Two-stage approach to real-time assignment of Web Studio customer support tasks

S. Begenova[1], T. Avdeenko[1]

[1]Novosibirsk State Technical University, Prospekt K. Marksa 20, Novosibirsk, 630073, Russia

Abstract

Many Web-site owners turn to Web Studios for help in solving the problems of their web projects' support, mostly on a subscriber basis. If the Web Studio is professionally engaged in technical support of web projects, then it can have dozens and hundreds of clients, who create hundreds of job requests per month. However, Web Studio's human resources are usually limited. Therefore, the resource allocation problem is of great current interest (by resources we mean the programmers involved in the implementation of this type of work). In this article, we propose two - stage approach for determining an optimal schedule of Web Studio's customer support tasks. The algorithm of dynamic assignment of tasks in real time is implemented taking into account the dynamic character of this process. The performance of the proposed approach is investigated.

Keywords: dynamic scheduling theory; mathematical model; customer service; real time assignment; Web Studio; optimization

1. Introduction

Wide range of specialists is involved while working on the projects implemented by Web Studios. They are managers, programmers, business analysts, designers and other technical personnel. However, according to the Web Studio managers, the main force on which projects are held, and the main critical resource, are programmers.

In accordance with [1] the main areas, the programmers of Web Studios are working on, are:

• Web - sites development (requirements gathering, development of key pages' prototypes, technical specifications development, development of Web-site design, layout of Web-site design, programming and setup, testing and project implementation);

• Guarantee (subscriber) service.

Many Web - site owners turn to Web Studios for help in solving the problems of their web projects' support, mostly on a subscriber basis. Subscriber service means site's technical support and solution of tasks set by the developer or the client within the paid hours. If the Web Studio is professionally engaged in technical support of web projects, then it can have dozens and hundreds of clients, who create hundreds of job requests per month. In this case the support contract usually fixes the maximum response time to the client's request. Web Studio human resources are usually limited. Therefore, the resource allocation problem is of great current interest (by resources we mean the programmers involved in the implementation of this kind of work).

Resource allocation problem is usually used to solve the problem of assigning resources (machines, programmers) to the tasks that need to be performed [2 – 5]. However, in the case of solving the problem of customer support allocation tasks, the usage of static methods of scheduling theory is not enough. In most real-world situations, unforeseen circumstances continually arise that require schedules revision or modification. Such circumstances can concern either resources or operations. Event related to operations is, for example, changing deadlines, canceling orders, late arriving orders, changes in the production process due to the replacement of resources, etc. Thus, the schedule often becomes irrelevant even before its completion.

Such situation forced the development of the so-called dynamic scheduling theory [6 – 8], a set of approaches that respond to unexpected events, either by adjusting the existing schedule, or by rescheduling the remaining operations. It is more efficient to use the dynamic scheduling theory within the terms of present dynamic environment. The present article considers and analyzes two - stage approach to scheduling Web Studio customer support tasks. At the first stage of the approach we adapt mathematical model for multi-skilled project scheduling [9 – 12], the solution of which allows us to construct the initial static schedule. At the second stage of the approach we propose special algorithm for real-time assignment based on the schedule prepared on the first stage.

The rest of the paper is organized as follows. Section 2 considers the current workload of Web Studio programmers based on the real data presented in the form of Gantt chart. Section 3 sees into two-stage dynamic approach. Subsection 3.1 introduces mathematical model for multi-skilled project scheduling and explains its application with usage of the IBM ILOG CPLEX software. The problem is formulated mathematically as a bi-objective optimization model to minimize total costs of processing the tasks and to minimize reworking risks of the tasks, concurrently. In the subsection 3.2 we propose an algorithm of assigning the tasks in real time. In section 4 we give the performance results of the implemented two-stage approach. Section 5 summarizes accomplished work and gives the conclusion.

2. Analysis of Web Studio data

To solve the problem of distribution of Web Studio customer support tasks, a particular set of real data on tasks, performed on a certain period of time within the subscriber service, was obtained and analyzed. The existing method of this Web Studio tasks allocation is the usage of software that works on the principle of priority and the "manual" assigning the tasks for resources (Web Studio programmers) by the project manager. In this case, the tasks are distributed using the priority principle in accordance with which the tasks are assigned to the following values depending on their urgency:

- normal,
- urgent,
- critical.

The consequence of manual assignment of the tasks is "idle periods" in the schedule of programmers, i.e. periods of time in which the programmer does not have tasks to perform. To analyze the workload of the programmers' current schedule, a Gantt chart was constructed on the data received by the Web Studio and presented in figure 1.

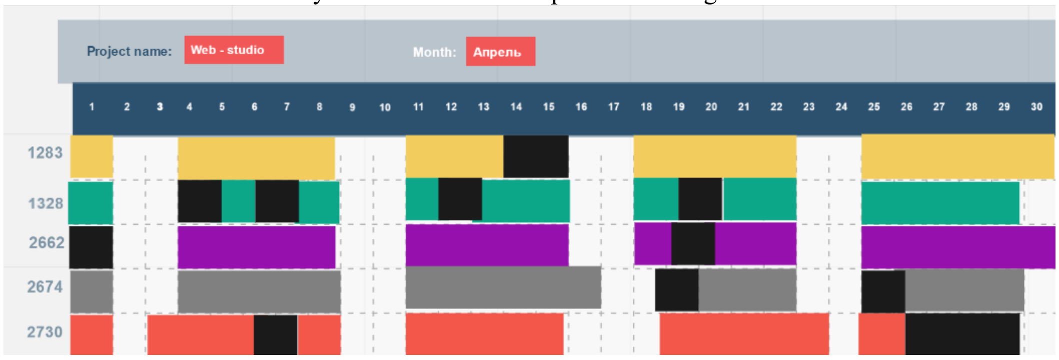


Fig. 1. Gantt chart for April, 2016.

This diagram shows the schedule of Web Studio programmers for April 2016. The black colored segments show "idle periods", which are the days when the programmers did not perform any customer support tasks. Vertically the numbers - identifiers of the programmers are shown, horizontally - the day of the month. The diagram shows that for the programmers with identifiers 1283, 2674 and 2662 idle times were noticed two times (idle time is considered the absence of tasks for the whole day). The programmers 1328, 2730 have such a number of idle times goes till 5. Taking into account that in these days programmers could perform project tasks alongside with the implementation of projects, the load of programmers in terms of subscriber service is far from complete. Therefore, we are going to consider two approaches of the dynamic construction of schedules as a way to improve and optimize the current allocation of customer service tasks.

3. The two-stage approach to real-time assignment

The schedule is built in two stages. Firstly, a long-term (static) schedule is built using multi – skilled project scheduling. Then, with the help of a dynamic approach, new tasks are built in the existing long-term schedule. Thus, new tasks are distributed without changing the current schedule.

3.1. Mathematical model for multi-skilled project scheduling with level-dependent rework risk

Let us use the following optimization problem to get long- term schedule – «Multi-skilled project scheduling with level-dependent rework risk» [13]. A multi-skilled version of the resource constrained project scheduling problem was first proposed by Neron and Baptista in [14]. Such mathematical model supposes that each worker has at least one skill. In this problem setting, a group of workforces with predetermined skills should be assigned to perform all required skills involved in each task. In addition, all the workforces assigned to the skills of each task should start executing the task at the same time and should be available until completing all skills of that task [15, 16].

Let i, j be indexes of the tasks; s be index for the skills; l be index for the levels of the skills; m be index of manpower; t, t' be indexes of time; P_i be processing time of task i.

Let us denote by C_{msl} – the cost of performing the skill s by manpower m at level l per unit time, by b_{is} – the required number of workforces that use skill s on task i, by Pr_{msl} – the risk of reworking if manpower m performs the skill s at level l and by

$$r_{msl} = \begin{cases} 1, \; if \;\; manpower \;\; m \;\; has \;\; skill \, s \;\; at \;\; level \;\; l \\ 0, otherwise \end{cases};$$

Decision variables that have to be determined have following meanings:

$$z_{it} = \begin{cases} 1, \; if \;\; task \;\; i \;\; is \;\; started \;\; at \;\; time \;\; t \\ 0, otherwise \end{cases};$$

$$x_{imt} = \begin{cases} 1, \textit{ if manpower } m \textit{ begins to work on task } i \textit{ at time } t \\ 0, \textit{otherwise} \end{cases};$$

$$y_{imsl} = \begin{cases} 1, \textit{ if manpower } m \textit{ performs level } l \textit{ of skill } s \textit{ on task } i \\ 0, \textit{otherwise} \end{cases};$$

Let us consider the following objectives for our mathematical model:

$$Z_1 = \sum_{m=1}^{M}\sum_{s=1}^{S}\sum_{l=1}^{L}(\Pr_{msl}\cdot\sum_{i=1}^{N}Y_{imsl}) \longrightarrow Min \; ; \tag{1}$$

$$Z_2 = \sum_{m=1}^{M}\sum_{s=1}^{S}\sum_{l=1}^{L}(C_{msl}\cdot\sum_{i=1}^{N}Y_{imsl}) \longrightarrow Min \; ; \tag{2}$$

$$Z_3 = w_1 \cdot Z_1 + w_2 \cdot Z_2 \rightarrow Min; \quad \textit{where } w_1 + w_2 = 1 \tag{3}$$

The first objective (1) minimizes the total cost of processing the tasks involved in a project. The second objective (2) minimizes the reworking risk of the processed tasks. Both objectives can be taken into account by using weighted coefficients as represented by formula 3. Values of weighted coefficients are chosen by decision – makers, who figure out which objective is more or less important than another one. In the research, the next combinations were used: $w_1 = w_2 = 0.5$; $w_1 = 0$ and $w_2 = 1$; $w_1 = 1$ and $w_2 = 0$.

The optimization problem has to be solved under the following constraints:

$$\sum_{t=0}^{T} t\cdot Z_{it} + P_i \le \sum_{t=0}^{T} t'\cdot Z_{jt'} \;, \; \forall (i,j)\in E; \tag{4}$$

$$\sum_{t=0}^{T} Z_{it} \le 1 \;, \forall i; \tag{5}$$

$$X_{imt} \le Z_{it} \;, \forall i,\, m,\, t; \tag{6}$$

$$X_{imt} + 1 \ge Z_{it} + \sum_{s=1}^{S}\sum_{l=1}^{L} Y_{imsl} \;, \forall i,\, m,\, t; \tag{7}$$

$$\sum_{m=1}^{M}\sum_{l=1}^{L} Y_{imsl} = b_{is} \;, \forall i,\, s; \tag{8}$$

$$Y_{imsl} \le r_{msl} \;, \forall\; i,\, m,\, s,\, l; \tag{9}$$

$$\sum_{m=1}^{M}\sum_{t=0}^{T} X_{imt} = \sum_{s=1}^{S} b_{is} \;, \forall i; \tag{10}$$

$$\sum_{i=1}^{N}\sum_{t'=t-p_i+1}^{t} X_{imt'} \le 1 \;, \forall m,t; \tag{11}$$

$$\sum_{t=0}^{T} X_{imt} \le \sum_{s=1}^{S}\sum_{l=1}^{L} Y_{imsl} \;, \forall i,\, m; \tag{12}$$

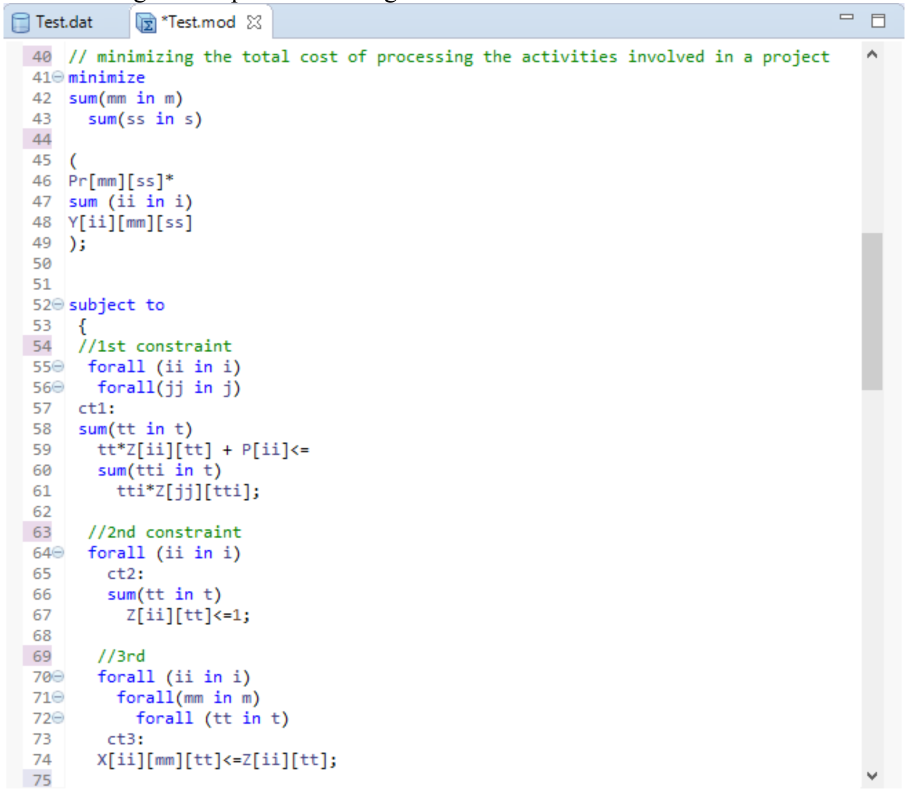
$$\sum_{l=1}^{L} Y_{imsl} \le 1 \;, \forall\; i,\, m,\, s; \tag{13}$$

$$Z_{it}, X_{imt}, Y_{imsl} = \{0,1\} \;, \forall\; i,\, m,\, s,\, l; \tag{14}$$

The constraint (4) preserves the precedence relations between the tasks. The constraint (5) ensures that each task must be started once. The constraints (6) and (7) imply that all the workforces assigned to various skills of each task should start their work, concurrently. The constraint (8) implies that the number of workforces assigned to different levels of each skill should be equal to the number of manpower required to perform that skill. The constraint (9) ensures that the manpower assigned to a level of each skill should be able to perform it appropriately. The constraint (10) implies that the number of workforces assigned to each task should be equal to the number of required manpower to accomplish that task. Constraint (11) ensures uninterrupted assignment of workforces to different skills of project tasks. In other word, the manpower assigned to a skill of each task must perform the assigned skill continuously without interruption. The constraint (12) implies that each task should be executed by the workforces assigned to different skills of that task. The constraint (13) ensures that the manpower assigned to each skill of an task should perform that skill in her/his predefined level. Finally, the constraint (14) denotes that all decision variables are binary.

Mathematical model for multi-skilled project scheduling with level-dependent rework risk was solved using an optimization software package IBM ILOG CPLEX Optimization Studio. This software is an enterprise analytical decision support toolkit. It enables rapid development and deployment of decision optimization models using mathematical and constraint programming. ILOG CPLEX combines completely featured integrated development environment (IDE) that supports Optimization

Programming Language (OPL) application development to create high-performance ILOG CPLEX optimizer solvers. ILOG CPLEX enables you to optimize your business decisions, develop and deploy optimization models quickly and create real-world applications. The CPLEX code fragment is presented in figure 2.



```
// minimizing the total cost of processing the activities involved in a project
minimize
sum(mm in m)
  sum(ss in s)

(
Pr[mm][ss]*
sum (ii in i)
Y[ii][mm][ss]
);


subject to
 {
 //1st constraint
  forall (ii in i)
   forall(jj in j)
 ct1:
 sum(tt in t)
   tt*Z[ii][tt] + P[ii]<=
   sum(tti in t)
     tti*Z[jj][tti];

  //2nd constraint
  forall (ii in i)
    ct2:
    sum(tt in t)
      Z[ii][tt]<=1;

   //3rd
   forall (ii in i)
     forall(mm in m)
       forall (tt in t)
    ct3:
   X[ii][mm][tt]<=Z[ii][tt];
```

Fig. 2. CPLEX code fragment.

3.2. Real-time assignment

The second stage of the construction of schedule for Web Studio customer support tasks is the integration of incoming new tasks into the schedule that was built at the first stage. To implement such an integration we apply the Real-time assignment algorithm.

The real-time assignment algorithm was proposed due to the spread of the supply chain paradigm. In the supply chain, each project encompasses a whole production cycle, starting from customer requirements to final calculations. In addition, the variety of products involved in the project is limited in terms of the number of tasks types. Today, a project is understood as a continuous stream of tasks. In particular, production systems are gradually moving from small-scale production to conveyor assembly lines, which guarantee not only the performance, but also the flexibility of the system. Thus, the assignment of job (set of tasks) in real time is to assign a job to a set of resources upon the arrival of the order in the production system. In the event of a violation of the performance of any task, previously planned unfinished task is redistributed in the order corresponding to the demand. The goal of this approach is to redistribute tasks in real time.

Consider the following problem of assigning jobs in real time with fixed previous assignments. Let us suppose that task i can be performed by any of the programmers $\{m_i^1, m_i^2, m_i^3, ..., m_i^{K_i}\}$, and the programmer $m_i^k, km\{1,2,\ldots,K_i\}$ is idle on the periods $I_i^k = \left\{\left[\alpha_{i,q}^k, \beta_{i,q}^k\right]\right\}_{q=1,2,\ldots,Q_{k,i}}$. Thus, K_i is the maximum number of programmers who are able to perform the task i, and $Q_{k,i}$ is the maximum number of idle periods available for task i of the programmer m_i^k.

The initial data for the algorithm are the programmers' initial schedules determined at the first stage, the idle periods of the programmers, and the incoming tasks with their time estimates. For the case of several programmers whose performances are identical, we group the idle periods associated with one group of such programmers as follows.

For $k_1 \neq k_2$, where $k_1, k_2 \in \{1,2,\ldots,K_i\}$, period $\left[\alpha_{i,q}^{k_1}, \beta_{i,q}^{k_1}\right]$,

$q \in \{1,2,\ldots,Q_{k_1,i}\}$ precedes $\left[\alpha_{i,r}^{k_2}, \beta_{i,r}^{k_2}\right], r \in \{q=1,2,\ldots,Q_{k_2,i}\}$ if:

1. $\alpha_{i,q}^{k_1} < \alpha_{i,r}^{k_2}$, или
2. $\alpha_{i,q}^{k_1} = \alpha_{i,r}^{k_2}$ и $\beta_{i,q}^{k_1} < \beta_{i,r}^{k_2}$.

The sequence of such sorted periods is denoted by $\left[\alpha_i^s, \beta_i^s\right]$, where $s = 1,2,\ldots,\sum_{k=1}^{K_i} Q_{k_i,i} = Q_i$

Let s_i be a rank of the idle period assigned to the i -th operation in the operation sequence; t_i be a starting time of the operation i; θ_i be processing time required to complete the operation i. Let us denote by δ_i - maximum overstay permitted for the operation i , on the corresponding resource; by $\alpha_i^{s_i}$ - starting instant of the s_i -th idle period that could be assigned to the operation i and by $\beta_i^{s_i}$ - finishing instant of the s_i -th idle period.

In this approach, the following Real-time assignment algorithm is used:

1. Set $s_i = 1$ for i =1,2,…,m;
2. Set $p_1 = \alpha_1^{s_1}$;
3. Set $p_i = MAX(\alpha_i^{s_i}, p_{i-1} + \theta_{i-1})$, i=2,…,m;
4. Set $p_{m+1} = p_m + \theta_m$;
5. Set $t_{m+1} = p_{m+1}$;
6. Set $t_i = MAX(p_i, t_{i+1} - \theta_i - \delta_i)$ for all i= m,m-1,…1;
7. If $(t_{i+1} > \beta_i^{s_i})$ for all i=1,2,…,m, then the optimum is reached;

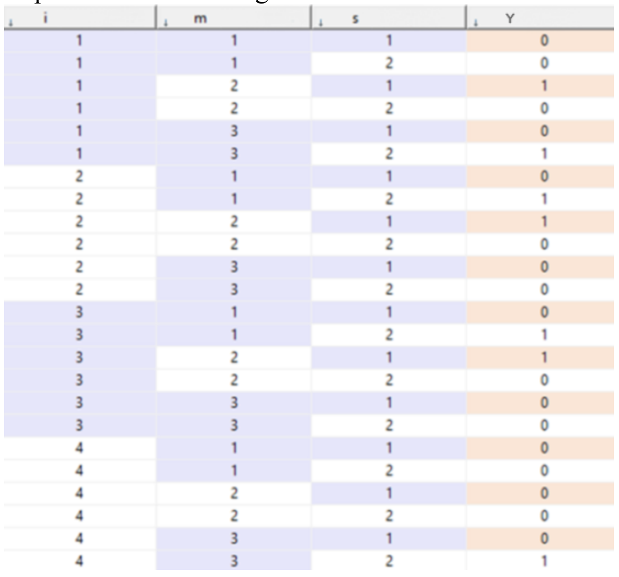
   else, for each i such that $(t_{i+1} > \beta_i^{s_i})$, set $s_i = s_i + 1$ and go to step 2.

4. Performance results

Here are the results of solving the optimization problem «Multi-skilled project scheduling with level-dependent rework risk» for the test case with 3 workers (programmers), 2 skills and 4 tasks, see figure 3, figure 4. The optimal value of the decision variable Y is presented in figure 3 (last column). The variable Y is equal to 1, if manpower m performs the skill s on the task i, otherwise 0.

For example, in the third row manpower m_2 does perform the skill 1 on the operation 1, but he does not perform the skill 2 on operation 1 (the fourth row). Dimension of the vector Y is $M \cdot S \cdot N$, where M is the number of manpower, S is the number of skills and N is amount of tasks. Dimension of the resulting vector Y presenting in figure 3 is equal to 3*2*4 = 24.

IBM ILOG CPLEX found the most optimal solution taking into account all the above-mentioned constraints.



i	m	s	Y
1	1	1	0
1	1	2	0
1	2	1	1
1	2	2	0
1	3	1	0
1	3	2	1
2	1	1	0
2	1	2	1
2	2	1	1
2	2	2	0
2	3	1	0
2	3	2	0
3	1	1	0
3	1	2	1
3	2	1	1
3	2	2	0
3	3	1	0
3	3	2	0
4	1	1	0
4	1	2	0
4	2	1	0
4	2	2	0
4	3	1	0
4	3	2	1

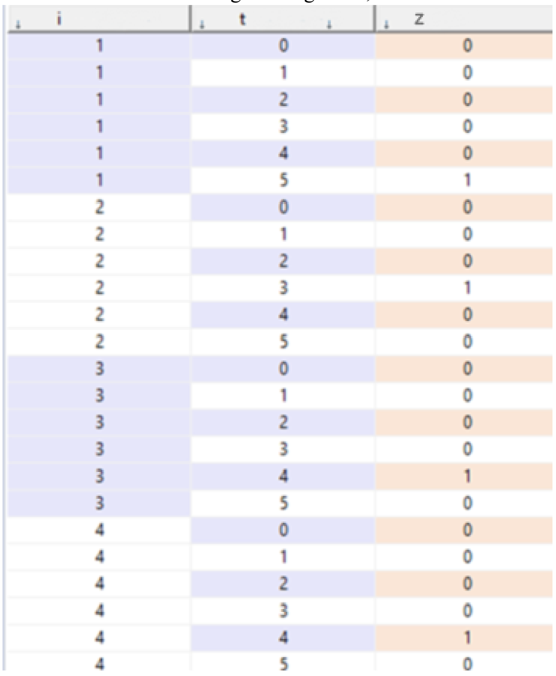
Fig. 3. The resulting vector Y.

The optimal value of decision variable Z is presented in figure 4. The variable Z is equal to 1, if the task i starts at the time t, and 0 otherwise. For example, operation 1 starts at time 5 (the sixth row). Dimension of the vector Z is $T \cdot N$, where T is upper bound of index t and N is number of tasks. Dimension of the resulting variable Z presenting in picture 4 is 6*4 = 24.

And combination of resulting variables gives us the long – term shedule of Web Studio customer support tasks for programmers.

Obtained resulting variables gives us following long – term schedule: task 1 starts at time 5 executed by manpower 2 perfoming skill 1; task 2 starts at time 3 executed by manpower 2 perfoming skill 2; task 3 starts at time 4 executed by manpower 1 perfoming skill 2 and manpower 2 perfoming skill 1; and task 4 starts at time 5 executed by manpower 3 using skill 2.

Let us study the dependence of the optimal solution finding time on the number of tasks, workers or skills. That would helps us to understand whether this mathematical problem is applicable in real conditions or not. The table 1 shows the dependence of computation time of optimal solution on the number of tasks performed. It can be seen from the table that with the increase in the number of tasks, the execution time also increases substantially. The study was done for the test case with 10 workers and 2 skills.



i	t	z
1	0	0
1	1	0
1	2	0
1	3	0
1	4	0
1	5	1
2	0	0
2	1	0
2	2	0
2	3	1
2	4	0
2	5	0
3	0	0
3	1	0
3	2	0
3	3	0
3	4	1
3	5	0
4	0	0
4	1	0
4	2	0
4	3	0
4	4	1
4	5	0

Fig. 4. The resulting vector Z.

Table 1. The dependence of the optimal solution finding time on the number of tasks.

The number of tasks	Computation time
15 tasks	4 sec
50 tasks	7 min 1 sec
100 tasks	43 min 34 sec

The table 2 revealed that the dependence of computation time of optimal solution on the number of workers performed is not as high as in the previous case. So from this side, program's execution time is feasible in the context of Web Studio work. The study was done for the test case with 50 tasks and 2 skills.

Table 2. The dependence of the optimal solution finding time on the number of workers.

The number of workers	Computation time
15 workers	42 sec
30 workers	2 min 37 sec
100 workers	6 min 23 sec

The table 3 also revealed the dependence of computation time of optimal solution on the number of skills. Obviously, the same as in the table 2, this dependence goes up not so quickly, which is positive in and on itself. The study was done for the test case with 50 tasks and 10 workers.

Table 3. The dependence of the optimal solution finding time on the number of skills.

The number of skills	Computation time
2 skills	20 sec
5 skills	2 min 32 sec
10 skills	10 min 37 sec

In all three cases obtained results are feasible in real - life terms, since long-term schedule is calculated once for a long period of time and then it gets corrected with Real-time assignment algorithm.

Now consider the tests of the software implementation of the Real-time assignment algorithm in multi-paradigm numerical computing environment MATLAB.

Test case with 2 programmers

Let us consider a test case where the tasks will be executed by two programmers m_1 and m_2. The initial data necessary for the execution of the algorithm is presented below.

Select the next execution time required to complete the tasks:

- The first task i_1 will be executed during 1 time unit ($\theta_1 = 1$)
- The second task i_2 will be executed during 2 time units ($\theta_2 = 2$)

The maximum waiting time that a programmer can have is 1 time unit. Let us define the periods of idle time for programmers, that is, the periods in which they can take a customer service task on a performance.

The idle periods I_1 of the first programmer m_1 are $[0, 3]$, $[5, 10]$, $[15, +\infty)$.

The idle periods I_2 of the second programmer m_2 are $[0, 6]$, $[16, 19]$, $[25, +\infty)$.

s is the result variable which represents the idle period number on which the specific task was assigned.

The results of algorithm execution are shown in Figure 5. In Figure 5 black rectangles represent the periods of business of the programmers, and green rectangles – the periods of time in which tasks will be performed.

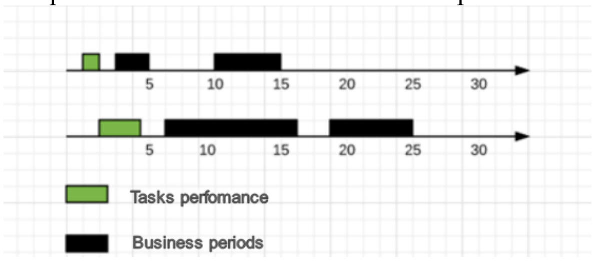


Fig. 5. The solution reached using the algorithm.

In test №1, as we can see two tasks i_1 and i_2 will be executed consistently in the first periods of idle periods of both programmers m_1 and m_2, respectively. All the idle windows are suitable for performing the related tasks. Furthermore, t_i is the start time and t_{t+1} the completion time of the i-th operation. For each resource, the upper limit of the last window is always $+\infty$. As a consequence, applying the above – mentioned algorithm always leads to a solution.

Test case with 4 programmers

In this test case, the tasks will be distributed among 4 programmers.

The execution time required to complete the task:

- for the first task i_1- θ_1= 3
- for the second task i_2- θ_2=4
- for the third task i_3- θ_3=5
- for the fourth task i_4- θ_4=5

The maximum waiting time for a resource is 1.

The idle periods I_1 of the first programmer m_1 are [0,2], [5, 15], [25, + ∞).

The idle periods I_2 of the second programmer m_2 are [0,15], [25, + ∞).

The idle periods I_3 of the third programmer m_3 are [0,5], [10,25], [30, + ∞).

The idle periods I_4 of the fourth programmer m_4 are [0.10], [20, + ∞).

The results of the algorithm execution are shown in figure 6.

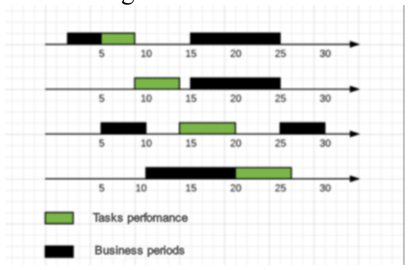


Fig. 6. The solution reached using the algorithm.

For this test, tasks i_1, i_2, i_3 and i_4 will be executed successively in the second idle periods of all programmers except the second one.

Test case with 8 programmers

Consider the results of this algorithm in the distribution of tasks for 8 programmers.

The execution time required to complete the task:

- for the first task i_1- θ_1=3
- for the second task i_2 - θ_2= 2
- for the third task i_3 – θ_3=3
- for the fourth task i_4- θ_4=2
- for the fifth task i_5 – θ_5=3
- for the sixth task i_6 – θ_6=5
- for the seventh task i_7 – θ_7=4
- for the eighth task i_8 – θ_8=5

The maximum waiting time for a resource is 1.

The idle periods I_1 of the first programmer m_1 are [0,2], [5, 15], [25,30], [35, + ∞).

The idle periods I_2 of the second programmer m_2 are [10,15], [25,27], [30, + ∞).

The idle periods I_3 of the third programmer m_3 are [0,7], [13, 17], [20,24], [29, + ∞).

The idle periods I_4 for the fourth programmer m_4 are [0.5], [10, 12], [16,22], [31, + ∞).

The idle periods I_5 for the fifth programmer m_5 are [10,15], [20,27], [30, + ∞).

The idle periods I_6 for the sixth programmer m_6 are [7,11], [15,20], [25,30], [35, + ∞).

The idle periods I_7 for the seventh programmer m_7 are [6,10], [14,24], [29,35], [43, + ∞).

The idle periods I_8 for the eighth programmer m_8 are [0,5], [10,12], [16, 18], [20,22], [31, + ∞).

The results of the algorithm execution are shown at figure 7.

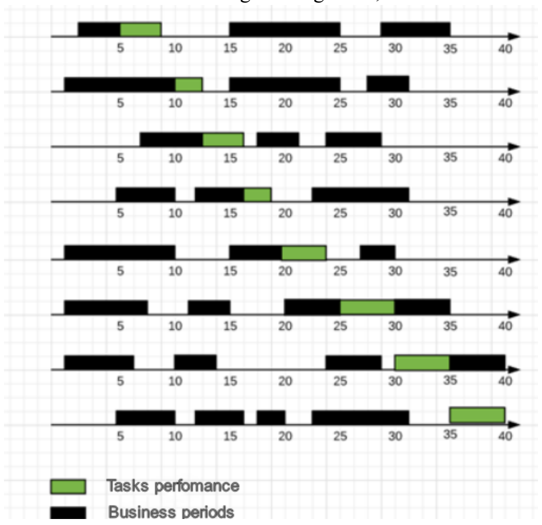


Fig. 7. The solution reached using the algorithm.

In the test cases (figures 5, 6, 7), the tasks were distributed uniformly, taking into account the idle periods of the programmers and their maximum waiting time. The above-given test cases show that the algorithm is looking for the most suitable idle period for the task. If the task does not fit in the idle period, or the maximum waiting time for the resource is exceeded, the algorithm starts looking for another nearest suitable idle period.

Test of the Real-time assignment algorithm revealed the main advantages of this approach. The need to build a new schedule in the event of any breakdown or any other unforeseen circumstance that make the current schedule irrelevant now disappears. The feature of this approach is the fixed previous assignments. Thus, this approach allows you to schedule the execution of incoming tasks without changing the current schedule of programmers. Also, the advantage of this method is the low labor input and high speed, which, undoubtedly, is important for the practical application of the method.

Another advantage of the proposed algorithm is the ability of filling and eliminating idle periods in the programmer schedule. This property allows us to apply this algorithm even when the original schedule is not optimal, as it turned out in the real conditions discussed in Section 2.

5. Conclusion

The data analysis of the workload of employees by Web Studio customer support tasks for a certain period showed that the current principle of tasks allocation gives non-optimal workload of the employees. Thus, the conducted analysis revealed the urgency of research and development of methods for more optimal allocation of the customer support tasks. As a result of present research we have proposed the two-stage approach based on dynamic scheduling theory.

The undoubted advantage of this approach is the ability to schedule the work of programmers without changing their current order of execution of the tasks. The Real-time assignment algorithm allows optimally filling the programmers' idle periods and evenly distributing the programmers' workload and reducing the idle periods.

At this stage of the research, the Real-time assignment algorithm considers the process of performing the tasks as a sequential procedure, which does not always suitable for the current principle of executing tasks in Web Studios. Therefore, promising direction for further work is more accurate adaptation of the algorithm to the problem of distribution of the customer support tasks. That is, to take into account the possibility of parallel execution of tasks by programmers [17- 20], the priority of tasks, as well as the priority of the programmer for the task. The latter means the priority of a certain programmer in distribution of the subscriber tasks: the programmer who implemented the web project has a higher priority in the queue for subscriber tasks from this project.

While analyzing the results of the solution of the optimization problem, a significant increase in the time of finding the optimal solution was revealed with an increase in the number of tasks. Therefore, we are going to try reducing this time by integrating the optimization model with different heuristics. This kind of reducing is really important because real – life problems' dimensions are huge. In our case the problem is solved once, and then dynamic approach, which is independent of problems' dimensions, makes its corrections.

Acknowledgements

The reported study was funded by Russian Ministry of Education and Science, according to the research project No. 2.2327.2017/4.6.

References

[1] Avdeenko TV, Petrov RV. On the possibility of applying methods and models of scheduling theory to optimization of working a web-studio. Sbornik nauchnyh trudov Novosibirsk State Technical University 2016; 2(84): 7–20. (in Russian)
[2] Brucker P. Scheduling Algorithms. Springer, 2007; 372 p.

[3] Pinedo M. Scheduling Theory, Algorithms, and Systems. Springer, 2008; 672 p.
[4] Lazarev AA, Gafarov ER. Scheduling theory. Problems and algorithms. M.: MSU, 2011; 222 p. (in Russian)
[5] Pavlov OA, Chernov SK, Misyura OB. Models and algorithms of scheduling theory in planning and project management problems. Trudy Odessa Polytechnic University 2006; 1(25): 150–159 . (in Russian)
[6] Dolgui A, Proth J-M. Supply Chain Engineering - Useful Methods and Techniques. Springer –Verlag, 2010; 541 p.
[7] Toroslu I. Personnel assignment problem with hierarchical ordering constraints. Proc. Personnel assignment Problem with hierarchical ordering constraints 2003; 493–510.
[8] Alcaraz J, Maroto C, Ruiz R. Solving the multi-mode resource-constrained project scheduling problem with genetic algorithms. Journal of the Operational Research Society 2003; 54(6): 614–626.
[9] Mika M, Waligora G, Weglarz J. Tabu search for multi-mode resource-constrained project scheduling with schedule-dependent setup times. European Journal of Operational Research 2008; 187(3): 1238–1250.
[10] Jarboui B, Damak N, Siarry P, Rebai A. A combinatorial particle swarm optimization for solving multi-mode resource-constrained project scheduling problems. Applied Mathematics and Computation 2008; 195(1): 299–308.
[11] Abbasi B, Shadrokh S, Arkat J. Bi-objective resource-constrained project scheduling with robustness and makespan criteria. Applied Mathematics and Computation 2006; 180(1): 146–152.
[12] Bouleimen K, Lecocq H. A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode version. European Journal of Operational Research 2003; 149(2): 268–281.
[13] Maghsoudlou H, Afshar-Nadjafi B, Akhavan Niaki ST. Multi-skilled project scheduling with level-dependent rework risk; three multi-objective mechanisms based on cuckoo search. Applied Soft Computing 2017; 54: 46–61.
[14] Zhu G, Bard JF, Yu G. Disruption management for resource-constrained project scheduling. Journal of the Operational Research Society 2005; 56: 365–381.
[15] Al-Fawzan M, Haouari M. A bi-objective model for robust resource-constrained project scheduling. International Journal of Production Economics 2005; 96(2): 175–187.
[16] Néron E, Baptista D. Heuristics for multi-skill project scheduling problem. Int.Symp. Comb. Optim.2002;
[17] Avdeenko TV, Mesentsev YA. Efficient approaches to scheduling for unrelated parallel machines with release dates. IFAC-PapersOnline 2016; 49(12): 1743–1748.
[18] Avdeenko TV, Vasiljev MA. Multiagent approach with use of fuzzy modeling in the task multicriterion decision making. Nauchny vestnik Novosibirsk State Technical University 2010; 1: 63–74. (in Russian)
[19] Avdeenko TV. Parameter identification problems in Mathematical modelling of processes. Obrazovatel'nye resursy i tehnologii 2014; 1(4): 115–124. (in Russian)
[20] Mezentsev YuA, Avdeenko TV. Scheduling and Optimization for Parallel-Serial Service Systems. Proceedings of 8th International Forum On Strategic Technology 2013; 271–275.