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 the benefit of hearing about the topic multiple times, and repetitions of a topic may count the benefit of hearing about the topic. We assume that for the first repetition of a topic the student gains the full benefit. For the second repetition we assume half of the benefit and so on.

For a topic $t$ in the sequence of topics $\mathcal{T}$, the repetition number $r$ of topic $t$ is denoted by $\text{req}(s, t)$. The benefit of topic $t$ in schedule $\mathcal{A}$, denoted by $B_t(\mathcal{A})$, is the sum of benefits of all repetitions of topic $t$ in schedule $\mathcal{A}$.

\[ B_t(\mathcal{A}) = \sum_{r=1}^{\text{req}(s, t)} b(s, \mathcal{A}[r]) \]
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)).

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

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 marginal benefit, $m(P, (t, i))$, from the $i$th repetition of $t$ (thus $(t, i)$) to $P$ is the increase in the group benefit if $(t, i)$ is added to $A$. The marginal benefit can be computed as the sum of benefits over all students in $P$ as given in Equation (4).

$$m(P, (t, i)) = \sum_{s \in P} b(s, (t, i)) \quad (4)$$

The Schedule algorithm is an iterative algorithm with $d$ iterations that in every iteration appends a topic to the schedule $A_P$. 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 $A_P$. In every iteration Schedule selects the topic $u_t$ with the largest marginal benefit from $B$ and adds it to $A_P$ (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 $O(d(|P| + \log(|T|))$.

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: $B \leftarrow [m(P, (t, 1))]$ for $t \in T$ |
| 3: $R \leftarrow [0]$ for all $t \in T$ |
| 4: while $|A_P| < d$ do |
| 5: Find topic $u_t$ with maximum marginal benefit in $B$ |
| 6: $A_P \leftarrow (A_P, R[u_t])$ |
| 7: $R[u_t] \leftarrow +$ |
| 8: Update $B[u_t]$ 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 $P = P_1 \cup P_2 \cup \cdots \cup P_K$ 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

$$B(P, d) = \sum_{P \in P} B(P, A_P) \quad (5)$$

is maximized, where $A_P = Schedule(P, d)$ 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 $req(s, t)$, 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 $req(s, t)$ for a student $s \in P$ and every topic $t \in T$ and a single schedule $A$. |
| Output: $Benefit(s, A)$ Benefit of $s$ from schedule $A$. |
| 1: $Benefit(s, A) = 0$ |
| 2: for all topics $t \in T$ do |
| 3: $Benefit(s, A) = Benefit(s, A) + \min(req(s, t), A[t])$ |
| 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.

1http://cs-peole.bu.edu/bahargam/edm/
\textbf{Algorithm 3 CohPart} for computing the partition $P$ based on the benefit of students from schedules.

\textbf{Input:} requirement $\text{req}(s,t)$ for every $s \in S$ and $t \in T$, number of timeslots $d$, number of groups $K$.

\textbf{Output:} partition $P$.

1. $A = \{A_1, A_2, \ldots, A_K\}$
2. $P = \{P_1, P_2, \ldots, P_K\}$
3. $i \in R[1,2,\ldots,K]$, $P_i \leftarrow s$ for every $s \in S$
4. for $i = 1,\ldots,K$ do
5.     $A_i = \text{Schedule}(P_i, d)$
6. end for
7. while convergence is achieved do
8.     for all students $s \in S$ do
9.         $P_i \leftarrow s$, $i = \arg\max_{j=1,\ldots,k} \text{Benefit}(s, A_j)$
10. end for
11. for $i = 1,\ldots,K$ do
12.     $A_i = \text{Schedule}(P_i, d)$
13. end for
14. end while

RandPart: Partition $S$ at random.

K\_means: We represent each student $s$ by the $|T|$-dimensional vector $(\text{req}(s,t_1), \text{req}(s,t_2), \ldots, \text{req}(s,t_T))$ 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 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 topics and letter grades were converted to numbers. For each group we selected 5 courses and assigned to the group that maximizes $\text{Benefit}(s, A'_i)$.

BUCSSyn 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 BUCCSSyn. 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 $\mu = 2\text{deadtime}$ 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 datasets 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 BU CS 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 1g illustrates the case when deadline is equal to the average sum of need vectors for different students.

BUCCSBase. 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.

BUCCSSyn dataset. We ran our algorithms on on BUCCSSynth 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
Figure 1: Total benefit achieved by different algorithms as a function of the number of groups of students.

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