Intelligent Automation And Soft Computing, 2019
Copyright © 2019, TSI® Press
Vol. 25, no. 3, 605–613
https://doi.org/10.31209/2019.100000115

Virtual Machine Based on Genetic Algorithm Used in Time and Power Oriented Cloud Computing Task Scheduling

Tongmao Ma[1,2], Shanchen Pang[1], Weiguang Zhang[1], Shaohua Hao[1]

[1] College of Computer and Communication Engineering, China University of Petroleum, Qingdao Shandong, China, 266580
[2] Departamento de Inteligencia Artificial, Universidad Politécnica de Madrid. Campus de Montegancedo, Boadilla del Monte (Madrid), Spain, 28660

ABSTRACT
In cloud computing, task scheduling is a challenging problem in cloud data center, and there are many different kinds of task scheduling strategies. A good scheduling strategy can bring good effectiveness, where plenty of parameters should be regulated to achieve acceptable performance of cloud computing platform. In this work, combined elitist strategy, three parameters values oriented genetic algorithms are proposed. Specifically, a model built by Generalized Stochastic Petri Nets (GSPN) is introduced to describe the process of scheduling in cloud datacenter, and then the workflow of the algorithms is showed. After that, the effectiveness of the algorithms is found to be valid by the simulations on CloudSim.

KEY WORDS: Cloud computing, Task scheduling, Genetic algorithm, Petri net, Time and power oriented.

1 INTRODUCTION

CLOUD computing is proposed by Eric Schmidt in 2006 in the Search Engine Session [1]. Cloud computing is now known as a kind of typical distribution computing and a kind of pay by use web computing servers. The users can get resources from a variety of resources of suppliers according to their own needs, including servers, storage, applications and services. There are mainly three types of forms to supply resources to customers: Infrastructure as a service (IaaS), Platform as a service (PaaS) and Software as a service (SaaS). And there are some problems in cloud computing need to be solved or improved, and many applications are based cloud computing [2-6]. With distribute computing on a large number of distributed computers, in general, the scale of cloud datacenter is large. At present, many large computer Internet companies, such as Google, Amazon, IBM, Microsoft and Alibaba. All the companies have their own cloud datacenters, which may have thousands or hundreds of thousand servers to achieve the request of the large scale computing tasks, and many tasks need to process deserve high performance. It needs a huge cost to maintain these machines. It is estimated that energy consumption account for 10% of the datacenter operation costs, and in the future, it will continue rapid growth [7]. Therefore, to research the energy consumption of cloud datacenter is of great significance, which is currently a hot topic studied by many researchers.

In recent years, many traditional and effective methods have been proposed for performance analysis and optimization of energy consumption in cloud datacenter. These methods can be divided into three levels: (1) hardware level, (2) servers level and (3) datacenter level.

(1) Hardware level aims to improve the performance or reduce energy consumption of datacenter by designing and optimization of the hardware;

(2) Servers level focuses on improving the performance or reducing energy cost of datacenter by taking effective tasks managing and scheduling strategy, VM migration strategy, manage the state of servers and so on;

(3) Datacenter level takes use of the differences in geographic location, considering the network transmission, energy prices and so on, to reduce the costs of maintaining datacenter.

E. Seo, et al. [8] proposed a DVS enabled algorithm based on time pieces and introduced an algorithm that can continuous monitor the load of the operation system. The method can also adjust the speed of the processor, in order to analysis the energy consumption. Based on the method, it is developed a

CONTACT Shanchen Pang pangsc@upc.edu.cn

Linux-based simulation platform to simulate the algorithm and achieved well performances. Following the research line, Jung et al proposed an abstract model of thermal management of multicore systems [9], where a random process is proposed to obtain the performance and thermal behavior of the system. In certain given levels of system and the peak workloads constraint conditions, they used a continuous process of Markov chain to minimize the dynamic thermal instead of simplifying the problem. Based on the DVFS technique and frequency scaling, Zheng W, et al. [10] proposed a novel scheduling heuristic, which takes the system and application characteristics and the overall energy consumption into account when making frequency scaling decision, and the simulation indicated the heuristic method can achieve significant energy saving. It is introduced by X. Bu, et al. [11] that an algorithm based on reinforcement learning VM and application parameters self-configuration to improve the performance. After that, Yunni Xia, et al. [12] proposed a stochastic analysis method for energy consumption aware and DVS enabled cloud datacenter. In the method, it is firstly proposed a DVS enabled stochastic framework for performance analysis, and then analysis is obtained based on queuing network. Chandnani L, et al. [13] proposed an energy management strategy based on DVS for multiprocessor system. It is verified the strategy can help save energy, guarantee the adaptability and robustness in theory and practice. There are also some methods of optimizing energy consumption based on dynamic pricing [14], and adaptive VM place management and multi-objective optimization methods [15].

Recently, bio-inspired computing methods, such as membrane computing [16-21], DNA computing [22-23], artificial neural network, deep learning and evolution computing, develop rapidly and provide abundant inspiration to construct high-performance computing models and intelligent algorithms, thus enabling powerful tools to solve real-life problems. Some bio-inspired computing methods has been applied in performance analysis and energy consumption optimization of cloud datacenter. Cao et. al. [24] proposed a scheduling algorithm based on genetic intelligence, where simulations showed that the proposed algorithm can effectively reduce the energy consumption of parallel tasks under the condition of guaranteeing the completion time. Such method can be used to optimize the energy consumption of the datacenter. Jianfeng Li et al. [25] proposed a genetic algorithm with double fitness (DFGA). It is obtained by the simulation results that such algorithm is superior to the adaptive genetic algorithms. Zhan, et. al. [26] proposed an algorithm of load balance to schedule the tasks. As results, it obtained good effects of load balance. An improved genetic algorithm is proposed by Amandeep Verma [27] by combing the Min-Min and Min-Max methods to deal with the independent task scheduling in cloud computing. Apart from those methods, there are also some other enhanced genetic algorithm based on task scheduling [28], and improved ant colony algorithm [29].

These methods solve the problem from the viewpoint of bionic and receive good effectiveness. But only a single parameter is analyzed in these methods, and few methods have analyzed the energy consumption, and there are no datacenter models in these methods [30-31]. Combine these methods and learn from bio-inspired computing, we model the cloud datacenter simply with Petri Net, take time and energy consumption into consideration, schedule the tasks by genetic algorithms to reduce the makespan and energy consumption [32].

In this work, combined elitist strategy, three parameter values oriented genetic algorithms are proposed. Specifically, a model built by Generalized Stochastic Petri Nets (GSPN) is introduced to describe the process of scheduling in cloud datacenter, which can help analyze the workflow of the task scheduling, and then the workflow of the algorithms is showed [33]. After that, the effectiveness and practicality of the algorithms is found to be valid by the simulations on CloudSim.

2 PREPARATION

2.1 Cloud datacenter

A cloud datacenter with its conceptual physical architecture is shown in Figure 1. As shown in Figure 1, there are four layers in the architecture, the task request layer is used to store the user's task request, the receiving control layer is responsible for receiving the task request, scheduling layer is responsible for scheduling the task received from the forward layer, the server layer contains a number of servers, in detail, each server in the layer contains more than one virtual machine, and the virtual machine is responsible for the implementation of the task requested.

In Figure 1, the task request layer is to receive tasks as said before, and it can receive only one task request for one time, and the task is transmitted to receiving control layer. After that, once the task arrives in scheduling layer, it is assigned to different virtual machines to execute.

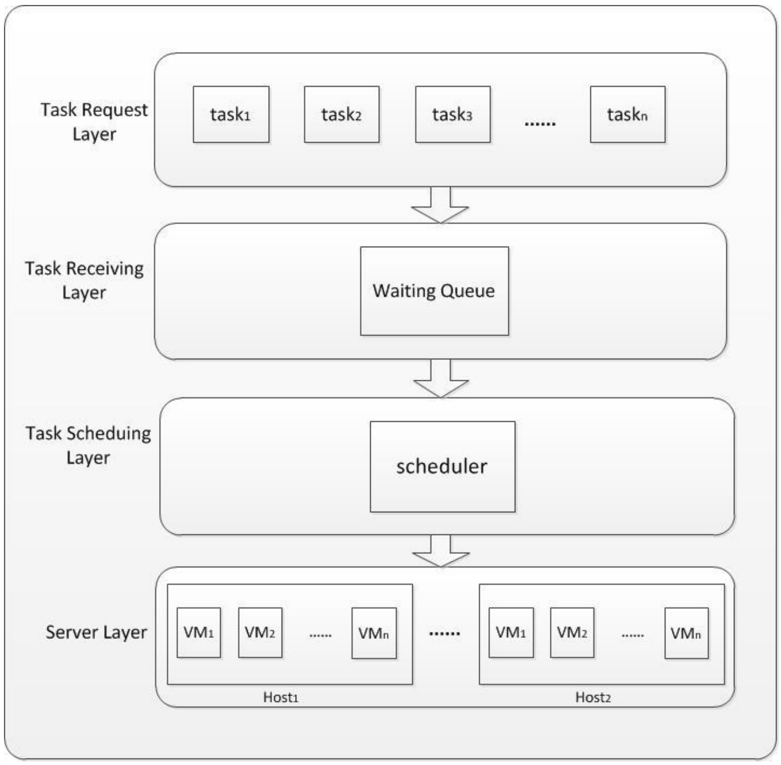

Figure 1. **A cloud datacenter with its conceptual physical architecture.**

2.2 Petri Net

Since Petri net was proposed by Carl Adam Petri in 1962, the theory of Petri net has been constantly enriched and the application of Petri net has gone far beyond the field of computer science, see Stochastic Petri nets and system performance evaluation [34]. Generalized Stochastic Petri Nets (GSPN) is an extension of Stochastic Petri Nets (SPN), in which the transitions can be divided into two kinds, that is instantaneous transition and time transition. Due to the instantaneous transition, it alleviates the state explosion problem.

Definition 1. (Generalized Stochastic Petri Nets, GSPN) A Generalized Stochastic Petri Net is of the form

$$GSPN = (S, T; F, W, M_0, \lambda) \qquad (1)$$

where $(S, T; F)$ is a net, $S = \{p_1, \dots, p_n\}$ is a set of location, $T = \{t_1, \dots, t_n\}$ is a set of transitions, F belongs to $(S \otimes T) \cup (T \otimes S)$, $WF \rightarrow N^+$ is the set of arc functions, $M_0 : S \rightarrow N$ is the initial marking, $\lambda = \{\lambda_1, \dots, \lambda_m\}$ is the average implementation rate of $t_i \in T$, which represents the average implementation times.

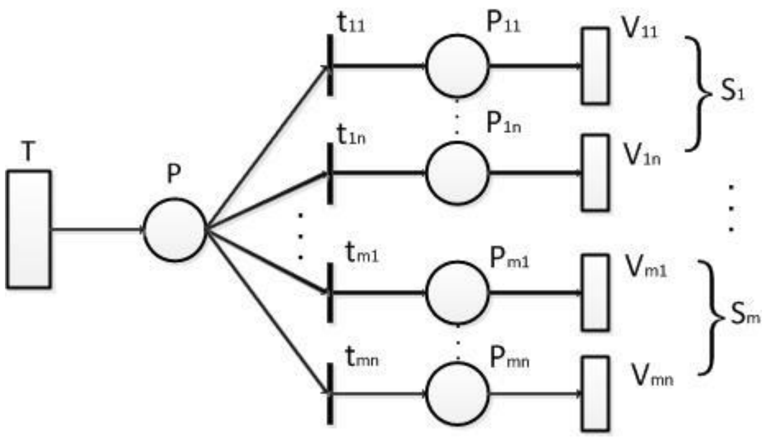

Figure 2. **Model of the process of scheduling in cloud datacenter.**

Combined with Figure 1, in this paper we model the process of tasks scheduling in datacenter with GSPN. The model is shown in Figure 2, where T is the transition represents the tasks arrivals, the number of tasks need to be scheduled is a pre-set value, P is the place represents the scheduler. It holds the task arrived transiently, and then the tasks are scheduled to the virtual machine according to the implementation predicates of t_{ij}, where t_{ij} is the execution of the scheduling or the decision. It sends the task to next layer, with implementation predicate:

$$y_{ij}v_i = v_{ij} \wedge M(p_{ij}) \leq b_{ij} \tag{2}$$

where $M(p_{ij})$ is the number of current markings of p_{ij}, i.e., the number of tasks p_{ij} holds currently; b_{ij} is the capacity of p_{ij}; v_i is the virtual machine index that task i is scheduled to by the scheduling strategy; v_i is the virtual machine index that t_{ij} is corresponding to; t_{ij} is the implement which means the task is sent to p_{ij}, the implementation time of task i is as follow:

$$T_i = Length_i / M_i p_{ij} \tag{3}$$

where $Length_i$ is the length of task i, $M_i p_{ij}$ is the implementation rate of virtual machine v_{ij}, and the implementation time of every virtual machine T_{ij} is the sum of all the time of tasks implemented on virtual machine v_{ij}:

$$T_{ij} = \sum_{i=0}^{n} T_i \wedge v_i = v_{ij} \tag{4}$$

It's the sum of all the execution time of the tasks scheduled to virtual machine v_{ij}. In this paper, it is defined the makespan is the execution finishing time of one host, which can be calculated as follows:

$$T_{imax} = max\{T_{i0}, T_{i1}, \dots, T_{im}\} \tag{5}$$

where m is the number of the virtual machines locate on the host. It is assumed that the power of one host divide into two level according to the Dynamic Voltage Scaling (DVS). The threshold value is set to be N. It indicates that the power of one host is P_{lower} when the number of tasks this host holds is smaller than N, otherwise the power is P_{higher}. The probability of the host stays in lower energy consumption state is:

$$p_{ilower} = T_{ilower} / T_{imax} \tag{6}$$

where T_{ilower} is the time that host i stays in lower energy consumption state, and the probability of the host stays in higher energy consumption state is $p_{ihigher} = 1 - p_{ilower}$, so the energy consumption of host i is:

$$W_i = (p_{ilower} \times P_{lower} + p_{ihigher} \times P_{higher}) \times T_{imax} \tag{7}$$

3 GENETIC ALGORITHM

GENETIC algorithm (GA) is a computational model. It simulates the natural selection and genetic mechanism of Darwin's biological evolution, and it is an effective method of searching the optimal solution through simulating the process of natural evolution. We define here two basic task scheduling GAs with different fitness functions. The first one is time oriented genetic algorithm TOGA in which the execution time of the time is a factor of the fitness function, and another one is power consumption oriented genetic algorithm POGA, the corresponding factor of the fitness function is power consumption. As a result, we combine these two approaches and get a comprehensive oriented genetic algorithm POGA, and we want to get a good result in execution time and power consumption at the same time by the scheduling algorithm. What's more, all these algorithms apply an elitist strategy by which an optimal individual in per generation can be directly copied and saved to the next generation. The chromosome encoding, population initialization, adaptation degree function, crossover and mutation in these algorithms are as follows.

3.1 Chromosome Coding

The paper uses virtual machine (VM) indexes to code the chromosome. The length of chromosome is the number of tasks that need to be scheduled, and each gene in the chromosome represents the corresponding VM that is used to execute the task, shown as:

$$C_i = \{C_{i0}, C_{i1}, \dots, C_{in}\} \tag{8}$$

where i is the index of the chromosome, n is the number of tasks, and $C_{ij}, (0 \leq j \leq n)$ is an integer that indicates task is executed on this VM.

3.2 Fitness Function

The individual with larger fitness is more likely to survive, since it is aimed to find individuals with smaller makespan and smaller energy, the fitness functions of GAs is formulated as follows:

$$f_1 = 1/makespan \tag{9}$$

where $makespan = \max(T_{imax}), i = 0, \dots, m$, indicating the maximum completion time.

$$f_2 = 1/\sum_{i=1}^{m} W_i \tag{10}$$

$$f = 1/makespan + f_2 = 1/\sum_{i=1}^{m} W_i \tag{11}$$

To calculate the fitness value of each chromosome, it needs to calculate the energy cost. We take here DVS into consideration, and give two different power according to the number of the hosts hold. If the number is smaller than the threshold value of the number of the host holds that we have set, the host work at the lower power consumption pattern;

otherwise, they work at the higher power consumption pattern.

3.3 *Crossover*

Select several individuals from current population randomly and find the best one. After repeating twice, we crossover these two results. Every gene is crossed under the guidance of a crossover probability and a new individual is created and saved to next generation. Repeat the above processing until a new generation is completed.

3.4 *Mutation*

We operate mutation to every gene regulated by a pre-defined mutation rate. Traverse all the genes of an individual, and for each gene, if the mutation probability condition can be satisfied, set a random integer value to this gene, which is less than the number of VM and more than zero to guarantee there is a VM can receive and process the task, the size of the mutation rate will affect the probability of new individuals in the next generation.

3.5 *Workflow of our method*

The basic workflow of the simulation of the algorithm is shown as follows.

```
//Begin
//Initialise the could datacenter
   CreateVM (int userID, int vms)
   //create all the VM in the datacenter
   CreateCloudlet (int userID, int cloudlets)
   //create all the tasks
   CreateDatacenter (String name)
   //create the datacenter
//Initialise the population
   For(i=0n)
   {
      Create a new individual;
      Save the individual to the population
   }
//EvolvePopulation
   Repeat until the finish condition
   saveIndividual (0, pop.getFittest())
   //save the fittest individual to the new population as the first individual
   Crossover (indiv1, indiv2)
   //choose two individuals to crossover and create a new individual
   For (i=0: n)
   {
      Mutate(individual(i))
      //mutate all the Individual except the first Individual
   }
   bindCloudletsToVmsGA (DatacenterBroker broker, Individual individual)
   //allocate the tasks to the VMs accourding to the results gets by the GA
   printCloudletList (List<Cloudlet> list)
   //get the print the simulation results
//End
```

4 EXPERIMENTAL RESULTS AND ANALYSIS

THE simulation experiments are operated on the CloudSim to analysis the effects of the proposed algorithm. We set the length of the population to be 10 and the threshold value about DVS to be 4000. The number of VMs is fixed to be 50, and the number of the hosts is 15. The mips of the VMs is 250 and the length of the tasks as 2000. Based on these settings, we generate the tasks randomly, carry on some experiments for different purposes, the results and the analysis is showed as follows. For the first experiment, the results of the execution times of the proposed three GAs are shown in Figure3, and we record 10 times of each algorithm at the condition of 10 thousand tasks. Then, the results of makespan, average task time and energy of the datacenter for different task numbers are shown in Figures 4, 5 and 6.

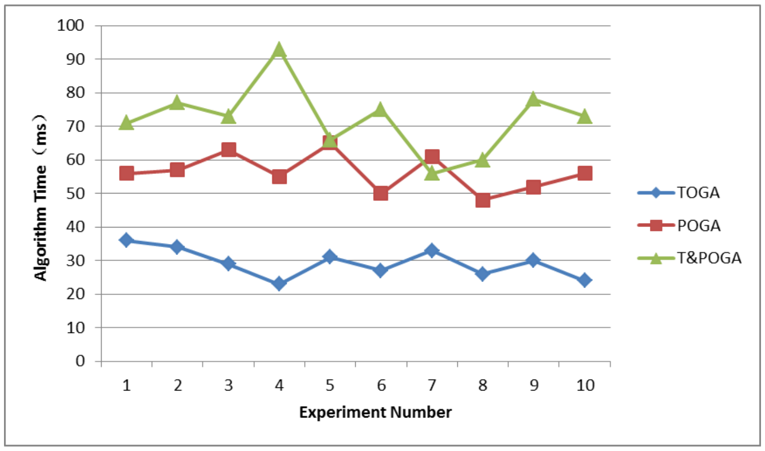

Figure 3. **Execution times of three Gas.**

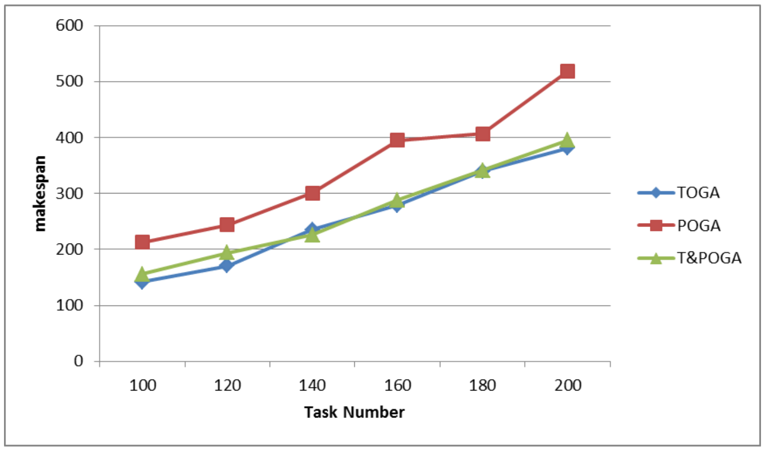

Figure 4. Makespan of three GAs.

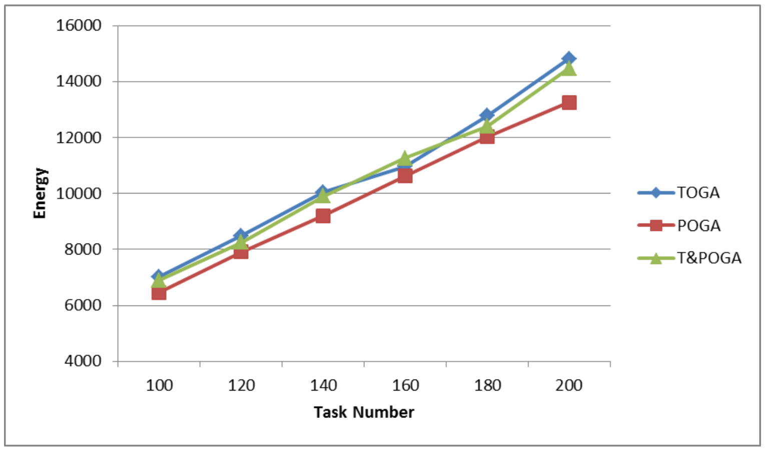

Figure 5. Energy consumption of three GAs.

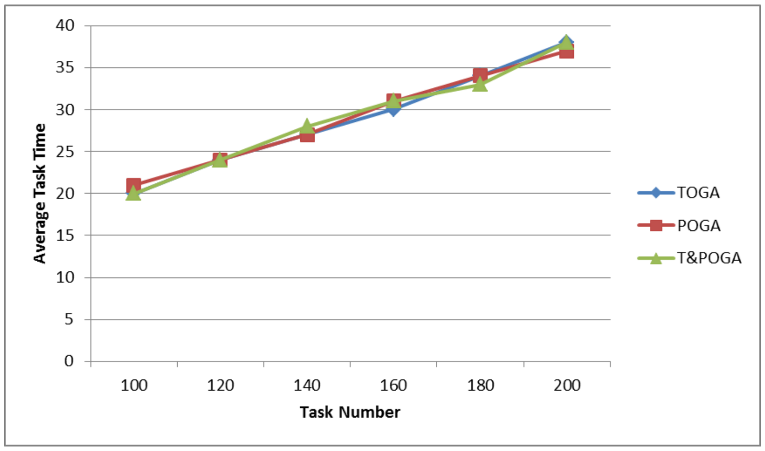

Figure 6. Average task time of three GAs.

It is shown in Figure 3 that the execution times of three algorithms are different. The execution time of T&POGA is smaller than the sum of the execution times of TOGA and POGA. Although T&POGA is more complicated than the other two algorithms, it saves time in some way, and sometimes it saves time than the POGA. These results suggest that the T&POGA is well since they can schedule the tasks more quickly and with acceptable cost of power.

It is shown in Figure 4 that the makespan gets bigger with the increase of task number, the makespans got by TOGA and T&POGA are approximate. They are obviously smaller than the makespan got by POGA. The obvious saving of makespan suggest TOGA and T&POGA are very effective in finishing the tasks with small VM time, T&POGA take power into consideration, but it doesn't bring any increase of the makespan.

In Figure 5, it shows the energy that the datacenter consumes increases with the number adding of the tasks. The energy for PGA is always the lowest, which suggests that PGA is effective in saving energy. What's more, T&POGA is more likely energy efficient than TOGA. Although the differences among the algorithms are not very obvious, but it also suggests that the T\&POGA is good for saving energy. It might be the related parameter settings that is not compatible that results in the little improvement.

In Figure 6, it shows that the average time of the task is always almost equivalent in any cases. This suggests these algorithms can bring the same effects in the average time of the task, mostly because the average time of the task depends on the processing speed of the hosts.

5 CONCLUSIONS

IN this paper, we mainly propose three different performance oriented GAs based on the cloud datacenter model build by Petri Net, and the analysis of the effectiveness by the proposed GAs are obtained. The simulations support the result that the proposed GAs is capable of improving the performance of the cloud datacenter by scheduling tasks, that is to say, the TOGA can save the execution time and the POGA can cut down the power consumption. Particularly, it is obtained that T&POGA can save more time and energy at the same time.

In the future work, we would try to take more factors into consideration not only time and power consumption, but also, for example, the state of the hosts can influence the energy of the datacenter; another promising future work directions is to try to use other bio-computing methods to solve some problem in cloud datacenter.

6 ACKNOWLEDGEMENT

THIS work was partially supported by the National Nature Science Foundation of China under Grant Nos.61170183, 61172063, 61272093, 61572522, 61572523 and 61402533, and special fund project for work method innovation of Ministry of science and technology of China No.2015IM010300.

7 DISCLOSURE STATEMENT

NO potential conflict of interest was reported by the authors.

8 REFERENCES

Beloglazov, A., R. Buyya, Y. C. Lee, A. Zomaya (2011). A taxonomy and survey of energy efficient data centers and cloud computing systems [J]. Advances in Computer, 82: 47-111.

Bu, X., J. Rao, and C.-Z. Xu (2012). Coordinated self-configuration of virtual machines and appliances using a model-free learning approach. IEEE Transactions on Parallel and Distributed Systems, 24(4):681-690.

Cao, Jie, Guosun Zeng (2013). Scheduling Method for Parallel Task of Dynamic Energy-aware of Computing Resources in Cloud Environment[J]. Computer Science, 40(10):39-44.

Chandnani L, Kapoor H K (2013). Formal Approach for DVS-Based Power Management for Multiple Server System in Presence of Server Failure and Repair[J]. IEEE Transactions on Industrial Informatics, 9(1):502-513.

Chuang Lin (2005): Stochastic Petri nets and system performance evaluation. Tsinghua University Press.

Fu Z, Ren K, Shu J, et al (2016). Enabling personalized search over encrypted outsourced data with efficiency improvement. IEEE transactions on parallel and distributed systems, 27(9): 2546-2559.

Fu Z, Wu X, Guan C, et al (2016). Toward efficient multi-keyword fuzzy search over encrypted outsourced data with accuracy improvement. IEEE Transactions on Information Forensics and Security, 11(12): 2706-2716.

Gartner Group [Online]. http://www.gartner.com/. (2010).

Jiang K, Song B, Shi X, et al (2012). An Overview of Membrane Computing. Journal of Bioinformatics & Intelligent Control, 1(1):17-26.

Jung, H. S., P. Rong, and M. Pedram (2008). Stochastic modeling of a thermally-managed multi-core system, in Proc. Annu. Design Autom. Conf., Anaheim, CA, USA, Jun. pp. 728C733.

Kaur R, Kinger S. Enhanced Genetic Algorithm based Task Scheduling in Cloud Computing[J]. International Journal of Computer Applications, 2014, 101:1-6.

Li, Jianfeng, Jian Peng. Task scheduling algorithm based on improved genetic algorithm in cloud computing environment[J]. Journal of Computer Applications, 2011, 31(01):184-186.

Li, Qiang, Qinfen Hao, Limin Xiao,et al (2011). Adaptive Management and Multi Objective Optimization for Virtual Machine Placement in Cloud Computing. Chinese Journal of Computers, 34(12):2253-2264.

Lin W C, Wu C C, Yu K, et al (2017). On the Use of Genetic Algorithm for Solving Re-entrant Flowshop Scheduling with Sum-of-processing-times-based Learning Effect to Minimize Total Tardiness. Intelligent Automation and Soft Computing, 2017:1-11.

Liu J, Xu S, Zhang F, et al (2016). A hybrid genetic-ant colony optimization algorithm for the optimal path selection. Intelligent Automation and Soft Computing, 23(2):1-8.

Pan Z, Lei J, Zhang Y, et al (2016). Fast motion estimation based on content property for low-complexity H. 265/HEVC encoder. IEEE Transactions on Broadcasting, 62(3): 675-684.

Sea, E., S. Park, J. Kim, and J. Lee (2008). TBS: A DVS algorithm with quick response for general purpose operating systems, J. Syst. Arch., vol. 54, nos. 1C2, pp. 1C14, Jul.

Shi, Xiaolong, Congzhou Chen, Xin Li, Tao Song, Zhihua Chen, Zheng Zhang, Yanfeng Wang (2015). Size Controllable DNA Nanoribbons Assembled from Three Types of Reusable Brick Single-Strand DNA Tile, Soft Matter. 11(43): 8484-8492.

Song, Tao, Linqiang Pan (2015). On the Universality and Non-universality of Spiking Neural P Systems with Rules on Synapses, IEEE Trans. on NanoBioscience, 148960-966.

Song, Tao, Linqiang Pan (2015). Spiking Neural P Systems with Rules on Synapses Working in Maximum Spikes Consumption Strategy, IEEE Trans. on NanoBioscience, 14(1): 37-43.

Song, Tao, Linqiang Pan (2015). Spiking Neural P Systems with Rules on Synapses Working in Maximum Spiking Strategy, IEEE Trans. on NanoBioscience, 14(4), 465 C 477.

Sun Y, Li X, Mao Y, et al (2017). PROXZONE: One Cloud Computing System for Support PaaS in Energy Power Applications. Intelligent Automation and Soft Computing, 23(4):1-11.

Verma A, Kumar P. Independent Task Scheduling in Cloud Computing by Improved Genetic Algorithm[J]. 2012, 2(5):111-114.

Wang, Xun, Tao Song, Faming Gong, Pan Zheng (2016). On the computational power of spiking neural P systems with self-organization, Scientific Reports, DOI:10.1038/srep27624.

Wei Wang, Junzhou Luo, Aibo Song (2013). Dynamic Pricing Based Energy Cost Optimization in Data Center Environments. Chinese Journal of Computers, 36(03):599-612.

Xia Z, Wang X, Sun X, et al (2016). A Secure and Dynamic Multi-Keyword Ranked Search Scheme over Encrypted Cloud Data. IEEE Trans. Parallel Distrib. Syst., 27(2): 340-352.

Xia, Yunni, MengChu Zhou, Xin Luo, Shanchen Pang, Qingsheng Zhu (2015). A Stochastic Approach to Analysis of Energy-aware Dvs-enabled Cloud Datacenters. IEEE Trans. on Systems, Man, and Cybernetics: Systems, 45(1): 73-83.

Yin, Yuyu, Fangzheng Yu, Yueshen Xu, Lifeng Yu, Jinglong Mu (2017). Network Location-Aware Service Recommendation with Random Walk in Cyber-Physical Systems. Sensors 17(9): 2059.

Yin, Yuyu, Song Aihua, Gao Min, Yueshen Xu, Wang Shuoping (2016). QoS Prediction for Web Service Recommendation with Network Location-Aware Neighbor Selection. International Journal of Software Engineering and Knowledge Engineering 26(4): 611-632

Yin, Yuyu, Yueshen Xu, Wenting Xu, Min Gao, Lifeng Yu, Yujie Pei (2017). Collaborative Service Selection via Ensemble Learning in Mixed Mobile Network Environments. Entropy 19(7): 358.

Yongjun Ren, Jian Shen, Jin Wang, Jin Han, and Sungyoung Lee (2015). "Mutual Verifiable Provable Data Auditing in Public Cloud Storage," Journal of Internet Technology, vol. 16, no. 2, pp. 317-323.

Zha, Yinghua, Jingli Yang (2013). Task scheduling in cloud computing based on improved ant colony optimization[J]. Computer Engineering and Design, 34(05):1716-1719.

Zhan Z H, Zhang G Y, Ying-Lin, et al. Load Balance Aware Genetic Algorithm for Task Scheduling in Cloud Computing[M]// Simulated Evolution and Learning. Springer International Publishing, 2014:644-655.

Zheng W, Huang S (2015). An adaptive deadline constrained energy-efficient scheduling heuristic for workflows in clouds[J]. Concurrency & Computation Practice & Expe rience, 27(18):5590-5605.

9 NOTES ON CONTRIBUTORS

Shanchen Pang received the graduation degree from the Tongji University of Computer Software and Theory, Shanghai, China, in 2008.He is a Professor from the China University of Petroleum, Qingdao, China. His current research interests include theory and application of Petri Net, Service computing, Trusted computing.

Tongmao Ma received the B.S. degree in China University of Petroleum, Qingdao, in 2015. He is a graduate student of China University of Petroleum, QingDao, China. His current research interests include Could Computing, Membrane Computing, and Bioinformatics.

Weiguang Zhang received the B.S. degree in Jinan University, Jinan, in 2015. He is a graduate student of China University of Petroleum, QingDao, China. His current research interests include Could Computing, Big Data.

Shaohua Hao received the graduate degree from China University of Petroleum, Qingdao, China. Her research interests include biocomputing, membrane computing.