AI Planning for Construction of Personal Learning Pathways

Irina Grishanova[1,†], Julia Rogushina[1,2,*,†]

[1] *Institute of Software Systems of the National Academy of Sciences of Ukraine, 40, Ave Glushkov, Kyiv, 03181, Ukraine*
[2] *Institute for Digitalisation of Education, National Academy of Educational Sciences of Ukraine, 9 M. Berlynskoho Str., Kyiv, 04060, Ukraine*

Abstract
We propose an algorithm for service composition based on the AI planning algorithm Q-learning and its optimization, aimed at selecting optimal service sequence with use of quality criteria (QoS). We analyze practical application of the proposed method in the task of personalized planning of educational materials study by students. Modified Q-learning method is used for generation of personal learning pathways where learning objects (LOs) with their metadata and their functional and non-functional properties are considered as services with specific QoS. Q-learning method generates a composite service that is represented by partially ordered set of accessible LOs that is optimal for a particular student in current environment by selected criteria.

Keywords
Q-learning, composite service, AI planning algorithm, learning object, personal learning pathway

1. Introduction

The importance of personalized learning of adult learners with diverse experiences has significantly increased under the war conditions in Ukraine as it enables the effective use of human resources and optimization of study duration. Traditional approaches for learning planning often prove to be insufficiently flexible or overly complex to implement in large educational programs and within tight time constraints.

A personalized approach allows for the definition an individual set of *learning objects* (LOs) and the creation a study plan tailored to the specific needs of the learner. However, this process demands substantial effort from qualified experts in adult learning and involves the consideration of numerous characteristics, including student goals and competencies, accessible learning tools and means for dynamic reaction on changes within the educational environment.

The application of reinforcement learning algorithms that can make decisions in complex and dynamic environments, supports the generation of such LO sequences based on selected efficiency criteria, such as study duration, resource costs and the quality of competencies given by LOs.

2. Problem definition

The main goals of LO concept involve the reuse of heterogeneous information modules developed for learning purposes [1]; their unified indexing that enables their search, storing and selection in special repositories; interaction between such objects, including their comparison [2]. In our study,

ITS-2024: Information Technologies and Security, December 19, 2024, Kyiv, Ukraine
* Corresponding author.
† These authors contributed equally.
i26031966@gmail.com (I. Grishanova); ladamandraka2010@gmail.com (J. Rogushina)
0000-0003-4999-6294 (I. Grishanova); 0000-0001-7958-2557 (J. Rogushina)

we consider LOs as a specific subclass of information objects accompanied by metadata that characterize different aspects of their use for learning purposes. LO metadata allows:

- retrieving relevant information from appropriate LO repositories according to its semantic similarity to some particular educational task (for example, metadata can be processed in requests for additional information about some learning course or for student to acquire their competencies);
- ensuring reuse of LOs developed for one learning course in other courses or in different educational tasks;
- determining semantics of domain-specific relations between LOs (for example, to specify the possible order of LO study or to find the intersection of LO content).

Let us introduce the basic terms that we use in this paper to define the practical educational task that can be solved on base of Q-learning.

Learning Object (LO*)* is a discrete element of educational content, accompanied by metadata and description of its role and functions in the learning process. Each LO can be formalized as a service, where the input data represents the learner's requirements, and the outputs are the competencies gained by studying the content of the LO.

Learning course (LC) is a standalone unit of learning content used for educational purposes that can be defined by: its topics and competencies received by student who study this course; requirements (competencies, skills and knowledge) of student who has an opportunity to understand course content; forms of content representation; access conditions, etc. LCs can use one or more LOs for representation of knowledge provided to students.

Personal Learning Trajectory (PLT) is the concrete route the learner actually follows inside one or more pathways for implementing the learning process for one or more LCs. It takes into account the type, form, and goal of education, the learner's competencies, skills, goals and preferences; knowledge about learning course; competencies and possibilities of teacher; set of pertinent LOs that are similar to learning course competencies; intermediate results of the learning process; test results and progresses [3]. It should be noted that defining a set of LC competencies, retrieval for pertinent LOs in repositories and other information storages, transformation LO metadata according to the needs of the task PLP constructing is beyond the scope of this study, but it uses the authors' previous research.

Personal learning pathway (PLP) is a PLT element of represented by partially ordered set of accessible LOs that defines the possible way of LO study for particular student according to his/her current competencies so that each next object is feasible for the learner’s current skills and constraints. The pathway reflects prerequisites and practical limits and is intended to move the learner toward the target competence [4].

Algorithm proposed in this work is adapted from our prior work [5] that addressed the problem of constructing a composite service using the Q-learning method [6]. *Service composition* is the process of selecting and combining individual services to achieve some previously defined final result. Here, both the functionality and the *quality of services* (QoS) [7] are critical evaluation criteria. The study of each LO by student we consider as a service characterized by:

- *functional requirements*: individual competencies (knowledge) necessary to study a given LO (inputs) and the competencies (knowledge) gained as a result (outputs).
- *non-functional properties* (QoS): some quality of services parameters such as cost, study duration, course rating, etc.

In accordance with the definitions provided above, the task of this research can be formulated as follows: using the functional parameters of the available LOs we have to construct a set of possible PLPs. Study of these PLPs achieves a defined set of the target competencies (a target state).

Then we have to determine the optimal PLP on base of the non-functional properties of analyzed LO services. A key requirement is to incorporate LO service QoS: cost, duration, quality rating, etc.

LO services are represented in a dynamic information environment. Both the set of available LOs and their parameters, as well as the initial and target states of the student, can be changed. Moreover, evaluation criteria and the relative priorities of LO parameters can also be changed.

We propose to use the *Reinforcement Learning method* (Q-learning) for PLT development to generate multiple possible PLPs (according to knowledge about LC domain) and then to select the optimal PLP based on characteristics of LO services.

If LOs are considered as services, then PLP construction can be seen as a specific case of service composition. In this context we use such understanding of ML terms:

- *initial state*: the set of student competencies before the start of study;
- *target state*: the set of competencies that student can achieve through the LC study.

The set of available LOs and their properties can be changed dynamically, while LO metadata allows determining the QoS values for the respective services can be changed as well.

3. Machine Learning and Components of ML Tasks

Machine Learning (ML) is a field of AI that studies algorithms that learn from data and experience to improve task performance without explicit programming; it uses statistical models to detect patterns and to make decisions in a data-driven way [8,9]. *Reinforcement learning* (RL) is a method for sequential decision making in which an agent acts in its environment and immediately observes the result of each choice.

After every step the agent receives scalar feedback: a reward when the action helps to reach the goal and a penalty when it does not. Using this stream of feedback the agent adjusts its behavior to maximize the total long-term return over a sequence of steps. [10].

The Q-learning method is a subclass of Reinforcement Learning techniques. In Q-learning the agent keeps a table of scores *Q(s,a)* for "state-action" pairs. After each step it increases the score of choices that lead to better outcomes and decreases the score of choices that do not. Over time the table helps the agent pick actions that bring a higher long-term reward.

We represent the PLP construction in terms of the ML task main components:

- *Environment* is a *set* of LOs defined by inputs (existing competencies) and outputs (new competencies);
- *Agent* is a mean for generation of the partially ordered LO set to achieve a specific goal;
- *State* is the set of competencies that agent possesses at a given moment in time;
- *Action* is the LO study selected by agent from the set of available objects;
- *Goal* is an acquiring a required set of competencies (Target state).

Main objective of ML use is to determine the sequence of actions (LO study) that transitions the student from the Initial state (prior competencies) to the Target state (desired competencies) with maximum efficiency.

AI methods provide the capability to automate action planning in dynamic environments to achieve specified goals. The specifics of the proposed approach is the adaptation of learning process planning to personified student's needs by use of Q-learning [11]. This adaptation is achieved by the selection of LOs that correspond to the defined goals. Q-learning enables the identification of the agent's optimal action strategy.

The reasons for the Q-learning algorithm choice are determined by its differences from other ML methods. Now many other AI methods are developed, but they have differences and shortcomings that can be critical to solving our problem:

- Greedy algorithms do not take into account future rewards and can generate non-optimal solutions;
- Dynamic programming requires complete knowledge of the environment that is not possible to PLP generation in real time;
- Graph algorithms are not adapted to the student actions and do not change the learning strategy dynamically;
- Deep learning (DNN, RNN) requires large amounts of data and complex processing of data sets that takes a lot of time.

Q-learning balances between ease of implementation, adaptability and efficiency. Thus, Q-learning is suitable for PLP building because it adapts to the competencies of learner for study path optimizing, takes into account values of LO QoS parameters and balances the quality of learning, flexible and easily scalable for LO number and structure complexity.

During the learning process, the agent selects LO services matching the available inputs and evaluates the reward for each selected service, taking into account QoS parameters such as course duration, cost, rating and learning complexity, in accordance with their semantics. For example, a learner might choose one service for studying a programming language after utilizing a foundational service about programming theory, considering factors such as one LO requiring more time, while another is less expensive.

4. Methodology of PLP construction

Proposed methodology of PLP construction based on RL contains the next elements:

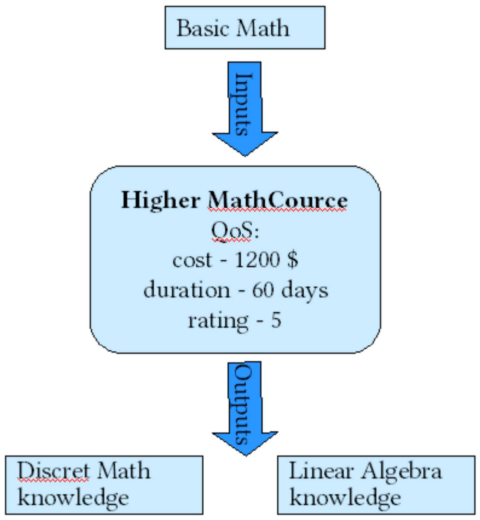

Figure 1: An example of inputs, outputs and QoS characteristics of LO service.

1. *Representation of the Learning Environment:* Each LO study is represented as a service that is defined by (Figure 1):

- *Inputs*: Prior knowledge (competencies) required to LO study;
- *Outputs*: Knowledge (competencies) gained after LO study;
- *QoS* (Quality of Service): Quality characteristics such as LO duration, cost and rating.

Every PLP is represented as a graph (Figure 2), where the nodes correspond to LOs, and the edges correspond to LO inputs and outputs.

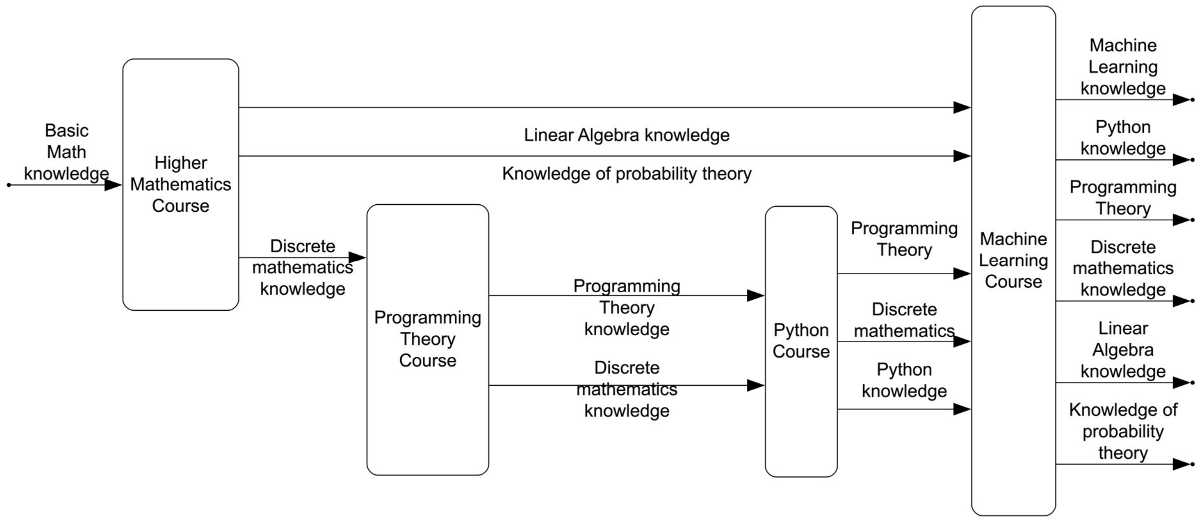
Basic Math knowledge
Higher Mathematics Course
Linear Algebra knowledge
Knowledge of probability theory
Discrete mathematics knowledge
Programming Theory Course
Programming Theory knowledge
Discrete mathematics knowledge
Python Course
Programming Theory
Discrete mathematics
Python knowledge
Machine Learning Course
Machine Learning knowledge
Python knowledge
Programming Theory
Discrete mathematics knowledge
Linear Algebra knowledge
Knowledge of probability theory

Figure 2: An example of PLP for study "Machine Learning" competence.

2. *Q-learning application*: Q-learning algorithm is applied to identify the optimal PLP from the feasible ones. This method enables the discovery of optimal action strategies in environments. During ML process, the agent evaluates the value of each LO based on QoS parameters, including course duration, complexity and rating. The outcome of this process is the optimal sequence of LOs.

3. *The workflow algorithm* at a general level includes the next steps:

- The initial state is determined by the actual set of student's competencies;
- The agent selects the LO that transform the set of competencies closer to the target state;
- Training concludes if the target state is obtained or the predefined number of iterations is completed.

Let's consider the key elements of Q-learning in the Task of LO Service Composition.

State. The set of current input parameters that determine the possibility of invoking a specific LO service. The set of all possible states represents the state space S, where the Initial and Target states are separately distinguished. In addition, we highlight separately the Terminal state, Negative and/or unpromising states. Terminal state is a state where the agent cannot perform any operations or reaches its goal. Negative and/or unpromising state is a state where the agent cannot find any action that would bring it closer to the Target state, i.e. it cannot move towards the goal, and therefore the episode must end with a penalty. A state is considered as negative in the following situations: 1. all actions are impossible for this state (i.e. no service can be performed); 2. all available actions (services) cannot transfer the agent to new useful state (i.e. any actions do not bring the agent closer to the Target state).

If an available action does not lead to the target state or does not add useful outputs, it can be considered *erroneous* and a penalty can be imposed for its execution.

Action: As stated above, action is the invocation (execution) of a LO service that transforms data (changes state of agent).

Transition Function: This rule determines the resulting state after invoking a specific service. The transition function maps how an operation on the current state (invoking a specific service) leads to a new state. Formally, it can be represented as: $T(s,a)=s`$ [12], where *s* is a current state (set of available input and output parameters); *a* is an action (service invocation); $s`$ s a new state (set of newly available parameters).

According to the WSC-MDP model [13]:

- each transition (service execution) has a transition probability $P(s`|s, ws)$ and an expected reward for the transition $R(s`|s,ws)$ defined by P and R;
- the sum of transition probabilities at a single service node always equals 1.

In the terms of our task, transition function depends on:

- *Outputs of the service*: These are added to the current state to form the next state;
- *Constraints on service availability*: Conditions that determine whether a specific service can be executed.

To simplify the problem, we represent the transition function in our implementation as a *transition table* that defines the next state for each state-action pair. This table is generated based on LO inputs and outputs. The transition function can be defined both with and without accumulation. We use the approach with accumulation, as it better reflects real-world scenarios: the set of student competencies expands during the LO study process. This concept can be illustrated by the following example (Figure 3). Let's assume the initial system state $s=\{X_1, X_2\}$. As a result of invoking the service Service_1, that has inputs $\{X_1, X_2\}$and outputs $\{X_3\}$, this operation modifies the state by adding a new output: $s`=\{X_1, X_2, X_3\}$.

Reward Function evaluates the quality of the chosen action based on QoS parameters (e.g., availability, execution time, throughput, etc.). We consider it in more detail below.

5. Key elements of Q-learning

Let's consider the key elements used by Q-learning method.

State (s) is defined by the available input and output data.

Action (a) is a specific service selected by the agent that can be executed in the current state.

Q-table is a table that stores values *Q(s, a)*, where s is a current state, a is an action that agent can take in this state.

$Q(s,a)$ is an expected value of this action in the given state, considering future rewards.

The base rule for updating the Q-value $Q(s,a)$ in Q-learning is:

$$Q(s,a)=Q(s,a)+\alpha*(R+\gamma*max\,Q(s`,a`)-Q(s,a)), \quad (1)$$

where:

- R is a reward received for performing action a in state s;
- $s`$ is a new state reached after performing action *a*;
- $a`$ is the best possible action for the next state $s`$;
- α (Learning Rate) determines how much the new information influences the current Q-value;
- γ (Discount Factor) accounts for the importance of future rewards that balances immediate and long-term rewards;
- $max\,Q(s`,a`)$ is the maximum Q-value of all possible actions in the new state $s`$.

Formula (1) incrementally defines updates the Q-value for a given state-action pair and helps the agent to learn the optimal policy by balancing immediate rewards and the expected future outcomes.

Reward(R) is a reward (positive because the agent's action contributes to achieving the goal) or penalty (negative because the agent's action leads to an undesirable outcome) received by the agent after performed action.

The reward in the Q-learning algorithm is the value that the agent receives after performing a specific action (invoking a web service) in a given state. It indicates how beneficial or detrimental the action was in achieving the ultimate goal, which is obtaining the target output data in the composite service.

We propose to use additional parameters**:** *Global Weights of QoS* and *Global QoS Modality***.** Global Weights are coefficients used to establish the relative importance of various QoS parameters (e.g., cost, duration, rating) in the decision-making process. They determine influences of parameters on efficiency and the selection of the optimal service composition path. The parameters of *Global QoS Modality* define the mode of optimality interpretation for QoS values (what is better - the maximal or the minimal ones). So, according to these values, the search for the optimal is determined by the requirement to maximize or minimize a certain *QoS* parameter.

Using the above parameters of Global Weight and Global Modality, we define the final calculation formula of the total value of Reward(R) as:

$$R(s,a)=\sum_{i=1}^{n} w_i * r_i(s,a)-1, \tag{2}$$

where

- n is a total number of QoS parameters;
- w_i is a weight (global importance) assigned to the *i*-th QoS parameter;
- $r_i(s,a)$ is a normalized value of the *i*-th QoS parameter (e.g., cost, duration, rating), determined by the following rule:

  $r_i(s,a) = att_i(s,a) - att_{min} \big/ att_{max} - att_{min}$ if parameter i needs to be maximized, or

  $r_i(s,a) = 1 - \dfrac{att_i(s,a) - att_{min}}{att_{max} - att_{min}}$ if parameter i needs to be minimized;
- $att_i(s,a)$ is the value of the QoS parameter for action a a in state s, and att_{min} and att_{max} minimum and maximum QoS values among all possible services.

Here is a brief explanation of formula (2). Normalizing QoS parameter values brings them to a common scale with values from 0 to 1 to avoid the influence of different measurement units. We take into account the semantics of QoS parameters (objective of maximization or minimization): if parameter needs to be maximized, we use its normalized value, and if parameter needs to be minimized, we invert its normalized value in the negative one.

Additionally, the normalized QoS parameter values are weighted according to their relative importance for current task. Then we sum up all weighted normalized values of parameters and apply penalties for redundant steps to discourage unnecessary ones: the reward is reduced by some constant value (we use -1). This approach allows for consideration of all QoS parameters and ensures an effective selection of the optimal path while adhering to the conditions of maximization or minimization.

We use formula (2) for aggregation of multiple QoS parameters into a single reward value, reflecting their relative importance in the decision-making process and optimizing the selection of LO services in the composition and enables the normalization of quality attributes to obtain a unified reward value.

This value can be utilized to enhance the efficiency of the Q-learning algorithm, ensuring an optimal combination of LO services in the composition while considering multiple QoS parameters with different importance and semantic modality.

In addition, we add parameter for *QoS Aggregation Method* to determine how the composite PLP need to be calculated (by summation or averaging).

Policy (π) is an agent strategy used to select actions based on the current information.

In Q-learning the agent follows an *ε-greedy policy*: agent chooses a random action with a probability ε; agent chooses the action with the maximum Q(s,a) value with 1-ε.

6. Main steps of Q-learning algorithm for PLP construction

Taking into account the specifics of the task and its conditions and constraints, Q-learning algorithm used for PLP construction consists of the following general steps and substeps:

- *Initialization* defines the initial state as the student's current competencies; sets the target state as the desired set of competencies; initializes the environment: space of states, transition table, QoS variables, Global QoS Weights and Global QoS Modality, QoS Aggregation Method (summation or averaging), Q-table; forms transition table;
- *Agent Training* implements the ε-greedy strategy to balance exploration and exploitation; calculates rewards based on QoS parameters and goal achievement; incorporates global weights and modalities of QoS parameters to refine reward calculations;
- *Optimization* prevents dead-end states; excludes excessive or unnecessary actions (check for terminal, negative and/or unpromising states);
- *Optimal Solution Retrieval* determines the optimal path based on QoS parameters from the generated paths and calculates the composite QoS values in accordance with QoS Aggregation method.

7. Example and explanation of test results

Let us consider an example of PLP graph generation on base of proposed approach. We analyze LC "Machine Learning Knowledge" that operates with the set of competencies (Table 1) that students achieve from study of some relevant LOs and then can use them to study more complex LOs. The vertices of this graph represent LOs, and the edges represent actions or the choice of a specific LO.

Table 1
Set of LC competencies

Competence denotation	Competence title
X1	"Basic Mathematics"
X2	"Linear Algebra Knowledge"
X3	"Probability Knowledge"
X4	"Python Knowledge"
X5	"Advanced Python Knowledge"
X6	"Calculus Knowledge"
X7	"Machine Learning Knowledge"
X8	"Data Science Expertise"
X9	"Programming"

The following LOs are relevant to LC and use competencies X1-X9 in their work (Table 2):

We start with an initial state that includes "Basic Mathematics" and aim to achieve knowledge of "Machine Learning Knowledge". In the graph (Figure 3), we see various LOs and possible paths of their study. This graph shows multiple possible PLPs from base knowledge to specialized expertise in the fields of machine learning and data science. It uses information represented into Table 1 and defines transformation of input data about student competence into the resulting one that is caused by learning of selected LCs.

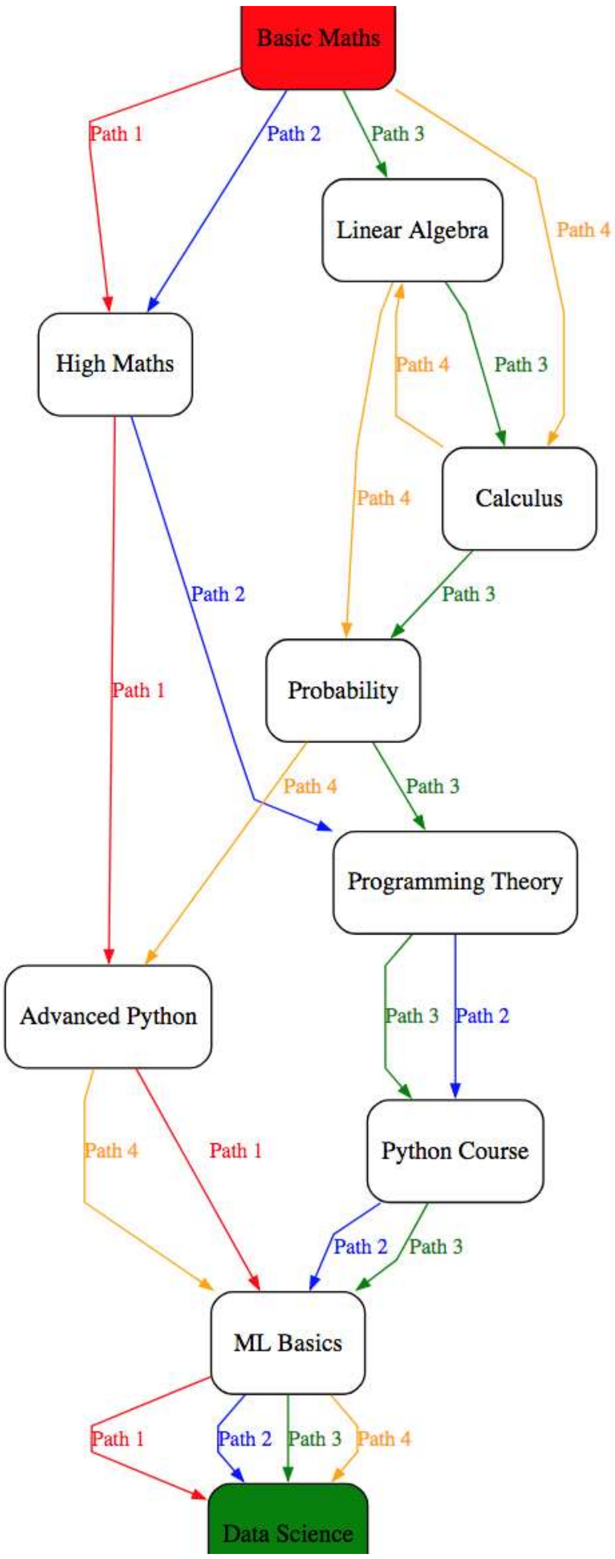

Figure 3: Example of LC competency study pathways for PLP

To analyze the properties of the proposed algorithm, we develop its software implementation using Python language with standard libraries. In addition, we use DiGraph library to visualize optimal path of LO study where edges represent actions (LO study) and nodes represent states (LO inputs/outputs). Information generated by DiGraph we use for building of Figure 4.

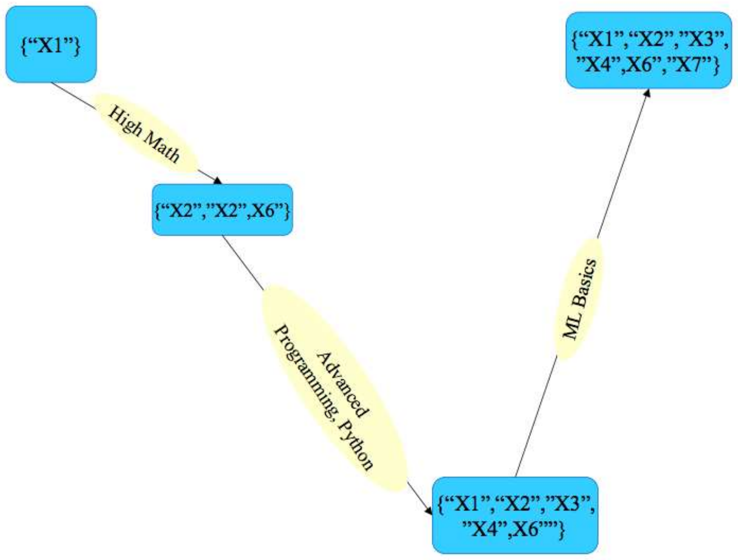
{“X1”}
High Math
{“X2”,”X2”,X6”}
Advanced Programming, Python
{“X1”,“X2”,”X3”, ”X4”,X6””}
ML Basics
{“X1”,“X2”,”X3”, ”X4”,X6”,”X7”}

Figure 4: Visualization of the optimal PLP.

Table 2 shows information about used LOs.

Table 2
Set of available LOs, their input and output competencies

LO title	LO requires competencies	LO provides competencies
“Linear Algebra”	"Basic Mathematics"	"Linear Algebra Knowledge"
“Probability theory”	"Basic Mathematics"	"Probability Knowledge"
“Calculus”	"Basic Mathematics"	"Calculus Knowledge"
“High Math”	"Basic Mathematics"	"Linear Algebra Knowledge", "Probability Knowledge", "Calculus Knowledge"
“Programming Theory”	"Basic Mathematics"	"Programming Theory Knowledge"
“Python”	"Programming Theory Knowledge"	"Python Knowledge"
“Advanced Programming Theory&Python”	"Basic Mathematics"	"Programming Theory Knowledge", "Python Knowledge"
“ML Basics”	""Calculus Knowledge", "Linear Algebra Knowledge", "Probability Knowledge", "Python Knowledge"	"Machine Learning Knowledge"
“Data Science”	"Machine Learning Knowledge"	"Data Science Expertise"

For testing, we generate a set of 1,000 services with 100 inputs/outputs, resulting in 27,357 states. The test was performed on a MacBook Pro with an Apple M1 Pro chip and 16 GB of memory. The execution time is acceptable (table generation time: 315.1149 seconds, training time: 6.6559 seconds), and the constructed LO study path demonstrates high quality.

The reward graph across episodes (Figure 5) illustrates how the reward is changed during the agent's learning. As observed, the initial reward is low but gradually increases, indicating the agent's learning process and its ability to find optimal PLP. The positive trend in the graph demonstrates the success of the learning and improvements in the Q-learning algorithm.

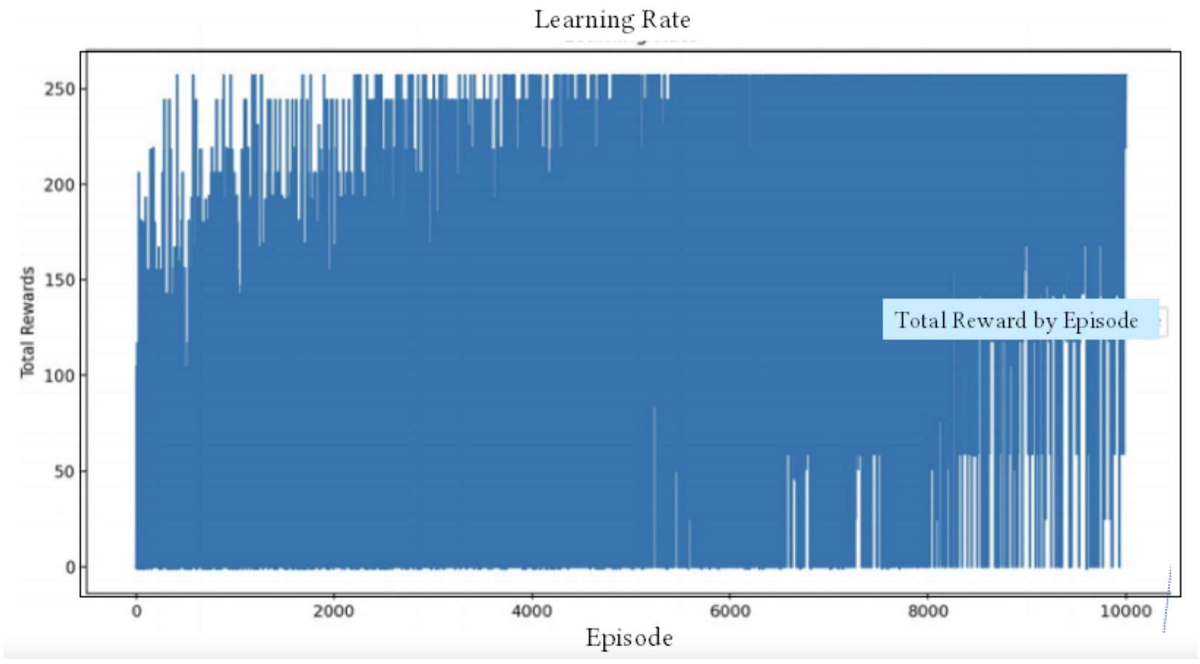

Figure 5: The Learning Rate graph (rewards across episodes)

The TD Error graph (Figure 7) reflects the temporal changes of difference error value during learning process. At the beginning, the error is high because an agent is just starting to explore the environment. Over time, the error decreased, indicating improvements in the agent's predictions regarding future rewards and the stabilization of the learning process.

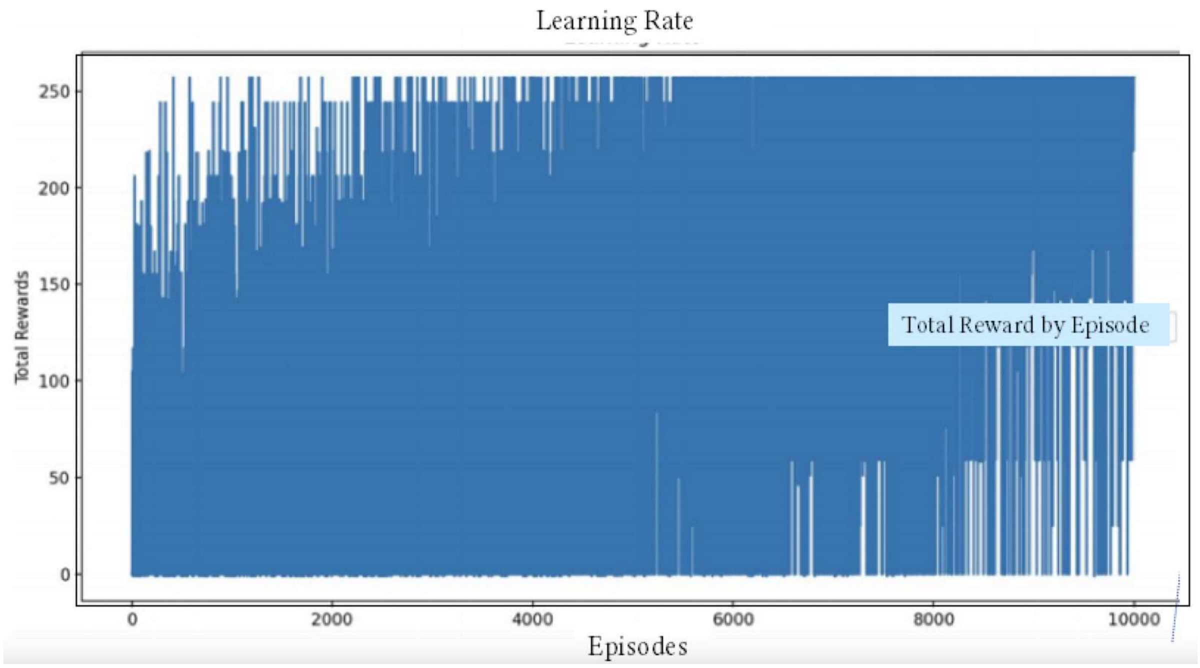

Figure 6: The temporal difference error graph across learning episodes

8. Differences of proposed method from basic Q-learning

Proposed method for construction of LO service compositions differs from the classical Q-learning algorithm by adapting to the specific requirements of the selected task.

The main differences are:

Dynamic selection and correction of agent actions. An agent selects a sequence of LO services taking into account specific QoS parameters. Classical Q-learning is mainly focused on a static environment, where the set of actions is predetermined, and in our case, agent actions depend on the current set of achieved states and the combination of inputs and outputs of each LO service.

QoS analysis. This method includes an assessment of the quality of services based on QoS to form a reward that provides more accurate assessment of the benefit from each service selection. The number of QoS parameters is not fixed. In addition, unlike common examples with QoS optimization, we introduce global weights and global modality of QoS parameters to take into account the relative importance of different aspects (for example, LO duration or rating, parameter modality) in process of calculating the reward for using a particular LO service. Such generalization allows to adapt the model to specific task conditions, increases its flexibility and accuracy in choosing the optimal composite service (PLP) while classical Q-learning focuses only on the final reward and does not take into account QoS that is critical for LO services.

Loop prevention mechanisms. The complexity of building multi-component PLPs causes the need in restrictions on the number of steps without progress to avoid loop route. These restrictions expand the capabilities of classical Q-learning, where such loops are not so critical.

Handling negative and unpromising states. The proposed method has a mechanism that prevents actions that have not transitions for current state, and penalty is used for such actions: agent checks for possible actions in each state and receives a penalty for unpromising actions. This extension is useful for handling exceptions or unpromising states, but it is not supported by classic Q-learning method.

Defining learning parameters according to the input data size. This extension provides the possibility for automatic determining: the number of episodes (it depends on the state table and the complexity of the task); the limit of steps per episode (it can be based on the number of states and the average length of the possible path); the threshold number of steps without progress (it avoids looping depending on the complexity of the environment); Q-learning hyperparameters: the epsilon learning strategy and the epsilon parameter decrease method (it ensures the optimal balance between exploration and exploitation); the alpha learning rate (it adapts the maximum path length and learning stability).

Such modification of Q-learning allows the agent not only to find possible paths through the list of available LOs, but also to improve adaptively the selection strategy based on feedback from the environment. Thereby, the system has an ability to generate automatically PLP with optimal functionality and values of QoS parameters. This approach is especially useful in dynamic environments, where the characteristics of LO services (duration, price, rating, etc.) can be changed. Use of Q-learning allows to select optimal composition options, taking into account not only the available services, but also their current performance. The main components of the proposed method remain within the framework of classical Q-learning (update formula, ε-greedy strategy, etc.), but the improvements made make its implementation more flexible and robust, that makes it more effective in environments with high dynamics and complex performance requirements.

9. Conclusion and prospects

The proposed AI planning algorithm enables the creation of LO service compositions based on Q-learning and optimizes the selection of optimal service execution path based on QoS criteria. We test this approach for PLP design task to automate selection of relevant LO sequences and their optimizing according to the individual needs of students.

The approach proposed in this research incorporates the use of global weights for QoS parameters, allowing precise prioritization of various aspects of LO service quality and optimizing PLPs based on these priorities. By leveraging additional characteristics for global weights and calculation methods for final values, the approach effectively formulates the problem to account for both the maximization and minimization of different QoS parameters. Such solution ensures a more precise and efficient decision-making process.

The use of different aggregation methods for each QoS parameter allows fine-tuned calculations of QoS values for the chosen composite service. For instance, upon completion of the algorithm, the output PLP is generated with account of QoS values of LOs such as total duration, overall cost, and the average rating of learning. The computation method is determined by the semantics of each characteristic.

Agent training utilizes the ε-greedy strategy to balance exploration and exploitation. Optimizations to avoid dead-end states include checks and exclusions of excessive or unnecessary actions. Training concludes once the target state is achieved or the predefined number of iterations is completed.

Testing process, particularly with large-scale tasks, shows a number of problems and limitations that become evident in large environments and complicate algorithm scalability. As the task complexity increases (e.g., it has more states, actions, and factors), the algorithm becomes harder to scale and requires substantial computational resources.

Q-learning and its modifications use Q-table to store values for each "states-action" pair (our modified method uses also a transition table). The number of states and actions in large or continuous environments grows exponentially and causes difficulties in processing and management of these tables. Another problem is the low generalization of similar states in Q-learning, which significantly slows down the learning process: states with common features are analyzed independently and require a large number of additional calculations. Q-learning is suitable only for discrete states and actions, but many real-world tasks involve continuous state and action spaces.

Other difficulties arise in selecting algorithm parameters. Q-learning efficiency heavily depends on ε-greedy strategy (ε defines the probability of random action selection) and requires additional efforts for parameter tuning. In complex environments, this algorithm can be unstable due to fluctuations in Q-values, especially with stochastic rewards or transitions. Furthermore, Q-learning lacks a memory mechanism (the agent updates values only for the current state and action), leading to the loss of information from previous explorations.

These limitations are significant for large-scale tasks that require high scalability and generalization. Therefore, we plan to explore other ML methods, such as Deep Q-Networks (DQN), and analyze their potential for more efficient handling of these challenges. The aim of this research is to improve efficiency by enhancing the agent's learning speed, increasing stability, addressing the limitations of the Q-learning method in complex environments with a large number of states or continuous data, and making the algorithm more scalable and effective.

Declaration on Generative AI

During the preparation of this work, the authors used elements of Chat GPT-4 for grammar and spelling check. After using these services, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] V. Dagiene, D. Gudoniene, R. Bartkute, The integrated environment for learning objects design and storing in Semantic Web. In: International journal of computers communications & control, 13(1), (2018): pp. 39–49.

[2] M. Doorten, B. Giesbers, J. Janssen, J. Daniels, R. Koper, Transforming existing content into reusable learning objects. In: Online education using learning objects, (2012): pp. 116–127.

[3] J. Rogushina, A. Gladun, O. Anishchenko, S. Pryima, Semantic Support of Personal Learning Trajectory Development, CEUR 3806, (2024): 487-505. https://ceur-ws.org/Vol-3806/S_19_Rogushina_Gladun_Anishchenko_Pryima.pdf.

[4] A. Basuki, Personalized learning path of a web-based learning system. In: International Journal of Computer Applications, 53(7), (2012): pp.17-22.

[5] I. Grishanova, J, Rogushyna The technology of mashine learning for a composite web service development. Problems in Programming, V.4, (2024): 3-13. (in Ukrainian). pp.isofts.kiev.ua/index.php/ojs1/article/view/669/721.

[6] J. Clifton, E. Laber, Q-learning: Theory and applications. Annual Review of Statistics and Its Application, 7(1), (2020): 279-301. http://dx.doi.org/10.1146/annurev-statistics-031219-041220.

[7] J. Li, X. L. Zheng, S. T. Chen, W. W. Song, D. R. Chen, An efficient and reliable approach for quality-of-service-aware service composition. In: Information Sciences, 269, (2014): 238-254.

[8] A. L. Samuel, Some studies in machine learning using the game of checkers. In: IBM Journal of research and development, 3(3), (1959): pp.210-229. http://dx.doi.org/10.1147/rd.33.0210.

[9] M. I. Jordan, T. M. Mitchell, Machine learning: Trends, perspectives, and prospects. In: Science, 349(6245), (2015): 255-260. http://dx.doi.org/10.1126/science.aaa8415.

[10] A.G. Barto, R. S. Sutton, Reinforcement Learning: An Introduction, The MIT Press: Cambridge, Massauchsetts, 2018. http://dx.doi.org/10.1109/tnn.1998.712192

[11] J B. Jang, M. Kim, G. Harerimana, J.W. Kim, Q-learning algorithms: A comprehensive classification and applications, IEEE Access, 7, (2019): 133653-133667. http://dx.doi.org/10.1109/access.2019.2941229.

[12] V. Uc-Cetina, F. Moo-Mena, R. Hernandez-Ucan, Composition of web services using Markov decision processes and dynamic programming. In: The Scientific World Journal, (1), , (2015): 545308. http://dx.doi.org/10.1155/2015/545308.

[13] H. Wang, X. Zhou, X. Zhou, W. Liu, W. Li, A. Bouguettaya, A. Adaptive service composition based on reinforcement learning. In: Service-Oriented Computing: 8th International Conference, ICSOC 2010, San Francisco, USA, 2010. Proceedings 8, pp. 92-107. Springer Berlin Heidelberg.