sciendo

BULGARIAN ACADEMY OF SCIENCES

CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **19**, No 3
Sofia • 2019 Print ISSN: 1311-9702; Online ISSN: 1314-4081
DOI: 10.2478/cait-2019-0024

Course Sequence Recommendation with Course Difficulty Index Using Subset Sum Approximation Algorithms

M. Premalatha, V. Viswanathan

School of Computing Science & Engineering, Vellore Institute of Technology, Chennai 600027, India
E-mails: premalatha.m@vit.ac.in viswanathan.v@vit.ac.in

***Abstract*:** *Choice Based Course Selection* (*CBCS*) *allows students to select courses based on their preferred sequence. This preference in selection is normally bounded by constraints set by a university like pre-requisite*(*s*)*, minimum and maximum number of credits registered per semester. Unplanned course sequence selection affects the performance of the students and may prolong the time to complete the degree. Course Difficulty Index* (*DI*) *also contributes to the decline in the performance of the students. To overcome these difficulties, we propose a new Subset Sum Approximation Problem* (*SSAP*) *aims to distribute courses to each semester with approximately equal difficulty level using Maximum Prerequisite Weightage* (*MPW*) *Algorithm, Difficulty Approximation* (*DA*) *algorithm and Adaptive Genetic Algorithm* (*AGA*). *The three algorithms have been tested using our university academic dataset and DA algorithm outperforms with 98% accuracy than the MPW and AGA algorithm during course distribution.*

***Keywords*:** *Course sequence recommendation, Course credits, Course difficulty, Pre-requisite weight, Approximation Algorithm.*

1. Introduction

University Grant Commission (UGC) of India has instructed the universities of India to implement the Choice Based Credit System (CBCS) in the order issued during 2015 [1]. In CBCS, students have the freedom to select courses of their interests and help them to pursue courses at different times in different semesters [2]. Vellore Institute of Technology (VIT) is the first university in India to introduce the Fully Flexible Credit System (FFCS) [3] in which 87% of our university students are comfortable with the choice of selecting the courses, timings, and the respective instructors [3, 4]. For four-year engineering graduation in VIT, as per the, a student needs to register a minimum of 4 courses (16 credits) per semester from the university curriculum with approximately 30 mandatory courses leaving the elective courses behind. Unplanned course selection without a clear plan increases the graduation duration of a student [5]. A personalized recommendation helps the student to register the desired course in a proper sequence, which will help them to complete their

courses without delay [5]. Apart from course selection, the difficulty of a course also affects the students' performance, lowers their grades, and sometimes leads to course dropout [6]. Students who register for courses with high difficulty index, however, tend to earn lower Grade Point Average (GPA) and experience lower retention [7]. Any effect of credit load on retention appears to work through GPA. Researchers have found that students find some required information technology programme courses to be quite difficult, e.g., mathematics and programming. Such types of difficult courses could lower students' grades, which in turn could lower their motivation to study, thus causing dropout [8].

This paper considers only mandatory courses from our university undergraduate computer science and engineering curriculum for personalized recommendations, and elective courses are not considered for the proposed method. For n set of mandatory courses in VIT curriculum, there are 35% of university core and 65% of programme core courses. With n courses and with a minimum of 1 prerequisite course, a student has n^{n-1} course sequence combinations out of which every student has the freedom to complete the required courses in any sequence at each semester, with constraints which are identified by its course code, course credits, course prerequisite(s), course difficulty index (assumed since estimating course difficulty index itself is our part of our future research work) and course prerequisite weight. Student registers a set of courses for a semester and completes the same by getting the required grades, and the set of completed courses will be added to the students' academic history with the acquired grades and Cumulative Grade Point Average (CGPA). Curriculum courses which are not completed by a student are considered for further recommendations for the respective semesters. The students are recommended with the course(s) until the students complete all the required courses and credits.

A personalized course sequence recommendation at each semester of the degree programme hence hit as to be computed by adhering to the following constraints.

Course Prerequisites: The courses have to be registered in sequence if a course has the prerequisite(s). A prerequisite is a course which has to be registered before a specific course is registered. If a course has a set of prerequisite(s), all prerequisites should be completed before registering that particular course.

Course Credits: Every course C_i is specified by a course credit cc_i based on the number of lecture hours needed per week to complete the course. There are k semesters $\{s_1, s_2, \ldots, s_k\}$ and every semester has a Credit Limit (CL) such that the sum of credits of all the courses selected in a semester should be less than or equal to CL.

Course Difficulty: Mundfrom et al. recorded the course difficulty index in the Likert-type scale of 9 points from very easy to very difficult [9]. As a reference, this paper considers the Likert-type scale of 5 points for recording the course difficulty as very easy, easy, moderate, difficult, and very difficult from 1-5, respectively. Every course C_i is assumed with a course Difficulty DC_i. Courses are distributed to k semesters such that the average course difficulty of every semester is approximately equal to the Average Difficulty (AD) of all the courses.

Course Prerequisite Weight: The number of courses for which a course is been identified as a prerequisite is specified as prerequisite weight. Let say if C_1 is the prerequisite course for C_2, C_4, and C_5, the prerequisite weight pw_1 for C_1=3.

During the course registration process, there is a possibility of any student selecting very difficult courses for a semester or very easy courses for another. Students grades increase when the courses are easy and decreased when the courses are more difficult [6, 7]. Since the above issues related to course planning and course difficulty affects the student's performance, the course distribution recommendation must consider difficulty index as the main factor such that distribution of courses to each semester should have approximately equal difficulty index. This helps the students to score good grades and complete the courses on time.

A Subset Sum Approximation Problem (SSAP) is proposed to distribute the courses using two methodologies. The first method initially subgroups the curriculum courses based on the hierarchy level and solves the problem by considering two algorithms named Maximum Prerequisite Weightage (MPW) Algorithm and Difficulty Approximation (DA) Algorithm. The second method considers the curriculum course list, constructs a dependency matrix based on the prerequisite courses and uses a disagreement fitness function to distribute the courses using Adaptive Genetic Algorithm (AGA).

The paper is organized as follows: Section 2 discusses the related work Section 3 discusses the motivation behind the proposed work. Section 4 discusses the results of the work done. Section 5 summarizes the overall work as a conclusion.

2. Related works

The related works analyzes the impact of course difficulty on student's performance, course planning, and course sequencing. The difficulty of a course is estimated by considering the average grades awarded, rank correlation coefficient (rho) – means, scaling analysis and cluster analysis as factors [9, 10]. A study of item difficulty assessment depends on any of the factors like the learners' course contents, students' scores, and subject matter expert using the Apriori Algorithm [11]. Impact of course difficulty in students' performance is studied by Liu et al. [13], analyses by the previous students' results and predicts the results of the existing student's using the bagging algorithms. Based on the learning dependency, difficulty of knowledge units are ranked with respect to subjective difficulty and objective difficulty [13]. Some research discusses on what way the rating of instructors [14] and feedback [15] by student helps the instructors to improve their performance in examinations. Bloom's taxonomy is used for evaluating the cognitive level of a question paper with respect to the action words and based on which course's difficulty level and the student's cognitive skills can be identified [16]. Students' course outcomes are used for assessing the performance of a student in their examination based on the difficulty index of a course. With the help of bloom's taxonomy, final exam papers are evaluated and a difficulty index is identified [17]. A cognitive map [18] is provided to the research scholars, which gives them a clear picture of how to start and to proceed with their research. Course difficulty is evaluated using the factors like

student's learning skills; course contents' difficulty, student's feedback on the course and the faculty handled in the existing research. The proposed method doesn't consider any of the said factors for the course difficulty index and the values used are mere assumptions. The algorithm proposed works the same even when the difficulty of a course is static or dynamic for various semesters. The difficulty of a course depends on various factors and hence is considered for our future scope of research.

In terms of course recommendation, courses are recommended in sequence automatically for each student using a Bayesian approach [19, 20]. Wang and Yuan [21] recommend the courses based on user profiles from users' interest description, browse log and subscriptions. Learning objects are recommended sequentially [22] in which a personalized learning route is suggested to learn the sequence of learning object and if the student fails in the assessment during the learning process, the route will be modified/repaired and recommended with new objectives. The research work proposed by Xu, Xing and Van der Schaar [5] recommends personalized course sequence recommendation such that time to graduation is reduced along with an improvement in the student's grades. The courses were ranked and the best sets of courses with prerequisites were recommended in sequence [23] using breadth-first pickings, greedy-value pickings, and top-down pickings. Along with ranking and recommending courses with constraints, Parameswaran, Venetis and Garcia-Molina [24] have developed a model which checks for requirements which have to be satisfied as the students have to five math course using Integer Linear Programming Algorithm and Max-Flow Algorithm. Betancur et al. [25] assess the course sequencing recommendation analytically, evaluate them by applying research questions about the student achievement and their relationship with psychology courses, and analyze the recommendation of taking the psychology courses before and after the methodological courses using regression techniques. A personalized learning pathway is been recommended for e-Learning courses in which the course contents are recommended in sequence using item response theory methodology [26]. Based on the student's history of grades and on the performance of the student at each state, a directed structure with the state transitions has been constructed and the courses are recommended such that the grades are always balanced in each state [27]. An optimized course sequence recommendation with prerequisite constraints is solved using multiple integer linear programming algorithms and used structure-based heuristics for reducing the time to a degree [28]. Courses are ranked and optimal course sequences are recommended based on the student population's performance using a rank aggregation framework [29]. Multi-Armed bandits Personalization for Learning Environment (MAPLE) is an approach which considers the difficulty of the educational content and personalizes them for the student in such a way that the student's performance is increased with maximum grades [30]. Complexity of optimizing the sequencing problem is greatly reduced when it is initially subdivided and then sequenced. Gunji et al. [31] solved this optimal sequencing problem using Teaching Learning-Based Optimization (TLBO) Algorithm that subdivides the helicopter parts and then sequentially assembles the same, which hugely reduced the number of iterations by optimally assembling through subsets. One such method is

followed in the proposed method, which initially divides the curriculum courses into subsets based on their hierarchy level and then the courses are distributed to each semester which has greatly reduced the number of iterations during the course distribution.

The credit system for four-year graduation has 15-16 Semester Credit Hours (SCH) for a semester and 30 SCH for a year and a student have to complete 120 credits for four-year graduation. A course might have 3 to 4 credits based on the number of Contact Hours per week. The existing research has recommended 4 to 5 courses with 15-16 SCH per semester and has not considered credit limit, course difficulty for sequencing the courses in the curriculum. Unlike the commonly used credit system, our university uses flexible credit selection in which the courses have different credits based on the number of contact hours of a course, credit limit as minimum 16 and maximum 27 per semester and require a total of 180 credits to complete four-year graduation. Since the courses have different credits, the credit-based course sequence is recommended in the proposed method, which distributes courses based on constraints like course prerequisites, credit limit and average difficulty.

3. Proposed work

A course sequencing methodology uses our proposed Subset Sum Approximation Problem (SSAP), which is the extension of Subset Sum Problem (SSP). In SSP, with a given set $S = \{S_1, S_2, \ldots, S_n\}$ of positive integers and ss, all possible subsets $S'$are formed such that the sum of each subset is equal to ss:

$$\{(S, \text{ss}), \text{there exists a subset } S' \subseteq S \text{ such that } \textstyle\sum_{s \in S'} s = \text{ss}\}.$$

For example, let a set $S = \{1, 2, 3, 4, 5, 6\}$ and $\text{ss} = 6$. The data items in S are positive integers and are independent of each other. The subsets are of S are formed as $\{1, 2, 3\}$, $\{2, 4\}$, $\{1, 5\}$, and $\{6\}$ in which the sum of items in each subset is equal to 6. A Multiple Subset Sum Problem (MSSP) discussed by Caprara, Kellerer and Pferschy [32], [33] has added a knapsack feature to SSP in which the subset sum is less than or equal to ss.

$$\{(S, \text{ss}), \text{there exists a subset } S' \subseteq S \text{ such that } \textstyle\sum_{s \in S'} s \leq \text{ss}\}.$$

Here, subsets are formed as $\{1\}$, $\{2\}$, $\{3\}$, $\{4\}$, $\{5\}$, $\{6\}$, $\{1, 2\}$, $\{1, 3\}$, $\{1, 4\}$, $\{1, 5\}$, $\{2, 3\}$, $\{2, 4\}$, and $\{1, 2, 3\}$ in which the sum of items in the each subset is less than or equal to 6. But in SSAP there exists a set of positive numbers such that the sum of items of each subset is equal to ss with an error quotient added or subtracted to it as in (2). Unlike the SSP, SSAP considers all the possible positive numbers. There exists a dependency among the items in the set and the subsets are formed sequentially one after the other. When there is a dependency for a data item, it should satisfy certain constraints to form a subset that is approximately equal to ss.

The SSAP with constraints involves steps like course hierarchy estimation, course distribution using MPW, DA Algorithm by considering the estimated hierarchy and an evolutionary AGA algorithm without considering the estimated hierarchy.

In Figs 1 and 2, the courses specified in green rectangle boxes are prerequisite courses. Courses specified in blue rectangle boxes are post requisite courses and are not prerequisites for any other courses. Courses specified with orange rectangle boxes are both prerequisite and post requisite courses. A course specified in red dotted rectangle box is neither a prerequisite course nor a post requisite course. A course dependency structure represents the prerequisite among the courses as specified in Fig. 1. The dependency of a course represented as MAT101→MAT105 specifies that MAT101 is the prerequisite course of MAT105, i.e., MAT105 should be registered by the students only if they have completed MAT101. The introductory courses without prerequisites are first level courses. Level two represents courses with one level of prerequisites and so on. For example, MAT101, MAT106, CSE101, and MAT206 are first level courses. MAT105, MAT202, CSE202, and CSE220 have prerequisites in level one and are represented as level two courses. MAT207, MAT203 have two levels of prerequisite courses that are represented as level three courses. The course recommendation initially selects courses from level one and continues its recommendation to further levels until all courses are recommended at each level. The hierarchy for each course is estimated based on the number of level of prerequisite a course has and it is validated through Algorithm 1.

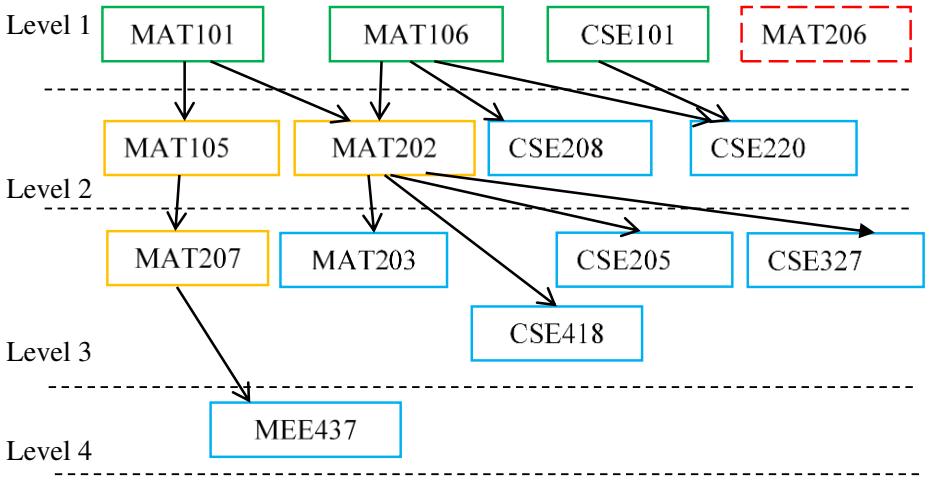

Fig. 1. Course dependency structure

As specified in Table 1, let $\{c_1, c_2,\ldots, c_n\}$ be the set of courses, $\{cc_1, cc_2,\ldots, cc_n\}$ be the set of course credits, $\{dc_1, dc_2,\ldots, dc_n\}$ be the set of course difficulties and $\{pw_1, pw_2,\ldots, pw_n\}$ be the set of prerequisite weights. The duration of the degree program is specified as k semesters and the curriculum courses are distributed to j semesters where $j = 1, 2,\ldots, N$. If a student completes all the required courses within the duration of the degree program, N will be equal to k otherwise the value of N is greater than k. Every semester S_j is distributed with a set of courses $\{c_i\}$, set of course credits specified as $SC_j = \{cc_i\}$, set of course difficulty specified as $SD_j = \{dc_i\}$, set of prerequisite weight specified as $SPW_j = \{pw_i\}$. The number of courses for S_j is specified as nc_j.

Table 1. Notations

Notation	Description
t	Number of students
x	Student $x = 1, 2,\ldots, t$
C	Set of all courses
n	Number of courses
i	Counter variable for courses: $i = 1, 2,\ldots, n$
c_i	i-th course
$p(c_i)$	Set of all prerequisites of c_i
cc_i, dc_i, pw_i, hc_i	Course credit, course difficulty index, course prerequisite weight, and course hierarchy level of the i-th course respectively
N	Number of semesters
j	Counter variable for semester: $j = 1, 2,\ldots, N$
nc_j	Number of courses in the j-th semester
r	Counter variable for $r = 1, 2,\ldots, nc_j$
max_l	Number of hierarchies
h	Counter variable for hierarchy level: $h = 1, 2,\ldots,$ max_l
S_j	Set of all courses of the j-th semester
SC_j, SD_j, SPW_j,	Course credits, course difficulty index and course prerequisite weight of j-th semester, respectively
SCC_j	Sum of credits of j-th semester
SDC_j	The average difficulty of j-th semester
CL_h, CCL_h, DCL_h, $PWCL_h$	Set of all courses, course credits, course difficulty index and course prerequisite weight of h-th hierarchy level, respectively
nc_h	Number of courses in each hierarchy
c_m, cc_m, dc_m, pw_m	Course name, course credits, course difficulty index and course prerequisite weight of m-th course in h-th hierarchy, respectively
m	Counter variable for courses in each hierarchy: $m = 1, 2,\ldots, nc_h$
$p(c_m)$	Set of all prerequisites of c_m
CC_l	All possible course combinations using SSAP
CCC_l, CDC_l, CPW_l	Sum of course credits, average difficulty index and the sum of the prerequisite weight of CC_l
l	$l = 2^{nc_h}$
k	Minimum number of semester required for the completion of the degree program
mk	Maximum number of semesters
TC	Total Credits of all the courses
AD	The average difficulty of all the courses
CL	The credit limit for each semester
e	Error threshold

Algorithm 1. Algorithm for finding the Hierarchy Level of a course

```
def recLevelCheck(prereq):
    for cor in prereq:
        if(cor == “None”):
            val=0
```

```
        else: val = max(recLevelCheck(corz[cor]), l)
            val = val+1
    return val
    for i in corz.keys():
        if(i != "None"): print(i+"\t"+str(recLevelCheck(corz[i])))
```

As a result of Algorithm 1, the hierarchy level of each course hc_i is estimated as $\{hc_1, hc_2, \ldots, hc_n\}$ and the total number of hierarchies max_l in which the curriculum courses are divided into is estimated as specified in the next equation:

$$\text{max\_l} = \max\{hc_i\}, \tag{1}$$

max_l in (1) denotes the maximum number of levels the courses have with respect to their prerequisite dependencies. The course dependency structure has four levels as specified in Fig. 1 and hence the value of max_l is 4. Let CL_h be the set of all courses falling under level h with corresponding credits, difficulty, the prerequisite weight of the courses specified as CCL_h, DCL_h, $PWCL_h$ respectively. Hierarchy of each course as per algorithm 1 and the prerequisite weights are specified in Table 4.

The proposed Subset Sum Approximation Problem (SSAP) is an optimization problem in which the sum of elements of a subset is approximately equal to ss with an error quotient e as specified in (2). The proposed algorithms were tested for the value of e from ±0.1 till ±0.5 and were analyzed with the number of iterations, percentage of loss attained and the numbers of semesters the courses are distributed. Based on which, it is finalized with ±0.4,

$$\text{SSAP} = (S, \text{ss}), \text{there exists a subset } S' \subseteq S \text{ such that } \sum_{s \in S'} s = \text{ss} + e. \tag{2}$$

The objective of the proposed method is to minimize the loss function during course distribution that is calculated in terms of the Root Mean Square Error (RMSE) as in (3). A credit constraint is that the sum of credits of a semester should always be less than CL as in (4). AD is specified as the average difficulty of all the courses in the curriculum as specified in (5).

During course recommendation, this paper proposes SSAP in each semester. It is recommended with a set of courses with average difficulty constraint such that the average difficulty of a semester should be approximately equal to AD with an error quotient as specified in (6). TC is specified as the total credits of all the courses in the curriculum as in (7). A disjoint constraint states that the course distributed to a semester should not be distributed to any other semester as in (8) and the subset union constraint states that the union of all semester courses should be equal to the set of all courses in the curriculum as specified in (9).

Objective Function:

Minimize the loss function

$$\min Z = \frac{1}{N} \sum_{j=1}^{N} \left(\text{AD} - \text{SDC}_j\right)^2, \tag{3}$$

subject to the following constraints:

$$\text{SCC}_j = \sum_{cc_r \in SC_j} cc_i \leq \text{CL} \tag{4}$$

(Credit Constraint),

$$\text{AD} = \frac{1}{n} \sum dc_i \tag{5}$$

(Average Difficulty),

(6) $$\mathrm{SDC}_j = \frac{1}{\mathrm{nc}_j}\left(\sum_{\mathrm{dc}_r \in \mathrm{SD}_j} \mathrm{dc}_i\right) = \mathrm{AD} \pm e,$$

where e is the error threshold value set between –0.4 to +0.4 (Difficulty constraint),

(7) $$\sum \mathrm{SCC}_j = \mathrm{TC}$$

(Total Credit constraint),

(8) $$\{S_1\} \cap \{S_2\} \cap \ldots \cap \{S_N\} = \{\emptyset\}$$

(Disjoint set constraint),

(9) $$\{S_1\} \cup \{S_2\} \cup \ldots \cup \{S_N\} = \{C\}$$

(Subset union constraint).

Educational organizations have a curriculum of courses. Suppose, if its students have the liberty to choose (flexible course selection) their courses for each semester at the beginning of the program, they might not have a clear idea of how to choose the courses and in what sequence the courses have to be selected. To solve this problem the proposed method recommends the sequence of courses to be taken in each semester. This paper proposes two methods for solving the Subset Sum Approximation Algorithm (SSAP). The first method considers the parameter hierarchy level of each course for distributing the courses across the semesters and uses Maximum Prerequisite Weight (MPW) Algorithm and Difficulty Approximation (DA) Algorithm. The second method does not consider the hierarchical level of each course and uses Adaptive Genetic Algorithm (AGA). Each algorithm is represented in the sections below.

3.1. MPW Algorithm

During course distribution, this algorithm distributes a set of courses to each semester with approximately equal difficulty level by considering the course combinations with maximum prerequisite weight and maximum credits. This helps the students to register all possible prerequisite courses during the prior semesters with maximum credits within CL.

Algorithm 2. MPW Algorithm

Step 1. Let picked_courses[x] = {$\emptyset$}

Step 2. Initialize N=1, mk=12, j=1, h=1

Step 3. Sub divide C into CL_h based on hierarchy h

Step 4. Compute CL as CL = TC/max_l

Step 5. Compute AD as

$$\mathrm{AD} = \frac{1}{n}\left(\sum \mathrm{dc}_i\right)$$

Step 6. for h=1 ≤ mk loop

Step 7. $\forall\, c_m$ in $\mathrm{CL}_h \neq \{0\}$

Step 8. if $(p(c_m)$ not in picked_courses$[x])$then

a. Add c_m to CL_{h+1}

b. Delete c_m from CL_h

Step 9. else

compute all possible CC_l using SSAP such that

$$\mathrm{CCC}_l = \sum_{\mathrm{cc}_m \in \mathrm{CL}_h} \mathrm{cc}_m \leq \mathrm{CL}$$

$$\mathrm{CDC}_l = \frac{1}{\mathrm{nc}_j} \sum_{\mathrm{dc}_m \in \mathrm{CL}_h} \mathrm{dc}_m = \mathrm{AD} \pm e$$

$$\mathrm{CPW}_l = \sum_{\mathrm{pw}_m \in \mathrm{CL}_h} \mathrm{pw}_m$$

Step 10. for j=1 to $N \leq$ mk loop
Step 11. Select CC_l max(CPW_l)
a. if count(CC_l) == 1 then distribute it to semester S_j
b. else if count(CC_l) > 1
i. if count(CC_l) with max(CCC_l) == 1 then distribute it to semester S_j
ii. else if count (CC_l) with max(CCC_l) > 1 then
a. if count(CC_l) with ≈ CDC_l == 1, distribute it to S_j
b. else if count(CC_l) with ≈ CDC_l > 1, then select CC_l in random
Step 12. End for j
Step 13. Set the state of c_m in CL_h to 1 or 0 otherwise
Step 14. Add C_m with state 0 to CL_{h+1}
Step 15. Increment j, h
Step 16. N=j
Step 17. End for h

The MPW algorithm prioritizes course distribution based on the maximum prerequisite weight followed by maximum credits and the approximate course difficulty for the combination of courses, which are selected for a semester. Course combinations with high prerequisite weights are prerequisites for more number of courses. Courses with no prerequisite weight are not prerequisites for any courses and courses with less prerequisite weight are prerequisites for fewer courses. Since the post requisite courses are more difficult than prerequisite courses [34, 35], a better sequencing has to be recommended to balance the difficulty index throughout the degree program. MPW gives higher priority for prerequisite weight (pw_i) in which courses with more pw_i are distributed in the early semesters and less pw_i are distributed in the later semesters. This balances the difficulty of prerequisite courses along with the post requisite courses whose prerequisites are already distributed in the previous semesters.

3.2. Difficulty Approximation (DA) Algorithm

During course distribution, this algorithm distributes a set of courses to each semester with approximately equal difficulty level by considering the course combinations with approximate difficulty such that the loss value for the difficulty index is decreased. This helps the students to have a balanced workload throughout the semesters.

Algorithm 3. DA Algorithm
Step 1. Consider steps 1-9 from Algorithm 2
Step 2. for j=1 to $N \leq$ mk loop
Step 3. Select CC_l with ≈ CDC_l
a. if count(CC_l) == 1 then distribute it to semester S_j

b. else if count(CC_l) > 1
    i. if count(CC_l) with max(CPW_l) == 1 then distribute it to semester S_j
    ii. else if count (CC_l) with max(CPW_l) > 1 then
        a. if count(CC_l) with max(CCC_l) == 1, distribute it to S_j
        b. else if count(CC_l) with max(CCC_l) > 1, then select CC_l in random

Step 4. end for *j*
Step 5. Set the state of c_m in CL_h to 1 or 0 otherwise.
Step 6. Add C_m with state 0 to CL_{h+1}.
Step 7. Increment *j*, *h*
Step 8. *N=j*
Step 9. end for *h* till $CL_h = \{0\}$

The proposed DA algorithm prioritizes the difficulty index for selecting courses for a semester such that the average difficulty of the semester is approximately equal to AD so that the loss function is greatly reduced.

3.3. Adaptive Genetic Algorithm (AGA)

The proposed AGA has an initial set of the population with courses and their respective prerequisites for which a dependency matrix is constructed as per Fig. 1 and as specified in Table 2. In Table 2, the prerequisite(s) of a course is specified with 1. Disagreement is a function specified for the presence and absence of course prerequisites. A fitness function with a disagreement value is generated by adding all the prerequisite values set for a particular course. The courses with disagreement value as null as specified in Table 2 alone are considered for course distribution since those are the courses which don't have prerequisites.

MAT101, MAT106, CSE101, and MAT206 are the courses with disagreement value 0 as per Table 2. With these course combinations, the Adaptive Genetic Algorithm selects a set of courses with a sum of credits less than or equal to CL and the average difficulty approximately equal to AD. After the distribution of courses with disagreement value 0 to the initial semester in iteration 1, the course dependency structure is reconstructed as specified in Fig. 2 by removing the courses which are already distributed. Corresponding dependency matrix is specified in Table 3. Courses with the disagreement value 0 are again considered for further course distribution for further semesters.

Algorithm 4. Algorithm AGA
Step 1. Initial population with a set of courses, its prerequisites represented as a dependency matrix
Step 2. Set RandomPairs=0
Step 3. Maximum RandomPairs= max(default=10)
Step 4. Threshold Disagreement=0
Step 5. For each combination $pr_i \leq CL$
    a. Compute the sum of disagreement
    b. Disagreement=0
Step 6. End for

Step 7. Sort population in decreasing order with disagreement 0

Step 8. Sample population with $pr_i \leq CL$ and $SDC_j = AD \pm e$ use MPW or DA algorithm to select course combination for semester *j*

Step 9. Crossover population with evolutionary computing

Step 10. Update population by filtering the courses distributed to semester *j*

Step 11. For the filtered courses, draw a dependency matrix

Step 12. Repeat steps 1-11 till all the courses are distributed and no courses left.

Table 2. Dependency matrix for courses with prerequisites

CCODE	MAT101	MAT106	MAT105	MAT202	MAT203	CSE101	Disagreement
MAT101							0
MAT106							0
MAT105	1						
MAT202	1	1					
MAT203			1				
MAT207	1		1				
CSE220		1				1	
CSE101							0
CSE205				1			
CSE208		1					
CSE327				1			0
CSE418				1			
MAT206							0
MEE437					1		
Prereq Weight	3	3	2	3	1	1	

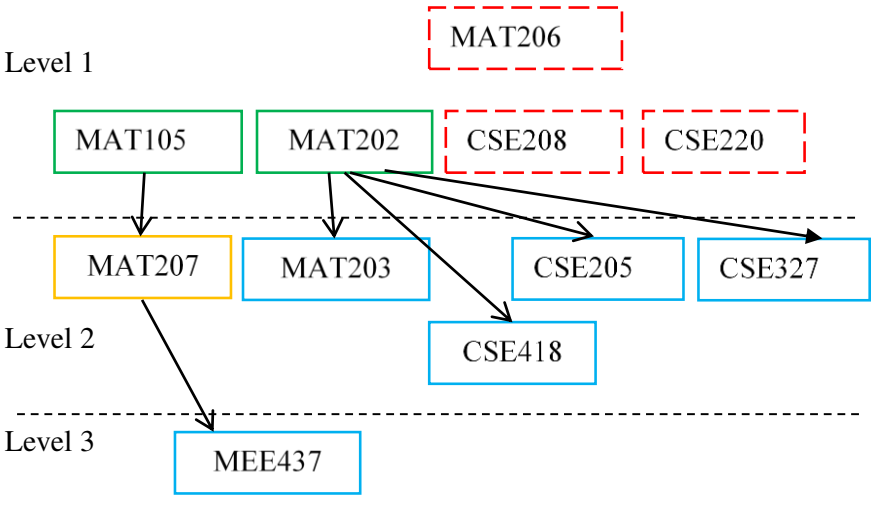

Fig. 2. Course Dependency Structure after iteration 1 of AGA

AGA can be used with either MPW or DA logic for the selection of courses for a semester based on the requirement of the student. For students with great CGPA, MPW works better and for students with less CGPA and backlogs can be recommended with the DA Algorithm.

The course dependency structure and the dependency matrix are reconstructed with further iterations until all the courses are distributed to the semesters and the course dependency structure and the corresponding dependency matrix is left with null courses.

Table 3. Dependency matrix for courses with prerequisites after iteration 1 of AGA

CCODE	MAT105	MAT202	MAT203	Disagreement
MAT105				0
MAT202				0
MAT203	1			
MAT207	1			
CSE220				0
CSE205		1		
CSE208				0
CSE327		1		
CSE418		1		
MAT206				0
MEE437			1	

4. Results and discussions

This section discusses the experimental analysis of SSAP algorithms, which use the curriculum courses of the undergraduate computer science and engineering from our university as the dataset. The sample data includes a set of all dependent courses of mathematics and set of all dependent courses of CSE101 with their course credit and course prerequisite(s). The parameters such as difficulty index, hierarchy level and pre-requisite weight of each course are also being considered for each course as specified in Table 4. Fig. 3 depicts the course dependency structure, which specifies the required prerequisite courses for the corresponding course that is to be completed. Considering these parameters, hierarchy estimation algorithm clusters the courses with respect to their prerequisite(s) and lists the courses in different hierarchy levels (level 1, 2, 3,…, etc.,) as specified in Fig. 4.

```
Course dependency Trees:
MAT101          ['MAT101']
MAT106          ['MAT106']
MAT105          ['MAT101', 'MAT105']
MAT202          ['MAT106', 'MAT101', 'MAT202']
MAT203          ['MAT101', 'MAT105', 'MAT203']
MAT207          ['MAT101', 'MAT105', 'MAT207']
CSE220          ['MAT106', 'CSE101', 'CSE220']
CSE101          ['CSE101']
CSE205          ['MAT106', 'MAT101', 'MAT202', 'CSE205']
CSE208          ['MAT106', 'CSE208']
CSE327          ['MAT106', 'MAT101', 'MAT202', 'CSE327']
CSE418          ['MAT106', 'MAT101', 'MAT202', 'CSE418']
MAT206          ['MAT206']
MEE437          ['MAT101', 'MAT105', 'MAT207', 'MEE437']
```

Fig. 3. Dependency course tree

Level 1 Courses offered: {'CSE101', 'MAT101', 'MAT206', 'MAT106'}
Level 2 Courses offered: {'CSE208', 'MAT202', 'CSE220', 'MAT105'}
Level 3 Courses offered: {'CSE418', 'CSE327', 'MAT207', 'CSE205', 'MAT203'}
Level 4 Courses offered: {'MEE437'}

Fig. 4. Course hierarchy level

Random distribution method distributes courses to each semester in random only by considering prerequisite and credit constraints. In MPW and DA, once the courses are sub grouped, the first level courses are selected for course distribution such that the course combinations satisfies the credit limit constraint as in (4) and also the difficulty approximation constraint as specified in (6). MPW and DA algorithms are quite similar with few variations in the course selection procedures. The Average Difficulty (AD) of all the courses is calculated as 4.07 which is considered as the expected Difficulty Index (DI). From the first level of courses, the two combinations such as combination1 {MAT101, MAT106, MAT206} and combination 2 {MAT101, MAT106, CSE101} are taken which yield the maximum sum of credits as 11, average difficulty as 4, 3.65, and prerequisite weight as 8, 9 respectively. The MPW chooses combination 2 since it selects the maximum credits with maximum prerequisite weights whereas; DA selects combination 1 since it has more approximate average difficulty. The MPW algorithm considers maximum credits initially and checks for maximum prerequisites whereas DA Algorithm considers course combination with lesser credits combination than maximum credit value which gives approximate average difficulty.

Table 5 specifies the list of courses recommended for each semester using Random Distribution (RD), MPW Algorithm and DA Algorithm and Table 6 specifies the corresponding total credits per semester and the average difficulty index per semester when RD, MPW, and DA methodologies were used. The analysis depicts that when MPW is used, each semester has the maximum credits and the final semester has least credits and when DA is used, all the semesters have approximately equal difficulty index.

MPW Algorithm gives priorities to maximize the credits while distributing the course to a semester. In the case of course combinations with the same credits, the course combinations with maximum prerequisite weights are chosen. If there are more combinations available for the same maximum prerequisite, the course combinations with approximate average difficulty are predicted SDC_j as specified in Table 6. DA Algorithm gives priority only for the approximation of the course difficulty of all the semesters. When the course combinations SDC_i are same, the next priority will be given to the prerequisite weight else the course combinations are distributed in random. Initially, course combination with maximum credits is taken into consideration. In case if the course combination has fewer SDC_j than the course combined with $\max(SC_j) - 1$, then the DA considers course combined with fewer credits with more approximate average difficulty. The loss value of the DA Algorithm will be comparatively less than MPW algorithm since the course distribution is purely based on approximate difficulty. The loss function for the DA Algorithm is calculated based on the expected AD and the predicted SDC_j as

specified in Table 6 and hence the objective function is attained with a minimum loss function.

Abbreviations:
NB – No Backlog
OB – One Backlog
TB – Two Backlogs
TCP – Total Credits per Semester
ADP – Average Difficulty Index per Semester

Table 4. Course list with hierarchy and prerequisite weight

Course code	Credits	Difficulty Index (DI)	Pre-requisite(s)	Hierarchy level	Prerequisite weight
MAT101	4	4	None	1	3
MAT106	4	4	None	1	5
MAT105	4	4	MAT101	2	2
MAT202	4	4	MAT101, MAT106	2	4
MAT203	3	4	MAT105	3	0
MAT207	4	4	MAT101, MAT105	3	1
CSE220	5	4	CSE101, MAT106	2	0
CSE101	3	3	None	1	1
CSE205	4	4	MAT202	3	0
CSE208	4	5	MAT106	2	0
CSE327	3	4	MAT202	3	0
CSE418	4	5	MAT202	3	0
MAT206	3	4	None	1	0
MEE437	3	4	MAT207	4	0

Table 5. Recommended course list with RD vs MPW and DA Algorithm

Semester	RD	MPW	DA
1	MAT101, MAT106, CSE101	MAT101, MAT106, CSE101	MAT101, MAT106, MAT206
2	MAT206, MAT105, MAT202	MAT105, MAT202, CSE220	MAT202, CSE208, CSE101
3	CSE220, CSE208, MAT203	MAT207, CSE205, CSE418	MAT207, MAT105, CSE220
4	MAT207, CSE327, CSE418	MAT203, CSE327, CSE208, MAT206	MAT203, CSE327, CSE418, MEE437
5	MEE437, CSE205	MEE437	CSE205

Table 6. Loss function for RD vs MPW and DA Algorithm

Semester	RD	MPW	DA	RD	MPW	DA	RD	MPW	DA
	Total Credits per semester			Total Difficulty per semester			Expected DI – Predicted DI per semester		
1	11	11	11	3.66	3.67	4	0.168	0.16	0.005
2	11	13	11	4	4	4	0.005	0.005	0.005
3	9	12	13	3.66	4.33	4	0.168	0.068	0.005
4	11	13	13	4.33	4.33	4	0.068	0.068	0.005
5	7	3	4	4	4	4	0.005	0.005	0.005
				Loss Function			0.049	0.037	0.002

Table 7. Credit comparison with and without backlogs

TCP	NB - RD	NB -MPW	NB -DA	OB-RD	OB-MPW	OB-DA	TB-RD	TB-MPW	TB-DA
Sem 1	11	11	11	11	11	11	11	11	11
Sem 2	11	13	11	11	13	11	10	11	11
Sem 3	9	12	13	13	11	13	13	13	13
Sem 4	11	13	13	11	12	11	11	12	11
Sem 5	7	3	4	10	9	10	11	12	11
Sem 6	0	0	0	0	0	0	3	0	3

Table 8. Difficulty index comparison with and without backlogs

ADP	NB-RD	NB-MPW	NB-DA	OB-RD	OB-MPW	OB-DA	TB-RD	TB-MPW	TB-DA
Sem 1	3.67	3.67	4	3.67	3.67	4	3.67	3.67	4
Sem 2	4	4	4	4	4.33	4	3.67	4	3.67
Sem 3	3.67	4.33	4	3.67	4	4	4.33	4	4
Sem 4	4.33	4.33	4	4.33	4.33	4	4	4.33	4
Sem 5	4	4	4	4	4	4.33	4.33	4	4.67
Sem 6							4		4

If a student is supposed he/she wishes to complete the entire course in the curriculum within the duration of the degree of study, MPW is used. When a student wishes to have a balanced workload with balanced credits and difficulty index, the course distribution with approximate difficulty index throughout the degree of study using the DA Algorithm is followed. The results of the RD, MPW and the DA algorithms are validated with the loss function as specified in (2).

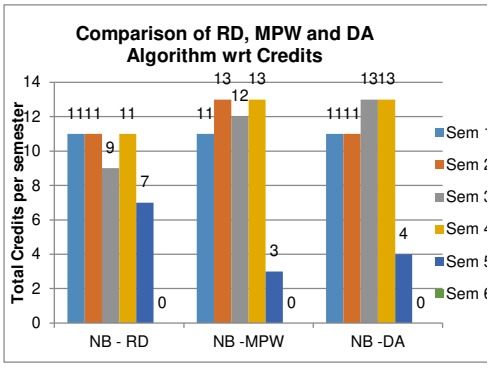

Fig. 5. Total credit per semester with no backlog

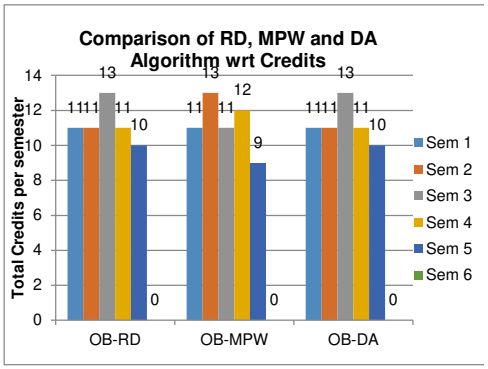

Fig. 6. Total credit per semester with one backlog

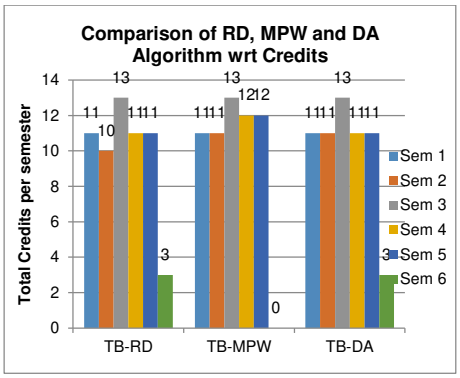

Fig. 7. Total credit per semester with two backlogs

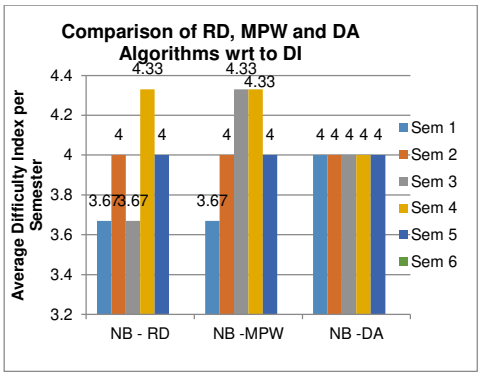

Fig. 8. Average difficulty index per semester with no backlog

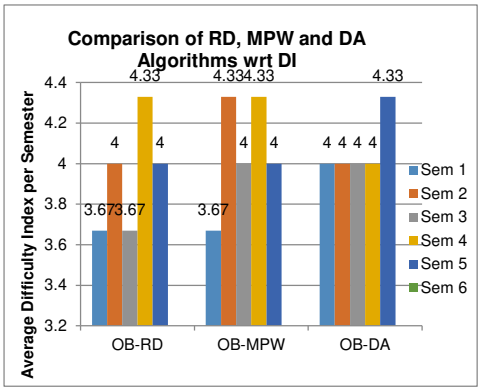

Fig. 9. Average difficulty index per semester with one backlog

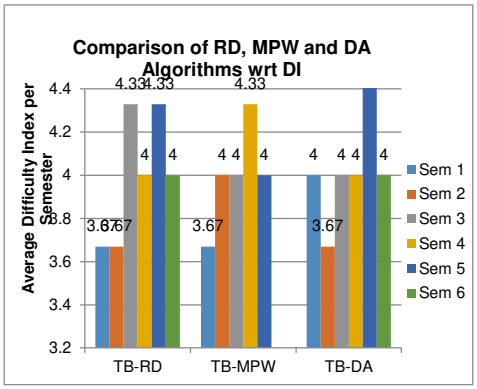

Fig. 10. Average difficulty index per semester with two backlogs

Accuracy with respect to loss function of DA Algorithm is more appropriate with 98% than the values during the usage of RD and MPW methodologies. All above scenarios work when the student completes each recommended course in all the semester without any deviation. From the recommended courses in Table 4, from semester 1, if a student fails in any of the semesters with 1 backlog or 2 backlogs, the scenario totally changes and this is analyzed with RD, MPW, and DA as specified in Table 7 and Table 8. The number of semesters *N* to which the courses are distributed depends on the students' pass/fail ratio. As per the example, the value of *N* is 5, i.e., the list of courses in Table 4 are distributed to 5 semesters. The value of *N* increases when a student fails again in the same course or different courses and the same is specified in Table 7 and 8. With no backlogs and one backlog, the courses were distributed to five semesters and when the backlog is increased to two, the courses were distributed to six semesters. If a student fails in a course, again and again, the course will be appended to the courses to be distributed and it automatically prolongs their duration of the degree. A minimum number of semesters required to distribute a course is max_l and maximum duration is MDD semesters. Our proposed algorithm stops distributing courses to a particular student until he passes all the required courses from the curriculum or when MDD is attained. Table 7 depicts the credit comparison and Table 8 depicts the difficulty index comparison among RD, MPW, and DA methodologies. When credits are considered, MPW algorithm works better as specified in Figs 5-7 and when the difficulty index is considered, the DA Algorithm works better as specified in Figs 8-10. By comparing the loss functions of MPW and DA Algorithms, DA Algorithm has given more accuracy with least loss value while distributing the courses to semesters with approximate difficulty. The adaptive genetic algorithm works better with DA Algorithm where the input to this algorithm does not consider the hierarchy level of a course but uses the disagreement function for filtering the courses for the combinations and a fitness function is used to validate the constraints specified in the objective function of the proposed methodology. For *n* courses, the complexity of SSP with brute force method is 2^n and since the proposed SSAP divides n courses into max_l subsets based on the hierarchy, the complexity is 2^{mal_l}.

5. Conclusion

In this paper, the factors affecting student's performance and the problems of the existing course sequence recommendation algorithms are studied and examined. Credit-based course selections with constraints are considered for recommending curriculum courses in sequence using the proposed MPW, DA and AGA algorithms. These personalized course recommendation algorithms are elaborated based on the individual's academic records which in turn helps the students to graduate on time with a balanced workload. These algorithms are analyzed and evaluated based on the loss function. The loss function is compared for the Random Distribution (RD) Algorithm, Maximum Prerequisite Weight (MPW) Algorithm and Difficulty Approximation (DA) Algorithm. Hence, every semester is distributed with a balanced workload with approximately equal difficulty index using the DA Algorithm, which gives minimum loss function compared with MPW Algorithm. Assuming the difficulty index for the courses is a limitation of the proposed method. Since the course difficulty index estimation relies on factors like course content difficulty, the difficulty of the question paper, feedback of the instructors/students, the same is adapted for our future research. Moreover, the proposed algorithm works the same for assumptions or adaptions of the existing course difficulty index methodologies

References

1. U. G. C. Guidelines on Adaptation of the Choice Based Credit System. University Grants Commission Bahadurshah Zafar Marg, New Delhi, 110 002.
2. Hasan, M., M. Parvez. Choice-Based Credit System in India: Pros and Cons. – Journal of Education and Practice, Vol. **6**, 2015, No 25, pp. 30-33.
3. **http://www.vit.ac.in/academics/ffcs**
4. Zafar, S., B. Manjurekar, N. P. Kumar, Z. A. Khan. Effects of FFCS (Fully Flexible Credit System) on Learning Experience and Academic Performance. – Procedia-Social and Behavioral Sciences, Vol. **143**, 2014, pp. 4-7.
5. Xu, J., T. Xing, M. Van der Schaar. Personalized Course Sequence Recommendations. – IEEE Transactions on Signal Processing, Vol. **64**, October 2016, No 20, pp. 5340-5352.
6. Paura, L., I. Arhipova. Student Dropout Rate in Engineering Education Study Program. – In: Proc. of 15th International Scientific Conference Engineering for Rural Development, Jelgava, Latvia, May 2016, pp. 641-646.
7. Szafran, R. F. The Effect of Academic Load on Success for New College Students: Is Lighter Better? – Research in Higher Education, Vol. **42**, 2001, No 1, pp. 27-50.
8. Kori, K., M. Pedaste, H. Altin, E. Tõnisson, T. Palts. Factors that Influence Students' Motivation to Start and to Continue Studying Information Technology in Estonia. – IEEE Transactions on Education, Vol. **59**, 2016, No 4, pp. 255-262.
9. Mundfrom, D. J. Estimating Course Difficulty. Ph.D. Dissertation, Statistics, Iowa State Univ., Ames, USA, 1991.
10. Bassiri, D., E. M. Schulz. Constructing a Universal Scale of High School Course Difficulty. – Journal of Educational Measurement, Vol. **40**, 2003, No 2, pp. 147-161.
11. Banerjee, S., N. J. Rao, C. Ramanathan. Rubrics for Assessment Item Difficulty in Engineering Courses. – In: Proc. of Frontiers in Education Conference (FIE), IEEE, 2015, pp. 1-8.
12. Kaur, K., K. Kaur. Analyzing the Effect of Difficulty Level of a Course on Students Performance Prediction Using Data Mining. – In: Proc. of 1st International Conference on Next Generation Computing Technologies (NGCT), IEEE, September 2015, pp. 756-761.

13. L i u, J., S. S h a, Q. Z h e n g, L. C h e n. Ranking Difficulty of Knowledge Units Based on Learning Dependency. – In: Proc. of 7th International Conference on e-Business Engineering, IEEE, November 2010, pp. 77-82.
14. S a f a v i, S. A., K. A. B a k a r, R. A. T a r m i z i, N. H. A l w i. What Do Higher Education Instructors Consider Useful Regarding Student Ratings of Instruction? Limitations and Recommendations. – Procedia-Social and Behavioral Sciences, Vol. **31**, 2012, pp. 653-657.
15. C o r e l l i, A. Direct Vs. Anonymous Feedback: Teacher Behavior in Higher Education, with Focus on Technology Advances. – Procedia-Social and Behavioral Sciences, Vol. **195**, 2015, pp. 52-61.
16. Z a i n u d i n, S., K. A h m a d, N. M. A l i, N. F. A. Z a i n a l. Determining Course Outcomes Achievement through Examination Difficulty Index Measurement. – Procedia-Social and Behavioral Sciences, Vol. **59**, 2012, pp. 270-276.
17. S w a r t, A. J. Evaluation of Final Examination Papers in Engineering: A Case Study Using Bloom's Taxonomy. – IEEE Transactions on Education, Vol. **53**, 2010, No 2, pp. 257-264.
18. Y a n g, F., F. W. L i, R. W. L a u. A Fine-Grained Outcome-Based Learning Path Model. – IEEE Transactions on Systems, Man, and Cybernetics: Systems, Vol. **44**, 2014, No 2, pp. 235-245.
19. P u m p u a n g, P., A. S r i v i h o k, P. P r a n e e t p o l g r a n g, S. N u m p r a s e r t c h a i. Using Bayesian Network for Planning Course Registration Model for Undergraduate Students. – In: Proc. of 2nd IEEE International Conference on Digital Ecosystems and Technologies, IEEE, February 2008, pp. 492-496.
20. P u m p u a n g, P., A. S r i v i h o k, P. P r a n e e t p o l g r a n g. Comparisons of Classifier Algorithms: Bayesian Network, C4. 5, Decision Forest and NBTree for Course Registration Planning Model of Undergraduate Students. – In: Proc. of IEEE International Conference on Systems, Man and Cybernetics, IEEE, October 2008, pp. 3647-3651.
21. W a n g, X., F. Y u a n. Course Recommendation by Improving BM25 to Identity Students' Different Levels of Interests in Courses. – In: Proc. of 2009 International Conference on New Trends in Information and Service Science, IEEE, June 2009, pp. 1372-1377.
22. G a r r i d o, A., L. M o r a l e s. e-Learning and Intelligent Planning: Improving Content Personalization. – IEEE Revista Iberoamericana de Tecnologias del Aprendizaje, Vol. **9**, 2014, No 1, pp. 1-7.
23. P a r a m e s w a r a n, A. G., H. G a r c i a-M o l i n a, J. D. U l l m a n. Evaluating, Combining and Generalizing Recommendations with Prerequisites. – In: Proc. of 19th ACM International Conference on Information and Knowledge Management, ACM, October 2010, pp. 919-928.
24. P a r a m e s w a r a n, A., P. V e n e t i s, H. G a r c i a-M o l i n a. Recommendation Systems with Complex Constraints: A Course Recommendation Perspective. – ACM Transactions on Information Systems (TOIS), Vol. **29**, 2011, No 4, Art. No 20.
25. B e t a n c u r, L., B. M. R o t t m a n, E. V o t r u b a-D r z a l, C. S c h u n n. Analytical Assessment of Course Sequencing: The Case of Methodological Courses in Psychology. – Journal of Educational Psychology, Vol. **111**, 2019, No 1, pp. 91-103.
26. C h e n, C. M., C. Y. L i u, M. H. C h a n g. Personalized Curriculum Sequencing Utilizing Modified Item Response Theory for Web-Based Instruction. – Expert Systems with Applications, Vol. **30**, 2006, No 2, pp. 378-396.
27. B r i d g e s, C., J. J a r e d, J. W e i s s m a n n, A. M o n t a n e z-G a r a y, J. S p e n c e r, C. G. B r i n t o n. Course Recommendation as Graphical Analysis. – In: Proc. of 52nd Annual Conference on Information Sciences and Systems (CISS), IEEE, March 2018, pp. 1-6.
28. M o r r o w, T., A. R. H u r s o n, S. S. S a r v e s t a n i. A Multi-Stage Approach to Personalized Course Selection and Scheduling. – In: Proc. of 2017 IEEE International Conference on Information Reuse and Integration (IRI), San Diego, CA, 2017, pp. 253-262.
29. C u c u r i n g u, M., C. Z. M a r s h a k, D. M o n t a g, P. R o m b a c h. Rank Aggregation for Course Sequence Discovery. – In: Proc. of International Workshop on Complex Networks and Their Applications, Springer, Cham., November 2017, pp. 139-150.
30. S e g a l, A., Y. B. D a v i d, J. J. W i l l i a m s, K. G a l, Y. S h a l o m. Combining Difficulty Ranking with Multi-Armed Bandits to Sequence Educational Content. – In: Proc. of International Conference on Artificial Intelligence in Education, Springer, Cham., June 2018, pp. 317-321.

31. G u n j i, A. B., B. B. B. V. L. D e e p a k, C. R. B a h u b a l e n d r u n i, D. B. B. B i s w a l. An Optimal Robotic Assembly Sequence Planning by Assembly Subsets Detection Method Using Teaching Learning-Based Optimization Algorithm. – IEEE Transactions on Automation Science and Engineering, Vol. **15**, 2018, No 3, pp. 1369-1385.
32. C a p r a r a, A., H. K e l l e r e r, U. P f e r s c h y. The Multiple Subset Sum Problem. – SIAM Journal on Optimization, Vol. **11**, 2000, No 2, pp. 308-319.
33. C a p r a r a, A., H. K e l l e r e r, U. P f e r s c h y. A PTAS for the Multiple Subset Sum Problem with Different Knapsack Capacities. – Information Processing Letters, Vol. **73**, 2000, No 3-4, pp. 111-118.
34. W i s n e s k i, J. E., G. O z o g u l, B. A. B i c h e l m e y e r. Investigating the Impact of Learning Environments on Undergraduate Students' Academic Performance in a Prerequisite and Post-Requisite Course Sequence. – The Internet and Higher Education, Vol. **32**, 2017, pp. 1-10.
35. A d j e i, S. A., A. F. B o t e l h o, N. T. H e f f e r n a n. Predicting Student Performance on Post-Requisite Skills Using Prerequisite Skill Data: An Alternative Method for Refining Prerequisite Skill Structures. – In: Proc. of 6th International Conference on Learning Analytics & Knowledge, April 2016, ACM, pp. 469-473.

Received: 11.04.2019; Second Version: 10.08.2019; Accepted: 22.08.2019