Journal of Universal Computer Science, vol. 23, no. 7 (2017), 636-651
submitted: 31/3/17, accepted: 30/6/17, appeared: 28/7/17 © J.UCS

Multi-Objective Evolutionary Algorithm Based on Decomposition for Energy-aware Scheduling in Heterogeneous Computing Systems[1]

Sisi Yuan
(Oklahoma State University, Stillwater, Oklahoma, USA
sisiyuan@foxmail.com)

Gaoshan Deng
(Xiamen University, Xiamen, Fujian, China
gaoshan.deng@foxmail.com)

Quanxi Feng
(Oklahoma State University, Stillwater, Oklahoma, USA
sisiyuan@foxmail.com)

Pan Zheng
(Swinburne University of Technology Sarawak Campus, Kuching, Sarawak, Malaysia
pzheng@swinburne.edu.my)

Tao Song
(Swinburne University of Technology Sarawak Campus, Kuching, Sarawak, Malaysia
tsong@swinburne.edu.my)

Abstract: Heterogeneous computing systems (HCSs) use many heterogeneous processors or cores to perform particular tasks. To address the requirement of green IT, several power management techniques have been developed to reduce the energy consumption of these systems. Dynamic voltage scaling, which dynamically changes the supply voltage of processors during the execution of an application, is widely used. However, reducing supply voltage decreases computation speed. Therefore, system makespan and energy consumption need to be considered at the same time. We propose a multi-objective scheduling algorithm based on decomposition for scheduling of the system workflow. Through experiments, we examine the performances of several algorithms, including the proposed one, in different benchmarks and real-world applications. Results show that our algorithm demonstrates better performance than other state-of-art evolutionary algorithms under various conditions involving the use of different crossover and mutation operators.

Keywords: Energy efficiency, Heterogeneous computing systems, Evolutionary algorithms, MOEA/D
Categories: D.4.1, H.1.2, H.4.0

[1] This paper is an extended version of a paper published on IEEE conference on Bio-Inspired Computing: Theory and Applications (BIC-TA).

1 Introduction

The increasing usage of computer systems in recent years has resulted in a rapid rising in energy consumption. A report from the World Data Center [Koomey, 11] indicates that electricity use was doubled from 2000 to 2005. Another report [Bianchini and Rajamony, 04] reveals that the annual energy consumption of data centers in the United States in 2011 exceeded 100 billion kWh, which amounted to \$7.4 billion. This high energy consumption is expected to cause global environmental and economic burdens. Traditional IT emphasizes only the minimization of an application's execution time. Green IT is gradually eliciting increased social awareness. Different from traditional IT, Green IT not only considers the speed of applications but also entails low power consumption. Owing to the successful use of Green IT to control software, the electricity consumption growth decreased significantly from 2005 to 2010. Many energy-saving technologies have been proposed in the Green IT field.

Dynamic voltage frequency scaling (DVFS) uses chip multiprocessors (CMPs) to improve energy efficiency. DVFS allows for the control of supply voltages. DVFS can also dynamically change voltages during execution. Several studies [Huang et al., 12][Semeraro et al., 02] on CMPs with multi-threaded commercial and scientific workloads have shown that DVFS is highly effective in conserving energy.

Slack reclamation [Lee and Zomaya, 11] is a post-processing slack-reclamation-based scheduling algorithm that uses DVFS. The algorithm takes an already constructed schedule and uses task slack times to reduce the performance of processors. During the execution of the processors, slack reclamation leads to reduced energy consumption, and it does not cause time losses.

The proper application of these two methods is a key issue in HCSs; it is an NP-hard problem [Garey and Johnson, 79] that involves many precedence constraints, a set of heterogeneous processors, and high communication and execution costs. Traditional deterministic scheduling methods cannot achieve a satisfactory scheduling result in a short time. To establish an optimal scheduling scheme, scholars have proposed many new algorithms (discussed in the section "Related Work"). However, these algorithms do not perform as expected, and the final solutions obtained by these algorithms present low diversity and convergence, particularly when solving large-scale tasks for large distribution systems with many heterogeneous processors.

The current study investigates a multi-objective evolutionary algorithm based on decomposition (MOEA/D) [Zhang and Li, 07]. MOEA/D has numerous advantages and won in the CEC2009 competition [Zhang et al., 09]. The algorithm exhibits good performance in solving discrete problems, such as the multi-objective 0-1 Knapsack problem. The experiment results show that MOEA/D also performs well when used in scheduling problems. The main contributions of this work are as follow:

(1) We propose an algorithm based on the MOEA/D framework to solve the scheduling problem and compare it with two other state-of-art evolutionary algorithms, namely, NSGA-II and SPEA2. The experimental results indicate that the proposed algorithm presents a significant improvement in terms of the diversity and convergence of the final solutions.

(2) We apply two newest crossover operators, namely, multi-parent crossover

operator (MPCO) and grouping crossover operator (GCO), in the algorithm. Our experiments examine their performances in solving the problem in different conditions.

The rest of the paper is organized as follows. The section “Problem Description” defines the energy-aware scheduling problem in HCSs. “Related Work” describes the background of and related work on this problem. “Algorithm Description” presents the details of our MOEA/D-based algorithm for the scheduling problem. “Experimentation” shows the results of the experiment and analyses. “Conclusions” provides the findings of this study.

2 Problem Description

2.1 Definition

A distributed computing system consists of a set of heterogeneous processors that execute a parallel application. Each processor can operate with a set of DVFS pairs. The DVFS pairs reflect the corresponding relation between execution speed and supply voltage.

The scheduling problem considers optimizing a parallel application that can be executed by a distributed system. The parallel application consists of a set of tasks, and the tasks represent a non-divisible computing unit that can be described as a directed acyclic graph (DAG) as follows:

$$\mathrm{G} \equiv (\mathrm{T}, \mathrm{E}, \mathrm{P}, \mathrm{C}) \qquad (1)$$

where T represents a set of nodes in the DAG. Each node represents a task. E is the set of all edges in the graph. P_{ij} is the computation time of $\mathrm{task_i}$ on processor $\mathrm{r_j}$. C_{ij} refers to the communication cost involved in transferring data from $\mathrm{task_i}$ to $\mathrm{task_j}$ between different processors. $\varphi(\mathrm{i})$ is the set of all the precedence constrains of $\mathrm{task_i}$. Entry task is the task that has no precedence constrains. Moreover, $\psi(\mathrm{i})$ is the set of all successors of $\mathrm{task_i}$, and the task that does not have any successors is called a sink task.

Fig. 1 provides an example of the DAG. The DAG has 8 tasks and 10 edges, with the communication cost between tasks. The entry task of the DAG is $\mathrm{task_0}$, and the sink task is $\mathrm{task_7}$.

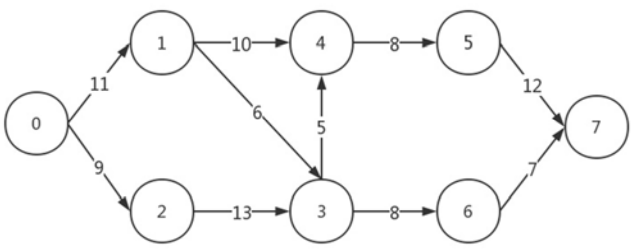


Figure 1: Example of a DAG

0 shows the expected computation costs of tasks when executed by three given heterogeneous processors (r_0, r_1 and r_2). The computation cost is spent on the high speed of the processors (speed rate of 100%). The value of the bottom level (b-level) reflects the task priorities of the application. The b-level uses a greedy strategy to determine the priorities of the tasks, which can be calculated by the longest path from the beginning of the task to the bottom of the graph.

Task	**r_0**	**r_1**	**r_2**	**b - level**
0	6	8	10	112.0
1	10	12	13	92.0
2	7	9	8	95.0
3	11	13	15	74.0
4	12	17	15	56.0
5	6	13	9	33.3
6	11	10	15	31.0
7	9	12	15	12.0

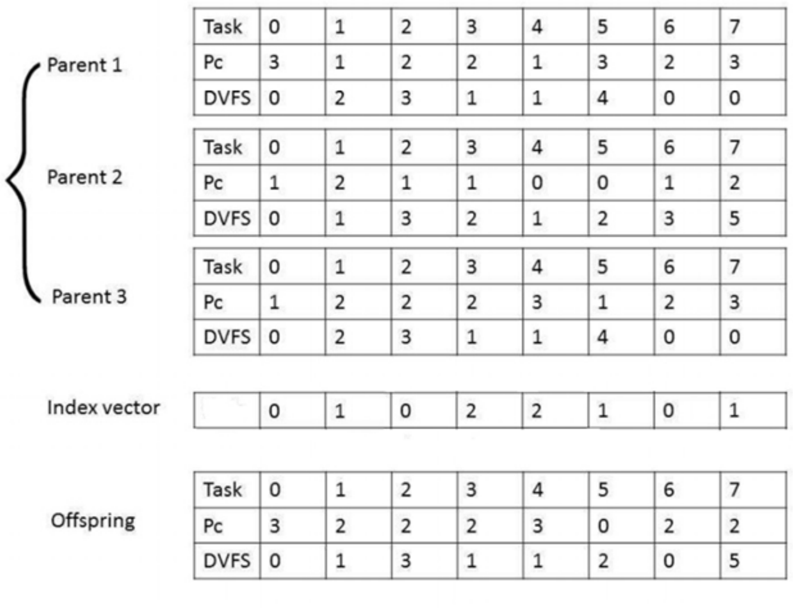
Table 1: Computation cost and task priorities

In DVFS technology, the connection between supply voltage and processor frequency can be established. A set of DVFS pairs is obtained by dividing a series of execution levels to represent the relationship between the voltage and frequency of a processor. 0 shows a set of available DVFS pairs of three heterogeneous processors (r_0, r_1 and r_2).

	r_0		**r_1**		**r_2**	
Level	**V_k**	**RS_s(%)**	**V_k**	**RS_s(%)**	**V_k**	**RS_s(%)**
0	2.20	100	1.50	100	1.75	100
1	1.90	85	1.40	90	1.40	80
2	1.60	65	1.30	80	1.20	60
3	1.30	50	1.20	70	0.9	40
4	1.00	35	1.10	60		
5			1.00	50		
6			0.90	40		

Table 2: Sample of DVFS pairs of processors

The evolutionary algorithm requires a gene scheme for problem solving, so we use an encoding gene according to the problem to represent a specific scheduling schema. The gene contains two vectors. The first vector contains the selected processor index for the task, the other vector stores the associated DVFS pairs during the task's execution (parent 1, parent 2 and parent 3 in Fig. 2).



Parent 1

Task	0	1	2	3	4	5	6	7
Pc	3	1	2	2	1	3	2	3
DVFS	0	2	3	1	1	4	0	0

Parent 2

Task	0	1	2	3	4	5	6	7
Pc	1	2	1	1	0	0	1	2
DVFS	0	1	3	2	1	2	3	5

Parent 3

Task	0	1	2	3	4	5	6	7
Pc	1	2	2	2	3	1	2	3
DVFS	0	2	3	1	1	4	0	0

Index vector

0	1	0	2	2	1	0	1

Offspring

Task	0	1	2	3	4	5	6	7
Pc	3	2	2	2	3	0	2	2
DVFS	0	1	3	1	1	2	0	5

Figure 2: Multi-Parent Crossover Operator (MPCO)

2.2 Time Model

The execution time of each task of the application needs to be calculated for the problem. To meet the precedence constraints of this DAG, the start time S_i of each task should be determined. S_i can be calculated as

$$\begin{cases} S_i = 0, & \text{if i is entry task} \\ \max\{C_{ij} + F_j \mid j \in \varphi(i)\}, & \text{else} \end{cases} \tag{2}$$

where C_{ij} is equal to the communication cost between $task_i$ and $task_j$ and F_i is the completion time of $task_i$ and

$$F_i = S_i + E_i \tag{3}$$

where E_i is the actual execution time of $task_i$. The sets EST (earliest start time) and EFT (earliest finish time) are defined respectively as

$$EST \equiv \{S_i | i \in n\} \tag{4}$$

$$EFT \equiv \{F_i | i \in n\} \tag{5}$$

where n is the number of tasks. After the scheduling is completed, makespan, which is the first objective of the problem, can be determined. The value of makespan equals the finish time of the sink task as follows:

$$maskspan = \max\{F_i | F_i \in EFT\} \tag{6}$$

2.3 Energy Model

Another objective is to determine the total energy consumption of the application. In our experiments, energy consumption is calculated with the formula of complementary metal-oxide semiconductor (CMOS) logic circuits [Lee and Zomaya, 11], which is defined as

$$P = ACV^2f \qquad (7)$$

where A is the number of switches per clock cycle's number, C is the total capacitance load, V is the supply voltage, and f indicates the clock frequency. Given that A and C are constant only for a machine, we can simplify them to single coefficient α which equals 1. Frequency f is proportional to the related speed. Hence, the total energy consumption is defined as

$$E_i = \sum_{i=0}^{n-1} ACV_i^2 f w_i^* = \sum_{i=0}^{n-1} \alpha V_i^2 w_i^* \propto \sum_{i=0}^{n-1} V_i^2 w_i^* \qquad (8)$$

where V is the supply voltage of the processor for task_i executed and w_i^* is the computation cost of task n_i (amount of time required for the execution of n_i) in the scheduled processor.

Slack reclamation is used to reduce the energy consumption. In the experiment, when the processors are free, we set them to their lowest available DVFS pairs to obtain low energy consumption.

The goal is to determine a scheduling schema that assigns an available processor to each task with a corresponding DVFS pair during execution and simultaneously minimize energy consumption and completion time.

3 Related Work

To solve diverse applications and computing platforms, a large number of DVS-based algorithms have been proposed. The majority of these are DVS-based scheduling heuristics. Several of them are implemented on server computing and single-processor systems [Bianchini and Rajamony, 04], whereas others are implemented on multiprocessor real-time systems [Zhu et al., 03].

The efficiency of distributed systems is an emerging research field. The authors in [Herbert and Marculescu, 07] completed several new observations regarding fine-grained dynamic voltage/frequency scaling in allusion to chip multiprocessors. They found that DVFS is highly effective in improving the energy efficiency of CMPs for multi-threaded commercial and scientific workloads. They successfully evaluated and achieved an $\mathrm{energy/throughput}^2$ reduction of 38.2% with the best ideal scheme.

Biological systems are rich sources of ideas for designing computing devices. In fact, they have inspired numerous classical computing devices, such as automatons and Turing machines. Computing devices inspired by cells (or molecules inside cells, such as DNA) have been thoroughly investigated. Most cell-inspired computing systems have been proven to be universal [Zeng et al., 14] and computationally efficient [Liu et al., 16].

Genetic algorithms (GAs) have been utilized to solve scheduling problems. A large number of MOEAs have been proposed to identify local regions [Xue et al., 17] or points of interest in the Pareto optimal front [Zhang et al., 15]. Kolodziej [Kołodziej et al., 11] developed two GAs with elitist and struggle replacement mechanisms as energy-aware schedulers. To obtain a set of Pareto front solutions, Tao proposed a new hybrid GA approach composed of a case library (CL) and a multi-objective genetic algorithm; the approach is called CLPS-GA [Tao et al., 14]. Guzek proved that three well-known evolutionary algorithms (NSGA-II, IBEA, and MOCell) are effective in problem solving [Guzek et al., 14].

Decomposition methods have been consistently used to handle multi-objective problems. For example, Pecero [Pecero et al., 09] presented a new GA for task scheduling through clustering based on graph decomposition. MOEA/D [Zhang and Li, 07] is a recent algorithm based on decomposition. The major advantage of MOEA/D is that performing a well-developed single-optimization local search within it is easy. Reference [Peng et al., 09] showed that MOEA/D without a local search performs better than NSGA-II in the multi-objective traveling salesman problem.

4 Algorithm Description

4.1 Algorithm Framework

MOEA/D decomposes a multi-objective optimization problem into a number of scalar optimization subproblems. The neighborhood relation between these subproblems is defined based on the distance of the vectors. MOEA/D is more effective than other algorithms in solving two- and three-objective problems. The MOEA/D framework is defined in Fig. 3.

4.2 Crossover Operation

Traditional GAs use a two-parent crossover operator for discrete problems. However, the searching range is limited by the two selected parents during each crossover operation. To search a wider range, several crossover strategies have been established to improve the crossover process. GCO, which was proposed by [Guzek et al., 14], and MPCO [Tao et al., 14] are examples of these strategies. This study considers these two types of crossover operation.

During the crossover process, MPCO randomly selects M (more than two) parents from the neighbor population of the current subproblem and creates a random index vector, for this process. The corresponding gene of a new individual's chromosome is determined by this index vector. Fig. 3 shows an example of MPCO (M = 3).

GCO, which was proposed by Falkenauer [Falkenauer, 98], considers a group of tasks as parents. It merges the entire group of the tasks and randomly selects a subset of all processors used in a solution.

4.3 Mutation

We adopt two types of mutation operators in each generation to maintain the diversity of the solution. These two types of mutation are independent and can occur with the same probability in our study. In the first type of mutation, both processor and DVFS pair assignments are changed. In the second type, we only change the DVFS pair.

MOEA/D Framework

```
     Input: $N$ for population size
01   Initialize the population P$\{x_1, x_2, \cdots, x_N\}$ and a set of weight
     vectors W$\{w_1, w_2, \cdots, w_N\}$, and the ideal point $Z^*$;
02   for each $i \in [0, \mathrm{N}]$ do
03       $\mathrm{B(i)} = i_1, i_2, \cdots, i_T$ as the T closest weight vectors to $w_i$;
04   end for
05   set EP = $\emptyset$;
06   while ! StopCondition() do
07       for each $i \in [0, \mathrm{N}]$ do
08           parents = selection(B(i));
09           offspring = crossover(parents);
10           mutation(offspring);
11           evaluate(offspring);
12           update (offspring);
13       end for
14   end while
15   return EP;
```

Figure 3: MOEA/D framework

4.4 Evaluation

In each generation for evolution, evaluating the makespan and energy consumption of the solution is required. Therefore, its performance exerts a critical influence on the overall efficiency. To improve efficiency, Ahmad [Ahmad et al., 96] proposed a greedy heuristic list, in which b-level is allocated to each task to determine task priority. After obtaining the b-level of tasks, all tasks are sorted in a non-increasing order according to their b-level values. The first one to execute is the task with the highest value of this indicator. Next, the priority based on the b-level can be applied to the rest of the tasks for the processor without violating the precedence constraint. Then, the real execution time of the solution can be obtained. The makespan and total energy consumption of the solution can be determined subsequently.

5 Experimentation

In our experiment, the behavior of MOEA/D is compared with that of two other algorithms, namely, NSGA-II [Deb et al., 02] and SPEA2 [Zitzler et al., 01]. These two algorithms are widely applied in solving real-world problems and demonstrate good performance. Both of them use classical strategies, such as a crowded-comparison operator, fitness assignment strategy, and enhanced archive truncation method, to improve diversity and convergence.

5.1 Test Instances

To represent various real-world applications, a wide range of instances are used in our work. Two metrics, namely, communication-to-computation ratio (CCR) and heterogeneity, are introduced to describe each problem.

1. CCR [Crovella et al., 92]. It is the average communication cost for all tasks, and this metric presents computation-intensive or communication-intensive DAGs.

2. Heterogeneity. This metric presents the difference between processors to test various systems ranging from quasi-homogenous (heterogeneity = 0.1) to fully heterogonous (heterogeneity = 1).

Type	**The value set**
Task count	{50, 100, 300, 500}
CCR	{0.1, 0.5, 1, 5, 10}
Processor count (PC)	{3, 5, 8, 10, 20}
Heterogeneity	{0.25, 0.5, 0.75, 1}

Table 3: Instance metrics used in our study

0 lists a set of random test instances created according to different metrics for different situations.

In our experiment, we use several real-world application problems for simulation. These problems include a robot control application, a sparse matrix solver, and the fpppp problem from the website www.kasahara.elec.waseda.ac.jp/schedule.

5.2 Performance metrics

To evaluate the performance of the final results of each algorithm, we use the inverted generational distance (IGD) metric to reflect the quality of the solution set. Another metric is the solution number to reflect the number of the final non-dominated solution.

The IGD metric [Li and Zhang, 09] shows how close the points are in the approximated set to the closest point of the Pareto front. However, to calculate IGD, a Pareto front is required. However, this is unknown to us, so it is replaced by a pseudo-optimal Pareto front in the experiment. The pseudo-optimal Pareto front consists of all the best non-dominated solutions of all the algorithms in the same test instance.

The unary additive epsilon [Zitzler et al., 01], which is used to measure convergence by calculating the smallest distance needed to make every point in the result set dominate the Pareto front points, is also used in our experiments.

6 Experiments and Result

This section shows the results of comparing MOEA/D, NSGA-II, and SPEA2 under different instance metrics.

6.1 Parameter setting

The parameter setting in our experiments is generally based on MOEA [Guzek et al., 14]. Population size is set to 100, and the crossover and mutation rates are set to $r_{crossover} = 0.9$ and $r_{mutation} = 1/t$, respectively. To make the experiment more valid, each instance is run 30 times. The average results are calculated to represent the final result of one condition. When an algorithm reaches100,000 evaluations, the instance is stopped.

6.2 Crossover test

GCO and MPCO are used in the test. The test instances are a set of random DAGs. The number of tasks is set from 50 to 500. During each test instance, the two crossover operations are adopted in the three algorithms mentioned above.

6.3 Processor number, CCR and Heterogeneity test

To determine the influence of different parameter settings, we use three real-world applications to examine performance.
Several parameters are related to each application; these include t for task number, e for edge number, p for processor number, and h for heterogeneity. By combining these different parameters, we can simulate a wide range of conditions.

The final solutions obtained by the three algorithms are shown in Table 4. MOEA/D can find the highest range of solutions among the three algorithms. In the right and left areas of the Fig. 7, MOEA/D can find extreme solutions. The two other algorithms have no solutions in these areas. The populations of the two other algorithms are limited to the middle of the figure.

However, MOEA/D still exhibits drawbacks. In the middle field of the figure, MOEA/D has less convergence than NSGA-II. The main reason is that the populations of NSGA-II tend to focus on one narrow field in the middle, and the offspring in the edges are quickly eliminated, which leads to premature convergence. No strategy can be used to deal with the premature convergence problem in NSGA-II.

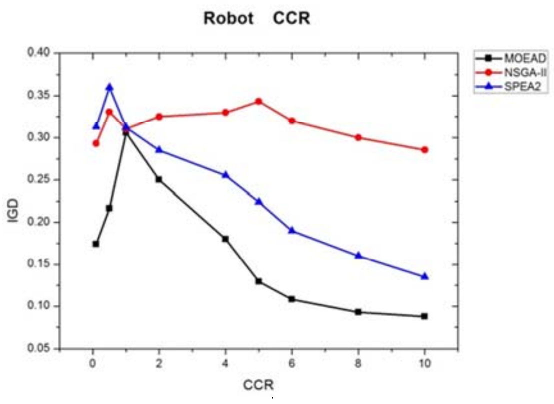
Operator \ Task Number			50	100	300	500
IGD	MOEA/D	MPCO	0.02333	0.080564	0.155258	0.171179
		GCO	0.077004	0.223734	0.462336	0.691859
	NSGA-II	MPCO	0.177005	0.179195	0.441121	0.55751
		GCO	0.247156	0.393298	0.435443	0.757908
	SPEA2	MPCO	0.07143	0.157082	0.388061	0.321148
		GCO	0.184032	0.264864	0.461158	0.497952
Epsilon	MOEA/D	MPCO	853	108	38	779
		GCO	154	228	113	396
	NSGA-II	MPCO	109	187	197	356
		GCO	648	1310	4477	7465
	SPEA2	MPCO	682	835	192	404
		GCO	534	255	223	753

Table 4: Comparison of two crossover operators (IGD and Epsilon)

6.4 Analysis of the results

In our experiment, each algorithm can obtain a set of Pareto front solutions. The result is shown in 0. Comparison of the different crossover operators indicates that the performance of MPCO is better than that of GCO for almost all the algorithms. MOEA/D is the most improved algorithm when MPCO is used instead of GCO.

In the CCR test (Fig. 4), processor count test (Fig. 5), and heterogeneity test (Fig. 6), MOEA/D outperforms the other methods in diversity metrics. Fig. 7 shows one of the average final solutions. The solution number obtained by MOEA/D exhibits a huge improvement compared with that obtained by NSGA-II and SPEA2. NSGA-II obtains a stable result regardless of the change in conditions. The IGD metric of the final solutions of NSGA-II is stable when the given processor is improved.



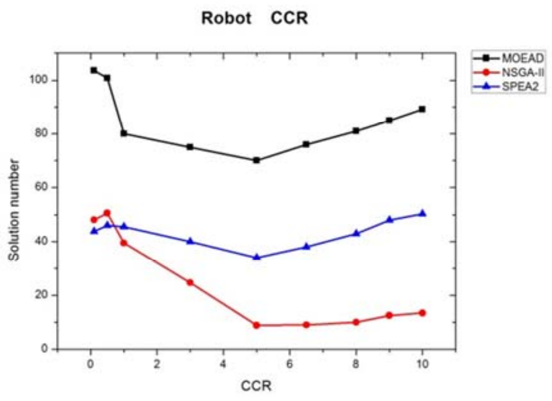
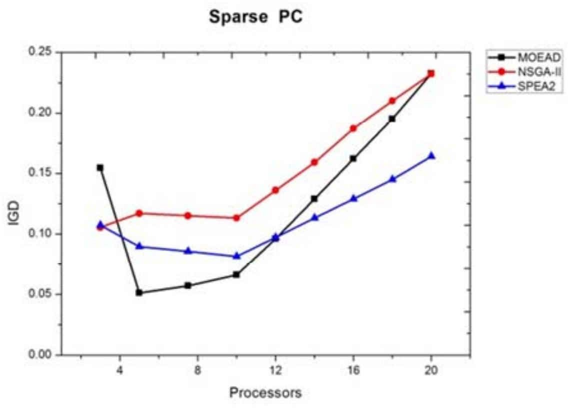


Figure 4: CCR test result



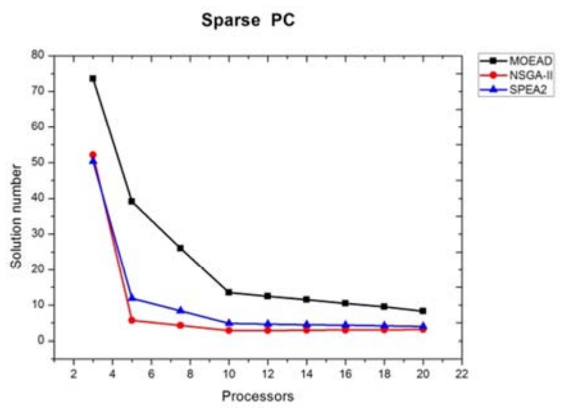


Figure 5: PC test result

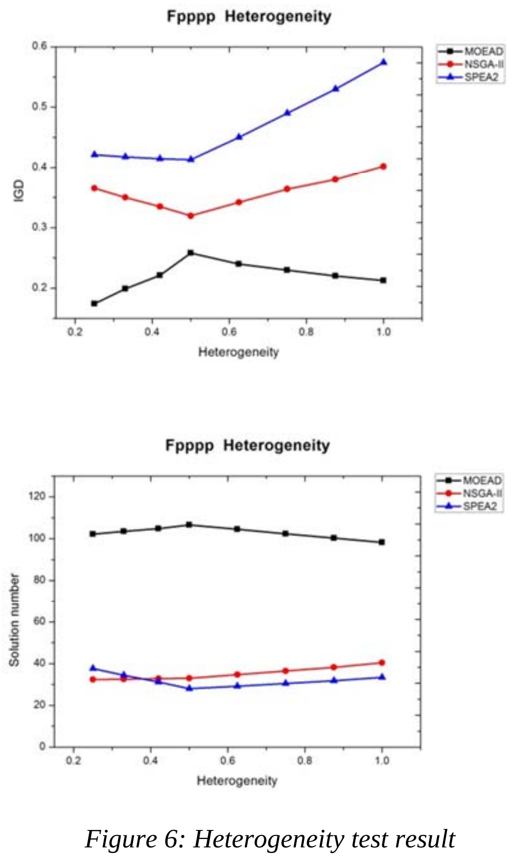


Figure 6: Heterogeneity test result

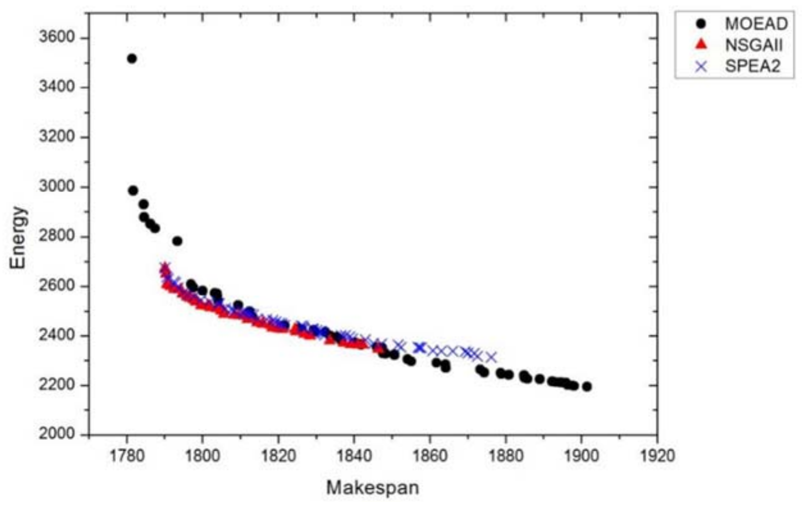


Figure 7: Final Solutions

7 Conclusions

An MOEA/D-based evolutionary algorithm for the scheduling problem is proposed. The algorithm addresses the energy-aware scheduling problem in HCSs. MPCO and GCO are used in MOEA/D to replace the traditional single-point crossover operator. According to the experimental results, MOEA/D demonstrates good performance in solving the benchmarks in terms of convergence, stability, and solution diversity.

However, our algorithm presents several limitations. The value of IGD is still very high, which means the final solutions are far from the true Pareto front. To obtain highly precise and improved solutions, a large amount of execution time is required. Moreover, the number of final solutions is insufficient, especially in solving scalability issues.

In the future, to obtain improved performance under the same parameters, scholars can introduce improvement strategies based on the MOEA/D framework or other EAs to solve difficult instances for large-scale systems. For the scheduling problem, the adoption of new objectives is another research direction.

References

[Ahmad et al., 96] Ahmad, I., Kwok, Y., Wu, M.: Analysis, evaluation, and comparison of algorithms for scheduling task graphs on parallel processors, Second International Symposium on Parallel Architectures, Algorithms, and Networks (I-Span '96), Proceedings, 207-213, 1996.

[Bianchini and Rajamony, 04] Bianchini, R., Rajamony R.: Power and energy management for server systems, Computer, 68-74, 2004.

[Crovella et al., 92] Crovella, M., et al.: Using communication-to-computation ratio in parallel program design and performance prediction, Parallel and Distributed Processing, 1992.

[Deb et al., 02] Deb, K., Pratap, A., Agarwal, S., et al: A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Transactions on Evolutionary Computation, 182-197, 2002.

[Falkenauer, 98] Falkenauer, E.: Genetic algorithms and grouping problems, John Wiley & Sons, Inc, 1998.

[Garey and Johnson, 79] Garey, M., Johnson, D.: Computers and intractability: a guide to the theory of NP-completeness, WH Free Co., San Fr, 1979.

[Guzek et al., 14] Guzek, M., Pecero, J., Dorronsoro, B., et al.: Multi-objective evolutionary algorithms for energy-aware scheduling on distributed computing systems, Applied Soft Computing, 432-446, 2014.

[Herbert and Marculescu, 07] Herbert, S., Marculescu, D.: Analysis of dynamic voltage/frequency scaling in chip-multiprocessors, ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED), 38-43, 2007.

[Huang et al., 12] Huang, Q., Su, S., Li, J., et al.: Enhanced energy-efficient scheduling for parallel applications in cloud, IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012), 781-786, 2012.

[Kołodziej et al., 11] Kołodziej, J., Khan, S., Xhafa, F.: Genetic algorithms for energy-aware scheduling in computational grids, International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 17-24, 2011.

[Koomey, 11] Koomey, J.: Growth in data center electricity use 2005 to 2010, A report by Analytical Press, completed at the request of The New York Times, 2011.

[Lee and Zomaya, 11] Lee, Y., Zomaya A.: Energy conscious scheduling for distributed computing systems under different operating conditions, IEEE Transactions on Parallel and Distributed Systems, 1374-1381, 2011.

[Li and Zhang, 09] Li, H. and Zhang Q.: Multiobjective Optimization Problems With Complicated Pareto Sets, MOEA/D and NSGA-II, IEEE Transactions on Evolutionary Computation, 284-302, 2009.

[Liu et al., 16] Liu, H., Cai W., Shen J., et al.: A speculative approach to spatial‐temporal efficiency with multi‐objective optimization in a heterogeneous cloud environment, Security and Communication Networks, 4002-4012, 2016.

[Pecero et al., 09] Pecero, J.E., Trystram, D., Zomaya, A.: A new genetic algorithm for scheduling for large communication delays, Lecture Notes in Computer Science, 241-252, 2009.

[Peng et al., 09] Peng, W., Zhang, Q., Li, H.: Comparison between MOEA/D and NSGA-II on the multi-objective travelling salesman problem, Multi-objective memetic algorithms, Springer Berlin Heidelberg, 309-324, 2009.

[Semeraro et al., 02] Semeraro, G., Magklis, G., Balasubramonian R., et al: Energy-efficient processor design using multiple clock domains with dynamic voltage and frequency scaling, High-Performance Computer Architecture, 29-40, 2002.

[Tao et al., 14] Tao, F., Feng, Y., Zhang, L., et al.: CLPS-GA: A case library and Pareto solution-based hybrid genetic algorithm for energy-aware cloud service scheduling, Applied Soft Computing, 264-279, 2014.

[Wang et al., 15] Wang, Y., Li, H., Yen, G., et al.: MOMMOP: Multiobjective optimization for locating multiple optimal solutions of multimodal optimization problems, IEEE transactions on cybernetics, 830-843, 2015.

[Xue et al., 17] Xue, Y., Jiang, J., Zhao, B., et al.: A self-adaptive articial bee colony algorithm based on global best for global optimization, Soft Computing, 1-18, 2017.

[Zeng et al., 14] Zeng, X., Zhang X., Song T., et al: Spiking neural P systems with thresholds, Neural computation, 1340-1361, 2014.

[Zhang and Li, 07] Zhang, Q., Li H.: MOEA/D: A multiobjective evolutionary algorithm based on decomposition, IEEE Transactions on Evolutionary Computation, 712-731, 2007.

[Zhang et al., 09] Zhang, Q., Liu W., Li H.: The Performance of a New Version of MOEA/D on CEC09 Unconstrained MOP Test Instances, 2009 IEEE Congress on Evolutionary Computation, 203-208, 2009.

[Zhang et al., 15] Zhang, X., Tian, Y., Cheng, R., et al.: An efficient approach to nondominated sorting for evolutionary multiobjective optimization, IEEE Transactions on Evolutionary Computation, 201-213, 2015.

[Zhu et al., 03] Zhu, D., Melhem R., Childers B.: Scheduling with dynamic voltage/speed adjustment using slack reclamation in multiprocessor real-time systems, IEEE Transactions on Parallel and Distributed Systems, 686-700, 2003.

[Zitzler et al., 01] Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the strength Pareto evolutionary algorithm, Eidgenössische Technische Hochschule Zürich (ETH), Institut für Technische Informatik und Kommunikationsnetze (TIK) Zürich, Switzerland, 2001.