UDC 004.94

COST OPTIMIZATION METHOD FOR INFORMATIONAL INFRASTRUCTURE DEPLOYMENT IN STATIC MULTI-CLOUD ENVIRONMENT

Rolik O. I. – Dr. Sc., Professor, Head of the Department of Information Systems and Technologies, National Technical University of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute”, Kyiv, Ukraine.

Zhevakin S. D. – Post-graduate student of the Department of Information Systems and Technologies, National Technical University of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute”, Kyiv, Ukraine.

ABSTRACT

Context. In recent years, the topic of deploying informational infrastructure in a multi-cloud environment has gained popularity. This is because a multi-cloud environment provides the ability to leverage the unique services of cloud providers without the need to deploy all infrastructure components inside them. Therefore, all available services across different cloud providers could be used to build up information infrastructure. Also, multi-cloud offers versatility in selecting different pricing policies for services across different cloud providers. However, as the number of available cloud service providers increases, the complexity of building a cost-optimized deployment plan for informational infrastructure also increases.

Objective. The purpose of this paper is to optimize the operating costs of information infrastructure while leveraging the service prices of multiple cloud service providers.

Method. This article presents a novel cost optimization method for informational infrastructure deployment in a static multi-cloud environment whose goal is to minimize the hourly cost of infrastructure utilization. A genetic algorithm was used to solve this problem. Different penalty functions for the genetic algorithm were considered. Also, a novel parameter optimization method is proposed for selecting the parameters of the penalty function.

Results. A series of experiments were conducted to compare the results of different penalty functions. The results demonstrated that the penalty function with the proposed parameter selection method, in comparison to other penalty functions, on average found the best solution that was 8.933% better and took 18.6% less time to find such a solution. These results showed that the proposed parameter selection method allows for efficient exploration of both feasible and infeasible regions.

Conclusion. A novel cost optimization method for informational infrastructure deployment in a static multi-cloud environment is proposed. However, despite the effectiveness of the proposed method, it can be further improved. In particular, it is necessary to consider the possibility of involving scalable instances for informational infrastructure deployment.

KEYWORDS: cost optimization, information infrastructure, initial placement, multi-cloud, parameters selection method, penalty function.

ABBREVIATIONS

VM is a virtual machine;

GA is a genetic algorithm;

AWS is an Amazon web services;

vCPU is a virtual central unit processor.

NOMENCLATURE

P is a set of cloud service providers;

G is a set of virtual machines with general-purpose specialization that are available in all cloud service providers P;

A is a set of application components which define information infrastructure;

S is a relation matrix ($A \times A$) of application components that must be deployed within the same cloud provider;

D is a relation matrix ($A \times A$) of application components that should be placed in different clouds;

R is a set of availability zones, that are available in all cloud providers P;

B_{ra} is a relationship matrix ($R \times A$) which defines the possibility of deploying application component $a \in A$ in availability zone $r \in R$.

C_{prg} is a hourly price of on-demand usage of a virtual machine at the provider $p \in P$, in the region $r \in R$ of class $g \in G$.

X_{aprg} is a binary decision variable equal to 1 when component $a \in A$ is placed at the provider $p \in P$, in the region $r \in R$, of the VM class $g \in G$, and 0 otherwise;

W_{pg} is a number of CPU cores of virtual machine type $g \in G$ from the cloud provider $p \in P$;

$W_a^{\min}$ is a minimum required number of CPU cores for application component $a \in A$;

E_{pg} is a amount of memory of virtual machine type $g \in G$ from the cloud provider $p \in P$;

$E_a^{\min}$ is a minimum required amount of memory for the application component $a \in A$;

s is a number of constraints that have been met;

m is a total number of constraints;

r_i is a penalty weight multiplier that inequity constraints impose when violated;

c_j is a penalty weight multiplier that equity constraints impose when violated.

INTRODUCTION

Cloud computing gained popularity in the last decade and continues to be relevant in our time [1]. Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., servers, storage, applications, services, and networks) that can be rapidly provisioned and released with minimal management effort

DOI 10.15588/1607-3274-2024-3-14

or service provider interaction [2]. The significant advantage of using the cloud provider's services is the ability to rapidly scale infrastructure components. This advantage is achieved due to the rapid provision of additional computing services for a certain period, which allows to overcome spikes in service demand. This makes cloud computing superior to on-premise solutions, as their computing resources cannot scale with the same ease and speed.

In turn, the combination of on-premise data centers and services of cloud providers is widely used in the industry for hosting information infrastructure. This approach is called a hybrid cloud. In such an approach, the services of cloud providers are often used to handle high demand, while the main computational resources are concentrated in on-premise data centers. However, while this approach solves the problem of limited scalability of on-premise data centers, it is not a long-term solution because on-premise data centers require constant support and updates.

According to statistics for 2023, about 92% of companies plan or already stick to a multi-cloud approach for service deployment [3, 4]. This statistic is primarily associated with the advantages provided by a multi-cloud in comparison to the use of a single-cloud provider. The biggest disadvantage of using the services of only one provider is the problem of vendor lock. Vendor lock refers to a situation where an organization becomes heavily dependent on a specific cloud service provider. This dependency restricts the ability to switch to another cloud provider without spending a substantial budget. The multi-cloud offers greater flexibility in service deployment, as all availability zones of selected cloud service providers can be used to deploy services. As shown in statistics [3], one of the main reasons for using a multi-cloud approach is cost optimization for service deployment and utilization. This is because multi-cloud provides access to different pricing policies for identical services among different cloud providers. For example, a virtual machine with 4 virtual processors and 16 gigabytes of RAM in the geographical area of Tokyo costs $0.2502 per hour of usage in AWS [5], compared to Google Cloud where a virtual machine with the same characteristics costs $0.2168 per hour of usage [6].

Despite the significant advantages provided by the multi-cloud, it also introduces additional complexity in building an information infrastructure deployment plan. Such complexity is associated with an increased set of possible options, across different providers. The dynamic changes in load and virtual machine price also add significant complexity during deployment plan construction. To overcome these problems, cloud service brokering mechanisms are often used [7]. In general, such mechanisms accept as input parameters the initial deployment plan of services and statistical data regarding load changes over a certain period. However, while the user can provide reliable information regarding the service placement of information infrastructure, collecting and presenting information about the dynamics of service load changes can be problematic. The main problem in providing such information lies in its collection. In order to collect data about the load, it is necessary to observe the operation of the services, which involves additional costs. Another problem is that by the time the data is transferred to the broker, it will already be out of date.

To address the identified problems, this study will propose a cost optimization method for information infrastructure deployment in a static multi-cloud environment. A static environment was chosen because the information infrastructure will be deployed to collect data on service loads and obtain patterns of their use. That is, it is assumed that the data collection period will be shorter than the period of updating prices for service usage by the cloud service provider. Information collected in this way will reflect the actual system needs. In the future, this information could be used by the broker for dynamic infrastructure component placement. Additionally, the resulting initial placement strategy can speed up the dynamic placement algorithm by using the provided initial deployment plan as a starting point in the algorithm's operation.

The object of study is the informational infrastructure deployment in a static multi-cloud environment. This involves analysing how different service pricing for comparable services from different cloud service providers can be utilized to minimize the cost of informational infrastructure operation.

The subject of study is the methods of combinatorial optimization for creating an information infrastructure deployment plan.

The purpose of the work is to develop a method which will create a deployment plan for informational infrastructure in a static multi-cloud environment. The resulting plan should consider the provided constraints on virtual machine parameters and placement strategy. The goal is to minimize the hourly operational cost of infrastructure utilization.

1 PROBLEM STATEMENT

Suppose that the hourly price for the use of virtual machines of class $g \in G$, across cloud providers $p \in P$, within availability zones $r \in R$ is static and set to C_{prg}. Also, provided a set of application components $a \in A$ that form informational infrastructure, which should be deployed. Then, the set of decision variables X_{aprg} should be found that will minimize the information infrastructure deployment cost as presented in (1–7).

$$\min \sum_{a\in A} \sum_{p\in P} \sum_{r\in R} \sum_{g\in G} C_{prg} X_{aprg}\,, \tag{1}$$

Subject to:

$$\sum_{p\in P} \sum_{r\in R} \sum_{g\in G} X_{aprg} W_{pg} \ge W_a^{\min}, \forall a \in A, \tag{2}$$

$$\sum_{p\in P} \sum_{r\in R} \sum_{g\in G} X_{aprg} E_{pg} \ge E_a^{\min}, \forall a \in A, \tag{3}$$

$$\sum_{p\in P}(\sum_{r\in R}\sum_{g\in G}X_{a_1prg}\times\sum_{r\in R}\sum_{g\in G}X_{a_2prg})\geq S_{a_1a_2}, \quad (4)$$

$$\forall a_1\neq a_2, a_1,a_2\in A,$$

$$\sum_{p\in P}(\sum_{r\in R}\sum_{g\in G}X_{a_1prg}\times\sum_{r\in R}\sum_{g\in G}X_{a_2prg})\leq D_{a_1a_2}, \quad (5)$$

$$\forall a_1\neq a_2, a_1,a_2\in A,$$

$$\sum_{p\in P}\sum_{g\in G}X_{aprg}\leq B_{ra},\quad \forall a\in A, \forall r\in R, \quad (6)$$

$$\sum_{p\in P}\sum_{r\in R}\sum_{g\in G}X_{aprg}=1,\quad \forall a\in A, \quad (7)$$

$$X_{arpg}, D_{a_1a_2}, S_{a_1a_2}, B_{ra}\in N_0,$$

$$C_{rpg}, W_a^{\min}, E_a^{\min}, E_{pg}, W_{pg}\in N.$$

Where (2) and (3) define the constraints regarding the virtual machine parameters on which the application component should be deployed. Where (2) describes the constraint for the minimal number of virtual cores and (3) describes constraints regarding the minimal amount of virtual memory. Constraints (4) specifies application components that must be deployed within the same cloud provider, while (5) specifies the components that must be deployed in different cloud providers. In (6) the constraints regarding deploying application components only in certain availability zones are described. Constraint (7) specifies that each application component must be deployed only in one instance on one of the available virtual machines. This limitation is introduced to simplify the gathering of necessary information.

2 REVIEW OF THE LITERATURE

The topic of virtual infrastructure deployment, or Virtual Machines (VMs) placement problem, is widely described in the literature. These problems are often classified as combinatorial problems. The main approach to solving these problems is mathematical optimization methods. In the literature, the VMs placement problem is reviewed from two perspectives: from the cloud service provider's perspective and from the cloud resource consumer's perspective.

The main goal of a cloud service provider is to maximize profit by utilizing existing resources. For instance, increasing the profit of the cloud provider in [8] is achieved by minimizing electricity consumption by the data centers. This is achieved by compactly rearranging VMs placement. Meantime the authors tried to find a balance in service level agreement change, simultaneously maximizing it. Additionally, cost optimization can be achieved through network traffic optimization by reducing the distance between servers, as demonstrated in [9].

On the other hand, the goal of cloud service consumers is to minimize the utilization costs of information infrastructure that is deployed on the cloud provider's services. In articles [7, 10–17], the main focus was concentrated on minimizing costs associated with operating the information infrastructure in the cloud. For example, in [10], the authors formulated the problem of virtual machine placement as a multi-objective optimization problem, the main goal of which was to minimize processing and memory resource usage. As a result, the authors proposed the VMPACS algorithm to solve this problem. The main goal of the proposed algorithm is to search the solution space more efficiently and obtain a Pareto set of solutions.

In recent years, the multi-cloud has increasingly attracted the attention of researchers. This trend is because multi-cloud provides the ability to use different pricing policies that are available among different cloud service providers. This leads to the expansion of search space in solving the problem of cost optimization of information infrastructure utilization in the cloud. Thus, in [11] the authors proposed an algorithm for the optimal placement of applications in a hybrid cloud. The described algorithms specialized in service-based applications (SBA) placement. The authors aimed to optimize the communication and hosting costs of SBA. By striking a balance between private and public cloud resources, the proposed algorithm aims to enhance the overall efficiency of hybrid cloud deployments. In [12], authors proposed an optimal virtual machine placement (OVMP) algorithm. The goal of the presented algorithm is to minimize the cost of hosting VMs in a multi-cloud environment with consideration of the uncertainty of demand and VMs prices. The problem in OVMP is described as a two-stage stochastic integer programming problem. In [13] a novel cloud brokering architecture was presented. This architecture provides a cost-optimized deployment plan for the placement of virtual resources in a multi-cloud environment. The main objective of the resulting deployment plan is to select the best cloud services with optimal cost, considering the value of the defined Service Measurement Index along with additional physical and logical constraints. The proposed cloud brokering architecture has been modeled using a mixed integer programming formulation. This formulation is solved using the Benders decomposition algorithm.

In many studies, the type of VMs placement is divided into static and dynamic. Dynamic placement involves optimization of the existing infrastructure placement plan through dynamic adjustments considering changes in service loads. Articles dedicated to dynamic placement operate with statistical data regarding changes in service prices, service loads, and other parameters. Dynamic placement algorithms are executed once in a certain period to respond to a change in load that occurred since the last algorithm invocation. These algorithms should be fast to make changes in VMs placement plan as quickly as possible to reduce the time gap when the number of available services mismatches the load. In [14], the authors proposed an optimal cloud resource provisioning (OCRP) algorithm to address the challenge of resource provisioning in cloud computing. The authors formulated a stochastic programming model to optimize the provisioning of computing resources. They have used both reservation and on-demand provisioning plans. The algorithm takes into account the uncertainty of demand and resource prices. The authors inspected different approaches to obtain the solution of the OCRP algorithm, such as deterministic equivalent formulation, sample-average approximation, and Bender's decomposition algorithm. In [7] authors addressed the Cloud Resource Management Problem in multi-cloud environments. The authors tried to reduce the cost and the execution time of consumer applications among Infrastructure as a Service services from multiple cloud providers. The authors used a Biased Random-Key Genetic Algorithm to solve the problem. This algorithm is based on a cloud brokerage mechanism and is designed to provide high-quality real-time solutions to automate the cloud resource management and deployment process.

Along with dynamic VMs placement algorithms, static VMs placement algorithms are appearing in the literature [15–17]. Unlike dynamic, static placement algorithms are invoked only once to determine the initial infrastructure placement plan. These algorithms are static due to neglecting the dynamic changes of various parameters such as load, service costs, service availabilities, etc. Such neglect occurs since the algorithm uses the available values at the time of invocation, without taking into account their subsequent change over time [15]. The purpose of the infrastructure deployed in such a way is to obtain metrics regarding its operation and interaction patterns. In [15], the authors proposed an architecture for a cloud broker that deploys VMs across multiple clouds. The goal of the broker was to minimize the total infrastructure cost by selecting the best cloud provider services based on the current conditions. The authors presented a binary integer programming formulation of the problem, which was then resolved using AMPL with the use of MINOS and CPLEX solvers. In [16] authors proposed a solution to optimize the placement of VMs across multiple cloud providers taking into account user-provided criteria. The developed algorithm was formulated as an integer programming problem. The authors tried to find a balance between price and performance tradeoffs for the resulting placement plan. Criteria provided by the user could steer the VM allocation by specifying the maximum budget along with minimum acceptable performance value along with constraints regarding load balance, hardware configuration of individual VMs, etc. In [17] authors formulated the problem of cost optimization in a multi-site multi-cloud environment. Optimization criteria were considered as the total price of infrastructure deployment which included the price of VMs reservation and the price of communication between them. To solve the problem authors formulated a greedy-based algorithm.

In reviewed articles, the VMs placement problem is carried out, almost without consideration of dependency between them. As some articles include data transmission cost into the total cost, it is not enough to describe dependency and communication between applications inside information infrastructure. Also, the described approaches consider that distinct applications in informational infrastructure should be placed within one virtual machine. Such a model does not reflect modern trends. Nowadays, with the development of architectural approaches to building informational infrastructure, the virtual machine is used to host application component rather than the whole application. Additionally, most of the reviewed articles ignore the possibility of deploying information infrastructure components in different availability zones. Therefore, they are using a multi-cloud approach only to expand the possible choice of VMs for placement. In contrast, this work will propose a method for information infrastructure deployment in a static multi-cloud environment, in which application components are placed on different VMs. Also, the possibility of deploying infrastructure components in different availability zones will be considered, as well as the constraints regarding service placement in certain availability zones that are associated with the legislation of different countries.

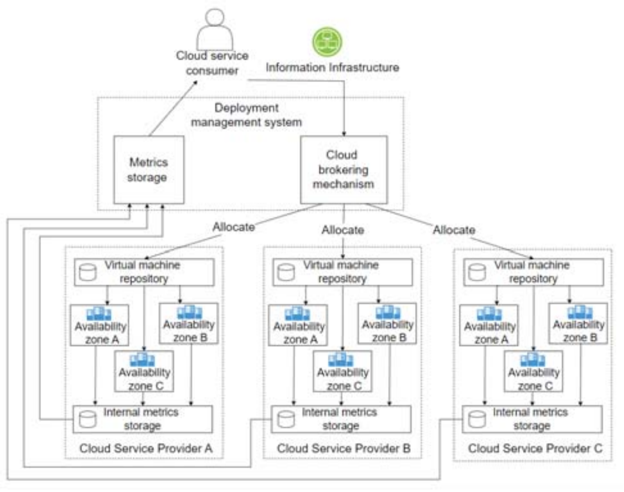


Figure 1 – System model

3 MATERIALS AND METHODS

As depicted in Figure 1, the system model consists of six core components: cloud service consumer, cloud service provider, virtual machines, information infrastructure, availability zones of cloud providers, and cloud brokering mechanism. The cloud service consumer wants to deploy information infrastructure.

Cloud service providers specialize in providing cloud services to customers on demand. Cloud service providers offer the following service models: Software as Service (SaaS) – the capability provided to the consumer to use the provider's applications running on cloud infrastructure, Infrastructure as Service (IaaS) – the capability provided to the consumer to provision processing, storage, networks, and other fundamental computing resources where the consumer can deploy and run arbitrary software, which can include operating systems and applications, Platform as Service (PaaS) – the capability provided to the consumer to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages,

© Rolik O. I., Zhevakin S. D., 2024
DOI 10.15588/1607-3274-2024-3-14

libraries, services, and tools supported by the provider [2]. Let $P = \{P_1, P_2, ..., P_{last}\}$ define the set of cloud service providers.

Cloud service providers offer non-standardized APIs for interacting with their services. Such makes those services not interchangeable among different providers. Consequently, the only model that has unified characteristics and configuration parameters, regardless of the selected cloud service provider is application-level services, specifically IaaS services. In this work, among all IaaS services, the deployment of virtual machines will be reviewed.

The virtual machines within each cloud provider are divided into classes by specialization in such a way that they fully cover customers' needs. Within each specialization, virtual machines are further divided into classes. The division into classes is based on the virtual machine parameters, with the most common criteria being the number of virtual processors and the amount of virtual memory. Thus, among cloud service providers, analogs can be found for each type of virtual machine within each specialization. This work will focus on virtual machines with general-purpose specialization. General-purpose virtual machines provide a balance of computing, memory, and networking resources. These instances are ideal for applications that use these resources in equal proportions, such as web servers [18]. Therefore, let $G = \{G_1, G_2, ..., G_{last}\}$ define a set of virtual machine classes within general-purpose specialization that are available among all providers.

Nowadays, there is no direct correlation between the set of applications in the information infrastructure and the resulting number of virtual machines that will be used for their deployment. This is associated with the use of new architectural approaches to building information infrastructure and its components. An example of such an architecture is service-oriented architecture (SOA) [19]. In this architecture applications can be divided into several components, each of which can be separately deployed on different virtual machines. Such a deployment approach enables independent scaling of application components which leads to more efficient resource utilization. Let $A = \{A_1, A_2, ..., A_{last}\}$ define the set of application components. Set A allows the deployment of different applications and therefore their components, to do so all application components of all applications should be placed inside the set.

During the application operation, its components constantly exchange requests and messages, generating a significant data flow. To optimize the cost of information infrastructure operation, application components are placed as close as possible to each other, usually within the same provider. Such placement occurs because most cloud providers have zero communication costs within the internal network, unlike the costs for communication with external networks [20]. Therefore, let S $(A \times A)$ define the relation matrix of application components that must be deployed within the same cloud provider. Thus, $S_{a_1 a_2}$ equals 1 when a_1 and a_2 components must be deployed within the same provider, and 0 otherwise.

Despite the benefits of services co-location, it is also associated with certain risks. The article [21] gives examples in which, in order to ensure system security, application components are deployed on separate clouds. Such fragmentation of the application is associated with mistrust towards cloud service providers, as they represent a closed system where it is impossible to track what happens to the application source code that is transferred for hosting. Cloud service providers have access to inbound and outbound application traffic. Using this data, application logic can be reconstructed. Due to such risks, some architectural approaches divide the application into components, so that the information about each component is insufficient to reconstruct an overall picture of the application's operation. Moreover, these components are deployed on different clouds. Therefore, let D $(A \times A)$ define the relationship matrix of application components that should be placed in different clouds. Thus, $D_{a_1 a_2}$ equals 0 when a_1 and a_2 components must be deployed in different providers, and 1 otherwise.

However not only safety considerations can make adjustments to the service deployment plan. Restrictions regarding service deployment may arise from the legislation of the country in where the user operates. Also, similar restrictions may arise for clients with whom the user operates. These restrictions are often related to regional zones where data can be stored and how it should be used, along with who can access it, and under which conditions. An example of such a restriction is the General Data Protection Regulation (GDPR). GDPR in the European Union imposes restrictions on the transfer of personal data outside the European Economic Area (EEA) to ensure an adequate level of protection. Data transfers to countries outside the EEA must adhere to specific safeguards outlined in the GDPR, such as adequacy decisions, standard contractual clauses, binding corporate rules, or explicit consent from data subjects [22]. In the United States, similar restrictions are imposed by laws such as HIPAA [23] and GLBA [24]. Additionally, countries such as China, Indonesia, Vietnam, and India have restrictions in which certain types of data should be stored and processed only within the country. Therefore, let $R = \{R_1, R_2, ..., R_{last}\}$ define the set of availability zones, such that each provider $p \in P$ has a data center located in the availability zone $r \in R$. Also, let B $(R \times A)$ define the relationship matrix between application component a and availability zone r. Such that, B_{ra} equals 0 when component a should not be deployed in the availability zone r, and 1 otherwise.

Cloud service providers offer access to resources on-demand basis, charging an hourly fee for their usage. The cost of using a virtual machine depends on its specialization, parameters, and the region in which it will be deployed. In addition to the on-demand plan, cloud service providers offer the opportunity to reserve virtual machines for a long period. In the reservation plan, the user pays an upfront fee along with an hourly usage fee.

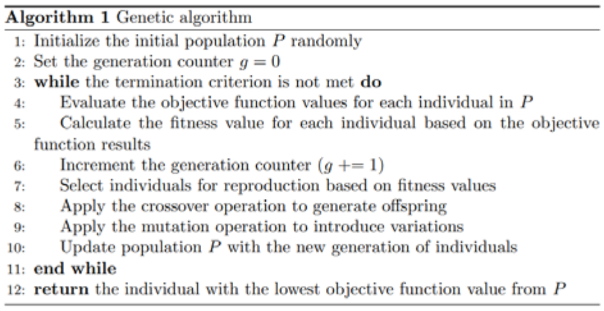
© Rolik O. I., Zhevakin S. D., 2024
DOI 10.15588/1607-3274-2024-3-14

The hourly cost of using a reserved virtual machine is lower than the on-demand rate [25].

This study focuses on the initial deployment of infrastructure in a multi-cloud environment, the main purpose of which is to collect the necessary metrics for further infrastructure utilization. It is considered that the observation period will be less than the minimum reservation period, i.e., one year, so only on-demand virtual machines will be used. Also, such a pricing scheme is appropriate given the uncertainty of the virtual machines' utilization post-observation period. Therefore, let the element C_{prg} from the matrix C $(P \times R \times G)$ define the hourly price of a virtual machine at the provider $p \in P$, in the region $r \in R$ of class $g \in G$ for on-demand usage.

The described problem is a combinatorial optimization problem, as the main challenge lies in the optimal or suboptimal deployment of virtual machines. Combinatorial optimization problems (COPs), especially real-world COPs, are challenging because they are difficult to formulate and solve. Additionally, choosing the proper solver algorithm and defining its best configuration is also a difficult task due to the existence of several solvers characterized by different parametrizations [26].

To solve the problem, a family of genetic algorithms was used. Genetic algorithms (GA) are a family of computational models inspired by evolution. These algorithms encode a potential solution to a specific problem on a simple chromosome-like data structure and apply recombination operators to these structures to preserve critical information [27]. The general scheme of the GA is presented in Figure 2.



```
Algorithm 1 Genetic algorithm
1: Initialize the initial population P randomly
2: Set the generation counter $g = 0$
3: while the termination criterion is not met do
4:     Evaluate the objective function values for each individual in P
5:     Calculate the fitness value for each individual based on the objective
       function results
6:     Increment the generation counter ($g$ += 1)
7:     Select individuals for reproduction based on fitness values
8:     Apply the crossover operation to generate offspring
9:     Apply the mutation operation to introduce variations
10:    Update population P with the new generation of individuals
11: end while
12: return the individual with the lowest objective function value from P
```

Figure 2 – General scheme of generic algorithm

Due to constraint (7), the solution string encoding is performed in the form of representing decision variables as a vector.

GA is suited for unconstrained optimization. For use in constrained optimization, a penalty function is introduced. The main goal of the penalty function is to add a penalty value to the fitness result for an infeasible solution. The penalty value depends on the constraint's violation amount or on the amount by which constraints are violated. Therefore, a feasible result would have more chances to pass through during the reproduction phase in comparison to an infeasible one. There are two types of penalty functions: exterior and interior [27]. In the interior penalty function, all elements from the initial population should be feasible. On the other hand, the exterior penalty function doesn't require an initial population to be feasible. Exterior penalty functions are used more often because, in some cases, finding a feasible solution is itself an NP-hard task.

In this work, different exterior penalty functions will be compared to find out the best-fitting one for the described problem, such as:

– Static penalty function;

– Dynamic penalty function;

– Additive penalty function.

The static penalty function adds a penalty based on the number of violated constraints. The static penalty function was taken from [28] and has the following form:

$$static\ (x) = \begin{cases} of\,(x), & \text{when } x \text{ is feasible,} \\ K(1 - \dfrac{s}{m}), & \text{otherwise.} \end{cases} \tag{8}$$

Where x is the solution vector, $of\,(x)$ is the value of the objective function for the solution vector x, K is a large positive constant, s determines the number of constraints that have been met, and m determines the total number of constraints.

A dynamic penalty function adjusts the penalty value during the optimization process using information about the current state of the population. These adjustments balance the exploration and exploitation phases of optimization, potentially improving both the speed and quality of results. The dynamic penalty function was chosen from [29] and presented in equations (9–12):

$$dynamic\ (x) = of\,(x) + (Yt)^{\alpha}\, SVC(\beta, x)\,, \tag{9}$$

$$SVC(\beta, x) = \sum_{i}^{q} D_i^{\beta}(x) + \sum_{j=q+1}^{m} D_j(x)\,, \tag{10}$$

$$D_i(x) = \begin{cases} 0, & \text{when } g_i(x) \le 0,\ i \le q, \\ |g_i(x)|, & \text{otherwise,} \end{cases} \tag{11}$$

Where in (9) x is the solution vector, $of\,(x)$ is the value of the objective function for the solution vector x. Y, α, β are the input parameters of the penalty function, which are defined by the user and t is the iteration counter. Equation (10) determines the value of the penalty function depending on the violated constraints, where m is the total number of constraints. The (11) equation determines the penalty value imposed by inequality constraints when the constraints are violated, suppose that inequality constraints have the following representation $g_i(x) \le 0$, $\forall\, i \in \{1, 2, \ldots, q\}$. The (12) equation in turn determines the penalty value for equality constraints with following form $h_j(x) = 0$, $\forall\, j \in \{q+1, q+2, \ldots, m\}$.

The additive penalty function uses weights that are defined for each constraint independently. Additive penalties have the following form:

DOI 10.15588/1607-3274-2024-3-14

$$additive\ (x) = of(x) + \sum_{i}^{q} r_i G_i(x) + \sum_{j=q+1}^{m} c_j L_j(x), \tag{13}$$
$$r_i \in M \quad \forall i \in \{1,2,\ldots,q\},\ \ c_j \in M \ \forall j \in \{q+1, q+2, \ldots, m\}$$

$$G_i(x) = \max[0, g_i(x)^{\beta}], \tag{14}$$

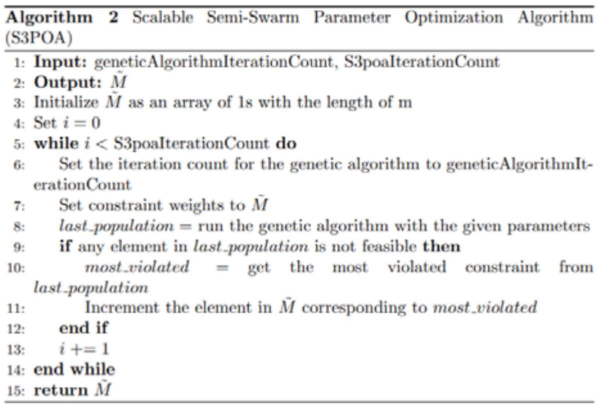
$$L_j(x) = |\,h_j(x)|^{\gamma}. \tag{15}$$

Where in (13) r_i and c_j define the penalty weight multipliers that inequality and equality constraints respectively impose when violated. Also, parameters β and γ in (14) and (15) are used to balance the impact that inequality and equality constraints have on penalty value. The set *M* contains the values of all weights. Compared to the previously described penalty functions, this function can be more precisely adapted to the problem due to a larger set of parameters and the possibility of fine-tuning them. But with such an opportunity, the problem of finding the values of these weights arises. The issue is that the penalty value directly depends on the values of the weights r_i and c_j. If they are set too small, the time to find the first feasible solution will increase, while the result could be better due to the possible search across the infeasible region [28]. Conversely, if r_i and c_j are given too large values, the GA will quickly enter the local optimum and will less frequently resort to searching across infeasible space.

To find the r_i and c_j parameter values of the additive function, the scalable semi-swarm parameter optimization algorithm (S3POA) was developed. The algorithm consists of two stages: finding relative parameter values and adjusting them.

In the first stage of the S3POA algorithm, the objective function was modified so that each constraint, when violated, was multiplied by a large value. Thus, the penalty function takes the following form (16).

$$Evaluate\ (x) = of(x) + B\left[\sum_{i}^{q} \tilde{r}_i G_i(x) + \sum_{j=q+1}^{m} \tilde{c}_j L_j(x)\right]. \tag{16}$$



```
Algorithm 2 Scalable Semi-Swarm Parameter Optimization Algorithm (S3POA)
1: Input: geneticAlgorithmIterationCount, S3poaIterationCount
2: Output: $\tilde{M}$
3: Initialize $\tilde{M}$ as an array of 1s with the length of m
4: Set $i = 0$
5: while $i <$ S3poaIterationCount do
6:     Set the iteration count for the genetic algorithm to geneticAlgorithmIterationCount
7:     Set constraint weights to $\tilde{M}$
8:     last_population = run the genetic algorithm with the given parameters
9:     if any element in last_population is not feasible then
10:        most_violated = get the most violated constraint from last_population
11:        Increment the element in $\tilde{M}$ corresponding to most_violated
12:    end if
13:    $i$ += 1
14: end while
15: return $\tilde{M}$
```

Figure 3 – Scheme of the first stage of S3POA

In Figure 3 the scheme of the first stage of S3POA is presented. The input parameters for the described algorithm include the number of iterations of the GA. This value should not be too small to allow the algorithm to reach a local optimum, but it also should not be too big to prevent the algorithm from conducting a long search in the infeasible region. The idea of the first stage algorithm involves conducting a fixed number of iterations of the GA with the (16) objective function. After the GA is finished, the final population is analyzed. If any element of the final population belongs to the feasible region, the next iteration of the algorithm is performed. Otherwise, among the elements of the final population, the constraint with the most violations is identified. After finding such a constraint, its corresponding relative weight ($ř_i$ for a non-strict constraint and $č_j$ for a strict constraint) is increased by one. The next iteration of the algorithm is conducted with updated weights. The algorithms continue until a given number of iterations is reached which is provided as an input parameter to the algorithm.

Gradually increasing the values of the stopping weights, decreases the probability that they will be violated in the next iteration of the algorithm. This leads to a more comprehensive consideration of the resulting weight values. As a result, the values of the relative weights $ř_i$ and $č_j$ are obtained such that $ř_i \in \tilde{M}\ \forall\, i \in \{1, 2, \ldots, q\}$, $č_j \in \tilde{M}\ \forall j \in \{q+1, q+2, \ldots, m\}$. These relative weights allow for examining the relationship between constraints.

The found relative weight values cannot be used immediately to solve the given problem, as their value increases with the number of algorithm iterations, which may result in assigning excessive values to them. Excessively large values of these parameters will lead to a situation where the GA will quickly find a feasible solution, but then it will have a harder time considering infeasible solutions because it will have excessively high fitness values [28]. To overcome this issue, found weight values are considered relative. To find the best fitting weight values, the objective function from (13) will be used, but with the following values:

$$r_i = \begin{cases} 1, & \breve{r}_i = 1, \\ \mu \breve{r}_i, & \text{otherwise}, \end{cases} \tag{17}$$

$$c_j = \begin{cases} 1, & \breve{c}_j = 1, \\ \mu \breve{c}_j, & \text{otherwise}. \end{cases} \tag{18}$$

In (17) and (18), the multiplier μ is responsible for the scale that should be applied to the variables $ř_i$ and $č_j$ to avoid the problem of insufficient and excessive weight assignment. The value of relative weights equal to 1 is not scaled since during the experiments these constraints were not violated, so they do not prevent reaching the feasible solution.

4 EXPERIMENTS

To conduct the experiments, the input parameters of the problem were set and presented in Tables 1 – 6. Table 1 shows the regions where virtual machines will be

DOI 10.15588/1607-3274-2024-3-14

deployed. As for cloud service providers, AWS, Azure, and Google Cloud were chosen as they occupy a significant portion of the cloud services market [30].

Table 1 – Correspondence of the geographic regions to the availability regions of the providers

	AWS	Azure	Google Cloud
EU west	eu-west-1	West Europe	europe-west1-b
US east	us-east-1	East US	us-east4-a
EU central	eu-central-1	Sweden Central	europe-north1-a
Japan	asia-south1-a	Japan East	asia-northeast1-a

Table 2 – Description of selected virtual machines among different cloud service providers

VM names	Provider	vCPUs	Memory Size (GB)	Price in West Europe ($)	Price in East US ($)	Price in North Europe ($)	Price in Japan ($)
t2.small	AWS	1	2	0.027	0.025	0.0286	0.0322
t2.medium	AWS	2	4	0.0539	0.0499	0.0571	0.0643
t2.xlarge	AWS	4	16	0.2086	0.1926	0.2214	0.2502
t2.2xlarge	AWS	8	32	0.4172	0.3852	0.4428	0.5004
Standart_A1_V2	Azure	1	2	0.041	0.043	0.041	0.054
Standart_A2_V2	Azure	2	4	0.087	0.091	0.0861	0.113
Standart_A4_V2	Azure	4	8	0.183	0.191	0.182	0.238
Standart_A8_V2	Azure	8	16	0.383	0.4	0.38	0.5
t2d-standard-1	Google Cloud	1	4	0.0465	0.0476	0.0465	0.0542
t2d-standard-2	Google Cloud	2	8	0.0929	0.0952	0.093	0.1084
t2d-standard-4	Google Cloud	4	16	0.1859	0.1903	0.1861	0.2168
t2d-standard-8	Google Cloud	8	32	0.3718	0.3806	0.3721	0.4336

Information about the selected classes of virtual machines, their characteristics, and the on-demand hourly cost of using them relative to regions is presented in Table 2. According to the information presented in Table 2, four different classes of virtual machines were formed based on the number of virtual processors. Tables 3–5 describe the requirements regarding application components deployment. Thus, Table 3 presents the requirements regarding application components deployment within the same cloud service provider, defining the matrix *S*. Table 4 presents the requirements regarding application components deployment within different cloud service providers, defining the matrix *D*. Table 5 describes the requirements regarding the characteristics of virtual machines on which application components should be deployed.

Constraints regarding the application components deployment in relation to availability zones were also introduced. Specifically, a_3 and a_6 application components must be located within the European geographic region, meaning they can be deployed in the EU West and EU Central geographical zones.

Table 3 – Constraints on the placement of application components within one provider

	1	2	3	4	5	6	7	8
1	0	1	0	0	0	0	0	0
2	1	0	0	0	0	0	0	0
3	0	0	0	0	0	1	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	1	0	0	0	0	0
7	0	0	0	0	0	0	0	1
8	0	0	0	0	0	0	1	0

Table 4 – Constraints on the placement of application components across different providers

	1	2	3	4	5	6	7	8
1	1	1	0	1	1	0	1	1
2	1	1	0	1	1	0	1	1
3	0	0	1	1	1	1	0	0
4	1	1	1	1	0	1	1	1
5	1	1	1	0	1	1	1	1
6	0	0	1	1	1	1	0	0
7	1	1	0	1	1	0	1	1
8	1	1	0	1	1	0	1	1

© Rolik O. I., Zhevakin S. D., 2024
DOI 10.15588/1607-3274-2024-3-14

Table 5 – Minimum required virtual machine parameter values for application components deployment

Application	1	2	3	4	5	6	7	8
Min vCPU	2	1	4	6	3	1	2	8
Min Memory (GB)	4	2	16	16	8	2	16	20

To conduct experiments, a virtual machine of class c7i.xlarge from the AWS provider was used. The c7i.xlarge is the compute-optimized family of instances with 4 vCPUs of Intel Xeon Scalable (Sapphire Rapids) with 3.2 GHz clock speed, 8.0 GB of memory, and up to 12.5 Gbps of bandwidth [31].

The same basic parameters of the GA were set for each penalty function. Also, the GA was run 100 times with each penalty function. The termination condition for the algorithm was reaching 1000 iterations.

Parameter tuning of the penalty function parameters as well as other GA parameters is carried out individually for each problem. Parameter tuning for GAs is widely covered in the literature. In this section, the focus will be on parameter tuning for the described penalty functions.

As for the static penalty function input parameter is K, which, as described in [29], should be a large number. Therefore, during the experiments, the value determined by the authors will be used, such that $K = 10^9$.

For the dynamic penalty function, the input parameters are the values Y, α, β. The authors in [29] suggest using values $Y = 0.5$, $\alpha = 2$, $\beta = 2$. To find the values of these parameters, a series of experiments were conducted, as a result of which it was found that, for the described problem, the best result was achieved with the values $Y = 0.7$, $\alpha = 1$, $\beta = 1$. These values will be used in further experiments.

For the additive penalty function, weight values were calculated with the S3POA algorithm. As input parameters for the S3POA algorithm, the value of *geneticAlgorithmIterationCount* was set to 89. With this number of iterations, GA reached a feasible solution in 47.8% of invocations, such an indicator showed the ability of the algorithm to find feasible value without conducting an extensive search. The value of *S3poaIterationCount* was set to 1000. The value of B was set to 10^9. As depicted in [29], common values of β and γ parameters are 1 or 2. During experiments, those parameters were assigned the following values: $\beta = 1$, $\gamma = 2$, since the described problem has only one type of equality constraint (7) which had a significant impact on the result and was most often violated.

5 RESULTS

Based on the results of the conducted experiments, the static penalty function failed to achieve a result that belongs to the feasible region. This outcome was due to the function definition, as it only considered the number of violated constraints without considering the degree of their violation. Unfortunately, this description was insufficient to guide the algorithm toward a feasible result. Therefore, the results of the static penalty function were excluded from the further experiment results.

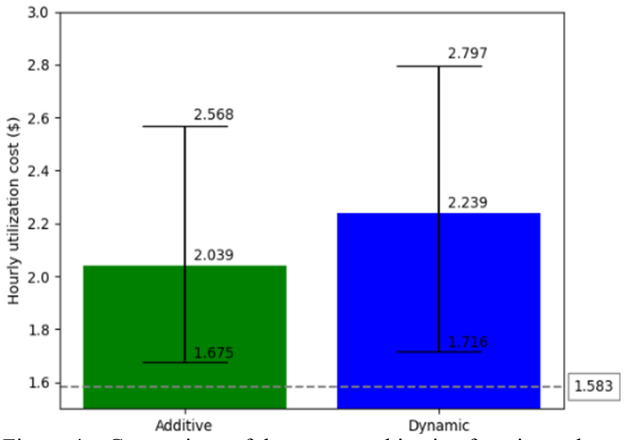


Figure 4 – Comparison of the average objective function values for the additive and dynamic penalty functions.

For the dynamic and additive penalty functions, the resulting values of the objective functions that belonged to the feasible region were compared. To analyze the obtained results, the value of the approximated global optimum was determined. This value was found during problem-solving without considering constraints (4)–(7). The excluded constraints described the deployment plan of virtual machines. Figure 4 demonstrates a comparison diagram of the average obtained results for the dynamic and additive penalty functions. It includes the average, maximum, and minimum values of the objective function obtained during the experiments. The value of the approximated optimum was displayed in Figure 4 with a dashed line.

Additionally, a comparison of the average time spent to find the first feasible and resulting solutions for the dynamic and additive penalty functions was conducted, and the obtained results are presented in Figure 5.

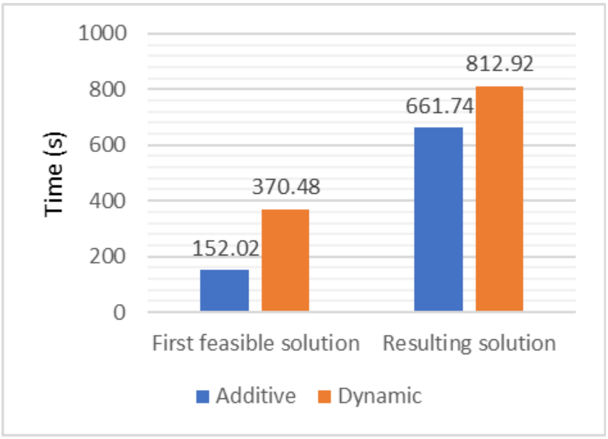


Figure 5 – Comparison of the average time spent to find the first feasible and the resulting solutions for the additive and dynamic penalty functions.

© Rolik O. I., Zhevakin S. D., 2024
DOI 10.15588/1607-3274-2024-3-14

6 DISCUSSION

Comparing the results of the dynamic and additive penalty functions, it can be concluded that the dynamic penalty function failed to achieve any feasible result in 33% of epy experiments, while for the additive penalty function, this indicator was only 10% of the conducted experiments.

To assess the possibility of the selected penalty functions exiting from the local optimum and continuing searching for a better solution, after reaching the first feasible solution, experiments were conducted in which the first founded feasible solution was compared with the resulting solution of the algorithm. Thus, among all the results that reached a feasible solution, the additive penalty function obtained a resulting solution that was better than the first feasible solution in 85% of cases, while for the dynamic penalty function, this indicator was 61%.

Analyzing Figure 4, it can be concluded that the additive penalty function, on average, showed on 8.933% better results than the dynamic function. Also, on average, the result of the additive function was by 22.36% greater than the value of the approximated optimum.

For the additive penalty function, the average algorithm runtime was 1241.53 seconds, while for the dynamic penalty function, this value was equal to 1239.59 seconds. Based on the obtained results, the use of different penalty functions had almost no effect on the overall algorithm runtime.

Analyzing the data presented in Figure 5, it can be concluded that, on average, the GA with the additive penalty function required 2.437 times less time to find the first feasible solution compared to the dynamic penalty function. Also, the average time spent to find the best solution for the additive penalty function was 18.6% less than the time required for the dynamic penalty function to find such a solution. Considering that 85% of the resulting solutions obtained by the additive penalty function were better than the first feasible solution, compared to 61% for the dynamic function, it can be concluded that, despite spending less time to find the best solution, the additive function less often stopped at the local optimum and had a higher chance of finding a more profitable solution.

Summing up, it can be concluded that during the experiments, the additive penalty function performed better than the dynamic and the static penalty functions. Also, this result indicates that the described S3POA parameter tuning algorithm is capable of balancing the input parameter values. On the one hand, the algorithm spends less time finding the first feasible solution, and on the other hand, upon reaching a local optimum, the algorithm retains the ability to search for better solutions among the infeasible region. Referring to [28], these characteristics indicate the selection of best-fitting parameter values for the penalty function. This result was achieved through direct interaction with the problem during the determination of parameter values. That is, the proposed parameter tuning method directly interacted with the problem, dynamically adjusting the parameter values to achieve better results, rather than statically enumerating possible values as was the case with the dynamic and static functions.

CONCLUSION

In this paper, the problem of informational infrastructure deployment in a static multi-cloud environment was formulated as an optimization problem, the main goal of which was to minimize the hourly cost of utilizing information infrastructure. A GA was used to solve the formulated problem. Various penalty functions were observed for the proposed algorithm, namely static, dynamic, and additive. The input parameter values were tuned for the selected penalty functions. Additionally, a S3POA parameter tuning algorithm for the penalty function parameters was proposed. The obtained results showed that the additive penalty function with parameters selected by the proposed S3POA method turned out to be better in comparison to others.

The scientific novelty. A new method was proposed that allows the decrease of information infatuation utilization cost in a static multi-cloud environment, which considers the division of the application into components and introduces restrictions on their placement. The use of different availability zones to deploy application components was also considered. Additionally, a new parameter optimization method for GA penalty functions was proposed, which allows obtaining better parameter values due to consistent interaction with the researched problem.

The practical orientation of the study. The information infrastructure deployed under the described assumptions can be used to collect data about application load and interaction patterns over a certain period. Such information provides a clear view of infrastructure needs and can be used during infrastructure operations to have reliable information on the possible service loads and resource requirements.

Prospects for further research. In future research, it makes sense to consider the usage of scalable instances. In addition, the proposed solution can be integrated into dynamic placement algorithms to be used as a starting point in the algorithm's operation.

REFERENCES

1. Zhang X., Yue Q., He Z. Dynamic Energy-Efficient Virtual Machine placement optimization for virtualized clouds, *Lecture Notes in Electrical Engineering*, 2014, pp. 439–448. DOI: 10.1007/978-3-642-53751-6_47.
2. Song F., Huang D., Zhou H., Zhang H., You I. An Optimization-Based scheme for efficient virtual machine placement, *International Journal of Parallel Programming*, 2013, Vol. 42, № 5, pp. 853–872. DOI: 10.1007/s10766-013-0274-5.
3. Amazing Cloud Adoption Statistics [2023]: Cloud migration, computing, and more [Electronic resource]. Access mode: https://www.zippia.com/advice/cloud-adoption-statistics/

4. Tawfeek M. A., El-Sisi A. B., Keshk A., Torkey F. A. Virtual machine placement based on ant colony optimization for minimizing resource wastage, *Communications in Computer and Information Science,* 2014, pp. 153–164. DOI: 10.1007/978-3-319-13461-1_16.
5. EC2 On-Demand Instance Pricing – Amazon Web Services. Amazon Web Services, Inc [Electronic resource]. Access mode: https://aws.amazon.com/ec2/pricing/on-demand/?nc1-=h_ls
6. Pricing | Compute Engine: Virtual Machines (VMs) | Google Cloud. [Electronic resource]. Access mode: https://cloud.google.com/compute/all-pricing
7. Heilig L., Lalla-Ruiz E., Voss S. A cloud brokerage approach for solving the resource management problem in multi-cloud environments, *Computers & Industrial Engineering*, 2016, Vol. 95, pp. 16–26. DOI: 10.1016/j.cie.2016.02.015.
8. Telenyk S., Zharikov E., Rolik O. Consolidation of virtual machines using stochastic local search, *Advances in Intelligent Systems and Computing,* 2017, pp. 523–537. DOI: 10.1007/978-3-319-70581-1_37.
9. Song F. Huang D., Zhou H., Zhang H., You I. An Optimization-Based scheme for efficient virtual machine placement, *International Journal of Parallel Programming,* 2013, Vol. 42, № 5, pp. 853–872. DOI: 10.1007/s10766-013-0274-5.
10. Tawfeek M. A., El-Sisi A. B., Keshk A., Torkey F. A. Virtual machine placement based on ant colony optimization for minimizing resource wastage, *Communications in Computer and Information Science*, 2014, pp. 153–164. DOI: 10.1007/978-3-319-13461-1_16.
11. Charrada F. B., Tebourski N., Tata S., Moalla S. Approximate Placement of Service-Based Applications in Hybrid Clouds, *2012 IEEE 21st International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises.* DOI: 10.1109/wetice.2012.76.
12. Chaisiri S., Lee B.-S., Niyato D. Optimal virtual machine placement across multiple cloud providers, *2009 IEEE Asia-Pacific Services Computing Conference (APSCC).* DOI: 10.1109/apscc.2009.5394134.
13. Subramanian T., Nickolas S. Application based brokering algorithm for optimal resource provisioning in multiple heterogeneous clouds, *Vietnam Journal of Computer Science,* 2015, Vol. 3, № 1, pp. 57–70. DOI: 10.1007/s40595-015-0055-8.
14. Màsdàrí M., Nabavi S. S., Ahmadi V. An overview of virtual machine placement schemes in cloud computing, *Journal of Network and Computer Applications,* 2016, Vol. 66, pp. 106–127. DOI: 10.1016/j.jnca.2016.01.011.
15. Lucas-Simarro J. L., Moreno-Vozmediano R., Montero R., Llorente I. Cost optimization of virtual infrastructures in dynamic multi-cloud scenarios, *Concurrency and Computation*, 2012, Vol. 27, № 9, pp. 2260–2277. DOI: 10.1002/cpe.2972.
16. Tordsson J., Montero R., Moreno-Vozmediano R., Llorente I. Cloud brokering mechanisms for optimized placement of virtual machines across multiple providers, *Future Generation Computer Systems,* 2012, Vol. 28, № 2, pp. 358–367. DOI: 10.1016/j.future.2011.07.003.
17. Bellur U., Malani A., Narendra N. C. Cost optimization in multi-site multi-cloud environments with multiple pricing schemes, *2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing,* pp. 115–122. DOI: 10.1109/cloud.2014.97.
18. General purpose instances – Amazon EC2. [Electronic resource]. Access mode: https://docs.aws.amazon.com/-ec2/latest/instancetypes/gp.html
19. SOA Source Book – Infrastructure for SOA. [Electronic resource]. Access mode: https://collaboration.opengroup-.org/projects/soa-book/pages.php?action=show&ggid=1336
20. Overview of data transfer costs for common architectures | Amazon Web Services. Amazon Web Services. [Electronic resource]. Access mode: https://aws.amazon.com/blogs/-architecture/overview-of-data-transfer-costs-for-common-architectures/
21. Kaviani N., Wohlstadter E., Lea R. Partitioning of web applications for hybrid cloud deployment, *Journal of Internet Services and Applications,* 2014, Vol. 5, № 1. DOI: 10.1186/s13174-014-0014-0.
22. Chapter 5 – Transfers of personal data to third countries or international organisations – General Data Protection Regulation (GDPR). General Data Protection Regulation (GDPR). [Electronic resource]. Access mode: https://gdpr-info.eu/chapter-5/
23. Health Insurance Portability and Accountability Act of 1996. ASPE. Online. 20 August 1996. [Electronic resource]. Access mode: https://aspe.hhs.gov/reports/health-insurance-portability-accountability-act-1996
24. GovInfo. [Electronic resource]. – Access mode: https://www.govinfo.gov/app/details/PLAW-106publ102
25. Introduction to Amazon EC2 Reserved instances. Amazon Web Services, Inc. [Electronic resource]. Access mode: https://aws.amazon.com/ec2/pricing/reserved-instances/
26. Peres F., Castelli M. Combinatorial Optimization Problems and Metaheuristics: review, challenges, design, and development, *Applied Sciences,* 2021, Vol. 11, № 14, P. 6449. DOI: 10.3390/app11146449.
27. Yeniay Ö. Penalty Function Methods for Constrained Optimization with Genetic Algorithms, *Mathematical and Computational Applications*, 2005, Vol. 10. № 1, pp. 45–56. DOI: 10.3390/mca10010045.
28. Morales K., Quezada C. A universal Eclectic genetic algorithm for constrained optimization, *6th European Congress on Intelligent Techniques & Soft Computing.* 1998. Vol. 518–522. [Electronic resource] Access mode: http://cursos.itam.mx/akuri/PUBLICA.CNS-/1998/Universal%20EGA%20%28EUFIT98%29.PDF.
29. Joines J. A., Houck C. R. On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GA's, *First IEEE International Conference on Evolutionary Computation*, 2002. DOI: 10.1109/icec.1994.349995.
30. RICHTER, Felix. Amazon maintains cloud lead as Microsoft Edges closer. Statista Daily Data. [Electronic resource]. – Access mode: https://www.statista.com/chart/-18819/-worldwide-market-share-of-leading-cloud-infrastructure-service-providers/
31. Introducing Amazon EC2 Flex instances (1:24[Electronic resource]. Access mode: https://aws.amazon.com/ec2/instance-types/c7i/

Received 22.04.2024.
Accepted 16.08.2024.

УДК 004.94

МЕТОД ОПТИМІЗАЦІЇ ВИТРАТ ДЛЯ РОЗМІЩЕННЯ ІНФОРМАЦІЙНОЇ ІНФРАСТРУКТУРИ В СТАТИЧНОМУ МУЛЬТИХМАРНОМУ СЕРЕДОВИЩІ

Ролік О. І. – д-р техн. наук, професор, завідувач кафедри інформаційних систем та технологій, Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», Київ, Україна.

Жевакін С. Д. – аспірант кафедри інформаційних систем та технологій, Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», Київ, Україна.

АНОТАЦІЯ

Актуальність. Останнім часом набула популярності тема розміщення інформаційної інфраструктури в мультихмарному середовищі. Дана тенденція пов'язана з тим, що мультихмарне середовище надає можливість використовувати унікальні сервіси різних хмарних постачальників. Таким чином, всі доступні сервіси хмарних постачальників можуть бути використані при побудові інформаційної інфраструктури. Крім того, різні цінові політики серед постачальників можуть бути розглянуті при виборі сервісів. Проте зі збільшенням кількості наявних постачальників хмарних послуг зростає складність побудови оптимального плану з розміщення інформаційної інфраструктури.

Мета роботи. Метою роботи є оптимізація витрат пов'язаних з експлуатацією інформаційної інфраструктури в мультихмарному середовищі з урахуванням цін на аналогічні сервіси, серед постачальників хмарних послуг.

Метод. В роботі пропонується новий метод оптимізації витрат для розміщення інформаційної інфраструктури в статичному мультихмарному середовищі, який мінімізує погодинну вартість її використання. Для вирішення цієї задачі було використано генетичний алгоритм. Були розглянуті різні функції штрафу для генетичного алгоритму. Також пропонується новий метод підбору параметрів для функцій штрафу.

Результати. Була проведена серія експериментів для порівняння результатів різних функцій штрафу. Результати показали, що функція штрафу із запропонованим методом підбору параметрів знаходила рішення, яке було у середньому на 8,933% кращим і вимагало на 18,6% менше часу, в порівняні з іншими. Отримані результати демонструють, що запропонований метод підбору параметрів забезпечує ефективний пошук серед областей допустимих і недопустимих рішень.

Висновок. Запропоновано новий метод оптимізації витрат для розміщення інформаційної інфраструктури в статичному мультихмарному середовищі. Однак, незважаючи на ефективність запропонованого методу, його можна значно покращити. Зокрема, необхідно розглянути можливість залучення масштабованих віртуальних машин при розміщенні інформаційної інфраструктури.

КЛЮЧОВІ СЛОВА: оптимізація витрат, інформаційна інфраструктура, початкове розміщення, мультихмара, метод підбору параметрів, функція штрафів.

ЛІТЕРАТУРА

1. Zhang X. Dynamic Energy-Efficient Virtual Machine placement optimization for virtualized clouds / X. Zhang, Q. Yue, Z. He // Lecture Notes in Electrical Engineering. – 2014. – P. 439–448. DOI: 10.1007/978-3-642-53751-6_47.
2. An Optimization-Based scheme for efficient virtual machine placement / [F. Song, D. Huang, H. Zhou et al.] // International Journal of Parallel Programming. – 2013. – Vol. 42, № 5. – P. 853–872. DOI: 10.1007/s10766-013-0274-5.
3. Amazing Cloud Adoption Statistics [2023]: Cloud migration, computing, and more [Electronic resource]. – Access mode: https://www.zippia.com/advice/cloud-adoption-statistics/
4. Virtual machine placement based on ant colony optimization for minimizing resource wastage / [M. A. Tawfeek, A. B. El-Sisi, A. Keshk, F. A. Torkey] // Communications in Computer and Information Science. – 2014. – P. 153–164. DOI: 10.1007/978-3-319-13461-1_16.
5. EC2 On-Demand Instance Pricing – Amazon Web Services. Amazon Web Services, Inc [Electronic resource]. – Access mode: https://aws.amazon.com/ec2/pricing/on-demand/?nc1-=h_ls
6. Pricing | Compute Engine: Virtual Machines (VMs) | Google Cloud. [Electronic resource]. – Access mode: https://cloud.google.com/compute/all-pricing
7. Heilig L. A cloud brokerage approach for solving the resource management problem in multi-cloud environments / L. Heilig, E. Lalla-Ruiz, S. Voss // Computers & Industrial Engineering. – 2016. – Vol. 95. – P. 16–26. DOI: 10.1016/j.cie.2016.02.015.
8. Telenyk S. Consolidation of virtual machines using stochastic local search / S. Telenyk, E. Zharikov, O. Rolik // Advances in Intelligent Systems and Computing. – 2017. – P. 523–537. DOI: 10.1007/978-3-319-70581-1_37.
9. An Optimization-Based scheme for efficient virtual machine placement / [F. Song, D. Huang, H. Zhou et al.] // International Journal of Parallel Programming. – 2013. – Vol. 42, № 5. – P. 853–872. DOI: 10.1007/s10766-013-0274-5.
10. Virtual machine placement based on ant colony optimization for minimizing resource wastage / [M. A. Tawfeek, A. B. El-Sisi, A. Keshk, F. A. Torkey] // Communications in Computer and Information Science. – 2014. – P. 153–164. DOI: 10.1007/978-3-319-13461-1_16.
11. Approximate Placement of Service-Based Applications in Hybrid Clouds / [F. B. Charrada, N. Tebourski, S. Tata, S. Moalla] // 2012 IEEE 21st International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises. DOI: 10.1109/wetice.2012.76.
12. Chaisiri S. Optimal virtual machine placement across multiple cloud providers / S. Chaisiri, B.-S. Lee, D. Niyato // 2009 IEEE Asia-Pacific Services Computing Conference (APSCC). DOI: 10.1109/apscc.2009.5394134.
13. Subramanian T. Application based brokering algorithm for optimal resource provisioning in multiple heterogeneous clouds / T. Subramanian, S. Nickolas // Vietnam Journal of Computer Science. – 2015. – Vol. 3, № 1. – P. 57–70. DOI: 10.1007/s40595-015-0055-8.

DOI 10.15588/1607-3274-2024-3-14

14. Màsdàrí M. An overview of virtual machine placement schemes in cloud computing / M. Màsdàrí, S. S. Nabavi, V. Ahmadi // Journal of Network and Computer Applications. – 2016. – Vol. 66. – P. 106–127. DOI: 10.1016/j.jnca.2016.01.011.
15. Cost optimization of virtual infrastructures in dynamic multi-cloud scenarios / [J. L. Lucas-Simarro, R. Moreno-Vozmediano, R. Montero, I. Llorente] // Concurrency and Computation. – 2012. – Vol. 27, № 9. – P. 2260–2277. DOI: 10.1002/cpe.2972.
16. Tordsson J. Cloud brokering mechanisms for optimized placement of virtual machines across multiple providers / [J. Tordsson, R. Montero, R. Moreno-Vozmediano, I. Llorente] // Future Generation Computer Systems. – 2012. – Vol. 28, № 2. – P. 358–367. DOI: 10.1016/j.future.2011.07.003.
17. Bellur U. Cost optimization in multi-site multi-cloud environments with multiple pricing schemes / U. Bellur, A. Malani, N. C. Narendra // 2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing. – P. 115–122. DOI: 10.1109/cloud.2014.97.
18. General purpose instances – Amazon EC2. [Electronic resource]. – Access mode: https://docs.aws.amazon.com/-ec2/latest/instancetypes/gp.html
19. SOA Source Book – Infrastructure for SOA. [Electronic resource]. – Access mode: https://collaboration.opengroup-.org/projects/soa-book/pages.php?action=show&ggid=1336
20. Overview of data transfer costs for common architectures | Amazon Web Services. Amazon Web Services. [Electronic resource]. – Access mode: https://aws.amazon.com/blogs/-architecture/overview-of-data-transfer-costs-for-common-architectures/
21. Kaviani N. Partitioning of web applications for hybrid cloud deployment / N. Kaviani, E. Wohlstadter, R. Lea // Journal of Internet Services and Applications. – 2014. – Vol. 5, № 1. DOI: 10.1186/s13174-014-0014-0.
22. Chapter 5 – Transfers of personal data to third countries or international organisations – General Data Protection Regulation (GDPR). General Data Protection Regulation (GDPR). [Electronic resource]. – Access mode: https://gdpr-info.eu/chapter-5/
23. Health Insurance Portability and Accountability Act of 1996. ASPE. Online. 20 August 1996. [Electronic resource]. – Access mode: https://aspe.hhs.gov/reports/health-insurance-portability-accountability-act-1996
24. GovInfo. [Electronic resource]. – Access mode: https://www.govinfo.gov/app/details/PLAW-106publ102
25. Introduction to Amazon EC2 Reserved instances. Amazon Web Services, Inc. [Electronic resource]. – Access mode: https://aws.amazon.com/ec2/pricing/reserved-instances/
26. Peres F. Combinatorial Optimization Problems and Metaheuristics: review, challenges, design, and development / F. Peres, M. Castelli // Applied Sciences. – 2021. – Vol. 11, № 14. – P. 6449. DOI: 10.3390/app11146449.
27. Yeniay Ö. Penalty Function Methods for Constrained Optimization with Genetic Algorithms / Ö. Yeniay // Mathematical and Computational Applications. – 2005. – Vol. 10, № 1. – P. 45–56. DOI: 10.3390/mca10010045.
28. Morales K. A universal Eclectic genetic algorithm for constrained optimization / K. Morales, C. Quezada // 6th European Congress on Intelligent Techniques & Soft Computing. 1998. Vol. 518–522. [Electronic resource] Access mode: http://cursos.itam.mx/akuri/PUBLICA.CNS-/1998/Universal%20EGA%20%28EUFIT98%29.PDF.
29. Joines J. A. On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GA's / J. A. Joines, C. R. Houck // First IEEE International Conference on Evolutionary Computation. – 2002. DOI: 10.1109/icec.1994.349995.
30. RICHTER, Felix. Amazon maintains cloud lead as Microsoft Edges closer. Statista Daily Data. [Electronic resource]. – Access mode: https://www.statista.com/chart/-18819/-worldwide-market-share-of-leading-cloud-infrastructure-service-providers/
31. Introducing Amazon EC2 Flex instances (1:24[Electronic resource]. – Access mode: https://aws.amazon.com/ec2/instance-types/c7i/

DOI 10.15588/1607-3274-2024-3-14