Personalized Education; Solving a Group Formation and Scheduling Problem for Educational Content

Sanaz Bahargam, Dóra Erdős, Azer Bestavros, Evimaria Terzi
Computer Science Department, Boston University, Boston MA
[bahargam, edori, best, evimaria]@cs.bu.edu

ABSTRACT

Whether teaching in a classroom or a Massive Online Open Course it is crucial to present the material in a way that benefits the audience as a whole. We identify two important tasks to solve towards this objective: (1.) group students so that they can maximally benefit from peer interaction and (2.) find an optimal schedule of the educational material for each group. Thus, in this paper we solve the problem of team formation and content scheduling for education. Given a time frame d, a set of students S with their required need to learn different activities T and given k as the number of desired groups, we study the problem of finding k group of students. The goal is to teach students within time frame d such that their potential for learning is maximized and find the best schedule for each group. We show this problem to be NP-hard and develop a polynomial algorithm for it. We show our algorithm to be effective both on synthetic as well as a real data set. For our experiments we use real data on students’ grades in a Computer Science department. As part of our contribution we release a semi-synthetic dataset that mimics the properties of the real data.

Keywords
Team Formation; Clustering; Partitioning; Teams; MOOC

1. INTRODUCTION

Many work has been dedicated on how to improve students’ learning outcome. We recognize two substantial conclusions; first, the use of personalized education. By shaping the content and delivery of the lessons to the individual ability and need of each student we can enhance their performance([6, 11, 12]. Second, grouping students; working in teams with their peers helps students to access the material from a different viewpoint as well [7, 4, 13, 1]. In this paper we study the problem of creating personalized educational material for teams of students by taking a computational perspective. To the best of our knowledge we are the first to formally define and study the two problems of team formation and personalized scheduling for teams in the context of education. We present a formal definition for these problems, study their computational complexity and design algorithms for solving them. In addition, we also apply our algorithms to a real dataset obtained from real students. We make our semi-synthetic dataset BUSSynthetic generated to faithfully mimic the real student data available on our website.

Related Work: Besides the work on improving students learning outcome, related problems have also been studied in computer science. Topics of interest are team formation [2, 3, 9, 10] and scheduling theory, see [5] for an overview.

2. PRELIMINARIES

We model a student’s learning process by a sequence of topics that she learns about. In this sequence topics may appear multiple times, and repetitions of a topic may count with different weights towards the overall benefit of the student. Let S = {s1, s2, . . . , sn} be a set of students and T = {t1, t2, . . . , tm} be a set of topics. We assign topics to d timeslots, a schedule A is a collision free assignment of topics to the timeslots. A can be thought of as an ordered list of (possible multiple occurrences) of the topics. For a topic t ∈ T the tuple ⟨t, i⟩ denotes the ith occurrence of t in a schedule. The notation A[r] = ⟨t, i⟩ refers to the tuple ⟨t, i⟩ that is assigned to timeslot r in A.

For student s ∈ S and topic t ∈ T the requirement req(s, t) is an integer depicting the number of times s needs to learn about t to master its content. We assume that for the first req(s, t) repetitions of t there is some benefit to s from every repetition of t, but for any further repetition there is no additional benefit to s. We call b(s, ⟨t, i⟩) (Equation (1)) the benefit of s from hearing about t for the ith time.

\[ b(s, ⟨t, i⟩) = \begin{cases} \frac{1}{\text{req}(s, t)} & \text{if } i \leq \text{req}(s, t) \\ 0 & \text{otherwise} \end{cases} \]  (1)

Note that for ease of exposition, we assume that all repetitions of t before req(s, t) carry equal benefit to s. However, the definition and all of our later algorithms could easily be extended to use some other function b′(s, ⟨t, i⟩).

Given the benefits \( b(s, ⟨t, i⟩) \) there is a natural extension to define the benefit \( B(s, A) \) that s gains from schedule A. This benefit is simply a summation over all timeslots in A,

\[ B(s, A) = \sum_{r=1}^{d} b(s, A[r]) \]  (2)
3. THE GROUP SCHEDULE PROBLEM

Given a group of students \( P \subseteq S \) our first task is to find an optimal schedule for \( P \). That is, find a schedule to maximize the group benefit \( B(P, A) \) that group \( P \) has from \( A \) (Equation (3)).

\[
B(P, A) = \sum_{s \in P} \sum_{r=1}^{d} b(s, A[r]) \tag{3}
\]

We call this the GROUP SCHEDULE problem (problem 1).

Problem 1 (Group Schedule). Let \( P \subseteq S \) be a group of students and \( T \) be a set of topics. For every \( s \in S \) and \( t \in T \) let \( req(s, t) \) be the requirement of \( s \) on \( t \) given for every student-topic pair. Find a schedule \( A_P \), such that \( B(P, A_P) \) is maximized for a deadline \( d \).

The Schedule algorithm. We first give a simple polynomial time algorithm, Schedule\( (P, d) \) (Algorithm 1), to solve problem 1. Schedule is a greedy algorithm that assigns to every timeslot an instance of the topic with the largest marginal benefit. We say that the signs to every timeslot an instance of the topic with the increase in the group benefit if \( t,i \rightarrow P \) (Equation (3)).

\[
\text{Benefit of topic } t \text{ in } P \text{ is } B(P, t) = \frac{\sum_{s \in P} max\{B(s, t) - B(s, t) - u[s, t], 0\}}{|P|}; t \in T \text{ and } P \subseteq S \tag{4}
\]

The output is a schedule \( A_P \) with maximum marginal benefit \( B(P, d) \).

Algorithm 1 Schedule algorithm for computing an optimal schedule \( A_P \) for a group \( P \).

Input: requirements \( req(s, t) \) for every \( s \in P \) and every topic \( t \in T \), deadline \( d \).

Output: schedule \( A_P \).

1: \( A_P \leftarrow [ ] \)
2: \( B \leftarrow [m(P, (t, 1))] \text{ for } t \in T \)
3: \( R \leftarrow [0] \text{ for all } t \in T \)
4: while \( |A_P| < d \) do
5: \( \text{Find topic } u_t \text{ with maximum marginal benefit in } B \)
6: \( A_P \leftarrow (u_t, R[u_t]) \)
7: \( R[u_t] \leftarrow +\)
8: \( \text{Update } B[u_t] \text{ to } m(P, (t, R[u_t])) \)
9: end while

4. THE COHORT SELECTION PROBLEM

The next natural question is, that given a certain teaching capacity \( K \) (i.e., there are \( K \) teachers or \( K \) classrooms available), how to divide students into \( K \) groups so that each student benefits the most possible from this arrangement. At a high level we solve an instance of a partition problem; find a \( K \)-part partition \( P = P_1 \cup P_2 \cup \ldots \cup P_K \) of students into groups, so that the sum of the group benefits over all groups is maximized. This is the Cohort Selection Problem.

Problem 2 (Cohort Selection). Let \( S \) be a set of students and \( T \) be a set of topics. For every \( s \in S \) and \( t \in T \) let \( req(s, t) \) be the requirement of \( s \) on \( t \) that is given. Find a partition \( P \) of students into \( K \) groups, such that

\[
B(P, d) = \sum_{P \in P} B(P, A_P) \tag{5}
\]

is maximized, where \( A_P = \text{Schedule}(P, d) \) for every group.

The Cohort Selection (Problem 2) is NP-hard as the Catalog Segmentation problem [8] can be reduced to it.

4.1 Partition algorithms.

In this section we introduce CohPart (Algorithm 3) as our solution to the COHORT SELECTION problem. The input to Algorithm 3 are the requirements \( \text{req}(s, t) \), number of groups \( K \) and length of the schedule \( d \). The output is a partition \( P = \{P_1, P_2, \ldots, P_K\} \) of the students and corresponding schedules \( \{A_1, A_2, \ldots, A_K\} \) for each group.

CohPart first assigns every student to one of the groups in \( P \) at random (Line 3) and an initial optimal schedule for every group is computed (Line 5). Then in every iteration of the algorithm first every student is assigned to the group with the highest benefit schedule for the student (Line 9) and then the group schedules are recomputed (Line 12). The runtime of each iteration is \( O(k|S||T|) \). In our experiments we observed that our algorithm converges really fast, less than a few tens of iterations.

Algorithm 2 Benefit algorithm to compute the benefit for student \( s \) from schedule \( A \)

Input: requirements \( \text{req}(s, t) \) for a student \( s \in P \) and every topic \( t \in T \) and a single schedule \( A \)

Output: \( \text{Benefit}(s, A) \) Benefit of \( s \) from schedule \( A \).

1: \( \text{Benefit}(s, A) = 0 \)
2: for all topics \( t \in T \) do
3: \( \text{Benefit}(s, A) = \text{Benefit}(s, A) + \frac{\min\{\text{req}(s, t), A[t]\}}{|A[t]|} \)
4: end for

5. EXPERIMENTS

The goal of these experiments is to gain an understanding of how our clustering algorithm works in terms of performance (objective function) and runtime. Furthermore, we want to understand how the deadline parameter impacts our algorithm. We used a real world dataset, semi synthetic and synthetic datasets. The semi synthetic dataset and the source code to generate it are available in our website. We first explain different datasets and then show how well our algorithm is doing on each dataset.

5.1 Algorithms

We compare CohPart to two baseline algorithms.

\begin{center}
\footnotesize
1http://cs-people.bu.edu/bahargam/edm/
\end{center}
We compute the optimal group schedules using K_means clustering performed on the space of the requirement vectors using Euclidean distance.

CohPart_S: We also investigate a speedup version of CohPart. We pick a subset of $n' << n$ students $S' \subset S$ at random. We compute the optimal group schedules $A'_1, A'_2, \ldots, A'_K$ for $S'$ using CohPart and then assign each student $s \in S'$ to the group that maximizes $\text{Benefit}(s, A'_i)$.

5.2 Datasets

BUCS data. This dataset consists of grades of real students who majored in CS at Boston University. The data consists of 398 students and 41 courses. Here the courses correspond to topics and letter grades were converted to the requirements of each topic. We assign students to groups based on the K_means clustering performed on the space of the requirement vectors using Euclidean distance.

BUCSSynthetic data. In order to see how well our algorithm scales to larger datasets, we generated a synthetic data, based on the obtained parameters from GRM. We call this dataset BUCSSynthetic. From BUCS dataset, we observed that the ability of students follows a normal distribution with $\mu = 1.13$ and $\sigma = 1.41$. Applying GRM to BUCS, we obtained difficulty parameters for 41 courses. In order to obtain difficulties for 100 courses, we used the following:

1. Choose one of the 41 courses at random.
2. Use density estimation, smoothing and then get the CDF of the difficulties.
3. Randomly sample from the CDF to get the difficulties for a new course.

Using these parameters, we generated grades for 2000 students and 100 courses and we transformed grades to number of requirements similar to what we did for BUCS dataset.

5.3 Results

5.3.1 Results on Real World Datasets

BUCS. The result on the BUCS data is depicted in Figure 1e where each point shows the benefit of all students when partitioning them into $K$ groups. As we see the RandPart has the lowest benefit and our algorithm has the best benefit. As the number of clusters increases (having hence fewer students in each cluster), the benefit also increases, means the schedule for those students is more personalized and closer to their individual schedule. In Figure 1f we show that the greater the deadline is, the closer $\text{Benefit}(s, A'_i)$ is to the ideal (as explained earlier) to 4 in all experiments.

BUCSSBase. We tried different values for base and step parameters (explained earlier) and the result is depicted in Figure 1g when the base and step are equal to 1. The larger is the value of base and step parameter, the better our algorithm performs.

BUCSSSynthetic dataset. We ran our algorithms on on BUCSSynthetic dataset to see how well our algorithm scales for large number of students. The result is depicted in Figure 1h.

5.3.2 Results on Synthetic Datasets

The result on synthetic data is illustrated in Figure 1a. As we see CohPart and CohPart_S both are performing well. For all of the courses the mean requirement is close to 10 with standard deviation 3. We expect that students in the same
group (when generating the data) should be placed in the same cluster after running our algorithm and the schedule should include the selected courses in each group. Students have different requirement values for the selected courses in each group, but the sum of these selected courses is equal to the deadline and our algorithm realized this structure and only considered these selected courses to obtain the schedule. But K\_means lacked this ability to find the hidden structure. The next studied datasets were uniform, pareto and normal datasets and the results are depicted in Figure 1b, 1c and 1d respectively. For these datasets also our algorithm outperformed K\_means and RandPart.

6. CONCLUSION
In this paper, we highlighted the importance of team formation and scheduling educational materials for students. We suggested a novel clustering algorithm to form different teams and teach the team members based on their abilities. The results we obtained shows that our proposed solution is effective and suggest that we have to consider personalized teaching for students and form more efficient teams.

7. ACKNOWLEDGMENTS
This work was partially supported by NSF Grants: #1430145, #1414119, #1347522, #1239021, #1012798, #1218437, #1253393, #1320542, #1421759.

8. REFERENCES