# Evolving of Cellular Automata for Task Scheduling in Cloud Computing Systems

Ingkwan Vashirashudej
Department of Computer Engineering,
Faculty of Engineering
King Mongkut's Institute of Technology Ladkrabang
Bangkok, Thailand
s5611513@kmitl.ac.th

Sakchai Thipchaksurat
Department of Computer Engineering,
Faculty of Engineering
King Mongkut's Institute of Technology Ladkrabang
Bangkok, Thailand
ktsakcha@kmitl.ac.th

*Abstract*— **Task scheduling is one of the major challenging issue in cloud computing system, which aims to assign tasks to virtual machines (VMs) and minimizes the total execution time of tasks. In this paper, we propose the evolving of cellular automata to the task scheduling (CATS) in cloud computing system. In CATS scheme, the cellular automata (CA) is used for setting the task scheduling rules. The effective rule can be discovered from genetic algorithm. We evaluate the effectiveness of CATS scheme by means of the simulation. The performance metric is compared in the terms of makespan with the traditional First-Come-First-Served (FCFS) scheme. The simulation result shows that the CATS scheme can provide the lower makespan than that of FCFS scheme.**

***Keywords-task scheduling; cloud computing systems; virtual machines; cellular automata; genetic algorithms;***

## I. INTRODUCTION

Cloud computing system is now very popular and promising system in information technology (IT) that moves computing and data away from computers into a large data centers. It refers to applications which are delivered as services over the Internet and the actual cloud infrastructure. Cloud-service clients can be done some functions by themselves such as adding more capacity at peak demand, experiment with new services, and removing unneeded capacity [1]. The benefits of cloud computing is that the cost for deploying new business or technology capabilities and increasing the economic efficiency by developing the expensive infrastructure can be reduced. Cloud computing deployment model comes in three forms: public clouds, private clouds, and hybrid clouds. A public cloud is one in which the services and infrastructure are provided off-site over the Internet whereas the services and infrastructure of a private cloud are maintained in a private network and it is designed for restricted access to a single enterprise (or extended enterprise). Therefore, it provides a higher level of security and control. A hybrid cloud is a composition of two or more private and public clouds to perform distinct functions within the same organization. According to the abstraction level of the capability provided and the service model of providers, the services of cloud computing system can be divided into three classes such as the Infrastructure-as-a-Service (IaaS), the Platform-as-a-Service (Paas), and the Software-as-a-Service (Saas) [2].

The characteristics of cloud computing are the virtualization, distribution and dynamically scalability. Virtualization is the main characteristic, with the support of virtualization technology in cloud computing, a cloud is built up of numerous physical machines which can run multiple virtual machines (VMs) at the same time. This is presented to the application layer of the system e.g. as Amazon Elastic Compute Cloud (EC2) [3]. There are several issues in cloud computing and task scheduling is one of the major issues. In cloud computing platforms, the traditional task scheduling algorithm is First-Come-First-Served (FCFS) scheme but it is not optimal and cannot utilize resources in parallel.

However, it is very difficult to manually assign tasks to computing resources in clouds. Therefore, we need an efficient algorithm for task scheduling in order to match suitable resources for tasks better. The scheduling problem can be classified as a NP-hard optimization problem.

We have studied the several task scheduling techniques. We find that cellular automata (CA) can be applied for task scheduling. The authors in [5] and [6] applied CA for multiprocessor scheduling. However, CA has not been applied for task scheduling in cloud computing system. This encourages us to apply CA for task scheduling in cloud computing.

In this paper, we present the evolving of cellular automata for task scheduling in cloud computing system. Our proposed is calls CATS scheme. In CATS scheme, we use cellular automata for setting the task scheduling rules. Then, the genetic algorithm is used for discovering the effective task scheduling rules. The main objective of this paper is to minimize the total execution time.

The rest of this paper is organized as following: Section Ⅱ reviews related works, and then in Section Ⅲ introduces the CloudSim toolkit. Section Ⅳ presents our proposed scheduling algorithm in details and how it works. Section Ⅴ presents the simulation results. Finally, Section Ⅵ concludes this paper.

## II. Related Works

The authors in [4] and [5] introduced a cellular automata-based scheduler to find an allocation which minimizes the total makespan of parallel tasks in the two-processor system. Cellular automata (CA) are interesting because complex global behavior arises from simple local interactions and CA can be applied for solving the scheduling problem according to some scheduling rules which must be found. Effective rules for CA are discovered by a genetic algorithm (GA).

CA is a discrete-time system that can be used for modeling several phenomena in the physical systems or natural systems. It is formed by a collection of cells also called lattice and each cell has its own state. The number of possible states in a cell depends on the value of *k*. For example, if *k* is equal to 2, the possible value for each cell can has the value of 0 or 1. The state of the cell *i* at time *t*+1 depends on the states of its neighborhood with radius *r* and itself at time *t*. Over time, the cells can change from state to state and the CA's rules will determine how the states change [6] [7].

The simplest CA is the one-dimensional binary cellular automata which the state of each cell *i* in a lattice can be either 0 or 1 ($k$ = 2). For example, assuming that $r$ = 1 with the transition rule 30 (00011110). There are two neighborhoods for each cell and it can be expressed in 8 ($2^3$) different states [8]. All possible neighborhood states, transition function, and the evolution of CA are illustrated in Fig.1.

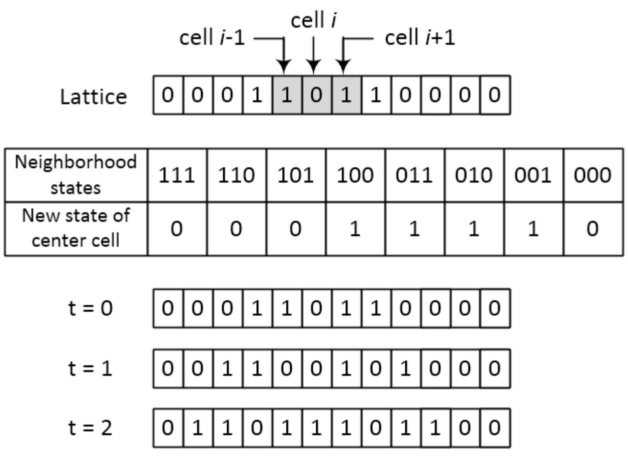

Figure 1. Example of rule 30 of one-dimensional binary cellular automata with r = 1

In cloud computing systems, the traditional scheme for task scheduling is First-Come-First Served (FCFS). FCFS scheme provides a simple process scheduling algorithm that saves valuable CPU resources. It uses nonpreemptive scheduling which a task is automatically queued and processed according to the arrival of task without other biases or preferences. But FCFS scheme is not an optimal solution for task scheduling. The authors in [4] and [5] used CA for task scheduling in two-processor system since the state of a cell of CA can represent as an allocation of a task to the specific processor and used GA for discovering the effective rules of CA which give the optimal or suboptimal makespan for task scheduling. This CA-based scheduler is divided into two phases: learning phase and operation phase. In learning phase, GA is applied to discover optimal rules of CA suitable for solving instances of a scheduling problem. In operation phase, discovered rules of CA are able to find an optimal solution of the scheduling problem for any initial allocation of a task set in two-processor system. Since there is no implementation of CA for task scheduling in cloud computing systems, this motivates us to apply CA in our work.

However, there are a large number of VMs in cloud computing systems and consequently, it can cause substantial complexity gain from the length of the rule of CA with increasing number of processor *k* and neighborhood radius *r*. When considering scheduling problem in the case of more than two processors, the length of rule grows rapidly with *k* and *r*. It is calculated as $k^{2r+1}$ as shown in Table I with different *k* when $r$ = 1 so it is difficult for GA to search in large spaces. This problem also becomes a subject of our work to make it works efficiency in cloud computing systems.

TABLE I. Lengths of Rules for Different *K*

| ***k*** | **The length of a rule** |
|---|---|
| 2 | 8 |
| 3 | 27 |
| 4 | 64 |
| 5 | 125 |
| 10 | 1000 |
| 20 | 8000 |
| 30 | 27000 |
| 50 | 125000 |

## III. CloudSim Toolkit

### A. *Characteristic and Modeling*

CloudSim is a toolkit for simulation and experimentation of cloud computing scenarios and it also supports modeling of large scale cloud computing infrastructures. CloudSim provides basic classes for describing data centers, virtual machines, service brokers, applications, users, computational resources, and policies for management (e.g., scheduling and allocation) [9].

The Datacenter is the core hardware infrastructure services related to the clouds. It is composed of a set of hosts, which are responsible for managing VMs by assigning a pre-configured processing capability, memory, storage, and a provisioning policy for allocating processing cores to VMs. A host represents a physical computing node which has one or more VMs inside. There are two steps in allocation of cloud resources: VM provisioning and Application provisioning. The allocation policy can be managed by assigning specific CPU cores to specific VMs called space-shared policy or to dynamically distribute the capacity of a core among VMs called time-shared policy, and to assign cores to VMs on demand, or to specify other policies [10] [11].

## *B. Design and Implementation of CloudSim*

CloudSim does not satisfy all the requests. CloudSim users are required to extend the specific classes (or entities) at user code level [12]. The main classes in CloudSim related to our work are listed as follows:

*1) Cloud Information Service (CIS) class*: The CIS is the core of simulate scheduling since it maps user or broker requests to suitable cloud providers. It also provides resource registration such as datacenter and broker.

*2) Datacenter class:* It is composed of a set of hosts which is resposible for managing VMs. Datacenter behaves like an IaaS provider by receiving requests for VMs from brokers and creating the VMs in hosts.

*3) DatacenterBroker class:* This class represents a broker acting on behalf of a user. It modifies two machanisms: *the* machanism for submitting VM provisioning requests to data centers and the mechanism for submitting the tasks to VMs. The CloudSim users have to extend this class for conducting experiments with their own policies.

*4) VirtualMachine class:* It represents a software implementation of a machine that executes applications called virtual machine (VM) which works like a physical machine. Each virtual machine divides the resources received from the host among tasks running on it.

*5) Cloudlet class:* A cloudlet class is also known as a task. CloudSim represents the complexity of an application in terms of its computational requirements. This class is managed by the scheduling policy which is implemented in DatacenterBroker Class.

To develop and evaluate our scheduling scheme, we extend and implement CATS scheme in DatacenterBroker class since it is responsible for managing task scheduling and resource allocation in cloud computing systems. The CloudSim simulation data flow is shown in Fig.2. When the simulation starts, each datacenter registers itself with the CIS and handle requests from a broker, all VMs are created afterwards. When a new task set arrives into the system, DatacenterBroker sends requests to all datacenters for scheduling the arrival tasks as the scheduling policy implemented in DatacenterBroker [13].

## IV. Our Proposed Scheme

In this section, we describe our task scheduling scheme called the evolving of cellular automata for task scheduling in cloud computing systems or CATS scheme. To implement it for a large number of virtual machines (VMs) in cloud computing, we design our scheduler working with multiple sets of VMs and it divides a task set and distribute to the sets of VMs.

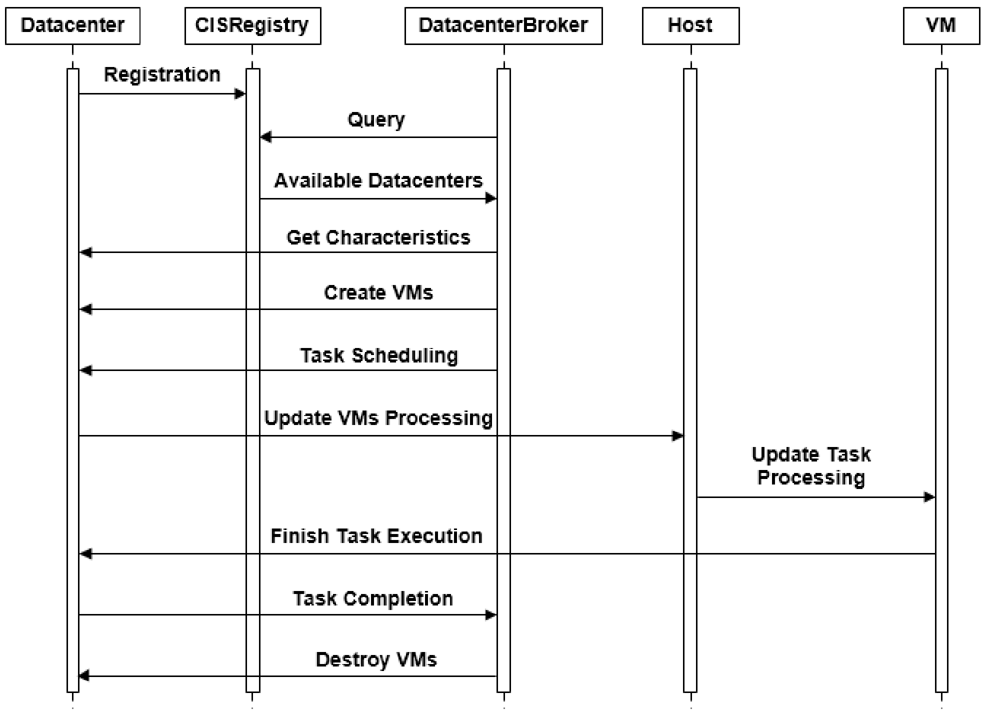

Figure 2. The flow diagram of communication among core entities in CloudSim

For example, there are 10 VMs with a task set of 200 tasks, all VMs are divided into 10 sets with two VMs in each set and the number of tasks in each VM set is 20 tasks. The scheduler has to work for 10 times to find the optimal allocation of a task set.

In addition, we assume that a nonlinear structure of a task set is approximated by a one-dimensional CA which each cell has two possible states 0 or 1. Our automaton is a binary one ($k$ = 2). The state 0 or 1 of a cell means that a corresponding task is allocated either in the VM0 or VM1 in a VM set. With each task of a task set, an elementary cell is associated. The CA has null boundary conditions. We assume that "absent cells" are always in state 0.

The CA corresponding to a task set evolves according to its rule. Initial states of CA is an initial allocation of tasks for two VMs. Changing states of CA results in changing an allocation of tasks in the system. Therefore, we use GA to discover the best rule of CA that provides the minimize makespan $T$.

CAT scheme can be divided into two phases: the learning phase and the operation phase as shown in Fig. 3. The purpose of the learning phase is to discover rules for task scheduling by using GA. The GA begins with randomly generating a population of rules with size $P$ and randomly choosing $I$ initial allocations which correspond to initial allocations of a task set. CA rules will run on each initial allocation for $M$ time steps then measure the makespan $T$ for each final allocation. A fitness value for each rule in the population is the average $T$. The GA selects the best rules ("elite") with number $E$ to the next generation without modification. The remaining $P$-$E$ rules are generated by crossover between elite rules and mutated with probability $P_m$. This process is continued a predefined number G of generations and discovered rules will be stored after completion. Searching rules is conducted with use of GA in the following way:

```
BEGIN
    Initialize $t$ to zero
    Initialize the population $P()$ of CA rules of a size $P$
    Initialize a set of makespan $T$ of a size $P$
    Initialize the number of generations $G$
    Initialize the number of time steps $M$
    Initialize the number of elite rules $E$
    Initialize the mutation probability $P_m$
    FOR G iterations
        Create a set of a size $I$ of initial allocations
        FOR $I$ iterations
            FOR $P$ iterations
                FOR M iteration
                    Determine the next state of CA cells
            ENDFOR
        ENDFOR
        Calculate the average $T$ of $P$
        Move $E$ of the best rules from $P(t)$ to $P(t+1)$
        Crossover with the selected rules from $E$
        Mutation with probability $P_m$
        Add rules to $P (t+1)$
        $t = t + 1$
    ENDFOR
    Store the best rules from $P()$
END
```

In operation phase, when a task set is randomly allocated, the best found rule of CA is able to find an optimal scheduling in a finite number of time steps by selecting the allocation which gives the minimum makespan. The simulation result of the performance evaluation is presented in the next section.

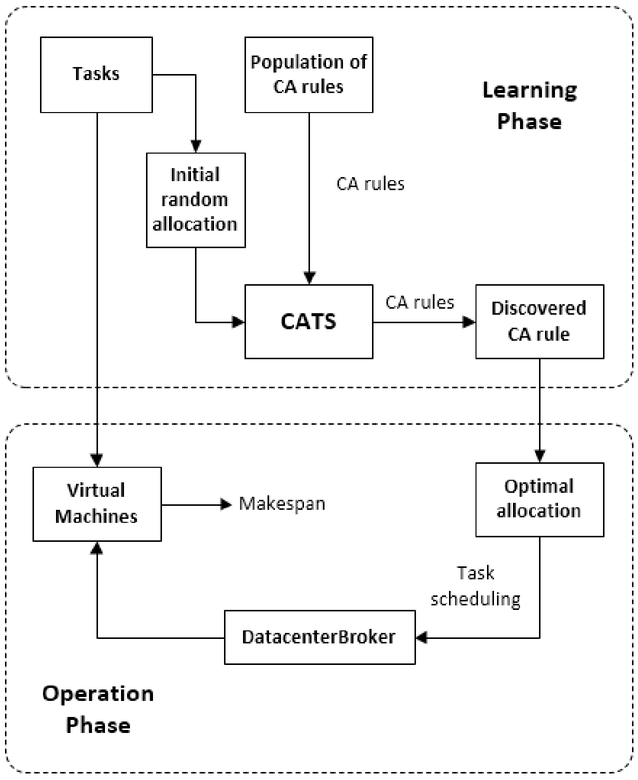

Figure 3. Learning phase and operation phase of CATS scheme

## V. Simulation and Results

### A. *Assumptions*

The performance of our proposed scheme or CATS scheme is evaluated by means of the simulation using CloudSim [11]. We assume that for a task set there is no precedence constraints among tasks that means each task is independent. They also cannot be interrupted during their execution.

### B. *Simulation Parameters for CloudSim*

We assume 10 datacenters with 100, 200, 300, 400 and 500 tasks and 50 VMs. The range of the length of each task is from 5000 Million Instruction (MI) to 15000 MI. Computing performance of VMs is vary between 250 (Million Instruction Per Second) MIPS to 2000 MIPS. All datacenters randomly use allocation policy for VM allocation with Time_shared or

Space_shared and all VMs use only Space_shared policy. The simulation parameters are shown in Table 2.

### *C. Simulation Parameters for CATS Scheme*

The parameters setting for CATS scheme is shown in Table 3. The radius *r* of CA is 1. We use 100 generations with the number of population is 50, also $I$ = 50, $M$ = 50, and $E$ = 25. For crossover and mutation, we use a two-point crossover and a bit-flip mutation with probability ($P_m$) = 0.03 in GA.

TABLE II. PARAMETERS SETTING OF CLOUD SIMULATOR

| Type | Parameter | Value |
|---|---|---|
| Datacenter | Number of datacenters | 10 datacenters |
| | Number of hosts | 2-6 hosts |
| | Type of manager | Space_shared |
| | | Time_shared |
| Virtual Machine (VM) | MIPS | 250-2000 MIPS |
| | VM memory (RAM) | 512-2048 Mbytes |
| | Type of manager | Space_shared |
| Task | Total number of task in a task set | 100-500 tasks |
| | Length of task | 5000-15000 MI |

TABLE III. PARAMETERS SETTING OF CAS-BASED SCHEDULER

| Parameter | Value |
|---|---|
| Number of generation (G) | 100 |
| Number of population (P) | 50 |
| Number of initial allocation (I) | 50 |
| Number of Time step (M) | 50 |
| Number of Elite rule (E) | 25 |
| Mutation probability ($P_m$) | 0.03 |
| CA's radius | 1 |

### *D. Simulation Results*

We compare the performance of our proposed scheme (CATS scheme) with that of the First Come First Served (FCFS). The aim of FCFS scheme is to find the earliest completion time of each task individually and the aim of CA-based scheduler is to find an allocation which minimizes the makespan of a given task set. The total execution of a task set (makespan) of two algorithms are compared. Figure 4 shows that the average makespan of CATS with 100-500 tasks is obviously lower than FCFS. Figure 5 shows a run of CATS with the best found rule after 100 generations, starting with a random initial allocation of 100 tasks. It presents a value of $T$ in 50 time steps. The GA can discover the CA rule at the average T equal 79 at $M$ = 9.

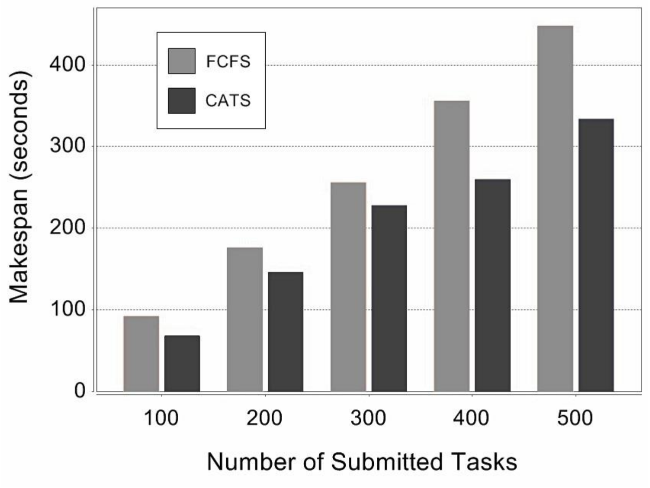

Figure 4. The average makespan of 100-500 tasks set

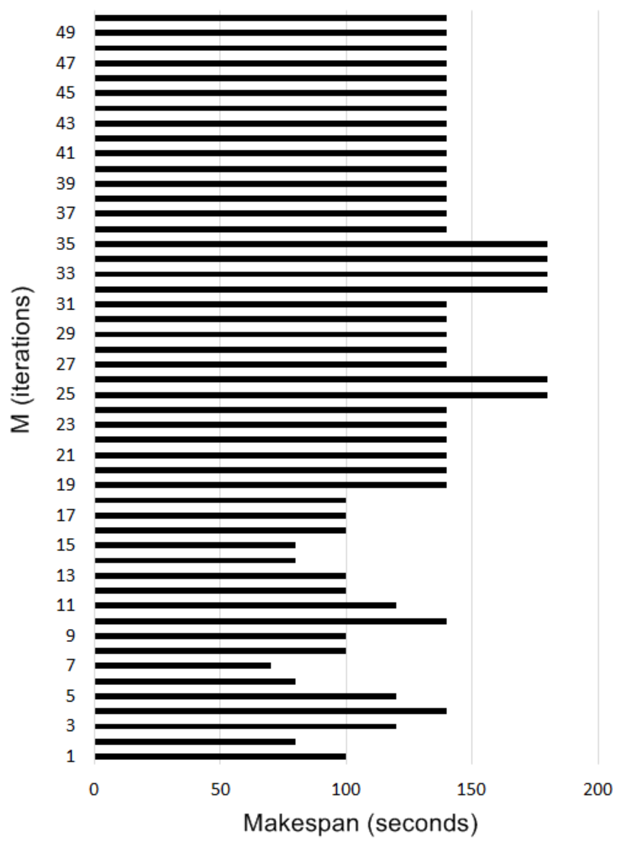

Figure 5. Time diagram of the best found rule for 100 tasks discovered by CATS scheme

## VI. CONCLUSIONS

In this paper, we have proposed the application of cellular automata and genetic algorithm to task scheduling in cloud computing systems call CATS scheme. The cellular automata

(CA) have been considered for setting the task scheduling rules. The effective rule has been discovered by using genetic algorithm (GA). The effectiveness of CATS scheme has been evaluated by means of simulation. The simulation result has shown that CATS scheme can provide the lower makespan comparing with that of the traditional First-Come-First-Serve (FCFS) scheme.

## REFERENCES

[1] M. D. Dikaiakos, "Cloud Computing: Distributed Internet Computing for IT and Scientific Research," Internet Computing, IEEE , vol.13, no.5, pp.10,13, Sept.-Oct. 2009.

[2] R. Buyya, J. Broberg , and A. M. Goscinski, Cloud Computing: Principles and Paradigms (Wiley Series on Parallel and Distributed Computing). Wiley, 2011.

[3] F. Chang and R. Viswanathan, "Optimal Resource Allocation in Clouds," Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on , pp.418,425, 5-10 July 2010.

[4] A. Swiecicka and F. Seredynski, "Applying cellular automata in multiprocessor scheduling," Parallel Computing in Electrical Engineering, 2002. PARELEC '02. Proceedings International Conference on , pp.177,182, 2002.

[5] F. Seredynski and P. Bouvry, "Multiprocessor scheduling algorithms based on cellular automata training," Intelligent Engineering Systems Through Artificial Neural Networks, vol. 13, pp. 255-260, 2003

[6] D. Rajarshi , M. Melanie, and J. P. Crutchfield, "A Genetic Algorithm Discovers Particle-Based Computation in Cellular Automata," Proceedings of the International Conference on Evolutionary Computation. The Third Conference on Parallel Problem Solving from Nature: Parallel Problem Solving from Nature, p.344-353, October 09-14, 1994.

[7] S. Wolfram, Cellular Automata And Complexity: Collected Papers. Westview Press, 1994.

[8] J. L. Schiff, Cellular Automata: A Discrete View of the World. Wiley-Interscience, 2008.

[9] D. E. Goldberg, Genetic algorithms in search, optimization, and machine learning. Addison-Wesley Professional, 1989.

[10] R. N. Calheiros, R. Ranjan, and R. Buyya, "CloudSim: A Novel Framework for Modeling and Simulation of Cloud Computing Infrastructures and Services," Technical Report, GRIDS-TR-2009-1, Grid Computing and Distributed Systems Laboratory, The University of Melbourne, Australia, 2009.

[11] R. N. Calheiros, R. Ranjan, A. Beloglazov, and R. Buyya, "CloudSim: A Toolkit for the Modeling and Simulation of Cloud Resource Management and Application Provisioning Techniques," Journal Software—Practice & Experience archive, vol. 41, pp. 23-50, 2011.

[12] B. Ghalem , Z. T. Fatima, and Z. Wieme, "Approaches to Improve the Resources Management in the Simulator CloudSim," ICICA 2010, LNCS 6377, DOI: 10.1007/978-3-642-16167-4_25, pp. 189–196, 2010.

[13] R. Buyya, R. Ranjan , R. N. Calheiros, "Modeling and Simulation of Scalable Cloud Computing Environments and the CloudSim Toolkit: Challenges and Opportunities," Proceedings of the 7th High Performance Computing and Simulation (HPCS 2009) Conference, Leipzig, Germany, DOI: 10.1109/HPCSIM.2009.5192685, 2009.