Research Article **Open Access**

Sergii Telenyk, Maksym Bukasov*, and Maksym Yasochka

Resource management for server virtualization under the limitations of recovery time objective

DOI 10.1515/phys-2016-0059
Received Jun 14, 2016; accepted Nov 07, 2016

Abstract: The problem of resource management for server virtualization under the limitation of recovery time objective was considered in this paper. Models and methods of workload and resource management of IT infrastructure with server virtualization in terms of time limit for services recovery in case of provision of services by virtual private servers were proposed. Mathematical models for excess and lack of resources was formulated as linear and nonlinear 0-1 programming problems. To solve these problems branch and bound, heuristic and modified genetic algorithms were proposed.

Keywords: virtual private server; disaster recovery; recovery time objective

PACS: 07.05.-t, 07.05.Bx

1 Introduction

Information and Telecommunication Systems (ITS) of leading corporations can be seen as prototypes and components of the future global information society. Based on the implementation of the concept of data centers, they provide access to the own resources (services, applications, computing power, data, etc.) and resources of partner companies, as well as the resources of public authorities and other institutions. For efficient organization and operation of the IT infrastructure it is necessary to solve a number of problems, primarily the creation of conditions for data-processing capabilities, resource management, reliability, safety and others. Hosting companies that invest in creating their own IT infrastructure to service the users, are faced with the above mentioned problems [1]. Customers and hosting companies agree on service level requirements, which usually include: the availability and manageability of IT infrastructure, data integrity, security, reliability and scalability. Achievement of user requirements at the lowest cost is the essence of the problem for development and functioning of the IT infrastructure. Usually, this complex problem is divided into a number of smaller problems, but they are not much easier. One of them is the problem of resource management and workload of the IT infrastructure.

According to research, servers in many companies typically run at 15–20% of their capacity, which may not be a sustainable ratio in the current economic conditions. Therefore, implementation of server virtualization technology to reduce the number of physical servers has become popular in recent years. Also virtualization can ease backup, reduce administrative demands and improve security for companies that hosts desktops on virtual machines that are running on centralized servers in data center.

Over the years, Virtual Private Server (VPS) hosting has emerged as good compromise between the affordability of Shared Hosting and the operational power of a Dedicated Server. The introduction of cloud infrastructures has taken that concept, and extended it even further.

Modern IT infrastructure for high-availability VPS hosting is a network of clusters in a virtualized environment, each of which contains a number of different highly available servers. These clusters contain redundant hardware, so if some node fails for any reason, customer's data will be safeguarded and customer's VPS will be booted up instantaneously on a different node. Logically these VPS clusters consist of Virtual Machines (VM) that are running under the control of Hypervisor on physical servers connected to Shared Data Storage (Fig. 1).

One of the main problems of creating virtualization based IT infrastructures is the problem of workload and resource management. The management of resources is important, since many consumers, such as virtual machine (VMs), zones, containers, or virtual desktops, request resources. Consolidating different workloads on one system often entails combining workloads that have different ser-

***Corresponding Author: Maksym Bukasov:** Faculty of Informatics and Computer Engineering NTUU "KPI", Kyiv, Ukraine; Email: bukasov@gmail.com
Sergii Telenyk, Maksym Yasochka: Faculty of Informatics and Computer Engineering NTUU "KPI", Kyiv, Ukraine

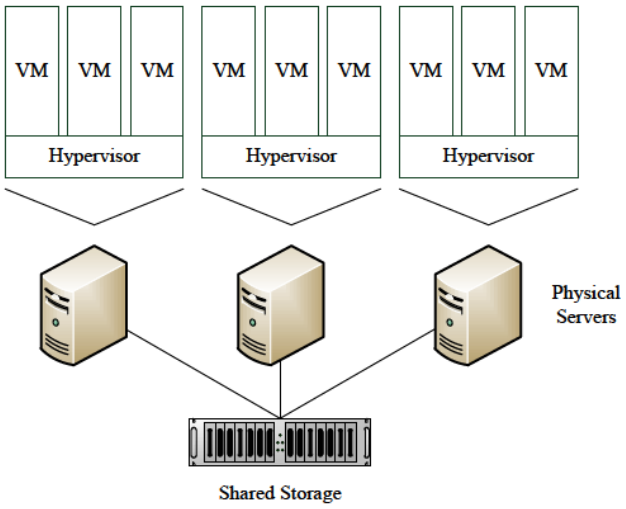

Figure 1: Scheme of VPS with a Shared Storage.

vice level agreements and different needs for throughput, response time, and availability [2]. The problem of workload and resource management for server virtualization is considered in the series of works in which mathematical models and methods was proposed [1, 3–5]. Also there are known practical implementation of workload and resource management for server virtualization [6, 7]. But some problems have not been solved yet. One of them is provisioning of guaranteed recovery time to services in the event server, that hosts virtual machines fails. Restarting those VMs on the reserve servers of the cluster is also of concern.

Disaster recovery (DR) involves a set of policies and procedures to enable the recovery or continuation of infrastructures and systems vital technology following a natural or human-induced disaster. A disaster recovery plan (DRP) is a documented process or set of procedures to recover and protect a business IT infrastructure in case of a fault.

Recovery as a service (RaaS), sometimes referred to as disaster recovery as a service (DRaaS), is a category of cloud computing used for protecting an application or data from a natural or human disaster or service disruption at one location by enabling a full recovery in the cloud. RaaS differs from cloud-based backup services by protecting data and providing standby computing capacity on demand to facilitate more rapid application recovery. There are three main RaaS architectural models depending on the location of the source application or data: To-cloud RaaS, In-cloud RaaS and From-cloud RaaS. For high-availability VPS hosting the recovery of VPS on other physical server inside same cluster can be considered as case of the In-cloud RaaS.

One of the most important customer parameters in the SLA is Availability. But often RTO and RPO are also important to customer (Fig. 2).

The Recovery Time Objective (RTO) is the duration of time and a service level within which a business process must be restored after a disaster in order to avoid unacceptable consequences associated with a break in continuity.

The Recovery Point Objective (RPO) is the maximum targeted period in which data might be lost from an IT service due to a major incident.

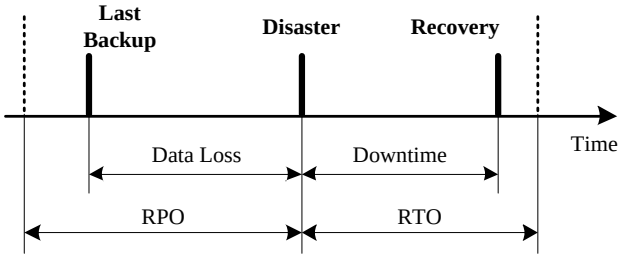

Figure 2: RTO and RPO.

For high-availability VPS hosting the data are stored in the shared storage (which are implemented as a reliable RAID-arrays). So, in case of a physical server failure only current transaction data will be lost and after restoring VM on a different physical server in the cluster there is no need in restoring data from a backup to achieve RPO.

The research problem formulation follows.

2 Problem Statement

Consider the case when the cluster consists of a number of physical servers, which execute VMs under the control of the hypervisor (Fig. 1). All data are stored in the storage area network (SAN) that allows for live migration of VMs between servers for the purpose of load balancing and restarting the VMs on backup server in the case of a physical server failure.

Each VM has parameters such as the requirements for the computational resources (memory, CPU time, disk space, IO subsystems bandwidth, etc.), VM boot time and recovery time objective that was agreed with the customer in the Service Level Agreement (SLA). The problem of the allocation of the server resources for VMs can be divided into three stages.

At the first stage, if possible, each VM is provided with all necessary resources and placed on the server so that in case of server failure, total restarting time of all VMs from

the failed server on a reserve server does not exceed the recovery time objective of any VM from failed server.

It is desirable to place VMs most densely for possible release the physical servers in the cluster and later switch them to the stand-by mode to reduce energy consumption. If such an allocation scheme is found then the problem can be considered as solved.

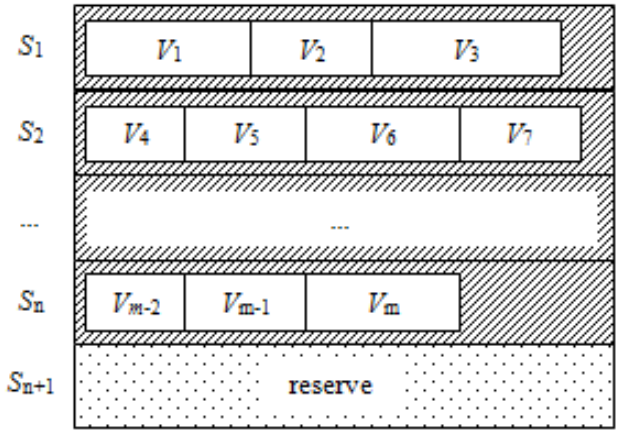

Figure 3: VMs allocation on servers.

If such a scheme could not be found, then the second stage is the problem of finding the location of the VMs to physical servers, where all the VMs will have all the necessary resources, without any guarantee of recovery time objective. Certainly, the problem of release of servers is no longer relevant.

If the second stage has no positive result (the situation is an acute shortage of resources) and at the third stage the problem of resourcing the most important VMs by less important ones is raised. Of course, at this stage both purposes of consolidation of free resources and providing guaranteed time of services restoration is no longer relevant, since there is no extra resources or free backup servers available.

3 Essence of Method

We introduce the following notation:
S_i – physical server in cluster, $i = 1, \ldots, n{+}1$,
e_i – power consumption of server S_i, $i = 1, \ldots, n{+}1$,
R_k – resource of type k (*e.g.* CPU, memory), $k = 1, \ldots, l$,
r_{ki} – amount of resource R_k on server S_i, V_j – virtual machine, $j = 1, \ldots, m$,
w_j – importance weight of V_j, $j = 1, \ldots, m$,
p_{kj} – demands of V_j in resources R_k,
t_j – time required to boot up V_j, $j = 1, \ldots, m$,
T_j – recovery time objective of V_j, $j = 1, \ldots, m$,
x_{ij} – Boolean variable; $x_{ij} = 1$ for V_j placed on S_i, otherwise $x_{ij} = 0$.

Above problems can be formulated as follows.

Problem 1. Firstly, demands in resources of all VMs on server S_i have to not exceed available resources of this server:

$$\sum_{j=1}^{m} x_{ij} p_{kj} \le r_{ki};\ k = 1, \ldots,\ l;\ i = 1, \ldots,\ n \tag{1}$$

Secondly, all VMs should be placed on a server, and each of them should be placed no more than on one server:

$$\sum_{i=1}^{n} x_{ij} = 1;\ j = 1, \ldots,\ m \tag{2}$$

Consider the case of the virtualization management system after the failure of some server automatically deploys all its VMs on the backup server. Let's denote time required to restart all VMs of server S_i on the backup server as

$$\sum_{j=1}^{m} x_{ij} t_j \tag{3}$$

If in the configuration of the virtualization management system user can not directly specify the order in which VMs will be recovered, the only option to ensure that time to recover any VM will not exceed the recovery time objective is to ensure that all VMs on the server will be recovered on time, not exceeding recovery time objective for any of these VMs:

$$\sum_{j=1}^{m} x_{ij} t_j \cdot x_{iz} \le T_z;\ z = 1, \ldots, m;\ i = 1, \ldots, n \tag{4}$$

Let's define the indication of no VMs on server S_i as

$$\prod_{j=1}^{m} \overline{x_{ij}} = 1,\ i = 1, \ldots,\ n \tag{5}$$

Then, in order to minimize power consumption of servers let's use the following criteria

$$\max \sum_{i=1}^{n} e_i \prod_{j=1}^{m} \overline{x_{ij}} \tag{6}$$

and formulate the problem as follows: satisfy (6) under constraints (1), (2), (4).

Problem 2. If no acceptable solution for Problem 1 was found, it will be appropriate to waive constraints (4),

which ensures the recovery time objective. But it is worth trying to find the VMs allocation scheme in which recovery time will not exceed the recovery time objective too much:

$$\min \sum_{i=1}^{n} \sum_{z=1}^{m} \left(x_{iz} \sum_{j=1}^{m} x_{ij} t_j - T_z \right) \tag{7}$$

Formulate the problem as follows: satisfy (7) under constraints (1), (2).

Problems 1 and 2 are non-linear 0-1 programming problems. To solve these problems we use heuristic greedy and genetic algorithms. The use of a genetic algorithm [9] to solve these problems will be appropriate, especially given their dimension. Controlled genetic algorithm [10] fits well for such problems.

Heuristic algorithm. Since we are interested in the densest allocation of VMs on the servers, let's formulate the idea of the algorithm as follows:

while (the list of nonallocated VM has at least one VM)
{
find nonallocated VM with the highest recovery time objective;
try to place this VM to the one of the highest loaded-servers;
}

This simple algorithm is very effective since VM recovery time t_j in most cases is directly dependent on its resource requirements p_{kj}.

Genetic algorithm (GA). Since each VM cannot be placed on more than on one server, for encoding genes let's move from n·m matrix x_{ij} of Boolean variables to the length m vector y_j of discrete variables. Each element of that vector is the server's number $i = 1,\ldots,n$, which contains the appropriate VM. For example:

$$x_{ij} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

$$y_j = \begin{bmatrix} 3 & 2 & 4 & 2 & 1 & 3 & 1 & 1 \end{bmatrix}.$$

This method of gene coding allows, firstly, to reduce the dimension of the problem, and secondly, to provide automatic satisfaction of constraints (2). Herewith, the mutation operation corresponds to VM transferring from one server to another and the crossover operation – to migrating of several VMs between servers [11].

A series of experiments, which differs in the number of VMs and strictness of requirements to the recovery time objective, was carried out. The number of servers m was changed from 10 to 40. Experiments were carried out under the conditions of the soft requirements (RTO soft: $t_j << t_j$) and hard requirements (RTO hard: $t_j < t_j$) to the recovery time objective. All experiments show the advantages of genetic algorithm over the heuristic algorithm at approximately 1–6% (Fig. 4).

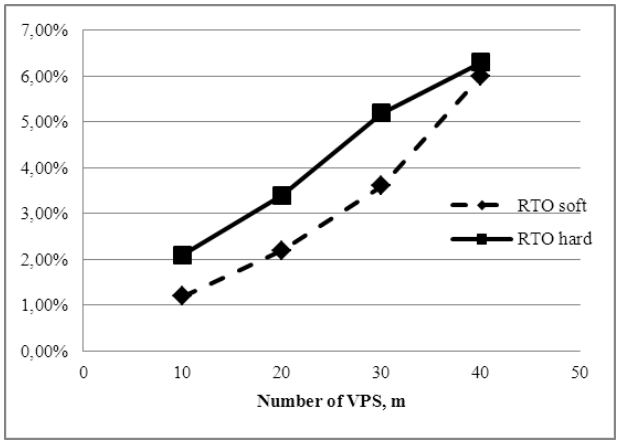

Figure 4: Advantage of genetic algorithm over heuristic algorithm.

Problem 3. If no acceptable solution to Problem 2 was found due to a lack of resources, we have a situation in which it is not possible to place all the VMs under resource constraints (1). In this case it would be appropriate to allocate all needed resources amongst the most important VMs at first phase, and residual resources among the rest of VMs at second phase [8]. So instead of constraints (2), should be used

$$\sum_{i=1}^{n} x_{ij} \le 1;\ j = 1, \ldots, m \tag{8}$$

and criteria

$$\max \sum_{i=1}^{n} \sum_{j=1}^{m} x_{ij} w_j \tag{9}$$

Formulate the problem as follows: satisfy (9) under constraints (1), (8).

This problem is a 0-1 integer linear programming problem. Use the branch and bound method to solve it. First show the reduction of the initial number of possible combinations from 2^{mn} by using the specificity of the problem.

Firstly, according to constraints (2), each VM can be placed on no more than on one server, *i.e.* instead of enumerating 2^n combinations for each VM V_j, we can consider only n+1 options: $y_j = 0, \ldots, n$, where y_j is a number of the

server on which the VM V_j is placed (y_j = 0 for the case when due to lack of resources VM V_j is not placed on any server or resources allocated to it by the residual principle).

Secondly, given the VM resource demands p_{ij} are non-negative numbers, if one of the constraints (1) is not satisfied at some stage of exploring the search tree, we can stop exploring this branch because all its solutions will not satisfy this constraint too.

Thirdly, it should be noted that typically to create the cluster, servers with identical configurations are used. This can be used to reduce the total number of the possible schemes of allocation of VMs on the servers, as shown below.

For example, if there are two identical servers in the cluster, the number of possible schemes can be reduced by 2 times because placing all VMs from the first server onto the second and all VMs from the second server onto the first neither affect the amount of resources used nor the amount of residual resources on these servers. In other words, switching of rows i_1, i_2 in matrix $X = ||x_{ij}||$, which correspond to the identical columns i_1, i_2 in matrix $R = ||r_{ki}||$, affects neither the satisfaction of constraints (1), (2) nor the criteria. Accordingly, for servers with the same configuration the important thing is not the absolute placement of VMs on these servers, but their relative location.

When the number of servers with the same configuration grows, the number of unnecessary "mirrored" combinations increases significantly from (n-1) in the case of placing all the VMs on the same server to (n! -1) in case of placing each VM on different server.

To simplify the description of algorithm for exploring of the search tree with exclusion of "mirrored" solutions let's name VMs as a letter ("A", "B", ...), servers as a number ("1", "2", ...) and allocation of VM on server as a combination of this letter and this number ("A1", "B2",...).

For example (fig. 5), in the case of three different VMs ("A", "B", "C") and three identical physical servers ("1", "2", "3") there are 27 possible VMs allocation schemes, only 5 of which are unique (A1-B1-C1, A1-B1-C2, A1-B2-C1, A1-B2-C2, A1-B2-C3) and the rest 22 schemes (written to the right of corresponding unique scheme) are "mirrored" to them and can be excluded from enumerating without degradation of results. So, even for such small values of m and n, the initial set of combinations can be reduced more than 5 times.

Let's formulate idea of algorithm for exploring of the search tree with exclusion of "mirrored" solutions. We start with allocating VM "A" on server "1" (in a real case, we can allocate first VM on its current server to minimize the number of the migrations of the VMs). We don't consider the allocation of VM "A" on servers "2" and "3" since all servers have same amount of resources which provide same result. On the next fork of the current branch (allocation of the next VM) the number of possible servers should be l+1 but no more than n, where l is maximum number of servers in the current branch and n is total number of servers. So, for VM "B" we need to consider two options: "A1-B1" and "A1-B2". We don't consider the scheme "A1-B3" as it gives the same result as scheme "A1-B2" (Fig. 5).

If it is not possible to place all the VMs due to a lack of resources, we have to consider schemes, in which resources for some VMs will be not allocated, *i.e.* those VMs will be placed on non-existing server "0" (Fig. 6).

Before enumerating the possible solutions of a branch, the branch is checked against upper and lower estimated bounds on the optimal solution, and is discarded if it cannot produce a better solution than the best one found so far by the algorithm. If on some stage of enumer-

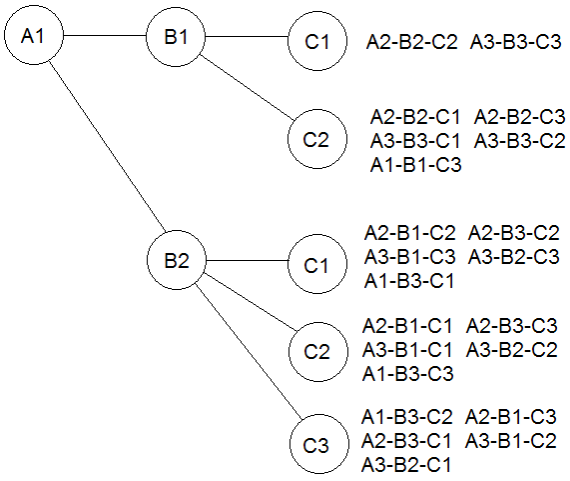

Figure 5: Example of branching with uniqe and "mirror" solutions.

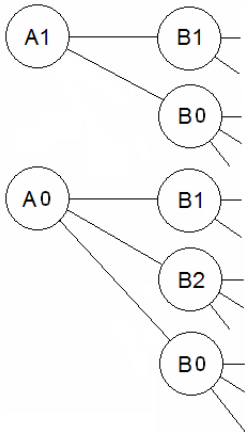

Figure 6: Branching in case of a lack of resources.

ation the current solution does not meet the requirements of constraints, the rest of branch can be discarded, because all VM's requirements are non-negative.

Let's define the procedure of estimating the upper and lower bounds of optimal solution in the current branch.

For estimating the upper bound let's use optimistic prediction which is that the rest of nonallocated VMs, can be placed on physical servers (not on non-existing server "0") under constraints (1), (2). For estimating the lower bound we can use the greedy algorithm under constraints (1), (8).

If on some stage all VMs will be successfully placed (*i.e.* there is no VMs on non-existing server "0") the solution is found and the search can be stopped because the global optimum is found and the criteria (9) is at maximum value.

However, depending on the availability of time, continuing the search can help to find a solution that will provide fewer VMs migrations for implementation of a new plan of VMs allocation than first solution and its "mirrors".

Here is branching algorithm described in Java-style pseudo code:

```
int n;  // number of servers
int m;  // number of VMs
int[] solution = new int[m + 1];  // location of VMs
        // solution[j] is server, where VM j is located
        // solution[] is 1-based array [1..m]

void main() {
   branch(1, 0); // start of recursion
}

void branch(int vm, int level) {
   for (int i = 1; i <= min(n, level + 1); i++) {
      solution[vm] = i;
      if (checkSolution() == false) {
         return;
      }
      if (vm < m) {
         branch(vm + 1, i);
      } else {
        rememberSolution();
        // or exit, if we need only one solution
      }
   }
}
```

In order to study the effectiveness of the improved algorithm with exclusion of "mirrored" solutions a series of experiments was conducted. Results of these are shown in Table 1.

Table 1: Effectiveness of exclusion of "mirrored" solutions.

Number of servers, n	Number of VMs, m	Average number of analyzed schemes	
		Without exclusion of "mirrored" solutions	*With exclusion of "mirrored" solutions*
4	16	1758080	174237
4	12	82380	7789
4	8	3928	333
4	4	164	10

As seen from the results, exclusion of "mirrored" solutions can significantly reduce the computational workload. But for large clusters and increasing the number of servers and VMs the dimensions of the problem increases significantly and other methods such as genetic algorithms may be much more effective.

4 Conclusions

Models and methods for solving the problem of resource allocation for server virtualization in the limitations of recovery time objective were proposed. Formulated problems were reduced to problems of 0-1 programming. Branch and bound, heuristic and modified genetic algorithms were proposed. Results of the experiments confirmed the efficiency of the proposed approach for service providers. The proposed models can be used in data centers of VPS service providers and other organizations that are using server virtualization.

References

[1] Telenyk S.F., Rolik A.I., Pokotylo A.A. Resource allocation and load management considering the assessed quality of the provided service and the use of agent technologies. Proc. of 23rd International Crimean Conference Microwave and Telecommunication Technology (CriMiCo), IEEE Conference Publications, 2013. pp. 535–536.

[2] Drewanz D. Resource management as an enabling technology for virtualization. eSTEP Blog. 2012. [Online] Available: https://blogs.oracle.com/eSTEP/entry/virtualization_oracle_part_5_resource.

[3] Telenyk S.F., Rolik O.I., Bukasov M.M., Kosovan O.A., Kobetc O.I. Models and methods of resource allocation in systems with server virtualization. Collection of research works of MITI NTUU "KPI". vol. 3, pp. 100–109, 2009 (in Russian).

[4] Telenyk S.F., Rolik O.I., Bukasov M.M., Labunskij A.Y. Models of virtual machine management for server virtualization. Visnyk NTUU "KPI": Informatics, operation and computer science. vol. 51, pp. 147–152, 2009 (In Ukrainian).

[5] Kimbrel T., Steinder M., Sviridenko M., TantawiA. Dynamic application placement under service and memory constraints. Proc. of Int'l Workshop on Efficient and Experimental Algorithms, Santorini Island, Greece, May 2005, pp. 1–12.

[6] Guthrie F., Lowe S., Saidel-Keesing M. VMware vSphere Design. John Wiley & Sons, 2011.

[7] DRS Performance and Best Practices. VMware Inc. 2008. [Online] Available: https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/techpaper/drs_performance_best_practices_wp.pdf

[8] Telenyk S.F., Rolik O.I., Bukasov M.M., Sokolovskiy R.L. Management system for corporate IT-infrastructure and telecommunication network. Visnyk NTUU "KPI": Informatics, operation and computer science. vol. 45, pp. 112–126, 2006 (In Ukrainian).

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

[10] Telenyk S.F., Rolik O.I., Bukasov M.M.,Androsov S.A. Genetic algorithms for problems of resource and workload management in datacenters. Automation, Automatization, Electrical complexes and systems. vol. 1(25), pp. 106–120, 2010 (In Ukrainian).

[11] Telenyk S., Rolik O., Bukasov M., Halushko D. Models and methods of resource management for VPS hosting. Technical transaction. Automatic control. Politechnica Krakowska. vol. 4-AC, pp. 41–52, 2013.