Turkish Journal of Computer and Mathematics Education *Vol.12 No.6 (2021), 1638-1644*
Research Article

Application of Reinforcement Learning to Optimize Business Processes in the Bank

Andrey A. Bugaenko
Sberbank PJSC (ORCID 0000-0002-3372-5652)

Article History: Received: 10 November 2020; Revised 12 January 2021 Accepted: 27 January 2021; Published online: 5 April 2021

Abstract: This article describes the application of reinforcement learning (q-learning, genetic algorithm, cross-entropy) to define the optimal structure of business processes in the bank. It describes the principle of creation of the environment, loss, and reward. Setting of hyperparameters for each method is considered in depth. Besides, it offers the variant of calculation of the maximum potential for saving, which can be arrived at through the business process optimization.

Keywords: Reinforcement Learning, Q-Learning, Genetic Algorithm, Cross-Entropy, Environment Creation, Business Process, Process Mining, Bank

1. Introduction

A search for opportunities for business processes optimization is a crucial task for the bank's analytical divisions. Currently, in the majority of cases, the rules for implementing a particular process are assigned subjectively based on the analytical divisions and management's general idea. This approach is unsafe: on the one hand, the management's vision of the process may differ from the objective reality; on the other hand, the manager may make a mistake in calculations when building an optimal sequence of actions in a process. The first reason is related with the fact that the process vision is built based on the miscellaneous data in reliance on some subjective observations of the manager himself, on the information according to employees, on the conclusions made by consultants, and does contain objective system-related facts. Indeed, the process is built with the use of financial and statistical information, but such information also contains the process vision as seen from certain aspects, rather than comprehensively. The second reason is related with the fact that building of the rules does not take into account the quantitative and probability indices of the process, and the rules are predominantly built based on the manager's vision which is "how to make it correctly", rather than "how to make it correctly in order to meet our needs in the situation at hand." All this normally leads to financial losses in business and income deficiency. In many cases, the management can understand the above stated problems in terms of quality, rather than be aware of the level of losses in terms of quantity, and respectively can lack motivation for changes. In order to level down the above problems, it's very important for the bank's management to consider the following issues: first, to understand the real structure of a business process (to have available a network graph of a business process, which is built based on objective data); second, to know the optimal sequence of actions of the process (to have available a quantitative calculation of the optimal path in the network graph); third, to see the total potential for optimization in order to learn how much we can receive in addition (save) if the process is optimal (to have available the calculation of the quantitative evaluation of the difference between the optimal path in the graph and the actual one).

2. Application of Reinforcement Learning to Optimize Business Processes in the Bank

The solution to the first problem can be arrived at through the application of the process mining methodology, when the objective drawing of the network graph of the business process is made using the logs from data bases of the initial bank systems (Van der Aalst, 2011; Van der Aalst, 2017; Van der Aalst, 2016; Born *et al*., 2009). The process drawing needs the data-set containing at least the index number of the process instance, id of the person performing the operation, and id of the process node. If the data do not contain the index number of the process instance (like in our case), and the data are represented as a single log of the actions of each manager, an additional marker action should be implemented to distinguish one process instance from another one. In our case, we implemented the action "manager's log in the process sub-system" as the start of the process instance, and the action "manager's log out the process sub-system" as the end of the process. The probability of the node in the process was calculated as follows:

$$Pnode_i = \frac{1}{m}\sum_{k=1}^{m} n_{\text{ki}}, \quad (1)$$

where $Pnode_i$ – probability of the occurrence of the *i*-th node of the process, $i \epsilon [1,h]$, where *h* - maximum possible number of nodes in the process, m - number of instances of the process in the sampled set, k ϵ[1,m] – number of repetitions of the *i*-the node of the process in the *k*-th instance of the process.

The probability of the node in the process was calculated as follows:

$$Pedge_{ij} = \frac{1}{m}\sum_{k=1}^{m} n_{\text{kij}}, \quad (2)$$

where $Pedge_{ij}$ – probability of the transition from the *i*-th node of the process to the *j*-th node, $i,j \epsilon [1,h]$, where *h* - maximum possible number of nodes in the process, m - number of instances of the process in the sampled set, $k \epsilon [1,m]$, n_{kij} – number of the repetitions of the transition from the *i*-th node of the process to the *j*-th one in the *k*-th instance of the process.

The probabilities are necessary not only for the vivid drawing of the process (the higher the probability, the lighter/thicker the lines in the graph), but also for the subsequent calculations.

In order to solve the second problem, it is necessary to define the optimal path in the network graph of the process. There exist the methods of classical search for a path in the network graph, which, in certain cases, give good results, but they are not always the most optimal ones (NetworkX).

Another way to find an optimal sequence of actions in the business process may be obtaining of a recommendation from the system specially taught for this task. The application of reinforced learning is a rather popular area in the recommender systems. Reinforcement learning is one of the ways of machine learning, during which the system under test (agent) is taught, influencing a particular environment and receiving answers to actions from it. The examples of the application of reinforcement learning are bots in chess and go, algorithms of self-driving cars or self-flying drones. In this case, the system receives a response to the bot's actions and learns from them (obtaining certain loss and reward).

However, our task may also be presented in terms of reinforcement learning. In our case, the network graph of the process may be taken *as the environment*, in which *the agent*-system takes a decision about an *action* based on the results of the configured *loss/reward*.

In our case, action is a transition between the process nodes, in other words, the selection of the node to transit to from the current one. The *stage* is the process node, where the manager is at the moment. If we could make actions and influence in some way the process node itself, we would also be able to present the process node as a set of sub-nodes and transitions, and so on ad infinitum. In any case, we would arrive at some minimum node (sub-node) of the process, which we would not be able to present as a set of sub-nodes and transitions, and nevertheless it would turn out to be the stage.

Actually, the task comes down to maximization of *loss/reward* when selecting the variety of transitions between the process nodes.

Eleven bank business processes were used to test the methods of searching for an optimal path: individuals' payments at additional offices, lodging cash into the current account, cash withdrawal from the account, opening a deposit, issuing a consumer credit, issuing the bank card, transfers between individuals, currency exchange by individuals, lease of safe deposit boxes, purchase of collectors' coins, expert examination of token money.

Kudenko and Grzes (2009) describe the principles of assigning the environment using RL, but such an approach lacks the requirement to the interpretation of *Loss/Reward*. When the principle of assigning *loss/reward* was chosen, the requirement which is very important for business was taken into account – interpretability, i.e. all *loss/reward* should correspond to certain financial or marketing indices. In this context, it was proposed to take the bank's general allocated operating expenses per each node of the process as loss. They may be calculated as follows:

$$Loss_i = t_i \times Pr \times Al \quad (3)$$

where t_i – time of the completion of the *i*-th node of the process (sec), $i \epsilon [1,h]$, where *h* - maximum possible number of nodes in the process, *Pr* – cost of a second of time of the employee (manager) (RUB), *Al* – coefficient of allocation defined as follows:

$$Al = \frac{OPEX}{HR\_OPEX} \quad (4)$$

where *OPEX* – operating expenses in business area, *HR_OREX* – operating expenses on employees' salary.

The net commission income on the process, or net present value on net interest income, or sum of these indices may be taken as *Reward*. *Reward* is placed on the point of the network graph (on the stage of the process node), where the income is actually recognized.

For the processes, in the framework of which the methods were tested, the *losses* were assigned according to the general principle described above. *Reward* was placed for: individuals' (Is') payments at additional offices, transfers between Is, currency exchange by Is, lease of safe deposit boxes by Is, purchase of collectors' coins (the processes related with the commission income) – in the stage of the process node "Individual has made the payment", i.e. in the point on the network graph, when the client actually makes payment, and is equal to the net commission income from the transaction; for the processes related with interest income: lodging cash into the current account, opening a deposit, issuing a consumer credit, issuing the bank card – reward was placed on the stage of the process node "The agreement has been approved" and was equal to the net present value (NPV) of the product; for the processes: expert examination of token money and cash withdrawal from the account – a synthetic *reward* was assigned (since the bank does not make profit from these transactions), which is equal to the total *loss*, weighted based on the probabilities, all over the network graph. The data on the processes were taken from the initial systems of Sberbank PJSC. The processes contained from 31 to 119 *nodes* and from 175 to 2583 *edges*.

In order to search for an optimal path in the network graph, three methods of reinforcement learning (RL) were compared: q-learning, cross-entropy, genetic algorithm, as well as classical method of searching for a path in the network graph from the library Networks (NetworkX). The selection of the method is considered by Lin and Pai (2000), Wang et al. (2013), and Huang et al. (2011); however, the proposed selection method either does not use RL, or the task differs from our one materially. In order to aggregate the results on all the processes, the resulting *LossReward* on each of them were normalized as follows:

$$\text{LossRewardN}_k = \begin{cases} -\frac{LossReward_k}{Min(LossReward)} - with\ LossReward < 0 \\ \frac{LossReward_k}{Max(LossReward)} - with\ LossReward\ > 0 \end{cases} \quad (5)$$

Subsequently, the normalized results were averaged for all the processes, and the methods were compared. Testing was performed on the server CPU: 2.1GHz/16-core, 32-core (without GPU), Core memory: 512GB.

The classical method of searching for a path in the network graph was used as a baseline (NetworkX). The algorithm browses all possible paths from the assigned "start" to the assigned "end" randomly, summing all *loss/reward* along the path. If *loss/reward* along the path is larger than along the best path, the path is saved, and the best loss/reward is updated. The algorithm provided a solution for all groups of the processes, i.e. it did not run into a cyclic path. The value of the maximum loss/reward improved logarithmically with the increase in the maximum number of cycles. The averaged normalized diagram of dependence of the maximum value of loss/reward on the number of cycles is as follows:

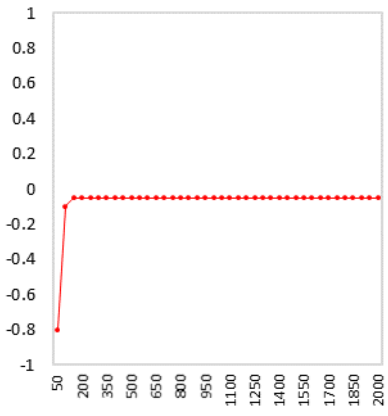
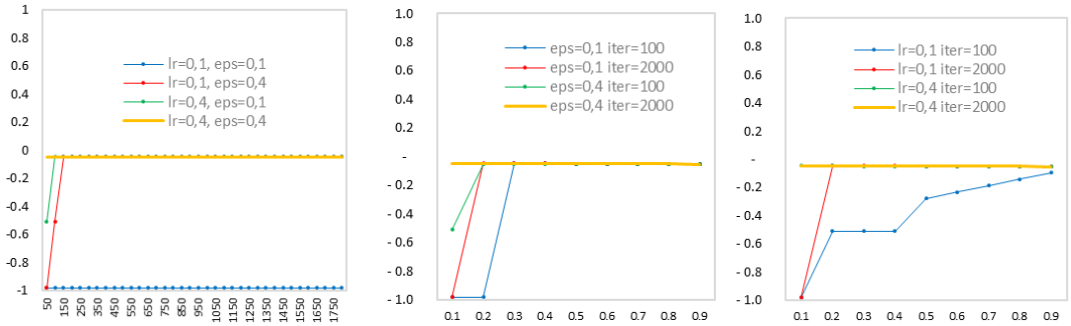


Figure 1. Diagram of dependence of the maximum value of loss/reward on the number of cycles

The first method of reinforcement learning, which was tested for all groups of processes is Q-learning. In particular, the modified algorithm Q-learning was used, which supposes that at the node of the selection of the action *decision*, the agent chooses the next step randomly with ε probability, and it chooses the action proceeding from $a=argmax(Q,s)$ with the probability of (1- ε), where *a – action*, Q - q-function, s - current stage (Leemans & Fahland, 2020). The use of q-learning for the process mining is described by Silvander (2019) and Arango et al. (2017); however, the environment assigning which is proposed in the works does not meet the requirement to interpretability and cannot be applied in our task. It should be noted that in the variant of the environment proposed by Silvander (2019), q-learning is very sensitive to hyperparameters setting.

Greed search was implemented in testing for all combinations of the parameters: ε from 0 to 1 with the step of 0.1, *learning rate* from 0 to 1 with the step of 0.01, numbers of cycles from 0 to 2000 with the step of 50 (when the number of learning cycles was set to a larger value, no result was obtained for an adequate period of time). Below are the diagrams of dependence of *LossReward* on *ε, learning rate*, number of cycles.



Figures 2, 3, 4. Diagram of dependence of the average normalized LossReward on learning rate (2), ε (3), and number of learning cycles (4)

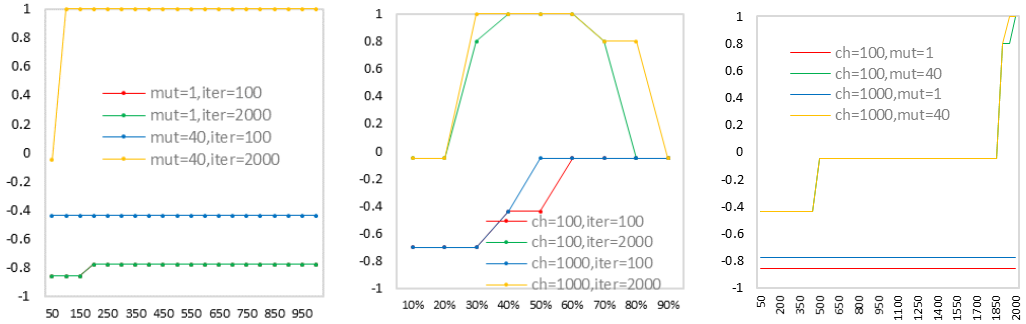
When *learning rate* grew, the algorithm, firstly, improved *LossReward* immaterially, but then, with the growing *learning rate*, almost no impact on *LossReward* was seen. At the same time, the larger the number of learning cycles, the weaker the influence of learning rate on the result.

The change in ε rendered weak impact on *LossReward*, except the situation when the values of *learning rate* and number of cycles were low: the growth of ε improved the result immaterially; in the rest of the cases it rendered almost no impact on the result.

When the number of learning cycles grew, *LossReward* improved up to a certain value, but then remained on one level. Notably, the larger the values of *learning rate* and *ε*, the stronger the influence of the value of the number of cycles. Besides, it should be noted that the speed of operation of one cycle with high values of *learning rate* was lower materially.

As a result, positive *LossReward* was not obtained with any values of *learning rate* and *ε* and their combinations. The algorithm was completing the learning without finding the *node* with *Reward*, it was building an optimal path on *Loss* only. In the majority of the processes, the best path was presented by transactions for the manager: [enter the system] => [exit the system].

Next, the genetic algorithm was tested for our task (Leemans & Fahland, 2020). The application of genetic algorithms for process mining is described by van der Aalst et al. (2005) and de Medeiros et al. (2005); however, it is used to filter and supplement data, rather than to search for a path in a graph; the application described by Li et al. (2018) does not fit our task because of the principle of assigning the environment. Greed search was implemented in testing for all combinations of the parameters: number of children from 0 to 1000 with the step of 50, number of mutations from 0% to 100% with the step of 10 p.p., and number of cycles from 0 to 2000 with the step of 50.



Figures 5, 6, 7. Diagram of dependence of the average normalized LossReward on the number of children (5), share of mutations (6), and number of cycles (7)

In our tasks, the change in the hyperparameter “number of children” in the algorithm rendered almost no impact on the result for *LossReward*. Just in some processes, very low value of this index worsened the search.

With the growing share of mutation, the result *LossReward*, firstly, improved up to a certain value (for the analyzed business processes, this limit value most often varied from 35% to 50%); however, after 80%, the algorithm switched to a random search. With very small values of the share of mutation, the algorithm worked for quite a long period of time and often looped on one and the same paths with *Loss*, without finding the *node* with *Reward*.

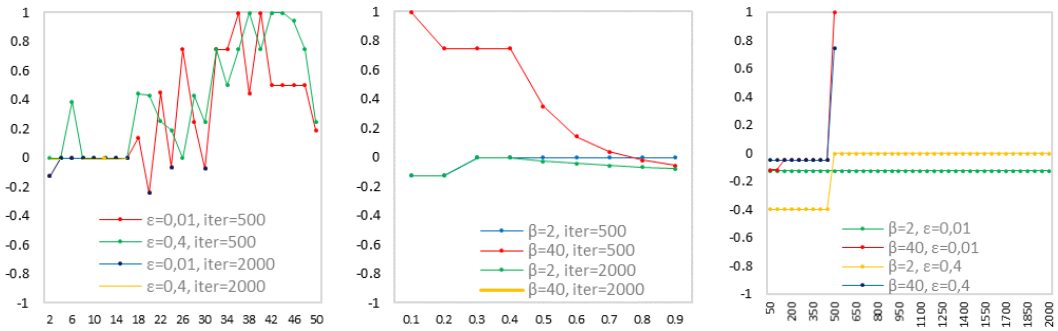
The most material impact was rendered by the number of cycles. With the maximum number of cycles (except the cases with the very minimum and very maximum values of the share of mutations), the algorithm finds the best way via the *node* with *Reward*, with the adequate number of returns, almost under any settings of the number of children and larger part of the diapason of the share of mutation. When the values of the number of cycles are not very high, the share of mutations started to influence the result.

As a result, in case of using the genetic algorithm, when a larger number of cycles is set, and when the share of mutations is set to a not very low value, the algorithm for all the processes found an optimal path. When the number of cycles was not very high, the efficiency of finding the optimal path (search for maximum *LossReward*) was influenced by the fine tuning of the share of mutations (for the studied business processes, it was equal to 35%-50%). The number of children almost had no impact on the result; as exception was the cases of extremely low values of the share of mutations and number of cycles.

Another method of reinforcement learning which was tested on business processes was the method of "cross-entropy" (Leemans & Fahland, 2020). Firouzian et al. (2019) describe the application of cross-entropy for the search for a path, but the way of assigning the environment does not fit our task because of the lack of interpretability.

In the algorithm, the standard deviation of the best transitions from it for β is calculated on each iteration for each point. Further, with the probability of (1- ε), the best calculated path is selected, and with the probability of ε – the random path is selected.

Greed search was implemented in testing for all combinations of the parameters, like in the previous two methods: β from 0 to 50 with the step of 2, ε from 0 to 1 with the step of 0,1, and number of cycles from 0 to 2000 with the step of 50.



Figures 8, 9, 10. Diagram of dependence of the average normalized LossReward on β (8), ε (9), and number of cycles (10) (including, on the "looped" logs)

All hyperparameters render material influence on the result of the search for an optimal path. When the averaging index of β is increased to a certain point, the index does not influence *LossReward*, since the algorithm does not find the node with Reward, but when reaching a certain β (for the tested business processes, β = 30 to 50), the found *LossReward* starts to grow, because the algorithm already minimizes the path on the node with Loss, each time finding the node with Reward. Subsequently, *LossReward* grows to a certain $LossReward_{max}(\beta_{opt})$ and starts to fall (not dramatically, however). Notably, $\beta_{opt}(\varepsilon)$ - the optimal coefficient of averaging depends on ε. The behavior of β with larger values of the number of cycles could not be analyzed, since the algorithm failed to come to an end (ran into a cyclic path).

The growth of the parameter ε on the average reduced the index *LossReward* with the growth of ε, where ε<0.4-0.6; and where ε>0.4-0.6, the value ε did not influence *LossReward*.

The growth of the number of cycles on the average increased *LossReward*.

On the average, for certain (but not for all) business processes of the tested ones, an optimal path in the network graph and maximum *LossReward* were found, but it was very time-consuming and was achieved through the extremely fine tuning of hyperparameters. In the majority of cases, the best variant of the selection

was as follows: to set the minimum ε, then to select the maximum value of the number of cycles, with which the algorithm does not run into a cyclic path (does not hang), then, with these two indices, to perform greed search on β. The result was not achieved through a simple maximization of the number of cycles, since the algorithm did not come to an end (looped endlessly on the *node* with *Reward*).

Having compared all four methods (q-learning, genetic algorithm, cross-entropy method and classical search for an optimal path in the network path), the following conclusions can be made. A dramatically larger number of cycles is required for a search for an optimal path in the method of the classical search for a path in the graph and q-learning, as compared with the cross-entropy and genetic algorithm; notably, when the value of ε is very small, q-learning may fail to find the *node* with *Reward* (like in our case). The simplest variant of the search for an optimal path in the network graph of the business process, provided that there is a possibility to launch a large number of cycles, is the genetic algorithm, which actually does not require setting of hyperparameters if the number of cycles is maximum. If there is no a possibility to set a large number of cycles for the genetic algorithm, there exist two alternative variants. The first of them is to use the genetic algorithm focusing on the setting of the share of mutations: if the value is very low, the algorithm will as well fail to find the node with Reward; if the value is very high, it will turn into a random search. The second of them is the cross-entropy method, but this method will require considerably larger efforts to set hyperparameters. A large value of the averaging numbers of β should be set, but after a certain *βopt LossReward* starts to fall slightly. Besides, a balance between the number of cycles and ε should be found, since in the majority of cases, the algorithm starts to run into a cyclic path on the *node* with *Reward* when it founds it. It should be noted that when hyperparameters were set correctly, the cross-entropy method for the tested processes found the optimal path for a smaller number of cycles than the genetic algorithm, but the setting of hyperparameters was considerably more time-consuming.

Arrival at the best *LossReward* with the optimal path in the network graph of the business process allows to calculate the maximum potential of optimization, i.e. to find the amount the bank can save, if absolutely all instances of the process are implemented in the optimal way only.

In the first place, the current $LossRewardn_{ow}$ for the whole network graph should be calculated as:

$$LossReward_{now}=\sum_{i=1}^{n}\left(LossReward_i \prod_{g=1}^{r} P_{ig}\right) \qquad (6)$$

where i – index number of the process node, n - total numbers of the process nodes, $LossReward_i$ – loss/reward at the i-th node of the process, r – number of transitions between the nodes of the process from the start to the i-th node, g – index number of the transition between the nodes of the process from the start to the i-th node, P_{ig} – probability of the g-th transition between the nodes along the way from the start to the i-th node.

Respectively, the maximum potential for optimization may be calculated as follows:

$$Profit_{max}=LossReward_{max}\times N_{opt}- LossReward_{now}\times N_{now}\text{-}opt\_price \qquad (7)$$

$LossReward_{max}$ – maximum *LossReward* with the optimal process (with the optimal path in the network path of the process), $LossReward_{now}$ – current *LossReward*, N_{opt} – number of instances of the process per year with the optimal process, N_{now} – number of instances of the process per year currently, *opt_price* – costs of optimization.

Calculation of *opt_price* is beyond the scope of this study. The number of instances of the process per year with the optimal process N_{opt} may be calculated as follows:

$$N_{opt} = \frac{NReward_{now}}{NReward_{opt}} N_{now} \qquad (8)$$

where N_{now} – number of instances of the process per year currently, $NReward_{now}$ – number of *Reward* in one instance of the process now, $NReward_{opt}$ – number of Reward in one optimal instance of the process.

3. Conclusion

The study offers and describes the application of reinforcement learning in order to define an optimal type of business processes in the bank. The environment was represented as a full network graph of all possible nodes of the process, where Loss – the bank's allocated operating expenses per each node of the process, Reward – net commission/interest income (or synthetic Reward). The methodology was applied to analyze eleven bank business processes. As a result, a conclusion was made to the effect that the reinforcement learning methodology is perfectly fit for a search for an optimal path in the graph of the process, for the purpose of its further inclusion in rules and regulations. Three methods of RL and classical search were compared. The results showed that the genetic algorithm and cross-entropy method are optimally suitable for solution of the task. Notably, the cross-entropy method is more sensitive to the settings of hyperparameters, and the genetic algorithm – to the number of cycles. Besides, the variant of calculation of the maximum potential of the saving from the business process

optimization was proposed. The results of the study were implemented in the operating procedures of Sberbank PJSC.

However, within the framework of the current task, there is a potential for further research. Currently, the erroneous actions of the managers performing the process, which are related with the mistakes when working in the logging system, rather than with the business process, are drawn in the network graph as the nodes with low probability. This does not distort the vision of the process materially, but can slow down data processing considerably. The crucial task is to reduce the number of nodes in the system by means of such cases. Another line of this task is to search for quick wins. The way of the definition of the optimal structure of the process described in the article allows to change the business process fundamentally, which takes time in real practice. However, business divisions often take larger interest in defining some local cases, which can be optimized quickly and bring profit.

References

[1]. Arango, M., Foster, M., Muller, R., & Vengerov, D. (2017). Business Process Optimization via Reinforcement Learning, *BIWA Summit 2017*. https://doi.org/10.13140/RG.2.2.21505.04964
[2]. Born, M., Brelage, C., Markovic, I., Pfeiffer, D., Weber, I. (2009). Auto-completion for executable business process models, *BPM 2008. LNBIP*, 17, 510–515.
[3]. de Medeiros, A. K. A., Weijters, A. J. M. M., & van der Aalst, W. M. P. (2005). Using genetic algorithms to mine process models: representation, operators and results, *Systems Research and Behavioral Science*.
[4]. Firouzian, I, Zahedi, M., & Hassanpour, H. (2019). Cycle Time Optimization of Processes Using an Entropy-Based Learning for Task, *International Journal of Engineering, 32*(8).
[5]. Huang, Z., Van Der Aalst, W. M. P., Lu, X., Duan, H. (2011). Reinforcement learning based resource allocation in business process management, *Data & Knowledge Engineering*, 70(1), 127–145.
[6]. Kudenko, D., & Grzes, M. (2009). Knowledge-Based Reinforcement Learning for Data Mining, *ADMI 2009: Agents and Data Mining Interaction*, 21-22. https://doi.org/10.1007/978-3-642-03603-3_2
[7]. Leemans, S. J. J., & Fahland, D. (2020). Information-preserving abstractions of event data in process mining, *Knowledge and Information Systems, 62*(3), 1143–1197.
[8]. Li, S., Xiao, S., Zhu, S., Du, N., Xie, Y., & Song, L. (2018). Learning Temporal Point Processes via Reinforcement Learning, *Neural Information Processing Systems Conference*.
[9]. Lin, F. R., & Pai, Y.H. (2000). Using multi-agent simulation and learning to design new business processes, *IEEE Transactions on Systems Man and Cybernetics - Part A Systems and Humans, 30*(3), 380–384.
[10]. NetworkX. Retrieved from: http://networkx.github.io/
[11]. Panfilov M., Goncharenko I., & Bugaenko A. (2019). Application of DS methods for solving applied task in finance, *AI Journey conference, Kaliningrad*.
[12]. Silvander, J. (2019). Business Process Optimization with Reinforcement Learning, International Symposium on Business Modeling and Software, *BMSD 2019: Business Modeling and Software Design*, 203–212.
[13]. Van der Aalst, W. M. P. (2011). *Process mining: discovery, conformance and enhancementof business processes*. New York: Springer.
[14]. Van der Aalst, W. M. P. (2016). *Process Mining, Data Science in Action*. Springer, 2016
[15]. Van der Aalst, W. M. P. (2017). Using process mining to deal with "events" rather than "numbers"? *Business Process Management Journal, 24*(4).
[16]. Van der Aalst, W. M. P., de Medeiros, A. K. A., & Weijters, A. J. M. M. (2005). Genetic Process Mining, *Lecture Notes in Computer Science, 14*, 48–69.
[17]. Wang, J., Wong, R. K., Ding, J., Guo, O., & Wen, L. (2013). Efficient selection of mining algorithm, *IEEE Transactions on Services Computing, 01/2013*.