A Binary Integer Programming Model for Global Optimization of Learning Path Discovery

Nabil Belacel
National Research Council Canada
100, des Aboiteaux St.
Moncton, E1A 7R1,Canada
+1.506.861.0963
nabil.belacel@NRC.gc.ca

Guillaume Durand
National Research Council Canada
100, des Aboiteaux St.
Moncton, E1A 7R1,Canada
+1.506.861.0961
guillaume.durand@NRC.gc.ca

François Laplante
Université de Moncton
60, Notre-Dame-du-Sacré-Cœur St.
Moncton, E1A 3E9,Canada
+1.506.381.6220
francois.laplante@umoncton.ca

ABSTRACT

This paper introduces a method based on graph theory and operations research techniques to optimize learning path discovery. In this method, learning objects are considered as nodes and competencies as vertices of a learning graph. A first step consists in reducing the solution space by obtaining an induced subgraph H. In a second step, the search of an optimal learning path in H is considered as a binary integer programming problem which we propose to solve using an exact method based on the well-known branch-and-bound algorithm. The method detailed in the paper takes into account the prerequisite and gained competencies as constraints of the optimization problem by minimizing the total competencies needed to reach the learning objective.

Keywords

Learning path, learning object recommendation, graph theory, clique, mathematical programming, binary integer programming, branch-and-bound algorithm.

1. INTRODUCTION

Global Positioning System (GPS) is a Global Navigation Satellite System (GNSS) that is massively used by car drivers. This large acceptance is easily understandable by the benefits that such a system can offer. Car navigation systems can dynamically calculate an itinerary between two points taking into account, depending on the system, several constraints like duration, distance, closed roads, traffic jams, etc....Drivers can focus exclusively on their driving limiting risks of accidents, stress, and losing their way.

To some extent, the learning path followed by a student could be seen as an itinerary between several learning objects [9]. In this context, constraints on learning objects are not distance or time duration to go from one learning object to the other but rather prerequisite and gained competencies. As a result the itinerary or path between learning objects is regulated by competency dependencies that lead a learner from an initial to a targeted competency state. For example, a learner with solid grounds in integer arithmetic (starting location) willing to learn the solving of systems with multiple variables (destination) should be advised to previously learn to solve one variable linear equations (next step of the itinerary).

Over the years, educational data mining and recommendation technologies have proposed significant contributions to provide learners with adequate learning material by recommending educational papers [18] or internet links [10], using collaborative and/or content-based filtering. These approaches usually aim at recommending learning material satisfying an immediate interest rather than fitting in the learner's sequential learning process.

Sequential pattern [28] and process mining [19] technologies have also been investigated. However, these technologies have been used to understand the learner's interaction with content to discover general patterns and trends rather than to recommend adapted learning paths to learners.

Other approaches, in the course generation research community, address the need for recommending not only the learning objects themselves, but sequences of learning objects. Sicilia et al. [17] or Ulrich and Melis [20] addressed learning design concepts and requirements through Course Generation. Though numerous solutions have been proposed, using statistical methods [13], decision rules [23], production rules [11], Markov processes [8] and Hierarchical Task Network Planning [17, 21, 22], most of them do not take into account eventual competency dependencies among learning objects and/or are not designed for large repositories of interdependent learning objects[1].

Therefore, we detailed in [7] a dynamic graph based model and a heuristic approach tailored to find a learning path in a graph containing millions of learning object nodes.

This paper is an extension of this previous work and summarizes the model, the heuristic presented in [7], and proposes a major optimization to calculate a global optimum learning path. In the previous work [7], we applied a greedy heuristic algorithm to obtain a pseudo-optimal learning path from a set of cliques. Greedy heuristics are efficient, but they sometimes get stuck in a local solution and fail to find a global optimum [26]. They are based on an intimate knowledge of the problem structure and have no scope of incremental improvement.

[1] A more complete discussion can be found in [7].

Therefore, in this work we slightly reformulate our model in order to fit as an integer programming problem and we propose an exact method based on the branch-and-bound algorithm.

2. PROBLEM CONSIDERED

In order to facilitate the understanding of the presented model, several key elements and assumptions need to be clearly defined.

A competency can be seen as a knowledge component being part of a "model that decomposes learning into individual knowledge components (KCs)" [16]. In this paper, a competency is "an observable or measurable ability of an actor to perform a necessary action(s) in given context(s) to achieve a specific outcome(s)" [12]. A competency in our situation can be a prerequisite to the efficient completion of a learning object. According to Wiley [25], a learning object is "any digital resource that can be reused to support learning". In the rest of the paper we define the learning object as any digital resource that can be reused to provide a competency gain.

A learner is a dynamic user interacting with learning objects in order to increase his/her competencies from an initial set to a targeted set of competencies. We assume that a learner completing a learning object will gain the competencies targeted to be transmitted by the interaction with the learning object. We also assume that a learner who would not possess the prerequisite set of competencies required by a learning object should not attempt this learning object since this would result in a limited competency gain.

Last but not least, we assume that the number of learning objects available is very large (millions to billions of learning objects) and that each learning object cannot provide the gain of a competency that is a pre-requisite to itself.

2.1 Graph Theory Contribution

Graph theory aims at studying mathematical structures composed of elements having relationships or connection between them. The use of directed graphs is not a novelty in e-learning systems [1, 3, 24, 25]; however, we were unable to find a formal model for discussing learning path problems based on graph theory, especially one taking into account the dynamic nature of a learning environment.

A directed graph, or digraph, *G = (V, E)* consists of:

- A non-empty finite set *V* of elements called vertices or nodes,
- A finite set *E* of distinct ordered pairs of vertices called arcs, directed edges, or arrows.

Let *G = (V, E)* be a directed graph for a personalized learning path. Each vertex or node in G corresponds to a learning object. Two vertices are connected if there exists a dependency relation, such that one vertex satisfies the prerequisites of the other. So, each edge between two vertices $Arc\{u, v\}$ means that the learning object v is accessible from u. The accessibility property required to define edges between vertices relies on post and pre-requisite competencies associated to each learning object. Considering $Arc\{u, v\}$, this edge means that after having completed the learning object *u*, the learner should have the required competencies to undertake resource *v*. By extension, each vertex *v* is represented by a pair (C_{pre}, C_{post}) where:

- C_{pre} is a set of the competencies required by vertex *v*
- C_{post} is a set of competencies offered by vertex *v*

The relationship between learning objects and competencies is multidimensional [6]: a learning object can require several competencies and transmit more than one competency to the learner as well. The existence of an edge between two learning objects *u* and *v* can be formalized by the following formula:

$$C_{pre}(v) \subseteq C_{post}(u) \Rightarrow Arc\{u, v\}$$

$$(Condition\ 1)$$

where $C_{pre}(v) \subseteq C_{post}(u)$ means that the competencies required by *v* are provided by learning object *u*. *Condition 1* is sufficient but not necessary. For example, before having completed *u*, the learner might already have some or the totality of the competencies required by *v*. This means that we may have an arc between *u* and *v* even though none the competencies required by *v* are provided by *u*. In other words, edge set *E* also depends on the learner's competency set at time *t*: $E = E(\mathrm{C}_{\mathrm{learner}}(t))$ and $\mathrm{C}_{\mathrm{learner}}(\mathrm{t}) = \{c_1, \ldots, c_n\}$ where $c_1 \ldots c_n$ are competencies which the learner possesses. As a result, graph *G* is a dynamic directed graph and *condition 1* can be strengthened by the necessary and sufficient *condition 2*:

$$Arc\{u, v\} \Leftrightarrow C_{pre}(v) \subseteq C_{post}(u) \cup C_{learner}(t)$$

$$(Condition\ 2)$$

2.2 Model Dynamicity

The dynamicity of our model is due to the fact that a learning object can bring competencies that could be among the prerequisites of future learning objects.

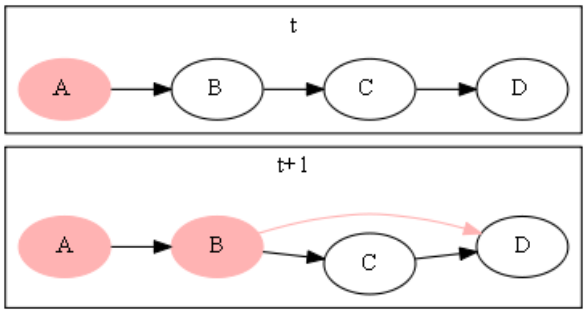


Figure 1. Edge dynamicity.

For example, as shown in Figure 1, a learning object D could be accessible to a learner if he has acquired the competencies c1 and c2. Assuming that competency c1 is provided by learning objects A and C and competency c2 is provided by learning objects B and C; D is reachable if learning objects A and B are completed or if learning object C is completed. If a learner completes learning object A at time t and learning object B at time t+1, the learner will have the competencies required to reach D and according to the condition 2, a new edge between B and D will be created (red edge on Figure 1).

3. INVESTIGATED SOLUTION

3.1 Reducing the solution space

Eliminating irrelevant learning objects is generally the first step of a course generation tool [1, 15]. In our case, as the learning object repository is supposed to be very large, the learning objects

cannot all be checked individually. The approach we chose consists in reducing the considered solution space by obtaining an induced subgraph H which consists of all the vertices and edges between the vertices in G that could be used in the learning path.

The algorithm can be seen as a loop generating complete sub-graphs, or cliques, until one such clique is generated whose prerequisites are a subset of the learner's competencies. Cliques are generated in a top-down fashion where we begin with the target clique, which is composed of a single learning object (we create a fictitious learning object, β, whose prerequisite competencies correspond to the list of the learner's target competencies). Cliques are then generated by finding every vertex where at least one output competency is found in the prerequisite competencies of the clique (the union of all prerequisite competencies of every learning object within the clique) to which it is prerequisite. As such, cliques contain the largest possible subset of vertices which satisfies the condition "if every learning object in the clique is completed, then every learning object in the following clique is accessible". We simplify the stopping condition by adding a second fictitious object, α, into the dataset with no prerequisite competencies and with the learner's current competencies as its output competencies. If a clique contains this object, the stopping condition is true.

	β_6	
v_1	$A^6_5 \quad E^6_{3,5}$	↑ **6**
v_2	$T^{3,2,4}_7 \; U^5_0$	↑ **3,5**
v_3	$L^{0,7}_{8,9} \; I^7_9 \; K^0_8$	↑ **0, 7**
	$A^{8,9}$	↑ **8, 9**

α: Fictitious LO with initial learner competency state

β: Fictitious LO with targeted learner competency state

$LO^{\text{list of gained competencies}} \; LO_{\text{list of prerequisite competencies}}$

Figure 2. Induced sub-graph generation.

Considering the target competency β as shown in Figure 2, all the vertices leading to those competencies (competency 6 in Figure 2) are selected in a set v_1, then the learning objects leading to the prerequisites of set v_1 (competencies 3 and 5) are selected from graph G to create the set v_2. This mechanism continues until the prerequisite competencies of the set v_n are all competencies which the learner has already acquired.

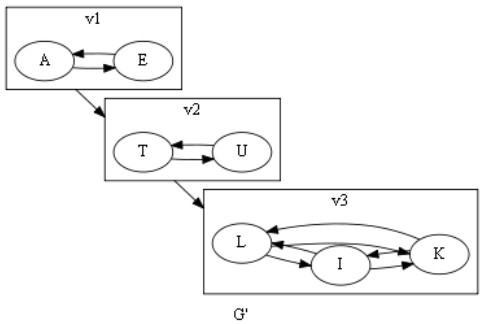


Figure 3. G' consists of connected cliques.

As shown in Figure 3, G', consisting of the vertices E of sets $v_1,\ldots,v_n$, is an induced sub-graph of G. If the learner has completed all the vertices of v_i, he/she will have access to all the vertices of v_{i+1}, thus all subsets of vertices of v_i can be considered to be a clique.

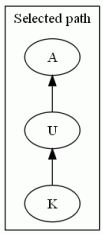
In addition to reducing the solution space, clique generation is also an efficient way to check whether a solution learning path exists between α and β. If the algorithm is not able to generate cliques linking α and β, there is no need to proceed forward with an algorithm aiming at finding one of the possible solutions.

3.2 Greedy Algorithm

Once the induced sub-graph is obtained, we use a greedy algorithm that searches for a local optimum within each clique. The definition of such a local optimum, depending on the dataset and the results pursued, has to follow a specific heuristic or strategy.

The shortest path strategy seems to be widely accepted in the literature [1, 27]. This strategy is not necessarily the best to adopt in any situation since the proposed learning path might lead to the learning of non–essential competencies and potentially cognitive overloads. For example a learning object could lead to competency gains that would not be required to reach the targeted learner competency state; there is no need to understand the proof of the Landau Damping to learn about the history of theoretical physics. Considering a learning object presenting an introduction to the perturbation theory and a second one introducing the theory and the proof of the Landau Dumping, it might make sense to choose the first one in order to minimize the cognitive load to the learner. Some might argue that using such "straight to the point" heuristic might limit too drastically the natural curiosity of the learner. As any heuristic, we agree that it is discussable but this is not the purpose of this paper.

The greedy algorithm considered attempts to find a path by considering each clique one after the other and reducing it to a minimal subset of itself which still verifies the condition "if every learning object in the clique is completed, then every learning object in the following clique is accessible".



	β_6	
v_1	$\mathbf{A^6_5}$ $E^6_{3,5}$	↑ **6**
v_2	$T^{3,2,4}_7$ $\mathbf{U^5_0}$	↑ **3,5**
v_3	$L^{0,7}_{8,9}$ I^7_9 $\mathbf{K^0_8}$	↑ **0,7**
	$A^{8,9}$	↑ **8, 9**

α: Fictitious LO with initial learner competency state

β: Fictitious LO with targeted learner competency state

$LO^{\text{list of gained competencies}}\ LO_{\text{list of prerequisite competencies}}$

Figure 4. Illustration of the greedy algorithm execution

The first clique considered will be the one leading to the targeted competencies (the clique satisfying the prerequisites of β). In the case of the three cliques v_1 to v_3 as illustrated by Figure 3, v_1 will be considered first followed by v_2 then by v_3.

For each clique, the local optimum is considered obtained when the minimum subset of vertices with a minimum "degree", being the sum of the number of prerequisite competencies and output competencies of the vertex, are found. In other words, the greedy algorithm select in each clique a set of learning objects minimizing the number of competencies required and gained in order to locally limit the cognitive load of the selected material. The greedy algorithm locally optimizes a function called "deg" (for degree) detailed in the following section.

For clique v_1, the selected learning object is A since its number of prerequisites is smaller than that of E while they share the same competency gain. As A has been chosen in v_1, only the objects in v_2 respecting the new v_1's prerequisites is chosen. As a result, the algorithm chooses U in v_2. In v_3, K and L lead to v_2's prerequisite but K requires fewer prerequisites than L, therefore K is selected and the proposed learning path is $K \rightarrow U \rightarrow A$.

4. OPTIMIZATION

In this section we present our mathematical model for learning path discovery and then we introduce the algorithm for solving our mathematical model.

After eliminating irrelevant learning objects in the first step, we generate the optimal solution from the obtained induced sub-graph as presented in Figure 4. For this purpose, we applied in [7] a greedy algorithm to obtain an optimal or pseudo-optimal learning path from a set of cliques. Greedy heuristics are computationally efficient, but they sometimes fail to find a global optimum as we explain in the following section.

4.1 Notation and limits of the Greedy heuristic

Let $Q_{n,m}$, $G_{n,m}$, $C_{n,v}$ the matrices representing the distribution of the m competencies that are prerequisite to the n items contained in the v cliques, the m competencies that are gained when the n items of the v cliques are performed, and the clique distribution of the n items. Note that the matrix $Q_{n,m}$ could be considered as a Q-Matrix [5].

Considering our example (*Example 1*):

- $N = \{A, E, T, U, L, I, K\}$,
- $M = \{0,2,3,4,5,6,7,8,9\}$,
- $V = \{v_1, v_2, v_3\}$.

$$Q_{n=7,m=9} = \begin{pmatrix} & 0 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ A & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 \\ E & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ T & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ U & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ L & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ I & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ K & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 \end{pmatrix}$$

$$G_{n=7,m=9} = \begin{pmatrix} & 0 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ A & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 \\ E & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ T & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ U & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 \\ L & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ I & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ K & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$C_{n=7,v=3} = \begin{pmatrix} & \mathrm{v}_1 & \mathrm{v}_2 & \mathrm{v}_3 \\ \mathrm{A} & \mathbf{1} & 0 & 0 \\ \mathrm{E} & 1 & 0 & 0 \\ \mathrm{T} & 0 & 1 & 0 \\ \mathrm{U} & 0 & \mathbf{1} & 0 \\ \mathrm{L} & 0 & 0 & 1 \\ \mathrm{I} & 0 & 0 & 1 \\ \mathrm{K} & 0 & 0 & \mathbf{1} \end{pmatrix}$$

From this example the solution sequence using the greedy algorithm is $K \rightarrow U \rightarrow A$.

To check if we get an optimal solution or not, we have to calculate the objective function called deg. The objective function deg returns the total number of prerequisite and gained competencies of a set of learning objects.

We can draw from the previous example the following conditions to check if we have an optimal solution or not.

Let $S = \{s_0, s_1, \dots, s_v, s_{v+1}\}$ a solution set (s_i contains at least one learning object as in example 3).

$$\forall s_{i=1..v} \in S, \quad s_0 = \alpha,\ s_{v+1} = \beta,\ Q_{s_i} \subseteq G_{s_{i-1}} \quad (i)$$

$$\forall\, j = 1 \dots v \neq i = 1 \dots v,\ C_{s_i} \cap C_{s_j} = \emptyset \quad (ii)$$

$$\deg(S = \{s_0, s_1, \dots, s_v, s_{v+1}\}) = \sum_{i=0}^{v+1} \sum_{j=1}^{m} (Q_{s_i,j} + G_{s_i,j}) \quad (iii)$$

$$\forall s_{i=1..v+1} \in S;\ \exists\, s^*_{i=1..v+1} \in S^*$$

$$\deg(S^* = \{s^*_0, s^*_1, \dots, s^*_v, s^*_{v+1}\}) \leq \deg(S = \{s_0, s_1, \dots, s_v, s_{v+1}\})$$

$$(iv)$$

Condition (i) and (ii) mean that the competencies required by a clique set have to be covered by the gains of the previous clique set and two different clique sets cannot share the same clique. While condition (iii) defines the deg function, condition (iv) introduces the optimality condition. A learning path is optimal if

no other path with a lower degree exists. However this doesn't apply at the clique level since the optimal s_i^* is not necessary the set of clique i having the lowest degree. The global optimum is not the sum of the local optima calculated by the greedy algorithm.

The following example highlights this case where local optima obtained by the greedy algorithm lead to non-optimal solution.

Example 2:

	β_6	
v_1	$M^6{}_5$ $N^{6,7}{}_4$	↑ **6**
v_2	$O^5{}_{3,4}$ $P^4{}_8$	↑ **4,5**
v_3	$T^8{}_7$ $R^{3.4}{}_7$	↑ **3, 4, 8**
	α^7	↑ **7**

$$\deg(\alpha, R, O, M, \beta) = 1 + 3 + 3 + 2 + 1 = 10$$
$$\deg(\alpha, Q, P, N, \beta) = 1 + 2 + 2 + 3 + 1 = 9$$

The solution obtained by the greedy algorithm is $S_a = \alpha \to R \to O \to M \to \beta$ and the associated value of the objective function deg (S_a) is equal to 10. As the algorithm starts from β, it chooses in each clique the learning object with the lowest degree which is M and keeps going until it reaches α.

The path $S_b = \alpha \to T \to P \to N \to \beta$ is an alternative that the algorithm did not find. It's even a better alternative since $\deg(S_b) = 9 \leq \deg(S_a) = 10$ and the optimal solution.

The following example highlights another case where local optima obtained by the greedy algorithm lead to a non-optimum solution. In this example, two learning objects are selected in one of the generated cliques.

Example 3:

	β_6	
v_1	$M^6{}_5$ $N^{6,7}{}_4$	↑ **6**
v_2	$O^5{}_{3,9}$ $P^4{}_8$	↑ **4,5**
v_3	$T^8{}_7$ $Y^9{}_7$, $Z^3{}_7$	↑ **3, 9, 8**
	α^7	↑ **7**

$$\deg(\alpha, Y, Z, O, M, \beta) = 1 + 2 + 2 + 3 + 2 + 1 = 11$$

The objective function of the path $(\alpha \to T \to P \to N \to \beta)$ is 9, which means that the path $(\alpha \to T \to P \to N \to \beta)$ is the optimal solution.

In the following section, we use the notation introduced here to propose a mathematical formulation of our learning path optimization problem as an integer programming problem.

4.2 Formulating the integer programming problem

Let us consider n items or learning objects and m competencies; $Q_{n,m}$ is the matrix representing m prerequisite competencies for the n items and $G_{n,m}$ is the matrix representing the m competencies that are gained when the n items are performed. In other words, if $Q_{ij} = 1$ means that the item i has competency j as one of its prerequisite competencies; and $G_{i,j} = 1$, means that the competency j is gained when the item i is performed. The personalized learning path may then be formulated as a binary integer programming (BIP) as follows:

Minimize:

$$\sum_{i=1}^{n}\left(\sum_{j=1}^{m}\left(Q_{i,j} + G_{i,j}\right)x_i\right) = \deg(X) \qquad (1)$$

Subject to:

$$Q_{i,j}x_i - \left(\sum_{k=1}^{i-1} G_{k,j}x_k\right) \times Q_{i,j} \leq 0 \qquad (2)$$

$$for \quad i = 2, \dots, n-1; \quad for \quad j = 1, \dots, m; \quad x_i \in \{0,1\};$$

X = $\{x_i, i=1,\dots,n\}$, are the decision variables such that:

$$x_i = \begin{cases} 1 & if\ the\ item\ i\ is\ selected; \\ 0 & otherwise, \end{cases} \qquad (3)$$

We suppose that $x_1 = 1$ and $x_n = 1$, knowing that:

$$x_1 = 1\ presenting\ the\ initial\ item\ \alpha$$
$$and\ x_n = 1\ presenting\ the\ resulting\ item\ \beta$$

The function (1) represents the total number of prerequisite and gained competencies to be minimized. The constraints (2) states that if the item i has competency j as one of its prerequisite competencies; the competency j should be gained from the items on the learning path (1,…, i-1). Our problem is to minimize the objective function (1) subject to (2) and (3).

To find the optimal learning path we have to solve the BIP problem with $(n+m)$ constraints and n decision variables $x_{i=1,\dots n} \in \{0,1\}$.

Considering example 3, the prerequisite and gain matrices Q and G can be written as follows:

The competencies that are required by the items are represented by the matrix Q (9x7).

$$Q = \begin{pmatrix} \frac{Comp.}{LO} & C1 \equiv 7 & C2 \equiv 8 & C3 \equiv 9 & C4 \equiv 3 & C5 \equiv 5 & C6 \equiv 4 & C7 \equiv 6 \\ \alpha & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ T & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ Y & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ Z & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ O & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ P & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ M & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ N & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ \beta & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

The competencies that are gained by the items are represented by the matrix G (9x7).

$$G = \begin{pmatrix} \frac{Comp.}{LO} & C1 \equiv 7 & C2 \equiv 8 & C3 \equiv 9 & C4 \equiv 3 & C5 \equiv 5 & C6 \equiv 4 & C7 \equiv 6 \\ \alpha & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ T & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ Y & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ Z & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ O & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ P & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ M & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ N & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ \beta & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

The BIP formulation of example 3 is given as follows:

Minimize :

$$\deg(X) = x_1+2x_2+2x_3+2x_4+3x_5+2x_6+2x_7+3x_8+x_9$$

Subject to:

$$x_2 - x_1 \leq 0$$
$$x_3 - x_1 \leq 0$$
$$x_4 - x_1 \leq 0$$
$$x_5 - x_3 \leq 0$$
$$x_5 - x_4 \leq 0$$
$$x_6 - x_2 \leq 0$$
$$x_7 - x_5 \leq 0$$
$$x_8 - x_6 \leq 0$$
$$x_9 - x_7 - x_8 \leq 0$$
$$x_i \in \{0,1\}, i = 1, \dots, 9$$

x_1 *is* the fictitious learning object α with initial learner competency state.

x_9 is the fictitious learning object β with targeted learner competency state.

Since $x_1 = x_9 = 1$, then our BIP becomes:

Minimize :

$$\deg(X) = 2x_2+2x_3+2x_4+3x_5+2x_6+2x_7+3x_8$$

Subject to:

$$x_5 - x_3 \leq 0$$
$$x_5 - x_4 \leq 0$$
$$x_6 - x_2 \leq 0$$
$$x_7 - x_5 \leq 0$$
$$x_8 - x_6 \leq 0$$
$$-x_7 - x_8 \leq -1$$
$$x_i \in \{0,1\}, i = 2, \dots, 8$$

4.3 The Branch-and-Bound (B&B) method for solving the BIP problem

Since the BIP problem is bounded, it has only a finite number of feasible solutions. It is then natural to consider using an enumeration procedure to find an optimal solution. However, in the case of large learning object repositories (millions of items), an enumeration procedure might be ill adapted (even after reducing the solution space); therefore, it is imperative to cleverly structure the enumeration procedure so that only a tiny fraction of feasible solutions need to be explored.

A well-known approach called branch-and-bound technique (B&B) provides such a procedure. B&B traces back to the 1960s' and the work of Land and Doig [14]. Since then, B&B algorithms have been applied with success to a variety of operations research problems. B&B is a divide and conquer method. It divides a large problem into a few smaller ones (This is the "Branch" part). The conquering part estimates the goodness of the solution that is obtained from each of the sub-problems; the problem is divided until solvable sub-problems are obtained (this is the "bound" part).

For the bounding part we use a linear programming relaxation to estimate the optimal solution [26]. For an integer programming model P; the linear programming model obtained by dropping the requirement that "*all variables must be integers*" is called the linear programming relaxation of P.

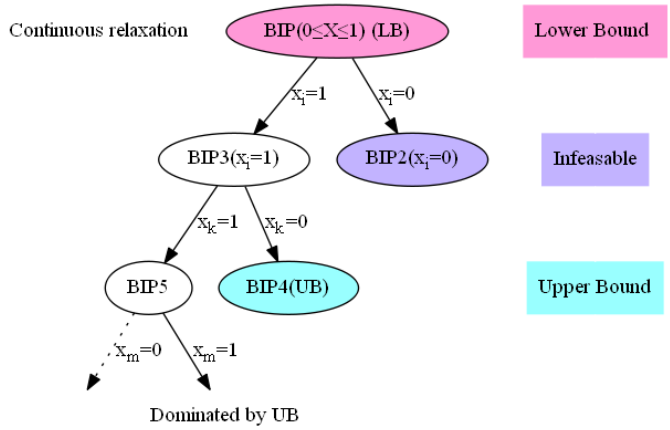


Figure 5. Branch and bound algorithm that traverses the tree by solving BIPs at every node of the tree.

The general approach of a BIP B&B algorithm [26] is presented in the following steps (see also Figure 5):

Initialization: Set deg* = $+\infty$.

The initial step represents the root node of the B&B search tree. The root node corresponds to the continuous relaxation of the BIP($0 \leq X \leq 1$), the solution value provides lower bound.

Apply the bounding step, fathoming step, and optimality test described below. If not fathomed, classify this problem as the one remaining "subproblems" for performing the first full iteration below.

Steps for each iteration:

1. Branching: Among the remaining (unfathomed) subproblems, select the one that was created most recently (break ties by selecting the subproblem with the larger bound). Branch from the node for this subproblem to create two new subproblems by fixing the next variable (the branching variable) at either 0 or 1 (see Figure 5).
2. Bounding For each new subproblem, obtain its bound by applying the simplex method to its LP-relaxation and rounding down the value of deg for the resulting optimal solution.
3. Fathoming (Pruning rules): The pruning rules for B&B BIP are based on optimality and feasibility of BIP. For

each new sub-problem, apply the fathoming tests and discard those sub-problems that are fathomed by any of the tests.

Optimality test: Stop when there are no remaining sub-problems:

- The current incumbent is optimal,
- Otherwise, return to perform another iteration.

A sub-problem is fathomed (dismissed from further consideration) if it verifies one of the following tests:

1. The relaxation of the sub-problem has an optimal solution with deg < deg∗ where deg* is the current best solution (The solution is dominated by upper bound);
2. The relaxation of the sub-problem (LP-relaxation) has no feasible solution;
3. The relaxation of the sub-problem has an optimal solution that has all binary values. (If this solution is better than the incumbent, it becomes the new incumbent, and test1 is reapplied to all unfathomed sub-problems with the new larger deg*).

For example, the *example 3* solved by B&B produces an optimal solution with deg* = 9 and x_2=1, x_6=1, x_8=1 where the number of nodes explored is 1 because the first LP-relaxation at node 1 gives an integer optimal solution with deg*=9 and the 3rd fathomed test is true, so we do not need to branch anymore.

Decision Variables	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9
LO	α	T	Y	Z	O	P	M	N	β
X*	1	1	0	0	0	1	0	1	1

Figure 6. Solution of example 3in the B&B algorithm.

As illustrated in Figure 6, the optimal solution of the B&B algorithm is X*={1, 1, 0, 0, 0, 1, 0, 1, 1} and the optimal path is: $\alpha \rightarrow T \rightarrow P \rightarrow N \rightarrow \beta$.

5. CONCLUSION

The clique based approach is an asset since it offers an efficient way to reduce the solution space and check the existence of a solution. However, a greedy search within the cliques to find a leaning path does not lead, in many cases, to the best learning path according to the criteria considered.

Binary integer programming is a well-known mathematical optimization approach. While reformulating the conditions an optimal learning path should meet, we realised how we could benefit from expressing the constraints as a binary programming problem.

Our preliminary implementation of the proposed optimization using the *bintprog* function (*MATLAB*), a function based on the branch- and-bound (B&B) algorithm, shows the accuracy of the proposed integer program model.

In future work, we will apply the proposed binary integer model in order to build a learning design recommendation system in the case where learning objects are stored in very large repositories. Even though the B&B algorithm is highly accurate and somehow computationally efficient, it is not efficient enough to deal with very large size problem instances. In some cases, the bounding step of B&B is not invoked, and the branch and bound algorithm can then generate a huge number of sub-problems.

Moreover, as mentioned in [7], the efficiency of reducing the solution space with the cliques' mechanism is highly dependent on the dataset topology (average number of gain and prerequisite competencies per learning object). The solution space may remain large after the reduction

Therefore, to deal with very large problems, we will implement a variant of the B&B algorithm such as Branch & Cut [2] or Branch & Price [4]. Applegate et al. [2] showed how Branch & Cut could get a global optimal for extremely large binary optimization problems. It will be then interesting to measure both in terms of computational time and accuracy how the greedy search compares to the B&B-like approach.

6. ACKNOWLEDGMENTS

This work is part of the National Research Council of Canada's Learning and Performance Support Systems (NRC LPSS) program. The LPSS program addresses training, development and performance support in all industry sectors, including education, oil and gas, policing, military and medical devices.

7. REFERENCES

[1] Alian, M. Jabri, R. 2009. A shortest adaptive learning path in e-learning systems: Mathematical view, *Journal of American Science* 5(6) (2009) 32-42.

[2] Applegate, D., Bixby, R., Chvatal, V. and Cook, W. 1998. On The solution of traveling salesman problems, *in: Proc. Int. Congress of Mathematicians, Doc. Math. J. DMV*, Vol. 645.

[3] Atif, Y., Benlarmi, R., and Berri, J. 2003. Learning Objects Based Framework for Self-Adaptive Learning, *Education and Information Technologies, IFIP Journal, Kluwer Academic Publishers* 8(4) (2003) 345-368.

[4] Bamhart, C, Johnson, E. L., Nemhauser, G. L., Savelsbergh, M. W. P. and Vance, P. H. 1998. Branch-and-price: column generation for huge integer programs, *Operations Research* 46:316.

[5] Barnes, T. 2005. The Q-matrix Method: Mining Student Response Data for Knowledge. *Proceedings of the Workshop on Educational Data Mining at the Annual Meeting of the American Association for Artificial Intelligence*.

[6] Carchiolo, V., Longheu, A., and Malgeri, M. 2010. Reliable peers and useful resources: Searching for the best personalised learning path in a trust- and recommendation-aware environment, *Information Sciences* 180(10) (2010) 1893-1907.

[7] Durand, G., Belacel, N., and Laplante, F. 2013. Graph theory based model for learning path recommendation. *Information Sciences.* 251(10) (2013) 10-21.

[8] Durand, G., Laplante, F. and Kop, R. 2011. A learning Design Recommendation System Based On Markov Decision Processes, Proc. *17th ACM Conference on Knowledge Discovery and Data Mining (SIGKDD) Workshop on Knowledge Discovery in Educational Data*, San Diego, CA.

[9] Durand, G., Downes, S. 2009. Toward Simple Learning Design 2.0. *In: 4th Int. Conf. on Computer Science &Education 2009*, Nanning, China, 894-897.

[10] Godoy, D., Amandi, A. 2010. Link Recommendation in E-learning Systems based on Content-based Student Profiles, In: Romero C., Ventura S., Pechenizkiy, M., Baker, R. (Eds.), *Handbook of Educational Data Mining, Data Mining*

and Knowledge Discovery Series, Chapman & Hall/CRC Press, 273-286.

[11] Huang, Y.M., Chen, J.N., Huang, T.C., Jeng, Y.L., and Kuo, Y.H. 2008. Standardized course generation process using Dynamic Fuzzy Petri Nets, *Expert Systems with Applications*, 34 (2008) 72-86.

[12] ISO 24763/final version: Conceptual Reference Model for Competencies and Related Objects, 2011.

[13] Karampiperis, P., Sampson, D. 2005.Adaptive learning resources sequencing in educational hypermedia systems. *Educational Technology & Society* 8 (4) (2005) 128-147.

[14] Land, A. H., Doig, A. G. 1960. An automatic method of solving discrete programming problems. *Econometrica* 28(3), 497–520.

[15] Liu, J., Greer J. 2004. Individualized Selection of Learning Object, *In: Workshop on Applications of Semantic Web Technologies for e-Learning*, Maceió, Brazil.

[16] Pavlik, P. I. Jr., Presson, N., and Koedinger K. R. 2007. Optimizing knowledge component learning using a dynamic structural model of practice, *Proc. 8th International Conference on Cognitive Modeling.* Ann Arbor, MI.

[17] Sicilia, M.-A., Sánchez-Alonso, S. and García-Barriocanal, E. 2006. On supporting the process of learning design through planners, Proc. *Virtual Campus Post-Selected and Extended*, 81–89.

[18] Tang, T.Y., Mccalla, G.G. 2010. Data Mining for Contextual Educational Recommendation and Evaluation Strategies, In: Romero C., Ventura S., Pechenizkiy, M., Baker, R. (Eds.), *Handbook of Educational Data Mining, Data Mining and Knowledge Discovery Series*, Chapman & Hall/CRC Press, Chapter 18,257-271.

[19] Trcka, N., Pechenizkiy, M. and Van-Deraalst, W. 2010. Process Mining from Educational Data, In: Romero C., Ventura S., Pechenizkiy, M., Baker, R. (Eds.), *Handbook of Educational Data Mining, Data Mining and Knowledge Discovery Series*, Chapman & Hall/CRC Press, Chapter 9, 123-141.

[20] Ullrich, C., Melis, E. 2010. Complex Course Generation Adapted to Pedagogical Scenarios and its Evaluation, *Educational Technology & Society*, 13 (2) (2010) 102–115.

[21] Ullrich, C., Melis, E. 2009. Pedagogically founded courseware generation based on HTN-planning, *Expert Systems with Applications* 36(5) (2009) 9319-9332.

[22] Ullrich C. 2005. Course Generation Based on HTN Planning, Proc. *13th Annual Workshop of the SIG Adaptivity and User Modeling in Interactive Systems*, Saarbrucken, Germany,74-79.

[23] Vassileva, J., Deters, R. 1998, Dynamic courseware generation on the www, *British Journal of Educational Technology*, 29(1) (1998) 5–14.

[24] Viet, A., Si, D.H. 2006. ACGs: Adaptive Course Generation System - An efficient approach to Build E-learning, *Proc. 6th IEEE International Conference on Computer and Information Technology*, Jeju Island, Korea, 259-265.

[25] Wiley, D.A. 2002. Connecting Learning Objects to Instructional Design Theory: A Definition, a Metaphor, and a Taxonomy, *In: The Instructional Use of Learning Objects, D. A. WILEY (Ed.)*, 3-23.

[26] Winston, W.L., Venkataramanan, M. 2003. *Operations Research: Introduction to Mathematical Programming*. Thompson, 4th Edition.

[27] Zhao, C., Wan, L. 2006. A Shortest Learning Path Selection Algorithm in E-learning, *Proc. 6th IEEE International Conference on Advanced Learning Technologies*, Kerkrade, The Netherlands, 94-95.

[28] Zhou, M., Xu, Y., Nesbit, J.C. and Winne, P.H. 2010. Sequential pattern analysis of learning logs: Methodology and applications, In: Romero C., Ventura S., Pechenizkiy, M., Baker, R. (Eds.), *Handbook of Educational Data Mining, Data Mining and Knowledge Discovery Series*, Chapman & Hall/CRC Press, Chapter 8, 107-120.