Team Formation for Scheduling Educational Material in Massive Online Classes

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 $\mathbf{S}$ 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; Large Classes

1. INTRODUCTION

Education has always been regarded as one of the most important tasks of society. Nowadays it is viewed as one of the best means to bridge the societal inequalities gap and to help individuals to achieve their full potential. Accordingly, many work has been dedicated to study how individuals learn and what is the best way to teach them (see [10, 5] for an overview). We recognize two substantial conclusions that studies in this area make on how to improve students' learning outcome. 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([32, 27, 25, 12, 37]. Second, grouping students; working in teams with their peers helps students to access the material from a different viewpoint as well [2, 6, 39, 27, 38].

In this paper we study the problem of creating personalized educational material for teams of students by taking a computational perspective. More specifically, we focus on two problems: the first problem is how to identify the right schedule for a group of students, when the group is apriori formed. The second problem is how to partition a set of students into groups and design personalized schedules per group so that the benefit of students in terms of how much they learn and absorb is maximized.

Significant amount of work has been carried out on designing personalized educational content, such as [29] in the context of online education services and more notably on designing personalized schedules by Novikoff et al. [32] which has inspired our current work. Team formation in education is another well-studied area [2, 14, 31] and it has been showed that students can improve their abilities by interaction and communication with other team members [34].

However, 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. Therefore, our contribution is to present formal definition of aforementioned problems, study their computational complexity and design algorithms for solving them. In addition to this 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.

Roadmap: The rest of the paper is organized as follows: After reviewing the related work in Section 2, we define our framework and settings in Section 3. In Section 4 we define group schedule problem. In Section 5 we formally define Cohort Selection and also show its computational complexity. In the same section, we present our `CohPart` to solve Cohort Selection . In Section 6 we show usefulness of our `CohPart` on synthetic and real world datasets. Finally we conclude the paper in Section 7.

2. RELATED WORK

Our problem is related to psychology, education and computer science including ability grouping, repetition in learn-

ing and team formation. We review some of these works here:

Ability grouping: Majority of the studies in this area find that over the whole population, there definitely is a gain in academic performance due to ability grouping [17, 39, 23, 24, 21, 9]. Most of the studies agree, that there is high increase to learning of students in high-ability groups. Some say there is only small gain, while others say there is zero gain for low-ability groups. But even in this case, gain to the medium and high ability groups counters these negative effects. The benefits of grouping on students' attitude has also been studied in [23]. Authors have shown that students in grouped classes developed more positive attitudes toward the subjects they were studying than did students in ungrouped classes.

Repetition in learning: Repetition has long been regarded as essential in learning. When learning a new activity for the first time, new information is gained and stored in the short-term memory. This information will be lost over time when there is no attempt to retain it [33, 36, 19, 11, 1, 16, 15] Repetition in learning and spacing effect has been even studied in computer science in [32]. In this work authors try to optimize a single student's learning in the light of Ebbinhaus's work. They model education process as a sequence of abstract units and these units are repeated over time. However they did not consider the importance of having a deadline for e.g. to prepare for a test and also the fact that after enough repetitions the information will move to long-term memory and there is negligible gain from repetition.

Team formation: An earlier version of this study has appeared in [7]. Team formation has been studied in operations research community [8, 13, 41, 42], which defines the problem as finding optimal match between people and demanded functional requirements. It is often solved using techniques such as simulated annealing, branch-and-cut or genetic algorithms [8, 41, 42]. It has also been studied in computer science [2, 3, 20, 26, 30, 35, 4] Majority of these work focus on team formation to complete a task and minimize the communication cost among team members. The focus of these studies is on finding only one team to perform a given task. [2] considers partitioning students in which each student has only one ability level for all the activities and each team has a set of leaders and followers. The goal is to maximize the gain of students where gain is defined as the number of students who can do better by interacting with the higher ability students. Our problem differs as we consider different ability levels for different activities.

3. PRELIMINARIES

Already Aristotle said that "it is frequent repetition that produces a natural tendency." The fundamental basis of our work is the realization that repetition is an essential part of learning; engaging with a topic multiple times [1] deepens and hastens students' engagement and understanding processes [11, 40]. In this paper we focus on developing optimal schedules for teaching groups of students (e.g. classes) that observe this dependency of learning quality on reiteration of topics. 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 based on two very simple rules; only one topic can be assigned to each timeslot but the same topic can appear in multiple slots. 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 $\mathcal{A}$.

Topic requirements. For every student $s \in \mathbf{S}$ and topic $t \in \mathbf{T}$ there is a number of times that s has to hear about t in order for s to learn every aspect of this topic. We call this number the *requirement* of s on t and denote it by the integer function $\texttt{req}(s, t)$.

Benefits from topic. In order for a student s to be fully knowledgeable about topic t, he has a requirement to learn $\texttt{req}(s, t)$ times about t. We assume that until s has met his requirements, he gains knowledge and hence, will benefit to some extent from every repetition of t. After $\texttt{req}(s, t)$ repetitions of t, while there is no detriment, there is also no additional benefit to s from hearing about t. We call $\mathbf{b}(s, \langle t, i\rangle)$ (Equation (1)) the *benefit* of s from topic t when hearing about it for the i^{th} time. We assume that s benefits equally from each of the first $\texttt{req}(s, t)$ occurrences of t in $\mathcal{A}$, thus $\mathbf{b}(s, \langle t, i\rangle) = \frac{1}{\texttt{req}(s,t)}$ if $i \leq \texttt{req}(s, t)$. Since after this point s has already mastered topic t, there is no additional benefit from any later repetition of t and hence $\mathbf{b}(s, \langle t, i\rangle) = 0$.

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

Note that for ease of exposition, we assume that all repetitions of t before $\texttt{req}(s, t)$ carry equal benefit to s. However, the definition and all of our later algorithms could easily be extended to use some other function $\mathbf{b}'(s, \langle t, i\rangle)$. A natural choice for example is a function, where earlier repetitions of t carry higher benefit than later ones, thus $\mathbf{b}'(s, \langle t, i\rangle) = \frac{1}{2^i}$. The intuition is that first you learn about the fundamentals of t and later you add on additional information.

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 $\mathcal{A}$,

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

Remember that in Equation (2), $\mathcal{A}[r]$ refers to the tuple $\langle t, i\rangle$ that is scheduled at timeslot r in $\mathcal{A}$.

Observe, that every time topic t appears in the schedule $\mathcal{A}$, it will contribute with the same amount of benefit towards

[1]For e.g. learning about a topic multiple times, reiterating it, possibly in different formats or from different viewpoints

$\mathbf{B}(s, \mathcal{A})$, regardless of the exact timeslot allocation within $\mathcal{A}$.

4. THE GROUP SCHEDULE PROBLEM

In this section we investigate the problem of how to divide students in such groups and assign schedules to each group to maximize the benefit of students in every group.

Group benefits. Let $P \subseteq \mathbf{S}$ be a subset of the students, we refer to P as a *group*. The notion of the benefit of a schedule $\mathcal{A}$ to a single student s lends itself to a straightforward extension to the *benefit of a group*. We define the benefit $\mathbf{B}(P, \mathcal{A})$ group P has from $\mathcal{A}$ in Equation (3) as the sum of the benefits over all students in P.

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

The GROUP SCHEDULE **problem.** Given a group P, our first task is to find an optimal schedule for this group, that is to find a schedule that maximizes the group benefit of P. 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. There is a simple polynomial time algorithm that solves problem 1. We cal this algorithm `Schedule`(P, d). We present `Schedule`(P, d) in Algorithm 1.

Remember that for any topic t the requirement $\texttt{req}(s, t)$ may be different for the different students in P. 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 of $\langle t, i\rangle$ can be computed as the sum of benefits over all students in P as given in Equation (4).

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

Observe that because students have different requirements for t, the subsequent repetitions of the same topic may have different (decreasing) marginal benefits.

Algorithm 1 is a greedy algorithm that at every timeslot chooses an instance of the topic with the largest marginal benefit. To achieve this we maintain an array B in which values are marginal benefit of topics $t \in \mathbf{T}$, if next repetition of t is added to the schedule $\mathcal{A}_P$. We keep the number that topic t has been added to $\mathcal{A}_P$ in array R.

The `Schedule` algorithm is an iterative algorithm that repeats until all d timeslots in the schedule are filled; it selects the topic u_t with the largest marginal benefit from B and adds it to the schedule $\mathcal{A}_P$ (Lines 5 and 6) . Then it updates marginal benefit of u_t, $B[u_t]$ (Lines 7- 8).

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**

Runtime of `Schedule`. The runtime of Algorithm 1 is best computed from the point of view of computing marginal benefits of topics in B. In each iteration of the loop, the marginal benefit is only recomputed for one of the topics, u_t with the largest benefit which has been added to the schedule $\mathcal{A}_P$ most recently. The total runtime of algorithm is $O(d(|P|+|\mathbf{T}|))$. If we keep the marginal benefits in a max-heap, we can reduce the running time to $O(d(|P| + log|\mathbf{T}|))$. Algorithm 1 yields an optimal schedule for a group P.

PROPOSITION 1. *The schedule $\mathcal{A}_P$ output by Algorithm 1 yields maximal benefit $\mathbf{B}(P, \mathcal{A})$ for the group P.*

PROOF. Observe, that the benefit of adding the i^{th} repetition $\langle t, i\rangle$ of topic t to $\mathcal{A}$ is only dependent on i and t but not on any other topic. Hence the choice that we make in algorithm 1 in any iteration does not change the marginal benefit $\mathbf{m}(P, \langle t, i\rangle)$. Thus choosing the topic t with the largest marginal benefit in any iteration of algorithm 1 results in a schedule with maximal total benefit for the group. □

5. THE COHORT SELECTION PROBLEM

So far we discussed how to find an optimal schedule of topics for a given group of students. 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; we have to 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. We call the problem of finding a partition that yields the highest sum of group benefits the 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 for every student-topic pair. 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 we assume that $\mathcal{A}_P =$ `Schedule`(P, d) for every group.

THEOREM 1. COHORT SELECTION *(Problem 2) is NP-hard.*

PROOF. We reduce the CATALOG SEGMENTATION problem [22] to COHORT SELECTION problem. CATALOG SEGMENTATION is the following problem; there is a universe of items $\mathcal{U} = \{\mathtt{u}_1, \mathtt{u}_2, \ldots, \mathtt{u}_m\}$ and subsets $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_n \subseteq U$ given. Find two subsets $\mathcal{X}$ and $\mathcal{Y}$ of $\mathcal{U}$, both of size $|\mathcal{X}| = |\mathcal{Y}| = r$, such that

$$\text{CS}(\mathcal{X}, \mathcal{Y}) = \sum_{i=1}^{n} \max\{|\mathcal{S}_i \cap \mathcal{X}|, |\mathcal{S}_i \cap \mathcal{Y}|\} \tag{6}$$

is maximized. It is proven by Kleinberg *et al.* [22] that CATALOG SEGMENTATION is NP-hard.

We now show that if we can solve COHORT SELECTION then we can also solve the CATALOG SEGMENTATION problem. More specifically, we map an instance of CATALOG SEGMENTATION to an instance of COHORT SELECTION as follows: every subset $\mathcal{S}_i$ in CATALOG SEGMENTATION is mapped to a student s_i in COHORT SELECTION and element of the universe $\mathtt{u}_i \in \mathcal{U}$ of CATALOG SEGMENTATION is mapped to a topic t_i in COHORT SELECTION. For student s_i and topic t_j we set the requirement $\mathtt{req}(s_i, t_j) = 1$ if $\mathtt{u}_j \in \mathcal{S}_i$, otherwise $\mathtt{req}(s_i, t_j) = nm^3$. We also set $d = r$ and $K = 2$.

We can also map a solution of COHORT SELECTION to a solution of CATALOG SEGMENTATION and vice verse; let $\mathcal{P} = \{X, Y\}$ be a partition of the students $\mathbf{S}$ in COHORT SELECTION and let $\mathcal{A}_X$ and $\mathcal{A}_Y$ be the optimal schedules for X and Y. We define the sets $\mathcal{X}$ and $\mathcal{Y}$ in CATALOG SEGMENTATION from $\mathcal{A}_X$ and $\mathcal{A}_Y$. Specifically, let $\{t_1^x, t_2^x, \ldots, t_r^x\}$ be the topics (possible with multiplicity) that appear in $\mathcal{A}_X$. Then we define $\mathcal{X} = \{\mathtt{u}_{t_1^x}, \mathtt{u}_{t_2^x}, \ldots, \mathtt{u}_{t_r^x}\}$ to contain the elements in $\mathcal{U}$ corresponding to the topics in $\mathcal{A}_X$. $\mathcal{Y}$ is derived in a similar way from $\mathcal{A}_Y$.

Given a solution $\mathcal{X}$ and $\mathcal{Y}$ to CATALOG SEGMENTATION, we can define the partition $\mathcal{P} = \{X, Y\}$ and the corresponding group schedules $\mathcal{A}_X$ and $\mathcal{A}_Y$. For every $s \in \mathbf{S}$ we assign s to X if $|\mathcal{S} \cap \mathcal{X}| > |\mathcal{S} \cap \mathcal{Y}|$ and assign s to Y otherwise, where $\mathcal{S}$ is the set in CATALOG SEGMENTATION that we identified with student s. Further, the group schedule $\mathcal{A}_X$ is the schedule that contains topic t if and only if $\mathtt{u}_t \in \mathcal{X}$. Similar, $\mathcal{Y} = \{t | \mathtt{u}_t \in \mathcal{Y}\}$.

We show if $\mathcal{P} = \{X, Y\}$ is an optimal solution to COHORT SELECTION, then the corresponding solution $\mathcal{X}$, $\mathcal{Y}$ has to be an optimal solution to CATALOG SEGMENTATION. First, observe that because of the choice of the requirements in COHORT SELECTION, if $\mathbf{B}$ is the value of a solution to COHORT SELECTION, then the value of $catalogsegmentation(\mathcal{X}, \mathcal{Y}) \geq \lfloor \mathbf{B} \rfloor$. Further, $\lfloor \mathbf{B}(\{X, Y\}, d) \rfloor = catalogsegmentation(\mathcal{X}, \mathcal{Y})$, where $\mathcal{P} = \{X, Y\}$ is derived from $\mathcal{X}$ and $\mathcal{Y}$.

Let us assume, that $\mathcal{P} = \{X, Y\}$ is an optimal solution to COHORT SELECTION, but the derived $\mathcal{X}$ and $\mathcal{Y}$ are not optimal for CATALOG SEGMENTATION. That means there exist $\mathcal{X}'$ and $\mathcal{Y}'$, such that $catalogsegmentation(\mathcal{X}, \mathcal{Y}) < catalogsegmentation(\mathcal{X}', \mathcal{Y}')$. However, in this case the partition $\mathcal{P}' = \{X', Y'\}$ with the schedules $\mathcal{A}_{X'}$, $\mathcal{A}_{Y'}$ derived from $\mathcal{X}'$ and $\mathcal{Y}'$ would yield a higher value for COHORT SELECTIONproblem, contradicting the optimality of $\mathcal{P}$. □

5.1 Partition algorithms.

We first describe briefly two popular algorithms for clustering, `K_means` and `Random Partitioning` and how it is applied to our problem. Then we proceed to present our solution, `CohPart` to the COHORT SELECTION and a sampling-based speedup, `CohPart_S` .

`Random Partitioning` is assigning each point randomly to a cluster. We use this partitioning as a baseline to compare our algorithm with. Also we use it as the initialization part of our `CohPart` algorithm.

`K_means` is a clustering method used to minimize the average squared distance between points in the same cluster. Solving `K_means` problem [18] exactly is NP-hard. Lloyd's algorithm [28] solves this problem by choosing k centers randomly and assigning the points to the closest center. Then the centers are recomputed based on the points assigned to it. These two phases are repeated until there is no more improvement on the cost of clustering. In our setting the students are the data points and the repetition for each topic represent each dimension.

`CohPart` algorithm. The `CohPart` algorithm (Cohort Partitioning) is presented in algorithm 3 and consists of two phases; first there is an initialization phase (Lines 3- 6), in which a random clustering is executed on all of the students (Line 3) and then for each partition p_i, the centers are computed (Lines 4- 6) using algo 1. When initial cluster centers are chosen, then there is an iterative phase (Lines 7- 14) where students get reassigned to clusters and cluster centers are updated again.

In our notations $\mathcal{A}$ and $\mathbf{R}$ both show the schedules (of a group of students or a single student). $\mathcal{A}$ shows the vector of size d consisting of topics and their repetitions $\langle t, R[t] \rangle$ for each time slot. $\mathbf{R}$ is a vector of size $|\mathbf{T}|$ and for each topic t, how many times it can be repeated in deadline d.

Algorithm 2 Benefit algorithm for computing the benefit of a single student s from a schedule $\mathbf{R}$

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

1: $b = 0$
2: **for** all topics $t \in \mathbf{T}$ **do**
3: $\quad b = b + \frac{min(req(s,t), \mathbf{R}[t])}{\mathbf{R}[t]}$
4: **end for**

Runtime : `CohPart` is a heuristic to solve COHORT SELECTIONproblem. In each iteration of the algorithm, the group that each student can benefit the most is found and student is assign to that group. This will take $O(k|\mathbf{T}|)$ for each student. Then the schedule of each group is updated and algorithm iterates until convergence is achieved. The total running time 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 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 $\mathcal{P}$.

1: $C =$
2: $\mathcal{P} = \{P_1, P_2, \ldots, P_K\}$
3: Run `Random Partitioning` on the students and obtain P_i's
4: **for** $i = 1, \ldots, K$ **do**
5: $\quad c_i = \texttt{Schedule}(P_i, d)$
6: **end for**
7: **while** convergence is achieved **do**
8: $\quad$ **for** all students $s \in \mathbf{S}$ **do**
9: $\quad\quad P_i \leftarrow s$, such that $i = \text{argmax}_{j=1,\ldots,k}\, \mathbf{b}(s, c_j)$
10: $\quad$ **end for**
11: $\quad$ **for** $i = 1, \ldots, K$ **do**
12: $\quad\quad c_i = \texttt{Schedule}(P_i, d)$
13: $\quad$ **end for**
14: **end while**

`CohPart_S` **algorithm.** The `CohPart_S` (Cohort Partitioning with Sampling,) resembles `CohPart` except that it performs clustering on a random sample of students of size n' and when clustering is finished assigns the remaining students to the cluster with the maximum benefit $\mathbf{b}(s, c_j)$. It reduces the running time to $O(kn'|\mathbf{T}|)$. We set $n' = k * c$ for different values of c.

5.2 Constraints on Topic Order

In real-life, most often we cannot pick any scheduling of topics we like. Instead, there are strict precedence constraints among the topics. For example, one has to learn addition before he can learn about multiplication during a math course. Therefore, we assume that along with the topics, a set of constraints is also given. The constraints can be simple ones, such as the first occurrence of topic t_i has to be before topic t_j, or more complicated ones, topic t_j can only be scheduled after at least r_1 repetitions of t_{i_1} and r_2 repetitions of t_{i_2}. Of course, the set of constraints can also be empty, if we do not have any of them. We can easily modify algorithm 1 to take into account these constraints and check for precedence constraints. To achieve this, after line 4 we can check for precedence constraints and in line 5 we choose only the topics which their precedence constraints are met.

6. EXPERIMENTS

The goal of these experiments is to gain an understanding of how our clustering algorithm works in terms of performance (objective function). 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 introduce Graded Response Model (GRM) briefly, then explain different datasets and finally show how well our algorithm is doing on each dataset.

Item Response Theory and Graded Response Model: In psychometric, Item Response Theory (IRT) is a framework for designing and evaluating tests, questions and questionnaires. In IRT models the probability of giving a correct answer by a student to a question is determined based on the ability of student and the difficulty of the question. For our work we used the Graded Response Model (GRM), an advanced IRT model which fits our data well and handles partial credit values. Using our data on grades of students for taken courses, GRM helps us to deduce ability scores for each student and difficulty scores for each course. Having these score parameters, then we can generate the missing grades for courses that a student did not take. We also used GRM to obtain a model to generate a larger dataset, i.e. BUCSSynth.

6.1 Datasets

This subsection describes each dataset and their attributes.

BUCS data: The original BUCS dataset consists of grades of students in CS courses at Boston University. This data was collected from Fall 2003 to Fall 2013. Each row of data looked like: FALL 2003, CS101, U12345, U1, C+ which shows the semester year, course number, students' BU id, undergraduate/graduate year and the grade. It consists of 9833 students. We only considered students who were taking CS330 and CS210 (required courses to obtain a major in CS) which consisted of 398 students and 41 courses. Here the courses correspond to topics. Obviously the new dataset had some missing values, not all 41 courses were taken by those 398 students. To fill the grades for missing (student, course) pairs, we used GRM. First using GRM, we obtained the ability and difficulty parameters for all students and all courses. The abilities [2] and difficulties' parameters are available online[3]. 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. After having all grades for all courses, we had to transform these grades to the number of required repetitions to learn a course. We assumed the number of required repetition to master a course (or topic) for the smartest student is 5 (base parameter). Note that throughout a semester students review the course materials to solve homework, do project and prepare for quizzes, midterm and final exams, so they review material for at least 5 times. Thus for students who got A, we considered 5 repetitions needed to fully master the course and as the ability (and grade) drops, number of repetition goes up (step parameter). We also tried different base and step values for our experiments.

BUCSSynth data: In order to see how well our algorithm scales to a larger dataset, 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 data, we obtained difficulty parameters for 41 courses. In order to obtain difficulties for 100 courses, we used the following approach:

1. Choose one of the 41 courses at random.
2. Use density estimation, smoothing and then get the CDF of the difficulties.

[2] `http://cs-people.bu.edu/bahargam/abilities`
[3] `http://cs-people.bu.edu/bahargam/difficulties`

3. Randomly sample from the CDF to get the difficulties for a new course.
Using the aforementioned parameters, we generated the grades for 2000 students and 100 courses and we transformed the grades to number of repetitions similar to what we did for BUCS dataset. This dataset [4] and the code [5] to generate it are available online.

Synthetic data: Our first synthetic dataset is to generate ground truth data to compare our algorithm to `Random Partitioning` and `K_means` . In this dataset we had generated 10 groups of students, each group containing 40 students. For each group we selected 5 courses and assigned repetitions randomly to those 5 courses such that the sum of repetition will be equal to the deadline[6]. Then for the remaining 35 courses, we filled the required number of repetitions with random numbers taken from a normal distribution with $\mu = \frac{deadline}{5}$ and $\sigma = 3$. We refer to this dataset as GroundTruth. We expect our algorithm to be able to find the right clusters of students while `K_means` cannot find this hidden structure.

We have also generated the repetitions for 400 students and 40 courses using Pareto, Normal and Uniform distributions. We refer to this datastes as `pareto`, `normal` and `uniform`. To generate number of repetitions for different courses using the pareto distribution, we used the shape parameter $\alpha = 2$. For normal distribution we used $\mu = 30$ and $\sigma = 5$ and for uniform dataset we generated random numbers in the range of [5,100].

6.2 Results:

Our experiments compare our algorithm in terms of our objective function (students' benefit) with `Random Partitioning` and `K_means` Ṙecall that the students' benefit is defines in Equation (5). The current algorithm is 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 `Random Partitioning` 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.

6.2.1 Results on Real World Datasets

BUCS: We executed our algorithm on `BUCS` dataset untill reaching convergence and show how well it maximized the benefit of learning while varying the number of clusters We compare `CohPart` and `CohPart_S` to `Random Partitioning` and `K_means` . The result is depicted in Figure 1e where each point shows the benefit of all students when partitioning them into k groups. As we see the `Random Partitioning` 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, when having one tutor for each student. The benefit grows dramatically from 1 cluster to 10 cluster. But after 10 cluster the increase in the potential is slower. We also show the 95 confidence interval, but it was small that cannot be seen in some plots.

BUCSdeadline: We also show the result for different values of deadline. As the deadline increases, the gap between `K_means` and our algorithm decreases. The reason is as deadline is greater we have to take into consideration more topics to teach to the students. Note that `K_means` algorithm behaves like our algorithm except it considers all the courses and ignores the deadline. So 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. We observe that when the base is equal to 1 and step is also small, `K_means` also performs well, but still our algorithm is doing better than `K_means` . The larger is the value of base and step parameter, the better our algorithm performs.

6.2.2 Results on Semi-synthetic Dataset

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

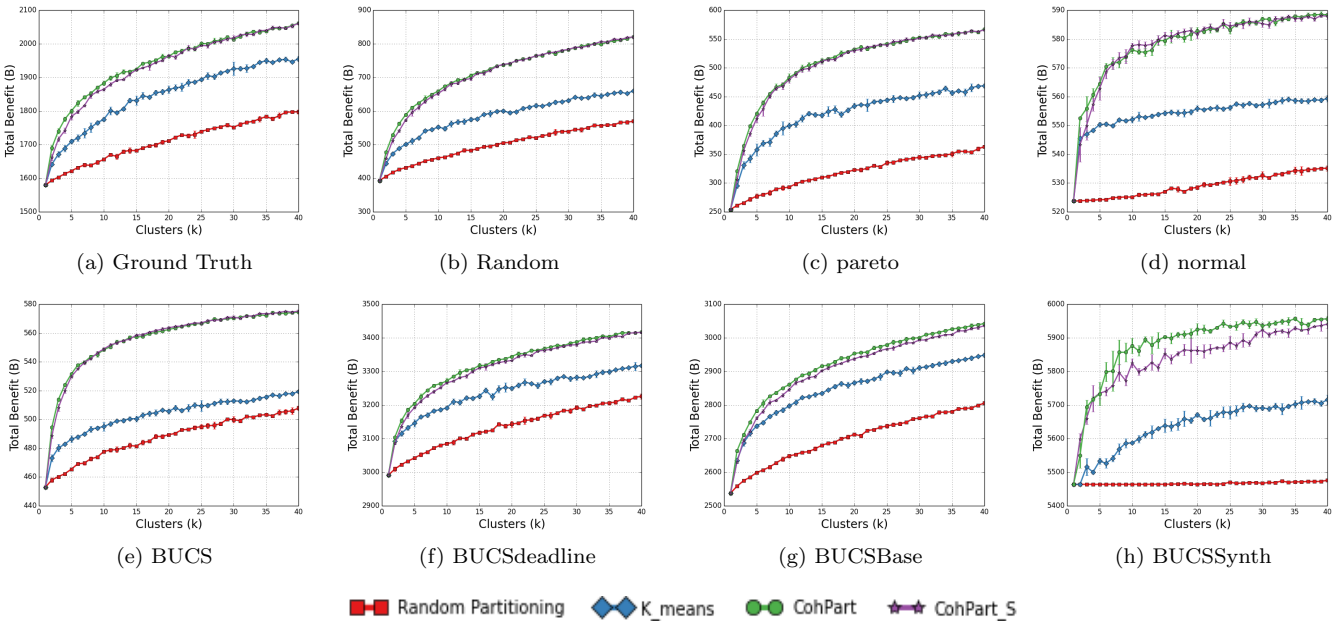
6.2.3 Results on Synthetic Datasets

Our first set of experiments on synthetic data used the `ground truth` dataset. The result is illustrated in Figure 1a. As we see `CohPart` and `CohPart_S` both are performing really well. For all of the courses the mean required repetition 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 as well after running our algorithm and the schedule should include the selected courses in each group. In each group students have different repetition values for the selected courses, 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 and did not cluster these students together. 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 `Random Partitioning` .

7. 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 in different topics. Our algorithm maximized the potential and benefit of team members for learning . The encouraging results that we obtained shows that our proposed solution is

[4] `http://cs-people.bu.edu/bahargam/BUCSSynth`
[5] `http://cs-people.bu.edu/bahargam/BUCSSynthCode`
[6] The repetition for those selected courses are not equal for the students in the same group, but for all the students in the group the sum of selected courses is equal to the deadline.



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

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

Figure 1: Performance of `Random Partitioning` , `K_means` , `CohPart` and `CohPart_S` for Cohort Selection problem on different datasets

effective and suggest that we have to consider personalized teaching for students and form more efficient teams.

8. REFERENCES

[1] R. Agrawal, B. Golshan, and E. Terzi. Forming beneficial teams of students in massive online classes. In *Proceedings of the first ACM conference on Learning@ scale conference*, pages 155–156. ACM, 2014.

[2] R. Agrawal, B. Golshan, and E. Terzi. Grouping students in educational settings. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 1017–1026, New York, NY, USA, 2014. ACM.

[3] A. Anagnostopoulos, L. Becchetti, C. Castillo, A. Gionis, and S. Leonardi. Power in unity: Forming teams in large-scale community systems. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, CIKM '10, pages 599–608, New York, NY, USA, 2010. ACM.

[4] A. Anagnostopoulos, L. Becchetti, C. Castillo, A. Gionis, and S. Leonardi. Online team formation in social networks. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12, pages 839–848, New York, NY, USA, 2012. ACM.

[5] J. Aronson, editor. *Improving academic achievement : impact of psychological factors on education.*

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

[7] S. Bahargam, D. Erdos, A. Bestavros, and E. Terzi. Personalized education; solving a group formation and scheduling problem for educational content. In *The 8th International Conference on Educational Data Mining*, 2015.

[8] A. Baykasoglu, T. Dereli, and S. Das. Project team selection using fuzzy optimization approach. *Cybern. Syst.*, 38(2):155–185, Feb. 2007.

[9] B. S. Bloom. The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring. *Educational Researcher*, 13(6):4–16, 1984.

[10] J. Bransford, A. Brown, and R. Cocking, editors. *How People Learn: Brain, Mind, Experience, and School - Expanded Edition.* 2000.

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

[12] P. Brusilovsky and C. Peylo. Adaptive and intelligent web-based educational systems. *International Journal of Artificial Intelligence in Education*, 13(2):159–172, 2003.

[13] S.-J. Chen and L. Lin. Modeling team member characteristics for the formation of a multifunctional team in concurrent engineering. *Engineering Management, IEEE Transactions on*, 51(2):111–124, May 2004.

[14] 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.

[15] E. Galbrun, B. Golshan, A. Gionis, and E. Terzi. Finding low-tension communities. *arXiv preprint arXiv:1701.05352*, 2017.

[16] B. Golshan, T. Lappas, and E. Terzi. Profit-maximizing cluster hires. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1196–1205. ACM, 2014.

[17] B. Grossen. *How should we group to achieve excellence*

with equity. PhD thesis, Unviersity of Oregon, July 1996.

[18] J. Hartigan and M. Wong. Algorithm AS 136: A K-means clustering algorithm. *Applied Statistics*, pages 100–108, 1979.

[19] M. Y. Jaber and H. V. Kher. Variant versus invariant time to total forgetting: The learn–forget curve model revisited. *Computers & Industrial Engineering*, 46(4):697–705, 2004.

[20] M. Kargar and A. An. Discovering top-k teams of experts with/without a leader in social networks. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, CIKM '11, pages 985–994, New York, NY, USA, 2011. ACM.

[21] A. C. Kerckhoff. Effects of ability grouping in british secondary schools. *American Sociological Review*, 51(6):842–858, 1986.

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

[23] C.-L. C. Kulik and J. A. Kulik. Effects of Ability Grouping on Secondary School Students: A Meta-analysis of Evaluation Findings. *Am Educ Res J*, 19(3):415–428, Jan. 1982.

[24] J. A. Kulik and C.-L. C. Kulik. Meta-analytic findings on grouping programs. *Gifted Child Quarterly*, 36(2):73–77, 1992.

[25] A. I. Lakatos. Introduction. *Journal of the Society for Information Display*, 8(1):1–1, 2000.

[26] T. Lappas, K. Liu, and E. Terzi. Finding a team of experts in social networks. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pages 467–476, New York, NY, USA, 2009. ACM.

[27] C. F. Lin, Y. chu Yeh, Y. H. Hung, and R. I. Chang. Data mining for providing a personalized learning path in creativity: An application of decision trees. *Computers & Education*, 68(0):199 – 210, 2013.

[28] S. P. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28:129–137, 1982.

[29] J. Lu. Personalized e-learning material recommender system. In *International conference on information technology for application*, pages 374–379, 2004.

[30] A. Majumder, S. Datta, and K. Naidu. Capacitated team formation problem on social networks. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, pages 1005–1013, New York, NY, USA, 2012. ACM.

[31] J. M. McPartland and M. Johns Hopkins Univ., Baltimore. *School Structures and Classroom Practices in Elementary, Middle, and Secondary Schools. Report No. 14 [microform] / James M. McPartland and Others*. Distributed by ERIC Clearinghouse [Washington, D.C.], 1987.

[32] 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.

[33] B. Pentland. The learning curve and the forgetting curve: The importance of time and timing in the implementation of technological innovations. In *49th annual meeting of the Academy of Management, Washington, DC*, 1989.

[34] N. M. Rachel Hertz-Lazarowitz. *Interaction in cooperative groups: The theoretical anatomy of group learning*. Cambridge University Press, 1995.

[35] S. S. Rangapuram, T. Bühler, and M. Hein. Towards realistic team formation in social networks based on densest subgraphs. In *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13, pages 1077–1088, Republic and Canton of Geneva, Switzerland, 2013. International World Wide Web Conferences Steering Committee.

[36] H. Roediger and J. Nairne. *The Foundations of Remembering: Essays in Honor of Henry L. Roediger III*. Psychology Press Festschrift Series. Psychology Press, 2007.

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

[38] A. P. Sergio Gutierrez-Santos, Manolis Mavrikis. Mining students' strategies to enable collaborative learning. 2014.

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

[40] C. J. Weibell. *Principles of learning: A conceptual framework for domain-specific theories of learning*. PhD thesis, Brigham Young University. Department of Instructional Psychology and Technology, 2011.

[41] H. Wi, S. Oh, J. Mun, and M. Jung. A team formation model based on knowledge and collaboration. *Expert Systems with Applications*, 36(5):9121 – 9134, 2009.

[42] A. Zakarian and A. Kusiak. Forming teams: an analytical approach. *IIE Transactions*, 31(1):85–97, 1999.