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

Wether 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 $\mathbf{S}$ with their required need to learn different activities $\mathbf{T}$ 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 `BUCSSynth`, 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 $\mathbf{S} = \{s_1, s_2, \ldots, s_n\}$ be a set of students and $\mathbf{T} = \{t_1, t_2, \ldots, t_m\}$ be a set of topics. We assign topics to d timeslots,a *schedule* $\mathcal{A}$ is a collision free assignment of topics to the timeslots. $\mathcal{A}$ can be thought of as an ordered list of (possible multiple occurrences) of the topics. For a topic $t \in \mathbf{T}$ the tuple $\langle t, i\rangle$ denotes the i^{th} occurrence of t in a schedule. The notation $\mathcal{A}[r] = \langle t, i\rangle$ refers to the tuple $\langle t, i\rangle$ that is assigned to timeslot r in $\mathcal{A}$.

For student $s \in \mathbf{S}$ and topic $t \in \mathbf{T}$ the *requirement* $\mathtt{req}(s,t)$ is an integer depicting the number of times s needs to learn about t to master its content. We assume that for the first $\mathtt{req}(s,t)$ 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 $\mathbf{b}(s, \langle t, i\rangle)$ (Equation (1)) the *benefit* of s from hearing about t for the i^{th} time.

$$\mathbf{b}(s, \langle t, i\rangle) = \begin{cases} \frac{1}{\mathtt{req}(s.t)} & \text{if } i \leq \mathtt{req}(s,t) \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

Note that for ease of exposition, we assume that all repetitions of t before $\mathtt{req}(s,t)$ carry equal benefit to s. However, the definition and all of our later algorithms could easily be extended to use some other function $\mathbf{b}'(s, \langle t, i\rangle)$.

Given the benefits $\mathbf{b}(s, \langle t, i\rangle)$ there is a natural extension to define the benefit $\mathbf{B}(s, \mathcal{A})$ that s gains from schedule $\mathcal{A}$. This benefit is simply a summation over all timeslots in $\mathcal{A}$,

$$\mathbf{B}(s, \mathcal{A}) = \sum_{r=1}^{d} \mathbf{b}(s, \mathcal{A}[r]) \tag{2}$$

3. THE GROUP SCHEDULE PROBLEM

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

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

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

PROBLEM 1 (GROUP SCHEDULE). *Let $P \subseteq \mathbf{S}$ be a group of students and $\mathbf{T}$ be a set of topics. For every $s \in \mathbf{S}$ and $t \in \mathbf{T}$ let $\texttt{req}(s,t)$ be the requirement of s on t given for every student-topic pair. Find a schedule $\mathcal{A}_P$, such that $\mathbf{B}(P, \mathcal{A}_P)$ is maximized for a deadline d.*

The `Schedule` algorithm. We first give a simple polynomial time algorithm, $\texttt{Schedule}(P, d)$ (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 *marginal benefit*, $\mathbf{m}(P, \langle t, i \rangle)$, from the i^{th} repetition of t (thus $\langle t, i \rangle$) to P is the increase in the group benefit if $\langle t, i \rangle$ is added to $\mathcal{A}$. The marginal benefit can be computed as the sum of benefits over all students in P as given in Equation (4).

$$\mathbf{m}(P, \langle t, i \rangle) = \sum_{s \in P} \mathbf{b}(s, \langle t, i \rangle) \tag{4}$$

The `Schedule` algorithm is an iterative algorithm with d iterations that in every iteration appends a topic to the schedule $\mathcal{A}_P$. We maintain an array B in which values are marginal benefit of topics t, and an array R that contains a counter for every topic in $\mathcal{A}_P$. In every iteration `Schedule` selects the topic u_t with the largest marginal benefit from B and adds it to $\mathcal{A}_P$ (Lines 5 and 6) . Then it updates marginal benefit of u_t, $B[u_t]$ (Lines 7- 8). It is easy to see that Algorithm 1 yields an optimal schedule for a group P and runs in $O(d(|P| + log|\mathbf{T}|))$.

Algorithm 1 `Schedule` algorithm for computing an optimal schedule $\mathcal{A}_P$ for a group P.

Input: requirements $\texttt{req}(s,t)$ for every $s \in P$ and every topic $t \in \mathbf{T}$, deadline d.
Output: schedule $\mathcal{A}_P$.

1: $\mathcal{A}_P \leftarrow [\ ]$
2: $B \leftarrow [\mathbf{m}(P, \langle t, 1 \rangle)]$ for $t \in \mathbf{T}$
3: $R \leftarrow [0]$ for all $t \in \mathbf{T}$
4: **while** $|\mathcal{A}_P| < d$ **do**
5:     Find topic u_t with maximum marginal benefit in B
6:     $\mathcal{A}_P \leftarrow \langle u_t, R[u_t] \rangle$
7:     $R[u_t] + +$
8:     Update $B[u_t]$ to $\mathbf{m}(P, \langle t, R[u_t] \rangle)$
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 $\mathcal{P} = P_1 \cup^* P_2 \cup^* \ldots \cup^* P_K$ 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 $\mathbf{S}$ be a set of students and $\mathbf{T}$ be a set of topics. For every $s \in \mathbf{S}$ and $t \in \mathbf{T}$ let $\texttt{req}(s,t)$ be the requirement of s on t that is given. Find a partition $\mathcal{P}$ of students into K groups, such that*

$$\mathbf{B}(\mathcal{P}, d) = \sum_{P \in \mathcal{P}} \mathbf{B}(P, \mathcal{A}_P) \tag{5}$$

is maximized, where $\mathcal{A}_P = \texttt{Schedule}(P, d)$ 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 $\texttt{req}(s,t)$, number of groups K and length of the schedule d. The output is a partition $\mathcal{P} = \{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_K\}$ of the students and corresponding schedules $\{\mathcal{A}_1, \mathcal{A}_2, \ldots \mathcal{A}_K\}$ for each group.

`CohPart` first assigns every student to one of the groups in $\mathcal{P}$ 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 $O(k|\mathbf{S}||\mathbf{T}|)$. 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 $\mathcal{A}$

Input: requirements $\texttt{req}(s,t)$ for a student $s \in P$ and every topic $t \in \mathbf{T}$ and a single schedule $\mathcal{A}$
Output: `Benefit`(s,$\mathcal{A}$) Benefit of s from schedule $\mathcal{A}$.

$\texttt{Benefit}(s, \mathcal{A}) = 0$
for all topics $t \in \mathbf{T}$ **do**
    $\texttt{Benefit}(s, \mathcal{A}) = \texttt{Benefit}(s, \mathcal{A}) + \frac{min(req(s,t), \mathcal{A}[t])}{\mathcal{A}[t]}$
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. [1] 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.

[1] `http://cs-people.bu.edu/bahargam/edm/`

Algorithm 3 `CohPart` for computing the partition $\mathcal{P}$ based on the benefit of students from schedules.

Input: requirement $\texttt{req}(s,t)$ for every $s \in \mathbf{S}$ and $t \in \mathbf{T}$, number of timeslots d, number of groups K.
Output: partition $\mathcal{P}$.

$\mathcal{A} = \{\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_K\}$
$\mathcal{P} = \{P_1, P_2, \ldots, P_K\}$
$i \in_R [1, 2, \ldots, K]$, $\mathcal{P}_i \leftarrow s$ for every $s \in S$
for $i = 1, \ldots, K$ **do**
  $\mathcal{A}_i = \texttt{Schedule}(P_i, d)$
end for
while convergence is achieved **do**
  for all students $s \in \mathbf{S}$ **do**
    $P_i \leftarrow s$, $i = \text{argmax}_{j=1,\ldots,k} \texttt{Benefit}(s, \mathcal{A}_j)$
  end for
  for $i = 1, \ldots, K$ **do**
    $\mathcal{A}_i = \texttt{Schedule}(P_i, d)$
  end for
end while

`RandPart`: Partition S at random.

`K_means`: We represent each student s by the $|T|$-dimensional vector $\left(\texttt{req}(s,t_1), \texttt{req}(s,t_2), \ldots, \texttt{req}(s,t_{|T|})\right)$ containing its requirements for each topic. We assign students to groups based on the `K_means` clustering performed on the space of the requirement vectors using Eucledian 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 $\mathcal{A}'_1, \mathcal{A}'_2, \ldots \mathcal{A}'_K$ for S' using `CohPart` and then assign each student $s \in S$ to the group that maximizes $\texttt{Benefit}(s, \mathcal{A}'_i)$.

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 requirement of students. That is, grades A – F were converted to $\texttt{req}(s,t)$ such that A = 5 and F = 50. We assumed the number of requirement to master a course for the smartest student is 5 (base parameter). As the ability drops, number of requirement goes up (step parameter). To compute missing requiements, i.e., fill values for missing (student, course) pairs, we used Graded Response Model (GRM). First, using GRM we obtain the ability and difficulty parameters for all students and all courses. Then for each pair of (student, course) in which student s did not take course c, we used the ability of s and difficulty of c to predict the grade of course c for that student.

BUCSSynth 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 BUCSSynth. 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.

Synthetic data. In ground truth dataset we had generated 10 groups of students, each group containing 40 students. For each group we selected 5 courses and assigned requirement randomly to those 5 courses such that the sum of requirement will be equal to the deadline. Then for the remaining 35 courses, we filled number of requirements with random numbers taken from a normal distribution with $\mu = \frac{deadline}{5}$ and $\sigma = 3$. We refer to this dataset as GroundTruth.

We have also generated the requirements for 400 students and 40 courses using Pareto ($\alpha = 2$), Normal ($\mu = 30$ and $\sigma = 5$) and Uniform (in the range of [5,100]) distributions. We refer to this datastes as `pareto`, `normal` and `uniform`.

5.3 Results

All algorithms are implemented in Python 2.7 and all the experiments are run single threaded on a Macbook Air (OS-X 10.9.4, 4GB RAM). We compare our algorithm with `RandPart`and the `K_means`algorithm, the built in k-means function in Scipy library. Each experiment was repeated 5 times and the average results are reported in this section. For sample size in `CohPart_S` algorithm, we set parameter c (explained earlier) to 4 in all experiments.

5.3.1 Results on Real World Datasets

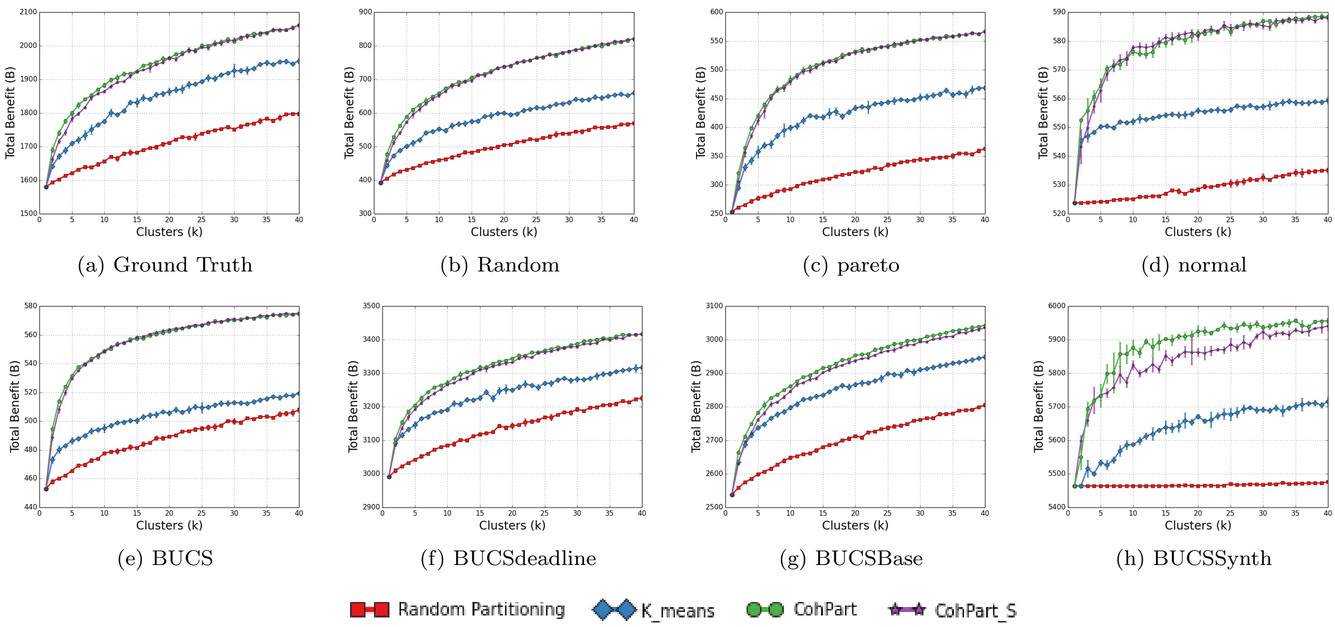
BUCS. The result on the BUSC data is depicted in Figure 1e where each point shows the benefit of all students when partitioning them into K groups. As we see the `RandPart` has the lowest benefit and our algorithm has the best benefit. As the number of clusters increases (having hence fewer students 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 `K_means` gets to our algorithm. But in real life, we do not have enough time to repeat (or teach) all of the courses (for e.g. for preparation before SAT exam). Figure 1f illustrates the case when deadline is equal to the average sum of need vectors for different students.

BUCSBase. 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.

BUCSSynth dataset. We ran our algorithms on on BUCSSynth 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



(a) Ground Truth (b) Random (c) pareto (d) normal

(e) BUCS (f) BUCSdeadline (g) BUCSBase (h) BUCSSynth

Figure 1: Total benefit achieved by different algorithms as a function of the number of groups of students.

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

[1] Data mining for providing a personalized learning path in creativity: An application of decision trees. *Computers & Education*, 68(0):199 – 210, 2013.

[2] R. Agrawal, B. Golshan, and E. Terzi. Grouping students in educational settings. In *ACM SIGKDD*, pages 1017–1026, 2014.

[3] A. Anagnostopoulos, L. Becchetti, C. Castillo, A. Gionis, and S. Leonardi. Power in unity: Forming teams in large-scale community systems. In *ACM International Conference on Information and Knowledge Management*, pages 599–608, 2010.

[4] A. Ashman and R. Gillies. *Cooperative Learning: The Social and Intellectual Outcomes of Learning in Groups*. Taylor & Francis, 2003.

[5] P. Brucker. *Scheduling Algorithms*. Springer-Verlag New York, Inc., 3rd edition, 2001.

[6] R. F. Bruner. Repetition is the first principle of all learning. *Social Science Research Network*, 2001.

[7] D. Esposito. Homogeneous and heterogeneous ability grouping: Principal findings and implications for evaluating and designing more effective educational environments. *Review of Educational Research*, 43(2):163–179, 1973.

[8] J. Kleinberg, C. Papadimitriou, and P. Raghavan. Segmentation problems. *J. ACM*, pages 263–280, 2004.

[9] T. Lappas, K. Liu, and E. Terzi. Finding a team of experts in social networks. In *ACM SIGKDD*, pages 467–476, 2009.

[10] A. Majumder, S. Datta, and K. Naidu. Capacitated team formation problem on social networks. In *ACM SIGKDD*, pages 1005–1013, 2012.

[11] T. P. Novikoff, J. M. Kleinberg, and S. H. Strogatz. Education of a model student. *Proceedings of the National Academy of Sciences*, 109(6):1868–1873, 2012.

[12] A. Segal, Z. Katzir, K. Gal, G. Shani, and B. Shapira. Edurank: A collaborative filtering approach to personalization in e-learning. 2014.

[13] R. E. Slavin. Ability Grouping and Student Achievement in Elementary Schools: A Best-Evidence Synthesis. *Review of Educational Research*, 57(3):293–336, 1987.