



Course Recommendation based on Sequences: An Evolutionary Search of Emerging Sequential Patterns

Mohammed Ibrahim Al-Twijri¹ · José María Luna² · Francisco Herrera¹ · Sebastián Ventura² 

Received: 22 October 2021 / Accepted: 31 March 2022 / Published online: 19 April 2022
© The Author(s) 2022

Abstract

To provide a good study plan is key to avoid students' failure. Academic advising based on student's preferences, complexity of the semester, or even background knowledge is usually considered to reduce the dropout rate. This article aims to provide a good course index to recommend courses to students based on the sequence of courses already taken by each student. Hence, unlike existing long-term course planning methods, it is based on graduate students to model the course and not on external factors that might introduce some bias in the process. The proposal includes a novel sequential pattern mining algorithm, called (ES)²P (Evolutionary Search of Emerging Sequential Patterns), that properly identifies paths followed by good students and not followed by not so good students, as a long-term course planning approach. A major feature of the proposed (ES)²P algorithm is its ability to extract the best k solutions, that is, those with a best recommendation index score instead of returning the whole set of solutions above a predefined threshold. A real study case is performed including more than 13,000 students belonging to 13 faculties to demonstrate the usefulness of the proposal not only to recommend study plans but also to give advices at different stages of the students' learning process.

Keywords Course recommendation · Sequential pattern mining · Emerging patterns · Supervised descriptive pattern mining

Introduction

The university period is one of the most decisive periods in one's life. Students often experience the university period as a very stressful time mainly due to the fear of failure [1]. From time to time, the lack of success is caused by reasons related to the students themselves, including the freedom to

plan their learning processes and the flexibility of the university schedules [2]. According to Eurostat, the statistical office of the European Union (EU), over 3 million young people in EU had been to university but had discontinued their studies at some point in their life. The main reasons for not continuing their education were numerous: a desire to work instead, finding their studies uninteresting or not meeting their needs, family reasons, etc.

Long-term course planning (LTCP) is an important task in academic advising [3], and it aims to help students in proposing a course list of all future semesters so the dropout rate might be reduced. Nevertheless, LTCP is especially challenging for several reasons such as the number of constraints related to university regulations, the students' abilities and background knowledge, or simply some personal preferences caused by external factors [4]. As a result, when building a study plan, a lot of different aspects are required to prioritize courses. Some of such aspects quantify the importance of including a specific course in the study plan (students' preferences, expected grades due to easy courses, etc.), whereas others rate the chronology of those courses (complexity of the semester based on the courses). LTCP,

✉ Sebastián Ventura
sventura@uco.es

Mohammed Ibrahim Al-Twijri
maltwijri@correo.ugr.es

José María Luna
jmluna@uco.es

Francisco Herrera
herrera@decsai.ugr.es

¹ Department of Computer Science and Artificial Intelligence, Andalusian Research Institute in Data Science and Computational Intelligence (DaSCI), University of Granada, 18071 Granada, Spain

² Department of Computer Science and Numerical Analysis, Andalusian Research Institute in Data Science and Computational Intelligence (DaSCI), University of Cordoba, 14071 Córdoba, Spain

however, does not consider graduate students to model the course priority, which might be a good starting.

Taking all the above into consideration, this paper aims to provide a course index [1] based on the sequence of courses a student has taken already, and the grades obtained by graduate students that followed a similar sequence. This analysis not only provides an index for a specific course but also enables general paths (courses grouped by semesters) that should be followed. The performed recommendation does not take any external factor, but the know-how of the system. In other words, it is based on the paths followed by other students and the final grade of such students not only in a specific course but in the degree. To this aim, we propose (ES)²P (Evolutionary Search of Emerging Sequential Patterns), a sequential pattern mining algorithm [5] to extract general paths (a set of semesters with different courses per semester) that were frequently followed by excellent students, but infrequently or never followed by not so good students. In this regard, (ES)²P gathers two well-known tasks in descriptive analysis: sequential pattern mining [6] and emerging patterns [7]. This synergy is key to identify paths that provide a course recommendation for each student.

The proposal should also deal with an extra issue: the provided set of solutions. Generally, frequent itemset mining algorithms [8] require a minimum frequency threshold value to be predefined, which is related to the number of solutions. It was studied [9] that a small change in that threshold value may lead to an extreme variation in the number of solutions as well as a significant increment in the execution time, especially on high-dimensional data [10]. Hence, to determine the right threshold value is key, and it is not trivial even though the user has a profound background in the application field (it needs to try different thresholds by guessing and re-executing the algorithms once and again until results are good enough). To achieve this goal, the proposed algorithm, which is based on evolutionary algorithms [11], is able to extract a reduced set of solutions without considering any frequency threshold. This proposal guides the search process through the growth ratio or difference in the frequency between two groups of students (excellent and not so good ones). The novelty of the paper can be summarized as follows:

- An evolutionary algorithm, known as (ES)²P, for mining emerging sequential patterns. Sequential pattern mining [5] is a descriptive data mining task that has been applied on discovering frequent patterns. This task, however, has not been considered for discriminative or emerging patterns whose frequency increases significantly from one group or dataset to another. (ES)²P extracts the top- k solutions discovered along the evolutionary process. It is a reduced set of solutions that provide useful information to the students, not requiring

any threshold value and, therefore, any background in the application field.

- A methodology for rating courses is also proposed. Unlike LTCP, the proposal requires nothing more than the courses (ordered by semesters) already passes by the student that is advised. This methodology provides recommendations for students on which courses should be taken, and it considers the proposed (ES)²P algorithm to obtain the most promising sequences of courses.
- A methodology for ordering courses that should be taken by students to success in the degree. This methodology, also based on the proposed (ES)²P algorithm, provides full study plans that should be followed by students to reduce the dropout and failure.

The rest of the paper is organized as follows. [Preliminaries and Related Work](#) introduces some important concepts and related works. [An Evolutionary Algorithm for the Search of Emerging Sequential Patterns](#) describes the proposed methodology, and [Experimental Analysis](#) presents some experimental studies to demonstrate the performance of proposal. Finally, [Cases of Study: Applying \(ES\) P to Course Recommendation](#) shows some study cases for a real scenario and [Conclusion](#) gives some conclusions.

Preliminaries and Related Work

This section describes some concepts concerning sequential pattern mining and emerging patterns that are needed to be understood. It also presents related studies for course recommendation systems.

Preliminaries

The sequential pattern mining task was introduced by Agrawal and Srikant [5] as a way to identify useful patterns in a set of sequences. Although the task was originally proposed to mine sequences of patterns, it has been extended to time series of ordered events [12]. Formally speaking, let $I = \{i_1, i_2, \dots, i_n\}$ be the set of n items contained in a database Ω . Let us also define an itemset X as a set of items from I , that is, $X \subseteq I \in \Omega$. A sequence s is described as an ordered list of itemsets $\langle X_1, X_2, \dots, X_m \rangle$. In a sequence s , an item i_j appears only once in an itemset X_k , but such an item i_j is allowed to appear multiple times in different itemsets belonging to s . As a matter of clarification, let us consider the set of items $I = \{a, b, c, d\}$ and the sequence $s = \langle \{a, c\}, \{b\}, \{a, c, d\} \rangle$. In this example, the item $\{a\} \in I$ appears in the itemsets $X_1 = \{a, c\}$ and $X_3 = \{a, c, d\}$, but it only appears once in each itemset. Additionally, the sequence s is formed by a set of itemsets: $X_1 = \{a, c\}$, $X_2 = \{b\}$ and $X_3 = \{a, c, d\}$. Generally

speaking, the meaning of a sequence is that events within an itemset occur at the same time, but itemsets take place one after another (the itemset X_1 does always appear before X_2).

Sequential pattern mining aims to extract any sequence that appears in Ω . A database Ω gathers a set of sequences $S = \langle s_1, s_2, \dots, s_l \rangle$ as it is shown in Table 1, where S comprises three different sequences, that is, $S = \{s_1, s_2, s_3\}$. Additionally, given two sequences $s_1 = \langle X_1, X_2, \dots, X_n \rangle$ and $s_2 = \langle Y_1, Y_2, \dots, Y_m \rangle$, s_1 is called a subsequence of s_2 , denoted as $s_1 \subseteq s_2$, if there exists integers $1 \leq t_1 < t_2 < \dots < t_n \leq m$ such that $X_1 \subseteq Y_{t_1}, X_2 \subseteq Y_{t_2}, \dots, X_n \subseteq Y_{t_n}$. Thus, $s_3 \subseteq s_1$ in Table 1 since $\{c\} \in s_3 \subseteq \{c\} \in s_1$, and $\{d, e\} \in s_3 \subseteq \{d, e, f\} \in s_1$.

Most sequential pattern mining algorithms focus on the extraction of frequent sequences. This frequency is quantified by the support quality measure, defined as the percentage of sequences in $S \subseteq \Omega$ that contains a specific sequence. It is also defined in terms of absolute values as the number of sequences in $S \subseteq \Omega$ that contains the sequence to be evaluated. Given a sequence s_j , the frequency of such a sequence in a database Ω is denoted as $support(s_j, \Omega)$ and formally defined, in relative terms, as shown in Eq. 1.

$$support(s_j, \Omega) = \frac{|\forall s \in S : s_j \subseteq s|}{|S|} \quad (1)$$

A wide variety of sequential pattern mining algorithms have been proposed so far. GSP, proposed by Srikant et al. [6], is considered as one of the first algorithms in the field. A wide variety of algorithms can be found in the specialized literature for mining sequences that appear in data above a predefined frequency value: Spade [13], Spam [14], PrefixSpan [15], CM-Spade [16] and CM-Spam [16]. The last two are considered as the ones that best perform. Some additional approaches were proposed for mining the top- k frequent sequences, not requiring any a minimum frequency threshold value to be predefined: TKS [17] and TSP [18]. Finally, there are some approaches based on evolutionary computation such as G-CSPM [19], which is a genetic algorithm for mining closed sequential patterns. Another genetic algorithm was proposed in [20] for mining negative sequential patterns. Finally, authors in [21] proposed a Particle Swarm Optimization algorithm for mining sequential patterns.

Table 1 Sample sequence database

Sequence ID	Sequence
1	$\langle \{a, b\}, \{c\}, \{d, e, f\} \rangle$
2	$\langle \{a, b\}, \{b\}, \{g, h\} \rangle$
3	$\langle \{c\}, \{d, e\} \rangle$

The finding of interesting patterns encompasses many additional tasks including periodic pattern mining, high-utility itemset mining, graph mining, among others. Emerging pattern mining (EPM), which is mainly categorized as a supervised descriptive pattern mining technique, is an additional task that aims to discover discriminative patterns. EPM seeks for patterns whose frequency greatly differs from one group or dataset Ω_1 to another dataset Ω_2 . The quality of a pattern p is therefore quantified by such a difference in frequency, generally known as growth rate (GR) that is formally defined in Eq. 2.

$$GR(p, \Omega_1, \Omega_2) = \frac{support(p, \Omega_1)}{support(p, \Omega_2)} \quad (2)$$

Many different algorithms have been proposed so far for mining emerging patterns: MDB-LLBorder [7], JEPProducer [22], ConsEPMiner [23], iEP-Miner [24]. Some of such research works focused on the discovery of intrinsic properties of each group, that is, patterns that do not appear in one group, but it appears at least once in the other group. In other words, patterns having a GR value of ∞ . These patterns, known as jumping emerging patterns, are really useful in fields such as image classification [25], human tasks recognition [26] and bioinformatics [27], among others. Emerging patterns have been also considered in a research work [28] to compare sequences in two groups. Nevertheless, to the best of our knowledge, no algorithm for mining emerging sequential patterns has been proposed yet.

Related Work

Recommender systems have received a lot of attention from different institutions to improve the overall satisfaction grade of their students that results in a higher number of students to enroll. According to a recent review [1], course recommendation systems can be categorized into collaborative filtering-based recommendation systems (CFRS), content-based recommendation systems (CRS), knowledge-based recommendation systems (KRS), hybrid approaches and data mining approaches.

CFRSs are based on the assumption that predictions are done by considering the choices of other students with similar preferences and interests. *Chen et al.* [29] proposed a collaborative filtering algorithm based on the history of students' course selection records. This filtering algorithm also considered introductory text from the courses and the students' performance in that courses. *Huang et al.* [30] recommended courses by means of a novel cross-user-domain collaborative filtering algorithm. The algorithm was able to predict a score for each student based on the course score distribution of similar students that already passed the course. On the other hand, CRSs rely on similarities between

course features. CRSs recommend courses to students based on previously studied courses and their degree of satisfaction. *Lessa et al.* [31] proposed the use of Likedin profiles to recommend appropriate courses. *Mostafa et al.* [32] also developed a recommendation system but, instead of taking the students' preferences from an external source, they analyzed the descriptions of the courses already done by such students. The final aim was to recommend courses that are similar to the previously chosen and, therefore, courses that are of interest for the students.

A major problem of CFRSs and CRSs is the huge amount of data they require to make recommendations. When these systems begin to be used, the recommendation power decreases significantly due to a lack of information, what is called as the *cold-start problem* [1] in literature. Knowledge-based recommendation systems (KRSs) are able to overcome that issue since the recommendation is performed by meeting users' requirements and courses. *Huang et al.* [33] designed a course recommendation system based on an ontology. The system relied on several curricular profiles needed by the students to meet the requirements of different jobs. Like KRSs, hybrid course recommendation systems are also commonly used to overcome the problems of CFRSs and CRSs. Authors in [34] proposed a hybrid system combining courses analyses and rates given by students to provide a course ranking. *Esteban et al.* [35] also proposed a hybrid system that combines information from both the student and the courses, and includes collaborative and content-based filters.

Last but not least, recommendation systems based on data mining approaches have been proposed to help students with the choice of the courses that best fit to them [36]. The UniNet [37] method was recently proposed as a recommendation system based on deep learning to help students to take the right decision on the order, combination and number of courses to take. *Britto et al.* [38] proposed to recommend courses according to the background and preferences of the students, paying special attention to those courses in which the students obtained the best grade. In [39], the authors proposed the use of clustering algorithms to group students and make recommendations based on similarities. Similarly, authors in [40] propose a subgroup discovery algorithm to group types of learners. *Wang et al.* [41] proposed the use of sequential pattern mining to build a course recommendation system. The system searches for sequences of courses followed by students with high GPAs. The largest sequences were considered since they are usually infrequent and, therefore, may offer a better connection to achieve the same academic success as previous students. The proposed algorithm took into consideration features such as the percentage of students enrolled in the course, the quantity of time spent by the students to graduate, the requirements of the courses, etc. A sequential pattern mining approach was also proposed

in [42] to recommend courses based on the learning outcome that each student would get if he/she enrolls in the course.

An Evolutionary Algorithm for the Search of Emerging Sequential Patterns

This section describes the proposed (ES)²P algorithm. First, it describes the data representation that should be followed by any problem including information about students: courses (organized into semesters) and their grades. Here, this section includes how each solution is represented in the proposed evolutionary algorithm together with each of the procedures of the proposal. Finally, it describes how to deal with the resulting set of patterns.

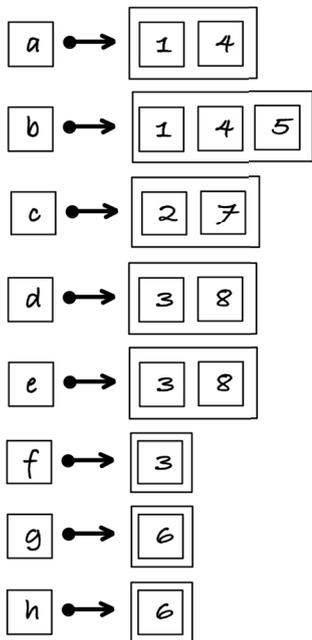
Data Representation

In the proposed approach, the original database is stored into two different data representations while data are read, transaction by transaction. The idea behind these data representations is to provide a fast data access, and to avoid useless information to be maintained. First, data are kept in memory through a vertical data representation that creates a list of indices (sequences in data) in which each item appears at least once. To obtain the frequency or support of each single item in data is quite simple since the algorithm just needs to calculate the length of the list associated with the item at hand. Given two or more items, this vertical data representation allows to obtain the set of data records that have at least one common instance of such items. The only operation to be performed is an intersection of the lists associated with each item. Second, an horizontal data representation is performed where a list of indices is stored. Here, instead of index of the sequences in data, it stores the index of the itemsets in which the item appears. This second data representation is based on a hashing function, so given a key k based on an item, it maps k to the corresponding set of indices on which k is included. Index values are in increasing order from sequence to sequence and it depends on the number of itemsets each sequence has, which is also saved.

Figure 1 illustrates the data representation followed by the proposed approach for the sample transactional dataset shown in Table 1. The vertical representation (see Fig. 1a) includes eight different lists of indices, one per item in data. Since each index denotes the data record (sequence) in which the item appears, the length of the list is the frequency of each item in data. Hence, the item $\{a\}$ appears twice in the dataset (first and second sequence). The item $\{b\}$ also appears twice in the dataset (first and second sequence), even when it appears twice in the second sequence, that is, $\langle\{a, b\}, \{b\}, \{g, h\}\rangle$. Similarly, the horizontal data representation (see Fig. 1b) is responsible for storing the itemsets in

a	b	c	d	e	f	g	h
1	1	1	1	1	1	2	2
2	2	3	3	3			

(a) Vertical representation



(b) Horizontal representation



(c) Number of itemsets in each sequence

Fig. 1 Data representations of the proposed algorithm

which each item appears. This data representation is as if all the sequences (IDs 1, 2, 3, etc.) were placed in a single row (single sequence) and we consider the place in which the item appears. For example, the item $\{a\}$ appears in the first itemset (sequence with ID 1) and the first itemset of the sequence with ID 2. In other words, $\{a\}$ appears in the first and fourth itemsets from Table 1 if all the sequences were placed in a row. Additionally, the item $\{b\}$ appears in the first itemset (sequence with ID 1) as well as in the first and second itemsets of the sequence with ID 2. In other words, $\{b\}$ appears in the first, fourth and fifth itemsets from

Table 1. Finally, the proposed data representation needs to store the number of itemsets included in each sequence (see Fig. 1c). The accumulated sum is maintained so the last value corresponds to the number of itemsets in data. This accumulated sum is really useful to determine whether the horizontal representation values belong to one or different sequences. In other words, taking the vector of values shown in Fig. 1c, any horizontal data representation value in the range $[1, 3]$ belongs to the first sequence; any horizontal data representation value in the range $[4, 6]$ belongs to the second sequence; and any value in the range $[7, 8]$ belongs to the third sequence.

The proposed data representation is really useful to compute the frequency in a fast way. The frequency of a single item is simply computed as the length of the vertical representation: the frequency of $\{a\}$ is 2, the frequency of $\{b\}$ is 2, etc. Additionally, the frequency of a sequence (including itemsets) is computed through both the vertical and horizontal representations. Let us consider the sequence $s = \langle \{c\}, \{d, e\} \rangle$. For this sequence, the intersection of the vertical data representation is performed for each single item, resulting as indices 1 and 3 (see Fig. 1a). At this point, it is required to check if every itemset in such sequences is also satisfied. The itemset $\{c\}$ appears in indices 2 and 7 according to the horizontal representation (see Fig. 1b). Due to the resulting indices from the vertical representation were 1 and 3, we have to check the first and third ranges of values from Fig. 1c as follows: $2 \in [1, 3]$ and $7 \in [7, 8]$. As a result, the itemset $\{c\}$ is satisfied in the first and third sequence. Let us do the same for the itemset $\{d, e\}$, which appears in indices 3 and 8 according to the horizontal representation (see Fig. 1b). Again, due to the resulting indices from the vertical representation were 1 and 3, we have to check the first and third ranges of values from Fig. 1c as follows: $3 \in [1, 3]$ and $8 \in [7, 8]$. As a result, the sequence s appears twice in data: first and third sequences.

Encoding Criterion

The proposed algorithm uses an encoding vector of variable length to represent each individual or solution to the problem. The vector includes the set of items that belong to the represented solution. Such a set of items, in turn, is grouped into subsets that represent the itemsets within a sequence. The proposed encoding criterion includes the only restriction that the same item cannot appear twice in the same sequence. Since the algorithm was proposed for mining sequences of subjects (items) ordered by semesters (itemsets), it does not make sense to include the same subject twice in a sequence. For a matter of clarification, let $I = \{a, b, c, d\}$ be a sample set of items (subjects). A valid sequence is $s = \langle \{a, b\}, \{c\}, \{d\} \rangle$, denoting that a student passed the subjects a and b in a semester. Then, a semester

later, the student passed the subject c . Finally, in a following semester, the student passed the subject d . An invalid solution would be $s = \langle \{a, b\}, \{a\} \rangle$ since once the subject a is passed by a student there is no sense in enrolling it again.

The proposed encoding criterion is easily adapted to the data representation since each item has two different pointers, one to the corresponding list of sequences (vertical data representation) and other to the list of itemsets (horizontal data representation). Thus, to determine in which sequences the items appear it only performs an intersection of the lists (vertical data representation) associated with the items in the sequence. Additionally, to obtain which itemsets appear in the sequences, it only has to perform an intersection of the lists (horizontal data representation) associated with the items for each itemset in the sequence. In order to clarify this methodology, let us consider again the valid sequence $s = \langle \{a, b\}, \{c\}, \{d\} \rangle$. The sequences in which the items belonging to s appear are obtained by intersecting the vertical representation of a, b, c and d . Additionally, the intersection of the lists obtained from the horizontal data representation returns the sequences in which the itemsets in s are satisfied. Such itemsets are $\{a, b\}, \{c\}$, and $\{d\}$.

(ES)²P Algorithm

The proposed (ES)²P algorithm comprises three main procedures, which were properly designed to the problem at hand. Descriptions of all these procedures as well as how they are combined to form the whole algorithm can be found below.

- Initial solutions.** The proposed algorithm creates the initial set of solutions randomly, each solution is a sequence including random itemsets. The number of itemsets that each solution (sequence) may include is limited by a maximum predefined value. Each itemset includes, in turn, a random subset of items from I . A maximum number of items per itemsets is also predefined. It is finally important to highlight that an item $i \in I$ cannot appear more than once in a sequence, as it was previously described in [Encoding Criterion](#).
- Evaluation procedure.** This procedure is responsible for assigning a fitness value F to each individual or solution s . The evaluation procedure calculates how close a given solution is to the optimum solution on a dataset Ω . In the proposed approach, F for a solution s is formally defined based on the support of s and the GR (it was previously described in Eq. 2, [Preliminaries](#)) of s on the dataset Ω as shown in Eq. 3. F is defined in the range $[0, 1]$ and the best solutions are close to 1. Ω is split into two groups, that is, Ω_1 for good students (those that obtained a high final mark) and Ω_2 for not so good students. The fitness value F is based on the frequency of s in the subset of good students and the normalized

growth rate obtained by s . In other words, a solution s is good if it represents a high percentage of good students and a low percentage of not so good students. Finally, it is important to clarify that, in those situations where the difference in the frequencies between Ω_1 and Ω_2 is maximum, that is, $GR(s, \Omega_1, \Omega_2) = \infty$, only the frequency of s in Ω_1 is considered to compute F .

$$F(s, \Omega_1, \Omega_2) = \frac{\text{support}(s, \Omega_1)}{GR(s, \Omega_1, \Omega_2)} \times (GR(s, \Omega_1, \Omega_2) - 1) \tag{3}$$

For a matter of clarification, let us consider a sample dataset (see Table 2) that is divided into Ω_1 (good students) and Ω_2 (not so good students). A sample solution $s_1 = \langle \{a, b\}, \{c\} \rangle$ appears in 60% of the good students (two first sequences as well as the last sequence in Ω_1). It is mathematically represented as $\text{support}(s_1, \Omega_1) = \frac{3}{5} = 0.6$. This solution s_1 appears in 20% of not so good students (third sequence in Ω_2), also denoted as $\text{support}(s_1, \Omega_2) = \frac{1}{5} = 0.2$. It implies that $GR(s_1, \Omega_1, \Omega_2) = \frac{0.6}{0.2} = 3$ and, therefore, $F(s_1, \Omega_1, \Omega_2) = \frac{0.60}{3} \times (3 - 1) = 0.40$. Let us now consider an additional sample solution $s_2 = \langle \{a, b\}, \{c\}, \{e\} \rangle$. Its support value in Ω_1 is calculated as $\text{support}(s_2, \Omega_1) = \frac{3}{5} = 0.6$, whereas in Ω_2 it is calculated as $\text{support}(s_2, \Omega_2) = \frac{0}{5} = 0$. Hence, $GR(s_2, \Omega_1, \Omega_2) = \frac{0.6}{0} = \infty$ and therefore, the fitness value F is only computed as the frequency of s_2 in Ω_1 or $F(s_2, \Omega_1, \Omega_2) = \text{support}(s_2, \Omega_1) = 0.60$.

- Genetic operators.** The proposal includes two genetic operators: the crossover operator, which focuses on exploiting current individuals by examining their neighbors; the mutation operator, which aims to diversify the search process and to explore new areas in the search space.

Table 2 Sample database including sequences for good and not so good students

Sequence ID	Ω_1 (excellent students)
1	$\langle \{a, b\}, \{c\}, \{d, e, f\} \rangle$
2	$\langle \{a, b\}, \{c\}, \{e\} \rangle$
3	$\langle \{a\}, \{b, c, d, e\} \rangle$
4	$\langle \{c, d\}, \{g, h, k\}, \{l, m\} \rangle$
5	$\langle \{a, b, j\}, \{h, i, k\}, \{c\}, \{d, e\} \rangle$
Sequence ID	Ω_2 (not excellent students)
1	$\langle \{b\}, \{a, c\}, \{f\} \rangle$
2	$\langle \{b, d\}, \{a, c\}, \{d, e\} \rangle$
3	$\langle \{d\}, \{a, b\}, \{c, e\} \rangle$
4	$\langle \{e\}, \{f, g, h\}, \{m\} \rangle$
5	$\langle \{a, d\}, \{b\}, \{k, l\} \rangle$

Algorithm 1 Pseudocode for crossover operator

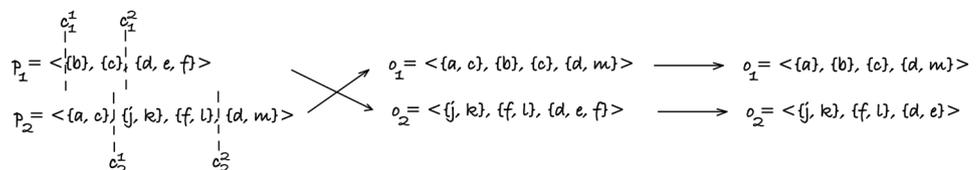
Input Parent individuals p_1 and p_2
Output Offspring individuals o_1 and o_2

- 1: $m_{p_1} \leftarrow$ number of itemsets in p_1
- 2: $m_{p_2} \leftarrow$ number of itemsets in p_2
- 3: $c_1^1, c_1^2 \leftarrow$ random cut-points in p_1
- 4: $c_2^1, c_2^2 \leftarrow$ random cut-points in p_2
- 5: **if** $\text{random}(0, 1) \leq 0.5$ **then**
- 6: $o_1^{(c_1^1, c_1^2)} \leftarrow p_1^{(c_1^1, c_1^2)}$
- 7: $o_2^{(c_2^1, c_2^2)} \leftarrow p_2^{(c_2^1, c_2^2)}$
- 8: **for** each itemset $X_j \mid j \in [0, c_1^1] \cup (c_1^2, m_{p_1}]$ **do**
- 9: $o_2 \leftarrow o_2 \cup \{i \in X_j \mid i \notin o_2\}$
- 10: **end for**
- 11: **for** each itemset $X_j \mid j \in [0, c_2^1] \cup (c_2^2, m_{p_2}]$ **do**
- 12: $o_1 \leftarrow o_1 \cup \{i \in X_j \mid i \notin o_1\}$
- 13: **end for**
- 14: **else**
- 15: $o_1^{[0, c_1^1] \cup (c_1^2, m_{p_1}]} \leftarrow p_1^{[0, c_1^1] \cup (c_1^2, m_{p_1}]}$
- 16: $o_2^{[0, c_2^1] \cup (c_2^2, m_{p_2}]} \leftarrow p_2^{[0, c_2^1] \cup (c_2^2, m_{p_2}]}$
- 17: **for** each itemset $X_j \mid j \in (c_1^1, c_1^2]$ **do**
- 18: $o_2 \leftarrow o_2 \cup \{i \in X_j \mid i \notin o_2\}$
- 19: **end for**
- 20: **for** each itemset $X_j \mid j \in (c_2^1, c_2^2]$ **do**
- 21: $o_1 \leftarrow o_1 \cup \{i \in X_j \mid i \notin o_1\}$
- 22: **end for**
- 23: **end if**
- 24: **return** o_1, o_2

The crossover genetic operator (see Algorithm 1) combines generic material of two solutions to act as parents (p_1 and p_2) to generate offspring (o_1 and o_2). This operator works as a two-points crossover by considering the itemsets as feasible points. Hence, it is not possible to split any itemset within p_1 or p_2 . An additional requirement of this operator is that the crossover points cannot produce the whole individual. This two-points crossover operator works differently depending on a probability. On the one hand, it leaves the itemsets of p_1 within the two cut-points unaltered (see lines 5 to 14, Algorithm 1). It adds the itemsets that appear out of the range of the cut-points of p_2 , but removing those items that were already in the itemsets obtained from p_1 . Additionally, it does the same for p_2 , that is, it takes the itemsets within the cut-points and adds those itemsets out of the cut-points from p_1 . Again, those items that were already included due to p_2 are removed. As a result, no item can appear

more than once either in o_1 or in o_2 . On the other hand, the genetic operator leaves the itemsets of p_1 and p_2 outside the two cut-points unaltered (see lines 14 to 23, Algorithm 1). It then adds the corresponding itemsets within the two cut-points by removing repeated items. As a matter of clarification, Fig. 2 illustrates an example of the proposed crossover operator. Let us consider the following individuals taken from a sample dataset (see Table 2) to act as parents $p_1 = \langle \{b\}, \{c\}, \{d, e, f\} \rangle$, $p_2 = \langle \{a, c\}, \{j, k\}, \{f, l\}, \{d, m\} \rangle$, and the cut-points marked as dotted lines. Let us also consider the range within the cut-points to be copied unaltered. Thus, o_1 is first initialized as $\langle \{b\}, \{c\} \rangle$. The itemsets outside the cut-points from p_2 are then added to o_1 , that is, $\{a, c\}$ and $\{d, m\}$. Since the item c is already in $\langle \{b\}, \{c\} \rangle$, then it is removed and the resulting offspring is $o_1 = \langle \{a\}, \{b\}, \{c\}, \{d, m\} \rangle$. Finally, o_2 is initialized as $\langle \{j, k\}, \{f, l\} \rangle$ and the itemsets outside the cut-

Fig. 2 Example of crossover operator



points from p_1 are added: $\{d, e, f\}$. Due to f is already in o_2 it is removed and the offspring is finally formed as $\langle \{j, k\}, \{f, l\}, \{d, e\} \rangle$.

set that are already included in p are not considered; 2) remove a random itemset from p (see lines 16 to 18); 3) replace a random itemset from p by a new one

Algorithm 2 Pseudocode for mutation operator

Input Parent individual p
Output New individual o

```

1:  $m_p \leftarrow$  number of itemsets in  $p$ 
2: if  $\text{random}(0, 1) \leq 0.5$  then
3:    $c \leftarrow$  random cut-point in  $p$ 
4:    $X \leftarrow$  generate random itemsets
5:   if  $\text{random}(0, 1) \leq 0.5$  then
6:      $o \leftarrow p^{[0,c]} \cup \{i \in X \mid i \notin p^{[0,c]}\}$ 
7:   else
8:      $o \leftarrow \{i \in X \mid i \notin p^{(c,m_p)}\} \cup p^{(c,m_p)}$ 
9:   end if
10: else
11:    $r \leftarrow \text{random}(0, 1)$ 
12:   if  $r \leq 0.33$  then
13:      $X \leftarrow$  generate a random itemset
14:      $c \leftarrow$  generate a random position within  $p$ 
15:      $o \leftarrow p^{[0,c]} \cup \{i \in X \mid i \notin p\} \cup p^{(c,m_p)}$ 
16:   else if  $r \leq 0.66$  then
17:      $X \leftarrow$  select a random itemset from  $p$ 
18:      $o \leftarrow p \setminus X$ 
19:   else
20:      $X^j \leftarrow$  select a random itemset from  $p$ 
21:      $o \leftarrow p \setminus X^j$ 
22:      $X^j \leftarrow$  generate a random itemset
23:      $o \leftarrow o \cup \{i \in X^j \mid i \notin o\}$   $\triangleright$  same position as  $X^j$  but removing duplicated items
24:   end if
25: end if
26: return  $o$ 

```

On the other hand, the mutation genetic operator (see Algorithm 2) has been designed to perform different tasks with a certain probability. It slightly modifies solutions, looks for near neighbors, and seeks for far unexplored areas of the search space, maintaining part of the information of the original sequence (solution or individual). Given an individual, we first give the same probability to apply a more disruptive operator or a subtler one (see line 2, Algorithm 2). The disruptive option (see lines 2 to 10, Algorithm 2) randomly selects a cut-point within the solution and any itemset before that cut-point (see lines 5 to 7) or after it (see lines 7 to 9) is replaced by a set of itemsets generated randomly. It adds a random number of itemsets, but ensuring that the sequence does not exceed the maximum number of itemsets. Those items that are already included in p are not considered. On the contrary, the less disruptive option (see lines 11 to 24, Algorithm 2) provides three different options: 1) to include a new itemset at a random position of the sequence (see lines 12 to 15). Those items within such an item-

randomly generated (see lines 19 to 24). Again, those items within such an itemset that are already included in p are not considered. As a matter of clarification, let us consider the following individual to act as a parent $p = \langle \{b\}, \{c\}, \{d, e, f\} \rangle$, which is a feasible solution from the sample dataset shown in Table 2. Considering the disruptive operator (see lines 2 to 10, Algorithm 2), the cut-point between the second and third itemsets, and removing any itemset on the left of such cut-point, the result is a partial new solution comprising just the itemset $\{d, e, f\}$. After generating the random itemset $X = \{a, b\}$, the resulting solution o obtained from p is $o = \langle \{a, b\}, \{d, e, f\} \rangle$. As for the less disruptive operators, let us consider the last one that replaces an itemset by another randomly obtained (see lines 19 to 24, Algorithm 2). A random itemset $\{b\}$ is chosen from the individual $p = \langle \{b\}, \{c\}, \{d, e, f\} \rangle$, and that itemset is replaced by a new one randomly generated: $\{a, c, h\}$. Due to the item c was already included in p , it is removed from the new itemset and the resulting individual is $o = \langle \{a, h\}, \{c\}, \{d, e, f\} \rangle$.

Algorithm 3 Pseudocode for the proposed (ES)²P algorithm

Input Dataset Ω , population size n , elite size e , max number of generations g , max number of generations without improvement m , probability α for crossover and β for mutation

Output \mathcal{E} elite set

```

1:  $\Omega_1, \Omega_2 \leftarrow$  split the dataset  $\Omega$ 
2:  $\mathcal{P} \leftarrow$  generate  $n$  individuals
3: evaluate  $\mathcal{P}$  according to  $\Omega_1$  and  $\Omega_2$ 
4:  $\mathcal{E} \leftarrow$  take the best  $e$  unrepeated individuals from  $\mathcal{P}$ 
5:  $it \leftarrow 0$ 
6: while  $it < g$  do
7:    $b \leftarrow$  gets the best individual from  $\mathcal{P}$ 
8:    $\mathcal{S} \leftarrow$  applies tournament selector to  $\mathcal{P}$ 
9:    $\mathcal{P} \leftarrow$  applies genetic operators on  $\mathcal{S}$ , using  $\alpha$  and  $\beta$ 
10:   $\mathcal{P} \leftarrow$  applies a restoration operator on  $\mathcal{P}$ 
11:  evaluate  $\mathcal{P}$  according to  $\Omega_1$  and  $\Omega_2$ 
12:  if  $b$  is better than  $\text{best}(\mathcal{P})$  then
13:     $\mathcal{P} \leftarrow \mathcal{P} \setminus \{\text{worst}(\mathcal{P})\}$ 
14:     $\mathcal{P} \leftarrow \mathcal{P} \cup \{b\}$ 
15:  end if
16:   $\mathcal{E} \leftarrow$  best  $e$  unrepeated individuals in  $\{\mathcal{E} \cup \mathcal{P}\}$ 
17:  if Average fitness of  $\mathcal{E}$  does not improve after  $m$  generations then
18:     $\mathcal{P} \leftarrow$  generate  $n$  individuals
19:  end if
20:   $it \leftarrow it + 1$ 
21: end while
22: return  $\mathcal{E}$ 

```

Finally, it is important to combine all the procedures described above to produce the final algorithm (see Algorithm 3). The proposed (ES)²P algorithm maintains a fixed size elite with the best individuals (sequences) produced along the evolutionary process, and this set of best solutions is finally returned. The first step is to split the dataset Ω into two datasets, and then, the algorithm creates the population or the initial set of solutions, which are evaluated according to the fitness function (see lines 2 and 3, Algorithm 3). At this point, the elite is also initialized with the best e individuals from the initial population (see line 4), and the number of generations (iterations of the algorithm) is set to 0. An iterative process starts and it is performed for a number of generations g (see lines 6 to 21, Algorithm 3). In each iteration, the algorithm performs as follows. First, a set of individuals are selected from \mathcal{P} to act as parents. This selection procedure is carried out by a tournament selector of size 2. The set \mathcal{P} is used to apply genetic operators by considering an α probability for the crossover, and a β probability for the mutation. Such genetics operators were already described in Algorithms 1 and 2. Right after the application of the genetic operators, a restoration operator is performed (see line 9) to check that invalid solutions are not formed so individuals can be evaluated again (see line 11). The population is then updated by replacing the previous population by the set of offspring obtained in the current generation. At this point, the best individual is never lost so if it is not in the new population then it is taken from the previous one (see lines

12 to 15, Algorithm 3). The following step carried out by the proposed algorithm is to update the elite set \mathcal{E} with the best e unrepeated solutions found along the evolutionary process (see line 16).

Last but not least, it is important to highlight that in the event that the algorithm is stuck (i.e., the elite does not improve after m generations), the current population restarts

Table 3 Sample set \mathcal{E} of sequences returned by the algorithm and their F_s values, paths s_t already followed by some sample students, and credit hours c_i of each course

Sequence ID	\mathcal{E}	F_s	
1	$\langle \{a\}, \{b, c\}, \{d\} \rangle$	0.8	
2	$\langle \{a, e\}, \{c\} \rangle$	0.7	
3	$\langle \{b\}, \{g, d\}, \{f\} \rangle$	0.6	
4	$\langle \{k\}, \{l\} \rangle$	0.5	
Student ID	s_t		
1	$\langle \{a, l\}, \{b, c, e\}, \{k\}, \{g, d\}, \{f\} \rangle$		
2	$\langle \{l\}, \{a, e\}, \{c, d\}, \{f\} \rangle$		
3	$\langle \{k, a\}, \{b, c\} \rangle$		
Course	c_i	Course	c_i
a	3	f	1
b	2	g	2
c	3	j	2
d	2	k	2
e	3	l	1

(lines 20 to 24, Algorithm 3). The population \mathcal{P} is formed by n random individuals, and the crossover and mutation probabilities are also reset to the default values. Finally, once the maximum number of generations is reached, the elite population is returned (see line 27).

Resulting Set

The previously described (ES)²P algorithm is really useful to extract not only paths followed by students during the degree but also to extract paths previous to some specific courses. Hence, the idea would be the same, but the input dataset is properly obtained by considering the full paths or the specific subpaths before the specific course. The resulting set \mathcal{E} , which is given by the elite of the algorithm for a specific purpose (for example to extract paths that reach to a specific objective course j) is key to perform course recommendations. Giving a course j , the recommendation is based on the analysis of the sequences that describe the paths followed by excellent students in the course j .

The set \mathcal{E} of sequences returned by the algorithm is ordered by the fitness value F_s for each $s \in \mathcal{E}$. Additionally, each item i within a sequence s has associated a $F_i^s = F_s \times c_i$, being c_i the credit hours of the course i . Thus, those courses with a higher number of credit hours are more important than those with lower credit hours. Once a new student t is analyzed by the system, his/her complete path s_t is taken into account, and every sequence $s \in \mathcal{E}$ is activated for t if $s \subseteq s_t$, i.e., if the student passed the courses in the same order denoted by s . A recommendation index d_t^j for the student t to take the course j is calculated based on Eq. 4. The numerator sums the highest F_i^s value for items in the activated sequences, i.e., in those sequences $s \subseteq s_t$. Finally, to provide a recommendation index in the $[0, 1]$ range, the denominator sums the maximum F_i^s value for items in all sequences in the

set. Values of d_t^j closer to 1 means that the student t is prepared to enroll in course j due to its background. Values of d_t^j closer to 0 means that this course should not be taken at this moment. Applying the proposed recommendation index to a given student t on all the courses returns the set of courses that are more appropriate for t .

$$d_t^j = \frac{\sum_{i \in s | s \subseteq s_t} \max(F_i^s | s \subseteq s_t)}{\sum_{i \in s | s \in \mathcal{E}} \max(F_i^s | s \in \mathcal{E})} \tag{4}$$

For a matter of clarification, let us consider the sample set \mathcal{E} shown in Table 3, which is ordered by the fitness value F_s . Let us also consider three sample students and the correspondence of credit hours for each of the courses. Considering the course j as objective, the first student $t = 1$ activates the sequences #1 and #3 from \mathcal{E} , since $s_1 \in \mathcal{E} \subseteq s_t$ and $s_3 \in \mathcal{E} \subseteq s_t$. Then, for each item i in $s_1 \cup s_3$, its maximum F_i^s is obtained, and added to calculate the recommendation index d_t^j . In the case of student $t = 1$, the numerator of d_t^j would be: $0.8 \times 3 + 0.8 \times 2 + 0.8 \times 3 + 0.8 \times 2 + 0.6 \times 2 + 0.6 \times 1 = 9.8$. Note that from s_3 only two courses f and g have incremented the index, since b and d were already present in s_1 with higher F_i^s value. Additionally, the denominator is calculated as if all the items in \mathcal{E} are satisfied. Hence, according to d_t^j (see Eq. 4), the recommendation index for student $t = 1$ would be $d_1^j = \frac{9.8}{13.4} = 0.731$. On the other hand, the recommendation index value for the other two students ($t = 2$ and $t = 3$) and the course j is $d_2^j = 0.112$, and $d_3^j = 0$. Note that student $t = 3$ does not match any of the sequences in the resulting set so the algorithm does not recommend him/her to enroll in course j at all.

Experimental Analysis

This section presents the experimental study, describing first the experimental set up. This section analyzes the performance of the proposed algorithm, which is finally compared to exhaustive search algorithms to demonstrate that the evolutionary process makes sense.

Experimental Setup

All the exhaustive search algorithms used in this comparison are available in the SPMF library [43]: Spade [13], Spam [14], PrefixSpan [15], CM-Spade [16] and CM-Spam [16]. Additionally, the algorithms TKS [17] and TSP [18] are also considered since they do not require any frequency threshold, and their aim is to extract the top- k most frequent itemsets from data. The experiments are carried out on a set of 13 real datasets (see Table 4) taken from King Abdulaziz University including information about

Table 4 Datasets and their main characteristics

Faculty	#Records	Length	#Courses
Arts	3,892	49.49	375
Business	1,413	46.80	216
Communication & Media	483	48.10	174
Computing & Inf. Tech.	144	51.01	109
Design & Arts	167	54.67	155
Economics & Admin.	2,206	48.31	387
Engineering	365	58.73	364
Engineering Rabeg	102	62.99	217
Home Economics	614	45.29	264
Information Technology	284	54.70	233
Law	946	49.18	183
Sciences & Arts	893	52.26	283
Sciences	1,928	51.87	467

Table 5 Average fitness values obtained by the proposed (ES)²P algorithm considering different population sizes (n) and number of generations without improvement (m) to reset the population. Best results are in bold type-face

Faculty	$n = 100$		$n = 200$		$n = 300$		$n = 400$		$n = 500$	
	$m = 50$	$m = 100$	$m = 50$	$m = 100$	$m = 50$	$m = 100$	$m = 50$	$m = 100$	$m = 50$	$m = 100$
Arts	0.201	0.202	0.208	0.207	0.210	0.212	0.215	0.214	0.222	0.219
Business	0.150	0.152	0.173	0.184	0.197	0.209	0.207	0.218	0.219	0.226
Communication & Media	0.335	0.339	0.372	0.424	0.409	0.430	0.433	0.464	0.427	0.487
Computing & Informat. Technology	0.374	0.386	0.386	0.386	0.399	0.410	0.408	0.415	0.414	0.419
Design & Arts	0.250	0.259	0.264	0.273	0.273	0.274	0.275	0.282	0.281	0.285
Economics & Administration	0.417	0.419	0.419	0.420	0.419	0.420	0.420	0.421	0.421	0.422
Engineering	0.243	0.247	0.250	0.251	0.251	0.251	0.251	0.251	0.251	0.252
Engineering Rabeg	0.293	0.294	0.297	0.302	0.298	0.301	0.302	0.307	0.305	0.309
Home Economics	0.315	0.323	0.339	0.338	0.348	0.345	0.351	0.350	0.351	0.351
Information Technology	0.367	0.368	0.379	0.378	0.395	0.399	0.398	0.407	0.403	0.407
Law	0.264	0.267	0.268	0.273	0.275	0.278	0.277	0.282	0.280	0.284
Sciences & Arts	0.157	0.156	0.184	0.173	0.189	0.183	0.199	0.191	0.203	0.201
Sciences	0.160	0.162	0.170	0.173	0.173	0.172	0.174	0.176	0.176	0.179

different faculties: sequences of courses taken by students. Additionally, all the gathered data were divided into 13 groups or datasets, one per faculty. #Records stands for the number of students, *Length* is the average number of subjects taken by the students to obtain the degree, and finally, #Courses is the number of different courses that each faculty provides. Last but not least, it is important to highlight that each dataset (faculty) is split into two: Ω_1 includes the 30% of students with best GPA in the degree; Ω_2 includes the rest of students. All the experiments are performed on a machine with 6 Intel Xeon E5-2620 CPUs at 2.10 GHz and 64 GB of RAM. The experiments are run ten times, and the average results are considered to reduce environment variations.

Analysis of the Proposed (ES)²P Approach

The goal of this first analysis is to demonstrate how well the proposal behaves on multiple datasets and to determine the best values for the hyperparameters, that is, those that provide best fitness values requiring a lower computational time. Table 5 shows the average results obtained by different combinations of values for the population size (n) and the population restarting after m generations without improvement. A hypothesis testing by means of non-parametric statistical tests has been conducted with the aim of determining whether there exist significant differences in the overall

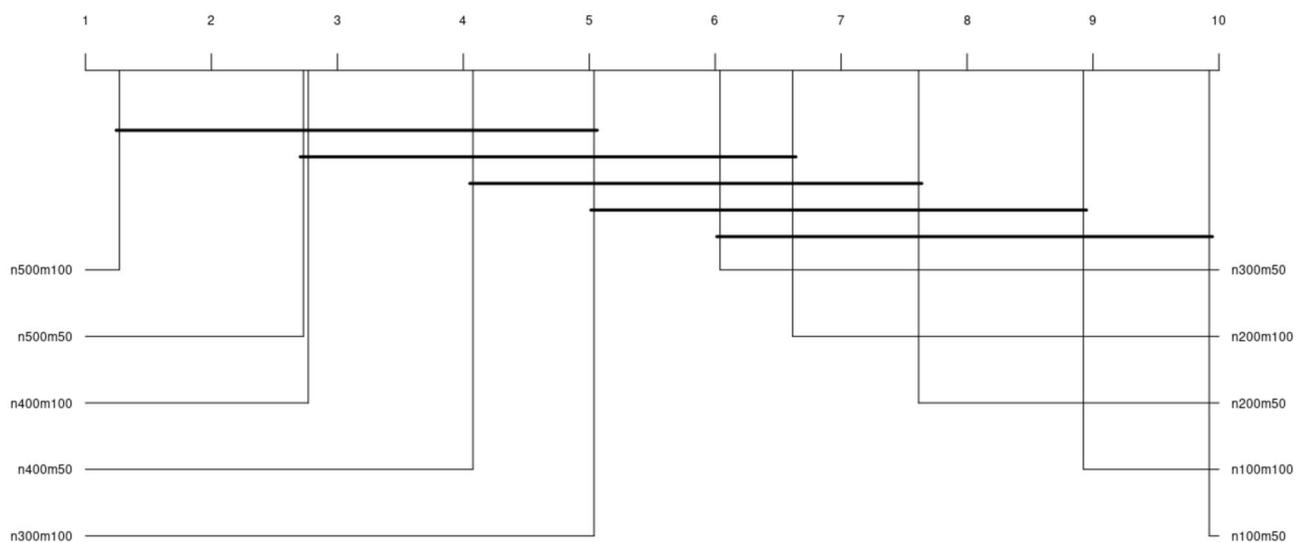
**Fig. 3** Critical difference diagram of different parameter combinations considered. The comparisons were performed using a Shaffer's test

Table 6 Average time in seconds obtained by the proposed (ES)²P algorithm considering different population sizes (*n*) and number of generations without improvement (*m*) to reset the population. Best results are in bold type-face

Faculty	<i>n</i> = 300	<i>n</i> = 400		<i>n</i> = 500	
	<i>m</i> = 100	<i>m</i> = 50	<i>m</i> = 100	<i>m</i> = 50	<i>m</i> = 100
Arts	4.91 × 10 ²	6.55 × 10 ²	6.00 × 10 ²	7.71 × 10 ²	7.41 × 10 ²
Business	3.01 × 10²	3.96 × 10 ²	3.73 × 10 ²	4.44 × 10 ²	4.59 × 10 ²
Communication & Media	1.64 × 10²	2.25 × 10 ²	2.19 × 10 ²	2.76 × 10 ²	2.67 × 10 ²
Computing & Inf. Tech.	1.10 × 10²	1.48 × 10 ²	1.43 × 10 ²	1.86 × 10 ²	1.79 × 10 ²
Design & Arts	1.39 × 10²	1.87 × 10 ²	1.81 × 10 ²	2.33 × 10 ²	2.29 × 10 ²
Economics & Admin.	3.30 × 10 ²	4.51 × 10 ²	4.32 × 10 ²	5.81 × 10 ²	5.24 × 10 ²
Engineering	2.07 × 10²	2.85 × 10 ²	2.72 × 10 ²	3.47 × 10 ²	3.38 × 10 ²
Engineering Rabeg	1.45 × 10²	1.97 × 10 ²	1.89 × 10 ²	2.45 × 10 ²	2.38 × 10 ²
Home Economics	1.72 × 10²	2.27 × 10 ²	2.28 × 10 ²	2.82 × 10 ²	2.80 × 10 ²
Information Technology	1.45 × 10²	1.95 × 10 ²	1.91 × 10 ²	2.41 × 10 ²	2.36 × 10 ²
Law	3.15 × 10 ²	4.63 × 10 ²	4.20 × 10 ²	5.54 × 10 ²	5.57 × 10 ²
Sciences & Arts	2.04 × 10²	2.74 × 10 ²	2.65 × 10 ²	3.38 × 10 ²	3.28 × 10 ²
Sciences	3.01 × 10²	3.99 × 10 ²	4.00 × 10 ²	5.02 × 10 ²	4.85 × 10 ²

performance for the aforementioned combination of values. The Friedman’s test [44] has been used to analyze the general differences, whereas the Shaffer’s post hoc test [45] has been employed to perform all pairwise comparisons. The Friedman’s test detected that there were general statistical differences in the ten combinations of values at a significance level of $\alpha = 0.01$, rejecting the null hypothesis with a *p*-value smaller than $2.2e-16$. Then, the Shaffer’s post hoc test was performed to detect where these significant differences were located. The results for this post hoc test, at a significance level of $\alpha = 0.01$, are summarized through the critical difference diagram shown in Fig. 3, illustrating that

the values $n = 500$ and $m = 100$ produce the best values. However, according to the post hoc test, no statistical difference is found among *n* values between 500 and 300, and *m* values between 50 and 100, being the only exception the combination of $n = 300$ and $m = 50$. At this point, it is interesting to analyze the runtime (see Table 6) for these six combinations of parameters that present the same performance, with a statistical significance of 99%. The Friedman’s test revealed statistical differences in these combination of values at a significance level $\alpha = 0.01$, thus rejecting the null hypothesis with a *p*-value smaller than $5.074e-10$. Finally, the Shaffer’s post hoc test at a significance level of $\alpha = 0.01$

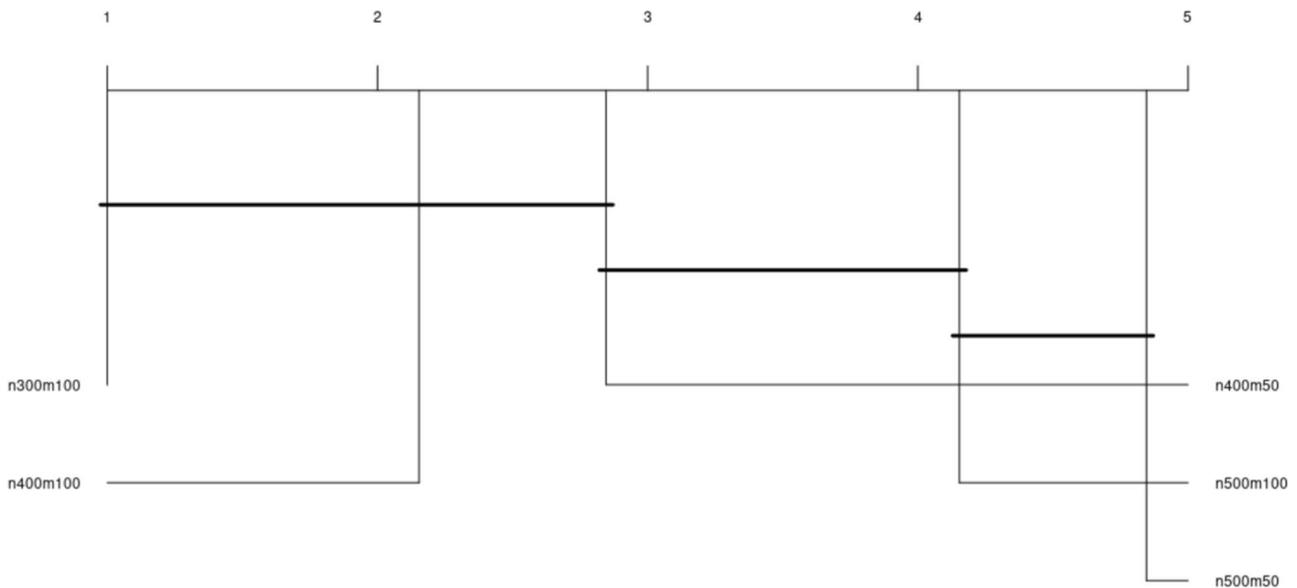
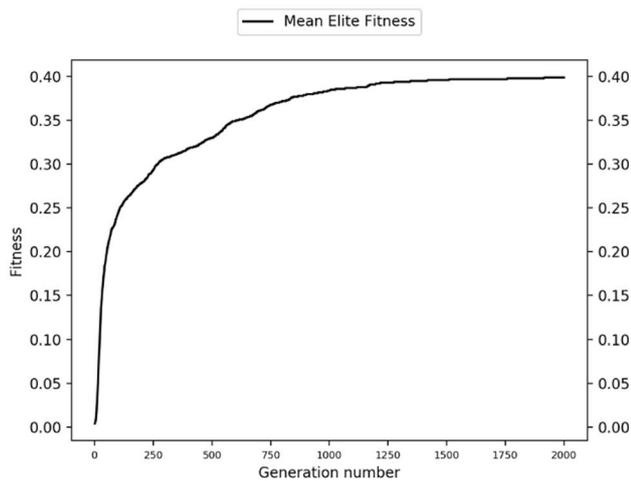
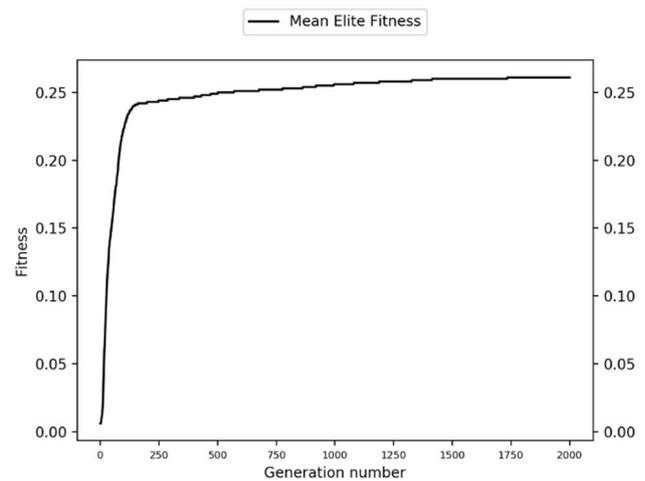


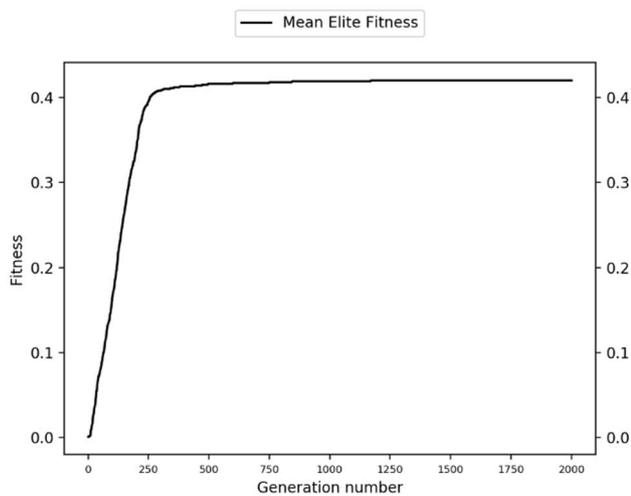
Fig. 4 Critical difference diagram of different execution times obtained for the parameter combinations considered. The comparisons were performed using a Shaffer’s test



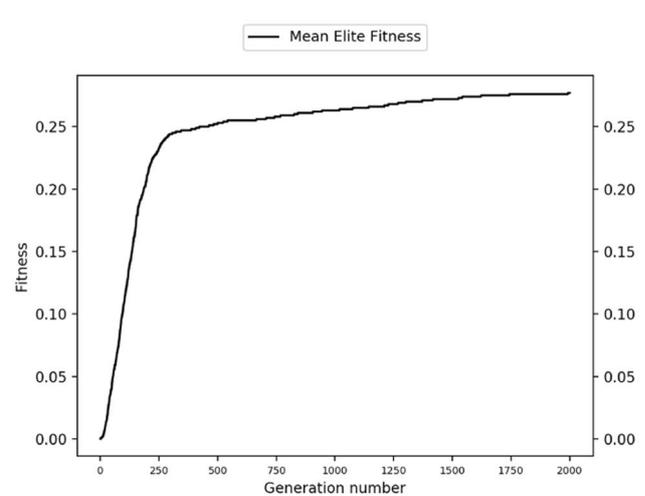
(a) Computing & Information Technology



(b) Design & Art



(c) Economics & Administration



(d) Law

Fig. 5 Analysis of the convergence of (ES)²P on different datasets

(see Fig. 4) revealed no significant differences for $n = 300$ and any m value, as well as $n = 400$ and $m = 100$. According to the results shown in Table 6, the combination of parameters $m = 100$ and $n = 300$ presents the best runtime. Taking all the above into consideration, we recommend the previous combination of parameters.

Let us continue now with the analysis of the convergence of the proposed approach. Figure 5 shows how the algorithm behaves on four different datasets: Computing & Information Technology; Design & Art; Economics & Administration; Law. The results on this heterogeneous group of datasets ($\#Records$ varies from 144 to 2,206; $Length$ is between 48.3 and 54.67; and $\#Courses$ varies from 109 to 387. See Table 4) demonstrate that the convergence of the algorithm

is high on different scenarios and it is around 1,500 generations for which the algorithm does not widely improve the results. Thus, in order to avoid spending time and computational resources on little fitness improvements, the number of generations is set to 1,500. Last but not least, the crossover and mutation probability values are also fixed to 0.8 and 0.3, respectively. With the aim of providing a better description of the experimental study, the readers could find a further study on the combination of probability values at the website <http://www.uco.es/kdis/course-recommendation>. In summary, to obtain the best combination of parameter values, more than 50 parameter configurations were considered, resulting in more than 6,500 executions. Additionally, 10 independent runs were performed for each dataset and parameter

Table 7 Number of solutions and average fitness value returned by exhaustive search algorithms (considering different support threshold values) and the proposed (ES)²P approach. *Memory* stands for memory problems when running (out of memory). The average fitness value was calculated by taking the best 50 solutions based on the fitness value

Faculty	#Solutions				
	0.5	0.6	0.7	0.8	(ES) ² P
Arts	2,642	811	749	583	50
Business	59,336	4,705	1,397	604	50
Communication & Media	184,168	93,667	45,942	14,952	50
Computing & Inf. Tech.	<i>Memory</i>	9,932,208	3,589,091	820,513	50
Design & Arts	4,091	2,413	1,730	751	50
Economics & Admin.	159,850	43,706	8,996	248	50
Engineering	325,611	19,824	10,384	6,764	50
Engineering Rabeg	<i>Memory</i>	8,512,968	1,279,911	180,360	50
Home Economics	<i>Memory</i>	1,661	1,147	790	50
Information Technology	8,027,089	510,522	112,565	38,561	50
Law	484,583	145,763	79,418	43,112	50
Sciences & Arts	1,164	674	65	59	50
Sciences	18,220	1,954	1,397	1,226	50
Average Fitness					
Faculty	0.5	0.6	0.7	0.8	(ES) ² P
Arts	0.313	0.313	0.310	0.225	0.208
Business	0.303	0.302	0.284	0.202	0.195
Communication & Media	0.659	0.659	0.659	0.640	0.412
Computing & Inf. Tech.	<i>Memory</i>	0.558	0.558	0.558	0.401
Design & Arts	0.295	0.288	0.220	0.158	0.267
Economics & Admin.	0.457	0.457	0.433	0.158	0.420
Engineering	0.266	0.210	0.145	0.126	0.249
Engineering Rabeg	<i>Memory</i>	0.464	0.464	0.464	0.300
Home Economics	<i>Memory</i>	0.437	0.436	0.333	0.340
Information Technology	0.446	0.416	0.323	0.289	0.389
Law	0.326	0.303	0.281	0.261	0.273
Sciences & Arts	0.332	0.323	0.040	0.034	0.172
Sciences	0.309	0.299	0.292	0.290	0.168

configuration to study the algorithm’s performance due to its stochastic component.

(ES)²P Against Other Sequential Pattern Mining Algorithms

This second analysis aims to study how well the proposed (ES)²P algorithm behaves when it is compared to existing exhaustive search algorithms. First, we analyze the number of solutions returned by any exhaustive search algorithm (results are exactly the same for any algorithm, as expected) when different frequency thresholds are considered (see Table 7). At this point, it is required to highlight that our proposal returns exactly the same for any dataset since it obtains the best 50 solutions found. As it is shown, the number of solutions extracted by exhaustive search algorithms highly vary and it depends on the dataset. This number varies between 59 to 148,461,607 solutions. Second, we analyze the average fitness value

obtained by the algorithms on different support threshold values (see Table 7). It is important to remark that this is not the average frequency value but the fitness value already described in (ES) P Algorithm. Additionally, since the number of solutions is completely different, we have taken the best 50 solutions to perform a fair comparison with regard to the proposal. As it is expected, exhaustive search algorithms obtained the best results (bold type-face) and, the lower the support threshold value the better results since a wider number of solutions are analyzed. However, for some datasets, our proposal achieved better results in average fitness: Design & Arts; Economics & Administration; Engineering; Home Economics; Information Technology; Law; Sciences & Arts. This interesting behavior occurs due to fitness function depends on the GR measure, which can be high for small support values. Thus, given a specific support threshold value, extremely good results for the fitness value might be missed. On the contrary, the proposal guides the searching process by the fitness value, achieving better results.

Table 8 Runtime, in seconds, required by each algorithm on different datasets and considering different support threshold values (0.5, 0.6, 0.7 and 0.8). Our proposal does not require any threshold. *Memory* stands for memory problems when running (out of memory)

Support threshold 0.5						
Faculty	CM-Spade	CM-Spam	PrefixSpan	Spade	Spam	(ES) ² P
Arts	112	113	111	111	115	372
Business	415	410	446	405	451	208
Communication & Media	619	555	585	587	584	120
Computing & Inf. Tech.	<i>Memory</i>	<i>Memory</i>	<i>Memory</i>	<i>Memory</i>	<i>Memory</i>	82
Design & Arts	10	10	10	10	10	103
Economics & Admin.	2,093	2,201	1,983	2,019	2,299	260
Engineering	723	705	713	728	707	153
Engineering & Rabeg	<i>Memory</i>	<i>Memory</i>	<i>Memory</i>	<i>Memory</i>	<i>Memory</i>	106
Home Economics	<i>Memory</i>	<i>Memory</i>	<i>Memory</i>	<i>Memory</i>	<i>Memory</i>	130
Information Technology	16,025	16,439	16,059	16,614	16,706	108
Law	2,460	2,651	2,774	2,516	2,852	267
Sciences & Arts	23	22	23	22	22	153
Sciences	232	238	220	229	232	233
Support threshold 0.6						
Faculty	CM-Spade	CM-Spam	PrefixSpan	Spade	Spam	(ES) ² P
Arts	76	74	78	75	72	372
Business	52	56	53	55	52	208
Communication & Media	292	301	306	291	288	120
Computing & Inf. Tech.	12,032	12,075	12,887	12,198	11,578	82
Design & Arts	8	8	8	8	8	103
Economics & Admin.	577	549	561	565	593	260
Engineering	53	51	55	53	51	153
Engineering & Rabeg	8,610	8,055	8,148	8,138	8,209	106
Home Economics	17	17	17	19	18	130
Information Technology	942	947	942	956	892	108
Law	762	806	783	786	791	267
Sciences & Arts	20	20	20	19	19	153
Sciences	52	51	51	52	50	233
Support threshold 0.7						
Faculty	CM-Spade	CM-Spam	PrefixSpan	Spade	Spam	(ES) ² P
Arts	74	74	75	75	73	372
Business	32	32	32	32	31	208
Communication & Media	156	144	154	160	141	120
Computing & Inf. Tech.	4,270	4,296	4,166	4,528	4,295	82
Design & Arts	7	8	8	7	7	103
Economics & Admin.	142	137	138	142	139	260
Engineering	29	30	29	30	30	153
Engineering & Rabeg	1,175	1,155	1,165	1,163	1,195	106
Home Economics	16	16	16	17	16	130
Information Technology	206	207	208	208	196	108
Law	409	406	421	419	433	267
Sciences & Arts	16	16	17	17	16	153
Sciences	46	46	46	46	45	233
Support threshold 0.8						
Faculty	CM-Spade	CM-Spam	PrefixSpan	Spade	Spam	(ES) ² P
Arts	72	70	71	70	69	372

Table 8 (continued)

Faculty	Support threshold 0.8					(ES) ² P
	CM-Spade	CM-Spam	PrefixSpan	Spade	Spam	
Business	26	27	27	26	25	208
Communication & Media	52	56	52	57	54	120
Computing & Inf. Tech.	914	942	887	946	935	82
Design & Arts	7	6	7	6	7	103
Economics & Admin.	37	38	37	38	37	260
Engineering	22	23	22	23	22	153
Engineering & Rabeg	147	148	158	154	152	106
Home Economics	16	15	15	16	17	130
Information Technology	73	72	72	73	74	108
Law	230	242	230	235	231	267
Sciences & Arts	17	16	18	17	17	153
Sciences	44	43	45	44	46	233

Additionally, let us analyze the runtime required by different algorithms on different support threshold values. In this analysis, we consider a set of exhaustive search algorithms that is denoted as the best ones in the specialized literature [12]: Spade [13], Spam [14], PrefixSpan [15], CM-Spade [16] and CM-Spam [16]. Table 8 shows the runtime, in seconds, required by the algorithms for a support threshold value of 0.5. At this point, it is important to remind that small differences in the average fitness value were obtained (see Table 7): 0.028 in Design & Arts; 0.037 in Economics & Administration; 0.017 in Engineering; 0.057 in Information Technology; 0.053 in Law. Additionally, three datasets cannot be run due to memory problems when using exhaustive search algorithms on such a threshold value (see Tables 7 and 8). In general terms, our proposal needs lower runtimes and these values do not widely vary from dataset to dataset. Huge differences are found on multiple datasets. For example, in Economics & Administration exhaustive search approaches need more than 2,000 seconds, whereas our proposal only needs 260 seconds. In fact, for this specific dataset the difference in the resulting average fitness value was really low (0.457 in exhaustive search algorithms and 0.420 in our proposal). Among all the results, the maximum differences in runtime are found when the Information Technology dataset is considered, since exhaustive search approaches require more than 16,000 seconds whereas our proposal just 108 seconds. Additionally, for this dataset, the difference in average fitness value was really small (0.446 in exhaustive search algorithms and 0.389 in our proposal). In summary, after analyzing all the results for a support threshold value of 0.5, it is possible to assert that the proposed approach is really useful to obtain really good results (according to the average fitness value) in a small quantity of time. Furthermore, this proposal is able to be run on any dataset, whereas exhaustive search approaches fail on some datasets due to memory requirements.

If we continue the analysis for other support threshold values (0.6, 0.7 and 0.8), it is obtained that the higher the threshold value, the lower the runtime required by exhaustive search approaches (see Table 8). However, analyzing the average fitness value (see Table 7), the higher the threshold value, the lower the average fitness value obtained by exhaustive search approaches. In fact, considering a support threshold of 0.8, our proposal obtains better results in seven datasets (see Table 7).

Last but not least, it is important to remark that those algorithms that require a minimum support threshold value to be predefined need an extra (previous) process to determine the exact value. This procedure is not trivial, and generally requires a profound background in the application field. Inexpert and many expert users need to try different thresholds by guessing and re-executing the algorithms once and again until results are good for them [9]. All of this, together with the large runtimes required on different datasets, and the small differences in the resulting average fitness values, let us to the conclusion that our proposal outperforms exhaustive search algorithms.

Comparison to Top-*k* Sequential Pattern Mining Algorithms

This third analysis aims to study how well the proposed (ES)²P algorithm behaves when it is compared to existing exhaustive search algorithms for mining the top-*k* solutions. The main advantage of these approaches is that they do not require a previous study to determine a good threshold value. Additionally, they return the same number of solutions regardless the input dataset, which is easier to be manage by experts. However, the main disadvantage of these approaches is related to the fitness values. Existing algorithms for mining top-*k* sequential patterns were proposed for mining the best results in terms of frequency (support values). Nevertheless, for the problem at hand, the support value cannot establish the importance of the sequence. A

Table 9 Runtime and average fitness value returned by top-*k* search algorithms and the proposed approach considering different *k* values (25, 50, 100, 200)

Faculty	<i>k</i> = 25						<i>k</i> = 50					
	Average Fitness			Runtime			Average Fitness			Runtime		
	TKS	TSP	(ES) ² P	TKS	TSP	(ES) ² P	TKS	TSP	(ES) ² P	TKS	TSP	(ES) ² P
Arts	0.004	0.004	0.208	62	0.004	372	0.004	0.004	0.208	63	0.004	372
Business	0.015	0.015	0.195	25	0.025	208	0.025	0.025	0.195	23	0.025	208
Communication & Technology	0.034	0.249	0.412	11	0.031	120	0.031	0.242	0.412	11	0.031	120
Computing & Inf. Tech.	0.250	0.460	0.401	7	0.235	82	0.235	0.443	0.401	8	0.235	82
Designs & Arts	0.022	0.022	0.267	6	0.022	103	0.022	0.022	0.267	5	0.022	103
Economics	0.001	0.001	0.420	34	0.002	260	0.002	0.002	0.420	36	0.002	260
Engineering	0.043	0.049	0.249	11	0.031	153	0.031	0.035	0.249	12	0.035	153
Engineering Rabeg	0.145	0.159	0.300	5	0.127	106	0.127	0.154	0.300	6	0.154	106
Home Economics	0.003	0.003	0.340	13	0.003	130	0.003	0.003	0.340	13	0.003	130
Information Technology	0.148	0.156	0.389	9	0.147	108	0.147	0.155	0.389	9	0.155	108
Law	0.031	0.036	0.273	21	0.028	267	0.028	0.034	0.273	21	0.034	267
Sciences	0.011	0.011	0.172	17	0.025	153	0.025	0.025	0.172	17	0.025	153
Sciences & Arts	0.002	0.005	0.168	33	0.002	233	0.002	0.003	0.168	34	0.003	233
<i>k</i> = 100												
Faculty	Average Fitness			Runtime			Average Fitness			Runtime		
	TKS	TSP	(ES) ² P	TKS	TSP	(ES) ² P	TKS	TSP	(ES) ² P	TKS	TSP	(ES) ² P
	0.018	0.018	0.208	63	0.065	372	0.065	0.077	0.208	64	0.077	372
Business	0.041	0.041	0.195	23	0.052	208	0.052	0.052	0.195	24	0.052	208
Communication & Technology	0.023	0.224	0.412	11	0.048	120	0.048	0.193	0.412	11	0.193	120
Computing & Inf. Tech.	0.227	0.418	0.401	7	0.192	82	0.192	0.386	0.401	8	0.386	82
Designs & Arts	0.016	0.016	0.267	6	0.031	103	0.031	0.066	0.267	7	0.066	103
Economics	0.002	0.002	0.420	35	0.068	260	0.068	0.067	0.420	36	0.067	260
Engineering	0.021	0.024	0.249	11	0.014	153	0.014	0.017	0.249	11	0.017	153
Engineering Rabeg	0.108	0.139	0.300	5	0.086	106	0.086	0.120	0.300	5	0.120	106
Home Economics	0.023	0.023	0.340	14	0.036	130	0.036	0.033	0.340	13	0.033	130
Information Technology	0.118	0.152	0.389	9	0.076	108	0.076	0.124	0.389	9	0.124	108
Law	0.024	0.031	0.273	21	0.019	267	0.019	0.027	0.273	21	0.027	267
Sciences	0.054	0.054	0.172	17	0.069	153	0.069	0.069	0.172	17	0.069	153
Sciences & Arts	0.002	0.003	0.168	33	0.004	233	0.004	0.005	0.168	33	0.005	233

Table 10 Top 5 complete paths returned by the proposal on Faculty of Business

Paths	Fitness	Support	GR
$\langle\{30\}, \{11, 43\}\rangle$	0.295	0.719	1.694
$\langle\{44\}, \{11, 43\}\rangle$	0.294	0.724	1.685
$\langle\{11, 43\}\rangle$	0.286	0.724	1.654
$\langle\{10, 7\}\rangle$	0.280	0.639	1.781
$\langle\{30\}, \{10\}, \{46\}\rangle$	0.246	0.802	1.442

sequence can be frequent for both excellent and not so good students and, therefore, the GR value is low. Additionally, a sequence can be infrequent for excellent students and zero for not so good students, providing an excellent GR value. This theoretical behavior is tested by running two algorithms that determine the state-of-the-art, that is, TKS [17] and TSP [18], on different datasets. The results (see Table 9) demonstrate that extremely bad results are obtained by these algorithms. The runtime needed by TKS and TSP is much lower than $(ES)^2P$, but the results are useless for the problem at hand (fitness values close to 0).

Cases of Study: Applying $(ES)^2P$ to Course Recommendation

In this section, we propose two different methodologies to apply the proposed $(ES)^2P$ algorithm for course recommendation. First, we propose a methodology for ordering courses that should be taken by students to success in the degree. This methodology, based on the proposed $(ES)^2P$ algorithm, provides full study plans that should be followed by students to reduce the dropout and failure. Second, we propose a methodology for rating courses with the aim of providing the students with advices on which courses should be taken at any specific moment of their degree. The aim is to recommend subjects that best fits to them according to their paths (previous courses). This methodology requires nothing more than the courses (ordered by semesters) already passes by the student that is advised. Last but not least, it is important to remark that courses are represented by IDs in these cases of study to simplify the results. The real name of the courses are explained at the website <http://www.uco.es/kdis/course-recommendation>.

Table 11 Top 5 complete paths returned by the algorithm for the Faculty of Information Technology

Paths	Fitness	Support	GR
$\langle\{38\}, \{35, 47\}\rangle$	0.409	0.565	3.625
$\langle\{6\}, \{35, 47\}\rangle$	0.409	0.565	3.625
$\langle\{50\}, \{39\}, \{35, 47\}\rangle$	0.401	0.541	3.846
$\langle\{19\}, \{35, 47\}\rangle$	0.399	0.565	3.405
$\langle\{51\}, \{35, 47\}\rangle$	0.399	0.565	3.405

Study Plan Recommendation Based on the Best Ordering of Courses

When no course is given, the algorithm extracts discriminative sequential patterns on complete paths carried out by students. As a study case, we have considered two different faculties: Business and Information Technology (see Table 4). As a matter of simplification, we have taken only the top 5 solutions returned by the proposed methodology. However, the whole set of paths obtained is available at the aforementioned website with any extra information. Additionally, the aforementioned website includes information about the real name of the courses.

Let us start with the Faculty of Business study case. Table 10 shows the 5 best solutions found according to the fitness value. The support on the set of good students and the GR value is also available. The path with the best fitness value denotes that more than 70% of the excellent students have passed course 30 in a semester and courses 11 and 43 together in a subsequent semester. This path is satisfied 1.69 times more often in excellent students than in not so good students. A similar behavior is denoted by the second ($\langle\{44\}, \{11, 43\}\rangle$) and the third paths ($\langle\{11, 43\}\rangle$). As a result, it is possible to assert that to take subjects with IDs 11 and 43 in the same semester is a synonymous of being an excellent student in the Faculty of Business. Nevertheless, it is fair to say that no excellent result was obtained in terms of courses that heavily denote a difference between excellent and not so good students. It is mainly due to, for this Faculty, there is not good paths to be performed by students and, generally, all the students equally behave.

Even more interesting are the results obtained on the Faculty of Information Technology (see Table 11). Analyzing the top 5 solutions, we obtain that the courses with id 35 and 47 appear in any of the paths and, in fact, they are studied in the same semester. In any of the cases, all the returned paths presents a behavior that is three times more often for excellent students than for not so good students. For example, focusing on the solution $\langle\{50\}, \{39\}, \{35, 47\}\rangle$, it determines that if a student pass the course with ID 50 in a semester, then in a different semester, such a student pass

Table 12 Top 5 courses recommended to the student $t_1 = \langle\{12, 22, 30, 44, 45, 75\}, \{1, 9, 14, 15, 17, 33\}, \{3, 11, 34\}\rangle$ belonging to the Faculty of Business

Course ID	Course name	$d_{t_1}^j$
49	ACC415	1
88	BLA322	1
166	ACC321	1
169	PE120	0.950
96	MRK303	0.885

Table 13 Top 5 courses recommended to the student $t_2 = \langle \{12, 22, 30, 44, 45, 75\}, \{1, 9, 14, 15, 17, 33\}, \{3, 21\}, \{2, 11, 26, 43\}, \{10, 24, 36, 74, 76\}, \{20, 46, 63, 77, 142\}, \{7, 8, 68, 69, 71\}, \{34\} \rangle$ belonging to the Faculty of Business.

Course ID	Course name	$d_{t_2}^j$
193	COM205	0.841
59	ACC411	0.834
121	MRK322	0.832
169	PE120	0.806
96	MRK303	0.799

the course with ID 39, and then, in a different semester, he/she pass courses with IDs 35 and 47 (in the same semester this time), such a student has 3.8 times more probability to be an excellent student and the end of the degree. Hence, this information is really useful to provide study plans and to analyze why such differences among students when they take such courses in that order.

Course Recommendation Based on the Previous Academic Path

This second study case is related to the recommendation of which courses should be taken by a student in a specific semester according to its path (courses already taken by him/her). The aim is to improve his/her academic success. In this study case, we have considered the same faculties discussed in the previous study case: Business and Information Technology. We have also taken four different students to provide them a recommendation (two of each faculty).

Let us start with a student t_1 from the Faculty of Business, having the path $t_1 = \langle \{12, 22, 30, 44, 45, 75\}, \{1, 9, 14, 15, 17, 33\}, \{3, 11, 34\} \rangle$. Thus, t_1 has passed 6 subjects in a semester, 6 subjects in a posterior semester, and 3 subjects in a subsequent semester. For this student, our proposal recommend to take the courses shown in Table 12 right now. The recommendation index score of the recommended courses are really good (close to the maximum of 1), meaning that

Table 14 Top 5 courses recommended to the student $t_3 = \langle \{5, 8, 9, 18, 25, 45\}, \{4, 12, 13, 23, 28\}, \{1, 10, 15, 21, 51\}, \{3, 6, 14, 27, 74\}, \{26, 29, 75, 78, 84\} \rangle$ belonging to the Faculty of Information Technology

Course ID	Course name	$d_{t_3}^j$
64	CPCS223	1
19	CPIT210	0.957
77	CPCS211	0.946
88	ISLS211	0.920
105	CPIS210	0.883

Table 15 Top 5 courses recommended to the student $t_4 = \langle \{5, 12, 23, 28, 45\}, \{4, 8, 9, 13, 18, 25\}, \{1, 10, 15, 21, 51\}, \{3, 6, 14, 27, 74\} \rangle$ belonging to the Faculty of Information Technology

Course ID	Course name	$d_{t_4}^j$
26	CPCS204	0.940
88	ISLS211	0.920
29	ARAB201	0.901
77	CPCS211	0.874
20	CPIT220	0.851

all (or almost all) students with the same path have obtained excellent marks in such courses. Analyzing what this student really did, we check that one of the recommended courses was taken (course with ID 88). In this course, the student t_1 obtained a GPA that is within the 5.4% of the best GPA obtained for that course among all the students. Thus, it is demonstrated that when a student follows the recommendations, he/she obtains really good GPAs.

Let us now consider a second student t_2 from the same faculty and who has followed the path $t_2 = \langle \{12, 22, 30, 44, 45, 75\}, \{1, 9, 14, 15, 17, 33\}, \{3, 21\}, \{2, 11, 26, 43\}, \{10, 24, 36, 74, 76\}, \{20, 46, 63, 77, 142\}, \{7, 8, 68, 69, 71\}, \{34\} \rangle$. This student is close to finish his/her degree since he/she has completed 8 semesters and 34 different subjects. The top 5 courses recommended by the algorithm and the recommendation index scores are summarized in Table 13. This time, the student did not followed any of the recommended courses and took some courses that were not appropriate at all for him/her. For example, analyzing the path finally followed by such a student, he/she took the courses with ID 72 and 88. Such courses presents a recommendation index score of 0.069 and 0.000, respectively. Thus, such courses were not appropriate for the student t_2 as it is finally proved by the GPA obtained for such courses. In course with ID 72, the student obtained a GPA of 87, which is within the 34.89% of the students (ranked by GPA for that course). Similarly, the student obtained a GPA of 85 in the course 88, which is within the 55.94% of the ranking of students. As it is demonstrated, to follow the recommendation is crucial to obtain good GPAs.

The following analysis is carried out on a different Faculty, that is, Information Technology. For this Faculty, we take a student t_3 that has followed the path $t_3 = \langle \{5, 8, 9, 18, 25, 45\}, \{4, 12, 13, 23, 28\}, \{1, 10, 15, 21, 51\}, \{3, 6, 14, 27, 74\}, \{26, 29, 75, 78, 84\} \rangle$. Table 14 summarizes the top 5 courses recommended to this student by the proposed methodology together with the recommendation index scores for each course. In this occasion, the student t_3 finally took two of the five best courses recommended to him/her, that is, courses with IDs 64 and 77. To show the adequacy of the proposal and the validity of

the recommendations proposed, let us analyze the GPA obtained by t_3 on those courses. t_3 obtained a GPA of 96 in the course with ID 64, being among the 11.11% best students for that course. As for the course with ID 77, he/she obtained a GPA of 95, which corresponds to the top 13.58% of the best students for that course.

Finally, let us consider a student t_4 belonging to the Faculty of Information Technology, which has passed the courses identified by the following path $t_4 = \langle \{5, 12, 23, 28, 45\}, \{4, 8, 9, 13, 18, 25\}, \{1, 10, 15, 21, 51\}, \{3, 6, 14, 27, 74\} \rangle$. Analyzing t_4 , he/she is recommended to take 5 courses as the top according to the recommendation index score (see Table 15). Analyzing what the student finally did, it is obtained that he/she finally took two of such five courses that were recommended. In this way, for the course with ID 26, t_4 obtained a GPA of 85 (top 37.32% in the total GPA ranking for that course). On the other hand, the GPA obtained by t_4 on the course with ID 77 was again 85, being this time in the top 27.16% of the ranking for the given course. However, if we take into account the rest of the courses taken by this student, they are present with a very low recommendation index score. For example, in the case of the course with ID 78, the recommendation index score is 0.365 (the student t_4 finally obtained a GPA of 68 in that course, which is within the 80.25% of the best GPAs of that course), whereas for the course with ID 84, the recommendation index value was 0. The student finally took this course and his/her GPA was 87 for that course and this GPA is within the 64.36% of best students. As it is demonstrated, the recommendation index score is low because taking such courses implies not to be in the group of excellent students. Last but not least, it is important to clarify that the number of credits of each course is taken into account to obtain the recommendation index score, what explains why lower score values may imply the student to be in a higher position of the ranking.

Conclusion

In this paper, we have proposed an evolutionary algorithm for mining top- k emerging sequential patterns, which is called (ES)²P. It is able to discover a reduced set of discriminative patterns whose frequency increases significantly from one group or dataset to another. Its main advantage is that it does not need any threshold as existing algorithms do and, therefore, any background in the application field. Additionally, a methodology for rating courses is also proposed, which does not require anything except for the courses (ordered by semesters) that already pass by those students that are advised. This methodology considers the proposed (ES)²P algorithm to obtain the most promising sequences of courses. Last but not least, we have also proposed a

methodology for ordering courses that should be taken by students so they can be successful in their degree. This methodology, also based on the proposed (ES)²P algorithm, provides full study plans that should be followed by students to reduce the dropout and failure. The experimental analysis has demonstrated that the proposed (ES)²P algorithm behaves really well in terms of runtime, and it is able to extract useful information in huge datasets where other algorithms fail. The efficiency of the proposal has been tested on two cases of study, providing excellent recommendations on a real scenario.

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature. This work was supported by the Spanish Ministry of Science and Innovation, project PID2020-115832GB-I00, and the University of Cordoba, project UCO-FEDER 18 REF.1263116 MOD.A. Both projects were also supported by the European Fund of Regional Development.

Declarations

Ethical Approval All procedures performed in studies involving human participants were in accordance with the ethical standards.

Conflict of Interest The authors declare that they have no conflict of interest.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. Guruge DB, Kadel R, Halder SJ. The state of the art in methodologies of course recommender systems—a review of recent research. *Data*. 2021;6(2):18.
2. Bhumichitr K, Channarukul S, Saejiem N, Jiamthapthaksin R, Nongpong K. Recommender Systems for university elective course recommendation. In: 2017 14th International Joint Conference on Computer Science and Software Engineering (JCSSE). IEEE; 2017. p. 1-5.
3. Shakhisi-Niaei M, Abuei-Mehrizi H. An optimization-based decision support system for students' personalized long-term course planning. *Comput Appl Eng Educ*. 2020;28(5):1247–64.
4. Wu K, Havens WS. Modelling an Academic Curriculum Plan as a Mixed-Initiative Constraint Satisfaction Problem. In: Kégl B, Lapalme G, editors. *Advances in Artificial Intelligence*, 18th Conference of the Canadian Society for Computational Studies

- of Intelligence, Canadian AI 2005, Victoria, Canada, May 9-11, 2005, Proceedings. vol. 3501 of Lecture Notes in Computer Science. Springer; 2005. p. 79-90. https://doi.org/10.1007/11424918_10.
5. Agrawal R, Srikant R. Mining sequential patterns. In: Proceedings of the eleventh international conference on data engineering. IEEE; 1995. p. 3-14.
 6. Srikant R, Agrawal R. Mining sequential patterns: Generalizations and performance improvements. In: International conference on extending database technology. Springer; 1996. p. 1-17.
 7. Dong G, Li J. Efficient mining of emerging patterns: Discovering trends and differences. In: Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining; 1999. p. 43-52.
 8. Luna JM, Fournier-Viger P, Ventura S. Frequent itemset mining: A 25 years review. *Wiley Interdiscip Rev Data Min Knowl Discov*. 2019;9(6). <https://doi.org/10.1002/widm.1329>.
 9. Wu C, Shie B, Tseng VS, Yu PS. Mining top-K high utility itemsets. In: Yang Q, Agarwal D, Pei J, editors. The 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '12, Beijing, China, August 12-16, 2012. ACM; 2012. p. 78-86. <https://doi.org/10.1145/2339530.2339546>.
 10. Padillo F, Luna JM, Ventura S. A Grammar-Guided Genetic Programming Algorithm for Associative Classification in Big Data. *Cogn Comput*. 2019;11(3):331–46. <https://doi.org/10.1007/s12559-018-9617-2>.
 11. Ventura S, Luna JM. Pattern Mining with Evolutionary Algorithms. Springer; 2016. <https://doi.org/10.1007/978-3-319-33858-3>.
 12. Fournier-Viger P, Lin JCW, Kiran RU, Koh YS, Thomas R. A survey of sequential pattern mining. *Data Science and Pattern Recognition*. 2017;1(1):54–77.
 13. Zaki MJ. SPADE: An efficient algorithm for mining frequent sequences. *Mach Learn*. 2001;42(1):31–60.
 14. Ayres J, Flannick J, Gehrke J, Yiu T. Sequential pattern mining using a bitmap representation. In: Proceedings of the eighth ACM SIGKDD International Conference On Knowledge Discovery and Data Mining; 2002. p. 429-35.
 15. Pei J, Han J, Mortazavi-Asl B, Wang J, Pinto H, Chen Q, et al. Mining sequential patterns by pattern-growth: The prefixspan approach. *IEEE Trans Knowl Data Eng*. 2004;16(11):1424–40.
 16. Fournier-Viger P, Gomariz A, Campos M, Thomas R. Fast vertical mining of sequential patterns using co-occurrence information. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining. Springer; 2014. p. 40-52.
 17. Fournier-Viger P, Gomariz A, Gueniche T, Mwamikazi E, Thomas R. TKS: Efficient Mining of Top-K Sequential Patterns. In: Motoda H, Wu Z, Cao L, Zai'ane OR, Yao M, Wang W, editors. Advanced Data Mining and Applications, 9th International Conference, ADMA 2013, Hangzhou, China, December 14-16, 2013, Proceedings, Part I. vol. 8346 of Lecture Notes in Computer Science. Springer; 2013. p. 109-20. https://doi.org/10.1007/978-3-642-53914-5_10.
 18. Tzvetkov P, Yan X, Han J. TSP: Mining top-k closed sequential patterns. *Knowl Inf Syst*. 2005;7(4):438–57. <https://doi.org/10.1007/s10115-004-0175-4>.
 19. Purushothama Raju V, Saradhi Varma GP. Mining closed sequential patterns using genetic algorithm. In: 2014 IEEE International Conference on Advanced Communications, Control and Computing Technologies; 2014. p. 634-7. <https://doi.org/10.1109/ICACCCT.2014.7019165>.
 20. Zheng Z, Zhao Y, Zuo Z, Cao L. An Efficient GA-Based Algorithm for Mining Negative Sequential Patterns. In: Zaki MJ, Yu JX, Ravindran B, Pudi V, editors. Advances in Knowledge Discovery and Data Mining, 14th Pacific-Asia Conference, PAKDD 2010, Hyderabad, India, June 21-24, 2010. Proceedings. Part I. vol. 6118 of Lecture Notes in Computer Science. Springer; 2010. p. 262-73.
 21. Ykhlef M, ElGibreen H. Mining sequential patterns using hybrid evolutionary algorithm. *Int J Comput Inform Eng*. 2009;3(12):2939–2946.
 22. Li J, Manoukian T, Dong G, Ramamohanarao K. Incremental maintenance on the border of the space of emerging patterns. *Data Min Knowl Disc*. 2004;9(1):89–116.
 23. Zhang X, Dong G, Kotagiri R. Exploring constraints to efficiently mine emerging patterns from large high-dimensional datasets. In: Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining; 2000. p. 310-4.
 24. Fan H, Ramamohanarao K. Efficiently mining interesting emerging patterns. In: International Conference on Web-Age Information Management. Springer; 2003. p. 189-201.
 25. Kobyliński Ł, Walczak K. Jumping emerging patterns with occurrence count in image classification. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining. Springer; 2008. p. 904-9.
 26. Sakr NA, Abu-Elkheir M, Atwan A, Soliman H. Data driven recognition of interleaved and concurrent human activities with nonlinear characteristics. *J Intell Fuzzy Syst*. 2019;37(4):5573–88.
 27. Poezevara G, Lozano S, Cuissart B, Bureau R, Bureau P, Croixmarie V, et al. A computational selection of metabolite biomarkers using emerging pattern mining: a case study in human hepatocellular carcinoma. *J Proteome Res*. 2017;16(6):2240–9.
 28. Nofong VM, Liu J, Li J. A Study on the Applications of Emerging Sequential Patterns. In: Wang H, Sharaf MA, editors. Databases Theory and Applications - 25th Australasian Database Conference, ADC 2014, Brisbane, QLD, Australia, July 14-16, 2014. Proceedings. vol. 8506 of Lecture Notes in Computer Science. Springer; 2014. p. 62-73. https://doi.org/10.1007/978-3-319-08608-8_6.
 29. Chen Z, Liu X, Shang L. Improved course recommendation algorithm based on collaborative filtering. In: 2020 International Conference on Big Data and Informatization Education (ICBDIE). IEEE; 2020. p. 466-9.
 30. Huang L, Wang CD, Chao HY, Lai JH, Philip SY. A score prediction approach for optional course recommendation via cross-user-domain collaborative filtering. *IEEE Access*. 2019;7:19550–63.
 31. Lessa LF, Brandão WC. Filtering graduate courses based on LinkedIn profiles. In: Proceedings of the 24th Brazilian Symposium on Multimedia and the Web; 2018. p. 141-7.
 32. Mostafa L, Oatley G, Khalifa N, Rabie W. A case based reasoning system for academic advising in egyptian educational institutions. In: 2nd International Conference on Research in Science, Engineering and Technology (ICRSET'2014) March; 2014. p. 21-2.
 33. Huang CY, Chen RC, Chen LS. Course-recommendation system based on ontology. In: 2013 International Conference on Machine Learning and Cybernetics. vol. 3. IEEE; 2013. p. 1168-73.
 34. Ng YK, Linn J. CrsRecs: a personalized course recommendation system for college students. In: 2017 8th International Conference on Information, Intelligence, Systems & Applications (IISA). IEEE; 2017. p. 1-6.
 35. Esteban A, Zafra A, Romero C. Helping university students to choose elective courses by using a hybrid multi-criteria recommendation system with genetic optimization. *Knowl-Based Syst*. 2020;194: 105385.
 36. Noaman AY, Luna JM, Ragab AHM, Ventura S. Recommending degree studies according to students' attitudes in high school by means of subgroup discovery. *Int J Comput Intell Syst*. 2016;9(6):1101–17. <https://doi.org/10.1080/18756891.2016.1256573>.
 37. Volk NA, Rojas G, Vitali MV. UniNet: Next Term Course Recommendation using Deep Learning. In: 2020 International Conference on Advanced Computer Science and Information Systems (ICACSIS). IEEE; 2020. p. 377-80.
 38. Britto J, Prabhu S, Gawali A, Jadhav Y. A Machine Learning Based Approach for Recommending Courses at Graduate Level. In: 2019 International Conference on Smart Systems and Inventive Technology (ICSSIT). IEEE; 2019. p. 117-21.
 39. Sankhe V, Shah J, Paranjape T, Shankarmani R. Skill Based Course Recommendation System. In: 2020 IEEE International

- Conference on Computing, Power and Communication Technologies (GUCON). IEEE; 2020. p. 573-6.
40. Luna JM, Fardoun HM, Padillo F, Romero C, Ventura S. Subgroup discovery in MOOCs: a big data application for describing different types of learners. *Interact Learn Environ*. 2022;30(1):127–45.
 41. Wang R, Zaiane OR. Sequence-Based Approaches to Course Recommender Systems. In: Hartmann S, Ma H, Hameurlain A, Pernul G, Wagner RR, editors. *Database and Expert Systems Applications - 29th International Conference, DEXA 2018, Regensburg, Germany, September 3-6, 2018, Proceedings, Part I*. vol. 11029 of *Lecture Notes in Computer Science*. Springer; 2018. p. 35-50. https://doi.org/10.1007/978-3-319-98809-2_3.
 42. Nguyen HQ, Pham TT, Vo V, Vo B, Quan TT. The predictive modeling for learning student results based on sequential rules. *Int J Innov Comput Inf Control*. 2018;14(6):2129–40.
 43. Fournier-Viger P, Gomariz A, Gueniche T, Soltani A, Wu C, Tseng VS. SPMF: a Java open-source pattern mining library. *J Mach Learn Res*. 2014;15(1):3389-93. Available from: <http://dl.acm.org/citation.cfm?id=2750353>.
 44. Friedman M. A Comparison of Alternative Tests of Significance for the Problem of m Rankings. *Annals Math Stat*. 1940 03;11(1):86-92. <https://doi.org/10.1214/aoms/1177731944>.
 45. Shaffer JP. Modified sequentially rejective multiple test procedures. *J Am Stat Assoc*. 1986;81(395):826–31.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.