A GRAPH-BASED APPROACH FOR PERSONALIZED LEARNING CURRICULUM

Tran To Que Phuong¹², Tran Thanh Tung¹²

¹International University, Ho Chi Minh City, Vietnam
²Vietnam National University, Ho Chi Minh City, Vietnam
queephuong99@gmail.com, ttung@hcmiu.edu.vn

ABSTRACT: Nowadays, personalized learning becomes more attractive and worth the investment to educators. There are many studies in this field and many solutions have been proposed. All these solutions focus on designing an optimal personalized curriculum from predefined strategies. In this paper, we proposed a method based on graph theory to design a platform for a personalized curriculum that supports many different strategies but not violating the compulsory requirements. Thanks to principles and logical conditions, this method maintains the accuracy of the results. First, an abstract graph is used to represent constraints and information on the educational program. Then, an algorithm is designed to traverse the graph to find a personalized curriculum with different strategies. Finally, this paper shows a proof of correctness of the algorithm, and several case studies on a real curriculum.

Keywords: Graph theory, personalized learning, learning path, recommendation system.

I. INTRODUCTION

Personalized learning is a proven technique to increase the learning results of learners in the last decade. Learners are encouraged to explore alternate navigational pathways using subject knowledge and resources from across the world in a web-based educational system. The structure of the provided domain and material, on the other hand, are frequently presented in the same way, without regard for the learners' browsing objectives, experience, or prior knowledge. Therefore, personalized learning becomes emergency research because of the increasing need of designing personalized learning plans in learners that they want to have a study plan that satisfies their significant backgrounds and preferences.

Because no one learning component is acceptable for all students, curriculum sequencing is an important study field in the learning process, especially in universities where students can flexibly design their learning pathway within the allowable framework. Curricular sequencing is another tool for students to manage their learning paths and accomplish curriculum objectives, it tries to provide an appropriate learning pathway for each student based on unique previous knowledge, preferences, and learning objectives.

Most published research on personalized curriculum systems focuses on finding the most optimal learning pathway based on a predefined criterion by the research team. And in this research paper, we want to build a platform for personalized learning path design that supports a variety of personalization criteria. The description of our system has two main parts: an abstract graph to represent constraints and information of the training program, an algorithm that runs on the graph and has different extensions to support various personalized strategies to find individual paths satisfied significant preferences, divergent backgrounds, and unique learning situation of each student.

The remainder of this research paper is structured as follows: Section II describes several related works. The main methodology throughout this research and demonstration would be included in Section III. The implementation of the research study and results are discussed in Section IV. Finally, Section V offers the paper's conclusion with directions for the future.

II. RELATED WORK

There have been many research papers on designing systems for a personalized learning path. Most of the published methods focus on finding the optimal learning path for each student according to certain predefined criteria. As in the article by [1] or [2] on this issue, the published solution used hybrid filtering such as content-based filtering, collaborative filtering to analyze, requirements, interests, information of students as well as the relationship between courses and study programs to find the most optimal learning path according to the query or the answers in the pretest that students provide to the system. In addition, in the research paper provided by Saarland University [3], the data-driven method is used to find a learning path that helps students graduate as soon as possible and achieve the best results. Especially, the proposed system also maintains that information about students and the educational program would be updated to the latest version at the beginning of each semester. With the same idea that not only optimizing completion time and cumulative grade average for students but also handling contextual data, [4] used machine learning techniques such as long short-term memory (LSTM), recurrent neural network (RNN) to analyze the semantics of information. The system updates data using sliding windows of student results. In another approach, the system can find a suitable learning path for each student based on exploiting students' needs in terms of subject difficulty level or their expectations about learning progress. To solve this problem, besides building a pretest kit to collect and analyze students' needs, [5] focus on find relationships between courses of interest to students and their
academic performance, then finally use a genetic algorithm to find the most appropriate learning path for each student. In addition, the system can find many different routes for the same student, and these studies have reduced each route to vector form and used several comparison techniques to find the most suitable path. Therefore, in general, the published methods focus on finding the most optimal route for each student based on certain criteria.

III. METHODOLOGY

A. Overview

In this paper, the proposed method would apply for credit programs in universities, educational programs in International University (IU) would be used as a good example. Subjects in the training program are divided into two types: compulsory subjects and elective subjects. On the one hand, students must complete all compulsory subjects. On the other hand, elective subjects can be divided into several subject groups, each group will have a corresponding list of subjects and required number of credits, students must complete the required minimum number of elective groups and there is no need to complete all the courses in the elective group once the required number of credits has been reached. A prerequisite means a list of specified subjects that students must complete before the semester of registration for the target subject. A co-requisite means the list of subjects specified that student must complete before the semester to register for the target subject or must register at the same time as the target subject. Some courses require students to complete a sufficient number of credits accumulated from the beginning of the education program before enrolling in the target course. A valid semester requires a minimum and a maximum number of credits. And students must learn a subject until they reach an average score of not less than 50 to pass the subject. In case failed the subject with having a prerequisite or co-requisite requirement for another subject, students are still allowed to register for the next subject before relearning until they pass the subject.

About the system, graph theory would be considered as the main method to build up the system for personalized curriculum in schools. As a result, this research could be considered like a recommender system, and the relationship between subjects could easily display by nodes and vertices so that graph theory would help the recommend process become more logical and understandable. With strict relationships between nodes and easily inserting logical conditions onto vertices, the rigid curriculum could be maintained. First, information about courses which are called input course data would be inputted into the system. For the assumption that input course data has already been preprocessed, the system would use provided data to define the structure of the abstract graph. Then an abstract graph is generated based on a defined structure to represent realistic constraints and components in the education program. Besides definition, each component, constraint, or relationship in abstract graph has disparate requisites to describe the contexts in different cases related to the learning path design process, which called semantics. By mapping semantics with a defined abstract graph, an exploration graph is generated as a result which contains all cases of the learning paths that satisfies realistic requirements of education programs in schools. Then more customized data which is called user information and personalized strategies could be additionally input to get the right learning path for each user from the exploration graph. User information can be accumulated scores, interests, completed the subject list... which supports the system to better understand the learning situation as well as the user’s expectations during the learning process. Personalized strategies are methods that are applied to the exploration graph to find the right learning path for each student. Target curriculum that considered as final output is called learning path.

B. Abstract Graph

An abstract graph represents the structure and constraints of an academic program. An abstract graph \( G_a \) is a direct graph with a set of nodes \( V_a \) is a union of a set of semester nodes and a set of subject nodes, \( V_a = V_{Sem} \cup V_{Subj} \) a set of edges is a union of a set of edges among semester nodes and subject nodes \( E_a = E_{SemSem} \cup E_{SemSubj} \cup E_{SubjSem} \) and a set of groups of subjects \( C_G \subseteq V_{Subj} \).

\[
G_a = (V_a, E_a, C_G) \\
= (V_{Sem}, V_{Subj}, E_{SemSem}, E_{SemSubj}, E_{SubjSem}, C_G)
\]
NODE SEMESTER (VSem): VSem is a set of semester nodes that represents semesters in the annual program. A vsem ∈ VSem consists of a set of functions including maxCred:VSem → ℕ to describe a maximum number of credits in the semester, and minCred:VSem → ℕ to describe a minimum credits allowed in the semester.

Example: A school year in International University requires two semesters called odd semester and even semester. In each semester, students must register at least 14 credits and the total registered credits must not be greater than 24 credits. Then in abstract graph, VSem = {Odd, Even}, semName(Odd) = "Odd", maxCred(Odd) = 24, minCred(Odd) = 14, semName(Even) = "Even", maxCred(Even) = 24, minCred(Even) = 14.

NODE SUBJECT (VSubj): Let VSubj be a set of subject nodes that represents subjects in a program. Each vsubj ∈ VSubj consists of a set of functions including cred:VSubj → ℕ to describe the number of credits of the subject, preReq:VSubj → 2VSubj to describe the list of prerequisite subjects vsubj which are subjects that must be completed before taking vsubj, coReq:VSubj → 2VSubj to describe the list of corequisite subjects of vsubj which are subjects that must be completed before or taken in the same semester with vsubj, and minCredReq:VSubj → ℕ to describe the requirement of minimum accumulated completed credits before taking the subject. In the case that a subject does not have any requirement of prerequisite, corequisite or minimum accumulated completed credits then the values of these corresponding functions would return empty set for preReq and coReq, and return 0 for minCredReq.

Example: In Computer Science department, Calculus 2 and Calculus 3 are the two subjects that belong to list of subjects required by syllabus. Credits of Calculus 2 and Calculus 3 both equal to 4, and student must complete Calculus 1 before the semester of registering Calculus 2, must complete Calculus 2 before the semester of registering Calculus 3, there are no requirement of accumulated credits. Then in abstract graph, VSubj = {Calculus2, Calculus3,...}, subjName(Calculus2) = "Calculus 2", cred(Calculus2) = 4, preReq(Calculus2) = Calculus1, coReq(Calculus2) = ∅ \in Calculus2, coReq(Calculus3) = ∅ \in Calculus3, minCredReq(Calculus2) = 0, subjName(Calculus3) = "Calculus 3'", cred(Calculus3) = 4, preReq(Calculus3) = Calculus2, coReq(Calculus3) = ∅ \in Calculus3, minCredReq(Calculus3) = 0.

COURSE GROUP (CG): Let CG be a set of subject groups such as a set of elective courses in a program. A cg ∈ CG consists of a set of functions including cgCredReq:CG → ℕ to describe the required number of credit students must take from the course group, cgSubjList:CG → 2VSubj to describe the list of subject nodes that belong to the subject group.

Example: In Computer Science department, Physics 1 (2 credits) and Physics 2 (2 credits) are required as compulsory courses, with total credits of compulsory courses are 137 credits, Calculus 3 (4 credits) and DifferentialEquations (4 credits) are required as elective courses which means student only have to learn one of the two subjects to complete Elective Group 1 (required 4 credits). Then in abstract graph, CG = \{group_0, group_1\}, cgCredReq(group_0) = 137, for vsubj ∈ VSubj, vsubj be compulsory courses, cgSubjList(group_0) = \{Physics1, Physics2\}, cgCredReq(group_1) = 4, cgSubjList(group_1) = \{Calculus3, DifferentialEquations\}.

EDGE SEMESTER-SEMESTER (ESemSem): Let ESemSem be a set of edge link semester to semester, ESemSem = (VSem × VSem × ℕ × ℕ). Each eSemsem ∈ ESemSem, eSemsem = (semStart, semDest, maxCredAvail, minCredAvail) represents a relationship between two semesters a starting semester, semStart, and a destination semester, semDest, where maxCredAvail and minCredAvail are the maximum and minimum number of credits the starting semester must complete before moving to the destination semester.

Example: International University has two semesters called odd semester and even semester, know that odd semester always be the initial semester in a school year, therefore exists edge link from odd semester to even semester. In each semester, students must register at least 14 credits and the total registered credits must not be greater than 24 credits. Then in abstract graph, ESemSem = \{even\even\}, semStart = vOdd, semDest = vEven, maxCredAvail = maxCred(vOdd) = 24, minCredAvail = minCred(vOdd) = 14.

EDGE SEMESTER-SUBJECT (ESemSubj): Let ESemSubj be a set of edges link semesters to subjects ESemSubj = (VSem×VSubj). Each eSemsubj ∈ ESemSubj = (sem, subj) ∈ ESemSubj represents that the subject subj is opened in the semester sem of the program.

Example: We have subject Calculus 3 would be opened annually in odd semester. Then in abstract graph, ESemSubj = \{OddCalculus3\}, semStart = vOdd, subjDest = vCalculus3

EDGE SUBJECT-SEMESTER (ESubjSem): Let ESubjSem be a set of edge links subjects to semesters, an edge eSubjsem ∈ ESubjSem exists if and only if there exists an edge eSemsubj = (sem, subj) ∈ ESemSubj.

Example: We have subject Calculus 3 would be opened annually in odd semester. Because there exists link from odd semester to subject Calculus3, therefore also exists link from subject Calculus 3 to odd semester. Then in abstract graph, ESubjSem = \{Calculus3Odd\}, subjStart = vCalculus3, semDest = vOdd.

Give an example for the abstract graph as shown in Figure 2. Assume that the program has 2 academic semesters “Odd”, “Even” every school year with 2 different course groups and 4 subjects “Calculus2”, “DiscreteMath”, “Calculus3”, “DifferentialEquations”. Each subject belongs to a course group and is included in the
subject list of that course group, and each course group requires the number of credits to complete. Furthermore, on the edge between two semesters also requires a semester gain from 10 to 24 credits before changing to the next semester.

C. Exploration Graph

An exploration graph of an abstract graph \( G_a(V_a, E_a, CG) \) is an transition system \( G_x \) that represents the semantic of the abstract graph. An exploration graph is defined as

\[
G_x = (C, I, E_x)
\]

where

- **CONFIGURATION NODE (C):** Let \( C \) be a node-set of configuration nodes in \( G_x \), \( C = (V \times V_{sem} \times 2^{V_{Subj}} \times 2^{V_{Subj}}) \).

For each \( c \in C \), \( c = (visitNode, currSem, selSubjList, complSubjList) \), visitNode is a node in the abstract graph, currSem is the last visited semester, selSubjList=\( 2^{V_{Subj}} \) is the list of subjects that have been selected in currSem, complSubjList=\( 2^{V_{Subj}} \) is the list of subjects that have been completed before currSem, and remSubjList=\( V_{Subj} \)\( \setminus \) complSubjList is the list of subjects that are uncompleted in currSem at the configuration node.

A set of functions also be defined to check the validity of constraints between configuration nodes, including:

- \( tcc: C \rightarrow \mathbb{N} \) is a function returns the total chosen credits in selSubjList of currSem at configuration node \( c \),
- \( accCred: C \rightarrow \mathbb{N} \) is a function returns the accumulated credits of complSubjList at configuration node \( c \), and
- \( availCredCG: CG \rightarrow \mathbb{N} \) is a function returns the remaining number of uncomplete credits in the course group which can be calculated from the subject list of education program subtracts complSubjList. Furthermore, the value of function availCredCG must not be less than 0, which means students are required to learn exactly the number of required credits in the course group.

- **INITIAL CONFIGURATION (I):** The initial configuration \( I \) of an exploration graph represents the initial state of a student such as his current semester, and his list of completed subjects. For a new student, the initial configuration is (semOdd, semOdd, \( \emptyset \), \( \emptyset \)).

- **EDGE (E_x):** Let \( E_x \) be a set of edge between configuration nodes in an exploration graph. For two configuration nodes \( c_i, c_j \in C \),

\[
\begin{align*}
& c_i = (n_i, s_i, s_{li}, c_{li}) \\
& c_j = (n_j, s_j, s_{lj}, c_{lj})
\end{align*}
\]

There is an edge \( e_{ij} \in E_x \) between \( c_i \) and \( c_j \) if and only if there is an abstract edge \( e_{ij} \) between \( n_i \) and \( n_j \) in the abstract graph and \( c_i \) and \( c_j \) satisfied the constraints in one of the following cases

- **A transition between two semesters:**

In this case the edge \( e_{ij} \in E_{SemSem} \) in the abstract graph represents requirements for a transition from one semester to another semester including the number of accumulated credits of the current semester before moving to the next semester.
Formally, there is an edge $c_i(n_i, s_i, s_{li}, c_{li}) \rightarrow c_j(n_j, s_j, s_{lj}, c_{lj})$ if there exists an edge $e_{\text{semsem}}(s_i, s_j, \text{maxCred}_{ij}, \text{minCred}_{ij}) \in E_{\text{semsem}}$ such as $n_i = s_i$, $n_j = s_j$, $tcc(c_i) \leq \text{maxCred}_{ij}$, $tcc(c_j) \geq \text{minCred}_{ij}$, $s_{lj} = \emptyset$, and $c_{lj} = c_{li} \cup s_{li}$.

In words, there is a transition from the configuration $c_i$ to $c_j$ using the edge $e_{\text{semsem}}$ only when the configuration $c_i$ is at the semester $s_i$ and the number of credits selected in the semester is in the range $\text{minCred}_{ij}$ and $\text{maxCred}_{ij}$. And after the transition, the resulting configuration $c_j$ resets the list of selected subjects $s_{lj}$ to empty set and add the list of selected subjects in the previous semester to the list of completed subjects $c_{lj}$.

b. A transition from a semester to a subject

In this case the edge $e_s \in E_{\text{SemSubj}}$ in the abstract graph represents requirements for taking a subject in a semester including the prerequisites, corequisite subjects of the target subject, the required accumulated credits of the target subject, and the required credits of the course group that the target subject belongs to.

Formally, there is an edge $c_i(n_i, s_i, s_{li}, c_{li}) \rightarrow c_j(n_j, s_j, s_{lj}, c_{lj})$ if there exists an edge $e_{\text{SemSubj}}(s_i, s_j, \text{maxCred}_{ij}, \text{minCred}_{ij}) \in E_{\text{SemSubj}}$, $cg \in CG$, sub $\in cg$ such as $n_i = s_i$, $n_j = s_j$, $s_j \in \text{remSubjList}(c_i)$, $\text{preReq}(sub_j) \subseteq \text{complSubjList}(c_i)$, $\text{coReq}(sub_j) \subseteq cl_j \cup s_{lj}$, $\text{minCredReq}(sub_j) \leq \text{accCred}(c_i)$, $\text{availCredCG}(cg) > 0$, $s_j = s_i$, $c_{lj} = cl_j$, and $s_{lj} = s_{li} \cup sub_j$.

In words, there is a transition from the configuration $c_i$ to $c_j$ using the edge $e_{\text{SemSubj}}$ from the semester $s_i$ to the subject $sub_j$ only when the configuration $c_i$ is at the semester $s_i$, the subject $sub_j$ is not in the list of remaining courses of the configuration, the list of prerequisite subjects of the subject is in the completed list of the configuration, the list of corequisite subject of $sub_j$ have been completed or been chosen in the current semester of the configuration, the accumulated credits at the configuration is greater than the required accumulated credits of the subject. Furthermore, the course group that the subject belongs to have not completed. And after the transition, the resulting configuration add the subject $sub_j$ to the list of selected courses and keep the current semester and the list of completed courses unchanged.

c. A transition from a subject to a semester

In this case the edge $e_s \in E_{\text{SubjSem}}$ in the abstract graph represents the completion of a subject.

Formally, there is an edge $c_i(n_i, s_i, s_{li}, c_{li}) \rightarrow c_j(n_j, s_j, s_{lj}, c_{lj})$ if there exists an edge $e_{\text{SubSem}}(s_i, s_j, s_i, s_j, s_{li} = s_{li}, c_{lj} = cl_j \cup sub_j$.

In words, the transition from a subject to a semester make the resulting configuration add the subject to the list of completed subjects and keep the other states unchanged.

![Figure 3. Part of an exploration graph](image-url)
PATH: A path in the exploration graph is begun at the configuration node of first semester until it can no longer extend, and the end node of the path would be the configuration at semester that no valid subject has been found or the required number of credits and subjects have been reached the graduation requirement. Initial node of a path would have visitNode and currSem at the first semester that has been considered, an empty selSubjList, and the value of compSubjList depends on which subjects does student have completed at the time using the system, if not any subject, then compSubjList would be empty.

LEARNING PATH: A learning path is a path in an exploration graph that starts from the configuration of the first semester and ends at the configuration that required the number of credits and subjects have been reached graduation requirement. As the result of the proposed system, exactly one learning path would display to recommend for the target students their referring pathway. The concept of recommended learning path is the same as the exploration graph because as proved above, the exploration graph contains all learning paths.

D. Soundness [6] of the exploration graph

Theorem: For every learning path \( LP \in G_x \), there exists a learning path \( LP' \) in reality.

Proof. The proof is by induction on the step length of the learning path

Initial step
- Initial step length of \( LP \): \( \text{len}(LP) = 1 \)
- Initial node: \( n_1 = v_{\text{sem} semId = 1} \)

Inductive step
Assume for every learning path with \( \text{len}(LP) = n \), the path from the root node to the current node always correctly reflects all conditions specified by a valid learning path. Consider a learning path \( LP' \) of step length \( (n+1) \) in the exploration graph:

\[ LP'' = LP + 1 \rightarrow \text{len}(LP'') = n+1 \]

By definition of configuration node in exploration graph (session \( C_n \)), then there must exist \( e_{\text{enl}} \in E_x \) with \( e_{\text{enl}} \) is the edge between configuration node \( c_n \) at \( LP \) and configuration node \( c_{n+1} \) at \( LP'' \). By definition of edge in exploration graph, if \( e_{\text{enl}} \in E_x \) exists then \( e_{\text{enl}} \) satisfied all defined conditions in relationship types between \( c_n \) and \( c_{n+1} \). These defined conditions correspond to realistic requirements. From inductive assumption, the path from the root node to the current node in \( LP(n) \) always correctly reflects all conditions specified by a valid learning path, and \( e_{\text{enl}} \) exists between configuration node at \( LP \) and configuration node at \( LP'' \) which means \( LP'' \) also satisfies all realistic requirements of the valid learning path. For every learning path, \( LP \in G_x \) exists learning path \( LP' \) in reality.

From the above theorem, we officially state that finding personalized curriculum is equivalent to find a learning path in exploration graph from given input and customized techniques. In addition, as the output is a learning path, our method can handle different situations of students. Every time using the system, students would update their information such as resume, the learning process, grading in education and choose personalized strategies that they expected their learning path would be, then the system would process these inputs to find the best fit learning path for each student.

IV. IMPLEMENTATION

At the extremely basic, our exploration graph can generate all paths of the curriculum which satisfy the requirements. To find the curriculum in need, the method must find an approach to choose from exploration graph the necessary learning path for each specific case of a student. With the variety of courses and constraints that has been predefined, the number of paths that could be generated from the exploration graph is enormous which has become the basis for the diversity of designing learning plans. And based on the predefined theory of the method, the easiest way is to completely generate an exploration graph and then choose the learning path that best suits the students. However, this method can be very laborious in many cases. Instead, we can use methods that do not need to generate the entire exploration graph like priority queue, or optimization algorithms that use weight functions. From the algorithm 1 in Figure 3, we can find a personalized learning path without totally generating an exploration graph. In the beginning, an initial configuration node has been created based on the input of the student, then at each configuration node, we would find all neighbor nodes can be reached from the current state without duplicating. Then exploration edges from the current node to each neighbor would be created. By default, at step 14, we would randomly choose an exploration edge to expand from the current configuration node until reaching the final state. But furthermore, thanks to the convinced theorem, this step could be substituted by any other methods to find the next step that best fits each student's criteria without affecting the accuracy of our method. In this paper, there are some personalized strategies that have been proposed based on simulated situations.
V. RESULT AND DISCUSSION

Input course data, input user information for all experiments below are the same. With input, course data is followed education program of IT program in IU. Input user information provided that the target student had completely learned the subject Physics 1.

1. Case 1 - Random

In the random approach, subjects are randomly selected but satisfy prerequisite constraints in IU program, subjects with yellow highlighted in Figure 4 are prerequisite subjects. Because the student’s input shown that Physics 1 had been completed then the initial configuration node has one subject in completed list and no exist in the output curriculum. The result in this case shown that our method can deal with different situations of student and again convince for its reliability.

2. Case 2 - Prioritizing prerequisite subjects

In this case, the student wants to complete all subjects in prerequisite first, and priority queue would be used as solution combining to weight function based on prerequisite orders. As the result, subjects that exist in prerequisite lists are prioritized to select first.
The target student wants to complete all subjects in prerequisite first. This case would be solved by priority queue. In credit training program, one subject could prerequisite for another subject in prerequisite that we can generate sequence of subjects based on prerequisite order. Then, the first subject in that sequence would be the most prioritized subject due to its impact on the remaining subjects in the series. And the weight function would handle this to maintain the more important the subject, in this case is the prerequisite importance, the smaller its weight. So, priority queue in this case study would prioritize subject with smallest weight to the higher.

3. Case 3 - Learning by difficulty level

Assume that the student wants to learn by difficulty level from low to high. The definition of difficulty level would be based on average score of each subject during the latest 8 semesters. From the range of average scores, subjects would be divided into three groups of difficulty level: easy, medium, hard. In addition, the prerequisite importance would be considered to take advantage of the effect of shortening the learning time. We also use priority queue to solve combining to weight function with higher priority for prerequisite and the lower for difficulty level. As the result, subjects which have high average score and exists in the prerequisite list is recommended to learn first, and the prerequisite has higher priority, therefore in sometimes subjects in hard group with high prerequisite order would prefer to learn before subjects in easy group but no prerequisite. And this strategy sometimes helps shorten the learning path.

The target student wants to learn by difficulty level from low to high, which means complete simple subjects before ones with higher level. This case also be solved by priority queue and required average score of each subject from the initial input course data. Average score of each subject would be calculated by the average of GPA of all students have learnt the subject within latest semester. From the range of average score, subjects would be divided into three groups of difficulty level: easy, medium, hard. In addition, the prerequisite importance would be considered to take advantage of the effect of shortening the learning time. Besides the prerequisite importance, the weight function would set weight for difficulty level with smaller weight for easy level and higher weight for hard level. So, priority queue in this case also prioritize subject with smallest weight to the higher.

4. Case 4 - The shortest learning process

In this case, the student wants to graduate as soon as possible which means the smaller number of semesters the better. This case would be solved by Bellman-Ford algorithm which is the classic approach to find on-the-fly results in shortest-path problem. Although as the result, we can trace to get the optimized curriculum from on-the-fly results, this approach required us fully expand the exploration graph which will overload the system because this graph is too large. Therefore, in future work, this case study should be solved by a heuristic approach such as the A* algorithm rather than the Bellman-Ford algorithm.

The target student wants to graduate as soon as possible, in this case the minimum number of semesters considered as the criterion for optimization. This case would be solved by two different approaches that are Bellman-Ford algorithm. Goal of this case study is to find the learning path with lowest number of semesters. Bellman-Ford algorithm would be applied to try to find shortest path in exploration graph without generating full graph. The algorithm would use the result of weight function on each edge in exploration graph as defined above.

5. Discussion

Firstly, throughout the results, our method can deal with all constraints in IT programs and different personalized case studies. Secondly, the system of finding a personalized curriculum depends on input so it can easily deal with different situations of students. In addition, we figured out that the proposed platform could be used to check validity from a given curriculum in future work. And finally, the experiment on Bellman-Ford shown that we should use other approaches such as heuristic or approximation algorithms to deal with optimization case studies.

VI. CONCLUSION

From the view of applying technologies in personalized learning and the main target is building a platform that could support many different strategies, this research has proposed an approach based on graph theory effectively support in a university environment. The proposed method has been proved to be reasonable and successfully initially implemented. Soon, with the belief that there not exist strategies for everyone, but each person has their strategies for effective learning, this research aims to complete the platform for people from everywhere to apply their strategies and get the best learning path from this system.

From what has been done in this paper, let discuss which directions should be included in future work. On the one hand, because this method is just at the very first stage, we can improve it by designing an interactive system with a friendly interface for end-users. In addition, we can automate the updating process if the system be able to deal with real-time data. On the other hand, from the correctness demonstration of proposed method, we can use it to check for validity of a given curriculum.
REFERENCES


CÁ NHÂN HÓA CHƯƠNG TRÌNH HỌC DỰA TRÊN LÝ THUYẾT ĐỒ THỊ
Trần Tô Quế Phương, Trần Thanh Tùng

TÓM TÁT: Ngày nay, việc cá nhân hóa chương trình học đã và đang nhận được sự quan tâm, đầu tư rất tích cực từ các nhà giáo dục. Có rất nhiều nghiên cứu trong lĩnh vực này và nhiều giải pháp đã được đề xuất. Hầu hết các giải pháp này đều tập trung vào việc thiết kế một chương trình giảng dạy được cá nhân hóa tối ưu từ các chiến lược được định nghĩa trước. Trong bài báo này, chúng tôi đề xuất một phương pháp dựa trên lý thuyết đồ thị để thiết kế một chương trình giảng dạy cá nhân hóa hỗ trợ linh hoạt nhiều chiến lược khác nhau nhưng không vi phạm các yêu cầu bắt buộc của chương trình học. Ngoài các nguyên tắc và điều kiện chặt chẽ, phương pháp này dựa tri dối chính xác của kết quả. Đầu tiên, một đồ thị trừu tượng được sử dụng để biểu diễn các ràng buộc và thông tin về chương trình giảng dạy. Sau đó, một thuật toán được thiết kế để duyệt qua đồ thị nhằm tìm ra chương trình giảng dạy phù hợp nhất với nhu cầu và đặc điểm của từng cá nhân, theo các chiến lược khác nhau. Cuối cùng, bài báo này cho thấy một bằng chứng về tính đúng đắn của thuật toán và một số nghiên cứu điển hình dựa trên một chương trình giảng dạy thực tế.