in distributed heterogeneous environment, the study of the literature [9] shows that if the workload is large enough in decreasing order according to the nodes of the communication rate, the schedule makes the task of the shortest completion time. In order to achieve the shortest time, a high efficient genetic algorithm is designed in the case of considering the release time of the processor, which is based on the [10]. In the paper [11, 15, 24], the research on the multi-source grid environment, the cloud computing platform, the wireless sensor

* Corresponding Author

network and the real-time environment, the task scheduling model and the effective distribution scheme are studied.

Almost all of the existing research is to find an optimal scheduling scheme, which will be allocated to the processor in accordance with a certain strategy to achieve the shortest time of dealing with the task. But with the improvement of the processor performance, the pursuit of the user's goal is not only to minimize task completion time, but to improve the quality of service during an acceptable period.

In [17], to minimize makespan and cost, a multi-objective work-flow grid scheduling algorithm using ε-fuzzy dominance sort based on discrete particle swarm optimization is proposed. For the sake of minimizing the completion time and utilizing the resources effectively, the objectives are formulated independently, and a novel Evolutionary Multi-Objective (EMO) algorithm by using the pareto dominate is proposed. To minimize makespan and failure probability, [19] developed a matching and scheduling algorithm. There are literature focus on the multi-objective load scheduling [19-20], however, neither of them focuses on the divisible load.

In summary, most of the existing divisible task scheduling models take the minimum makespan as their optimization objective, and few works focus on the system consumption and fault tolerant. However, divisible task scheduling mainly works in distributed network system, today's network service providers (NSP) are offering different levels of service for business according to different requests, so the concept of quality of service is necessary to be introduced to make the problem more practical. As a result, we set up a multi-objective optimization model which is totally different from the existing one-objective models. Because multi-objective objective model is totally different from one-objective models, it is impossible to make the fair comparison between these two kinds of models.

In general, the mainly contributions of this paper are as follows:

(1) Different from the previous works, not only makespan but also quality of service is introduced in the divisible task scheduling problem.

(2) To cope with the divisible task scheduling problem in more practical way, we establish a bi-objective optimization model not only by minimizing the makespan, but also by maximizing the quality of service.

(3) A well-designed genetic algorithm based on MOEA/D, which includes two populations, two new crossover operators and a specific-designed mutation operator, is designed to solve the optimization model.

The rest of the paper is organized as follows: Section 2 introduces related concepts about multi-objective optimization. Section 3 describes the problem and establishes a bi-objective optimization model with the makespan to be minimized and the quality of service to be maximized. The presented genetic operators are described in Section 4. The main framework of genetic algorithm and MOEA/D algorithm are given in Section 5. To evaluate the proposed algorithm, simulation experiments are conducted in Section 6, and the experimental results are analyzed. Conclusions with a summary are drawn in Section 7.

2 Related Concepts of Multi-objective Optimization

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

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

Where Ω is the feasible region of the multi-objective optimization problem. $f_i(x)(i = 1, 2, \cdots m)$ are the objective functions. For most cases, the objectives are conflicting to each other in a multi-objective problem. That is to say, no solution in feasible region can minimize all the objective functions simultaneously. So, multi-objective optimization algorithms are used to find a set of Pareto optimal solutions in the feasible region for a multi-objective optimization problem.

Definition 1 (Pareto domination). Considering the minimization problem of each objective, for the $x_A, x_B \in x_f$, x_B is dominated by the solution x_A (denoted as $x_A \succ x_B$) if and only if formula (2) is satisfied.

$$\forall i = 1,2,\cdots m, f_i\left(x_A\right) \le f_i\left(x_B\right) \wedge \exists j = 1,2,\cdots m, f_j\left(x_A\right) < f_j\left(x_B\right), \quad \textbf{(2)}$$

Where x_f is the feasible region for the multi-objective optimization problem.

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

$$\neg\exists x \in x_f : x \succ x^*. \quad \textbf{(3)}$$

Definition 3 (Pareto optimal solution set). Pareto optimal solution set which includes all the Pareto optimal solution is a set. It can be defined as follows:

$$P^* \triangleq \left\{x^* \middle| \neg\exists x \in x_f : x \succ x^*\right\}. \quad \textbf{(4)}$$

Definition 4 (Pareto front). Pareto front is a surface which consists of all the objective function vectors that obtained by all the solutions in Pareto optimal solution set. It can be defined as follows:

$$PF^* \triangleq \left\{F\left(x^*\right) = \left(f_1\left(x^*\right), f_2\left(x^*\right), \cdots, f_m\left(x^*\right)\right)^T \middle| x^* \in P^*\right\}. \quad \textbf{(5)}$$

3 Task Scheduling Optimization Model Considering Quality of Service and Makespan

3.1 System Model

In this paper, the platform used is a heterogeneous distributed system. All the slave processors and the master processor are connected in a star topology. The system is consist of a head node (master processor), denoted as P_0, and the slave processors are denote by $\{P_1, P_2, \ldots, P_N\}$, where N is the number of slave processors. P_0 connects with slave processors by communication links $\{l_1, l_2, \ldots, l_N\}$, where l_i is the communication link between processor P_0 and P_i. Each processor $P_i (i = 1,2,\ldots,N)$ is associated with a speed index ω_i, which is the time taken to process a unit workload on processor P_i. l_i is associated with a speed or bandwidth index g_i, and g_i is the time taken to communicating a unit workload over the link. The system model assumes a typical distributed environment in which the head node does not participate in computation. The role of the master processor is to accept incoming loads, execute the scheduling algorithm, divide the workload and distribute load chunks to the salve processor nodes. Finally, we assume that each computing node has adequate data storage to store and compute any amount of tasks. The master processor divides the incoming load into n $(n \le N)$ load fractions, denoted as $\alpha_1, \alpha_2, \cdots \alpha_n$, and distributes the fractions among the n-processors in a special sequence. Therefore, we have

$$\sum_{i=1}^{n} \alpha_i = W_{total}, \quad \textbf{(6)}$$

Where W_{total} is the entire workload. When the slave processors receive the load fractions, the slave processors start computing their respective load fractions at once. For the sake of minimizing the makespan and maximizing the quality of service, the problem is then to determine the optimal sizes of these load fractions that are distributed to the slave processors. In addition, the special sequence of the load fractions distributed to each slave processor should be determined too.

3.2 Proposed Model

In this section, a multi-objective optimization model will be given. The master processor divides the incoming load into n $(n \le N)$ load fractions, denoted as $\alpha_1, \alpha_2, \cdots \alpha_n$, and distributes the fractions among the n-processors in a special sequence. When the slave processors receive the load fractions, the slave

processors start computing their respective load fractions at once. The time diagram of divisible load scheduling on heterogeneous distributed systems is shown in Fig. 1.

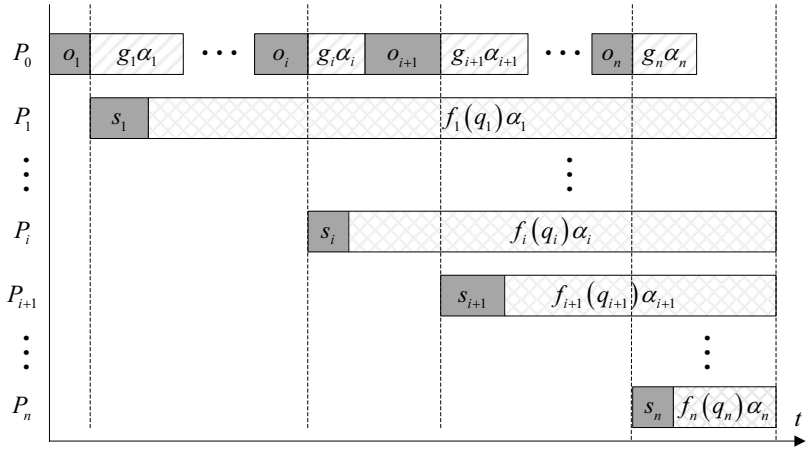

Fig. 1. Task execution time

By [2], only when all processors complete the calculation at the same time, the completion time of the task is the shortest. So we have $T_i = T_j \ (i \neq j)$, where T_i is the completion time of the processor $P_i (i = 1,2,\cdots,n)$ dealing with tasks. As $T_i = T_{i+1}$, we can get:

$$s_i + \alpha_i f_i(q_i) = g_i\alpha_i + o_{i+1} + s_{i+1} + \alpha_{i+1} f_{i+1}(q_{i+1}), \tag{7}$$

where $i = 1,2,\cdots,n-1$, let

$$\eta_i = \begin{cases} 1 & i = 1 \\ \prod_{k=2}^{n} \mu_k & i \geq 2 \end{cases} \quad \gamma_i = \begin{cases} 0 & i = 1 \\ (w_1 - g_1)/w_2 & i = 2 \\ \sum_{k=2}^{i} \left(v_k \prod_{j=k+1}^{i} \mu_j \right) & i \geq 3 \end{cases}, \tag{8}$$

Where $\mu_i = (w_{i-1} - g_{i-1})/w_i$, $v_i = (s_{i-1} - (o_i + s_i))/w_i$, $(i = 2,3,\cdots,n)$. Then the amount of tasks α_i assigned to the processor P_i and α_i can be expressed as formula (9) and the completion time of the task can be expressed in the Eq.(10).

$$\alpha_i = \eta_i\alpha_1 + \gamma_i, i = 2,3,\cdots,n, \tag{9}$$

$$T = T_1 = o_1 + s_1 + f_1(q_1)\alpha_1 = o_1 + s_1 + f_1(q_1)\left(W_{total} - \sum_{k=2}^{n}\gamma_k\right)\Big/\left(1 + \sum_{k=2}^{n}\eta_k\right). \tag{10}$$

Among the $n(n \leq m)$ processors that execute the tasks, if the processor $P_i (i = 1,2,\cdots,n)$ is executing the task at the quality of service level of $q_i \left(q_{lower} \leq q_i \leq q_{upper}\right)$, where q_{lower} and q_{upper} are the minimum and maximum of the quality of service, respectively. the quality of service workload is the minimum of all the quality of service level.

$$Q = \min_{1 \leq i \leq n} \{q_i\}. \tag{11}$$

The main objective of this study is to maximize the quality of service while minimizing the completion time of the task. From the formula (9) and the formula (10), we can know that the quality of service is mainly determined by the quality of service level of the processor and the completion time of the task depends on the order in which the task is assigned to the processor. Therefore, the optimization model of maximizing the quality of service and minimizing the makespan in divisible task scheduling is established as the formula (12).

$$
\begin{cases}
\max\ Q = \max\left\{\min\limits_{1\le i\le n}\{q_i\}\right\} \\
\min\ T = \min\left\{\max\limits_{1\le i\le n}\{T_i\}\right\} \\
s.t. \\
\quad (1) T = o_1 + s_1 + f_1(q_1)\alpha_1 \\
\quad (2)\ \alpha_1 = \left(W_{total} - \sum_{k=1}^{n}\gamma_k\right)\Big/\left(1+\sum_{k=1}^{n}\eta_k\right) \\
\quad (3)\alpha_i = \eta_i\alpha_1 + \gamma_i,\quad i = 2,3,\cdots,n;\ n \le m; \\
\quad (4)\eta_i = \begin{cases} 1 & i = 1 \\ \prod_{k=1}^{n}\mu_k & i \ge 2 \end{cases} \\
\quad (5)\gamma_i = \begin{cases} 0 & i = 1 \\ (w_1 - g_1)/w_2 & i = 2 \\ \sum_{k=1}^{i}\left(v_k\prod_{j=k+1}^{i}\mu_j\right) & i \ge 3 \end{cases} \\
\quad (6)\alpha_i > 0;\ i = 2,3,\cdots,n; \\
\quad (7) q_{lower} \le q_i \le q_{upper},\ i = 1,2,\cdots,n;
\end{cases}
\tag{12}
$$

4 Multi-objective Optimization Genetic Algorithm

Genetic algorithm has been widely used in the practical application of engineering technology, especially to solve the problem of NP combinatorial optimization, which shows its excellent performance [16]. Therefore, this paper designs a new multi-objective global optimization genetic algorithm to solve the task scheduling model which can minimize the makespan and maximize the quality of service.

4.1 Encoding and Decoding

Encoding. The purpose of encoding is to map the solution from the problem domain to a chromosome representation. A good encoding scheme is very significant for describing the problem and solving the problem, which can limit the search space so that the solution converges quickly to the global optimum. In this paper, both the transmission sequence and quality of service are needed to be encoded. The two are using real coding method.

For task assignment order, the sequential encoding method is used. Let $C = (c_1, c_2, \cdots, c_n)$ represents a possible allocation sequence as the relative order of the processor that currently haven't been allocated. A simple example will used to explain the method. Assuming that there are 4 processors, their codes are 1, 2, 3, 4 respectively. First, assuming that the processor numbered 3 is selected, and its relative position is 3, so it is encoded as 3. Then the remaining processors are [1, 2, 4], and then selecting the processor numbered 2, its relative position is 2 now, and so it is encoded as 2. At this point, the processors [1, 4] are left, and if the processor 4 is selected, its relative position is 2, and it is coded as 2. In the end, only the processor numbered 1 is left, and it is encoded as 1. The final coding sequence is [3, 2, 2, 1].

For the corresponding quality of service of each processor, the scope of the service level in this paper is from 1 to 9, so it generates a float number between 1 and 9 as the corresponding quality of service for each processor.

Decoding. Still take the four processors as an example. First, producing an ordered sequence $L = [1, 2, 3, 4]$, whose length is same as the number of processors. While decoding the *ith* code of $C = (c_1, c_2, \cdots, c_n)$, the first is to find the c_i position in the sequence L of the elements. The decoding process will be explained by a simple example. Let the encoding sequence assume to be $C = [3, 2, 2, 1]$. As $c_1 = 3$ which is in the third position in L, so the decoded value is 3, then deleting the number in the

third positions, now $L=[1,\ 2,\ 4]$. As $c_2=2$, and the number $L=[1,\ 2,\ 4]$ of the second positions is 2, so the decoded value is 2, deleting 2 in L, then the $L=[1,\ 4]$. According to this method, when L is empty, the decoding sequence $S=[3,\ 2,\ 4,\ 1]$ can be obtained.

As quality of service is directly encoded as a possible service level, the quality of service is not needed to be decoded.

4.2 Crossover Operator

Crossover operator is an important operator in the genetic algorithm. Offspring in obtaining parental good genes can produce better individuals. In this paper, two crossover operators have been designed to improve the search ability of the algorithm by using the genetic algorithm to solve the model.

Crossover operator 1. First, the two parents that used to participate in the crossing process should be chosen. After the step, a (0,1) sequence should be generated randomly which has the same length with the encoded parents. Then the crossover operator works bit by bit.

If and only if the bit is 1, then the corresponding gene can participate in the crossover process. If cross operation can be performed, the bits in parental individual one and parental individual two can be respectively seen as the axis of coordinates x and coordinates y, which together determine the position of a point in a two-dimensional Cartesian coordinate system. The next is to randomly generate an integer θ between 0 and 360 as the rotation angle, (x,y) will rotate θ around the origin $(0,0)$ of the degree and get the new coordinates (x',y') as shown in Fig. 2. As the code should be positive, absolute values must be taken for the coordinates (x',y'), and (x'',y'') is obtained. The third is to replace the original x by $x"$, the original y by y''.

It should be noted that there may be some genes that no longer meet the coding rule, in this case modification is needed. The correction method is to randomly generate a feasible data within the range to replace the inappropriate one.

When talking about quality of service, as the quality of service is encoded as the real number between q_{lower} and q_{upper}, it is only needed to exchange the corresponding genes in the two parents, if the bit is 1.

Crossover operator 2. Two parents' corresponding genes can be seen as the axis of coordinates x and coordinates y, which together determine the coordinates of a point. And the point can be seen as the center of a circle whose radius is 1. Then two points are selected randomly. Now they can be seen as a coordinates that is composed of the corresponding gene of the two new offspring. Since the new offspring are located in the vicinity of the parents, it can meet the requirements of the local correction. The new offspring generated by this crossover operator can satisfy the constraint conditions.

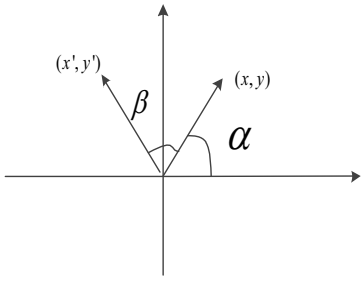

Fig. 2. Crossover operator 1

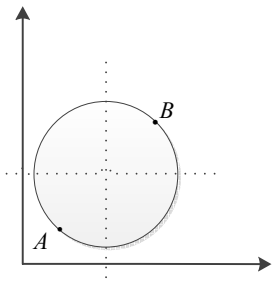

Fig. 3. Crossover operator 2

4.3 Mutation Operator

Mutation operator can produce new individuals, and the purpose is to prevent premature convergence and fall into local optimum. An excellent mutation operator can help to obtain the global optimal solution.

The process of mutation operator is as follows: first, randomly selecting an individual as the parent, and then generating a number within the acceptable range as the gene to take part in mutation. Then the process for mutating scheduling sequence ends.

For the quality of service, as the goal is to make the quality of service as large as possible, the first step is to get the average value of all individuals' quality of service in the current population. For a certain individual, if its quality of service is less than the average, then the value should be added one. If the value is greater than the average, then mutation operator degenerates: a random number between the lowest quality of service and the highest quality of service is chosen to replace the current quality of service. If the current quality of service is more than the upper limit, the quality of service is to take the maximum value.

4.4 Selection Operator

The selection operator defines the generation strategy of the next generation. An appropriate algorithm can select excellent individuals into the next generation while eliminating the poor individuals, and ensure that the entire population is iterated to the optimal solution. The elitist retention strategy and roulette strategy are adopted in this paper.

5 An Effective Genetic Algorithm Based on MOEA/D

In this section, the framework of genetic algorithm and the framework of multi-objective optimization algorithm are given. The framework of genetic algorithm is shown in algorithm 1, and the framework of MOEA/D algorithm is shown in algorithm 2.

Algorithm 1: *A genetic algorithm for task scheduling model*

Step 1: (Initializing) Randomly generating initial population $P(0)$ according to coding rules. Evolutionary algebra $t = 0$.

Step 2: (Crossover) By the probability of p_c, selecting two parents for crossover operation from $P(t)$, the offspring of the individual set is defined as O_1.

Step 3: (Mutation) By the probability of p_m, selecting a parent for mutation operation from $P(t)$, the offspring of the individual set is defined as O_2.

Step 4: (Selection) Selecting the individuals in the external file EP directly to the next generation of population to speed up the convergence rate if EP is no more than the size of the population, Otherwise, the algorithm of roulette wheel is used to select E individual from EP. For the rest of the (N-E) individuals, the same operation is used to select enough individual to next generation from the set of $p(t) \cup O_1 \cup O_2$.

Step 5: (Stopping Criteria) If the termination condition is satisfied, the algorithm terminates; otherwise, go to step 2.

Algorithm 2: *Multi-objective global optimization genetic algorithm*

Step 1: (Initializing) Initializing the external file, the weight vector, determining the distance between the individual and its neighbors, initializing the positive point of each target and choosing the decomposition algorithm of multi-objective optimization.

Step 2: (Updating) According to the nearest neighbor of each individual, selecting two parent individuals who participate in the evolutionary computation for crossover and mutation operation. Updating the positive value point, each individual's nearest neighbor and external file.

Step 3: (Stopping Criteria) If the termination criteria is satisfied, the algorithm terminates; otherwise, go to step 2.

6 Experiment Results and Analysis

6.1 Experiment Parameters

The parameter data of the processor in the distributed environment is decided by the data provided in the literature [9], as shown in table 1. In the genetic algorithm, the following parameters are used: the number of neighbors of each individual is $Neighbor = 10$, population size is $N = 100$, crossover probability is $p_c = 0.6$, mutation probability is $p_m = 0.1$, the value of elite reserves is $E = 5$ and of iterations is $t_{\max} = 1000$. According to the linear relationship described in literature [14] and quality of service, $\omega` = \omega \times Qos$ are used to measure the real execution rate when the quality of service are considered, where *Qos* means the quality of service. In the experiments, q_{lower} and q_{upper} are selected as 1 and 9, respectively.

Table 1. Parameters of the processors in heterogeneous system

P	o	s	g	ω
P_1	70.554750	57.951860	0.533424	2.895625
P_2	30.194800	1.401760	0.774740	7.607236
P_3	81.449010	4.535275	0.709038	4.140327
P_4	86.261930	37.353620	0.790480	9.619532
P_5	87.144580	94.955670	0.056237	3.640187
P_6	52.486840	5.350452	0.767112	5.924582
P_7	46.870010	62.269670	0.298165	6.478212
P_8	26.379290	82.980160	0.279342	8.246021
P_9	58.916300	91.096430	0.986093	2.268660
P_{10}	69.511550	24.393140	0.980003	5.338731
P_{11}	10.636970	67.617590	0.999415	1.570390
P_{12}	57.518380	10.302260	0.100052	7.988844
P_{13}	28.448030	29.577290	0.045649	3.820107
P_{14}	30.090500	97.982940	0.948571	4.013743
P_{15}	27.828000	16.282160	0.160442	6.465871

6.2 Experiment Results and Analysis

In this paper, a task scheduling model with the objective of minimizing the makespan and maximizing the quality of service is established, and the genetic algorithm with new crossover and mutation operators are designed. In order to verify the above model and algorithm, a series of experiments have been carried out.

In the experimental part, the workload is of the size of 100 to 9900. The genetic algorithm based on MOEA/D proposed in this paper is expressed by Mul-GA. In [9], an algorithm is proposed according to the increment of g_i, denoted as IG, and [8] proposed an algorithm to allocate the workload according to the increment of w_i, denoted as IW. Because both IG and IW are aimed at minimizing the completion time of the task, the quality of service is not taken into account and there have not been relevant literature that takes the quality of service into consideration, only makespan of the three algorithms are compared in this part. As shown in Table 2.

Table 2. makespan comparison

Algo	W_{total}	Makespan	Algo	W_{total}	Makespan	Algo	W_{total}	Makespan
Mul-GA		183.5	Mul-GA		2103.7	Mul-GA		3721.6
IG	100	283.6	IG	3500	2156	IG	6900	3766.3
IW		235.2	IW		3697.1	IW		6809.6
Mul-GA		440.3	Mul-GA		2234.1	Mul-GA		3841.6
IG	300	471.1	IG	3700	2250.7	IG	7100	3861.1
IW		499.1	IW		3883.5	IW		6990.5
Mul-GA		521.1	Mul-GA		2301.4	Mul-GA		3933.1
IG	500	599.4	IG	3900	2345.5	IG	7300	3955.8
IW		729.5	IW		4069.9	IW		7171.1
Mul-GA		681.8	Mul-GA		2411.4	Mul-GA		4041.9
IG	700	718	IG	4100	2345.5	IG	7500	4050.6
IW		959.9	IW		4069.9	IW		7351.5
Mul-GA		827.5	Mul-GA		2503.1	Mul-GA		4146.8
IG	900	835.8	IG	4300	2543.9	IG	7700	4145.3
IW		1190.3	IW		4442.7	IW		7532.1
Mul-GA		948.9	Mul-GA		2583.1	Mul-GA		4201.8
IG	1100	952.5	IG	4500	2629.6	IG	7900	4240
IW		1392.6	IW		4624.8	IW		7711.7
Mul-GA		1012.9	Mul-GA		2684.9	Mul-GA		4301.2
IG	1300	1067.6	IG	4700	2724.4	IG	8100	4334.1
IW		1601.5	IW		4807.8	IW		7891.3
Mul-GA		1146.3	Mul-GA		2783.1	Mul-GA		4401.3
IG	1500	1174.4	IG	4900	2819.1	IG	8300	4429.5
IW		1793.6	IW		4990.8	IW		8070.8
Mul-GA		1251.3	Mul-GA		2901.8	Mul-GA		4513.7
IG	1700	1279.5	IG	5100	2913.8	IG	8500	4524.2
IW		1988.2	IW		5173.8	IW		8250.3
Mul-GA		1327.6	Mul-GA		2998.7	Mul-GA		4604.9
IG	1900	1384.7	IG	5300	3008.5	IG	8700	4618.9
IW		2182.7	IW		5355.9	IW		8429.8
Mul-GA		1437.4	Mul-GA		3013	Mul-GA		4700.4
IG	2100	1489.8	IG	5500	3103.3	IG	8900	4713.7
IW		2377.3	IW		5537.7	IW		8609.4
Mul-GA		1557.3	Mul-GA		3170.2	Mul-GA		4789.2
IG	2300	1587.6	IG	5700	3198	IG	9100	4808.4
IW		2568.2	IW		5719.4	IW		8788.9
Mul-GA		1652.3	Mul-GA		3249	Mul-GA		4903.1
IG	2500	1682.4	IG	5900	3292.7	IG	9300	4993.1
IW		2759.2	IW		5901.1	IW		8968.4
Mul-GA		1767.7	Mul-GA		3382.1	Mul-GA		4913.4
IG	2700	1777.1	IG	6100	3387.5	IG	9500	4997.9
IW		2947.7	IW		6082.8	IW		9147.9
Mul-GA		1854.6	Mul-GA		3460.1	Mul-GA		5071.2
IG	2900	1871.8	IG	6300	3482.2	IG	9700	5092.6
IW		3136.1	IW		6264.5	IW		9327.5
Mul-GA		1927.6	Mul-GA		3551.1	Mul-GA		5113.4
IG	3100	1966.6	IG	6500	3576.9	IG	9900	5187.3
IW		3324.3	IW		6446.2	IW		9507
Mul-GA		2003.6	Mul-GA		3641.2			
IG	3300	2061.3	IG	6700	3671.7			
IW		3510.7	IW		6627.9			

It can be seen from Table 2 that the processing time used by the proposed algorithm is much less than that used by the other two compared algorithms for different workload.

To evaluate the performance of the genetic algorithm based on MOEA/D, the workload size ranges from 100 to 9900 and the pareto front are given in Fig. 4. In addition, the following two metrics are used

to evaluate the pareto solution:

Spacing Index (SI). defined by Eq.(13) below.

$$\begin{cases} SI(A) = \sqrt{\dfrac{1}{|PF^*| - 1} \sum_{z \in PF^*} \left(\bar{d} - d(z)\right)^2} \\ d(z) = \min\left\{ \|z - z'\| \,\middle|\, z \neq z', z' \in PF^* \right\} \\ \bar{d} = \dfrac{1}{|PF^*|} \sum_{z \in PF^*} d(z) \end{cases} \tag{13}$$

Spacing Index is used to metric the uniformly of the pareto front. The smaller the *SI* is, the better the solution is.

Hypervolume Index (HI). which is used to test the uniformity, convergence and diversity of the solution, and is defined by the following Eq.(14).

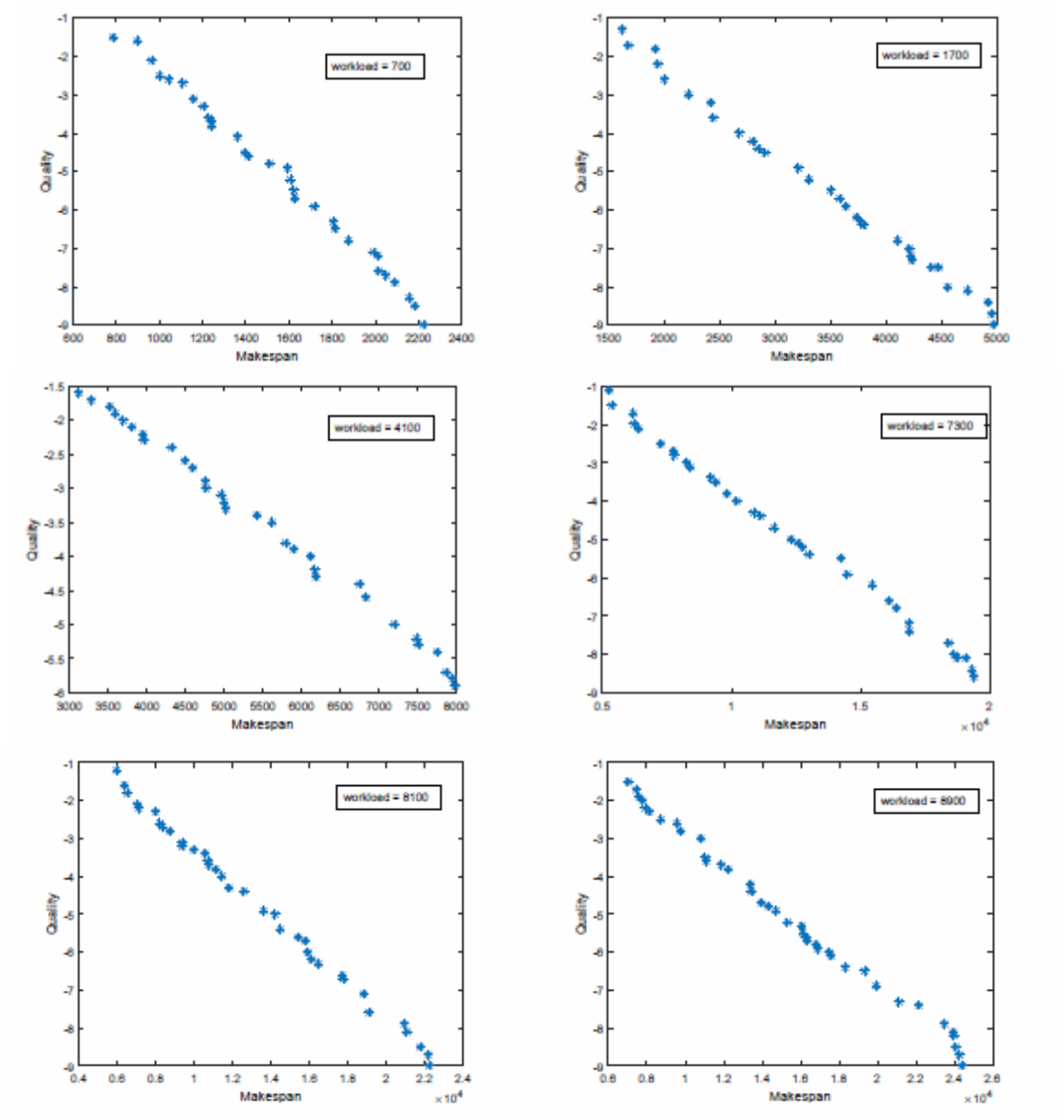

Fig. 4. multi-objective Pareto front for different workload

$$HI\left(PF^{*}\right)=\bigcup_{z\in PF^{*}} vol\left(z\right) \tag{14}$$

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

Table 3 shows the two evaluate results of the pareto optimal solution obtained by algorithm Mul-GA. $\lg\left(SI\right)$ and $\lg\left(HI\right)$ are used in Table 3.

Table 3. Evaluation index

Workload	100		1100	
Metrics	SI	HI	SI	HI
Value	3.4062	8.1422	6.6084	9.3298
Workload	2100		3100	
Metrics	SI	HI	SI	HI
Value	7.3776	9.5965	8.3100	10.1302
Workload	4100		5100	
Metrics	SI	HI	SI	HI
Value	8.4207	10.3840	10.0516	11.3101
Workload	6100		7100	
Metrics	SI	HI	SI	HI
Value	10.1482	10.8685	10.1998	11.2102
Workload	8100		9100	
Metrics	SI	HI	SI	HI
Value	10.4080	11.7733	10.1435	11.2430

By combining the pareto front in Fig. 4 and the evaluate results in Table 3, the following conclusions can be drawn: the model and algorithm proposed in this paper can solve the multi-objective optimization problem. By running the algorithm once, multiple sets of solutions can be obtained, and the evaluate metrics shows the diversity and uniformity of solutions.

Several experiments are conducted to improve the effectiveness of the algorithm proposed in this paper. In the first experiment, as there have no literatures that have taken both the makespan and quality of service into consideration, so Table 2 gives only the makespan comparison. The makespan obtained by Mul-GA are much smaller than those obtained by IG and IW with various workload. In the second experiment, makespan and the quality of service are both considered, and the pareto fronts obtained by Mul-GA in Fig. 4 are wide spread and uniformly distributed according to the SI and HI index in Table 3, what's more, multiple sets of solutions can be obtained in one time, so it is more possible to satisfy the requirements of decision makers.

7 Conclusion

We first introduce a concept called quality of service which provides another measure for a task schedule, and then we set up a multi-objective optimization model by minimizing the make-span and maximizing the quality of service. This model contains more information of the problem considered and can more fit to the real world situations. Thus it is more practical and more reasonable. To solve the proposed multi-objective model efficiently, we design a new genetic algorithm. The experimental results show that: the proposed model is effective and the proposed algorithm is efficient.

Acknowledgments

This work was supported by National Natural Science Foundation of China (No. 61472297 and No. 61572391).

References

[1] V. Bharadwaj, D. Ghose, T.G. Robertazzi, Divisible load theory: a new paradigm for load scheduling in distributed systems, Cluster Computing 6(1)(2003) 7-18.

[2] T.G. Robertazzi, Ten reasons to use divisible load theory, Computer 36(2003) 63-68.

[3] V. Mani, D. Ghose, Distributed computation in linear networks: closed form solutions, IEEE Transactions on Aerospace and Electronic Systems 30(1994) 471-483.

[4] D. Ghose, V. Mani, Distributed computation with communication delays: asymptotic performance analysis, Journal of Parallel and Distributed Computing 23(1994) 293-305.

[5] V. Bharadwaj, D. Ghose, V. Mani, Optimal sequencing and arrangement in distributed single-level networks with communication delays, IEEE Trans on Parallel and Distributed Systems 5(1994) 968-976.

[6] H.J. Kim, G.I. Jee, J.G. Lee, Optimal load distribution for tree network processors, IEEE Trans on Aerospace and Electronic Systems 32(2)(1996) 607-612.

[7] S. Suresh, V. Mani, S.N. Omkar, The effect of start-up delays in scheduling divisible load on bus networks: an alternate approach, Journal of Computational and Applied Mathematics 46(10-11) 1545-1557.

[8] V. Bharadwaj, X.L. Li, C.K. Chung, On the influence of start-up costs in scheduling divisible load on bus networks, IEEE Trans on Parallel and Distributed Systems 11(12)(2000) 1288-1305.

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

[10] X.L. Wang, Y.P. Wang, K. Meng, Release time aware divisible-load scheduling optimization model, Journal of XiDian University 43(1)(2016) 55-60.

[11] G. Murugesan, C. Chellappan, Multi-source task scheduling in grid computing environment using linear programming, International Journal of Computer Science and Engineering 9(1)(2014) 80-85.

[12] G.N. Iyer, B. Veeravalli, S.G. Krishnamoorthy, On handling large-scale polynomial multiplication in compute cloud environments using divisible load paradigm, IEEE Transactions on Aerospace and Electronic Systems 48(1)(2012) 820-831.

[13] H. Shi, W. Wang, N.M. Kwok, et al, “Adaptive Indexed Divisible Load Theory for Wireless Sensor Network Workload Allocation,” International Journal of Distributed Sensor Networks, Vol. 2013, pp. 8-10, 2013.

[14] X.M. Zhu, X. Qin, M.K. Qiu, QoS-aware fault-tolerant scheduling for real-time tasks on heterogeneous clusters, IEEE Transactions on Computers 60(6)(2011) 800-812.

[15] M. Hu, B. Veeravalli, Requirement-aware strategies for scheduling real-time divisible loads on clusters, Journal of Parallel and Distributed Computing 73(8)(2013) 1083-1091.

[16] A. Costa, F.A. Cappadonna, S. Fichera, A novel genetic algorithm for the hybrid flow shop scheduling with parallel batching and eligibility constraints, The International Journal of Advanced Manufacturing Technology 75(5-8)(2014) 833-847.

[17] R. Garg, A.K. Singh, Multi-objective workflow grid scheduling using fuzzy dominance sort based discrete particle swarm optimization, The Journal of Supercomputing 68(2)(2014) 709-732.

[18] C. Grosan, A. Abraham, B. Helvik, Multiobjective evolutionary algorithms for scheduling jobs on computational grids, in International Conference on Applied Computing, 2007.

[19] Atakan Dogan, Füsun Özgüner, Biobjective scheduling algorithms for execution time? reliability trade-off in heterogeneous computing systems, The Computer Journal 48(3)(2005) 300-314.

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

[21] D. Liu, X. Yang, Z. Cheng, An energy-aware scheduling algorithm for divisible loads in a bus network, Concurrency & Computation Practice & Experience 28(5)(2016) 1612-1628.

[22] S. Ghanbari, M. Othman, M.R.A. Bakar, W.J. Leong, Multi-objective method for divisible load scheduling in multi-level tree network, Future Generation Computer Systems 54(2016) 132-143.

[23] C.Y. Chen, C.P. Chu, Novel methods for divisible load distribution with start-up costs on a complete b-ary tree, IEEE Transactions on Parallel & Distributed Systems 26(2015) 2836-2848.

[24] L. Zeng, B. Veeravalli, A.Y. Zomaya, An integrated task computation and data management scheduling strategy for workflow applications in cloud environments, Journal of Network & Computer Applications 50(2015) 39-48.