Applying Genetic Algorithm to Rearrange Cloud Virtual Machines for Balancing Resource Allocation

Yu-Lun Huang
National Chiao Tung University
ylhuang@g2.nctu.edu.tw

Zong-Xian Li
National Chiao Tung University
norego379482.eed00@g2.nctu.edu.tw

Abstract - Advance of the novel cloud computing technologies reduces hardware cost by sharing resources operated by a cloud service provider (CSP). If an inappropriate distribution strategy is applied resulting in unbalanced resource allocation, the virtual machines on the overloading hosts cannot perform as expected. In this paper, we present a framework to apply genetic algorithms for balancing resource allocation and reallocation to help CSP managing resources of physical machines. The framework collects CPU utilization data for the control server to construct a proper distribution strategy using the proposed algorithm, GASd (Genetic Algorithm-based Strategy Daemon). GASd allows CSP to select a dimension of load (CPU and memory of IO utilization) and to control the migration ratio of virtual machines running on different physical machines. We conduct several experiments to evaluate GASd and the result shows that GASd can come out an optimized strategy when considering different load dimensions in distributing VMs on physical machines. We further show that GASd can determine an optimized strategy in distributing 10000 virtual machines on 2500 physical machines, in 139 seconds (using TS selection) with a genetic distance of 6.25%.

Keywords - cloud resource management, genetic algorithm, virtual machine rearrangement

I. INTRODUCTION

In a cloud system, several physical machines (PMs) are used to provide resource for virtual machines (VMs) by adopting modern virtualization technologies. Due to the variety of resource required by different VMs, a proper resource management strategy is needed for balancing resource allocation among VMs and to guarantee the performance of VMs running in the cloud system. Since a cloud service provider needs to manage a large amount of resource in a cloud system, it is not easy to generate a proper distribution strategy.

Many researchers intend to leverage genetic algorithms to design a resource allocation algorithm to solve the above issue, such that a cloud service provider (CSP) can make a good distribution strategy for VMs and ensure the performance of the VMs. Considering a situation of managing 20 VMs on 5 PMs, the CSP needs to compare 520 possible combinations to find out proper strategies for allocating resource for the VMs. To solve such a difficult problem, genetic algorithms (GA) are adopted to make the distribution strategy for CSP.

GA is a kind of global search algorithm developed by John Holland in the late 1960s and early 1970s. The fundamental concept of GA is to apply evolution to different generations, then the generations may gradually converge to an optimized solution. Running GA on a control server, a CSP can rearrange VMs to different PMs to better utilize the physical resource on PMs, as illustrated in Fig. 1.

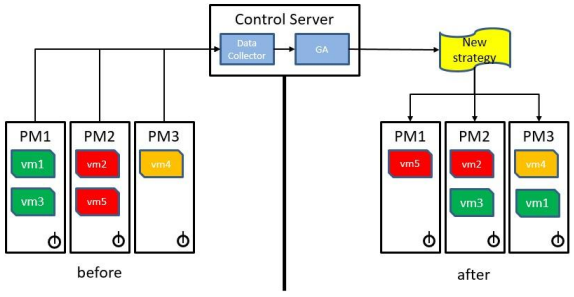

Fig. 1: The application of Genetic Algorithm

II. THE EXISTING GA-BASED RESOURCE ALLOCATION ALGORITHMS

In recent years, some algorithms based on GA were proposed to help allocating physical resource for virtual machines in a cloud system. Jinhua Hu et.al [1] designed an algorithm (called HGA) to allocate resource and simultaneously keep load balanced on CPU in 2010. In 2012, Shin Chen et.al [2] presented another algorithm to allocate resource by considering more factors.

Hu's algorithm searches for an optimized solution to reallocate physical resource such that all CPUs are running in a balanced state while keeping the migration cost as low as possible. The algorithm applies a high mutation rate in the former generations, and then gradually reduce the mutation rate to speed up the converge. In Hu's algorithm, only one factor is considered, which is no longer practical since memory

and network I/O are important factors in managing virtual machines.

Shin Chen et.al presented HGA to reallocate resource for VMs by considering more factor, less migration cost and fewer active physical machines. Three dimensions (load, amount of active PMs and migration cost) are used in HGA. In HGA, load dimensions, instead of load balance, are considered. Two PMs are considered to be load balanced if they have the same ratio of CPU utilization and network throughput.

The above algorithms use fitness functions to evaluate the survival ability of an individual data in GA, where a large fitness value implies the high survival ability. Since load deviation is the denominator in the fitness functions, a small decrease of load deviation may cause a dramatic increase in fitness value and lead to a distorted situation.

In this paper, we propose a framework for resource management (VRMF) and a GA-based strategy algorithm (GASd) by designing unfitness functions and unfitness values to evaluate individual's survival ability. To guarantee the service provided by a CSP, no idle PM is allowed in VRMF. Hence, only system load and migration cost are considered in GASd.

III. VRMF

VRMF is a framework responsible for allocating resource and keeping load balancing among all PMs/ VRMF is composed of two components: U-probe and Resource Manager. U-probe operates in each VM and Resource Manager (running GASd) executes in the cloud control server (CCS), as illustrated in Fig. 2. Two load dimensions, including CPU utilization and memory utilization are considered in GASd and hence U-probe collects the two factors for GASd by invoking /proc/stat and /proc/meminfo.

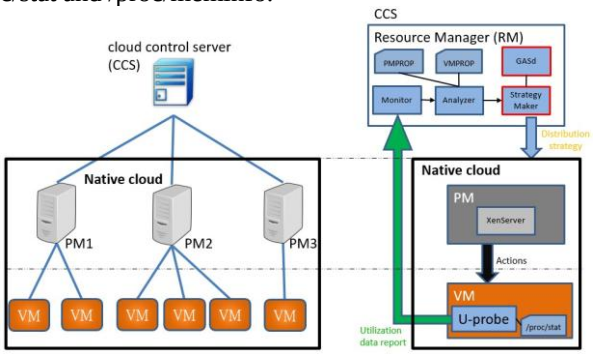

Fig. 2: The system architecture of VRMF: (a) The physical cloud computing environment (b) The software corresponding to physical environment

Resource Manager receives the utilization data from U-probe and then applies GASd to make a new distribution strategy to keep load balanced. As shown in Fig. 3, Resource Manager is composed of Monitor, Analyzer and Strategy Maker. Initially, Resource Manager keeps the hardware specification of each PM and resource requirement of each VM in PMPROP and VMPROP, respectively. Monitor collects and stores the raw probing data from U-probe forward the raw data to the Analyzer. The Analyzer first smoothes the sequence of the raw data to better evaluate the resource utilization of a VM. The exponentially weighted moving average (EWMA), a first-ordered low-pass filter function, is adopted to smooth the sequence of utilization data in Analyzer. The input of Analyzer is a sequence of raw probing data, while the output is a single average utilization of a VM. Then, Strategy Maker runs GASd and generates new distribution strategy to keep load balanced among PMs in a cloud system.

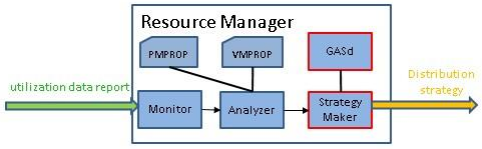

Fig. 3: The Resource Manager

GASd is an algorithm based on genetic algorithm (GA). GASd runs in Strategy Maker in Resource Manager to determine a strategy and keep load balanced among PMs in terms of both CPU and memory. GASd concerns about the migration cost when a new strategy was made. The migration cost should consider the environment factors, such as VM's image size, the disk I/O rate of source PM and destination PM and the network bandwidth between the two PMs. These factors are decided by use cases, hence, GASd estimated migration cost by the ratio of moving VM amount to total VM amount instead. Before conducting GASd, we define the workload caused by VM and its unfitness functions.

Taking GA as its base, GASd performs a series of operations, like selection, crossover and mutation. Finally, the output of GASd can be an optimized distribution strategy according to the utilization data obtained in the previous period, as illustrated in Fig. 4.

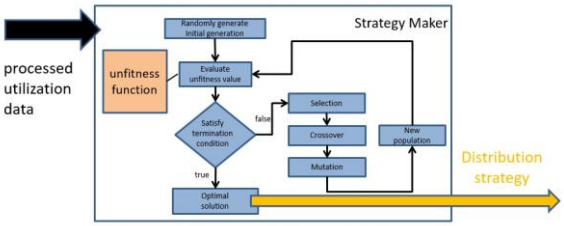

Fig. 4: The workflow of GAsd

GASd consists of four steps: Initial, update, selection, crossover, and mutation, explained in the following subsections:

A. Initial

Initially, GASd loads the utilization data and environment parameters, the hardware specification of each PM and the resource requirement of each VM are generated.

B. Update

GASd updates its current solution in the 'update' step. At the beginning of update, the proposed unfitness function is used to calculate every individual's unfitness value for this generation, then updates the current solution according to these new unfitness value.

In the first generation, this step simply chooses the individual with the lowest unfitness value to be the current solution. In the following generations, this step is conducted after the operation of mutation, and chooses the individual with the lowest unfitness value to compare with the current solution.

If the chosen individual has a lower unfitness value than the current solution, it takes place of the current solution. When GASd finishes its execution, this current solution becomes the final solution. Except updating the current solution, this step also issues a termination condition to determine the termination. The termination condition generally occurs either when reaching the maximum iteration or when the user requirements are satisfied. For example, when the termination condition is set to 300 iterations or standard deviation becomes smaller than 5%. If current solution reaches a standard deviation smaller than 5%, GASd terminates its execution.

C. Selection

'Selection' step is responsible for choosing parent individuals from current generation and putting the chosen individuals into the selected pool for the next generation. In the view of diversity, selection step absolutely reduces diversity of the next generation, and prompts GASd to converge. Basically, selection step tends to choose those individuals with lower unfitness values. With different selection methods, the tolerance to those bad individuals is different, and the performance of GASd is also different. In the paper, we apply three selection methods in GASd: Proportional Roulette Wheel Selection (PRWS), Rank-based Roulette Wheel Selection (RRWS) and Tournament Selection (TS).

1) PRWS (Proportional Roulette Wheel Selection)
PRWS first maps every individual on a wheel according to their fitness value. And the occupied proportion of an individual decided by its fitness value. The bigger fitness value leads to more occupation on this wheel. After mapping the individuals, this wheel starts to spin, and choose an individual by the pointer when the wheel stops each time until reaching the population count as shown in Figure 5.

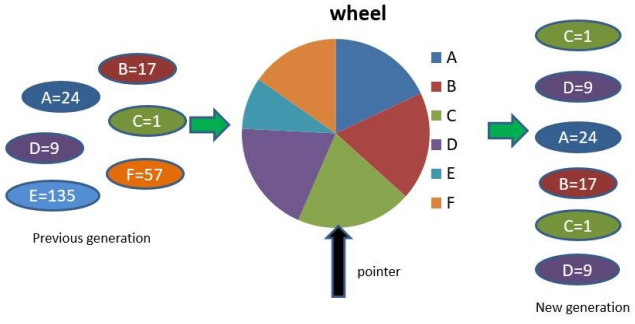

Fig. 5: PRWS

Each time this wheel stops spinning, the pointed individual is selected by the pointer. With this mechanism, it is obvious that the bigger proportion an individual occupied, the higher probability for being selected.

2) RRWS (Rank-based Roulette Wheel Selection)
RRWS uses a rank value to map individuals on the wheel instead of their unfitness values. This idea helps GASd prevent from premature (early converge). When a little amount of individuals is much better than the other individuals, these stronger individuals have very high probability being chosen to become the parents of the next generation. This situation drastically reduces the diversity of population, and makes GASd to give up the potential of those weak individuals. So, RRWS uses a rank value to map individuals on wheel to eliminate the bias caused by the big gap unfitness values. Note that, in RRWS, a parameter SP is used to adjust the tolerance to poor performance individuals. Comparatively, larger SP normally has smaller tolerance to the individuals with poor performance individuals.

3) TS (Tournament Selection)
TS selects several candidates from the previous generation, and puts the best candidates into the pool. The parameter ts in TS stands for the proportion of candidates to the population size. In Fig. 6, 3 out of 6 individuals are selected (since ts is set to 3), and then the best one is selected to be the final result of TS.

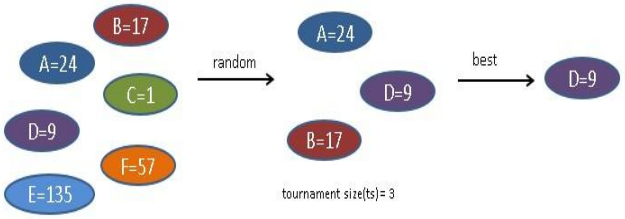

Fig. 6: TS with ts=3

D. Crossover

Crossover is the key to a new generation. Crossover exchanges some features of an individual, increases the diversity in generating new individuals and approaches to an optimized solution. In this step, GASd randomly chooses a pair of individuals the corss point to obtain new individuals with new chromosome, as shown in Fig. 7.

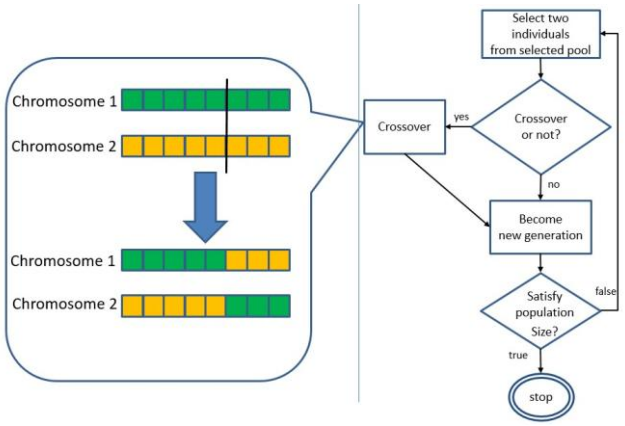

Fig. 7: The flow chart of crossover and its detail.

E. Mutation

Mutation further increases the diversity of the next generations. GASd selects one bit of the chromosome and changes its value randomly. The mutation repeats until all new individuals are generated. Fig. 8 indicates the flow chart of mutation and its detail.

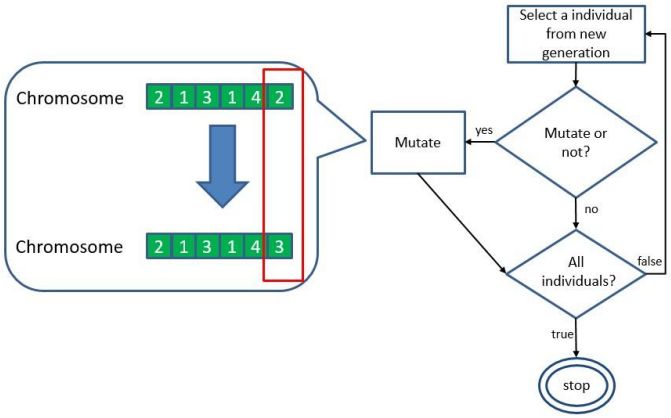

Fig. 8: The flow chart of mutation and its detail.

IV. EVALUATION

VRMF and GASd is realized in a cloud system running Xen as its hypervisor. Each PM installs XenServer as its host OS, and these PMs connect to the cloud control server through network. The cloud control sever runs a program call XenCenter to manage the PMs. The control system designed by Xen allows the administrator easily to control every PMs and VMs through XenCenter.

Since XenServer provides hypervisor-based virtualization interfaces, XenServer can be directly installed on a PM and can manage hardware resource easier. In such a system, Dom0, a special VM, provides the service console and the management tools, while DomU, managed by Dom0, runs guest OSs on the PM. Resource requests from DomU are managed by Dom0 for better resource control, as illustrated in Fig. 9.

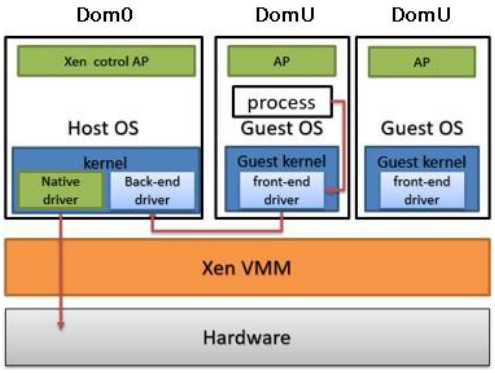

Fig. 9: XenServer's system architecture.

In the above Xen-based cloud system, every VM runs U-probe to collect utilization data for GASd. In the paper, we only show how GASd works for rearranging VMs after analyzing the utilization data from U-probes to keep load balanced in the whole cloud system. In this experiment, we adopt RRWS with SP =2 for selecting individuals and generating new individuals. The experimental cloud system runs 4 PMs, each of them has 4 cores with 16GB memory. In the beginning, 16 VMs were launched on the 4 PMs. Table I shows the CPU loads and memory requirements for each VM.

TABLE I: CPU loads collected by U-Probe in the Experiment

VM ID	CPU loads	memory requirements (MB)	VM ID	CPU loads	memory requirements (MB)
vm1	28.	2048	vm9	18.	2560
vm2	23.	1024	vm10	9.2	3584
vm3	17.	2048	vm11	38.	4096
vm4	16.	1536	vm12	7.3	2048
vm5	12.	3072	vm13	8.1	3072
vm6	22.	4096	vm14	28.	4096
vm7	33.	2048	vm15	24.	1024
vm8	40.	2048	vm16	26.	1536

After performing GASd in the cloud control server, different distribution strategies may come out for different weight of CPU loads and memory requirements. If we assign 0.9 for weighting the CPU loads and 0.1 for memory requirements, we may obtain Distribution Strategy I (See Fig. 10). If we concern more on memory requirements and assign its weight to 0.9, then we may obtain Distribution Strategy II (See Fig. 11). Moreover, after applying Distribution Strategy I, we obtain a deviation of 0.32 for CPU loads and 7.44 for memory requirements. This means that the four PMs in the cloud system have similar computational loads. After applying Distribution Strategy II, we obtain a deviation of 4.02 for CPU loads and 1.81 for memory requirements, which means that the four PMs have similar memory requirements.

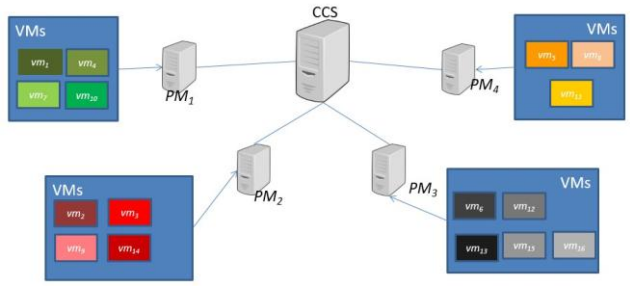

Fig. 10: Distribution Strategy I (concern more on CPU loads)

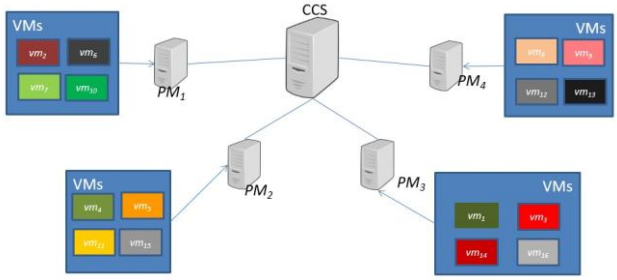

Fig. 11: Distribution Strategy II (concern more on memory requirements)

From the experiment, we show that GASd is able to make a distribution strategy according to the policy given by the cloud service provider.

V. SCALABILITY

We have proved the reliability of GASd to find a optimal distribution strategy in experiments. In the experiments, only few PMs and VMs are used. Here, we show GASd's performance in a real environment. We run 2000~10000 VMs on 500~2500 PMs with different selection methods in GASd. Fig. 12 shows the execution time and Fig. 13 shows the distances of each case. In short, GASd can come out a distribution strategy mapping 10000 VMs on 2500 PMs in 139 seconds with distance of 6.25% (if TS selection is applied). This proves the scalability of GASd.

Fig. 12: The execution time for different amounts of machines (second)

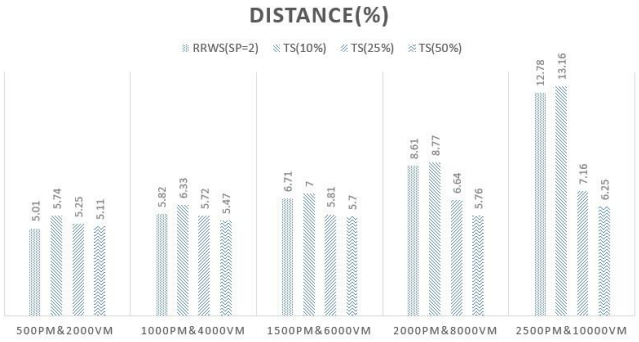

Fig. 13: The distance of different amounts of machines (%)

VI. CONCLUSION

In this article, we present a resource management framework (VRMF) for cloud service providers. VRMF intends to reallocate resources for each VM and keep load balanced among PMs. A probing software (U-probe) is installed in each VM to collect the utilization data of the VM. Then, the Resource Manager running on the cloud control server launches the proposed GA-based algorithm to generate an optimized distribution strategy according to the preference of a cloud service provider. We conduct a simple experiment to show that different strategies may be obtained for different preferences given by the cloud service provider. Further, we discuss the scalability of GASd, and prove that GASd can complete the execution in 139 seconds (using TS selection) with distance equal to 6.25% when allocating 10000 VMs on 2500 PMs.

REFERENCES

[1] J. Hu, J. Gu, G. Sun, and T. Zhao, "A scheduling strategy on load balancing of virtual machine resources in cloud computing environment," Parallel Architectures, Algorithms and Programming (PAAP), 2010 Third International Symposium on, pp. 89–96, Dec 2010.

[2] S. Chen, J. Wu, and Z. Lu, "A cloud computing resource scheduling policy based on genetic algorithm with multiple fitness," Computer and Information Technology (CIT), 2012 IEEE 12th International Conference on, pp. 177–184, Oct 2012.

[3] N. M. Razali and J. Geraghty, "Genetic algorithm performance with different selection strategies in solving tsp," Proceedings of the World Congress on Engineering, vol. 2, Jul. 2011.

[4] R. Patel, M. M. Raghuwanshi, and L. G. Malik, "An improved ranking scheme for selection of parents in multi-objective genetic algorithm," Communication Systems and Network Technologies (CSNT), 2011 International Conference on, pp. 734–739, June 2011.

[5] L. Zhang, H. Chang, and R. Xu, "Equal-width partitioning roulette wheel selection in genetic algorithm," 2012 Conference on Technologies and Applications of Artificial Intelligence, pp. 62–67, Nov 2012.

[6] J. Grover and S. Katiyar, "Agent based dynamic load balancing in cloud computing," Human Computer Interactions (ICHCI), 2013 International Conference on, pp. 1–6, Aug 2013.

[7] Z. Gao, "The allocation of cloud computing resources based on the improved ant colony algorithm," Intelligent Human-Machine Systems and Cybernetics (IHMSC), 2014 Sixth International Conference on, vol. 2, pp. 334–337, Aug 2014.

[8] V. Behal and A. Kumar, "Cloud computing: Performance analysis of load balancing algorithms in cloud heterogeneous environment," Confluence The Next Generation Information Technology Summit (Confluence), 2014 5th International Conference, pp. 200–205, Sept 2014.

[9] F. Bei, "An improved ant colony algorithm based on distribution estimation," Intelligent Systems Design and Engineering Applications (ISDEA), 2014 Fifth International Conference on, pp. 161–164, June 2014.

[10] M. Rana, S. Bilgaiyan, and U. Kar, "A study on load balancing in cloud computing environment using evolutionary and swarm based algorithms," Control, Instrumentation, Communication and Computational Technologies (ICCICCT), 2014 International Conference on, pp. 245–250, July 2014.

[11] P. A. Pattanaik, S. Roy, and P. K. Pattnaik, "Performance study of some dynamic load balancing algorithms in cloud computing environment," Signal Processing and Integrated Networks (SPIN), 2015 2nd International Conference on, pp. 619–624, Feb 2015.

[12] Y. Dai, Y. Lou, and X. Lu, "A task scheduling algorithm based on genetic algorithm and ant colony optimization algorithm with multi-qos constraints in cloud computing," Intelligent Human-Machine Systems and Cybernetics (IHMSC), 2015 7th International Conference on, vol. 2, pp. 428–431, Aug 2015.

[13] S. Aslam and M. A. Shah, ”Load balancing algorithms in cloud computing: A survey of modern techniques,” 2015 National Software Engineering Conference (NSEC), pp. 30–35, Dec 2015.
[14] F. T. Novais, L. S. Batista, A. J. Rocha, and F. G. Guimares, ”A multiobjective estimation of distribution algorithm based on artificial bee colony,” Computational Intelligence and 11th Brazilian Congress on Computational Intelligence (BRICS-CCI CBIC), 2013 BRICS Congress on, pp. 415–421, Sept 2013.
[15] P. K. Sundararajan, E. Fellery, J. Forgeaty, and O. J. Mengshoel, ”A constrained genetic algorithm for rebalancing of services in cloud data centers,” 2015 IEEE 8th International Conference on Cloud Computing, pp. 653–660, June 2015.
[16] R. Buyya, R. Ranjan, and R. N. Calheiros, ”Modeling and simulation of scalable cloud computing environments and the cloudsim toolkit: Challenges and opportunities,” High Performance Computing Simulation, 2009. HPCS ’09. International Conference on, pp. 1–11, June 2009.
[17] C. Y. Liu, C. M. Zou, and P. Wu, ”A task scheduling algorithm based on genetic algorithm and ant colony optimization in cloud computing,” Distributed Computing and Applications to Business, Engineering and Science (DCABES), 2014 13th International Symposium on, pp. 68–72, Nov 2014.
[18] S. H. Z. Edwin K. P. Chong, An Introduction to optimization, 3rd edition, 3rd ed. John Wiley & Sons Inc, Feb. 2008.
[19] J. Zhao, W. Zeng, M. Liu, G. Li, J. Zhao, W. Zeng, M. Liu, and G. Li, ”Multi-objective optimization model of virtual resources scheduling under cloud computing and it’s solution,” Cloud and Service Computing (CSC), 2011 International Conference on, pp. 185–190, Dec 2011.
[20] N. Chen, X. Fang, and X. Wang, ”A cloud computing resource scheduling scheme based on estimation of distribution algorithm,” Systems and Informatics (ICSAI), 2014 2nd International Conference on, pp. 304–308, Nov 2014.
[21] Z. I. M. Yusoh and M. Tang, ”A penalty-based genetic algorithm for the composite saas placement problem in the cloud,” Evolutionary Computation (CEC), 2010 IEEE Congress on, pp. 1–8, July 2010.
[22] Z. Zheng, R. Wang, H. Zhong, and X. Zhang, ”An approach for cloud resource scheduling based on parallel genetic algorithm,” Computer Research and Development (ICCRD), 2011 3rd International Conference on, vol. 2, pp. 444–447, March 2011.
[23] F. Xie, Y. Du, and H. Tian, ”A resource allocation strategy based on particle swarm algorithm in cloud computing environment,” Digital Manufacturing and Automation (ICDMA), 2013 Fourth International Conference on, pp. 69–72, June 2013.
[24] D. Kumar and Z. Raza, ”A pso based vm resource scheduling model for cloud computing,” Computational Intelligence Communication Technology (CICT), 2015 IEEE International Conference on, pp. 213–219, Feb 2015.
[25] S. Varshney, L. Srivastava, and M. Pandit, ”Comparison of pso models for optimal placement and sizing of statcom,” Sustainable Energy and Intelligent Systems (SEISCON 2011), International Conference on, pp. 346–351, July 2011.
[26] S.-C. Wang, K. Q. Yan, W.-P. Liao, and S.-S. Wang, ”Towards a load balancing in a three-level cloud computing network,” Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on, vol. 1, pp. 108–113, July 2010.

AUTHOR BIOGRAPHY

Yu-Lun Huang received the B.S., and Ph.D. degrees in Computer Science, and Information Engineering from the National Chiao-Tung University, Taiwan in 1995, and 2001, respectively. She has been a member of Phi Tau Phi Society since 1995. She is now an associate professor in the Department of Electrical & Computer Engineering of National Chiao-Tung University (NCTU). She is now the Associate Dean of NCTU Academic Affairs, Director of Center for Continuing Education and Training at NCTU, and Director of Center for Teaching and Learning Development at NCTU. She has been serving the Secretary General of Taiwan Open Course Consortium since 2014. Her research interests include wireless security, virtualization security, embedded software, embedded operating systems, risk assessment, secure payment systems, VoIP, QoS and critical information infrastructure protection (CIIP), IoT Security, LTE Security, creative and innovative teaching model, etc.

Zong-Xian Li received the B.S., and Master degrees in Electrical and Computer Engineering from the National Chiao-Tung University, Taiwan in 2015 and 2016, respectively. His research interests include embedded operating systems, cloud computing and virtualization technologies.