Mingzhang Wu, Janne Koljonen and Timo Mantere*

Addressing Resource Allocation Issues in Cloud Computing Environment with Ant Colony Optimization

Abstract: Cloud computing is a fast growing and attractive paradigm in information technology, since it allows using resources on-demand wherever and whenever needed. The use of dynamic cloud resource allocation allows immediate accommodation to unpredictable demands and improvement in the return on investment as for the computing infrastructure. The cloud resources allocation optimization model is one of the core parts in cloud computing. However, despite the recent growth of the research in the cloud computing area, several problems with the process of resource allocation remain unaddressed. Cost and performance are two important but contradictive objectives in the cloud resources allocation process. Cost-performance trade-off constitutes a challenging multi-objective optimization problem in cloud resources allocation. In this paper, a new optimization model is proposed to solve this multi-objective optimization problem effectively. An ant colony optimization algorithm that optimizes the Quality of Service (QoS) and the response time in a simulated CloudSim environment that models five servers of varying characteristics. Experimental results demonstrate the effectiveness of the designed algorithms. Ant colony algorithm shows mostly higher performance than the round robin and greedy assignment algorithms that were used as benchmarks.

Keywords: ant colony optimization, cloud computing, CloudSim, cost-performance, resource allocation, trade-off problem

***Corresponding Author: Timo Mantere:** University of Vaasa, School of Technology and Innovations, E-mail: timo.mantere@uva.fi

Mingzhang Wu: E-mail: mingzhang.wu@gmail.com

Janne Koljonen: University of Vaasa, School of Technology and Innovations, E-mail: janne.koljonen@uva.fi

1 Introduction

1.1 Cloud computing

In the recent years, information technology (IT) has been integrated into our daily life more and more. The major applications are build up based on network and internet technologies. We are now in an era of "big data" with rapid growth on the number of transactions, information, and data. However, low cost, fast speed, and efficient computing are desired. The traditional network and local computation capacity are unable to meet these needs. Instead, distributed network technologies are developed to enable the utilization of distributed computing resources from the internet. How to integrate and distribute the resources, such as servers, over the internet give new research topics to be considered.

Cloud computing, as a new emerging information and communication technology concept, has been an interesting topic recently. There are many definitions of cloud computing. Cloud computing is a result of the convergence of several technologies, such as, (1) hardware, (2) internet technologies, (3) distributed computing, and (4) systems management. The main advantage of cloud computing is providing computing resources based on the public utility model (compare to water, electricity, gas, and telephony) to enhance reliability, scalability, and performance [1].

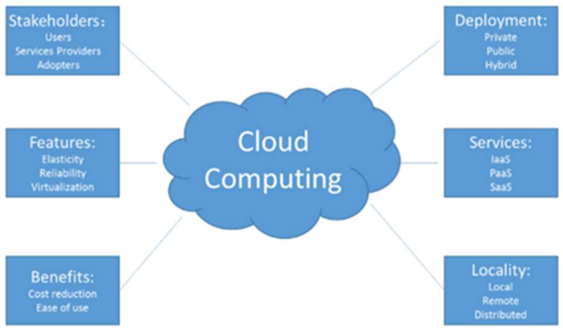


Fig. 1. A holistic view of cloud computing [2]

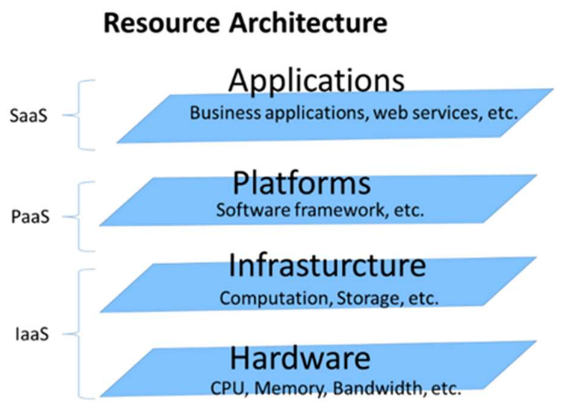


Fig. 2. Network topology of virtual resources in cloud computing

From the technical perspective, cloud computing is the integration of many aspects of technologies, such as (1) virtualization, (2) utility computing, and (3) distributed computing, among others.

From the business perspective, cloud computing is a new business model. It enables (1) sharing information among users, (2) buying resources on-demand without large investments, (3) selling capacity to many users, and subsequently (4) improving the return on investments due to better rates of capacity use. Furthermore, (5) investing on the latest, high-performance infrastructure should give a business advantage to the service provider.

Cloud computing model has changed and will affect many companies' business model and operational status, not only for the IT industry. Cloud computing can provide three types of services: Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS). A summary of cloud computing properties are given in Figs. 1 and 2.

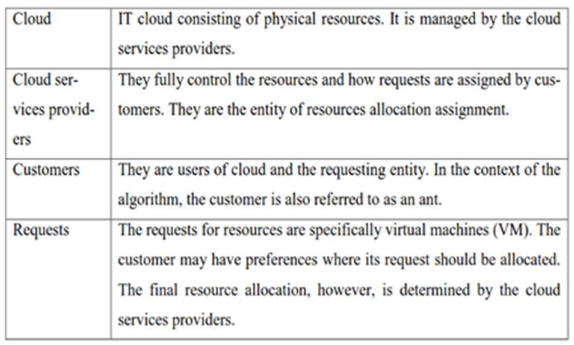
The National Institute of Standards (NIST) has emphasized the elasticity feature of computing resources in their definition of cloud computing [3], which is largely accepted and frequently cited. It defines cloud as follows: "Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction." The terminology used in this study is clarified in Table 1.

This article is based in major parts on the first author's master's thesis [4].

1.2 Ant colony optimization

Ant Colony Optimization (ACO) is a heuristics proposed by Marco Dorigo [5] in the early 1990s. It was inspired by the observation of the collaboration activities of ants searching for food. Ants can gradually find out the shortest path between a food source and the nest of the colony. ACO is suitable for many optimization problems that can be modeled as a graph, including resource assignment. However, in its original form, ACO modifies the edges of a graph, not the nodes as will be done in this paper.

Table 1. Terminologies used in this research



Cloud	IT cloud consisting of physical resources. It is managed by the cloud services providers.
Cloud ser-vices provid-ers	They fully control the resources and how requests are assigned by cus-tomers. They are the entity of resources allocation assignment.
Customers	They are users of cloud and the requesting entity. In the context of the algorithm, the customer is also referred to as an ant.
Requests	The requests for resources are specifically virtual machines (VM). The customer may have preferences where its request should be allocated. The final resource allocation, however, is determined by the cloud services providers.

The weakness of many optimization methods is their inability to handle more than one objective. In addition, these methods often employ local and greedy (e.g., hill-climbing) approaches, which are prone to find only a local optimum [6]. Instead, ACO is considered as a part of the family of evolutionary algorithms that use multiple parallel trials and stochastic search to improve the probability to find the global optimum. How ACO is implemented in this study, is reported in Section 3.

ACO has previously been applied successfully to a number of benchmark combinatorial optimization problems (see Section 2 for more details). In this study, it is proposed how to use ACO to solve the multi-objective model for Cost-Performance trade-off problem (CPTOP). The concept of ACO-based multi-objective CPTOP model is designed and tested using *CloudSim*, which is an extendable discrete-event simulation toolkit that enables modeling and simulation of cloud computing environments and the application-provisioning environment.

2 Related work

Resource allocation for clouds has been studied with a very wide scope in the literature. The problem of determining an optimal allocation of the requests to a pool of resources is NP-hard (non-deterministic polynomial-time hard) problem. Nevertheless, many optimization strategies may be used to solve it efficiently. In particular, several heuristic algorithms have been proposed by researchers for optimal allocation of cloud resources [7].

Fidanova [8] proposed an adaptive resource allocation algorithm in cloud computing environment.

That paper used adaptive min-min scheduling and list scheduling, but it was used in a static manner [9].

In Foster et al. [10], the authors proposed an optimal virtual machine placement algorithm for minimizing the cost that cloud customer have to pay to the cloud service provider, when they need virtual machine from cloud computing environment access as a part of the cloud service. In Chaisiri et al. [11], the authors described the multi-objective mechanism for scheduling applications that take various cost constraints and the availability of resources into account.

In Frincu and Craciun [12], the focus is more on the resource allocation strategy in selecting the cloud provider, but the approach is static as for selecting the data center from the distributed environment where the global data center is available, with taking care of timing parameter as in [9]. Chimakurthi [7] propose an energy-efficient mechanism to minimize the number of servers to be used for hosting the services and allocating the cloud resources to the applications.

The paper by Hua et al. [13] propose an ant colony optimization algorithm for resource allocation, in which all the characteristics in cloud are considered. It has been compared with a genetic algorithm and a simulated annealing algorithm, proving that it is suitable for computing resource search allocation in cloud computing environment.

Omara et al. [14] propose an optimization solution to the allocation of shared resources to minimize the estimated cost and enhance virtual machine configuration. Banerjee et al. [15] propose optimization method by using modified Ant Colony Framework to optimize the scheduling throughput to the service for all the diversified requests using different resource allocators available. Wei et al. [16] suggest a deadline and budget constraint cost-time optimization algorithm for scheduling dependent subtasks by using game theory.

In [17], the cost-performance tradeoff in cloud IaaS was addressed, where the problem has been formulated as a multi-objective optimization. The proposed model was built based on a fine-grained charging model and a normalized performance model. The implementation using genetic algorithms and the experimental results proved the effectiveness of the proposed model.

3 Experimental setup

As mentioned earlier, the optimal cloud resource allocation problem will be studied. Resource allocation in a cloud is understood here as the allocation of virtual machines (VM) to physical resources. The cloud network is clearly dynamic, so rather than allocating according to the physical resources of a node, it should be done with respect to the instantaneously available free resources of a node. The result of the optimization is an assignment of VM-node pairs [18] and the related performance metrics.

3.1 ACO-based Multi-objective CPTOP Model

The master-slave architecture is a mature architecture with a single master server or job tracker and several slave servers, which has been widely used in cloud computing like in Google's MapReduce and Hadoop. Fig. 3 shows a typical scenario of the network topology of virtual resources in cloud computing, which is based on the master-slave architecture.

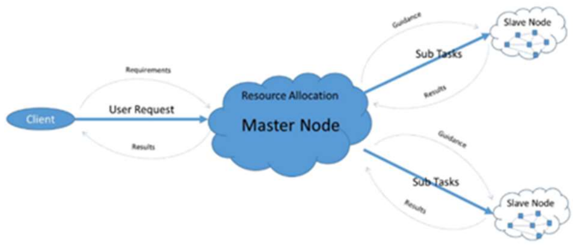


Fig. 3. Cloud computing resource architecture

In the master-slave architecture, a request is first submitted to a master node in the cloud platform by the user. Then the request is divided into several executable tasks in the master node and the generated tasks are distributed to different slave nodes.

After receiving the assigned sub tasks, the slave nodes will find appropriate resources. The tasks are executed in the slave nodes separately with the guidance of the master node, and the results are returned to the master node. The results include information about processing abilities, characteristics (number of CPU cores, amount of main memory, etc.), and cost.

Finally, the distributed results are combined together in the master node and sent to the requesting user. Furthermore, the master node is responsible for monitoring the all the steps and re-executing the failed tasks.

In this paper, the role of ACO is simulate several generations of artificial ants that search for the optimal solution. Every ant of a generation builds a solution step-by-step going through several probabilistic decisions. In general, ants that find a good solution mark their paths through the decision space by putting some amounts of pheromone on the edges of the path.

The pheromone attracts the ants of the next generations, and the result is that they search the solution space near the previous good solutions. In addition to the pheromone values, the ants are usually guided by some problem-specific heuristic for evaluating the trial solutions.

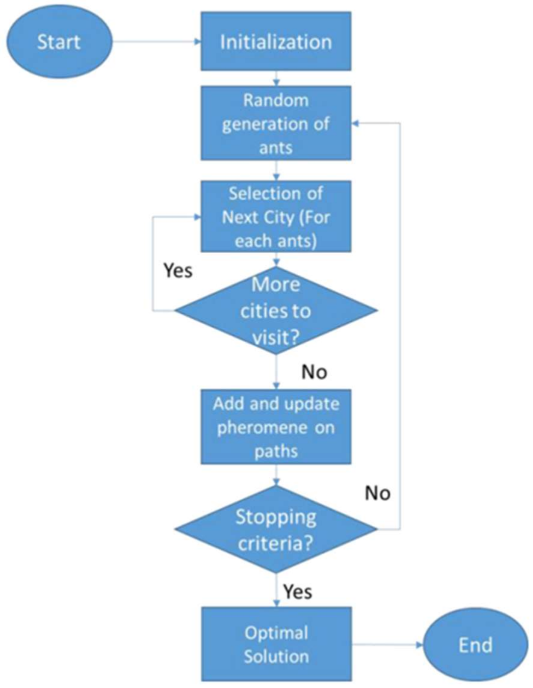

Start
Initialization
Random generation of ants
Selection of Next City (For each ants)
Yes
More cities to visit?
No
Add and update pheromene on paths
No
Stopping criteria?
Yes
Optimal Solution
End

Fig. 4. Flowchart of Ant Colony Optimization

In order to apply ACO to tackle Cost-Performance trade-off problem (CPTOP), the problem should be transformed into a Travel Salesman Problem (TSP) in Fig. 4. Moreover, the separate objectives of cost and performance should be integrated into a single objective function.

An analysis of a cloud computing platform reveals several characteristics that are in common with the standard ACO: slave nodes being analogous to food locations, master nodes to nests, and resource allocation to foraging activity. For each of the slave nodes, ACO needs to calculate the free cloud resources. If the available resource exceeds the user's requirements for accomplishing the sub tasks, then this slave node should be allocated to the respective sub task. If the remaining resource is insufficient for the minimal requirement of the user, another appropriate slave node is searched for.

When the slave node sends back the results, the pheromone values are updated and saved. The master node will send new guidance according to the returned information to other slave nodes. This method can optimize the final results. The search for suitable slave node activity is conducted in a certain range to decrease the cost and increase the performance.

3.2 CloudSim

CloudSim provides many ways for managing and utilizing the resources, such as virtual machine (VMs), datacenter, and so on. It supports the research and development of cloud computing in testing the performance of a newly developed application service in a controlled and easy to set-up environment. CloudSim offers: (1) support to modeling and simulation of a large cloud computing infrastructure, (2) a self-contained support data center, service agent, scheduling and allocation strategy platform.

The framework and architecture of CloudSim consist of four main layers:

– *SimJava layer* supports several core functionalities required for simulation, such as queuing and processing of events, creation of system components (services, host, data center, broker, VMs), and management of the simulation clock.
– *GridSim* layer includes libraries that support high-level software components for modeling multiple grid infrastructures, including networks and associated traffic profiles, and fundamental grid components.
– *CloudSim* layer provides support for modeling and simulation of virtualized cloud-based data center environments including dedicated management interfaces for VMs, memory, storage, and bandwidth.
– *CloudSim stack* is the top layer, and it includes the user code that defines basic entities for hosts (number of machines, their specification, and so on), applications (number of tasks and their requirements), VMs, number of users and their application types, and broker scheduling policies.

3.3 Optimization

An important step in defining an algorithm for the resource allocation problem is defining the objective of the optimization. The objectives for the customers and the cloud service providers are different. In a simplified view, the objective of the customer is to maximize the performance of resources with a fixed cost. For the cloud service provider, the total amount of resources is fixed, and the objective is to add as many customer requests to the cloud as possible.

The more detailed rationale is as follows:

– As for the customers: When customers send requests to the cloud services provider for execution of their tasks, they seek to reduce their costs by transferring their operation to the cloud environment. Then the best offer is selected and the corresponding resources will be allocated to run the task. Moreover, they want to receive as good service as possible (within the limits of the cost). From the customers' perspective, they are selfish, because the customers are not concerned about the other customers in cloud.
– As for the cloud services providers: They want to

increase the profits obtained from the limited resources through the increase of income from hosting more users while minimizing the cost (investments) by optimal assignment of customers' requests to the resources.

Cost and performance are two competing objectives in cloud resources allocation. It is a NP-hard and multi-objective optimization problem without a unique solution. The number of possible solutions grows exponentially with respect to the number of resources and customers.

The optimization goal is to find, in some respect, the best trade-off between cost and performance. There are two challenges:

– A multi-optimal approach seems infeasible due to the hardness of the problem. Solutions optimal with respect to different criteria will tend to be vastly different, and there is no way to find a trade-off by interpolation due to the discrete nature of the resource assignment.
– It is difficult to define performance and quality from a system perspective. In the dynamic resource allocation, requests are assigned one by one, and simple heuristics would give no guarantee of fairness in performance. In this paper, the minimum level of performance is therefore determined by the Service Level Agreement (SLA) of each request.

In the ACO, the movements of the ants are controlled by probabilities that are products of two parts. The first is an assignment probability that is proportional to the attractiveness of a match from the customer point of view (called visibility in [5]). The second is a memory of the best past assignments represented by the fictitious pheromone trail. As long as the SLA of a request can be fulfilled the attractiveness is non-zero, otherwise it is zero.

The probability of transition to another (including current) node is evaluated only for feasible assignments:

$$P_{ij} = \begin{cases} \frac{(\tau_{ij}(t))^{\alpha}\,\eta_{ij}(t))^{\eta}}{\Sigma(\tau_{ij}(t))^{\alpha}((\eta_{ij}(t))^{\beta}} \\ 0, \end{cases} \quad (1)$$

where the pheromone $\tau_{ij}(t)$ and the attractiveness $\eta_{ij}(t)$ dependent on time t. The pheromone density changes in each cycle, while the attractiveness in every move within a cycle.

Cost can be defined in terms of idle capacity, that is, unoccupied capacity that cannot be assigned to another VM due to limitations in some other resource type. The cost will depend on the applications, or, in other words, the distribution of demands of arriving requests.

The cost for a cloud service providers can be expressed in the degree of infrastructure utilization or, equivalently, return on investments. The service providers wish to allocate jobs to resources in a best-fit manner, so that an allocated customer occupies no more than the necessary minimum.

The greedy principle from the cloud provider's perspective is that the more VMs can be allocated, the higher the utilization and the return on investment is. As a metric for system efficiency, the energy of the relative free resources is used. The optimization then follows the principle of minimum energy. The system energy is defined as:

$$E = \sum_{i=1}^{n}(C_i - \sum_{j=1}^{v_i} r_{ij})^2 \quad (2)$$

where C_i is the capacity of the server i and r_{ij} is the VM capacity requirement of VM j on node i. The total energy sums over the squared free resources of all servers.

It is important to performance features into account while allocating resources, since it allows providing the customers high Quality of Service (QoS), with the best response time as an example, and to meet the Service Level Agreement (SLA) established. Indeed, it is not easy to handle efficiently resource allocation processes in Cloud, since the applications deployed in Cloud obey non-uniform usage patterns, and the cloud allocation architecture needs to provide different scenarios of resource allocation to satisfy the demands and provide quality [19].

Now the actual algorithm is an adaptation of the ACO algorithm for solving the Traveling Salesman Problem (TSP), described in [5]. The assignment problem is modeled as a complete graph on the set of n nodes.

Initially, the ants are distributed between the nodes in a round-robin fashion. They could also originate from a source node (a nest), but this is not necessary, as the algorithm only performs a single iteration in each cycle. The ants could also be distributed randomly, which would affect the order of assignments. In the example below, however, this has no or little effect.

The ants move according to a matrix of transition probabilities, where self-loops are allowed, so that an ant may request its job to be assigned to the node it originally occupies. As opposed to the TSP, where the attractiveness is fixed, the system state changes with assignment of a new job (the property of the ant, or customer). Therefore, after each move, the transition probabilities change and must be recalculated. The attractiveness of a server to a given customer decreases when resources are assigned to another customer. As a measure of attractiveness, the (scaled) available CPU processing power of the server is used.

Next, the system cost is calculated according to (2). This energy could be a composite measure that includes other resources, such as RAM as suggested in [1]. However, now the energy is based only on the CPU processing power.

The deposited amount of pheromone, $\Delta\tau$ on each edge is now dependent on the system cost, rather than the trail of a single ant as in the TSP. This quantity is given by:

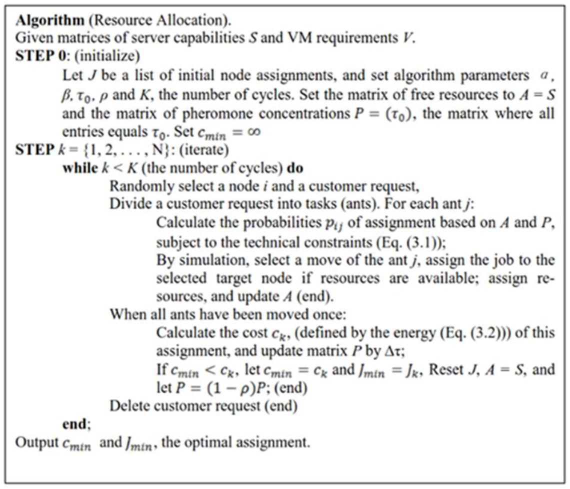
$$\Delta\tau = Q/c_k \tag{3}$$

where Q is a scaling constant and c_k the cost in cycle $k \in \{1, 2, ..., N\}$. Since c_k can be zero, a maximum limit on $\Delta\tau$ is set to one. This limit is rather arbitrary, and it is an additional system parameter that affects the convergence properties of the algorithm.

The cost is used to update matrix **P**, first by multiplying all previous pheromone levels p_{ij} by the evaporation constant $(1 - \rho)$, and then by adding $\Delta\tau$ onto edges describing the assignments used in the iteration.

The minimum cost and the corresponding assignment obtained so far is recorded after each cycle. Matrix **A** and the vectors of the free node resources and assignments are restored to their initial values, corresponding to not yet assigned jobs. Table 2 summarizes the ACO algorithm.

Table 2. Resource allocation algorithm



```
Algorithm (Resource Allocation).
Given matrices of server capabilities $S$ and VM requirements $V$.
STEP 0: (initialize)
    Let $J$ be a list of initial node assignments, and set algorithm parameters $\alpha$,
    $\beta, \tau_0, \rho$ and $K$, the number of cycles. Set the matrix of free resources to $A = S$
    and the matrix of pheromone concentrations $P = (\tau_0)$, the matrix where all
    entries equals $\tau_0$. Set $c_{min} = \infty$
STEP $k = \{1, 2, \dots, N\}$: (iterate)
    while $k < K$ (the number of cycles) do
        Randomly select a node $i$ and a customer request,
        Divide a customer request into tasks (ants). For each ant $j$:
            Calculate the probabilities $p_{ij}$ of assignment based on $A$ and $P$,
            subject to the technical constraints (Eq. (3.1));
            By simulation, select a move of the ant $j$, assign the job to the
            selected target node if resources are available; assign re-
            sources, and update $A$ (end).
        When all ants have been moved once:
            Calculate the cost $c_k$, (defined by the energy (Eq. (3.2))) of this
            assignment, and update matrix $P$ by $\Delta\tau$;
            If $c_{min} < c_k$, let $c_{min} = c_k$ and $J_{min} = J_k$, Reset $J$, $A = S$, and
            let $P = (1 - \rho)P$; (end)
        Delete customer request (end)
    end;
Output $c_{min}$ and $J_{min}$, the optimal assignment.
```

4 Experiments

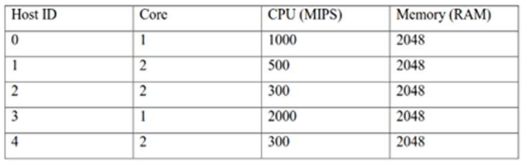
To test the algorithm, the small cluster setup described in [18] was used. The simplicity of this scenario with five servers having different characteristics and a single type of virtual machine (VM) makes the manual comparison with other assignment schemes straightforward. The algorithm as such should be easily extended to larger and more general cases.

To compare the algorithm with other assignment schemes, the results with the round-robin and a customer greedy heuristic schemes were both tested. In the CloudSim experiments, the goal was to keep things as simple as possible apart from the hosts and VMs. Only one user, one datacenter and one broker were created and initiated. The VMs represent the ants, and the cloudlets jobs assigned to the VMs.

The three assignment strategies (round-robin, greedy, and ACO) were implemented in CloudSim using Java. The round-robin assignment was implemented on the basis of a project in Github [20]. The number of cloudlets was set to 10 as the code in [20] also used.

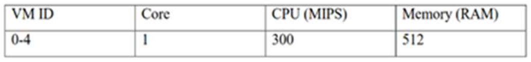
The number of ants (VMs) is set equal to the number of nodes. The properties of the host servers in the cluster are listed in Table 3, and of the virtual machines in Table 4.

Table 3. Host servers specification: MIPS and RAM capacities



Host ID	Core	CPU (MIPS)	Memory (RAM)
0	1	1000	2048
1	2	500	2048
2	2	300	2048
3	1	2000	2048
4	2	300	2048

Table 4. Virtual machine specification: requirements on MIPS and RAM



VM ID	Core	CPU (MIPS)	Memory (RAM)
0-4	1	300	512

Tables 5, 6, and 7 show the results of the three algorithms. The reported metric for each of the hosts is the percentage of free capacity, calculated as:

$$1 - \frac{\text{occupied capacity}}{\text{total host capacity}}. \tag{4}$$

Two versions of the metrics are calculated: taking and not taking into account the number CloudSim Processing Elements (PEs), i.e., cores.

The cost as defined in Equation (2), that is, the sum of the squared entries. The round-robin and the greedy algorithms are deterministic, whereas the ACO algorithm is stochastic. ACO may therefore give a different result at each run, and not even a reasonable convergence is guaranteed. This depends on the random number generator.

In the round-robin scheme, the VMs are simply distributed one at each node, and the relative free capacity is shown in Table 5. The cost is 1.3725 and taking the number of processors into account, the energy is 2.2025.

Since the round-robin assignment scheme is deterministic and not an optimization method, it is likely to perform poorly when there are VMs with different requirements. In this example, however, the round-robin assignment is good, under the energy metric.

Table 5. Efficiency of round-robin assignment

Host ID	No. VM	Free Cap. w/o PEs	Free Cap. with PEs
0	1	0.7	0.7
1	1	0.4	0.7
2	1	0.0	0.5
3	1	0.85	0.85
4	1	0.0	0.5
Cost (Energy)		1.37	2.20

The greedy assignment method lets each customer choose server according to the largest amount of available processing capacity. These results are shown in Table 6. The cost (energy) in this case is 3.65. The same value is achieved when taking the number of processors into account, since there are zero VMs in all multi-core hosts (Host IDs 1, 2, and 4, compare to Table 3). The greedy scheme is essentially what would be expected from a single iteration of the algorithm.

Table 6. Efficiency of greedy assignment

Host ID	No. VM	Free Cap. w/o PEs	Free Cap. with PEs
0	1	0.7	0.7
1	0	1.0	1.0
2	0	1.0	1.0
3	4	0.4	0.4
4	0	1.0	1.0
Cost (Energy)		3.65	3.65

The ACO algorithm applied to the same problem gave the assignments shown in Table 7. It can be seen that the lower capacity nodes (1, 2, and 4; see Table 3 for the Host specifications) are assigned VMs, but not node 3. The minimum energy obtained is 1.32. After reaching the minimum energy, the algorithm was run for up to N = 10,000 without showing any further improvement. Taking the number of processors into account, the energy is 2.15 for this policy.

Table 7. Efficiency of ACO assignment

Host ID	No. VM	Free Cap. w/o PEs	Free Cap. with PEs
0	2	0.4	0.4
1	1	0.4	0.7
2	1	0.0	0.5
3	0	1.0	1.0
4	1	0.0	0.5
Cost		1.32	2.15

The parameter values used are α = 0.5, β = 0.5, ρ = 0.1 and υ_0 = 0.1. The cut-off limit for the inverse of the cost was set (rather arbitrarily) to 1. Elaborating on the system parameters would probably influence the convergence of the algorithm significantly, but it has not been studied in detail here.

The convergence performance (Fig. 5.) shows one run trace of the algorithm. This case shows an initial guess that already better than the cost of the greedy assignment (compare to Table 6). After that ACO finds a local minimum in two more iterations. The figure shows a case of an optimization with a particular (lucky guess) seed, which converged very fast. Normally, such an optimization would show a jagged curve stretching much longer on the x-axis. The figure is intended to show the actual ideal convergence rather than the convergence performance in general.

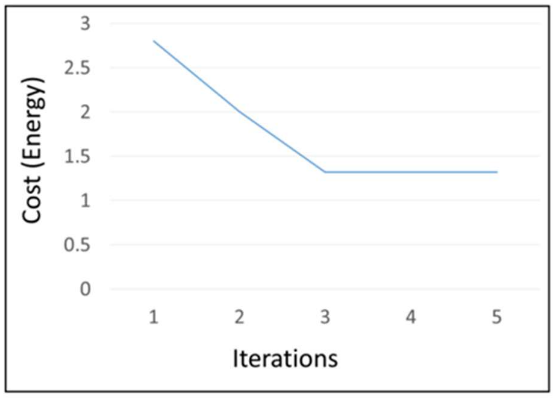


Fig. 5. Ideal convergence of ACO

Fig. 6. shows the energy for each of the three assignment strategies for an increasing number of standard size VMs as defined in Table 4. The ACO algorithm described in this study (green line) has lower energy than the other two, although the round-robin strategy (blue line) is close to optimal. The greedy algorithm is evidently the worst of these three with practically any number of VMs.

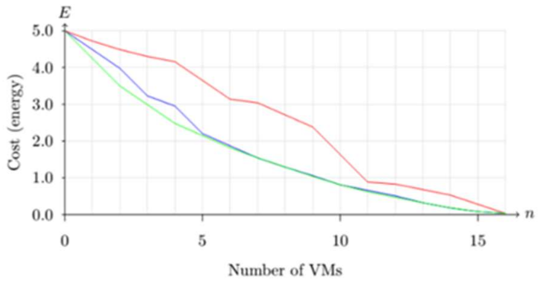


Fig. 6. Comparison of the efficiency of the algorithms: round-robin (blue), greedy (red), and ACO (green).

5 Conclusions

Cloud computing enables a more efficient way to use utility computing resources and services. Users can access the computing resources with virtualized technologies and pay only for the resource accessed while getting the level of quality of service (QoS) wanted. In this paper, a modified ant colony optimization algorithm for cost and performance trade-off optimization problem is developed to encourage the formation of solutions to achieve the cost minimization.

An optimal assignment was found by minimizing a combined energy function that measures cloud provider's costs. However, the test setup was relatively limited and simplified, due that a full-scale simulation of cloud computing services are complicated. The benefit for the cloud provider is to maximize the possibility to add further VMs to the existing cloud infrastructure without performance degradation or delays.

Three assignment policies were simulated and tested in CloudSim: round-robin, a customer greedy heuristic, and an optimized allocation implemented as an Ant Colony Optimization algorithm. When comparing the three assignment policies, the round-robin can be said to be both simple and efficient. The greedy assignment, where a customer can choose to allocate a VM to the host with the freest capacity was rather expensive. However, in this implementation, each processor had the same MIPS, so the results might be different in an environment with more versatile set of resources available.

Finally, the way that ACO uses resources differed from the round robin method in this case only so that one extra VM was assigned to host number 0, instead of host number 3. Nevertheless, this little change makes ACO to obtain the best energy function values. However, in this simple setup the difference is only slight. The further tests with varying numbers of VMs validated the mutual order of the three algorithms; ACO consistently outperforms the simple round-robin method slightly, while the round-robin method outperforms the simple greedy method significantly.

By using this dynamic optimization, the new request will be given to some host, and in the same time, an already assigned VM can in principle be re-assigned, but this case has not been tested in this study. Instead, to reach an optimal solution, the algorithm starts afresh in each iteration. It converges after a (random) number of iterations, and this converged result is then the assignment policy.

These experiments have some limitations. Actually, the implementation of ACO in CloudSim makes a solution 'all at once', not just a list of nodes like in CloudSim. Therefore, it is recommended to develop an ACO variant that could find an optimal policy with a more dynamic situation, where VMs are created and terminated all the time.

References

[1] Asha N., Rao G. R., A Review on Various Resource Allocation Strategies in Cloud Computing, International Journal of Emerging Technology and Advanced Engineering (IJETAE), 2013, 3(7).

[2] European Commission, The Future of Cloud Computing – Opportunities for European Cloud Beyond 2010, Public Report, European Commission, 2010.

[3] Mell P., Grance T., The NIST definition of cloud computing (version 15), 2009, retrieved from https://www.nist.gov/sites/default/files/documents/itl/cloud/cloud-def-v15.pdf.

[4] Wu M., Addressing Resources Allocation Issues in Cloud Computing Environment, M.Sc. thesis, University of Vaasa, Vaasa, Finland, 2016.

[5] Dorigo M., Maniezzo V., Colorni, A., Ant system: optimization by a colony of cooperating agents, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 1996, 26(1), 29–41.

[6] Afshar A., Kaveh A., Shoghli O. R., Multi-objective optimization of time-cost-quality using multi-colony ant algorithm, Asian Journal of Civil Engineering (Building and Housing), 2007, 8(2), 113–124.

[7] Chimakurthi L., Power efficient resource allocation for clouds using ant colony framework, 2011, arXiv preprint arXiv:1102.2608.

[8] Fidanova S., ACO algorithm for MKP using various heuristic information, In: Dimov I., Lirkov I., Margenov S., Zlatev Z. (eds.), Proceedings of the International Conference on Numerical Methods and Applications (20–24 August 2002, Borovets, Bulgaria), Springer, Berlin Heidelberg, 2002, 438–444.

[9] Parikh K., Hawanna N., Haleema P.K., Jayasubalakshm R., Virtual Machine Allocation Policy in Cloud Computing Using CloudSim in Java, IJGDC, 2015, 8(1), 145–158.

[10] Foster I., Kesselman, C. (Eds.), The Grid 2: Blueprint for a new computing infrastructure, 2nd ed., Morgan Kaufmann, 2003.

[11] Chaisiri S., Lee B. S., Niyato, D., Optimal virtual machine placement across multiple cloud providers, In: Proceedings of the IEEE International Conference on Services Computing (21–25 September 2009, Bangalore, India), IEEE Asia-Pacific, 2009, 103–110).

[12] Frincu M. E., Craciun C., Multi-objective meta-heuristics for scheduling applications with high availability requirements and cost constraints in

multi-cloud environments, In: The Proceedings of 4th IEEE/ACM International Conference on Utility and Cloud Computing (5–7 December 2011, Melbourne, Australia), IEEE, 2011 267–274.

[13] Hua X. Y., Zheng J., Hu W. X., Ant colony optimization algorithm for computing resource allocation based on cloud computing environment, Journal of East China Normal University (Natural Science), 2010, 1(1), 127–134.

[14] Omara F. A., Khattab S. M., Sahal R., Optimum Resource Allocation of Database in Cloud Computing. Egyptian Informatics Journal, 2014, 15(1), 1–12.

[15] Banerjee S., Mukherjee I., Mahanti P. K., Cloud computing initiative using modified ant colony framework, World academy of science, engineering and technology, 2009, 56, 221–224.

[16] Wei G., Vasilakos A. V., Zheng Y., Xiong N., A game-theoretic method of fair resource allocation for cloud computing services, The journal of supercomputing, 2010, 54(2), 252–269.

[17] Kong S., Li Y., Feng, L. (2012). Cost-performance driven resource configuration for database applications in IaaS cloud environments, In: Ivanov I., van Sinderen M., Shishkov B. (eds.), Cloud Computing and Services Science (18–21 April 2012, Porto, Portugal), Springer, New York, 2012, 111–129.

[18] Lee H. M., Jeong Y. S., Jang, H. J., Performance analysis based resource allocation for green cloud computing, The Journal of Supercomputing, 2014, 69(3), 1013–1026.

[19] Sagbo K. A. R., Houngue, P., Quality architecture for resource allocation in cloud computing, In: Service-Oriented and Cloud Computing, Springer, Berlin Heidelberg, 2012, 154–168.

[20] AnanthaRajuC, CloudSim Example with Round Robin Data center broker & Round Robin Vm Allocation Policy with Circular Hosts List, 2015, retrieved from https://github.com/AnanthaRajuC/CloudSim-Round-Robin.