A Multi Objective Optimization Model for Virtual Machine Mapping In Cloud Data Centres

Fahimeh Ramezani[1], Mohsen Naderpour[1,2], Jie Lu[1,2]
[1] Centre for Quantum Computation & Intelligent Systems
[2] Global Big Data Technologies Centre
Faculty of Engineering and Information Technology, University of Technology Sydney (UTS)
PO Box 123, Broadway NSW 2007 Australia
{Fahimeh.Ramezani, Mohsen.Naderpour, Jie.Lu}@uts.edu.au

***Abstract*—Modern cloud computing environments exploit virtualization for efficient resource management in order to reduce computational cost and energy budget. Virtual machine (VM) migration is a technique that enables flexible resource allocation and increases the computation power and communication capability within cloud data centers. VM migration helps successfully cloud providers to achieve various resource management objectives such as load balancing, power management, fault tolerance, and system maintenance. However, the process of VM migration can affect applications performance unless attended by smart optimization methods. This paper presents a multi-objective optimization model for this matter. The objectives are minimizing power consumption, maximizing resource utilization, and minimizing VM transfer time. The fuzzy particle swarm optimization (PSO), which improves the efficiency of convectional PSO by using fuzzy logic systems, is relied upon to solve the optimization problem. The model is implemented in a cloud simulator to investigate the performance. The results verify the performance of the proposed model.**

Keywords—Cloud computing, Resource management, Fuzzy particle swarm optimization.

I. INTRODUCTION

Cloud computing is a service-oriented computing paradigm that has significantly revolutionized computing by offering three web-based services – Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) [1]. These large scale services can be provided by applying shared virtualized cloud resources. In general, the cloud resources are provided as a collection of several proprietary processes in a virtual environment, called a virtual machine (VM).

Virtualized computational resources are applied in a cloud environment to provision resources on demand. Virtualization also provides the opportunity of using an auto-scaling technique that dynamically allocates computational resources to the services to match their current loads precisely, thereby removing resources that would otherwise remain idle and cost [2]. Advances in virtualization techniques and the construction of numerous large commodity data centers around the world have resulted in a new approach to computing referred to as cloud computing becoming an important topic of research and development. The recent surge in the popularity and usage of cloud computing services by both enterprise and individual consumers has necessitated the efficient and proactive management of data center resources that host services with a variety of characteristics.

One of the major issues concerning both cloud service providers and consumers is real time autonomic resource management in response to highly unpredictable demands [3]. There are two main process related to optimal autonomic resource management in cloud environments: (1) virtual resource discovery and selection to execute cloud services, and (2) physical resource allocation to virtual resources —or load balancing among PMs— to cover part of the self-configuring process.

This paper considers the load balancing issue through the use of VM migration. Load-balanced systems are desirable for several reasons, such as to avoid large discrepancies between the level of service afforded to various VMs from the same service class and to keep an even ambient temperature to reduce cooling cost. Live migration enables hot spot mitigation and server consolidation with minimal disturbance to applications running inside the migrating VMs [4]. However, an improper migration can result to the VMs being executed on unsuitable hosts, which can then lead to unwarranted effects. Therefore, an efficient scheme is required to correctly allocate the VMs to the hosts that support their execution. This decision problem is called the VM mapping problem which is a NP-Hard problem. This problem can be considered as a multi-objective optimization problem. Literature review shows that a large number of approaches have been proposed to solve the VM mapping problem. Most of these approaches employed evolutionary techniques. For instance, Gao et al. have developed a multi-objective ant colony system that tries to minimize total resource wastage and power consumption for VM placement [5]. Genetic algorithm has also been used in some models, such as [6, 7], to adaptively self-reconfigure VMs in heterogeneous cloud data centers considering minimizing total resource wastage, power consumption and thermal dissipation costs. In addition, bin-packing technique has been relied upon in several studies. For example, a mapper system has been proposed using bin-packing to tackle power-cost tradeoffs under a fixed performance constraint by minimizing migration costs while packing VMs in a small number of machines [8]. Our research shows that in addition to above mentioned objectives, network traffic is another important factor that needs to be considered in the mapping model. For example, many web applications rely on analysis of traffic data to optimize customers' experience. Network operators also need to know how traffic flows through the network in order to make many of the management and planning decisions.

This paper attempts to overcome the limitations of current studies by proposing a multi-objective optimization model that takes into account power consumption, resource utilization, and VM transfer time. New formulas are proposed for the objectives to effectively handle the cloud environment characteristics. In addition, the paper uses fuzzy swarm intelligence to solve the optimization model. Fuzzy logic, which mathematically emulates human reasoning, provides an intuitive way of designing function blocks for intelligent systems. It allows an expert to express his/her knowledge in the form of related imprecise inputs and outputs in terms of linguistic variables, which simplifies knowledge acquisition and representation, and the knowledge obtained is easy to understand and modify. Therefore, in this paper, a fuzzy logic system is utilized to tune the inertia weight in the particle swarm optimization resulting better results in comparison with typical particle swarm optimization. Furthermore, the solution can reduce the time processing for producing the optimal result compared to other evolutionary algorithms such as genetic algorithm. The performance of the proposed solution is investigated in a cloud simulation environment, and the results are compared to that of a bi-objective model.

The paper is organized as follows: Section II describes the related background to this research. Section III presents the problem statement, and it is followed by proposed algorithm in Section IV. Section V shows the evaluation part, and Section VI summarizes the conclusion and future work.

II. Background

A. *Virtualization Architecture*

The informal interpretation of virtualization is that of a mechanism for running concurrently several operating system (OS) instances on a single computer node. These nodes are called physical machines (PMs). A typical virtualization architecture is shown in Fig. 1. The hypervisor is the core component of a virtualization platform. The main responsibility of the hypervisor is to delegate computer hardware to virtual-machine monitors (VMMs). Each VMM is responsible for providing hardware abstraction for exactly one running VM. The VM typically hosts a guest OS. By running multiple VMs simultaneously on a PM, the hardware can be used more efficiently [4].

B. *Virtual Machine Migration*

There are three different approaches to migrate a VM. When cold migration is used, the guest OS is shut down, the VM is moved to another host and then the guest OS is restarted there. Hot migration suspends the guest OS instead of shutting it down. The guest OS is resumed after the VM is moved to the destination host. The benefit of hot migration is that applications running inside the guest OS are not restarted from scratch. Some platforms offer a feature called live migration, as presented in Fig.2.

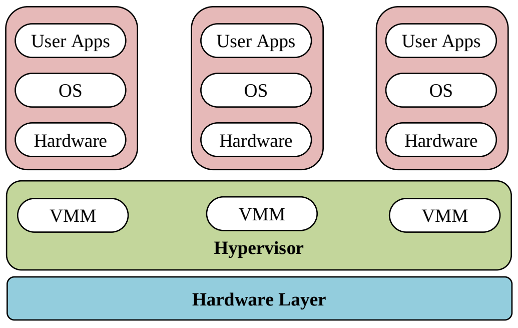


Fig. 1. Virtualization architecture.

This feature allows a VM to be moved from one host to another while the guest OS is running. Live migration reduces the downtime dramatically for applications executing inside the VM. It is highly valuable if the migrations can be performed automatically, without the involvement of a human operator [4]. There are also several VM live migration techniques that consider power consumption reduction as well as downtime and migration time. Liao et al. [9] developed a live VM mapping framework to map VMs onto a set of PMs without significant system performance degradation while reducing power consumption. Sallam and Li [10] also suggested a multi-objective VM migration technique that considers power and memory consumption, thus making live VM migration more beneficial for cloud providers. Lin et al. [11] believed that the load balancing strategies that focus on VM migration for optimizing on-demand resource provisioning needed to be improved. They proposed a threshold-based dynamic resource allocation approach for load balancing that dynamically allocates the VMs among the cloud's applications based on their load changes. Atif and Strazdins [12] also developed a similar cloud utilization optimization framework for Application as a Service. They used VM monitor facilities (which have traditionally been used for live migration) to create sets of homogenous clusters of computing frame (VMs). They used these clusters to schedule or migrate application tasks over a set of homogenous VMs based on estimated task execution time to optimize resource utilization and enhance application performance. However, this method cannot be used when a determined homogenous cluster has high utilization and is in an overloaded state.

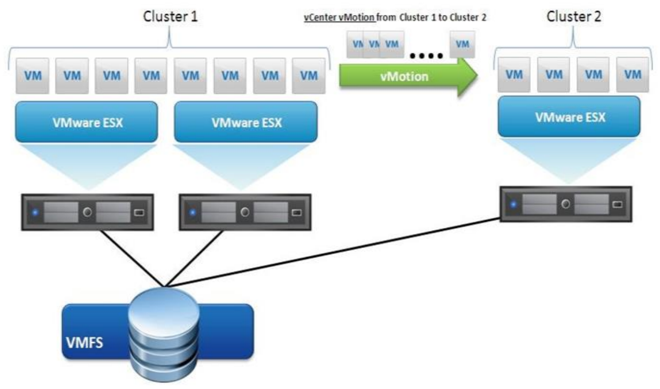


Fig. 2. VM live migration.

C. *Fuzzy Logic Systems*

Fuzzy logic is a concept for dealing with uncertainty, vagueness, or imprecise problems that uses membership functions with values between 0 and 1. Unlike conventional set theory based on Boolean logic, a particular object or variable in fuzzy set theory based on fuzzy logic has a degree of membership in a given set that may be anywhere in the range of 0 (completely not in the set) to 1 (completely in the set).

A fuzzy logic system includes three parts: fuzzification, fuzzy inference engine and defuzzification. In the fuzzification process, the fuzzy sets are formed for all input variables. The fuzzy inference engine takes into account the input variables and the logic relations/rules between them, and uses fuzzy logic operations to generate the output. In the defuzzification process, the output fuzzy set is converted into a crisp value.

A rule contains information obtained from a human expert, and represents that information in the form of IF–THEN. The rule can then be used to perform operations on data to inference in order to reach appropriate conclusion. These inferences are essentially a computer program that provides a methodology for reasoning about information in the rule base or knowledge base, and for formulating conclusions [13].

There are several inference methods, however, Mamdani [14] and Takagi and Sugeno [15] methods are most commonly used in industrial and fuzzy software tools. The characteristic of Mamdani's model also known as the Max-Min fuzzy rule based inference are presented in Table I.

TABLE I. CHARACTERISTICS OF MAMDANI'S MODEL.

Operation	Operator	Formula
Union (OR)	MAX	$\mu_C(x) = \max(\mu_A(x), \mu_B(x)) = \mu_A(x) \vee \mu_B(x)$
Intersection (AND)	MIN	$\mu_C(x) = \min(\mu_A(x), \mu_B(x)) = \mu_A(x) \wedge \mu_B(x)$
Implication	MIN	$\min(\mu_A(x), \mu_B(x))$
Aggregation	MAX	$\max(\min(\mu_A(x), \mu_B(x)))$
Defuzzification	CENTROID	$\mathrm{COA} = Z^* = \frac{\int z\mu_C(z)\,dz}{\int \mu_C(z)\,dz}$

Note:
$\mu_C(x)$ = value of the resultant membership function.
$\mu_A(x)$ = value of the membership function of fuzzy set A.
z = abscissa value, ($\mu_C(z)$ is the ordinate).

D. *Fuzzy Swarm Intelligence*

Particle swarm optimization (PSO) is a population-based search algorithm based on the simulation of the social behavior of birds which was originally proposed by Kennedy and Eberhart [16]. Although originally adopted for balancing weights in neural networks, PSO soon became a very popular global optimizer, mainly in problems in which the decision variables are real numbers. In PSO, particles are flown through hyper-dimensional search space. Changes to the position of the particles within the search space are based on the social–psychological tendency of individuals to emulate the success of other individuals. The position of each particle is changed according to its own experience and that of its neighbors. Let $\vec{X}_i(t)$ denote the position of particle *i*, at iteration *t*. The position of $\vec{X}_i(t)$ is changed by adding a velocity $\vec{V}_i(t+1)$ to it, i.e.:

$$\vec{X}_i(t+1) = \vec{X}_i(t) + \vec{V}_i(t+1) \qquad (1)$$

The velocity vector reflects the socially exchanged information and, in general, is defined in the following way:

$$\vec{V}_i(t+1) = W\vec{V}_i(t) + C_1 r_1 \left(\vec{x}_{pbest_i} - \vec{X}_i(t)\right) + C_2 r_2 \left(\vec{x}_{gbest_i} - \vec{X}_i(t)\right) \qquad (2)$$

where C_1 is the cognitive learning factor and represents the attraction that a particle has towards its own success; C_2 is the social learning factor and represents the attraction that a particle has towards the success of the entire swarm; W is the inertia weight, which is employed to control the impact of the previous history of velocities on the current velocity of a given particle; $\vec{x}_{pbest_i}$ is the personal best position of the particle *i*; $\vec{x}_{gbest}$ is the position of the best particle of the entire swarm; and $r_1, r_2 \in [0,1]$ are random values [17].

In PSO, the search process is a nonlinear and dynamic procedure. Therefore, when the environment is dynamically changing over the time, the optimization algorithm needs to adapt dynamically itself to the changing environment. The change of the particle's situation is directly correlated to the inertia weight. Proper choice of the inertia weight *W* provides a balance between global and local optimum points. Several methods have been applied to handle the inertia weight during the progression of the optimization process. Constant inertia weight, linearly decreasing inertia weight and random inertia weight are some examples [18]. In this paper, a FLS is used to adaptively control the inertia weight of PSO.

III. PROBLEM STATEMENT

The VM mapping is a multi-objective optimization problem. The aim is to find an optimal placement, which is a mapping VMs over PMs, Fig.3 shows an example. Consider a system with "m" host machines and "n" VMs. Each VM is represented as *VM$_i$* and each host is represented as *PM$_j$*. A single decision variable for the problem is denoted by *y$_{ij}$*. The value of this decision variable is 1 when the *i*th VM is allocated to the *j*th PM and 0 otherwise.

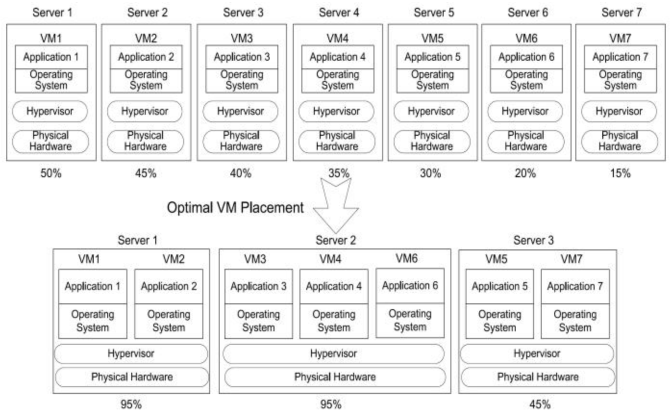


Fig. 3. An example of VM mapping [5].

In this problem, each PM can be represented by a vector like *PM_j* = (*ID_j*, *CPU_j*, *MEM_j*, *BW_j*) *j*=1,…,*m* where ID provides an identification number, CPU_j gives the processing power, MEM_j gives the amount of memory, and BW_j is the amount of bandwidth for *j*th PM. Likewise, each VM can be represented by a vector $VM_i = (ID_i, CPU_i, MEM_i, SIZE_i)$ $i=1,\dots,n$ where ID gives the identification of the VM, CPU_i gives the processing power required by the VM, MEM_i gives the amount of memory requested by the VM, and $SIZE_i$ represents the VM's size. To properly define the problem, the objectives are firstly introduced.

A. Power Consumption

It has been proved that an idle server consumes around 70% of the power consumed by a fully utilized server [19, 20]. Therefore, having fewer active PMs leads to lower power consumption in a cluster. Considering this fact, the ratio of active PMs to all available PMs is minimized to reduce power consumption:

$$PowerCunsumption = \frac{N_{aPM}}{m} \tag{3}$$

where N_{aPM} is the number of active PMs:

$$N_{aPM} = \sum_{j=1}^{m} \min\left(\sum_{i=1}^{n} y_{ij}, 1\right) \tag{4}$$

B. Resource Utilization

The idle resources available on each PM may vary largely with different VM placement solutions. In anticipation of future requests, the resources left on each PM should be balanced along different dimensions. Otherwise, unbalanced residual resources may prevent any further VM placement, thus wasting computing resources. For example, Fig. 4 shows that the PM has a lot of unused CPU capacity but little memory available, causing the PM to not be able to accept any new VM because of memory scarcity [7]. Therefore, to balance the resource usage along different dimensions, the following equations are proposed:

$$IdleMemory = \sum_{j=1}^{m} \left(PM_j^{A-MEM} / \left(\sum_{i=1}^{n} y_{ij} * VM_i^{MEM} \right) \right) \tag{5}$$

where PM_j^{A-MEM} is the amount of available memory (GB) on PM_j.

$$IdleCPUs = \sum_{j=1}^{m} \left(PM_j^{A-CPU} / \left(\sum_{i=1}^{n} y_{ij} * VM_i^{CPU} \right) \right) \tag{6}$$

where PM_j^{A-CPU} is the number of available CPUs on PM_j.

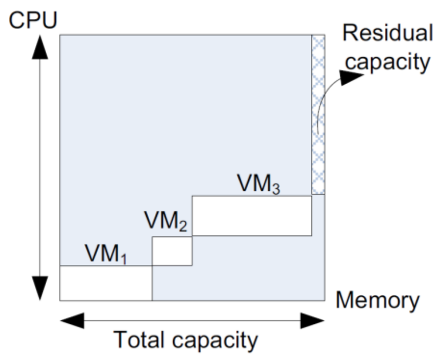


Fig. 4. Resources allocated to three VMs on a single PM [7].

C. VM Transfer Time

The total VM transfer time is the time it takes from when the migration is initiated for a VM on the source PM until the VM is resumed on the destination host. In order to be able to provide the capability of live migration, a low downtime is required. Downtime is the time during which the virtual machine is not responsive. Suspending the state of the VM includes pausing the CPUs as well as other connected devices.

$$Transfer\ Time = \sum_{j=1}^{m} \sum_{i=1}^{n} y_{ij} * \frac{VM_i^{SIZE}}{PM_j^{BW}} \tag{7}$$

where VM_i^{SIZE} is in GB.

D. Problem

The problem is defined as finding a pattern to map the set of VMs over the set of PMs such that the cluster power consumption is minimized, the physical resource utilization is maximized, and VM transfer time is minimized as follows:

$min\ f_{PowerCunsumption}$

$min\ f_{IdelMemory}$

$min\ f_{IdelCPUs}$

$min f_{TransferTime}$

Subject to

$$\sum_{j=1}^{m} y_{ij} = 1, \forall i = 1, \dots, n$$

$$y_{ij} \in \{0,1\}, \forall i = 1, .., n \ \&\ j = 1, \dots, m$$

$$0 < N_{aPM} \le m$$

$$PM_j^{A-MEM} \le \left(\sum_{i=1}^{n} y_{ij} * VM_i^{MEM} \right)$$

$$PM_j^{A-CPU} \le \left(\sum_{i=1}^{n} y_{ij} * VM_i^{CPU} \right)$$

The constraints ensure that each VM is allocated to only one PM, however; one PM may have more than one VM, and the load on each PM is not greater than its capacity.

IV. Heuristic Algorithm

The proposed algorithm with the use of fuzzy PSO (FPSO) is presented in the following.

A. Fuzzy Particle Swarm Optimization

A normalized fitness value (NFV), which is between 0 and 1, is defined as:

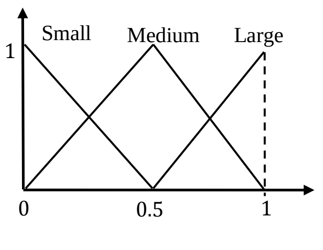
$$NFV = \frac{FV - FV_{min}}{FV_{max} - FV_{min}} \tag{8}$$

where FV_{max} is the worst solution for the minimization process. For the first iteration, the calculated value of FV may be selected as FV_{min} for the next iterations [18].

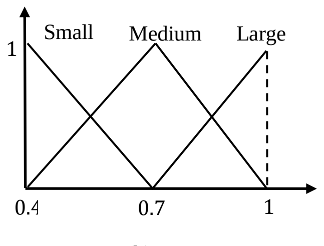
It is worth noting that for fuzzification process, triangular membership functions are lied upon for sake of simplicity. The membership function of inputs is presented in three linguistic levels Small, Medium, and Large, and the membership function of output is described as Negative, Zero and Positive as illustrated in Fig. 5. The fuzzy rules as shown in Table II are derived by an expert to achieve optimum results. The fuzzy rules are used to select the inertia weight correction (ΔW). In addition, Mamdani' fuzzy inference method is used to evaluate the results.

TABLE II. RULES FOR INERTIA WEIGHT VARIATIONS.

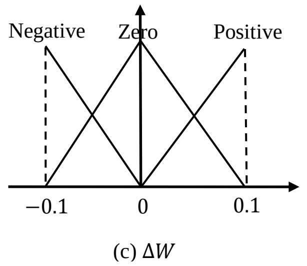
	NFV		W		ΔW
IF	Small	AND	Small	THEN	Zero
IF	Small	AND	Medium	THEN	Negative
IF	Small	AND	Large	THEN	Negative
IF	Medium	AND	Small	THEN	Positive
IF	Medium	AND	Medium	THEN	Zero
IF	Medium	AND	Large	THEN	Negative
IF	Large	AND	Small	THEN	Positive
IF	Large	AND	Medium	THEN	Zero
IF	Large	AND	Large	THEN	Negative



(a) NFV



(b) W



(c) ΔW

Fig. 5. Membership functions of inputs and output.

B. *Proposed Algorithm*

In summary, the following steps should be conducted by VM mapping algorithm:

VM mapping algorithm

```
1. Gather data and information about VMMs, VMs, and PMs as input data.
2. Determine a set of PMs as a new host for VMs.
3. Determine the set of VMs which need to be mapped.
4. Apply the FPSO method:
    4.1. Create an initial population array of every particle $i$ ($\vec{X}_i$) with random positions and velocities on $n$ dimensions in the search space.
    4.2. Convert continuous position values vector of $\vec{X}_i$ to discrete vector $d(\vec{X}_i)$ to determine optimal pattern for mapping VMs over PMs.
    4.3. Determine the value of $VM_i^{MEM}, VM_i^{CPU}, VM_i^{size}, PM_j^{A-MEM}, PM_j^{A-CPU}$, and $PM_j^{bw}$ based on $d(\vec{X}_i)$ to calculate the value of every fitness function.
    4.4. For each particle, calculate the values of objective functions by applying Equations 3, 5, 6, and 7.
    4.5. For each particle, evaluate the desired optimization fitness function.
    4.6. Compare each particle's fitness evaluation with its personal best fitness function value ($\vec{x}_{pbest_i}$). If the current value is better than $\vec{x}_{pbest_i}$, then set $\vec{x}_{pbest_i}$ equal to the current value, and the best position $p_i$ equal to the current location $\vec{x}_i$ in $n$-dimensional space.
    4.7. Identify the particle in the neighborhood with the best global success so far as $\vec{x}_{gbest_i}$, and assign its index to the variable $g$ as the best global position.
    4.8. Update the parameters, the proper choice of inertia weight is updated by the FLS.
    4.9. Change the velocity and position of the particle according to Equations 1 and 2.
    4.10. If a criterion is met (usually a sufficiently good fitness or a maximum number of iterations) then best particle position in $n$-dimensional space $d(\vec{X}_{gbest})$ is the optimal VM mapping schema.
    Else go to 4.2
5. Update the current PMs and VMs properties.
```

V. EXPERIMENTAL EVALUATION

To evaluate the proposed model, a cloud simulation environment called CloudSim toolkit [21] is utilized. It is an open source tool used by majority of the researchers to simulate the cloud environment. CloudSim allows the extension and definition of policies in all components of the software stack.

A. *Experiment setup*

The experimental setup is composed of five PMs that need to run ten VMs in total. The properties of PMs and VMs are summarized in Tables III and IV respectively.

TABLE III. PM PROPERTIES.

PM ID	Number of cores	Memory (GB)	Bandwidth
1	8	16	1024
2	8	8	512
3	6	8	1024
4	6	4	512
5	4	2	512

TABLE IV. VM PROPERTIES.

VM ID	VM image size (GB)	VM memory (GB)	Number of CPUs	VMM
1	20	8	4	Xen
2	20	4	2	Xen
3	20	4	2	Xen
4	10	2	2	Xen
5	10	2	1	Xen
6	8	2	1	Xen
7	5	1	2	Xen
8	5	1	1	Xen
9	4	1	1	Xen
10	4	1	2	Xen

B. *Results Analysis*

The simulation under the environment that was defined in previous section is performed to evaluate the proposed multi-objective method of solving VM mapping problem with conflicting objectives by considering optimization transfer time, idle memory, idle CPUs and power consumption. The efficiency of the proposed four-objective model is evaluated in comparison with current bi-objective model in terms of optimizing cloud utilization and power consumption.

Fuzzy PSO is applied to optimize both the four-objective model and bi-objective model. The graph for the four objectives in 300 out of 2000 iterations is illustrated in Fig. 6. The values of the axis on the right show the range of of $f_{TransferTime}$.

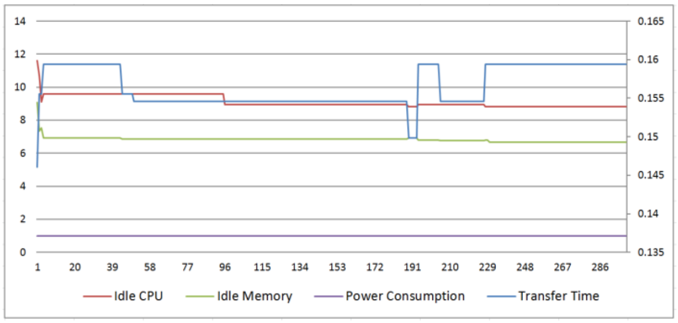


Fig. 6. The value of objective functions.

As can be seen from Fig. 6, objective functions for idle CPUs and idle memory are in conflict with each other and there is not unique solution for them. Therefore, as it was discussed before, considering them in one function would not be correct.

The output VM mapping patterns resulted from Cloudsim using FPSO for the four- and bi-objective models are summarized in Table 5. For instance, the best particle position suggested by the optimal solution of the four-objective model is $d(\vec{X}_i) = (d_1, d_2, \dots, d_{10})$ = (1, 3, 2, 2, 3, 4, 4, 5, 4, 5) after converting the continuous position values to discrete. According to this solution, VMs: vm_1, vm_2, vm_3, vm_4,…,vm_{10} are mapped to PMs as shown in first row of Table 5. The second row illustrates suggested solution for the bi-objective model.

TABLE V. VM MAPPING PATTERNS.

VMs	1	2	3	4	5	6	7	8	9	10
4Objs	PM_1	PM_3	PM_2	PM_2	PM_3	PM_4	PM_4	PM_5	PM_4	PM_5
2Objs	PM_1	PM_4	PM_3	PM_5	PM_2	PM_2	PM_3	PM_2	PM_1	PM_2

In the bi-objective model, the optimal value of $f_{IdleCPUandMemory}$ and $f_{PowerConsumption}$ is determined by FPSO, then the corresponding value of $f_{IdleCPU}$, $f_{IdleMemory}$ and $f_{TransferTime}$ are calculated by applying the pattern of VMs mapping over PMs that resulted from the bi-objective optimal solution. The algorithm has been run 25 times for both models, and the results are almost the same. The optimal results of the models are summarized in Table 6.

TABLE VI. COMPARISON RESULTS.

Model	Transfer Time (s)	Idle Memory ratio	Idle CPUs ratio	Power Consumption ratio	Idle Memory and CPUs ratio	Iteration	Number of Idle PMs
Four-objectives	0.15	6.6	8.8	1	--	231	0
Bi-objectives	0.16	6.7	9.7	1	16.4	251	0

As can be seen from Table 5, the estimated values for transfer time, the number of idle CPUs and the amount of idle memory obtained from the proposed model are less than these values achieved by bi-objective model. However, they have used all PMs in the set as host for VMs resulting the same amount of power consumption.

Based on this, the model achieves the highest resource utilization because the less number of CPUs and amount of memory are idle after VM mapping. In addition, this model decreases the bandwidth traffic because the same amount of data (VMs' size) is transferred in lower time where a micro second is important. This means, the model selects the optimal path to transfer VMs to their new host.

As a result, the proposed model that considers more aspects of VM mapping optimization, determines an optimal trade-off solution for the multi-objective problem with objective functions that are in conflict with one another, and results the best possible compromise between objectives.

VI. Conclusion and Future Work

Virtualization technology is used to increase resource utilization and reduce operating costs in cloud data center. Two key mechanisms for flexible resource utilization offered by virtualization are: a) allocating resources dynamically to virtual machines (i.e., changing CPU share or memory allocation), and b) migrating VMs to other PM. In this paper, the VM mapping is formulated as a multi-objective combinatorial optimization problem aiming to simultaneously optimize possibly conflicting objectives. The objectives include making efficient usage of multidimensional resources, reducing energy consumption, and reducing network traffic. The paper uses a fuzzy particle swarm optimization algorithm including a fuzzy logic system for tuning inertia weight, and the conventional particle swarm optimization to solve the mapping problem. Computational experiments on benchmark problems are carried out. The results show that the proposed algorithm can compete efficiently with other promising approaches to the problem.

As interdependencies and inter-communication patterns among VMs are overlooked in this paper, the future research direction is to consider the complete application context running on top of the VM when choosing the most appropriate PM to host that VM. It takes into account the communication dependencies among VMs of a multi-tier enterprise application, the underlying data center network topology, as well as the capacity limits of the PMS of the data center.

Acknowledgment

The work presented in this paper was supported by the Australian Research Council (ARC) under Discovery Project DP140101366.

References

[1] F. Ramezani, J. Lu, J. Taheri, and F. Hussain, "Evolutionary algorithm-based multi-objective task scheduling optimization model in cloud environments," *World Wide Web,* vol. 18, pp. 1737-1757, 2015/03/10 2015.

[2] B. Dougherty, J. White, and D. C. Schmidt, "Model-driven auto-scaling of green cloud computing infrastructure," *Future Generation Computer Systems,* vol. 28, pp. 371-378, 2012.

[3] F. Ramezani, M. Naderpour, and L. Jie, "Handling uncertainty in cloud resource management using fuzzy Bayesian networks," presented at the IEEE International Conference on Fuzzy Systems (FUZZ-IEEE) Istanbul, Turkey, 2015.

[4] M. Forsman, A. Glad, L. Lundberg, and D. Ilie, "Algorithms for automated live migration of virtual machines," *Journal of Systems and Software,* vol. 101, pp. 110-126, 3// 2015.

[5] Y. Gao, H. Guan, Z. Qi, Y. Hou, and L. Liu, "A multi-objective ant colony system algorithm for virtual machine placement in cloud computing," *Journal of Computer and System Sciences,* vol. 79, pp. 1230-1242, 12// 2013.

[6] M. Haibo, W. Huaimin, Y. Gang, Z. Yangfan, S. Dianxi, and Y. Lin, "Online Self-Reconfiguration with Performance Guarantee for Energy-Efficient Large-Scale Cloud Computing Data Centers," presented at the IEEE International Conference on Services Computing (SCC), Miami, USA, 2010.

[7] J. Xu and J. A. B. Fortes, "Multi-Objective Virtual Machine Placement in Virtualized Data Center Environments," presented at the IEEE/ACM International Conference on Green Computing and Communications (GreenCom), Hangzhou, China, 2010.

[8] A. Verma, P. Ahuja, and A. Neogi, "pMapper: Power and Migration Cost Aware Application Placement in Virtualized Systems," presented at the Middleware 2008: ACM/IFIP/USENIX 9th International Middleware Conference Leuven, Belgium,, 2008.

[9] X. Liao, H. Jin, and H. Liu, "Towards a green cluster through dynamic remapping of virtual machines," *Future Generation Computer Systems,* vol. 28, pp. 469-477, 2012.

[10] A. Sallam and K. Li, "A multi-objective virtual machine migration policy in cloud systems," *The Computer Journal,* vol. 57, pp. 195-204, 2014.

[11] W. Lin, J. Z. Wang, C. Liang, and D. Qi, "A threshold-based dynamic resource allocation scheme for cloud computing," *Procedia Engineering,* vol. 23, pp. 695 – 703, 2011.

[12] M. Atif and P. Strazdins, "Adaptive parallel application resource remapping through the live migration of virtual machines," *Future Generation Computer Systems,* vol. 37, pp. 148-161, 2014.

[13] M. Naderpour, J. Lu, and G. Zhang, "An intelligent situation awareness support system for safety-critical environments," *Decision Support Systems,* vol. 59, pp. 325-340, 2014.

[14] E. H. Mamdani, "Application of fuzzy logic to approximate reasoning using linguistic synthesis," *IEEE Transactions on Computers,* vol. C-26, pp. 1182-1191, 1977.

[15] M. Sugeno, "An introductory survey of fuzzy control," *Information sciences,* vol. 36, pp. 59-83, 1985.

[16] J. Kennedy and R. Eberhart, "Particle swarm optimization," presented at the Proceedings of the IEEE International Conference on Neural Networks 1995.

[17] F. Ramezani, J. Lu, and F. K. Hussain, "Task-Based System Load Balancing in Cloud Computing Using Particle Swarm Optimization," *International Journal of Parallel Programming,* vol. 42, pp. 739-754, 2013.

[18] M. Nafar, G. B. Gharehpetian, and T. Niknam, "Using modified fuzzy particle swarm optimization algorithm for parameter estimation of surge arresters models," *Int J Innov Comput Inf Control,* vol. 8, pp. 567-582, 2012.

[19] R. Buyya, A. Beloglazov, and J. Abawajy, "Energy-efficient management of data center resources for cloud computing: A vision, architectural elements, and open challenges," *arXiv preprint arXiv:1006.0308,* 2010.

[20] B. Priya, E. S. Pilli, and R. C. Joshi, "A survey on energy and power consumption models for Greener Cloud," in *IEEE 3rd International Conference on Advanced Computing (IACC)*, 2013, pp. 76-82.

[21] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and Experience,* vol. 41, pp. 23-50, 2011.