Information Sciences 379 (2017) 241–256

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Bi-objective workflow scheduling of the energy consumption and reliability in heterogeneous computing systems

Longxin Zhang[a,b], Kenli Li[a,*], Changyun Li[b], Keqin Li[a,c]

[a] College of Computer Science and Electronic Engineering, National Supercomputing Center in Changsha, Hunan University, Changsha,410082, China
[b] College of Computer and Communication, Hunan University of Technology, Zhuzhou,412007, China
[c] Department of Computer Science, State University of New York, New Paltz, NY 12561, USA

ARTICLE INFO

Article history:
Received 25 November 2015
Revised 28 July 2016
Accepted 2 August 2016
Available online 3 August 2016

Keywords:
Multi-objective
Precedence constraints
Reliability
Scheduling
Workflow

ABSTRACT

Recent studies focus primarily on low energy consumption or execution time for task scheduling with precedence constraints in heterogeneous computing systems. In most cases, system reliability is more important than other performance metrics. In addition, energy consumption and system reliability are two conflicting objectives. A novel bi-objective genetic algorithm (BOGA) to pursue low energy consumption and high system reliability for workflow scheduling is presented in this paper. The proposed BOGA offers users more flexibility when jobs are submitted to a data center. On the basis of real-world and randomly generated application graphs, numerous experiments are conducted to evaluate the performance of the proposed algorithm. In comparison with excellent algorithms such as multi-objective heterogeneous earliest finish time (MOHEFT) and multi-objective differential evolution (MODE), BOGA performs significantly better in terms of finding the spread of compromise solutions.

1. Introduction

Modern data centers consume a tremendous amount of power and produce a large amount of pollution. The Tianhe-2, a heterogeneous computing system, is one example, which is located at the National Supercomputer Center in Guangzhou (China). It is the worlds fastest supercomputer as of July 13, 2015. The power consumption of Tianhe-2 is 17,808 kW [1]. Assuming that the electricity charge is 1 yuan per kilowatt hour, the electricity cost of Tianhe-2 is approximately 156 million yuan (USD 25 million) per year. In addition, high energy consumption also has a negative effect on system reliability. High-performance heterogeneous computing systems attract the attention of many researchers, who have conducted numerous studies on different service quality of simple objectives based on heterogeneous computing systems; such objectives include minimum completion time, low energy consumption, minimum execution cost, and higher system reliability. Such studies are more challenging when the focus is on more than one objective. Dongarra et al. [9] presented bi-objective scheduling algorithms for optimizing makespan and reliability of heterogeneous systems. They constructed a task graph and transformed a heuristic that targets makespan minimization to a bi-criteria heuristic. Boeres et al. [5] developed an efficient weighted

* Corresponding author. Tel.: +86-731-88664161.
E-mail addresses: longxinzhang@hnu.edu.cn, longxinzhang@hut.edu.cn (L. Zhang), lkl@hnu.edu.cn (K. Li), lcy469@126.com (C. Li), lik@newpaltz.edu (K. Li).

http://dx.doi.org/10.1016/j.ins.2016.08.003
0020-0255/

bi-objective scheduling algorithm for heterogeneous systems. This study also introduced a classification of the solutions provided by weighted bi-objective schedulers aimed at helping users to adjust the weighting function such that an appropriate solution can be matched in accordance with different needs.

To devise matching and scheduling algorithms that account for both makespan and failure probability and balance them in a parallel application, Doğan and Özgüner [8] explored the bi-objective scheduling algorithms for execution time-reliability trade-off in heterogeneous computing systems. For this purpose, the bi-objective dynamic level scheduling algorithm is presented by modifying an existing scheduling algorithm, the dynamic-level scheduling [24] algorithm. With respect to workflow scheduling in heterogeneous environment, Fard et al. [13] proposed a four-objective case study that comprises makespan, economic cost, energy consumption, and reliability based on ASKALON environment for grid and cloud computing. Jeannot et al. [17] explored the optimization of performance and reliability on heterogeneous parallel systems. For scheduling independent tasks, constrained minimum lambda tau was designed. For scheduling tasks with precedence constraints, a general Pareto front approximation was devised. On the basis of multi-objective swarm algorithms, Arsuaga-Ríos et al. [3] presented the multi-objective artificial bee colony and the multi-objective gravitational search algorithm in search of a bi-objective optimization for both execution time and cost. Durillo et al. [10] proposed a novel multi-objective heterogeneous earliest finish time (MOHEFT) list-based workflow scheduling algorithm, which is an extension of the heterogeneous earliest finish time (HEFT) [26] algorithm. In the beginning of the MOHEFT, high-quality initial populations are generated under the state-of-the-art HEFT strategy. The MOHEFT is capable of computing a set of tradeoff optimal solutions in terms of makespan and energy efficiency.

Bhuiyan et al. [4] devised excellent strategies in deploying wireless sensor networks with fault tolerance for structural health monitoring. Wang et al. [12] proposed a look-ahead genetic algorithm (LAGA) which utilizes reliability-driven reputation to optimize the makespan and the reliability of a workflow application. Xu et al. [31] presented a hybrid chemical reaction optimization scheme for task scheduling on heterogeneous computing systems. Zhang and Li [33] investigated the energy minimization problem for real-time software systems with task execution time following a statistical distribution. Huang et al. [16] presented novel heuristic speculative execution strategies in a cluster that runs Hadoop MapReduce. The non-dominated sorting genetic algorithm (NSGA-II) was presented by Deb et al. [28] in 2002. Zhang et al. [34] developed novel algorithms to maximize system reliability with energy constraint in heterogeneous systems. Xu et al. applied the conventional chemical reaction optimization framework in [30] and used a multiple priority queues genetic algorithm in [32] to address the DAG scheduling problem on heterogeneous computing systems. Given its relatively low time complexity and rapid convergence rate, NSGA-II is used widely in many fields, such as industry, transportation, and finance. In addition, Penmatsa and Chronopoulos designed a load-balanced scheduling strategy for the execution of job flows in heterogeneous-distributed systems on the basis of multi-objective optimization [21,22]. Penmatsa et al. presented a load-balanced scheduling algorithm for the execution of loop-level tasks in grids [23].

However, none of the above studies consider low energy consumption and high system reliability at the same time. These two metrics are essential parts of modern green computing. Modern data centers need to provide a set of solutions with different performance metrics to meet the diverse demands of users, such as high system reliability and low energy consumption in different cases. Hence, the major purpose of this paper is to propose a general approach to solve the bi-objective of low energy consumption and high system reliability on workflow application scheduling in heterogeneous computing systems.

This study addresses the bi-objective optimization problem of high system reliability and low energy consumption for parallel tasks as a combinatorial optimization problem. It includes three nontrivial subproblems, i.e., energy saving, reliability maximizing, and solution selecting. First, appropriate voltage levels are selected for processors so that the schedule length of parallel tasks is acceptable and the total energy consumption is minimal. Second, the selected speed for each parallel task is relatively high so that the total system reliability reaches a high level. Third, as energy saving and system reliability are two conflicting objectives, an effective and efficient solution strategy is devised while selecting the speed and processor for each candidate parallel task.

These problems are efficiently treated to develop an algorithm with an overall good performance. With the adoption of the non-dominated idea and dynamical voltage frequency (DVFS) scaling technology, a novel bi-objective genetic algorithm that aims to achieve low energy consumption and high system reliability is developed.

The major contributions of this paper are listed as follows.

- We show that the problems of minimizing energy consumption and maximizing system reliability under precedence and schedule length constraints are conflicting. A set of non-dominated solutions with different system reliability and energy consumption are provided by BOGA when a user submits a job.
- We develop a class of algorithms to solve the above three nontrivial subproblems. These algorithms are crucial components that constitute BOGA.
- We present an efficient selection strategy that works in the front part of the BOGA loop to determine the non-dominated solutions. The elite individuals are selected as a new population when BOGA finishes each iteration. A fine Pareto front is achieved after the stopping criteria of BOGA are satisfied.
- Through experiments over a large set of randomly generated graphs as well as through the graphs for real-world applications with different characteristics, we prove that the proposed BOGA algorithm outperforms several related multi-objective optimization algorithms in terms of energy consumption and system reliability.

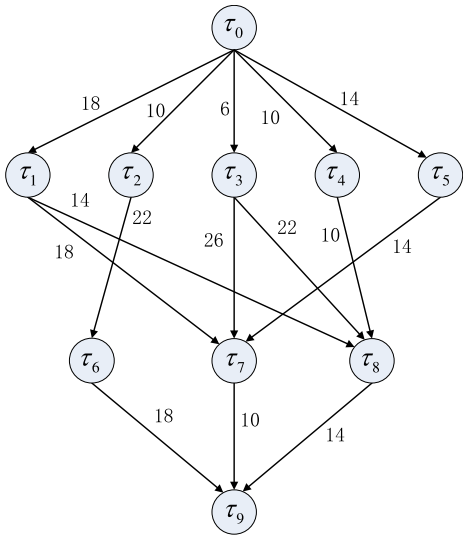

Fig. 1. A simple DAG.

Table 1
Voltage-relative frequency pairs.

Level	Pair 1		Pair 2		Pair 3	
	vol.	fre.	vol.	fre.	vol.	fre.
0	1.75	1.00	1.50	1.00	2.20	1.00
1	1.50	0.80	1.40	0.90	1.90	0.85
2	1.40	0.70	1.30	0.80	1.60	0.65
3	1.20	0.60	1.20	0.70	1.30	0.50
4	1.00	0.50	1.10	0.60	1.00	0.35
5	0.90	0.40	1.00	0.50		
6			0.90	0.40		

The rest of the paper is organized as follows. Section 1 describes relative works on high-performance research in heterogeneous computing systems. Section 2 introduces the models used in this work. Section 3 shows the basic concepts of multi-objective optimization problems. Section 4 presents the algorithms developed in this work. Section 5 evaluates the performance of the compared algorithms. Section 6 gives the conclusion and future works.

2. Models

In this section, the models used in this paper are presented.

2.1. Workflow model

A workflow application is usually modeled as a directed acyclic graph (DAG). A DAG, $W = (T, E)$, where T is a finite set of tasks τ_i $(1 \le i \le n)$ and $E = \{(\tau_i, \tau_j, Data_{ij}) | (\tau_i, \tau_j) \in T \times T\}$ is the set of edges which denote task precedence constrains, where $Data_{ij}$ is the volume of the data transferred from activity τ_i to activity τ_j. $parent(\tau_i) = \{\tau_k | (\tau_k, \tau_i, Data_{ki} \in E)\}$ is used as the predecessor set of activity τ_i. All the activities are completed before τ_i is released. As shown in Fig. 1, each directed arc is associated with a numeric label, which represents data flow. Its value denotes communication cost, which indicates the activity of data flow performed in different processors.

2.2. System model

The system model consists of a set of *PE*, which includes p heterogeneous cores/processors in a cluster. Each processor *pe* supports the DVFS technology wherein each processor works at several different speeds. For simplicity, the present study adopts the processor parameter configuration reported in [19], in which each *pe* has f different available frequency levels (AFLs). Since the frequency switching overhead occupies only an insignificant part of time, it is ignored in the following study. The voltage-frequency pairs used in this study are shown in Table 1. In addition, the communication subsystem explored in this study comprises a set of fully connected processors with each processor communicating with any other processors at any time that is completely free of contention.

2.3. Power model

The classic power model proposed in [35] is adopted to capture the system power in this study:

$$P = P_s + \hbar\left(P_{ind} + P_d\right) = P_s + \hbar\left(P_{ind} + C_{eff} f^{\alpha}\right), \tag{1}$$

where P_s is the static power consumption, P_{ind} is the frequency-independent active power, and P_d is the frequency-dependent dynamic power. Static power includes the power to maintain basic circuits, to keep the clock working, and to keep memory to stay in sleep mode which can be removed only by turning off the whole system. P_{ind} is a constant, independent of system operation frequency (i.e., power consumption occurs while accessing external devices like the main memory, I/O, and so on), that is decreased to a very small value by setting the system to standby mode [6]. P_d is the dynamic power dissipation which is the dominant component of energy consumption in widely popular CMOS technology. It is shown by $P_d = C_{eff} \cdot V_{dd}^2 \cdot f$, where C_{eff} is the effective loading capacitance, V_{dd} is the supply voltage, and f is the clock frequency. Since $f \propto v^{\gamma}$ $(0 < \gamma < 1)$ [20], in other words, $v \propto f^{1/\gamma}$, it is reckoned that the frequency-dependent active power consumption is $P_d \propto f^{\alpha}$, where $\alpha = 1 + 2/\gamma \geq 3$. In the present study, $P_d = C_{eff} f^{\alpha}$ is used. $\hbar$ denotes the system mode and represents the active power consumption that presently occurred. Particularly, $\hbar = 1$ indicates that the system is currently active. Otherwise, $\hbar = 0$ refers to a sleep mode that the system is in. In the context of this paper, all frequencies are normalized with respect to the maximum frequency $f_{\max}$ (i.e., $f_{\max} = 1.0$). The energy consumption of task τ_i can be calculated according to Eq. (2):

$$E_i(f_i) = P_{ind_i} \cdot \frac{c_i}{f_i} + C_{eff} \cdot c_i \cdot f_i^2, \tag{2}$$

where c_i is the computation cost of the task τ_i.

2.4. Reliability model

As an application is running, a fault is hard to avoid due to hardware failure, software bugs, devices that work in high temperature, and so on. Accordingly, transient faults happen more frequently. Based on the previous study [34], the reliability model used in this study can be formalized as follows:

Definition 1. The reliability of a task is the probability of executing the task successfully. If the transient fault follows a Poisson distribution, the reliability of node τ_i with the corresponding computation cost c_i is [35]

$$R_i(f_i) = e^{-\lambda(f_i) \times \frac{c_i}{f_i}}, \tag{3}$$

where f_i denotes the processing frequency, $\lambda(f) = \lambda_0 \cdot g(f) = \lambda_0 \cdot 10^{\frac{d(1-f)}{1-f_{\min}}}$, d and λ_0 are constants.

2.5. Problem definition

The problem addressed in the study consists of scheduling the workflow tasks applications in heterogeneous computing system under certain given constraints. The objectives are to minimize energy consumption and probability of failure (POF; equal to $1 - R$) while satisfying the shared deadline constraint of workflow tasks. This workflow scheduling problem is formalized as follows:

$$\begin{aligned} \text{Minimize:} \quad & F = (F_1, F_2), \\ & F_1 = \min_{\tau_i \in T,\ f_j(\tau_i) \in PE} Energy\left(\tau_i, f_j(\tau_i)\right), \\ & F_2 = \min_{\tau_i \in T,\ f_j(\tau_i) \in PE} POF\left(\tau_i, f_j(\tau_i)\right), \\ \text{subject to:} \quad & makespan(T) < D, \end{aligned} \tag{4}$$

where D is the shared deadline of the entire tasks set T.

To meet the task scheduling requirement, a prior order is established in this phase. Each task is set with its *URank*, which is computed recursively according to the following expression:

$$URank(\tau_i) = \overline{w_i} + \max_{\tau_j \in child(\tau_i)} \left(c_{i,j} + URank(\tau_j)\right), \tag{5}$$

where $child(\tau_i)$ is the set of immediate children of task node τ_i, $c_{i,j}$ is the communication cost between nodes τ_i and τ_j and $\overline{w_i}$ is the average computational time of τ_i when it is executed on different processors. Rank is recursively computed by traversing from the bottom to the top in a DAG. It should be apparent to draw such a conclusion that $URank(\tau_{exit}) = \overline{w_{exit}}$.

3. Multi-objective optimization and a motivational example

In this section, several basic concepts of multi-objective optimization theory are introduced to understand this work better. Moreover, a motivational example is illustrated.

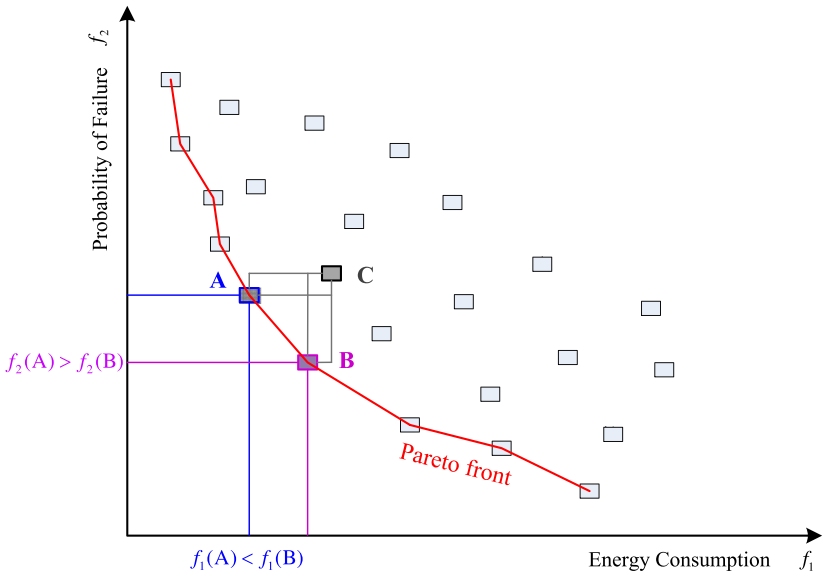

Fig. 2. Multi-objective Optimization.

3.1. Multi-objective optimization problem overview

Since any maximization problem is transformed to a minimization problem, without loss of generality, the goal of all the objectives can be defined as minimization problem. Concepts from the multi-objective optimization (MOP) theory are introduced for a better understanding of this work. Since there are several performance metrics, such as energy saving, makespan, and system reliability in the workflow scheduling problem, it is usually formalized as a multi-objective optimization problem. In most cases, these metrics are a group of conflicting objectives when they are optimized at the same time. For Eq. (4), decreasing energy consumption brings an increment of the probability of failure. Classically, multi-object optimization problem is formally defined as follows:

Definition 2. Given an n-dimensional vector $\vec{S} = [x_1, x_2, x_3, \ldots, x_n]$, MOP is to minimize the following objective function:

$$\vec{F}(\vec{S}) = [f_1(\vec{S}),\ f_2(\vec{S}),\ \ldots,\ f_m(\vec{S})]. \tag{6}$$

Since finding a solution which minimizes both energy consumption and the probability of system reliability is not an easy work, the definition of dominance is introduced in the following.

Definition 3. A solution is considered to dominate another solution if it is as good as the other and better in at least one objective. In other words, for two solutions $\vec{S}^*$ and $\vec{S}$ in the solution space, $\vec{S}^*$ dominates $\vec{S}$ if and only if Eq. (7) holds:

$$\forall i \in [1,\ 2,\ 3,\ \ldots,\ m],\ f_i(\vec{S}^*) \le f_i(\vec{S}) \wedge \exists j \in [1,\ 2,\ 3,\ \ldots,\ m],\ f_j(\vec{S}^*) < f_j(\vec{S}). \tag{7}$$

As shown in Fig. 2, solution A dominates solution C as the energy consumption and probability of system reliability of A are better (smaller) than those of C. Similarly, compared to solution C, solution B has smaller energy consumption and probability of system failure, so solution B dominates solution C. However, for solutions A and B, solution A is better in energy consumption and B is better in probability of system reliability. Hence, solutions A and B are non-dominated.

For a particular problem in this study, n denotes the number of the task set T ($|T| = n$), and the ith component of the solution $\vec{S}$ represents the resource on which task τ_i is released for execution. In the bi-objective optimization problem scenario, m=2 is obtained, where $f_1(\vec{S})$ is the energy consumption and $f_2(\vec{S})$ is the POF.

As reported in our previous study [34], energy consumption and system reliability are two conflicting objectives. It is impossible to achieve the minimum for both POF and energy consumption simultaneously. In such problems, no single optimal solution exists but rather a set of potential solutions. A solution is considered to dominate another solution if it is as good as the other and better in at least one objective. Conversely, two solutions are considered to be non-dominated whenever neither of them dominates the other (one is better in energy saving and the other is better in system reliability).

Another important concept is the Pareto set which is a fine solution set. The Pareto set consists of a set of non-dominated solutions. It captures a set of tradeoff solutions among different objectives. Each solution in the Pareto set denotes a distinct task mapping of different processors with diverse energy consumption and POF. The Pareto front can be used as a tool to help the user make a decision on choosing the kind of mapping strategy for the workflow tasks resources.

Table 2
Computation costs on different processors.

Node number	P_1	P_2	P_3
0	11	16	10
1	12	17	11
2	10	12	15
3	12	6	11
4	14	13	17
5	13	8	12
6	9	12	11
7	14	10	17
8	15	17	10
9	10	18	16

3.2. A motivational example

When a DAG workflow model is submitted to a data center, the only constraint condition is the tasks completion time (makespan). As an illustration, the workflow is shown in Fig. 1, and the computation cost of each task node on different processors is captured in Table 2. The constraint condition for this application is that the makespan does not exceed 135. While applying the BOGA, the data center provides the user with alternatives (shown by pairs in terms of energy consumption ratio and POF): (1.902854, 1.14e-06), (1.962252, 9.97e-07), (2.189743, 1.20e-06), (2.362726, 1.20e-06), (2.390827, 1.17e-06), (2.487445, 1.02e-06), (2.769279, 1.18e-06), (2.871348, 1.18e-06), and (3.051540, 1.24e-06). These solutions spread in the Pareto front.

4. Algorithms

Several parts including encoding, initial population, fitness measure, selection, one-point crossover, mutation, and the main algorithm, respectively, are presented in this section. The novel algorithm, named BOGA, is devised in this section to address the workflow application scheduling problem in a heterogeneous cluster.

4.1. Encoding

In this approach, each chromosome comprises two components, which include the mapping and the scheduling string. The scheduling represents the DAG topological sort, which guarantees the precedence constrains. The chromosome length is equal to the number of the task set T ($|T| = n$). Let ch (a vector of length $|T|$) be the scheduling string. T_i appears only once in ch.

4.2. Initial population

In the initial stage of BOGA, the quality of initial population is crucial. The strategy used in this stage is efficient and has a relatively low time complexity. Several outstanding heuristic list scheduling algorithms, i.e., HEFT [26], heterogeneous critical parents with fast duplicator (HCPFD) [15], predict earliest finish time (PEFT) [2] and so on, are proven to perform well in static task scheduling research. In the priority-establishing stage, these excellent strategies are selected randomly to generate distinct chromosomes.

4.3. Fitness measure

A fitness function is used to measure the quality of the solutions according to the given optimization objectives. Two objectives which include system reliability and energy consumption are primary metrics in this study.

The energy consumption of a workflow application is measured by energy consumption ratio (ECR) expressed as follows:

$$ECR = \frac{E_{total}}{\sum_{\tau_i \in CP} \min_{pe_j \in PE} \left\{ P_{ind_i} \cdot \frac{c_i}{f_i} + C_{eff} \cdot c_i \cdot f_i^2 \right\}}, \tag{8}$$

where E_{total} is the total energy consumption of the scheduled tasks, and CP is the set of tasks in the DAG critical path.

A high system reliability problem can be transformed to a low POF. Hence, POF is used to measure the second objective of this study. Apparently, POF is defined as:

$$POF = 1 - R = 1 - \Pi_{i=1}^{n} R_i(f_i). \tag{9}$$

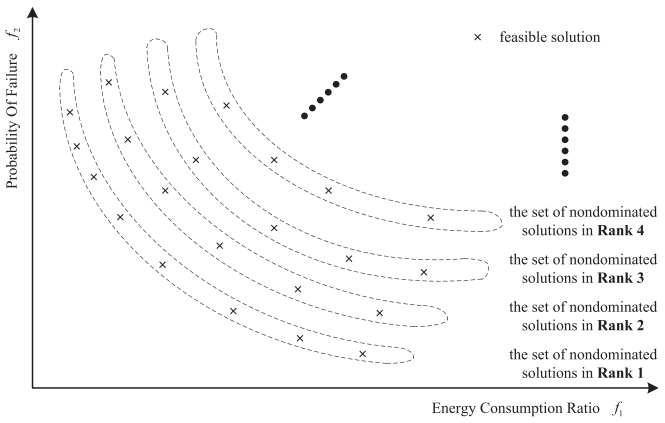

Fig. 3. The selection procedure.

4.4. Selection

Applying the non-dominated sorting reported in [7], each solution is set with a *rank* of non-dominated solutions. According to this *rank*, all of the feasible solutions are classified. The non-dominated level of each candidate solutions is established. The solutions with equal *rank* values are placed in the same level. The smaller the *rank* value is, the higher is the solution priority which maintains elitism to the next iteration. For the particular objectives in this study, low POF and low energy consumption are the two main metrics. After the procedure of non-dominated sorting, *rank*1 is found to be at the inner position. From the inner to the outside positions, *rank*2, *rank*3, and so on followed. The solutions in the inner levels are prior to the one in the outside levels when the solutions are selected as an elite population for the next iteration.

To assess chromosome quality, i.e., the performance of both POF and energy consumption in the solution is very important. During algorithm iterations, two populations ($2n$) are obtained after the mutation operation. The selection operation chooses the best n solutions from the candidate population (the size is $2n$). The rule to evaluate the solutions proceeds with the rank of solutions. The detailed rules of priorities selection are expressed as follows:

1. For solutions in different levels, the non-dominated solutions with lower level are preferred.
2. For the solutions in the same level, the solutions with large crowding distance are a priority than the small ones.

The selection procedure is shown in Fig. 3.

4.5. One-point crossover

The traditional crossover operation in genetic algorithm contains many kinds of strategies, such as one-point crossover, multi-point crossover, uniform crossover, cycle crossover, and so on. However, these common strategies are not directly applied to DAG chromosomes. The tasks sequence on a chromosome must satisfy the precedence constraints of the task set. After the completion of the cross operation, the tasks of the chromosome representation must maintain the topology of the task graph to ensure the correctness of the crossover strategy. For DAG, the following is a specific single-point cross way.

Let *parent*1 and *parent*2 be the two random chromosomes in the population, they are also the two feasible solutions to the DAG workflow task. Accordingly, the order of genes in each of the two chromosomes satisfies the precedence constraint of the task graph. *parent*1 and *parent*2 are the parent generation; after one-point crossover, two new chromosomes named *child*1 and *child*2 are generated. As shown in Fig. 4, let T ($|T| = n$) be the length of the chromosome. The one-point crossover operation is expressed as follows:

1. Two positive integers are randomly picked up as the crossover points when the condition $0 \le i \le n-1$ is met.
2. The genes in *parent*1 which locate before the crossover point i to *child*1 are located in the same location. When conditions $k < i$ and $0 \le k \le n-1$ are satisfied, $child1_k = parent1_k$ is the result.
3. The genes in parent2 which do not appear in child1 currently appeared and temporarily stored in the middle chromosome $r2$. The length of $r2$ is $n - i$.
4. The $r2$ from Step 3 is combined to the rear of *child*1.
5. The one-point crossover operations are repeated for *parent*2 as in the Steps mentioned above.

The crossover algorithm is shown in Algorithm 1. In Algorithm 1, Step 1 initializes the new population which contains two individuals. Steps 2 to 9 form the loop guaranteeing that the produced new individuals are distinct. Step 5 invokes Algorithm 2, which presents a normal crossover procedure. Algorithm 3 ensures that the crossover operation gets a feasible individual without violating the precedence constraint among the DAG graph. In algorithm 3, Step 1 randomly selects two

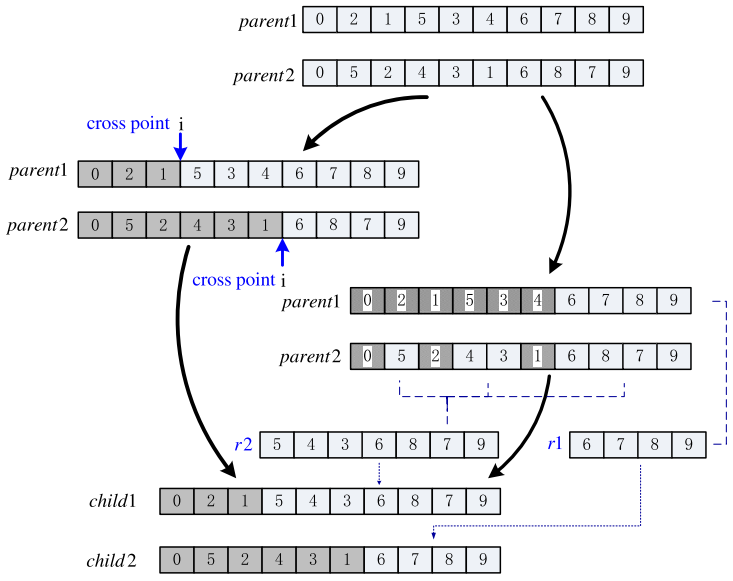

Fig. 4. one-point crossover.

Algorithm 1 The crossover algorithm

Require: two parent individuals *parent*1 and *parent*2.
Ensure: two child individuals *child*1 and *child*2.

```
1: children ← ∅
2: while i ∈ (1... POPULATION_SIZE) do
3:   if random([0, 1])≤ CROSSOVER_RATE then
4:     (parent1, parent2) ← select(parents)
5:     (child1, child2) ← crossover1(parent1, parent2)
6:     add(children, child1)
7:     add(children, child2)
8:   end if
9: end while
10: return children
```

Algorithm 2 The crossover1 algorithm

Require: two parent individuals *parent*1 and *parent*2.
Ensure: two child individuals *child*1 and *child*2.

```
1: child1 ← crossover2(parent1, parent2)
2: child2 ← crossover2(parent2, parent1)
3: return (child1, child2)
```

crossover points. Steps 2 to 4 are initial phases. Steps 6 to 9 compose the loop which duplicates the range before crossover point. Steps 10 to 14 compose the loop which shift the genes left after Steps 6 to 9 while satisfying the constraints among the parallel tasks. The time complexity of crossover algorithm is $O(n{\cdot}\log N)$, where N is the population size and n is the workflow size.

4.6. Mutation

In general, mutation operation includes simple, uniform, boundary, Gaussian, and non-uniform mutation. To escape from local optima, mutation operations are used in order to explore better solutions to some extent. As mentioned above, *URank* can maintain the topological order of the workflow tasks, i.e., the precedence constraints among tasks. As illustrated in Fig. 5, the mutation strategy used in this study is described as follows: two positive integers i and j are randomly selected while satisfying the condition $0 \le i < j \le n-1$. When the *URank* of the selected tasks τ_i and τ_j (corresponding to genes i and j in the candidate chromosome, respectively) meet the inequality $URank(\tau_i) < URank(\tau_j)$, the genes i and j are swapped and a new chromosome is generated.

Algorithm 3 The crossover2 algorithm

Require: two parent individuals *parent*1 and *parent*2.
Ensure: a child individual.
1: $(i, j) \leftarrow$ random$(1 \le i < j \le N)$
2: **for** all $k \in (1, \ldots, i-1, j+1, \ldots, N)$ **do**
3: task$(child_k) \leftarrow$ task$(parent1_k)$)
4: **end for**
5: $m \leftarrow 0$
6: **for** all $[k \in (1, \ldots, N)] \wedge [\text{task}(parent2_k) \notin \text{tasks}(child)]$ **do**
7: task$(solution_buffer_m) \leftarrow$ task$(parent2_k)$
8: $m \leftarrow m+1$
9: **end for**
10: **for** all $k \in (i, \ldots, j)$ **do**
11: **while** all solution $\in$ population **do**
12: task$(child_k) \leftarrow$ task$(solution_buffer_{k-i+1})$
13: **end while**
14: **end for**

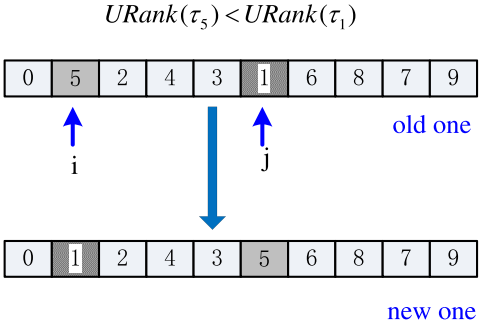

Fig. 5. Mutation.

Algorithm 4 The mutation algorithm

Require: all children individuals.
Ensure: new children individuals.
mutation(children)
2: **for** all solution $\in$ children **do**
if *random*$([0, 1]) \le$ MUTATION_RATE **then**
4: mutation(solution)
end if
6: **end for**
mutation(solution)
8: $(i, j) \leftarrow$random$(1 \le i < j \le N \wedge$ check_level$(ch_i, ch_j))$
swap(task(ch_i), task(ch_j))
10: check_level(ch_i, ch_j)
return $URank(ch_i) < URank(ch_j)$

As shown in Algorithm 4, the mutation helps the candidate solution to escape from a local optimum. In Algorithm 4, Steps 2 to 6 make each new candidate solution mutate once. Steps 3 to 5 control the degree of mutation which depends on the MUTATION_RATE. Step 8 randomly selects two mutation points. Step 10 guarantees that the entire mutation conforms to the precedence constraints among parallel tasks. The time complexity of mutation algorithm is $O(N)$, where N is the population size.

4.7. The main algorithm

The BOGA pseudocode is shown in Algorithm 5. In the population initialization, the priority queue is established with several famous strategies such as HEFT [26], HCPFD [15], PEFT [2], and that is used to create a feasible precedence queue. This operation also improves the convergence speed of the algorithm. In BOGA, Step 2 calculates the entire energy consumption and POF for each individual when it is evaluated. Steps 5–11 are repeated until the iteration circle reaches the specified

Algorithm 5 BOGA

Require: A DAG $G =< V, E >$ and a set PE of DVS available processors.
Ensure: A pareto front which contains a set of schedule for G onto PE.
1: initialize(population, POPULATION_SIZE)
2: evaluate(population)
3: GENERATION ← 1
4: **while** GENERATION ≤ GENERATION_MAXIMUM **do**
5: parents ← select(population)
6: children ← one_point_crossover(parents)
7: mutation(children)
8: evaluate(children)
9: replace(population, children)
10: update(archive, children)
11: GENERATION ← GENERATION+1
12: **end while**

maximum. Two parent individuals are randomly selected and the selected individuals are guaranteed to be different in Step 5. After Step 6, two new individuals are generated. In order to prevent the early convergence and escape from the local optima, mutation operation is devised for parallel tasks in Step 7 which is employed to expand the solution space of this study. For the new generated individuals, only the one which dominates its parent is selected for the next population. When BOGA proceeds to Step 10, the size of the current population is twice ($2N$) that of the initial population. Inspired by the fast and elitist multi-objective genetic algorithm reported in [7], the $2N$ individuals are processed with fast non-dominated sorting and crowding distance calculation. Then, each individual is assigned with *rank* values and crowding distance, respectively. During the process of selection, the individual with a low *rank* has a higher priority to be an elite. Next is the one with a large crowding distance value in the same *rank* level. This process repeats until the size of the elite individuals reaches N. The whole algorithm runs until the iterations reach the upper bound. The latest individuals N elites are the Pareto set. For a workflow with n nodes, e is the number of directed edges allowing the p processors to schedule the n tasks. The complexity of the BOGA algorithm is $O(n \log n + (e+n)p)$.

5. Performances evaluation

In this section, experiments are conducted to verify the effectiveness of this study's proposed BOGA. In the following descriptions, two performance metrics are firstly introduced. Then, the configuration of the experiment parameters is shown. Real-world and randomly generated application graphs are used to assess the performance of the proposed algorithms. Lastly, the experiment results are analyzed.

5.1. Performance metrics

(1) Hypervolume (HV, I_H)

This indicator calculates the volume, in the objective space, covered by members of a non-dominated set of solutions Q. Let v_i be a hypercube. v_i is constructed with a reference point X and the solution i ($i \in Q$) is the diagonal corners of the hypercube [11]. The hypervolume is calculated:

$$I_H = \text{volume}\left(\bigcup_{i=1}^{|Q|} v_i\right). \tag{10}$$

The hypervolume different indicator I_H^- [14] is used to measure the differences among non-dominated fronts generated by the compared algorithms. The objective portion is assessed by I_H^-. The lower the value of I_H^- is, the better the algorithm performs.

(2) Epsilon ($I_{\varepsilon+}$)

Given a computed front for a problem, A, this indicator gives a measure of the smallest distance one needs to translate every solution in A so that it dominates the optimal Pareto front of this problem. Let pf^* be the optimal Pareto front of A. Given solutions $\vec{s^1} = (s_1^1, \ldots, s_n^1)$ and $\vec{s^2} = (s_1^2, \ldots, s_n^2)$, where n is the number of objectives:

$$I_{\varepsilon+}(A) = \inf_{\varepsilon \in \mathbb{R}} \left\{ \forall\, \vec{s^2} \in pf^* \quad \exists\, \vec{s^1} \in A : \vec{s^1} \prec_\varepsilon \vec{s^2} \right\}, \tag{11}$$

Table 3
Selected workflow models.

Workflow	# Nodes	Reference	Note
T1	15	[27]	Gauss-Jordan Algorithm
T2	14	[27]	LU decomposition
T3	16	[29]	Laplace Transform

where, $\vec{s^1} \prec_\varepsilon \vec{s^2}$ if and only if $\forall 1 \leqslant i \leqslant n : s_i^1 < \varepsilon + s_i^2$. This indicator gives the idea on how separate distributions differ from each other according to their degree of $\varepsilon +-$ non-domination (for detailed descriptions, please refer to [14]). The lower value indicates a better Pareto front.

5.2. Experimental setting

In this section, several benchmarks used in workflow model are exploited to test BOGA. These workflows were designed to resolve several particular parallel numeric computation problems. These benchmarks include parallel Gauss-Jordan algorithm to solve systems of equations [27], parallel LU decompositions [27], and discrete Laplace transformation [29]. Detailed parameters are shown in Table 3.

Before making a comparison with multi-objective differential evolution (MODE) and MOHEFT algorithms, a brief introduction is firstly given. The parent and several other individuals randomly selected are combined to generate new candidate solutions. The parent is replaced by a candidate when it has a worse fitness value. It is a great selection strategy that often surpasses other classic evolution algorithms. In addition, the concept of dominance is used to preserve a uniformly spread front of non-dominated solutions. Then, the population size grows. The MODE truncates to prepare it for the next round of iterative evolution operations. While the MOHEFT algorithm is based on HEFT, a solution is created by iteratively mapping tasks onto appropriate resources. Its main objective is to minimize the entire completion time of each task. When multiple objectives are simultaneously considered, the goal changes to calculate the set of tradeoff solutions. In MOHEFT, several solutions are allowed to generate simultaneously instead of creating a single solution. In addition, the tasks are allowed to map onto other resource which has a late finish time when the tradeoff of this resource is between the considered objectives.

The experiments of the present study are carried out using a workstation at the National Supercomputing Center in Changsha. It is equipped with an Intel Core i3-540 dual-core CPU, 8 GB DRAM, and 500 GB hard disk. The machine runs with Windows 7 (64-bit OS) SP1. The proposed algorithm is implemented in an open-source framework jMetal [11], which is an object-oriented Java-based framework aimed at the development, experimentation, and study of metaheuristics for solving multi-objective optimization problems. The earliest version of jMetal is used to implement the multi-objective optimization problem of continuous functions. In the past 2 years, several researchers developed it to solve discrete optimization problem. The workflow scheduling problem developed in this study refers to a discrete and combinatorial multi-objective optimization problem in which each task has precedence constraint. In order to further exhibit BOGA performance, it was compared to the performance of MODE [25]. MODE introduces a different evolution idea to solve workflow scheduling in the global grid. Each algorithm iterates 100 times. The benchmark used in this study involves three different types of workflows. Under these workflows in the benchmark, the minimum extent in each objective is used to evaluate the quality of the proposed algorithms.

5.3. Real world application graphs

Task graphs from four kinds of real-world applications are used in these experiments: Gauss-Jordan [27], LU decomposition [27], Laplace transformation [29], and molecular dynamic code [18].

5.3.1. Three kinds of classic DAG graphs

According to the experimental parameters in Table 3, three heterogeneous processors configurations are used to schedule the three different types of workflow, i.e., T1, T2, and T3, respectively. The performance metric used for comparison are ECR and POF. The communication to computation ratio (CCR) of a DAG is used to depict the feature of the graph. It is either a communication-intensive or a computation-intensive application. When the value of CCR is higher, DAG is a communication-intensive task graph. On the contrary, the lower the CCR value is, the more computation-intensive the DAG is. The comparisons of the three workflows under algorithms BOGA, MODE, and MOHEFT are illustrated as Figs. 6, 7 and 8, respectively, when CCR is set to 1.

Fig. 6 shows the Gauss-Jordan workflow under the three kinds of algorithms. MODE, based on the differential evolution idea, finds a better Pareto front compared with MOHEFT. As the value of ECR is above 2.62, the POF of MOHEFT significantly increases. For BOGA, a fine initial population is obtained when the famous list schedule algorithms such as HEFT, HCPDF, and PEFT are employed. During each BOGA iteration, the solutions in the population proceeds with non-dominated sorting and crowding distance operations to get nondominated fronts. Then, specific one-point crossover and mutation operators for parallel tasks are employed so that the generated individuals are invalid and they reserve the most elitist individuals from

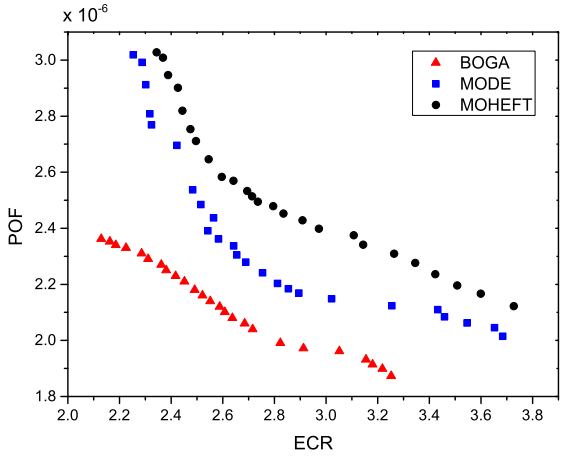

Fig. 6. Comparisons of Gauss-Jordan (the number of processors equals 3 with CCR=1.0).

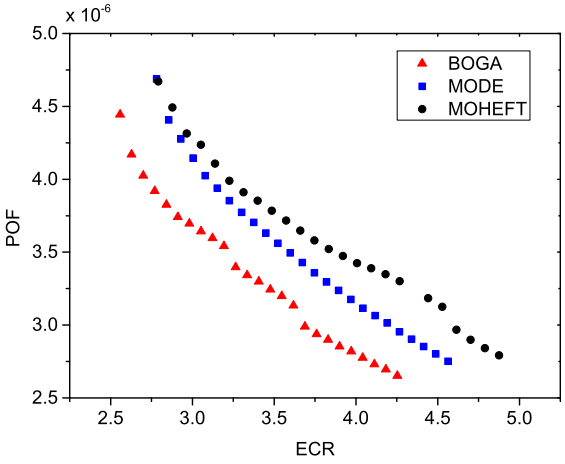

Fig. 7. Comparisons of Laplace (the number of processors equals 3 with CCR=1.0).

its parent generations. Therefore, BOGA obtains the best Pareto front in comparison with MODE and MOHEFT. In addition, the classical algorithms used in the beginning of BOGA greatly improve the efficiency of initial population constructing, which has no impact on the time complexity of BOGA.

Likewise, as shown in Figs. 7 and 8, BOGA consistently performs well. I_H^- and $I_{\varepsilon+}$ of BOGA are the minimum of the three algorithms. The difference is that T1, T2, and T3 are three different kinds of workflow in structure. The average in-degree (or the average out-degree) of T1, T2, and T3 are not the same because the task set have different parallel degrees. When the number of workflow is almost the same, the ECR and POF of the obtained Pareto front under the three existing algorithms have several differences.

5.3.2. Molecular dynamic code

DAG, extracted from the molecular dynamic code presented in [18], is used to evaluate the performance of the scheduling algorithms in this experiment. DAG is shown in Fig. 9.

The maximum out-degree in this specific DAG is 7 as shown in Fig. 9. The parameter of the maximum out-degree in a DAG has a great impact to the performance of algorithms. The parallelism degree largely depends on these parameters to some extent. When the value of the maximum out-degree is large enough, further increasing the number of processors has less effect in improving performance.

Fig. 10 shows the performance of the three algorithms when the molecular dynamic code DAG graph is scheduled on six different processor with CCR = 1.0. These results suggest that MODE and MOHEFT effectively performs to explore feasible solutions. The BOGA presented in this study delivers more consistent performance in finding better Pareto front solutions. In Fig. 10, the ECR increases as the POF decreases. This result once again verifies that BOGA is able to strike a fine balance between energy consumption and system reliability.

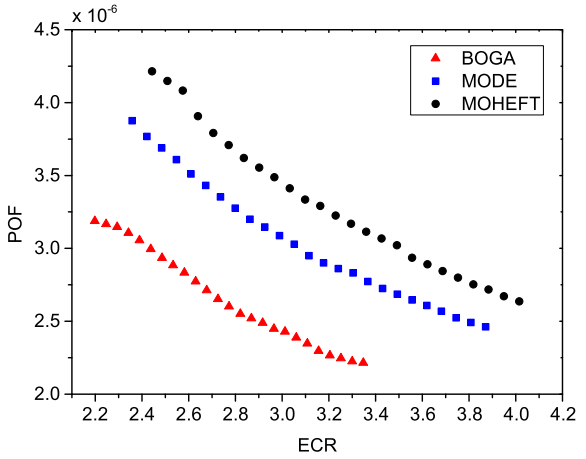

Fig. 8. Comparisons of LU (the number of processors equals 3 with CCR=1.0).

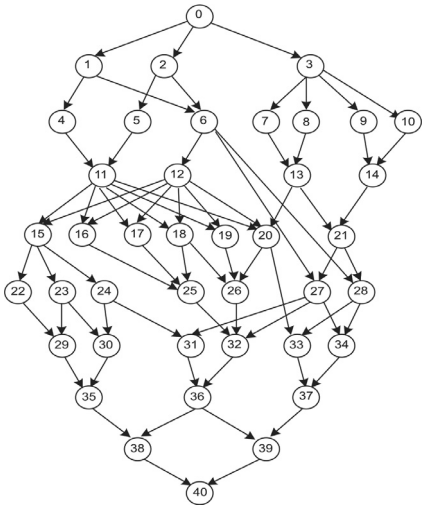

Fig. 9. A molecular graph.

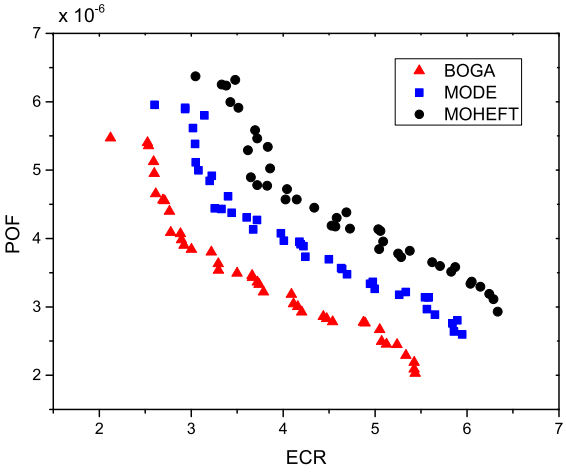

Fig. 10. Comparisons of molecular graph (the number of processors equals 6 with CCR=1.0).

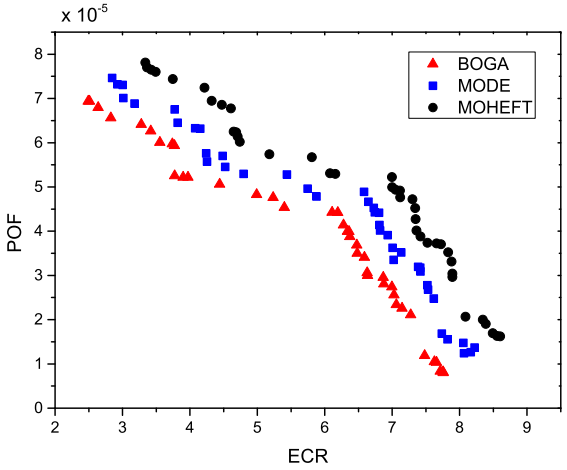

Fig. 11. Comparisons of randomly generated DAG graph (the number of task graphs equals to 100, the number of processors equals 6 with CCR=0.5).

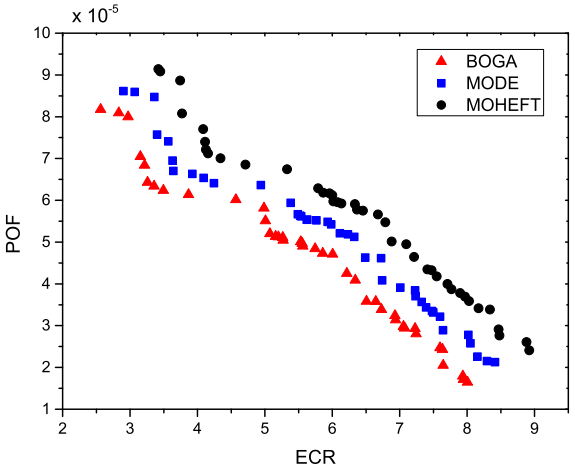

Fig. 12. Comparisons of randomly generated DAG graph (the number of task graphs equals to 100, the number of processors equals 6 with CCR=1.0).

5.4. Randomly generated application graphs

Without loss of generality, randomly generated task graphs are used to assess the performance in these experiments. Xu et al. [31] developed a random graph generator, which specifies different graphs of DAG input parameters (i.e., input degree, out-degree, the number of DAG, the level of DAG, the value of CCR, and so on) by the users. With this task generator, a large set of random task graphs are generated with different characteristics. All of these graphs are scheduled on heterogeneous computing systems. To evaluate the performance of BOGA, MODE, and MOHEFT under different parameters involving various numbers of computing processors, DAG task different numbers and different CCR values are compared. In this study, the input parameters used in the experiments is as follows: the number of task is 100, the levels of tasks ranges from 7 to 57, the maximum DAG in- and out-degrees are 47 and 57, respectively, and the average in-degree is 7.81. With respect to the number of CCRs, three kinds of values are specified–0.5, 1.0, and 5.0, respectively.

Figs. 11 to 13 show the comparisons of the three algorithms which run on three heterogeneous processors under different CCRs. As shown in Fig. 11, the application is computation intensive with CCR = 0.5 wherein the solutions explored by BOGA are significantly better than MODE and MOHEFT. When the user needs a lower POF with a modest energy consumption, the obtained Pareto front produced by BOGA offers a large number of candidate solutions. For Fig. 12, the application is modest in computation and communication. The POFs and ECRs explored by BOGA, MODE, and MOHEFT maintain the same trend as CCR = 0.5. BOGA outperforms the other two algorithms in finding a better set of feasible solutions. As illustrated in Fig. 13, when CCR is equal to 5, DAG is a communication-intensive graph and BOGA performs consistently better over MODE and MOHEFT. The fine performance of BOGA is attributed to the fact that the mechanisms of the initial population and selection in BOGA are more intelligent.

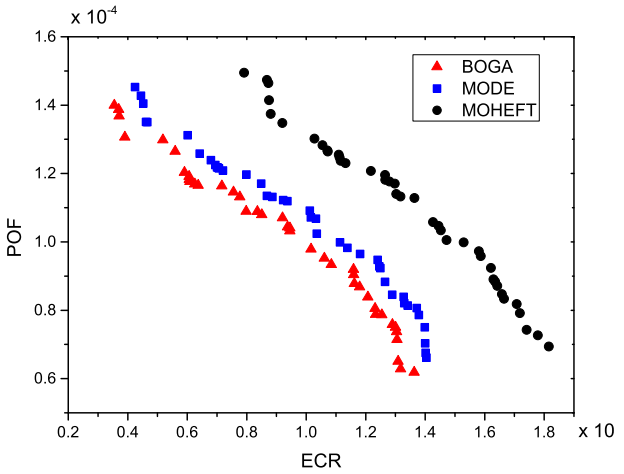

Fig. 13. Comparisons of randomly generated DAG graph (the number of task graphs equals to 100, the number of processors equals 6 with CCR=5.0).

6. Conclusion

The data center provides users with different quality of service such as maximum completion time, energy-saving effect, and system reliability to satisfy their different requirements. This paper addresses the dual objectives of energy saving and system reliability for the workflow with precedence constraints in heterogeneous systems. BOGA is developed to obtain a fine Pareto front. In the beginning of BOGA, several well-known, efficient strategies are adopted to the initial population without introducing additional computation complexity. For the selection operation, the two objectives of low energy consumption and low POF are used to determine the non-dominated solutions. The specific one-point crossover is designed to generate new individuals without violating the precedence constraint. The selected solutions that fulfill the *URank* constraint perform a particular mutation operation, thereby helping the BOGA to escape from the local optima and explore a new solution space. While entering the next iteration, two operations, namely, non-dominated sorting and crowding distance, are performed, thereby reserving the diversity of individuals. Unlike in modified MODE and MOHEFT, three kinds of workflows, i.e., Gauss-Jordan, parallel LU decomposition and discrete Laplace transform, and real-world application graphs (i.e., molecular dynamics code graph and randomly generated application graphs), are exploited to assess performance. Experiments show that the proposed BOGA algorithm significantly outperforms other algorithms in terms of system reliability and energy consumption.

One planned future research is to develop multiple objectives such as makespan, energy saving, and system reliability problem for workflow in heterogeneous computing systems.

Acknowledgment

The research was partially funded by the Key Program of National Natural Science Foundation of China (Grant Nos. 61133005, 61432005), the National Natural Science Foundation of China (Grant Nos. 61370095, 61472124), the International Science & Technology Cooperation Program of China (Grant No. 2015DFA11240), the National High-tech R&D Program of China (Grant No. 2015AA015303), the Research Foundation of Education Bureau of Hunan Province (No. 15C0400), the Natural Science Foundation of Hunan Province (Grant No. 2016JJ4002), the Project of the National Housing and Construction (No. 2014-K8-062), and the Key Foundation of Education Bureau of Hunan Province (No. 14A037).

References

[1] 2015, (http://www.top500.org/).
[2] H. Arabnejad, J. Barbosa, List scheduling algorithm for heterogeneous systems by an optimistic cost table, IEEE Trans. Parallel Distrib. Syst. 25 (3) (2014) 682–694, doi:10.1109/TPDS.2013.57.
[3] M. Arsuaga-Ríos, M.A. Vega-Rodríguez, F. Prieto-Castrillo, Meta-schedulers for grid computing based on multi-objective swarm algorithms, Appl. Soft Comput. 13 (4) (2013) 1567–1582.
[4] M. Bhuiyan, G. Wang, J. Cao, J. Wu, Deploying wireless sensor networks with fault-tolerance for structural health monitoring, IEEE Trans. Comput. 64 (2) (2015) 382–395, doi:10.1109/TC.2013.195.
[5] C. Boeres, I.M. Sardiña, L.M. Drummond, An efficient weighted bi-objective scheduling algorithm for heterogeneous systems, Parallel Comput. 37 (8) (2011) 349–364.
[6] T.D. Burd, R.W. Brodersen, Energy efficient cmos microprocessor design, in: System Sciences, 1995. Proceedings of the Twenty-Eighth Hawaii International Conference on, 1, IEEE, 1995, pp. 288–297.
[7] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: Nsga-ii, IEEE Trans. Evol. Comput. 6 (2) (2002) 182–197.
[8] A. Doğan, F. Özgüner, Biobjective scheduling algorithms for execution time–reliability trade-off in heterogeneous computing systems, Comput. J. 48 (3) (2005) 300–314.

[9] J.J. Dongarra, E. Jeannot, E. Saule, Z. Shi, Bi-objective scheduling algorithms for optimizing makespan and reliability on heterogeneous systems, in: Proceedings of the nineteenth annual ACM symposium on Parallel algorithms and architectures, ACM, 2007, pp. 280–288.
[10] J.J. Durillo, V. Nae, R. Prodan, Multi-objective energy-efficient workflow scheduling using list-based heuristics, Future Gener. Comput. Syst. 36 (2014) 221–236.
[11] J.J. Durillo, A.J. Nebro, jmetal: A java framework for multi-objective optimization, Adv. Eng. Software 42 (10) (2011) 760–771.
[12] J.J. Durillo, R. Prodan, Multi-objective workflow scheduling in amazon ec2, Cluster Comput. 17 (2) (2014) 169–189.
[13] H.M. Fard, R. Prodan, J.J.D. Barrionuevo, T. Fahringer, A multi-objective approach for workflow scheduling in heterogeneous environments, in: Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012), IEEE Computer Society, 2012, pp. 300–309.
[14] C.M. Fonseca, J.D. Knowles, L. Thiele, E. Zitzler, A tutorial on the performance assessment of stochastic multiobjective optimizers, in: Third International Conference on Evolutionary Multi-Criterion Optimization (EMO 2005), 216, 2005, p. 240.
[15] T. Hagras, J. Janeček, A high performance, low complexity algorithm for compile-time task scheduling in heterogeneous systems, Parallel Comput. 31 (7) (2005) 653–670.
[16] X. Huang, L. Zhang, R. Li, L. Wan, K. Li, Novel heuristic speculative execution strategies in heterogeneous distributed environments, Comput. Electr. Eng. (2015), doi:10.1016/j.compeleceng.2015.06.013.
[17] E. Jeannot, E. Saule, D. Trystram, Optimizing performance and reliability on heterogeneous parallel systems: Approximation algorithms and heuristics, J. Parallel Distrib. comput. 72 (2) (2012) 268–280.
[18] S. Kim, J. Browne, A general approach to mapping of parallel computation upon multiprocessor architectures, in: International conference on parallel processing, 3, 1988, pp. 1–8.
[19] Y.C. Lee, A.Y. Zomaya, Energy conscious scheduling for distributed computing systems under different operating conditions, IEEE Trans. Parallel Distrib. Syst. 22 (8) (2011) 1374–1381.
[20] K. Li, Scheduling precedence constrained tasks with reduced processor energy on multiprocessor computers, IEEE Trans. Comput. (2012) 1668–1681.
[21] S. Penmatsa, A.T. Chronopoulos, Cooperative load balancing for a network of heterogeneous computers, in: the 20th IEEE International Parallel and Distributed Processing Symposium, IEEE, 2006, pp. 8–15.
[22] S. Penmatsa, A.T. Chronopoulos, Cost minimization in utility computing systems, Concurrency Comput. 26 (1) (2014) 287–307.
[23] S. Penmatsa, A.T. Chronopoulos, N.T. Karonis, B.R. Toonen, Implementation of distributed loop scheduling schemes on the teragrid, in: the 21th IEEE International Parallel and Distributed Processing Symposium, IEEE, 2007, pp. 1–8.
[24] G.C. Sih, E.A. Lee, A compile-time scheduling heuristic for interconnection-constrained heterogeneous processor architectures, IEEE Trans. Parallel Distrib. Syst. 4 (2) (1993) 175–187.
[25] A. Talukder, M. Kirley, R. Buyya, Multiobjective differential evolution for scheduling workflow applications on global grids, Concurrency Comput. 21 (13) (2009) 1742–1756.
[26] H. Topcuoglu, S. Hariri, M.-y. Wu, Performance-effective and low-complexity task scheduling for heterogeneous computing, IEEE Trans. Parallel Distrib. Syst. 13 (3) (2002) 260–274.
[27] T. Tsuchiya, T. Osada, T. Kikuno, Genetics-based multiprocessor scheduling using task duplication, Microprocess. Microsyst. 22 (3) (1998) 197–207.
[28] X. Wang, C.S. Yeo, R. Buyya, J. Su, Optimizing the makespan and reliability for workflow applications with reputation and a look-ahead genetic algorithm, Future Gener. Comput. Syst. 27 (8) (2011) 1124–1134.
[29] M.-Y. Wu, D.D. GAJsKI, Hypertool: a programming aid for message-passing systems, IEEE Trans. Parallel Distrib. Syst. 1 (3) (1990) 330–343, doi:10.1109/71.80160.
[30] Y. Xu, K. Li, L. He, T.K. Truong, A dag scheduling scheme on heterogeneous computing systems using double molecular structure-based chemical reaction optimization, J. parallel distrib. comput. 73 (9) (2013) 1306–1322.
[31] Y. Xu, K. Li, L. He, L. Zhang, K. Li, A hybrid chemical reaction optimization scheme for task scheduling on heterogeneous computing systems, IEEE Trans. Parallel Distrib. Syst. 26 (12) (2015) 3208–3222, doi:10.1109/TPDS.2014.2385698.
[32] Y. Xu, K. Li, J. Hu, K. Li, A genetic algorithm for task scheduling on heterogeneous computing systems using multiple priority queues, Inf. Sci. 270 (2014) 255–287.
[33] L. Zhang, K. Li, Energy minimization for software real-time systems with uncertain execution time, in: 2011 Fourth International Symposium on Parallel Architectures, Algorithms and Programming (PAAP), 2011, pp. 87–91.
[34] L. Zhang, K. Li, Y. Xu, J. Mei, F. Zhang, K. Li, Maximizing reliability with energy conservation for parallel task scheduling in a heterogeneous cluster, Information Sciences 319 (2015) 113–131. http://dx.doi.org/10.1016/j.ins.2015.02.023. URL http://www.sciencedirect.com/science/article/pii/S0020025515001231
[35] D. Zhu, R. Melhem, D. Mossé, The effects of energy management on reliability in real-time embedded systems, in: Computer Aided Design, 2004. ICCAD-2004. IEEE/ACM International Conference on, IEEE, 2004, pp. 35–40.