Dynamic application deployment in federations of clouds and edge resources using a multiobjective optimization AI algorithm

Ram Govinda Aryal
Technology Management, Economics, and Policy
Seoul National University
Seoul, Republic of Korea
aryal.rg@gmail.com

Jörn Altmann
Technology Management, Economics, and Policy
Seoul National University
Seoul, Republic of Korea
jorn.altmann@acm.org

***Abstract*—Cloud federation brings, besides various opportunities, challenges for resource allocation. Optimized allocation of resources to applications is one of these challenges. The challenge is further enhanced by the heterogeneity of provider resources, which can be edge resources or cloud resources, and by the perception of the relative importance of optimization objectives. What is required by a federation broker, in such a context, is an efficient decision-making algorithm for virtual machine (VM) placement. Such an algorithm should first identify eligible resources and, then, select optimal combinations of resources within the federated cloud, considering multiple optimization objectives that are derived from the application requirements. The relative weights of which should be tuned to meet individual application needs. Furthermore, it should be able to work dynamically throughout the application lifecycle rather than only during the initial deployment to address changes in application and user behaviors. This paper proposes such an algorithm for the BASMATI cloud federation architecture. The results of multiple runs of our simulation demonstrate that the algorithm, which is a genetic algorithm, is efficient, can provide optimal solutions for VM placement decision making, and can be tuned to address specific application needs.**

Keywords—cloud federation, resource allocation, optimization, VM placement, cloud service broker, mobile cloud computing, fog and edge computing, application deployment, genetic algorithm, platform management, artificial intelligence.

I. INTRODUCTION

Cloud computing is a major computing paradigm at present. The wider adoption of clouds brings along with it, new opportunities and challenges [1]. On the one hand, resource pooling capability of clouds has created opportunities for new business models like cloud brokering [2]. On the other hand, requirements of economic efficiency and performance bring challenges to resource allocation. Issues of economic nature (e.g., underutilization due to over-provisioning and violation of SLAs due to under-provisioning of resources) come into play [3], which are mutually conflicting. Besides, the economics of scale discriminates small providers by limiting their competitive strengths [4]. These opportunities and challenges act as push and pull factors for a cloud service market, to move towards a computing model that aggregates resources from multiple cloud providers.

Cloud federation, which is enabled through standard interfaces, protocols, and data formats, is considered as a way to address such problems with resource sharing. A cloud federation can be defined as voluntary arrangements among a number of cloud providers with an objective to interconnect their cloud infrastructures and enable resource sharing [5]. With mutual sharing of resources, which allows resource aggregation, increased resource utilization, elastic service provisioning, and reliable multi-site deployment cloud providers can overcome service limitations, improve cost-efficiency, ensure QoS guarantees, and maintain the availability of cloud services.

An important implementation challenge to be addressed by cloud federation is the issue of efficient resource allocation on heterogeneous resources. For any application that is to be deployed on the federated cloud, a deployment plan, which maps the application's virtual machine requirements to providers' resources, is required. Such a deployment plan should identify a combination of provider resources that can offer the best deal. For a deal to be the best, the cost is an important factor but not the only one. Factors like application performance, response time, privacy, and security are also important. Furthermore, the best deal could be the one that integrates resources from multiple providers rather than a single one. Therefore, a more advanced federation broker is required to identify the best plan for application deployment by considering all these factors.

The challenge is further increased by the heterogeneity in application requirements and optimization objectives. Some applications (and application provider for that matter) may consider one parameter to be more important than others, while other applications consider different parameters most important. Such parameters are cost, QoS, reliability, performance, etc. The application deployment plan needs to be set by considering these parameters accordingly. This makes the deployment plan for each application a unique optimization problem.

In this context, a federation broker as the one of BASMATI, which is an integrated brokerage platform targeting federated clouds that supports the dynamic needs of mobile applications and users, requires such an efficient decision-making algorithm for determining an optimal resource combination from the members of the cloud federation. The

978-1-5386-5896-3/18/$31.00 ©2018 IEEE

algorithm should be able to tune each application's unique optimization objectives. Besides, to address dynamic application environment and user behaviors, it should work dynamically throughout the application lifecycle rather than only during initial deployment.

The majority of prior works on optimized VM placement is based on single objective optimization, mostly focusing on either cost minimization or QoS maximization or energy consumption minimization [6][7][8][9]. Some multi-objective optimization algorithms [10][11], too, are unable to meet the specific needs of BASMATI, which require the algorithm to consider the computing resource strength of VM instances, the application footprint, and the distance of VM instances. To the best of our knowledge, no existing research addresses each application's unique optimization requirement as in BASMATI.

In detail, in this paper, we address the problem of generating an application deployment plan for the BASMATI federation platform, which can satisfy unique application requirements by identifying an optimum balance between cost, performance, and response time. Our contributions are:

- We provide a description of the BASMATI architecture and investigate the resource allocation problem by considering cost, computing resources (CPU and memory), application footprint, and inter-node distance.
- We formulate a multi-objective optimization problem for generating application deployment plans by applying an artificial intelligence (AI) method (i.e., a genetic algorithm based optimization algorithm.
- We perform simulations to evaluate and demonstrate the effectiveness of the proposed algorithm compared to the benchmark (i.e., a random search algorithm).

The remainder of the article is organized as follows. Section II of the paper contains related works. A description of BASMATI architecture is provided in Section III. Section IV contains details on the optimization algorithm. Section V includes details on the simulation setup and the results analysis. Finally, the conclusion is given in Section VI.

II. Related Works

Resource allocation has been an active area of cloud computing research [9][12][13]. Various studies have been carried out with a focus on finding optimal ways of resource allocation in a single cloud as well as in a cloud federation environment.

An algorithm based on stochastic integer programming has been given by Chaisiri et al., which aims at reducing the cost of hosting in a multi-cloud environment [6]. Hadji et al. use Gomory-Hu trees for optimal VM placement and focus on reducing hosting and network connectivity cost [7]. Another algorithm optimizing a single objective (i.e., cost) is proposed by Simarro et al. This algorithm minimizes cost by dynamically placing VMs according to the pricing scheme for VM instances [8]. These algorithms solve only single objective optimization problems and do not meet the multi-objective optimization need of BASMATI.

Wang et. al. propose an algorithm that considers multiple objectives (i.e., energy and QoS) [10]. The algorithm proposed by Kumrai et al. considers response time, profit, and energy [11]. These works relate to our research to the extent that they also consider multiple optimization objectives. However, unlike the approach of Kumrai et al., which uses particle swarm optimization, we apply a genetic algorithm for solving the multiobjective optimization problem. We also consider the application footprint and the inter-node distance, which are an essential requirement of BASMATI.

Wu et. al apply an artificial intelligence method for solving the optimization problem, and their algorithm is also based on a genetic algorithm [14]. The algorithm minimizes energy consumption by servers and the communication network in a data center. Unlike this algorithm, we solve the problem for a federated cloud with multiple objectives rather than a single cloud environment with a single optimization objective.

III. Architecture

A. BASMATI Architecture

BASMATI is a EU Horizon 2020 project that developed a platform for cloud federation. The platform handles changing conditions in the environment of the user, the application, and the system. A simplified layered architecture of the BASMATI platform is presented in Fig. 1 (for details refer to Altmann et al.[15]). This three-layered architecture includes: Cloud Provider Management Layer at the bottom to provide management functions for edge and cloud providers; Federation Management Layer in the middle to deal with federation functions for cloud providers; and Application Management Layer providing management functions to the applications.

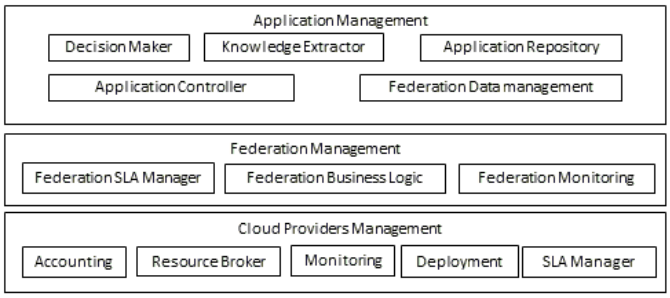

Fig. 1. Simplified BASMATI architecture

1) Cloud Provider Management Layer: It includes components that enable the secure use of provider resources. Major components at this layer include accounting, deployment, monitoring, resource brokering, and SLA management. The Accounting component maintains records of the user and the use of providers' resources for charging and payment purposes. The Deployment component is responsible for accessing the services of cloud providers through exposed APIs. The collection of monitoring data, filtering it, aggregating and providing it to the federation management layer is performed by the Monitoring component. The SLA Manager component maintains SLAs that have been agreed between a cloud provider and the cloud federation. The

Resource Broker module, based on heuristics, selects provider resources that can fulfill application requirements.

2) Federation Management Layer: This layer is responsible for handling members of the federation. It constitutes three modules: Federation Business Logic, Federation SLA Manager, and Federation Monitoring. The Federation Business Logic enables accounting of federation resource uses and specifies revenue sharing schemes among federation members [16]. Federation Monitoring collects information about application performance and resource utilization across clouds. The Federation SLA Manager deals with federation level SLAs between members of the federation and between the application and the cloud federation [17].

3) Application Management Layer: The Application Management Layer is responsible for all aspects of application management relating to the efficient allocation of resources from cloud providers to BASMATI applications. It consists of 5 modules: Federation Data Management, Knowledge Extractor, Decision Maker, Application Repository, and Application Controller.

B. BASMATI Decision Maker

The main role of Decision Maker is to decide on the most optimal allocation of resources (i.e., the most optimal mapping of VMs to provider resources (VM instances)) to an application running on the BASMATI platform. The notion of optimal is different to different applications and is based on how the application provider or the federation operator gives relative importance to different factors (e.g., cost, performance, application footprint, and latency). Maximization or minimization of these factors, while deciding on the optimal resource allocation decision making process, is termed as the objective function. Often, the decision requires consideration of more than one objective functions and, hence, the function of the Decision Maker is a multi-objective optimization. The input for this multi-objective optimization comes from other modules of the BASMATI architecture.

1) Interaction Description: For the optimization, the Decision Maker makes use of information from and interacts with the Application Controller, the Application Repository, the Resource Broker, and the Knowledge Extractor (Fig. 2)

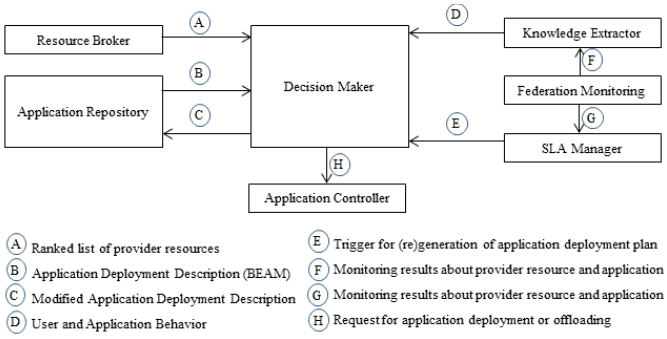

Fig. 2. Interaction of Decision Maker with other BASMATI components.

a) Resource Broker: The role of Resource Broker is to identify potential provider resources (VM instances) for the deployment or re-deployment of applications from among all available resources. The Resource Broker gets periodically in contact with the Monitoring component, which installed benchmarking algorithms on cloud provider servers [18]. The Resource Broker gets these benchmarking results from the Monitoring component. Based on these results, the Resource Broker provides on request a ranked resource list, according to which the application could be deployed most optimal, to the Decision Maker.

b) Application Repository: The role of Application Repository is to maintain application deployment description for the BASMATI federation platform. The deployment description is written in the Basmati Enhanced Application Modeling (BEAM) format. The BEAM specification is based on Topology and Orchestration Specification for Cloud Applications (TOSCA), an international standard that enables portability and operational management of cloud applications and services throughout their lifecycle [19]. BEAM provides a simplification of TOSCA suitable for the management and deployment of Mobile Applications in a Federated Cloud Scenario. The Decision Maker requests a BEAM file from the Application Repository and reads the deployment description for an application. Important parameters in the application description (BEAM file) that are of interest to the Decision Maker include the requirement specification (e.g., number of CPUs, memory capacity in GB) for all nodes and information about the existence or non-existence of inter-node relationships for data communication. After performing the optimization process, the Decision Maker modifies the deployment description (BEAM file) with an update about which VM should be deployed to which provider resource (VM instance). Then, the new BEAM file is submitted to the Application Repository. This can, then, be used by the Application Controller for executing the application deployment.

c) Knowledge Extractor: The Knowledge Extractor provides on request a prediction of the number of application users at various Points of Interest (POI) to the Decision Maker. This information forms an important factor for the optimization by the Decision Maker. The Knowledge Extractor predicts these values based on prediction meta-models implementing a combination of regression, deep learning, and decision trees methods.

d) Federation Monitoring: The federation monitoring continuously gathers information with respect to application performance and resource utilization. It works in close coordination with the monitoring component operating at the management layer of cloud provider's data center to get information about the resource utilization. It also coordinates with the SLA Manager to get information about the application performance. The result generated by this module is processed by the Knowledge Extractor and the Federation SLA Manager to provide feedback to the Decision Maker for optimized decision making.

e) Federation SLA Manager: Within the context of he Decision Maker's job, the Federation SLA Manager performs two actions. First, when an SLA is established with an application provider, it initiates the Decision Maker for generating the application deployment plan. Second, during

the life cycle of an application, it triggers the Decision Maker for the regeneration of an optimized application re-deployment in case of a SLA violation. To detect SLA violations, it receives input from the Federation Monitoring.

f) Application Controller: The Application Controller performs the task of managing the life cycle of an application. Three important sub-tasks are: i) initial deployment, where it takes the deployment plan from the Decision Maker and performs the actual deployment; ii) application migration in events of SLA violations. For this, it uses an alternative re-deployment plan from the Decision Maker and migrates the application VM as per the plan; and iii) stopping an application at the end of the application lifecycle.

2) Roles division among Decision Maker, Resource Broker, and Knowledge Extractor: The VM placement decision-making job mainly revolves around three components: Decision Maker, Resource Broker, and Knowledge Extractor. This distinction is one of the major features of the resource allocation solution of the BASMATI federation architecture. The devision allows for restricting the available resources for consideration by the Decision Maker. It improves the optimization process, as it reduces the length of the design variable and, therefore, reaches a solution faster. The role of the Resource Broker is vital in filtering the resources of cloud providers based on rating and providing only those resources that are valuable to the Decision Maker's optimization problem. The Knowledge Extractor provides input to the Decision Maker that are relevant for finding an optimal solution for the application at hand.

3) Optimization Process: The optimization process encompasses an application deployment scenario and an application lifecycle management scenario. With respect to the work of the Decision Maker, the only difference between these scenarios is how the Federation SLA Manager initiates the process. In the former scenario, the Federation SLA Manager triggers the process after a SLA has been *e*stablished, while in the latter scenario, the Federation SLA Manager triggers the process after a SLA violation has been detected. Otherwise, the optimization process remains the same. The Decision Maker gets the application description from Application Repository, the application and user behavior from the Knowledge Extractor, and available resources from the Resource Broker. The Decision Maker, then, executes its optimization algorithm, to develop the optimal plan for the deployment based on those inputs. Then, it updates the application deployment description (BEAM file) with the new plan and stores it in the Application Repository. Finally, the Decision Maker triggers the Application Controller for the availability of new application deployment plans.

IV. Optimization Algorithm

A. Multi-objective Optimization

Multi-objective optimization (MOO) is the process of simultaneously and systematically optimizing a collection of objective functions [20]. The multi-objective optimization problem for the BASMATI Decision Maker can be expressed as:

$$\mathrm{F}(x) = [F_1(x), F_2(x), \dots, F_k(x)]$$

$$\text{Subject to } g_j(x) \le 0, \quad j = 1,2,\dots,m, \text{ and}$$
$$h_l(x) = 0, \quad l = 1,2,\dots,e,$$

where $x \in E^n$ is a vector of resources x_i, and n is the number of independent resources x_i. $F(x)$ is a vector of objective functions like cost, latency, and time. $F_f(x)$ is an objective function, and k is the number of objective functions. The function $g_j(x)$ is an inequality constraint, m is the number of inequality constraints, $h_l(x)$ is an equality constraint, and e is the number of equality constraints.

B. Optimization Objectives

The objective of the BASMATI Decision Maker is to find the optimal combination of provider resources for the placement of VMs. This decision is guided by multiple objectives, and the solution is found by solving the problem as a multi-objective problem. TABLE I provides a list of objectives that are considered.

TABLE I. List of Optimization Objectives Considered

Objectives	Description
Cost Minimization	Choose provider resources such that the user or application requirements are fulfilled at minimal cost.
User support Maximization	Choose VM instances with a high number of CPUs and a high memory capacity so that it can support potentially more users .
Latency Minimization	Choose a mix of provider resources such that it minimizes the inter-node distance and, hence, latency.
User footprint Maximization	Choose instances such that they are located closer to the Point of Interests (POI) where the number of users is high.

C. Input Parameters

The Decision Maker requires inputs from other components to solve the optimization problem. These inputs are related to provider resources (instances), data centers, application, and user footprint. Inputs related to provider resources are received from the Resource Broker, and they include datacenter id, instance type (defined through memory size, number of CPUs, storage capacity), the available number of instances, cost per hour, and rating. Similarly, application related inputs are obtained from the Application Repository, and they include application node details (i.e., node id, number of CPUs, memory size, storage capacity) and node dependency. Data center related inputs are also received from the Resource Broker, and they include data center id, geo-location (latitude, longitude), owner, and time zone of the data center. Input related to the user footprint is received from the Knowledge Extractor, and they include *geo-location (latitude, longitude)* of the Point of Interest (POI) and the predicted number of users in that location.

D. Algorithm

Our algorithm for optimization is based on the concept of Genetic Algorithms (GA). A GA is an evolutionary algorithm that mimics the process of the natural evolution. With each generation (i.e., iteration), the solution approaches towards a more optimal (desired) solution. A GA can also be adapted to

solve an optimization problem that consists of multiple objectives.

Our GA algorithm, which produces as output the application deployment plan describing the mapping of virtual machines to instances, is specified as follows:

Step 0: Read inputs for application requirements, available resources, user prediction, and weights for all objectives.

Step 1: Generate a population of 1000 individuals at random that represents different deployment plans (see subsection 1 below).

For i = 1 to GENERATION_MAX

Step 2: Evaluate the fitness of each individual based on their relative values of cost, CPU numbers, memory size, inter-node distance, POIs to node distance, and the predicted number of users at each POI (see subsection 2 below).

Step 3: Select individuals based on their fitness values using the tournament selection method.

Step 4: Perform crossover and mutation of the selected individuals and produce offsprings (i.e., new population of solutions).

End of For Loop

Step 5: Choose the individual with the best fitness, presenting the best deployment plan.

Step 6: Generate the application deployment plan based on this selected individual in the required BEAM format.

1) Design of Solution (Individual) Variable and Population Generation: Fig. 3 shows an example of our formulation of the solution variable and the mapping of the virtual machines to instances. There are 3 cloud service providers having altogether 10 instance types. The application, which needs to be deployed in the BASMATI ecosystem, requires four VM nodes with varying specifications.

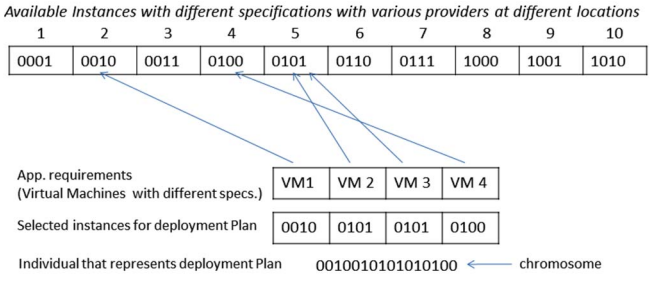

Fig. 3. Formulation of solution variable for genetic algorithm.

In order to design the chromosomes, we index all the available instances and represent the instances as their binary conversions. Binary representations should be padded with the required numbers of 'Zeros', to make them uniform. Now individual chromosomes are generated by making a string of randomly generated 0s and 1s. The string length is made equal to the number of characters required for binary representation of an instance multiplied by the number of VM nodes required. We repeat this process a number of times to generate desired population size (i.e., the number of individuals).

2) Evaluation of Fitness of Individuals (Solutions): As the optimization problem to be solved by the BASMATI Decision Maker has multiple objectives, the GA is adapted to include multiple fitness functions. For this, we adopted the scalarization method, which is a common way for converting a multi-objective problem to a single objective optimization problem [20]. The optimal solutions to the single-objective optimization problem is a Pareto optimal solution to the multi-objective optimization problem [21]. In detail, the scalarized fitness function for the multi-objective optimization problem of the BASMATI Decision Maker is expressed as:

$$F(i) = \alpha f_c(i) + \beta f_p(i) + \gamma f_m(i) + \delta f_u(i) + \Omega f_n(i) \qquad (1)$$

where $F(i)$ is the overall fitness of the i^{th} individual. $f_c(i)$, $f_p(i)$, $f_m(i)$, $f_u(i)$, and $f_n(i)$ are individual fitness values for cost, CPU, memory, user-node latency, and inter-node latency, respectively. Similarly, α, β, γ, δ, and Ω are weights for the importance of cost, CPU, memory, user-node latency, and inter-node latency, respectively.

a) Cost Fitness is evaluated as a relative function, and its normalized form is expressed as:

$$f_c(i) = \frac{\sum_{j=1}^{m} cost_i^j - min(cost)}{max(cost) - min(cost)} \qquad (2)$$

where $f_c(i)$ is the cost fitness function. The cost of j^{th} virtual machine instance of the i^{th} solution is denoted as $cost_i^j$, and m is the total number of required VMs, while $min(costs)$ and $max(costs)$ are the cost of the cheapest and the most expensive solutions among all individuals in the population, respectively.

b) CPU Fitness in its normalized form is expressed as follows:

$$f_p(i) = \frac{\sum_{j=1}^{m} cpu_i^j - min(cpu)}{max(cpu) - min(cpu)} \qquad (3)$$

where $f_p(i)$ is the CPU fitness function. The variable cpu_i^j represents the number of CPUs specified for the j^{th} VM instance of the i^{th} solution, and m is the total number of required VMs. The functions $min(cpu)$ and $max(cpu)$ are the number of CPU counts of those individuals having the lowest and the highest number of CPU counts among all individuals in the population.

As the equation for the evaluation of *Memory Fitness* takes the same form, we do not show it here.

c) User-Node Distance Fitness is sum of the product of the distances from the instance location (i.e., data center of the data center) to the POI and the number of users at the POI across each POI and VM instance. Its normalized value is:

$$f_u(i) = \frac{\sum_{p=1}^{q} \sum_{j=1}^{m} un\_dis(p,j)_i * users_p - min(un\_dis * users)}{max(un\_dix * users) - min(un\_dis * users)} \qquad (4)$$

where $f_u(i)$ is the fitness function for user-node distance. $un\_dis(p,j)_i$ is the geodesic distance between POI p and VM instance j for the i^{th} solution. $users_p$ denotes the number of predicted users at POI p. Similarly, $min(un\_dis * users)$ and

$\max(un\_dis * users)$ are the sum of products of user-node distance and number of users across all POIs and VMs for the solution with the lowest and highest values, respectively.

d) Inter-node Distance Fitness is calculated on the basis of *the* distance between any pair of nodes that have a dependency in terms of data communication. Its normalized value is calculated as:

$$f_u(i) = \frac{\sum_{j=1}^{m-1}\sum_{k=j+1}^{m} nn\_dis(j,k)_i - min(nn\_dis)}{max(nn\_dis) - min(nn\_dis)} \quad (5)$$

where $f_n(i)$ is the inter-node fitness function. $nn\_dis(j,k)_i$ is the geodesic distance between a pair of nodes j and k in the i^{th} solution with a data communication dependency. Furthermore, $min(nn\_dis)$ and $max(nn\_dis)$ are the sum of geodesic distances between pairs of nodes with a data communication dependency for the solutions with the lowest and highest values, respectively.

V. Simulation

A. Simulation Setup

We implemented our algorithm based on GA in Python. To evaluate the effectiveness of our algorithm and to obtain insight in AI-based resource allocation, we carried out multiple simulation runs. The parameter settings used for the simulation runs vary but follow the following principles.

In detail, we considered three different virtual machine types (TABLE II) and two inter-node dependencies for data communication between virtual machines (TABLE III).

TABLE II. Specification of Three Virtual Machine Types

Webserver1		Webserver2		DBServer1	
CPU core	Memory	CPU core	Memory	CPU core	Memory
2	8G	2	8G	4	64G

TABLE III. Inter-node Relationship of Application

Node 1	Node 2
Webserver1	DBServer1
Webserver2	DBServer1

We considered 163 different instance types with specification and price values based on Amazon's on-demand EC2 pricing [22]. Samples of which are shown in TABLE IV. Furthermore, random geolocation codes have been added, as they are needed for our algorithm.

TABLE IV. Important Specifications of Sample Instances

Instance ID	CPU Core	Memory [GB]	Latitude	Longitude	Cost [$/hr]
t2.micro	1	1	38.89	-77.03	0.0116
t2.medium	2	4	51.50	-0.12	0.0464
t2.2xlarge	8	32	37.43	98.13	0.3712

TABLE V. Sample of Application Footprint Data

POI ID	Latitude	Longitude	Predicted Users
Poi1	-27.46794	153.02809	5678
Poi2	26.06139	119.30611	9765

Finally, we considered 150 POIs with a random number of users at each POIs. Sample data are shown in TABLE V.

B. Simulation Result

For each set of 100 simulation runs, we considered the same application requirements for VMs, and the analysis results are based on the mean of the results obtained for each set of simulation runs. In detail, the analysis has been performed to: i) evaluate the stability of the algorithm through a convergence test; ii) evaluate the effectiveness of the proposed algorithm through a comparative analysis with a benchmark solution; iii) evaluate the robustness of our algorithm for varying optimization requirements of different applications through tests with ranges of weights for all optimization parameters; and iv) evaluate the optimality of the solution.

1) Convergence of solution: In order to evaluate the effectiveness of the proposed algorithm, we ran multiple simulation experiments by assigning full weight values to one parameter at a time. The results show how all of these parameter values start at one point and tend to be closer to that of the final solution as the generations progress and, finally, converges to the final solution well before the 150th generation (TABLE VI).

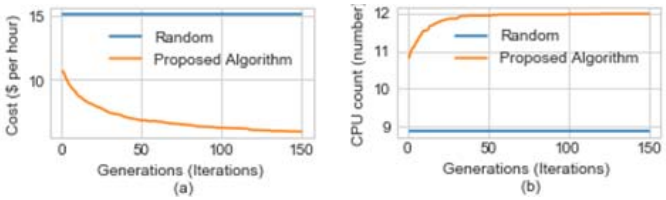

Fig. 4. Convergence of values of parameters in objective functions.

In Fig. 4(a), we can observe that the cost (in $/hr) starts with 10.65, then, decreases rapidly to 8.84 until the 10th generation, and further decreases slowly to 5.92 until the 145th generation (iteration). At that point, the value does not change and becomes the final solution. A similar behavior can be observed in case of the user-node distance and inter-node distance parameters (TABLE VI). These parameters (i.e., cost, user-node distance, and inter-node distance) are of objective type *minimization* and, hence, desire low values. As depicted in Fig. 4(a), our algorithm generates a desirable downward sloping curve.

TABLE VI. Convergence of Values of Parameters and Comparison With Random Solution

	Cost [$/hr]	CPU Core	Mem. [GB]	User* Distance [million]	Inter-Node Distance
Generation 1	10.65	10.82	45.48	82.51	7.531
Generation 10	8.84	11.54	46.52	78.56	5.857
Generation 150	5.92	11.99	47.84	69.68	0.00
Generation, at which the solution converges	145	120	84	143	125

The other two parameters - CPU and memory are of objective type *maximization*. The solution that provides a high value is desirable with respect to performance and concurrent user support. As shown in Fig. 4(b), the curve for CPU starts at a low value and, then, inclines. It starts at 10.82 in the first

generation, then, increases rapidly to 11.56 until the 10th generation, and further increases slowly to 12.0 in the 120th generation, and remains constant thereafter. A similar behavior can be observed for the memory parameter as well, which converges to the final solution after the 84th generation (TABLE VI). The curves for none of the parameters show sign of fluctuations across the mean values, as it proceeds through the generations. Thus, our algorithm provides a stable solution.

2) Comparison with a benchmark solution: A comparison of the values obtained with the proposed solution and a benchmark solution (i.e., the solution generated by a random selection algorithm) is given in Fig. 5. In detail, the comparison is carried out for three different weight scenarios: i) a weight value of 0 to the parameter in foci and equal weights of 0.25 to the remaining parameters; ii) equal weight of 0.2 to all five parameters; and iii) a weight value of 1.0 to parameter in foci and 0 to all remaining parameters.

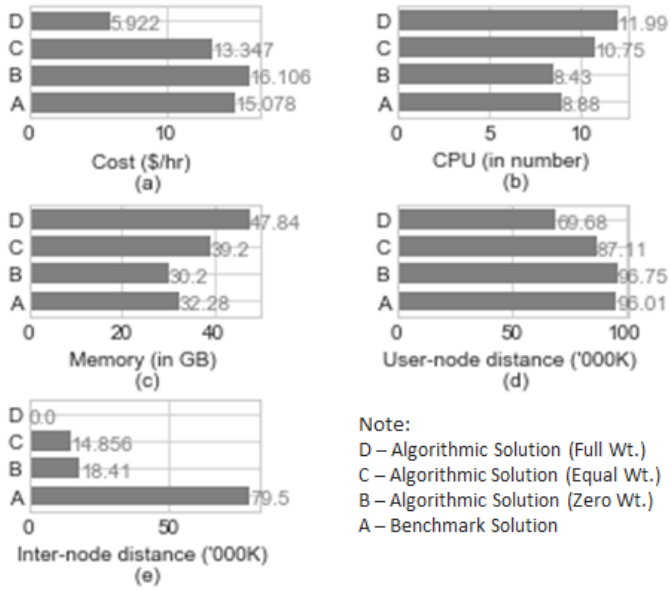

Fig. 5. Comparison of parameter values achieved through different methods.

As shown in Fig. 5(a), in the case of assigning zero weight to cost, the cost value for the proposed algorithmic solution is 16.10, which is higher than the benchmark solution. However, in the second case of assigning the weight of 0.2 to cost (equal weight scenario), the cost decreases to 13.347, which is already 11.3% less than the 15.078 of the benchmark solution. Furthermore, in the case of assigning the full weight of 1.0 to cost, the cost value decreases to 5.92, providing a cost reduction of 60% compared to the benchmark solution.

Similarly to the cost parameter, for the user-node distance parameter, we can observe a reduction by 9.26% in the equal weight scenario and by 27.42% in the full weight scenario compared to the benchmark solution (Fig. 5(d)). For the inter-node distance parameter, we can also observe a reduction by 79.5% and 100% in case of the equal weight scenario and the full weight scenario, respectively (Fig. 5(e)).

For the parameters that are maximized, CPU and memory, the observation is also positive. For the equal weight scenario and the full weight scenario, the values for CPU are 10.75 and 11.99, respectively. Both values are higher than the value of the benchmark solution, which is 8.88. For the memory parameter, the values are 39.2 and 47.84 for the equal weight scenario and the full weight scenario, respectively. It is higher than the 32.28 of the benchmark solution.

3) Evaluation of robustness of algorithm: To evaluate the robustness of the algorithm for different optimization objectives of the application, we tested the algorithm for a range of weight values for the optimization parameters in the fitness function. TABLE VII shows the results for three weights.

TABLE VII. PARAMETER VALUES FOR VARIOUS WEIGHTS

Optimization Parameter	Weights		
	Zero (0.0)	Equal (0.2)	Full (1.0)
Cost [$/hr]	16.10	13.34	5.92
CPU	8.43	10.75	11.99
Memory [GB]	30.20	39.20	47.84
User-node distance [million]	96.75	87.11	69.68
Inter-node distance	14.85	18.41	0.00

From TABLE VII, we can observe that, with appropriate weights, we can generate various deployment plans with a cost ranging between 5.96 (lowest cost) and 16.10, with the CPU number between 8.43 and 11.99 (highest number), with a memory size between 30.2GB and 47.84GB, with a user-node distance (multiplied by users at at POIs) between 69680000 and 96750000, and, finally, with an inter-node distance between 0 and 14.85. This demonstrates that, by adjusting the weights appropriately, we can obtain the desired tradeoff among different parameters of the optimization objective.

4) Evaluation of optimality of solution: Our algorithm uses the scalarization method and converts the multi-objective optimization problem to a single objective problem. Consequently, according to theory, an optimal solution to the single objective problem becomes the Pareto optimal solution to the multiple objective optimization problem. Therefore, in order to evaluate the effectiveness of the proposed algorithm, not only a comparison with the benchmark solution (random search algorithm) is necessary but also a test whether the solution provided by the proposed algorithm is Pareto optimal.

For this test, we took a solution (deployment plan) generated by our multi-objective optimization algorithm with an equal weight to all optimization parameters. We evaluated the values of cost, the number of CPUs, the memory size, the user-node distance, and the inter-node distance corresponding to this solution. Then, with respect to each VM instance in the solution, we identified from among all available VM instances, those VM instances whose specifications meet the application requirement but have costs lower than that of the VM instance obtained with the optimization algorithm. Then, we replaced the VM instance in focus with the one identified with a lower cost, and re-evaluate the values of all other parameters for the solution formed after replacement.

Following this process, we found no instances that have a lower cost than that of the calculated solution, while having no higher or equal values of CPU number and memory size and no lower or equal values of user-node distance and inter-node

distance. We repeated the process for all other parameters and found the same results. Based on these results, we can state that we could not replace the VM instance in the solution of the optimization algorithm with a VM instance that can provide a better deal for one of the optimization parameters without negatively affecting the deal for the other parameters. To the extent of the scenarios considered in this simulation, the solutions found with our algorithm are Pareto optimal.

VI. Conclusion

With wider and increasing acceptance of cloud computing as promising computing model, the number of cloud providers entering the market increases. This is expected to even accelerate with the introduction of flog computing and edge computing, which will allow small providers to over their services on the Internet. As this requires the collaboration with other providers, this will only be possible through cloud federations. Cloud federation will allow capacity pooling of resources, improved resource utilization, and combining heterogeneous resources.

However, one of the challenges for the sustained and effective operation of a cloud federation is the optimized use of resources that meet multiple objectives of application providers and the federation operator. The facts that every application is unique in terms of optimization objectives and the heterogeneity of resources is large, makes the process of optimal resource allocation to the application complicated.

Within this article, we presented a resource allocation algorithm that is based on the genetic algorithm concept and that is part of the optimization process of the BASMATI federation architecture. With subsequent iteration, our proposed algorithm provides a more optimal solution for identifying matches of cloud provider resources to the application. We demonstrated that our proposed algorithm is effective in finding an optimal match of cloud provider resources based on the value of relative importance assigned to each of the optimization objectives and can be tuned to specific application needs. We also demonstrated that the solutions generated by the algorithm are Pareto optimal. Our future work will include a provider resource ranking model and an application user prediction model.

Acknowledgment

The research was conducted within the EU-Korea project BASMATI (Cloud Brokerage Across Borders for Mobile Users and Applications), which was supported from the ICT R&D program of the Korean MSIT/IITP [R0115-16-0001].

References

[1] K. Jeferry *et al.*, "Challenges emerging from future cloud application scenarios," *Procedia Comput. Sci.*, vol. 68, pp. 227–237, 2015.

[2] J. Altmann, C. Courcoubetis, and M. Risch, "A marketplace and its market mechanism for trading commoditized computing resources," *Ann. Telecommun. des télécommunications*, vol. 65, no. 11–12, pp. 653–667, 2010.

[3] N. Haile and J. Altmann, "Value creation in software service platforms," *Futur. Gener. Comput. Syst.*, vol. 55, pp. 495–509, 2016.

[4] K. Kim, S. Kang, and J. Altmann, "Cloud Goliath versus a federation of cloud Davids," in *International Conference on Grid Economics and Business Models*, 2014, pp. 55–66.

[5] A. J. Ferrer *et al.*, "OPTIMIS: A holistic approach to cloud service provisioning," *Futur. Gener. Comput. Syst.*, vol. 28, no. 1, pp. 66–77, 2012.

[6] S. Chaisiri, B.-S. Lee, and D. Niyato, "Optimal virtual machine placement across multiple cloud providers," in *Services Computing Conference, 2009. APSCC 2009. IEEE Asia-Pacific*, 2009, pp. 103–110.

[7] M. Hadji, B. Aupetit, and D. Zeghlache, "Cost-Efficient Algorithms for Critical Resource Allocation in Cloud Federations," in *Cloud Networking (Cloudnet), 2016 5th IEEE International Conference on*, 2016, pp. 1–6.

[8] J. L. L. Simarro, R. Moreno-Vozmediano, R. S. Montero, and I. M. Llorente, "Dynamic placement of virtual machines for cost optimization in multi-cloud environments," in *High Performance Computing and Simulation (HPCS), 2011 International Conference on*, 2011, pp. 1–7.

[9] J. Altmann and M. M. Kashef, "Cost model based service placement in federated hybrid clouds," *Futur. Gener. Comput. Syst.*, vol. 41, pp. 79–90, 2014.

[10] S.-H. Wang, P. P.-W. Huang, C. H.-P. Wen, and L.-C. Wang, "EQVMP: Energy-efficient and QoS-aware virtual machine placement for software defined datacenter networks," in *Information Networking (ICOIN), 2014 International Conference on*, 2014, pp. 220–225.

[11] T. Kumrai, K. Ota, M. Dong, J. Kishigami, and D. K. Sung, "Multiobjective Optimization in Cloud Brokering Systems for Connected Internet of Things," *IEEE Internet Things J.*, vol. 4, no. 2, pp. 404–413, 2017.

[12] I. Breskovic, J. Altmann, and I. Brandic, "Creating standardized products for electronic markets," *Futur. Gener. Comput. Syst.*, vol. 29, no. 4, pp. 1000–1011, 2013.

[13] J. Altmann, M. Hovestadt, and O. Kao, "Business support service platform for providers in open cloud computing markets," in *Networked Computing (INC), 2011 The 7th International Conference on*, 2011, pp. 149–154.

[14] G. Wu, M. Tang, Y. C. Tian, and W. Li, "Energy-efficient virtual machine placement in data centers by genetic algorithm," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2012, vol. 7665 LNCS, no. PART 3, pp. 315–323.

[15] J. Altmann *et al.*, "BASMATI: An Architecture for Managing Cloud and Edge Resources for Mobile Users," in *International Conference on the Economics of Grids, Clouds, Systems, and Services*, 2017, pp. 56–66.

[16] R. G. Aryal and J. Altmann, "Fairness in Revenue Sharing for Stable Cloud Federations," in *International Conference on the Economics of Grids, Clouds, Systems, and Services*, 2017, pp. 219–232.

[17] M. Risch and J. Altmann, "Enabling open cloud markets through WS-agreement extensions," in *Grids and Service-Oriented Architectures for Service Level Agreements*, Springer, 2010, pp. 105–117.

[18] G. Z. Santoso *et al.*, "Dynamic Resource Selection in Cloud Service Broker," in *High Performance Computing & Simulation (HPCS), 2017 International Conference on*, 2017, pp. 233–235.

[19] OASIS, "OASIS Topology and Orchestration Specification for Cloud Applications (TOSCA) TC," 2017. [Online]. Available: https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=tosca.

[20] R. T. Marler and J. S. Arora, "Survey of multi-objective optimization methods for engineering," *Struct. Multidiscip. Optim.*, vol. 26, no. 6, pp. 369–395, 2004.

[21] C.-L. Hwang and A. S. M. Masud, *Multiple objective decision making—methods and applications: a state-of-the-art survey*, vol. 164. Springer Science & Business Media, 2012.

[22] "http://aws.amazon.com/ec2/pricing/." [Online]. Available: http://aws.amazon.com/ec2/pricing/.