Probabilistic Update Policy to Deal with Bandits for Staged Problems Applied to Cases

Carmen Constanza Uribe Sandoval and Luis Oliverio Chaparro Lemus

Abstract Some decision problems are represented in sequential stages within which an action is executed without knowing its effect until the action of the last stage is completed. A dynamic case management modeling and notation problem has been remodeled with these features to improve its automation. This chapter describes a bandit-based application with a probabilistic learning policy tested with simulated data and a stage graph and proposes its application in case automation. The results of the simulations and an initial model of the application as a graph of stages are presented.

Keywords Machine learning · Decision-making · Bandit models · Case · Dynamic processes

1 Introduction

Dynamic processes automation in organizations is a problem in the Business Process Management Suits—BPMS industry; for this purpose, Case Management Modeling and Notation—CMMN tools have emerged that allow the modeling and automation of cases. Business Process Management Notation—BPMN and CMMN—environments pursue the modeling and automation of all types of human activity (business) that can be executed as a process, whether determined or not. However, automation with CMMN can be improved (decrease its uncertainty) by

C. C. Uribe Sandoval (✉)
Universidad de Boyacá, Tunja, Colombia - Universidad Autónoma de Nuevo León, Monterrey, México
e-mail: ccuribe@uniboyaca.edu.co

L. O. Chaparro Lemus
Universidad de Boyacá, Tunja, Colombia
e-mail: lochaparro@uniboyaca.eu.co; http://www.uniboyaca.edu.co

F. Torres-Guerrero et al. (eds.), *2nd EAI International Conference on Smart Technology*, EAI/Springer Innovations in Communication and Computing,
https://doi.org/10.1007/978-3-031-07670-1_3

learning certain patterns after repeated execution of the case model implementation [8].

Decision-making has reached a large volume of work and research due to its application to private and corporate problems. Some researchers and scholars have made use of graphs to model decision-making situations [1] and have proposed algorithms to solve them, several of which have been studied in different undergraduate programs such as engineering [3]. On the other hand, Artificial Intelligence has been the basis for the development of programs that allow approximating recommendations that previously would only be made by a human expert [9], which, clearly, may also have applications in decision-making problems.

In Artificial Intelligence, three types of machine learning are used: supervised, unsupervised, and reinforcement learning. For the last one, the system does not learn from the information provided by an external subject; instead, it has to discover what it must learn according to the consequences of its actions. A simple technique within reinforcement learning is the one that allows solving the problem of the multi-arm bandit (MAB), where an agent must decide the action that generates the best reward from among a set of them, for which its associated value is only known until a large number of attempts have been made and the value can be deduced. This technique is approached in this chapter to solve stationary processes related to the management of probabilities associated with each action since it uses the bandit gradient model [11].

The set of actions available to the user to find the one that gives the highest reward can be modeled using a graph. Actually, in this chapter, the multi-armed bandit technique is used with some restrictions, suitable for problems that can be modeled in stages, where the individual value of the reward in each action is only known when the final objective has been achieved; that is, there is uncertainty throughout the process.

Uncertainty is one of the characteristics of dynamic business cases [5]. The issue shown above inspired this work; since a decision-maker does not have a standard procedure that ensures the sequence of actions for a case to be resolved but knows the actions eventually involved, the restrictions present between those actions and the final result when selecting a sequence of them; that is, the achievement or not of the expected objective. Some samples of decision problems appear in everyday life; they are made up of a set of activities that are executed, and where the reward is only observed after they have been executed many times, such as medical health care processes, in which only after performing a certain set of activities on a patient, can the effect of the process be established.

On the other hand, the multi-armed bandit technique has already been studied as an alternative solution to complex problems with little information, thanks to its characteristic balance between exploitation and exploration, which means good results with a lower computational cost compared to other techniques [14].

As an alternative that allows the recommendation of a tasks' sequence, each task belonging to a well-defined stage, with a low computational cost, we implemented the multi-armed bandit model of Reinforcement Learning in this research and describe it in this chapter.

2 Multi-Armed Bandit Problem

"Multi-armed bandits is a simple but very powerful framework for algorithms that make decisions over time under uncertainty" [10]. There are several types of multi-armed bandits related to the degree of complexity of the algorithms, the way they interact with the environment, and the mathematical supports that characterize them, among other aspects.

In [6], the bandit process is described as a single-armed bandit process, as a machine that has associated a sequence of states, each of them with a reward, and a state transition function that operates according to the states already visited and an independent real-valued random variable, and with a known statistical description. And, the multi-armed bandit process is described as a set of single-armed bandit processes with a controller that operates one of these machines for each time, based on a policy that it adopts to maximize the rewards received.

Thus, the accumulated benefit that is received when selecting an action "a" at different times i is a simple average of the rewards that have been accumulated when selecting action "a," since the reward will not necessarily be the same at different times, as it can be seen in formula 1 [11].

$$Q_t(a) = \frac{R_1 + R_2 + \ldots + R_{Nt(a)}}{N_t(a)}. \tag{1}$$

The same author [11] considers a gradient bandit when he learns a number preference $H_t(a)$ for each action a, but it "has no interpretation in terms of reward." This preference directly influences the probability that the action will be taken in the next time, according to the Boltzmann distribution (Eq. 2), and this is updated in each step, according to the reward and the average of all the rewards up through and including that time.

$$\pi_t(a) = \frac{e^{H_t(a)}}{\sum_{b=1}^{n} e^{H_t(b)}}. \tag{2}$$

Initially, all $H_t(a)$ preference values are equal, regardless of their value, so all actions have the same probability of being selected.

3 The Problem of Dynamic Cases

Business Process Management—BPM, according to [4], refers to the design, representation, analysis, and control of business processes and brings together a set of process management best practices with tools and information technologies. For [2], it is the integration of information technologies to improve, innovate, and manage business processes to facilitate the achievement of business objectives.

The management of processes has been a constant concern for researchers who have been searching for their optimization, specifically the Workflow, such as some of the models presented in [13]. It has gone from managing purely static processes to increasingly dynamic processes, which has given rise to new challenges for companies that generate Business Process Management Suites (BPMS) where all the technological developments that support business processes are concentrated [12], a challenge that this research aims to support.

As stated above, the term "case" arose, which in [13] is defined as a situation that may occur in the business for which the resolution procedure is not necessarily predefined. One of the emerging standards for modeling cases is Case Management Model and Notation (CMMN) that uses a set of graphic symbols, composition rules, and artifacts for this purpose; a comprehensive description of this notation is found in [7], but it is emphasized that the dotted lines around the activities make them optional, and this reveals the uncertainty that characterizes the cases.

An example of a case modeled with CMMN is shown in Fig. 1. This is the model that is discussed and analyzed later.

4 Model

In this way, decision problems are modeled where it may be necessary and sufficient to select one and only one of the nodes in each stage or level (see Fig. 2), to form a set that, in the end, will allow a planned objective to be reached or not. Each of the nodes in the graph has an associated bandit. At the end of many iterations, it is expected that the path found—the convergent path by this algorithm—will be made up of the actions that have achieved the highest preference, and therefore, it must match a route that will reach the goal with a high reward.

The probability of taking a path made up of the L nodes will be equivalent to the multiplication of the transition probabilities between its nodes, as expressed in Eq. 3 where i denotes the decision node selected at a given stage and L is the number of stages.

$$P(i_0 \to i_1, i_1 \to i_2, \ldots, i_{L-1} \to i_L) = P(i_0 \to i_1)P(i_1 \to i_2)\ldots P(i_{L-1} \to i_L). \tag{3}$$

Since the model must be adapted to problems where the reward associated with each node is not known, but rather a global reward for each complete path, the concept of preference for node selection of the Gradient bandit mentioned above is used. Here, the preference associated with node i is called v_i, and the transition probabilities between adjacent nodes are estimated from those preference values, as observed in Eq. 4, where A is the adjacency matrix of the network of bandits.

$$P(i \to j) = \frac{e^{v_j}}{\sum_k A_{i,k} e^{v_k}}. \tag{4}$$

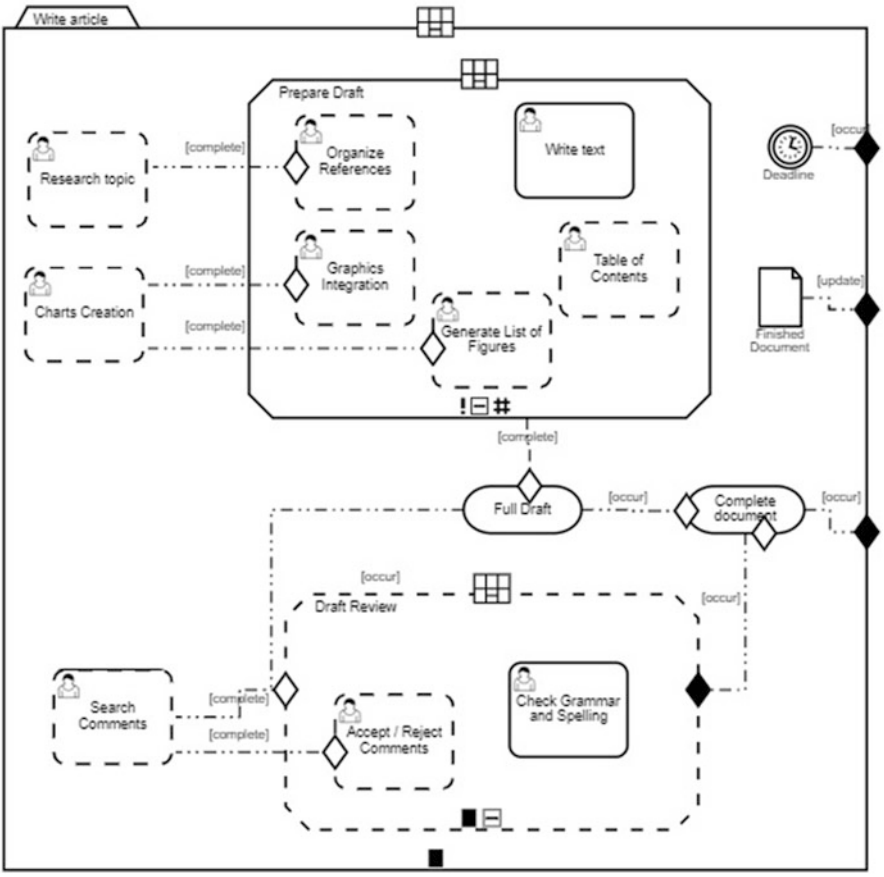


Fig. 1 CMMN diagram. Adapted from [7]

The initial values of v are zero, so the formula 4 calculates the same probability to go from one node to each of its next or neighboring nodes, that is, with a uniform probability; in this way, the decision to move from node i to another node k is totally random.

After each stage τ, the total reward is observed, and according to its sign, a positive or negative reward will be given to the preferences associated with all the nodes of the path, as Eq. 5 where $\delta \geq 0$ is a parameter that controls the learning rate.

$$v_j(\tau + 1) = v_j(\tau) \pm \delta. \tag{5}$$

Although values close to zero are expected for δ, after the implementation of the model, it became necessary to handle values with magnitudes related to the rewards of the problem being tested, as will be explained later.

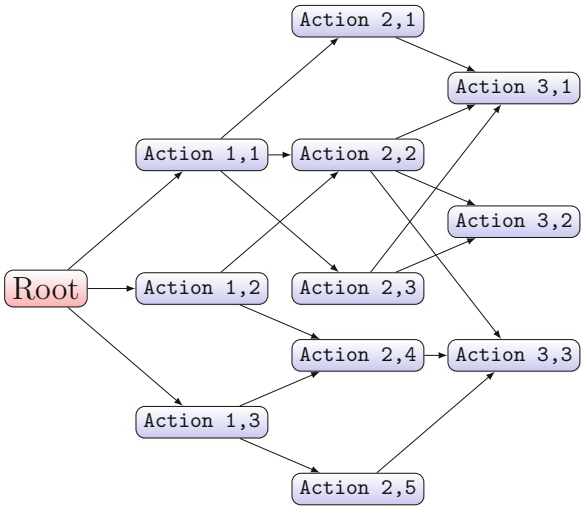


Fig. 2 A network of hidden bandits with $L = 3$ levels

These new preference values v_i of the nodes are taken into account to calculate a new route, with a selection priority for the nodes that have a higher value, according to the formula 4; this guarantees that, finally, the nodes with the highest preference value will correspond to the solution path.

5 Exploitation and Exploration

Exploration and exploitation are very important characteristics in reinforcement learning models.

Exploitation refers to the tendency of the agent to continue selecting the activity that has given the best result so far, in search of improvements. Exploration refers to the agent's ability to evaluate other activities that are not necessarily working well, looking for a better selection in the future [11].

The algorithm that is presented gives greater importance to exploration when it is beginning to learn or to know the effects of its decisions, which gradually decreases, while the importance of exploitation increases toward the end of learning. Although one of them prevails at a certain moment, there is always a percentage of probability for the other. This happens because the selection criterion is given by the transition probabilities between nodes according to Eq. 4, which with the initial values of v_i at zero guarantees uniform probabilities to pass from a node to its adjacent ones for the next stage, giving all the space to the exploration, since there is no preferred node. In each time or execution, these probabilities are updated according to the modifications made to the preference values v_i with increases or decreases δ whose magnitude has been adjusted to guarantee that a sufficient percentage of selection of alternatives that have not yet been chosen, during the first iterations.

In intermediate cases in which the values of the probabilities begin to mark their superiority over those of other transitions, there is a lower percentage of probability that other actions will be selected, increasing exploitation and decreasing exploration; but the moment will come when the nodes that make up the answer to which the search converges have almost 100% of the probability of being chosen, and it is at that moment that the answer of the recommended path is obtained.

6 Algorithmic Implementation of the Model

The generation of a graph for the tests of the learning algorithm has been implemented, whose adjacency matrix is stepped so that no node is adjacent to another of the same stage. When generating the matrix, it is also taken into account that each node must have at least one adjacent node from the next stage. The nodes of the graph correspond to the actions that are taken, and each of them is assigned a reward to calculate the gain of the path at the end. Each node is associated with a bandit that a normal distribution has a mean in its reward and standard deviation of a magnitude according to those of the rewards.

Once the gain or loss is known at the end of the iteration, the preference values of each node of that route are updated, according to the formula 5; this value is used to calculate the new transition probabilities with the formula 4, for the next iteration. It should be noted that, at the end of each stage, only the transition probabilities of the nodes involved in the selected route are affected, but that at the beginning of each iteration, all the modifications that these probabilities have undergone in the previous stages of the simulation are taken into account.

The selection for the next neighbor of each node is strongly influenced by the probabilities of transition between nodes, which give rise to the exploitation of the preferred routes, and to the exploration as explained in Sect. 5. So, at the end of the defined iterations, the simulator converges to the best route it has studied, which may be the optimal one.

Algorithm 1 shows the pseudocode of the proposed model. It knows the values of L (the number of stages), M (vector with the number of nodes of each stage), and the vector w (preference values of each node) that is initialized to zeros. Calculate n that is the total number of nodes in the graph, randomly to generate the adjacency matrix and the bandits for each node. For each iteration, the simulator calculates the transition probabilities between nodes, with which, in each stage (up to the penultimate), it selects a node and finds its neighbors and does the same in the next stage.

With the approximate bandits' values of the selected nodes, the gain at the end of the L stages is calculated. According to the value and the sign of the gain, the preference values of each node (w) are updated with which the probabilities of transition to its neighbors are calculated again for the next iteration.

Algorithm 1 L-n-bandit(L, M[L])

Calculate: n
Initialize: v[n] = 0
Generate: Ad[nxn] = Adjacent Matrix
Generate: B[n] = Real Bandits
for t = 1 **to** T **do**
  Initialize: orig = 0
  Add: orig in path
  $P(i \rightarrow j) = \frac{e^{v_j}}{\sum_k A_{i,k} e^{v_k}}$,
  for l = 1 **to** L-1 **do**
    Generate: nvz_{orig} = neighbours
    Select: $dest \in nvz_{orig}$ by $P(i \rightarrow j)$
    $orig = dest$
    Add: orig in path
  end for
  Calculate: $Gain_{path}$
  if $Gain_{path} > 0$ **then**
    v[n+1] = v[n] + δ by i $\in$ path
  else
    v[n+1] = v[n] - δ by i $\in$ path
  end if
end for

7 Numerical Experiments

The code is executed 10 consecutive times to obtain an initial estimate of its effectiveness; each time with 999 iterations, a percentage of seven out of ten executions is obtained that lead to the correct answer following the approximate generation procedure of the probability transition matrix, as can be seen in the composite Fig. 3.

Then, to estimate the performance of the algorithm and the most suitable value for the parameter δ, we take a fixed graph arranged as shown in Fig. 4, that is, with five stages and thirteen nodes, connected as shown the edges and their adjacency matrix.

This graph complies with the defined restrictions has a high density of allowed connections and in each stage has one of the values of its bandits higher than that of the other ones, to ensure that it finds the route that has the highest gain associated with it. The adjacency matrix and the bandit vector values (B) were generated as follows.

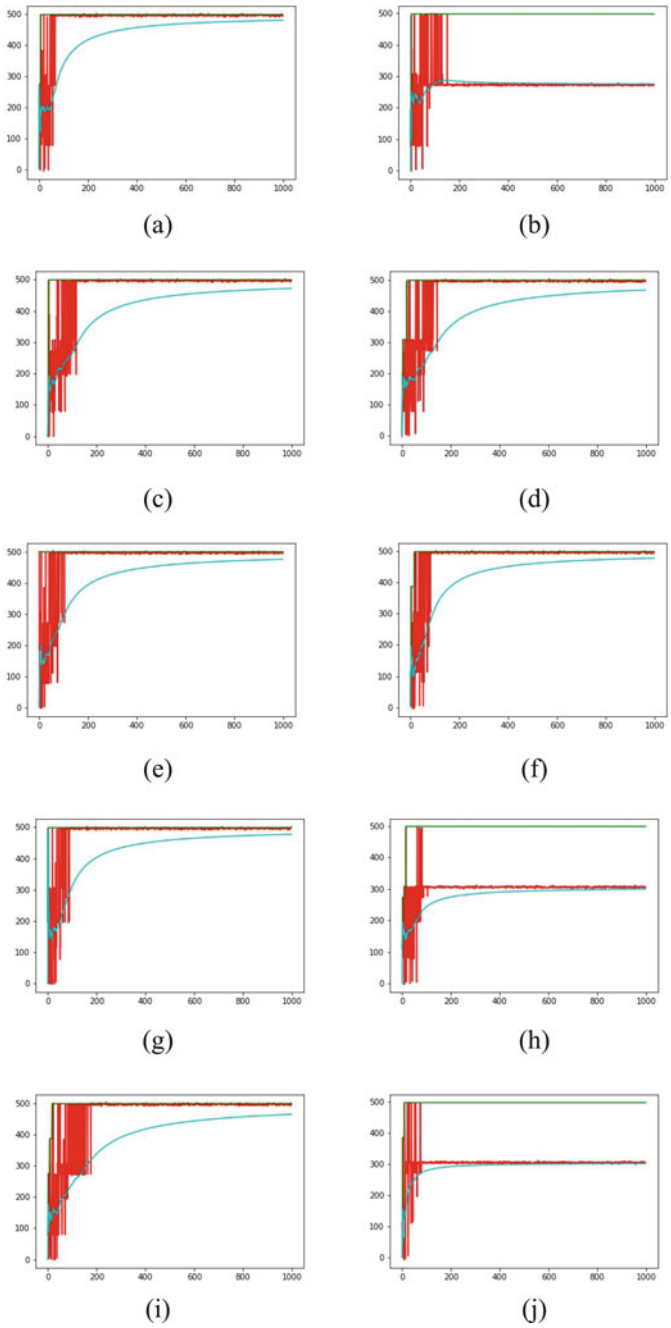
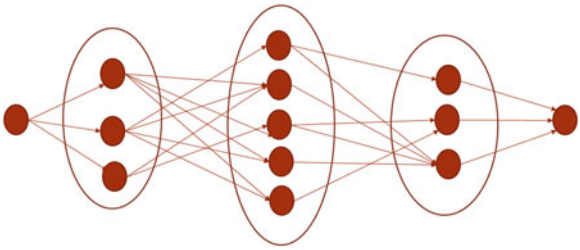


Fig. 3 Results with random graphs

Fig. 4 Model graph



$$A = \begin{bmatrix} 0,0,1,1,0,0,0,0,0,0,0,0,0 \\ 0,0,0,0,1,0,0,0,0,0,0,0,0 \\ 0,0,0,0,1,0,0,1,1,0,0,0,0 \\ 0,0,0,0,1,1,0,1,1,0,0,0,0 \\ 0,0,0,0,0,0,0,0,0,0,0,1,0 \\ 0,0,0,0,0,0,0,0,0,0,1,1,0 \\ 0,0,0,0,0,0,0,0,0,0,1,1,1,0 \\ 0,0,0,0,0,0,0,0,0,0,1,0,0,0 \\ 0,0,0,0,0,0,0,0,0,0,0,1,0,0 \\ 0,0,0,0,0,0,0,0,0,0,0,0,1,0 \\ 0,0,0,0,0,0,0,0,0,0,0,0,1,0 \\ 0,0,0,0,0,0,0,0,0,0,0,0,1,0 \\ 0,0,0,0,0,0,0,0,0,0,0,0,0,0 \end{bmatrix}$$

$$B = [-0.833e{-}01,\ 6.12e{+}02,\ 9.49e{-}02,\ 3.18e{+}00,\ 3.75e{-}01,\ 6.33e{-}01,\ 7.73e{-}01,\ 5.47e{-}01,\ 5.77e{+}02,\ -6.95e{-}01,\ 8.23e{+}02,\ -1.83e{+}00,\ -3.13e{-}01].$$

The result obtained with 99 iterations for the proposed model is the 5-stage nodes vector [0,2,4,11,12] that is the optimal vector. In Fig. 5, it can be seen how the average gain is approaching to the real gain and how the gain converges to the maximum real gain curve.

In the same figure, the upper curve corresponds to the best-tested route gain calculated with the assigned bandit value; the most unstable curve shows the gain obtained for the selected route, to the extent that the automaton is learning the value of the bandits; and the lower greenish curve shows the accumulated average of these gains, which converges to the first line.

We proceed to establish how the value of δ influences, to give an informed recommendation on the value that should be used.

The value of δ in Eq. 5 was initially selected with values between 0.1 and 0.9, which are positive in case of gain and negative in case of loss, to motivate or demotivate the automaton, so that in the next step select the nodes that receive the reinforcement. The normalized differences that were found between the real gain and each of the gains that were obtained on average in 100 iterations for each parameter's combination are shown in Table 1.

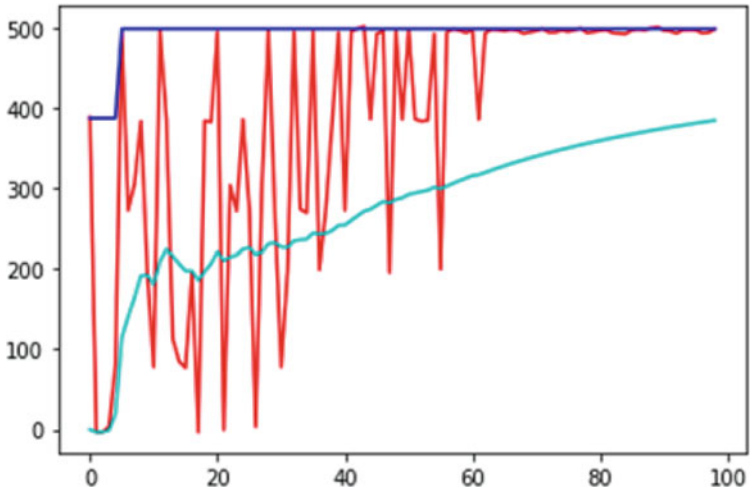


Fig. 5 Probability-weighted result

Table 1 Results with δ

	Iterations			
Deltha	100	300	600	900
1e−1	0.7146873	0.6781488	0.7089705	0.7325367
1e−2	0.6965094	0.6706974	0.6768052	0.6542384
1e−3	0.2909915	0.2335418	0.2185707	0.1505568
1e−4	0.6362933	0.3897829	0.2263654	0.1647789
1e−5	0.7059966	0.6926970	0.6712759	0.6435113

Table 2 Results with gamma

	Iterations					
Gamma	100	300	500	700	900	1000
0.00001	0.368	0.360	0.344	0.328	0.322	0.272
0.00025	0.102	0.044	0.034	0.066	0.052	0.042
0.00050	0.050	0.078	0.082	0.072	0.082	0.074
0.00075	0.080	0.092	0.086	0.090	0.078	0.090
0.00099	0.092	0.102	0.084	0.084	0.100	0.106

The difference ranges are over 15%, with some little differences for the number of iterations that are executed. Best δ values are in magnitudes of thousands as are viewed in Table 1.

To improve it, we proceed to create a self-adjusting δ, which depends on the value of the gain received by the tested route. A new variable γ is involved, which, when multiplied by the profit or reward received at the end of the path, will give the value δ for Eq. 5.

In fact, it was necessary to take values for γ of the order of ten thousandths, so that when multiplying it by the gain, which for this example reaches values close to 500 units, values of δ less than one can be obtained as viewed in Table 2.

Due to it, the results improved considerably, obtaining an average in the error for this table of 13% and of 8% eliminating the first row where the highest values are found. Also, it can be seen in this table that the errors are influenced both by the value of Gamma and by the number of iterations that are executed.

8 Adaptation of the CMMN Model

Taking into account Fig. 1, it is established that it is possible to express the information recorded there in the form of the graph proposed in this chapter. As the dotted lines that surround some activities make them optional, they must be replicated in several stages to offer the possible alternative ways to achieve the final goal: the complete document of the article.

The two well-defined sectors of actions in Fig. 1 are analyzed separately, and the graphs in stages of Figs. 6 and 7 are proposed. The idea is that once the graph of the "Prepare draft" stage is concluded, the paths are continued with the graph of the "Review draft" stage.

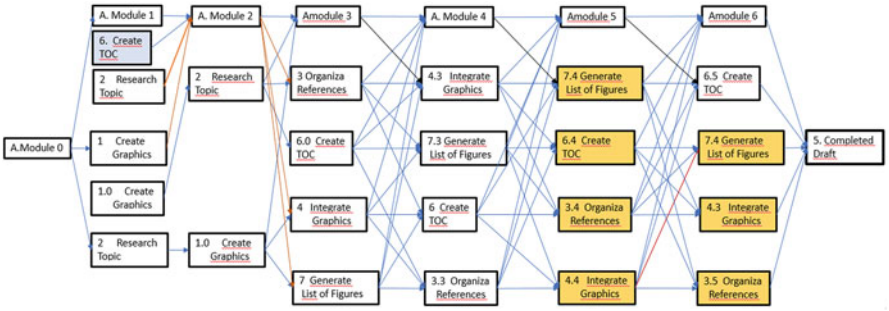


Fig. 6 Prepare draft in stages

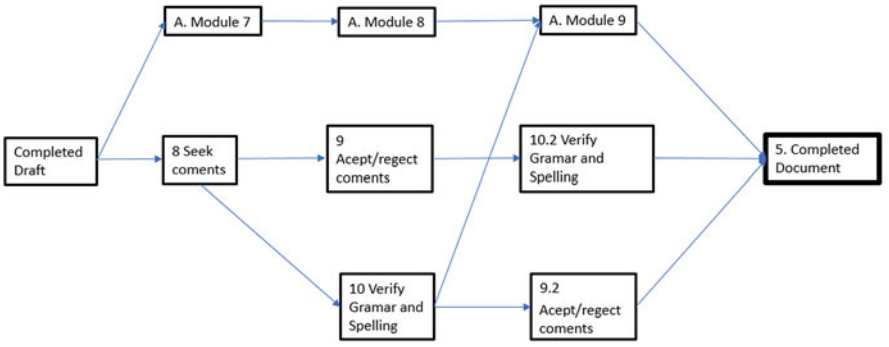


Fig. 7 Review draft in stages

In addition to the fact that it was necessary to repeat activities in each step of the graph, a null activity was also added in some stages of Fig. 6, so that in this part, a sequence with less than seven activities can be formed, even with only the mandatory activity "write text."

This is the beginning of a new experimental research work that establishes the validity of this proposal and its implementation in Case Management in some BPMS.

9 Conclusions

Various topologies of graphs are found in the literature, but the step graph proposed in this paper is novel and allows the particular modeling of some problems in which activities have to be selected in sequential times, where the result of the decisions that are taken in each stage, it will only be known at the end of an amount L of given stages.

Reinforcement learning and, in particular, the multi-armed bandit (MBA) model offer good results, with low computational cost, thanks to the importance it gives to both exploration and exploitation when searching for solutions.

The calculation of the transition probabilities between nodes, implemented in this proposal, makes it possible to recommend the best possible solution vector. The proposed model was tested with a 5-stage graph with 13 nodes and reaches its convergence in less than 100 iterations.

The BPMS industry can benefit from the proposal that was released, so it is necessary to recommend the continuity of this work in specific cases.

On the other hand, it is necessary to finish the implementation of the model in CMMN cases, making tests in several topics until generalize for any case of any topic.

The proposed algorithm for the stage problems could be improved, compared with other algorithms, and tested for more scenarios.

Bibliography

1. A. Baykasoglu, A review and analysis of "graph theoretical-matrix permanent" approach to decision making with example applications. Artif. Intell. Rev. **42**, 573–605 (2014). https://doi.org/10.1007/s10462-012-9354-y
2. T. Benedict , N. Bilodeau, P. Vtkus, M. Powell, D. Morris, M. Scarsig, D. Lee, G. Field, T. Lohr, R. Saxena, BPM CBOK Version 3.0: Guide to the business process management common body of knowledge. CreateSpace Independent Publishing Platform (2013)
3. S. Even, *Graph Algorithms* (Cambridge University Press, Cambridge, 2011)
4. K. Garimella, M. Lees, B. Williams, *Introducción a BPM para Dummies® Edición especial de Software AG* (Wiley Publishing, Inc. Indianápolis, 2008)
5. R.J. Madachy, *Software Process Dynamics* (Wiley, Hoboken, 2007)

6. A. Mahajan, D. Teneketzis, Multi-armed bandit problems, in *Foundations and Applications of Sensor Management* (Springer, Berlin, 2008), pp. 121–151
7. Object Management Group, *Case Management Model and Notation (CMMN)* (Object Management Group, Needham, 2013)
8. Object Management Group, *Case Management Model and Notation (CMMN) Version 1.1* (Object Management Group, Needham, 2016)
9. J.-C. Pomerol, Artificial intelligence and human decision making. Eur. J. Oper. Res. **99**(1), 3–25 (1997)
10. A. Slivkins, Introduction to multi-armed bandits (2019). arXiv:1904.07272
11. R.S. Sutton, A.G. Barto, *Reinforcement Learning: An Introduction* (MIT Press, Cambridge, 2018)
12. W.M.P. Van Der Aalst, Business process management demystified: a tutorial on models, systems and standards for workflow management, in *Advanced Course on Petri Nets* (Springer, Berlin, 2003), pp. 1–65
13. W.M.P. Van der Aalst, M. Weske, D. Grünbauer, Case handling: a new paradigm for business process support. Data Knowl. Eng. **53**(2), 129–162 (2005)
14. P. Zhou, J. Xu, W. Wang, Y. Hu, D.O. Wu, S. Ji, Toward optimal adaptive online shortest path routing with acceleration under jamming attack. IEEE/ACM Trans. Netw. **27**(5), 1815–1829 (2019)