Sequence-based Approaches to Course Recommender Systems

Ren Wang and Osmar R. Zaïane[0000−0002−0060−5988]

University of Alberta, Canada {ren5,zaiane}@cs.ualberta.ca

Abstract. The scope and order of courses to take to graduate are typically defined, but liberal programs encourage flexibility and may generate many possible paths to graduation. Students and course counselors struggle with the question of choosing a suitable course at a proper time. Many researchers have focused on making course recommendations with traditional data mining techniques, yet failed to take a student's sequence of past courses into consideration. In this paper, we study sequence-based approaches for the course recommender system. First, we implement a course recommender system based on three different sequence related approaches: process mining, dependency graph and sequential pattern mining. Then, we evaluate the impact of the recommender system. The result shows that all can improve the performance of students while the approach based on dependency graph contributes most.

Keywords: Recommender Systems, Dependency Graph, Process Mining

1 Introduction

After taking some courses, deciding which one to take next is not a trivial decision. A recommendation of learning resources relies on a recommender system (RS), a technique and software tool providing suggestions of items valuable for users [14]. The typical approaches to recommend an item are based on ranking some other items similar to another item a user or a customer has already taken, purchased, or liked. These are called Content-based recommender systems [3]. However, recommending a course simply based on similarity with previously taken courses may not be the right thing to do. In practice, in addition to course prerequisite constraints, when the curriculum is liberal, students typically chose courses where their friends are, or based on their friends suggestions (i.e. ratings). Collaborative filtering [16] is another approach for recommender systems that could be used to recommend courses. It relies on the wisdom of the crowd, -i.e. the learners that are similar to the current students in terms of courses taken or "liked". However, the exact sequence these courses are taken is not considered. The order and succession of courses is indeed relevant in choosing the next course to take. The questions students may ask include but are not restricted to: how can I finish my study as soon as possible? Is it more advantageous to take course A before B or B before A? What is the best course for me to take this semester?

Will it improve my GPA if I take this course? Answering such questions to both educators and students can greatly enhance the educational experience and process. However, very few course RS (CRS) currently take advantage of this unique sequence characteristic.

Recommender systems are widely used in commercial systems and while rarely deployed in the learning environments, their use in the e-learning context has already been advocated [24][9]. The overall goal of most RS in education is to improve students' performance. This goal can be achieved in diverse ways by recommending various learning resources [18]. A common idea is to recommend papers, books and hyperlinks [17] [6] [8]. Course enrollment can also be recommended [10] [5]. However, most RS only apply content-based or collaborative filtering approaches, and none have considered exploiting the order of how students take courses. This missing link is what this paper tries to address.

The goal of our paper is to investigate a sequence-based CRS and show that it is possible. We study three sequence-based approaches to build this RS using process mining, dependency graphs, and sequential pattern mining.

2 CRS based on Process Mining

2.1 Review of Process Mining

Process mining (PM) is an emerging technique that can discover the real sequence of various activities from an event log, compare different processes and ultimately find the bottlenecks of an existing process and hence improve it [20]. To be specific, PM consists of extracting knowledge from event logs recorded by an information system and discovering business process from these event logs (process discovery), comparing processes and finding discrepancies between them (Process Conformance), and providing suggestions for improvements in these processes (Process Enhancement).

Some attempts have already been made to exploit the power of PM in curriculum data. For instance, authors of one section in [15] indicate that it can be used in educational data. However, the description is too general and not enough examples are given. The authors of [19] point out the significant benefit in combining educational data with PM. The main idea is to model a curriculum as a coloured Petri net using some standard patterns. However, most of the contribution is plain theory and no real experiment is conducted. Targeted curriculum data and thereby curriculum mining is explored in [11]. Similar with the three components of PM, it clearly defines three main tasks of curriculum mining, which are curriculum model discovery, curriculum model conformance checking and curriculum model extensions. The authors explain vividly how curriculum mining can answer some of the questions that teachers and administrators may ask. However, no RS is built upon it.

2.2 Implementation of a CRS based on Process Mining

We recommend courses to a student that successful students who have a similar course path have taken. Our course data are different from typical PM data

at least in the following three aspects: First, the order of the activities is not rigidly determined. Students are quite free to take the courses they like and they do not follow a specific order. Granted that there are restrictions such as prerequisite courses or the courses we need to take in order to graduate, these dependencies are relatively rare compared with the number of courses available. Second, the dependency length is relatively short. In the course history data, we do not have a long dependency. We may have a prerequisite requirement, e.g., we must take CMPUT 174 and CMPUT 204 first in order to take CMPUT 304, but such dependency is very short. Third, the type of activities in the sequence are not singletons. Data from typical PM problems are sequence of single activities, while in our case they are a sequence of sets. Students can take several courses in the same term, which makes it more difficult to represent in a graph.

For these reasons, we do not attempt to build a dependency graph, and proceed directly to conformance checking. The intuition behind our algorithm is to recommend the path that successful students take, i.e., to recommend courses taken by the students who are both successful and similar to our students who need help. We achieve this by the steps in Algorithm 1.

Algorithm 1 Algorithm of CRS based on PM

Input :
Logs L of finished students course history
Student *stu* who needs course recommendations
Execute :
1: Find all high GPA students from L as HS
2: Set candidate courses $CC = \emptyset$
3: **for all** $stuHGPA$ in HS **do**
4: Apply Algorithm 2 to compute the similarity *sim* between *stu* and $stuHGPA$
5: **if** *sim* is greater than a certain threshold **then**
6: Add courses that $stuHGPA$ take next to CC
7: **end if**
8: **end for**
9: Rank CC based on selected metrics
10: Recommend the top courses from CC to *stu*

In Algorithm 1 we first find the history of all past successful students. We assume success is measured based on final GPA. Other means are of course possible. From this list we only keep the successful students who are similar to the current student based on some similarity metric, and retain the courses they took as candidate courses to recommend. These are finally ranked and the top are recommended. The ranking is explained later.

The method we use to compute the similarity between two students is highlighted in Algorithm 2. It is an improved version of the casual footprint approach for conformance checking in PM. Instead of building a process model, we apply or method directly on the sequence of sets of courses to build the footprint ta-

bles. In addition, we define some new relations among activities, courses in our case, due to the special attributes of course history and the sequence of set.

- Direct succession: $x \rightarrow y$ iff x is directly followed by y
- Indirect succession: $x \rightarrow\!\!\rightarrow y$ iff x is indirectly followed by y
- Reverse direct succession: $x \leftarrow y$ iff y is directly followed by x
- Reverse indirect succession: $x \leftarrow\!\!\leftarrow y$ iff y is indirectly followed by x
- Same term: $x \parallel y$ iff x and y are in the same term
- Other: $x \# y$ for Initialization or if x and y have the same name

With the relation terms defined, we can proceed to our improved version of the footprint algorithm which computes the similarity of two course history sequences.

Algorithm 2 Algorithm of computing the similarity of two course history sequences

Input :
Course history sequence of the first student s_1
Course history sequence of the first student s_2
Output :
1: Truncate the longer sequence to the same length with the shorter sequence
2: Build two blank footprint tables that map between s_1 and s_2
3: Fill out two footprint tables based on s_1 and s_2
4: Calculate the total elements and the number of elements that are different
5: Compute the similarity
6: Return the similarity of s_1 and s_2

In most cases, finished students' course histories are much longer than the current students'. To eliminate this difference we truncate the longer sequence to the same length of the shorter sequence. The next step is to build a one-to-one mapping of all courses in both sequences. Our CRS computes the above defined relations based on the two sequences and fills the relations in the footprint table separately. Lastly, our CRS calculates $differenceCount$ which is the number of elements in the footprint tables that s_1 differs from s_2, and $totalCount$ which is the total number of elements in one footprint table. $similarity$ is then:

$$similarity = 1 - \frac{differenceCount}{totalCount}.$$

3 CRS based on Dependency Graph

3.1 Review of Dependency Graph (DG)

A primitive method to discover DG from event data is stated in [1]. The dependency relation is based on the intuition that for two activities A and B, if B follows A but A does not follow B, then B is dependent on A. If they both

follow each other in the data, they are independent. In fact, this simple intuitive idea lays the foundation for many process discovery algorithms in PM. These are, however, more advanced, as they use Petri nets [13] to deal with concurrency and satisfy other criteria, such as the Alpha Algorithm [21], the heuristic mining approach [23], and the fuzzy mining approach [7]. These approaches are, however, not quite suitable for our task. Our method here is based on [4]. The authors developed an approach of recommending of learning resources for users based on users' previous feedback. It learns a DG by users ratings. Learners are required to give a rating or usefulness of the resources they used. The database evolves by filtering learning objects with low ratings as time goes by. The dependencies are discovered based on these ratings, positive or negative, using an association rule mining approach.

3.2 Implementation of a CRS based on Dependency Graph

The method in [4] is to recommend resources to learners based on what learners have seen and rated. It creates dependencies between items i and items j only if an item j is always positively rated immediately upon appearing *after* an always positively rated i when it is *before* j, and independent or ignored otherwise. Resource j is dependent on i in the pair (i, j) based on ratings.

Admittedly, the approach is simple but has drawbacks (i.e. linear, no context used, and ignores noise), but we propose to adapt it to make it more suitable to our case of courses, and improved it as follows. We cannot ask students to rate all the courses they have taken, as these may not be very reliable for building dependencies. The indicator we built our dependencies upon is the mark obtained by students in courses. A good mark for course i before a good mark of course j often implies course i is the prerequisite or positively influencer of course j. Moreover, instead of using a universal notion of positive and negative as for the ratings, A positive mark in a course or a negative mark is defined relative to a student. A $B+$ may be a good mark in general, but for a successful student whose mark is A on average, $B+$ is not that good. Moreover, we use association rule mining parameters *support* (indicating frequency) and *confidence* (indicating how often a rule has been found to be true) to threshold pairs of courses with positive marks, and thus reduce potential noise.

Algorithm 3 outlines our approach with the above rationale. The CRS first learns dependencies from the finished students' course history. For a student who needs recommendations, the CRS checks the previous course history of this student and compares this history with the dependencies the CRS has learned. A ranking of the candidate courses constitutes the final recommendation.

4 CRS based on Sequential Pattern Mining

4.1 Implementation of a CRS based on Sequential Pattern Mining

Sequential pattern mining (SPM) consists of discovering frequent subsequences in a sequential database [2]. There are many algorithms for SPM but we adopt the widely used PrefixSpan [12] because of its recognized efficiency.

Algorithm 3 Algorithm of CRS based on DG

Input :

Logs L of finished students course history

Student stu who needs course recommendations

Execute :

1: Convert all marks of courses from L to positive or negative signs. The standard may differ based on GPA to make it relative to individual students
2: Build the projected dataset of positive courses P_{i+} and negative courses P_{i-} with the highlighted modification. Remove courses in P_{i-} from P_{i+}
3: Set candidate courses $CC = \emptyset$
4: Add to CC courses in P_{i+} whose prerequisites are finished
5: Rank CC based on selected metrics
6: Recommend the top courses from CC to stu

SPM was introduced and is typically used in the context of market basket analysis. The sequences in the database are the progression of items purchased together each time a purchaser comes back to a store, and SPM consists of predicting the next items that are likely to be purchased at the next visit. Students take few courses each term. There is no order of courses in a specific term, yet the courses of different terms do follow a chronological order. The analogy with market basket analysis is simple. A semester for a student is a store visit, and the set of courses taken during a semester are the items purchased together during one visit. Just like frequent sequence patterns of items bought by customers can be found, so can frequent sequence patterns of courses taken by students.

Our CRS Algorithm 4 based on SPM works as follows. Since we only want to find the sequential patterns of positive courses, i.e., sequences of courses taken by students with good outcome, we first filter all the course records and only keep a course record when the mark is A or $A+$. Here $A+$ and A are taken as reference examples. Note that a course deleted in one sequence of a student may be selected in another sequence for another student. For instance, a student who took CMPUT 101 and received an A then this course is kept in this student's sequence. If another student who also took CMPUT 101 but received a B this course is filtered from their sequence. After this step, the course records left in students history are all either A or $A+$. The second step in the algorithm is to treat these courses like the shopping items and process them with PrefixSpan [12] to find all the sequential patterns of courses. Among the course sequential patterns we find, some are long, while some are short. Ideally, we want to recommend courses from the most significant patterns.

Suppose we have a student who needs course recommendations and has already taken courses 174, 175, and 204. We have discovered a short frequent pattern $s_1 = \langle 174, 206 \rangle$ while another long frequent pattern s_2 we discovered is $\langle 174, 175, 204, 304 \rangle$. A more intuitive recommendation should be 304 because the student has already finished three courses in s_2.

Based on this intuition, the courses we recommend are the next unfinished elements from the sequential patterns that have the longest common elements

Algorithm 4 Algorithm of CRS based on SPM

Input :
Logs L of finished students course history
Student stu who needs course recommendations

Execute :
1: Filter all the course records of L with a predefined course mark standard as FL
2: Find all the course sequential patterns SP from FL with PrefixSpan [12].
3: **for all** Sequential pattern p from SP **do**
4: Compute the number of elements num of this sequential pattern that is also contained in stu's course history
5: Add the next course of this p to the Hashtable HT where the key is num
6: **end for**
7: Rank courses from HT's highest key as candidate courses CC based on selected metrics
8: Recommend the top courses from CC to stu

with our student's current course history. By this algorithm, the course we recommend for our example student earlier will be course 304 since the length of common elements of s_2 and this student is three, longer than one which is of s_1.

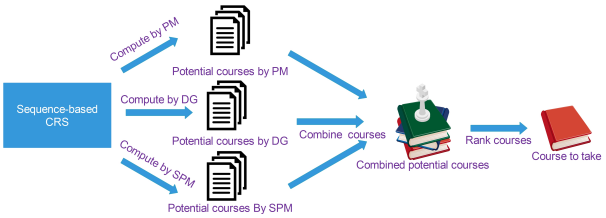

Fig. 1. The overall workflow of our CRS that combines all 3 sequence-based algorithms

In addition to the three approaches for CRS, PM-based, DG-based, and SPM-based, we combine all of our three sequence-based methods into one comprehensive one. We call it "Comprehensive" in our experiments. Since each of them produces a potential list of recommended courses, it is straight forward to combine the result of potential courses of all three methods and rank the result. The overall structure of this approach is shown in Figure 1.

5 Ranking Results

All methods previously mentioned focus more on student's course performance, which we approximate with the GPA. Of course, other learning effectiveness measure alternatives exist. Since the quickness of a program before graduation is also of concern to many learners who would like to graduate as soon as possible, we also consider the length of sequences of courses before graduation in our

recommendation. To do this, we incorporate this notion in the ranking of the candidate courses before taking the top to recommend.

The sequence of some courses and the number of courses and the compulsory courses to graduate are dictated by the school or department program. These requirements can be obtained from the school guidelines. Most of these programs, however, are liberal not enforcing most constraints and contain many electives. These optional courses can be further considered in two aspects: First, these courses may be very important that many students decide to take them even though they are not in the mandatory list. We can compute the percentage of students who take a specific course and rank courses based on this percentage from high to low. It could be a must for students who want to graduate as soon as possible if the percentage of students who take this course is above a certain threshold. The second aspect to distinguish courses that can speed up graduation is their relationship with the average duration before graduation. For one course, we can compute the average time needed to graduate by students who take this specific course. We do this for all the courses and rank them based on the average graduation time from low to high, the lower the number the faster a student graduates, i.e. the likelier it contributes to the acceleration of graduation. In short, there are three attributes we consider: First, the course is mandatory from the department's guideline; Second, is the percentage of students who take this course; Third, is the average time before graduation by students who take this course. The second category can actually be merged into the first category since they both indicate how crucial a course is, either by the department or the choice of students. We combine the courses that are chosen by more than 90% (this threshold can be changed) of students with the compulsory courses specified by educators as one group we call *key courses*.

This "agility strategy" is used to rank the potential recommended courses selected by our three sequence-based algorithms. This ranking process is always the last step of these three sequential based algorithms. To be more exact, after selecting a few courses in the potential course list by one of the three sequence-based approaches, there are three methods to rank them with this "agility" algorithm.

1. No "agility": Rank courses merely on the GPA contribution of courses
2. Semi "agility": Always rank key courses that are in the potential course list first. The key course list and the non-key course list will be ranked based on each course's GPA contribution respectively.
3. Full "agility": Always rank key courses that are in the potential course list first. The key course list and the non-key course list will be ranked based on each course's average graduation time by students who take this course.

6 Experiments

6.1 Data Simulator

The Computing Science Department of the University of Alberta collects for each semester and for each student the courses they register in and the final

mark they obtain. While there are prerequisites for courses and other strict constraints, the rules are not enforced and are thus often violated, giving a plethora of paths to graduation. This history for many years, constituting the exact needed event log, is readily available. However, such data cannot be used for research purposes or for publication even though anonymized due to lack of ethical approval. Indeed, we would need inaccessible consent from alumni learners. It is hopeless to gather the consent of all past students, and impractical to start collecting written consent from new students as it would require years to do so. We were left with alternative to simulate historic curriculum data for proof of concept and publication, and use real data for local implementation. For this paper we opted for the simulation of the event log. A simulator was developed to mimic the behaviours of undergraduate students with different characters in higher education. The simulator encompasses the dynamic course directory and the rules of enrollment, as well as student behaviour such as performance and diligence in following guideline rules. The detail of the simulator simulating arriving and graduating students one semester at a time can be found in [22].

6.2 Result Analysis

In this section we compare the performance of our CRS based on different sequence-based algorithms. We want to see which sequence-based algorithm performs better, whether the "speedup" algorithm works, and what additional insights our CRS can provide. Moreover, we add one more approach to all experiments, which is called "comprehensive" that combines all results from the three methods. If not otherwise specified, the parameters of each algorithm are the ones that performed best. The numbers presented in each table and figure for this section are the average scores of their corresponding experiment three times since the simulation is stochastic.

The first experiment is to compare the performance of different sequence-based approaches at different student stages. "Different stages" means when do students use our CRS. For example, "Year 4" means students only begin to take courses recommended by our CRS in the fourth year, while "Year 1" means students start using our CRS from the first year. Table 1 with its corresponding Figure 2 shows the result of this experiment: 200 students' average GPAs varied by the year of starting CRS in different approaches. The blue line in the middle is our baseline 3.446 which is the average GPA if students do not take any recommendations. From Table 1 and Figure 2 we can observe the following. Firstly, we can see a substantial effect for students who use our CRS in the first two years. This steady increase indicates students can benefit more if they start using our CRS earlier in their study. Secondly, the performance of CRS for all methods is about the same with the baseline if students only start to use our CRS in the fourth year, which means it may be too late to improve a student's GPA even with the help of a CRS. Other than Year 4, our CRS does have a positive impact. Thirdly, CRS based on DG outperforms all in nearly all scenarios while other approaches are equally matched. Note that the comprehensive approach does not outperform others. Our interpretation is that

by combining the candidate courses from all three methods, it obtains too many candidates and cannot perform well if the candidates are not ranked properly. As to why CRS based on DG performs best, it may be due to the intrinsic attribute of our data simulator. The mark generation part of our simulator considers course prerequisites, which may favour the DG algorithm. Thus, other approaches may outweigh DG if we are dealing with real data.

Table 1. 200 students' average GPAs varied by the year CRS is used by different approaches

Approach	Year 4	Year 3	Year 2	Year 1
PM	3.453	3.516	3.569	3.588
DG	3.433	3.529	3.617	3.652
SPM	3.447	3.498	3.545	3.602
Comprehensive	3.441	3.512	3.564	3.593

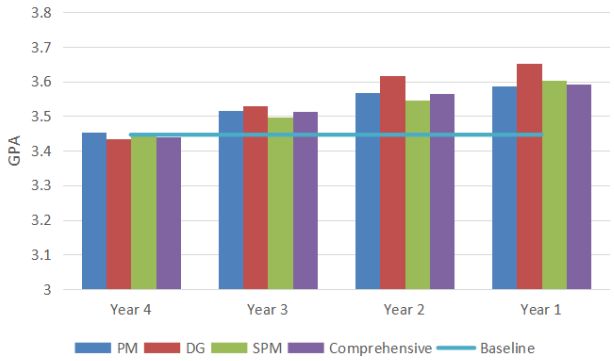

Fig. 2. 200 students' average GPAs varied by the year CRS is used by different approaches

The next experiment is to check whether increasing the training data in the number of students would lead to a better performance of our CRS. Table 2 and Figure 3 demonstrate 200 students' average GPAs varied by the number of training students of CRS in different approaches. We can see that, as the training data size increases from 500 to 1000, the performance of our CRS improves. However, when this size further increases from 1000 to 1500, the performance of our CRS does not improve significantly. We than fixed the training data size to 1500 in all our experiments. This can be explained by the fact that the number of courses in a program is finite and small (even though dynamic) and all important dependencies are already expressed in a relatively small training dataset.

Table 2. 200 students' average GPAs varied by the number of training students of CRS in different approaches

Approach	500	1000	1500
PM	3.513	3.57	3.586
DG	3.535	3.607	3.639
SPM	3.528	3.581	3.598
Comprehensive	3.522	3.582	3.597

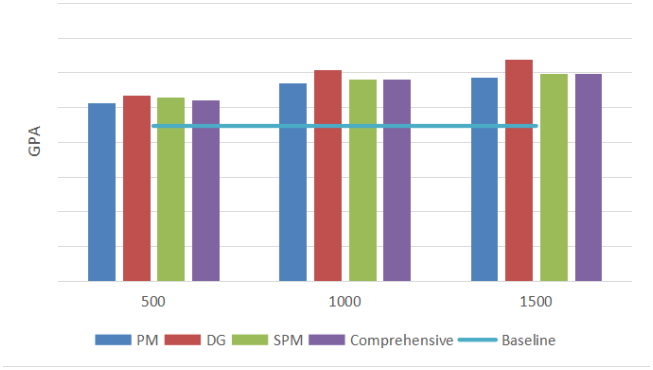

Fig. 3. 200 students' average GPAs varied by the number of training students of CRS in different approaches

Besides improving students' performance in grades, our CRS can also speed up students' graduation process by ranking the candidate courses selected by sequence algorithms properly. Table 3 and Figure 4 show the effect of using the full "agility" ranking setting to recommend courses based on DG to 200 students. Same as the first experiment in this section, Year X means students start to use our CRS from year X. We can see a remarkable decrease in the number of terms needed to graduate if students start using our CRS from the third year. However, after that, such change is not very notable. Since the pivotal fact to graduate fast is to take all key courses as soon as possible, our explanation is that taking key courses from the third year is timely. There is no particular need to focus on key courses in the first two years. Note that although the graduation time improvement of our CRS is only in a decimal level, it is already quite a boost considering students only need to study 12 terms in normal scenarios.

Other than recommending courses, our CRS may provide some insights to educators and course counselors. We previously mentioned computing courses' GPA contribution and graduation time contribution. A course's GPA contribution is the average GPA of students who take this course, while a course's graduation time contribution is the average time before graduation of students who take this course. These indicators are used to rank the candidate courses obtained by sequence-based algorithms. Yet, these indicators themselves may have values. Table 4 demonstrates the top 5 GPA contribution courses and graduation

Table 3. 200 students' average graduation terms varied by the year of starting CRS based on DG with the full "agility" setting

Starting Year	Average Graduation Terms
Year 4	11.917
Year 3	11.615
Year 2	11.567
Year 1	11.532

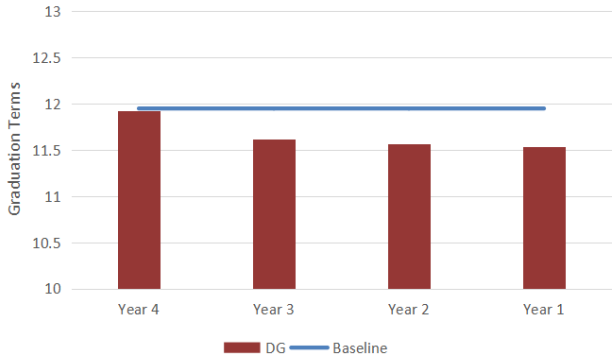

Fig. 4. 200 students' average graduation terms varied by the year of starting CRS based on DG with the full "agility" setting

time contribution courses. One interesting finding is course CMPUT 201. This course is not one of the preferred courses in our simulator but is a prerequisite course for many courses. A preferred course is a course that will have a very high probability to be taken in a particular term because it is the "right" course for that term. Being a prerequisite course but not a preferred course means that, CMPUT 201 has to be taken in order to perform well in other courses but many students do not take it. Thus, finding this course actually means that our CRS found an important course that is not in the curriculum but is necessary for students to succeed. Sometimes it is risky to force to do so. For example, CMPUT 275 is in the top position in the GPA contribution list, but we cannot know whether this course causes students to succeed or successful students like to take it. Nevertheless, this contribution list would still provide some insights to educators and course counselors if it is trained on real students' data and is carefully interpreted.

Finally, our CRS can assist educators and administrators to gain deep insights on course relations and thus improve the curriculum. Figure 5 (Left) shows the DG of courses with edge colours representing discovery sources (green=imposed and confirmed; blue=expected but not found; red=new discovered). It combines the prerequisite relations used by our simulator and the dependencies discovered by our DGA. On one hand, we can consider the prerequisite course relations used by our simulator as the "current curriculum" or behaviours we expect to see from

Table 4. The top 5 GPA contribution courses and graduation time contribution courses

Ranking	Top GPA Courses	Top Time Courses
1	CMPUT 275	CMPUT 301
2	CMPUT 429	CMPUT 274
3	CMPUT 350	CMPUT 300
4	CMPUT 333	CMPUT 410
5	CMPUT 201	CMPUT 366

students. On the other hand, the courses' prerequisite relations discovered by our CRS based on the DG algorithm can be deemed as the prerequisite relations in reality or the actual behaviours by students. Many dependencies used by our simulator are found by our DG algorithm (green edges) like 204⇒304, which means that these rules are successfully carried out by students. Some dependencies used by our simulator are not found in the data (blue edges) like 175⇒229 because the students did not actually follow them, which indicates there are some discrepancies between what we expect from students and what students really do. Administrators may want to check why this happens. There are also some dependencies found by our DG algorithm but are not in the rules for our simulator (red edges), such as 304⇒366 and 272⇒415. These dependencies indicate some relations among courses unknown and unexpected to administrators but are performed by students. Educators and administrators may want to consider to add these new found prerequisites to the curriculum in the future if these are indicative of good overall performance in terms of learning objectives.

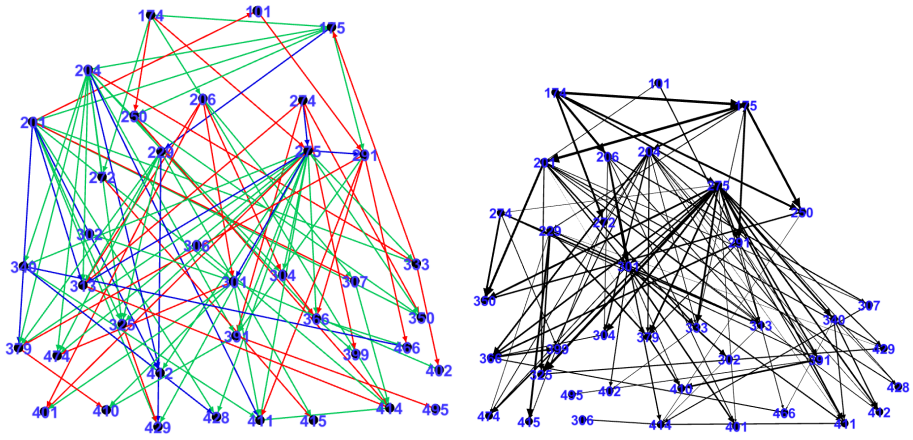

Fig. 5. Left: The DG of courses with edge colours representing discovery sources (green=imposed and confirmed; blue=expected but not found; red=new discovered). Right: The paths of successful students filtered from the 1500 training students with the weight of edges representing the number of students.

Figure 5 (Right) shows the paths of successful students (GPA above 3.8) filtered from the 1500 training students with the weight of edges representing the number of students. The thick edges mean many successful students have gone through these paths and they should be considered when trying to improve the curriculum. All in all, the benefits of these findings can be considerable when sequences of courses are taken into account.

7 Conclusions and Future Work

We built a course recommender system to assist students choose suitable courses in order to improve their performance. This recommender is based on three different methods yet all three are related to the sequence of taken course. We considered conformance checking of process mining as a first approach, recommending courses to a student that successful students, who have a similar a course path, have taken. We have also suggested a new approach based on dependency graphs modeling deep prerequisite relationships, by recommending courses whose prerequisites are finished. We also advocated a third method based on sequential pattern mining discovering frequent sequential course patterns of successful students. Finally, we combined all the approaches in a comprehensive method and proposed ranking methods to favour reducing the program length.

We conduct several experiments to evaluate our course recommender systems and to find the best recommendation approach. All three approaches can improve students performance in different scales. The best recommendation method is based on the dependency graph, and the number of recommended courses accepted by students have a positive correlation with the performance. Moreover, the course recommender system we build can speed up students' graduation if set properly, and provide some useful insights for educators and course counselors.

References

1. R. Agrawal, D. Gunopulos, and F. Leymann. *Mining process models from workflow logs*. Springer, 1998.
2. R. Agrawal and R. Srikant. Mining sequential patterns. In *Proc. 11th International Conference on Data Engineering*, pages 3–14. IEEE, 1995.
3. R. Burke. Hybrid web recommender systems. In *The adaptive web*, pages 377–408. Springer, 2007.
4. D. Cummins, K. Yacef, and I. Koprinska. A sequence based recommender system for learning resources. *Australian Journal of Intelligent Information Processing Systems*, 9(2):49–57, 2006.
5. E. García, C. Romero, S. Ventura, and C. De Castro. An architecture for making recommendations to courseware authors using association rule mining and collaborative filtering. *User Modeling and User-Adapted Interaction*, 19(1-2):99–132, 2009.
6. K. I. Ghauth and N. A. Abdullah. Learning materials recommendation using good learners ratings and content-based filtering. *Educational technology research and development*, 58(6):711–727, 2010.

7. C. W. Günther and W. M. van der Aalst. Fuzzy mining–adaptive process simplification based on multi-perspective metrics. In *Business Process Management*, pages 328–343. Springer, 2007.
8. J. Luo, F. Dong, J. Cao, and A. Song. A context-aware personalized resource recommendation for pervasive learning. *Cluster Computing*, 13(2):213–239, 2010.
9. N. Manouselis, H. Drachsler, R. Vuorikari, H. Hummel, and R. Koper. Recommender systems in technology enhanced learning. In *Recommender systems handbook*, pages 387–415. Springer, 2011.
10. M. P. O'Mahony and B. Smyth. A recommender system for on-line course enrolment: an initial study. In *Proceedings of the 2007 ACM conference on Recommender systems*, pages 133–136. ACM, 2007.
11. M. Pechenizkiy, N. Trcka, P. De Bra, and P. Toledo. Currim: Curriculum mining. In *International Conference on Educational data Mining*, pages 216–217, 2012.
12. J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, and M.-C. Hsu. Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In *Proc. 17th International Conference on Data Engineering*. IEEE, 2001.
13. J. L. Peterson. *Petri net theory and the modeling of systems*, volume 132. Prentice-hall Englewood Cliffs (NJ), 1981.
14. F. Ricci, L. Rokach, and B. Shapira. *Introduction to recommender systems handbook*. Springer, 2011.
15. C. Romero, S. Ventura, M. Pechenizkiy, and R. S. Baker. *Handbook of educational data mining*. CRC Press, 2010.
16. B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295. ACM, 2001.
17. T. Y. Tang and G. McCalla. Smart recommendation for an evolving e-learning system. In *Workshop on Technologies for Electronic Documents for Supporting Learning, AIED*, 2003.
18. N. Thai-Nghe, L. Drumond, A. Krohn-Grimberghe, and L. Schmidt-Thieme. Recommender system for predicting student performance. *Procedia Computer Science*, 1(2):2811–2819, 2010.
19. N. Trcka and M. Pechenizkiy. From local patterns to global models: Towards domain driven educational process mining. In *9th International Conference on Intelligent Systems Design and Applications (ISDA)*, pages 1114–1119. IEEE, 2009.
20. W. M. van der Aalst. *Process Mining: Discovery, Conformance and Enhancement of Business Processes*, volume 136. Springer, 2011.
21. W. M. van der Aalst, A. Weijters, and L. Maruster. Workflow mining: Discovering process models from event logs. *IEEE Transactions on Knowledge and Data Engineering*, 16(9):1128–1142, 2004.
22. R. Wang. Sequence based approaches to course recommender systems. Master's thesis, University of Alberta, March 2017.
23. A. Weijters, W. M. van der Aalst, and A. A. De Medeiros. Process mining with the heuristics miner-algorithm. *Technische Universiteit Eindhoven, Tech. Rep. WP*, 166:1–34, 2006.
24. O. R. Zaïane. Building a recommender agent for e-learning systems. In *Proceedings International Conference on Computers in Education*, pages 55–59. IEEE, 2002.