An Effective Scheduling Method in the Cloud System of Collective Access, for Virtual Working Environments

Leonid V. Legashev, Irina P. Bolodurina

Orenburg State University, Pobeda avenue 13, 460018 Orenburg, Russia,
ais@mail.osu.ru, prmat@mail.osu.ru

Abstract: This paper describes the cloud system of collective access to virtual working environments as a means of providing remote access to paid and free software, for students of educational institutions in secondary education. The problems of efficient cloud system scheduling and optimizing the usage of cloud virtual machines and paid software licenses has been studied in detail, herein. In addition, the mathematical model of cloud system resources control is presented, as well as, the functional model. Two evolutionary scheduling algorithms are proposed. Also, the UML class diagram of the cloud system simulator is described. The statistical analysis of the fitness function value distribution, to evaluate the proposed algorithms, is also performed.

Keywords: desktop as a service; cloud computing; scheduling; simulated annealing; genetic algorithm, statistical analysis

1 Introduction

Currently, cloud computing is one of the most popular technology solutions for data processing, various calculations on cloud servers, and services providing. Many educational institutions of secondary education are facing problems with regular renewal of the computer park and software purchase, installation and configuration. Taking into account the existing restrictions on the number of software licenses and the duration of user's sessions, the development of efficient methods of using the resources of the cloud system is an urgent task to be solved.

One of the solutions is to create a cloud system of collective access (CSCA) to virtual working environments [1], as shown in Figure 1. This system is based on the DaaS (Desktop-as-a-Service) scheme [2]. Users get access to virtual desktops with installed software via Internet browser. Each user forms one request from the educational organization. The user specifies a required number of virtual machines (VM) instances, choose required software from the list of available software and

sets a working schedule of organization. Data Storage System (DSS) contains the VM images and license servers for paid software. A materials database stores educational-methodical materials information, concerning the subjects of the school courses. Resources administration module is used to find a close-to-optimal schedule.

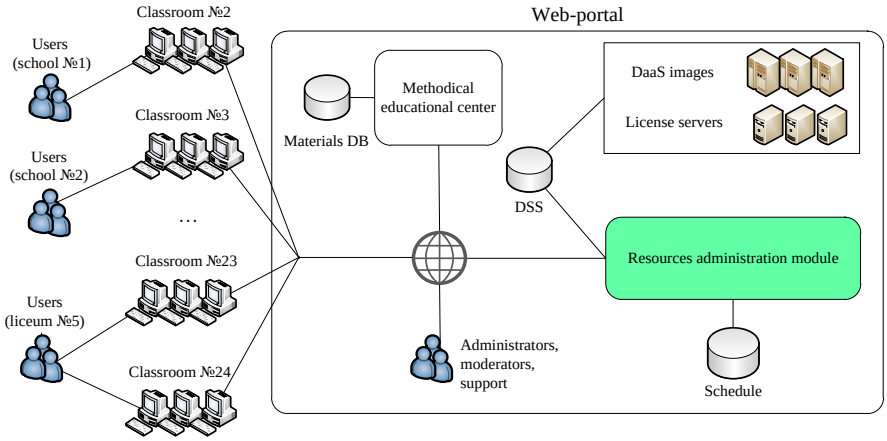

Figure 1
CSCA architecture

Administrators and moderators provide the technical support. To experience CSCA functionality, the end users only needed low-cost computers with Internet access. Collective usage of cloud system leads to reduced costs of the software and computer components purchase and leads to opportunity to use modern software in the educational process.

There is a great deal of research on scheduling problems, virtual machine allocation and cloud technologies using in the educational process. In paper [3], a scheduling algorithm that optimizes virtual machine placement across multiple clouds is described. The purpose of the algorithm is to allocate *n* virtual machines between *m* available clouds improving criteria such as performance, cost, or load balancing. The authors of paper [4] considered a problem of increasing energy consumption in a cloud server environment. The paper presented three heuristic scheduling algorithms of virtual machines allocation to minimize energy consumption in the memory system. An algorithm based on genetic programming as a meta-heuristic method for solving static scheduling problem in heterogeneous computing systems proposed in [5]. The authors of paper [6] proposed the usage of cloud computing technologies in the business processes of universities. The significant impact of cloud computing on input-output data quality dimensions of universities collaborative processes is shown. The authors of paper [7] described load balancing of virtual machines resources in cloud environments. Authors proposed scheduling strategy of virtual machines placement based on genetic

algorithm. Software as a Service (SaaS) model in the Chinese education system is described in paper [8]. Students get access to the office suite of applications, libraries and cloud applications (such as GoogleApps and Zoho Office). The disadvantage is in the usage of open-source software only. The authors of paper [9] introduced background and fundamentals about emerging of three technology paradigms – Cloud Computing, Internet of Things, and Connectivism – in the educational sector. Analysis of the existing literature shows the lack of efficient solutions for the CSCA resources administration within the constraints of corresponding time slots and software licenses.

It should be noted, there are many solutions for implementing remote access to information resources. Amazon company offers its own solution Amazon WorkSpaces, which provides Desktop as a Service mechanism and allocates virtual desktops for users based on Microsoft Windows operating systems. Amazon WorkSpaces is not suitable as a turnkey solution of the problem due to the need to set up a virtual desktop for each user individually. Two alternative solutions are offered based on Microsoft Azure Remote Desktop Services – MyCloudIT platform and Workspot solution. Virtual desktops based on MyCloudIT and Workspot come with pre-installed software (Microsoft Office, Outlook, Skype, Adobe Reader, Mozilla Firefox, Internet Explorer). We can also mention Citrix XenDesktop, VMware Horizon, Quest vWorkspace as existing platforms for implementing the Desktop as a Service scheme. They are not suitable too as a turnkey solution of the problem for several reasons: high cost, limited ability to function in a browser, the usage of simplistic administration algorithms, the lack of software license constraints and collective usage in the specifics of educational institutions. TSPlus Server [10] technology is chosen to implement the DaaS mechanism for our CSCA. The main advantages of using the cloud systems are as follow:

- Economic benefits. Organizations do not need to buy expensive paid software and modern computers. They rent only actual services of the cloud system.
- End-user data storage. Various user data, which is necessary for work (documents, files, projects, databases, etc.), are stored in the network storage of the cloud system.
- Broad access to educational software. A larger list of paid and open source software is available for students of educational institutions to perform virtual experiments, mathematical calculations, modelling, machine learning, rendering, etc.

Implementation of the CSCA faces the NP-complete problem of requests placement in the schedule within the constraints of corresponding time slots and limited software licenses.

2 Mathematical Model of Cloud System Resources Control

The cloud system mathematical model is developed to describe the basic stages of operation of the system and to provide DaaS model experience to end-users.

Cloud system of collective access to virtual working environments is formalized as tuple $C_{cloud} = \{Z_{temp}, S_{soft}, R_{req}, U_{users}, F_{flav}, D_{data}, \hat{S}\}$, where its elements are defined as follows: $Z_{temp} = \{Z^1_{temp}, Z^2_{temp} \ldots, Z^i_{temp}\}$, $i \in N$ – the set of scheduling templates, $S_{soft} = \{S^1_{soft}, S^2_{soft}, \ldots, S^j_{soft}\}$, $j \in N$ – the set of accessible software, $R_{req} = \{R^1_{req}, R^2_{req}, \ldots, R^n_{req}\}$, $n \in N$ – the set of requests, U_{users} – the number of users, $F_{flav} = \{F^1_{flav}, F^2_{flav}, \ldots, F^m_{flav}\}$, $m \in N$ – the set of virtual machines flavors, $D_{data} = \{D^1_{data}, D^2_{data}, \ldots, D^p_{data}\}$, $p \in N$ – the set of datacenters, $\hat{S}$ – the set of schedules.

Schedule template is defined as tuple $Z^i_{temp} = \{Z^i_{title}, Z^i_{length}, Z^i_{\text{intervals}}\}$, where Z^i_{title} is the template title, $Z^i_{length} \in N$ – the duration of working session in minutes, $Z^i_{intervals} = \{[z_r, z_r + Z^i_{length}] | z_r \in \{0,1,\ldots,1439\}, r \in N\}$ – the set of schedule time slots, z_r – the beginning of the working session. The interval [0,1439] corresponds to a twenty-four-hour time segment (00:00-23:59).

Free and paid software of the CSCA is defined as tuple $S^j_{soft} = \{S^j_{OS}, S^j_{title}, S^j_{license}, S^j_{install}, S^j_{delete}\}$, where its elements are defined as follows: S^j_{title} – the software title, $S_{OS} \in \{OS_1, OS_{2,}, \ldots, OS_o\}, o \in N$ – the supported operating system, $S^j_{license}$ – the number of licenses, $S^j_{install}$ – the software installation time, S^j_{delete} – the software uninstallation time. Virtual machine flavor is formalized as tuple $F_{flav} = \{C_m, M_m, D_m\}, m \in N$, where C_m – is the number of CPU cores, M_m – the RAM size in Gb, D_m – the disk volume in Gb.

Each user request is defined as tuple $R^k_{req} = \{Z^{i'}_{temp}, R^k_{count}, \hat{S}^k_{soft}, t^k_{arrive}, T_k, \hat{T}_k, w_k, R^k_{status}\}, k = \overline{1,n}$, where its elements are defined as follows: $Z^{i'}_{temp} \in Z_{temp}, i' = 1,..,i$ – the chosen schedule template, R^k_{count} – the number of virtual machines, $\hat{S}^k_{soft} \subseteq S_{soft}$ – the subset of software,

t^k_{arrive} – the request arrival time, $T_k \subseteq Z_{intervals}$ – the required time slots, $\hat{T}_k \subseteq Z_{intervals}$ – the picked time slots, w_k – the weight coefficient, $R^k_{status} = \{-1,0,1\}$ – the request status. R^k_{status} is equal to "1" in case, when user request sets in the schedule at required time, $\hat{T}_k \equiv T_k$. Otherwise, R^k_{status} is equal to "0", and user get access to the virtual machines at different time, $\left(\hat{T}_k \neq T_k\right) \wedge \left(\left|\hat{T}_k\right| = \left|T_k\right|\right)$. If there is no possibility to satisfy user request within current constraints of cloud system, then R^k_{status} is equal to "–1", and $\hat{T}_k = \varnothing$. Cloud system schedule *S* is represented as follow:

$$S = \begin{vmatrix} R^1_{status} & \hat{T}_1 & \hat{S}^1_{soft} & R^1_{count} & T_1 & \hat{Z}^1_{temp} \\ R^2_{status} & \hat{T}_2 & \hat{S}^2_{soft} & R^2_{count} & T_2 & \hat{Z}^2_{temp} \\ R^3_{status} & \hat{T}_3 & \hat{S}^3_{soft} & R^3_{count} & T_3 & \hat{Z}^3_{temp} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ R^n_{status} & \hat{T}_n & \hat{S}^n_{soft} & R^n_{count} & T_n & \hat{Z}^n_{temp} \end{vmatrix}. \tag{1}$$

Each row of schedule *S* corresponds to single request R_{req}. Generated requests added to the queue with FCFS (First-Come, First-Served) service disciplines. Scheduler maximizes value of fitness function, which is described by equation (2):

$$F(S) = \sum_{k=1}^{n} w_k x_k(S) \to \max_{S \in S}, \tag{2}$$

$$\forall S \in \tilde{S},\ \forall k = \overline{1,n}: \quad x_k(S) = \begin{cases} \alpha \cdot R^k_{count}, \text{ if } \hat{T}_k \equiv T_k, \\ \beta \cdot R^k_{count}, \text{ if } \left(\hat{T}_k \neq T_k\right) \wedge \left(\left|\hat{T}_k\right| = \left|T_k\right|\right) \\ -\gamma \cdot R^k_{count}, \text{ if } \hat{T}_k = \varnothing. \end{cases}$$

Constraints on paid software licenses are described by the following inequalities:

$$\forall j = 1..|S_{soft}|, \forall l = [t_{start}, t_{end}]: G_{S,j,l} \leq S^j_{license}, \tag{3}$$

where $G_{S,j,l}$ is the number of virtual machines running the j^{th} software at the l^{th} time interval according to the schedule *S*, t_{start} – the launch time of virtual machines; t_{end} – their shutdown time.

Constraints on the number of placed virtual machines are described by equation (4):

$$\forall k = \overline{1,n}: H(R^k_{req}) = R^k_{count}, \tag{4}$$

where $H(R^k_{req})$ is the number of placed VMs.

Functional model (Figure 2) of the CSCA is based on mathematical model. It describes the main operations within the cloud system from the moment of requests generation to the moment of virtual machines assignment.

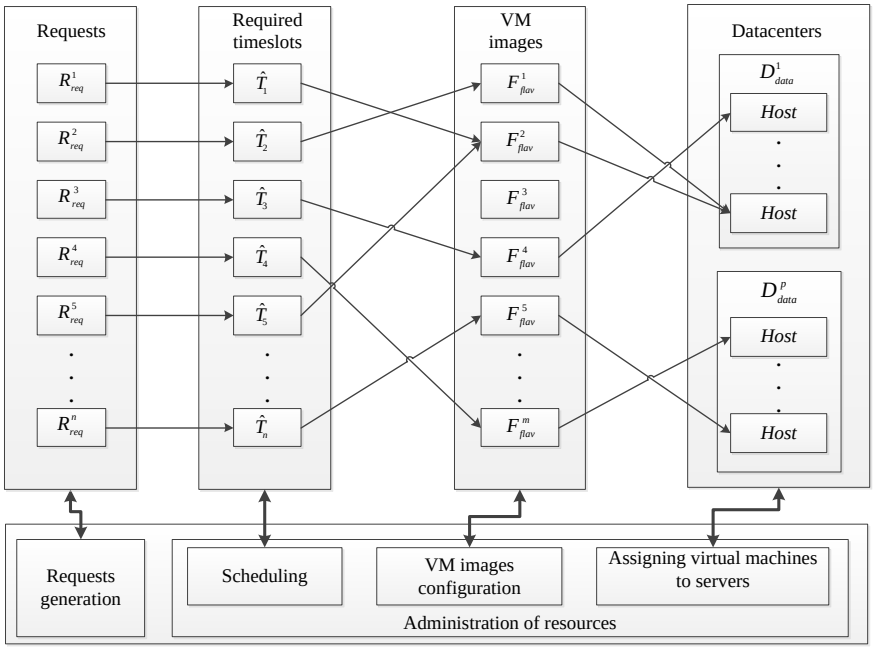

Figure 2
Functional model of the CSCA

Administration of cloud system resources denotes the sequential solution of three tasks:

- Virtual working environments scheduling. End-user requests R_{req} are placed to the schedule *S* to maximize the fitness function value (2) under constraints (3) and (4).
- Dynamic creation of virtual working environments F_{flav} according to the current schedule of CSCA.
- Selection of the physical servers of the cloud provider for launching the required virtual working environments at the specified time periods.

To satisfy constraints (3) and (4) during scheduling operation we define the schedule matrix *SWCheck* and initialize it with zeros (Figure 3). Each row of *SWCheck* corresponds to single software, and each column – to a single minute of time.

$$SWCheck = \begin{vmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 \end{vmatrix}$$

Figure 3
Schedule matrix initialization

For each request in queue in all required time slots, the R^{k}_{count} virtual machines are adding to corresponding elements of matrix *SWCheck*, as it shown in Figure 4.

$$SWCheck = \begin{vmatrix} 15 & 15 & 15 & 15 & 15 & 15 & 15 & 15 & 15 & 15 & 15 & 15 & 15 & \dots & 0 \\ 0 & 0 & 0 & 0 & 0 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & \dots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 10 & 10 & 10 & 10 & 10 & 17 & 17 & 17 & 17 & 17 & 17 & 17 & 17 & \dots & 0 \end{vmatrix}$$

Figure 4
Schedule matrix filling

If the total number of licenses of some software exceeds the maximum number of licenses for a certain period of time (Figure 5), the user virtual machines are subtracted from the schedule matrix.

$$SWCheck = \begin{vmatrix} 15 & 15 & 15 & 15 & 15 & 15 & 15 & 15 & 15 & 15 & 15 & 15 & 15 & \dots & 0 \\ 0 & 0 & 0 & 0 & 0 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & \dots & 0 \\ 12 & 12 & 12 & 12 & 23 & 23 & 23 & 23 & 23 & 11 & 11 & 11 & 11 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 10 & 10 & 10 & 10 & 10 & 17 & 17 & 17 & 17 & 17 & 17 & 17 & 17 & \dots & 0 \end{vmatrix}$$

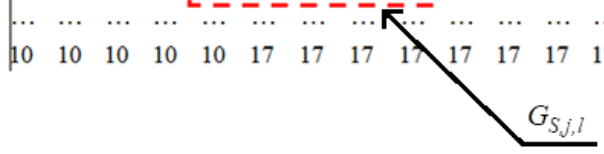

Figure 5
Excess of maximum number of licenses for 5 minute intervals

3 Scheduling Algorithms

3.1 Simulated Annealing Technique

The scheduling problem is NP-complete. We cannot use exhaustive search methods to solve the problem under consideration due to its exponential complexity $O(n^k)$ (where k is the number of requests, n – the number of combination without repetitions from all time slots of the schedule template by the number of required time slots). We have implemented two evolutionary algorithms: Simulated Annealing (SA) and Genetic Algorithm (GA), to solve this problem in polynomial time. The initial schedule is created by using the Round-Robin (RR) method of cyclic load distribution which distributes requests over the initial schedule in a uniform manner, and the primary assignment of request statuses is occurred.

The Simulated Annealing is a stochastic algorithm, which allows finding a close-to-optimal, solution of the scheduling problem and avoids a local maximum of the fitness function value (2). The parameters of the SA algorithm are presented in Table 1.

Table 1

Simulated annealing algorithm parameters

№	Parameter	Value
1	Maximum temperature	t_{max}=100
2	Temperature step	t_{step}=0.9
3	Mutation probability of schedule	0.01
4	Temperature breakpoint	0.05
5	Reward/penalty coefficients	$\alpha,\ \beta,\ \gamma$

In each iteration of the SA algorithm, the current schedule mutates randomly, and the value of fitness function is calculated. The current schedule is accepted as the best solution if the fitness function value is greater than the previous one. In case when the fitness function value is less than the previous one, the current schedule is also accepted as the best solution with acceptance probability function (5):

$$P = e^{-\left|\frac{F(s_{current})-F(S_{best})}{t_{curr}}\right|}. \quad (5)$$

At the first iterations of the SA algorithm, the acceptance probability function (5) will be quite high. As a consequence, the “worst” schedule, will often be chosen as an optimal schedule. This feature prevents the SA algorithm from becoming stuck at a local maximum of the fitness function value (2).

3.2 Genetic Algorithm

Another way to solve the scheduling problem is to use genetic algorithms. Each individual in the genetic algorithm represents one of the schedule options. We can depict it as columns of request statuses (Figure 6).

$$\begin{vmatrix} 1 & -1 & 1 & \cdots & 1 \\ 1 & 1 & 1 & \cdots & -1 \\ -1 & 1 & -1 & \cdots & -1 \\ 1 & 1 & 1 & \cdots & 1 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ -1 & -1 & 1 & \cdots & -1 \end{vmatrix}$$

Figure 6
Initial population of schedules

The parameters of the genetic algorithm are presented in Table 2. The initial population size corresponds to the number of requests. To ensure the diversity of the initial schedule options, we apply the Round-Robin method to do the random permutations of requests in the queue.

Table 2
Genetic algorithm parameters

№	Parameter	Value
1	Initial population size	$\lvert R_{req}\rvert$
2	Temporary population size	$2\cdot\lvert R_{req}\rvert$
3	Crossover ratio	100 %
4	Mutation ratio	50 %
5	Mutation probability of schedule	0.01
6	Selection ratio	50 %
7	Number of epochs	20
8	Reward/penalty coefficients	$\alpha,\ \beta,\ \gamma$

The crossover operator provides the inheritance by a child of optimal VM assignments from both parents within the previous population as it shown in Figure 7.

$$\begin{array}{ccccc} Parent_1 & & Parent_2 & & Child \\ 1 & \times & 1 & \rightarrow & 1 \\ 1 & \times & -1 & \rightarrow & 0 \\ -1 & \times & -1 & \rightarrow & -1 \\ 1 & \times & 1 & \rightarrow & 1 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ -1 & \times & 1 & \rightarrow & 1 \end{array}$$

Figure 7
Crossover operation

The mutation operator changes a single schedule, which is randomly chosen from the current population (Figure 8).

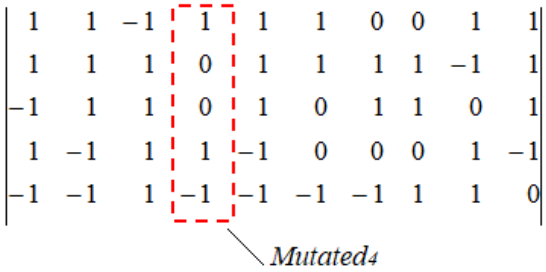

Figure 8
Mutation operation

During the selection operation the individuals with the maximum fitness function value (2) are saved as the base for the next population as shown in Figure 9. When the epochs' number is reached, the first individual will represent the best close-to-optimal schedule *S*.

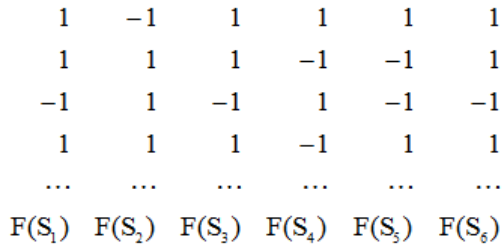

Figure 9
Selection operation

4 Experimental Research

To carry out the experiments, we implemented the CSCA simulator, using C++ language and Visual Studio 2012 IDE. To correctly generate the parameters of the cloud system and the requests we reviewed the needs of virtual resources for educational institutions of the Orenburg region. We also performed a statistical analysis of the distribution of fitness function to evaluate proposed scheduling algorithms.

4.1 CSCA Simulator

Figure 10 shows the UML class diagram, which depicts classes and relationships between them. Classes and methods correspond to the mathematical model of the CSCA.

The **ScheduleTemplate** class has field 'Intervals', which contains a set of time slots for each schedule template. The methods setDate() and getDate() are used for time slot write/read operations. The **Software** class contains information about software available in CSCA such as the software index (id), the supported operating system, the software title, the maximum number of licenses, and the software installation and uninstallation time.

The **Request** class represents the end-user request. Its TemplateIndex attribute refers to the chosen schedule template, SoftwareSet attribute is a subset of required software. The method setSoftware() randomly generates a subset of software according to the user request. The method setTimeSlots() randomly generates the time slot indices according to the user request. CloneT() is used to copy time slots indices, CloneS() – to copy software indices. The method SoftwareCheck() of the **RoundRobin** class is used to create initial schedule for CSCA.

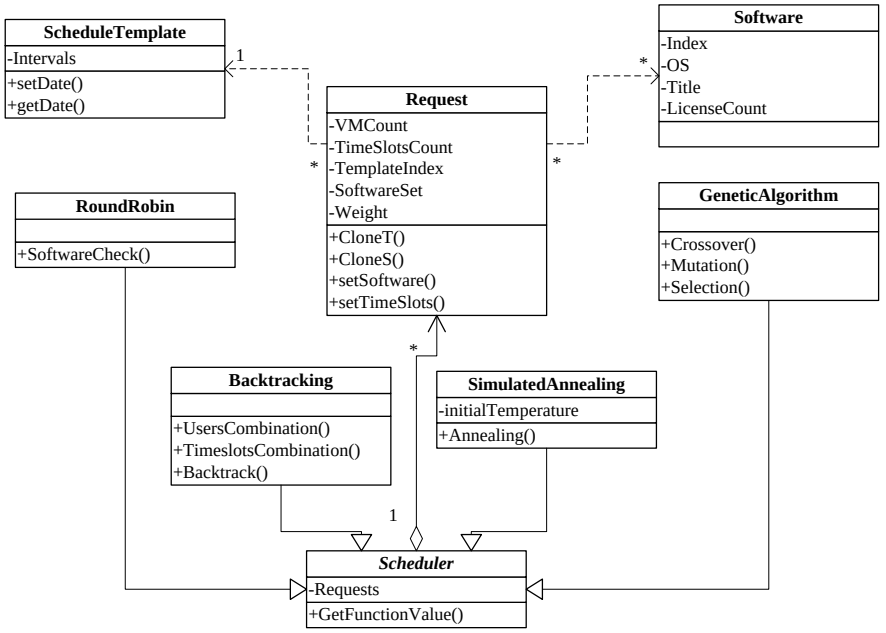

Figure 10
UML class diagram

The **Backtracking** class describes exhaustive search implementation, which is used to estimate the efficiency of proposed simulated annealing and the genetic algorithms. Its methods UsersCombination() and TimeSlotsCombination() are used to build all the possible combinations of requests placement in the CSCA schedule.

The method Annealing() of the **SimulatedAnnealing** class creates a close-to-optimal schedule of the CSCA. The field 'initialTemperature' is used to control the number of iterations of SA algorithm. The methods Crossover(), Mutation() and Selection() are used to implement corresponding operations of the **GeneticAlgorithm** class.

Scheduler is a parent abstract class for other classes, which implements corresponding heuristic scheduling algorithms: Round-Robin, Backtracking, Simulated Annealing and Genetic Algorithm. The GetFunctionValue() method is used to calculate the fitness function value (2).

150 experiments of requests generation and scheduling were performed within CSCA simulator for a large number of users. Examples of generated requests are shown in Figure 11.

```
                    UsersCount = 15

VM count: 5                     VM count: 10
Template index: 0               Template index: 1
0 1 2 3 4 5 6 7 8 9             0 1 2 3 4 5 6 7 8 9
1 2 7 8 4 5 3 0 6 9             9 6 8 5 1 7 3 0 4 2
Software count: 4               Software count: 4
Software indexes: 1 2 8 4       Software indexes: 9 5 7 3
Timeslots indexes:5             Timeslots indexes:7 0 3

VM count: 5                     VM count: 8
Template index: 0               Template index: 0
0 1 2 3 4 5 6 7 8 9             0 1 2 3 4 5 6 7 8 9
6 0 3 1 8 5 9 2 4 7             5 1 4 7 2 9 0 3 8 6
Software count: 1               Software count: 1
Software indexes: 6             Software indexes: 5
Timeslots indexes:4             Timeslots indexes:6

VM count: 13
Template index: 0
0 1 2 3 4 5 6 7 8 9
1 0 7 9 2 3 5 6 8 4                     ...
Software count: 1
Software indexes: 7
Timeslots indexes:0 4
```

Figure 11
Generated requests

The values of reward coefficients α, β and the penalty coefficient γ obtained empirically during the testing of the simulator. These values are interdependent quantities. When choosing the coefficients of the fitness function, we are guided by the following rule: $\alpha + \beta + \gamma = 2$. The worst case is the rejection of the user request for access to information resources, therefore the penalty coefficient has the greatest value of all three coefficients. However, the successful placement of the user virtual machines to the schedule is generally prevalent, and placing the virtual machines at the required time is the most preferable option. Thereby, the reward/penalty coefficients are chosen as follow: $\alpha = 0.7,\ \beta = 0.4,\ \gamma = 0.9$.

The simulated annealing algorithm satisfies 89.7% of generated requests in average. It is 1.88 times more efficient than the Round-Robin method. In addition, the genetic algorithm satisfies 86.5% of generated requests on average. And it is 1.48 times more effective than the Round-Robin method. The average execution time of the Simulated Annealing algorithm is 35 ms, of the genetic algorithm – 210 ms.

4.2 Statistical Analysis

The values of fitness function are varying in the range from -40 to 80. We divide this interval into 24 segments, each of length 5, and calculate the hit frequency of the value of fitness function (2) in each segment for the proposed heuristic algorithms as shown in Figure 12.

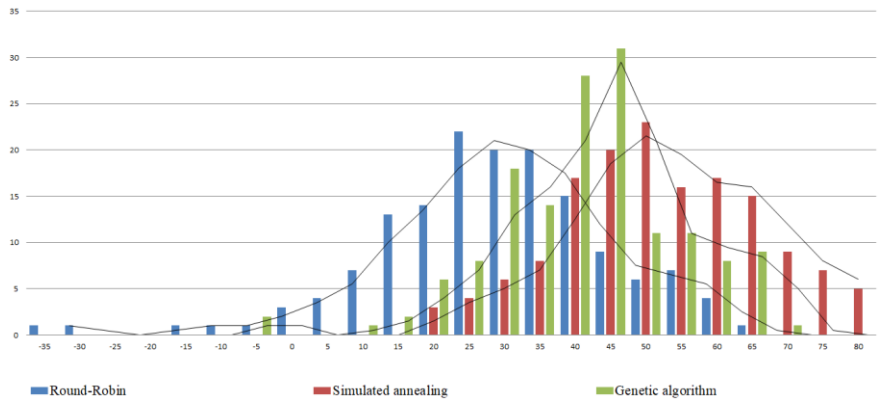

Figure 12
Interval distribution of value of fitness function (2)

After analyzing the graphs, we put forward a hypothesis H_0, that the observed frequency distribution of the value of fitness function (2) is a normal distribution.

Pearson's chi-squared statistical test [11] (χ^2) is used to accept or reject this hypothesis at significance level $\alpha_P = 0.05$ and the number of degrees of freedom $K = 24\text{-}2\text{-}1 = 21$. The test values are compared to one-sided critical value, which is equal to $K_{crit} = \mathbf{32.7}$ for the corresponding α_P level and the K number. Table 3 shows the statistical test results.

Table 3
Pearson's chi-squared statistical test (χ^2)

Algorithm	$\overline{X}$	σ	K_{obs}
Round-Robin	26.26	16.52	124.39
Simulated annealing	49.36	13.91	**6.82**
Genetic algorithm	38.73	13.26	87.33

Hypothesis H_0 is accepted for SA algorithm data set since the test statistic is less than the critical value of χ^2, and it is rejected for both the GA and RR algorithms because the test statistic greatly exceeds the critical value of χ^2.

In both cases, there are outliers in univariate data sets. Grubbs's test [12] is used to detect such outliers and reject them. To test whether the minimum value x_{min_obs} is an outlier, the Grubbs' test statistic is:

$$\tau_1 = \frac{\overline{X} - x_{\min\_obs}}{\sigma} \tag{6}$$

The critical value of the τ-distribution with a significance level of α_P is $\tau_{crit} = 3.81$. For both the GA and RR algorithms the x_{min_obs} and x_{min_obs+1} values were detected as outliers and rejected. Table 4 shows the χ^2-test results for the corrected data sets. Hypothesis H_0 is accepted for the Round-Robin and Genetic Algorithms since the test statistic is less than the critical value of χ^2.

Table 4
Grubbs' test to detect outliers in a univariate data set

Algorithm	$\overline{X}$	σ	K_{obs}
Round-Robin	27.09	15.00	**8.11**
Genetic algorithm	39.35	12.20	**17.96**

The hypothesis H_0 is accepted for all three data samples of fitness function value distribution, therefore, we can use Student's t-test [13] for independent samples with unequal variances (also known as Welch's t-test). Statistic t_e is defined by the following formula:

$$t_e = \frac{\overline{X_1} - \overline{X_2}}{\sqrt{\frac{\sigma_1}{n_1} + \frac{\sigma_2}{n_2}}} \tag{7}$$

Table 5 shows the significant differences between the data samples of proposed evolutionary algorithms. It can be concluded that the simulated annealing algorithm with the highest expected value is the most efficient scheduling solution for implementation.

Table 5
Student's *t*-test for independent samples with unequal variances

Data sample № 1	Data sample № 2	t_e	t_{crit}
Round-Robin	Simulated annealing	13.28	1.96
Round-Robin	Genetic algorithm	7.71	
Simulated annealing	Genetic algorithm	6.60	

Conclusions

The main results of this work are the following:

- The cloud system of collective access as the means of providing an economically profitable remote access to paid and free software was implemented for educational institutions of secondary education of the Orenburg region.
- The approach in solving the problem of efficient scheduling is described in detail.
- A new mathematical model which formalizes the process of cloud system operation, time limits and software license limits is developed.
- Two evolutionary scheduling algorithms based on simulated annealing and genetic programming are proposed.
- The UML class diagram of the CSCA is presented to give a proper look at finding a close-to-optimal schedule of a cloud system.
- The statistical analysis of the distribution of fitness function values is performed. Proposed simulated annealing algorithm shows the best results in experiments and was chosen to be implemented as a part of the Resources administration module of CSCA architecture.
- Machine learning prediction models (such as SARIMAX, Prophet, etc.) can be used to improve end-user request generation accuracy. See our paper [14] for more relevant information.
- The obtained results can be used in future research, to create multi-cloud shared access systems aimed primarily at providing access to virtual working environments with installed software.

Acknowledgement

This work was supported by the Russian Foundation for Basic Research (project 20-07-01065) and the the Russian Federation Presidential grants for support of leading scientific schools (NSh-2502.2020.9).

References

[1] Shukhman A. E., Bolodurina I. P., Polezhaev P. N., Ushakov Yu. A., Legashev L. V.: "Creation of regional center for shared access to educational software based on cloud technology" In: Selected Papers of the XI International Scientific-Practical Conference Modern Information Technologies and IT-Education SITITO-2016, 2016, pp. 180-188

[2] Eaves A., Stockman M.: "Desktop as a service proof of concept". In: Proc. of the 13th annual conference on Information technology education (SIGITE-12), 2012, pp. 85-86

[3] Tordsson J., Montero R. S., Moreno-Vozmediano R., Llorente I. M.: "Cloud brokering mechanisms for optimized placement of virtual machines across multiple providers" In: Future Generation Computer Systems. 2012, pp. 358-367

[4] Jang J. W., Jeon M., Kim H. S., Jo H., Kim J. S., Maeng S.: "Energy Reduction in Consolidated Servers through Memory-Aware Virtual Machine Scheduling" In: IEEE Transactions On Computers, 2011, pp. 552-558

[5] Akbari M., Rashidi H., Alizadeh S. H.: "An enhanced genetic algorithm with new operators for task scheduling in heterogeneous computing systems". Engineering Applications of Artificial Intelligence Journal, 2017, pp. 35-46

[6] Petkovics I., Tumbas P., Matković P., Baracskai Z.: "Cloud computing support to university business processes in external collaboration". Acta Polytechnica Hungarica, 11(3), 2014, pp. 181-200

[7] Hu J., Gu J., Sun G., Zhao T.: "A scheduling strategy on load balancing of virtual machine resources in cloud computing environment" In: Parallel Architectures, Algorithms and Programming (PAAP), 2010 Third International Symposium on, 2010, pp. 89-96

[8] Wang B., Xing H. "The application of cloud computing in education informatization".: In: Proc. of International Conference on Computer Science and Service System (CSSS), 2011, pp. 2673-2676

[9] Sarıtaş M. T.: "The Emergent Technological and Theoretical Paradigms in Education: The Interrelations of Cloud Computing (CC), Connectivism and Internet of Things (IoT)". Acta Polytechnica Hungarica, 12(6), pp. 161-179

[10] All-in-one solution for Remote Access and Web Portal / Terminal Service Plus, 2016, URL: https://www.tsplus.net/

[11] Pearson K.: "On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling". Philosophical Magazine Series, 1900, 157-175

[12] Grubbs F.: “Procedures for Detecting Outlying Observations in Samples”. Technometrics, 1969, 11: 1-21

[13] Ruxton G. D.: “The unequal variance t-test is an underused alternative to Student's t-test and the Mann–Whitney U test”. Behavioral Ecology, 2006, pp. 688-690

[14] Polezhaev P., Legashev L., Shukhman A., Bolodurina I.: “Simulation Modeling of Virtual Machine Based Services in Multi-Cloud Shared Access Centers” In: 7th Scientific Conference on Information Technologies for Intelligent Decision Making Support (ITIDS 2019), 2019, pp. 219-224