

C3Ro: An efficient mining algorithm of extended d-close d contiguous robust sequential patterns in noisy data

Y Abboud, Armelle Brun, Anne Boyer

► To cite this version:

Y Abboud, Armelle Brun, Anne Boyer. C3Ro: An efficient mining algorithm of extended d-close d contiguous robust sequential patterns in noisy data. Expert Systems with Applications, Elsevier, 2019, 131, pp.172 - 189. 10.1016/j.eswa.2019.04.058 . hal-02977461

HAL Id: hal-02977461

<https://hal.inria.fr/hal-02977461>

Submitted on 25 Oct 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

C3Ro: an Efficient Mining Algorithm of Extended-Closed Contiguous Robust Sequential Patterns in Noisy Data

Y. Abboud, A. Brun, A. Boyer

Université de Lorraine, LORIA UMR 7503, France

Abstract

Sequential pattern mining has been the focus of many works, but still faces a tough challenge in the mining of large databases for both efficiency and apprehensibility of its resulting set. To overcome these issues, the most promising direction taken by the literature relies on the use of constraints, including the well-known closedness constraint. However, such a mining is not resistant to noise in data, a characteristic of most real-world data. The main research question raised in this paper is thus: how to efficiently mine an apprehensible set of sequential patterns from noisy data?

In order to address this research question, we introduce 1) two original constraints designed for the mining of noisy data: the robustness and the extended-closedness constraints, 2) a generic pattern mining algorithm, C3Ro, designed to mine a wide range of sequential patterns, going from closed or maximal contiguous sequential patterns to closed or maximal regular sequential patterns. C3Ro is dedicated to practitioners and is able to manage their multiple constraints. C3Ro also is the first sequential pattern mining algorithm to be as generic and parameterizable.

Extensive experiments have been conducted and reveal the high efficiency of C3Ro, especially in large datasets, over well-known algorithms from the literature. Additional experiments have been conducted on a real-world job offers noisy dataset, with the goal to mine activities. This experiment offers a more thorough insight into C3Ro algorithm: job market experts confirm that the constraints we introduced actually have a significant positive impact on the apprehensibility of the set of mined activities.

Keywords: Data mining, Sequential pattern mining, Closed contiguous pattern, Noisy data, Constraints, Efficiency

1. Introduction

Pattern mining [5] is one of the most studied topics in the data mining literature. A sequential pattern is a pattern whose order of items is considered and applications rely on their mining: pattern discovery in protein sequences [59, 36], analysis of customer behavior in web logs [10, 11], sequence-based classification [4], etc. Consequently, sequential pattern mining [6] has become a significant part of pattern mining studies. Many sequential pattern mining algorithms have been proposed, such as regular sequential pattern mining [46, 26, 27, 56, 7, 13], string mining [12, 41], closed sequential pattern mining [42, 43, 57, 51, 47, 23, 13], constraint-based sequential pattern mining [22, 44, 8, 32], etc., which all

have substantially improved the domain.

Although large databases are becoming quite common, mining sequential patterns in such data is still a computationally expensive task, when not an intractable one. In addition, the complete set of sequential patterns is often difficult to apprehend for the final user, due to its large size and to the presence of useless or redundant patterns. This is especially limiting when the final user is a practitioner who expects a set of patterns that fits his/her requirements. Therefore, the domain still faces tough challenges related to both

- the efficiency of the mining algorithms
- the apprehensibility of the resulting set of patterns.

We propose to define the efficiency of an algorithm as the ability to run in an optimal execution time and memory usage; and the apprehensibility of a set of patterns as a trade-off between, on one side its size and redundancy in its patterns, and on the other side the information contained in the set.

Email addresses: yacine.abboud@gmail.com (Y. Abboud), armelle.brun@loria.fr (A. Brun), anne.boyer@loria.fr (A. Boyer)

34 To overcome both issues, the most promising direction 86
35 adopted by the literature relies on the use of constraints 87
36 on the patterns that results in a reduced set of pat- 88
37 terns [44, 32, 24]. We identify two types of constraints: 89
38 one applies to a set of sequential patterns, and the 90
39 other one applies to the intrinsic characteristics of each 91
40 sequential pattern. 92

41
42 The first type of constraints includes the closed- 94
43 ness [42, 43, 57, 51, 47] that preserves all the 95
44 information, the generator-pattern [19, 53, 14, 33] and 96
45 the maximal-pattern [46, 39, 21, 31, 16] constraints 97
46 that reduce the size of the set of patterns at the cost 98
47 of a part of the information. Closedness has become 99
48 popular as the resulting set of patterns is less redundant, 100
49 which increases its apprehensibility. However, despite 101
50 this improvement [57, 48, 51, 23, 13], the set remains 102
51 unmanageable in large databases [58] and in noisy 103
52 data [52, 38, 54, 37] as most of the benefit of closedness 104
53 is lost. Noise, which is a characteristic inherent to many 105
54 real-world applications [20], is the random appearance 106
55 of disturbances in data, such as loss, substitution, or 107
56 addition of items. 108

57 With the double objective to mine a more apprehen- 109
58 sible set and to increase the efficiency of the mining 110
59 algorithm, we introduce a new constraint: the *extended-* 111
60 *closedness constraint*. This new constraint relaxes the 112
61 constraint of identical values required by the closedness 113
62 constraint. Thus, the set of extended-closed sequential 114
63 patterns can be viewed as a trade-off between the set 115
64 of closed sequential patterns, which contains all the 116
65 information, and the set of maximal sequential patterns, 117
66 which is an even more reduced set, but that contains a 118
67 subset of the information. 119

68
69 The second type of constraints includes a large num- 120
70 ber of constraints such as monotonic or anti-monotonic 121
71 constraints [44], regular expression constraints [22], 122
72 gap constraints [34], contiguity constraint [8, 58, 2] (a 123
73 gap constraint with a value of 0), etc. The contiguity 124
74 constraint is the most popular one. Many real-life tasks 125
75 greatly benefit from this constraint: text mining [18], 126
76 Web log mining [10, 9], DNA and amino acid se- 127
77 quences mining [30, 31], etc. The contiguity constraint 128
78 leads to the extraction of much fewer patterns, with 129
79 a shorter average length, and with almost no loss of 130
80 information [58], resulting in a more apprehensible 131
81 set, while improving the efficiency of the algorithm. 132
82 However, the added value provided by the contiguity 133
83 constraint faces limitations in case of noisy data. 134
84 Indeed, the support of the contiguous patterns becomes 135
85 unreliable [38], making some them being frequent or 136

at the opposite being not frequent. The set of patterns 86
is thus unreliable too. Wildcards are a way to manage 87
noisy data [37], even with a contiguity goal [2]. A wild- 88
card [34, 49, 50, 37] is a joker of one item, regarding 89
the contiguity constraint. However, using wildcards in 90
contiguous sequential pattern mining may result in the 91
mining of semi-contiguous sequential patterns, which 92
decreases the apprehensibility of the resulting set. 93

94 To make the contiguity constraint noise-resistant and 95
thus the set of patterns more apprehensible, we intro- 96
duce a second constraint: *the robustness constraint*. 97
A robust sequential pattern is a frequent contiguous 98
sequential pattern that occurs a limited number of times 99
with wildcards. 100

101 Each constraint from the literature allows to solve a 102
specific issue in sequential pattern mining. However, 103
practitioners' final needs are often so specific that they 104
correspond to a combination of such constraints. For 105
example, contiguous sequential patterns, in noisy data, 106
which contain a particular set of items. To date, in order 107
to obtain the set that fits the desired combination of con- 108
straints, the use of several algorithms is often required. 109
However, there is no guarantee of compatibility, consis- 110
tency or scalability of the entire process [8]. Combining 111
several algorithms and solving their possible compatibil- 112
ity issues can be inaccessible to many practitioners 113
which limits their usability. 114

115 To alleviate this issue, we introduce C3Ro, a simple 116
and generic algorithm. The key idea of C3Ro is to in- 117
tegrate several constraints and the associated param- 118
eters, to mine a wide range of sequential patterns, go- 119
ing from closed or maximal contiguous sequential pat- 120
terns to closed or maximal sequential patterns, includ- 121
ing semi-contiguous sequential patterns. In addition, it 122
is noise-resistant thanks to the use of wildcards and of 123
the extended-closedness constraint. It thus increases the 124
efficiency and the apprehensibility of the set of mined 125
patterns. 126

Hence, the main contributions of this paper are:

- we introduce two new constraints: robustness and extended-closedness, designed to improve both the efficiency of the mining process and the apprehensibility of a set of closed contiguous sequential patterns in noisy data. These constraints have many practical applications in different fields, especially those related to humans who are often a source of noisy data. Apprehensibility is specifically thought for practitioners, with the goal to mine the smallest set that contains the highest amount of information;

- we introduce C3Ro, a generic and highly parameterizable algorithm that can be used in various contexts and adaptable to many user needs. It is especially designed for practitioners to help them mine patterns using a single tool, while considering all their requirements. To reach this generality, C3Ro manages a set of monotonic or anti-monotonic constraints, a number of wildcards, robustness and extended ratios.

This paper is organized as follows. In section 2 we discuss the related work. In Section 3 we introduce concepts and notations. Section 4 defines the two new constraints. The C3Ro algorithm is introduced in Section 5. Section 6 is dedicated to the experiments conducted, including a real-world dataset in the domain of job market and Section 7 concludes this work and discusses some perspectives.

2. Related Work

The sequential pattern mining problem was first introduced by Agrawal and Srikant in [6], with the Apriori algorithm. Apriori is based on the monotonic constraint of the frequency: “all nonempty subsets of a frequent itemset must also be frequent” [5]. Since then, many sequential pattern mining algorithms have been proposed, still based on the monotonic constraint, with the goal of improving the mining efficiency in terms of execution time and memory usage, such as GSP [46], PrefixSpan [27], SPADE [56], SPAM [7], CM-SPADE [13], CM-SPAM [13], etc. The PrefixSpan algorithm uses a pattern-growth philosophy through projected databases to mine frequent sequential patterns. PrefixSpan recursively extends a prefix sequential pattern by adding a frequent item from the projected database of this prefix. The SPADE algorithm, released the same year, is based on a vertical IDLists representation and uses a lattice-theoretic approach to decompose the original search space into smaller spaces. One year after, the SPAM algorithm used a different representation, it is a vertical bitmap representation dedicated to the mining of long sequential patterns. While being faster than SPADE and PrefixSpan for the mining of long patterns, SPAM uses a large amount of memory. The newer CM-SPADE and CM-SPAM algorithms introduced a co-occurrence matrix to substantially reduce the number of join operations between the IDLists of the candidate sequential patterns and thus improve the efficiency of the mining. Although all these algorithms contributed to the improvement of the mining process, they still face tough

challenges in the mining efficiency and in the apprehensibility of the results. Several directions are used in the literature to overcome those issues. The two main directions are the use of a compact representation of sequential patterns [25] and the reduction of the number of sequential patterns mined [46]. In this paper, we focus on the latter and the most promising way reach this goal is through the use of constraints [46, 22, 44, 8]. Using constraints during the mining process allows to extract less patterns, and improves the efficiency. However, the usage of constraints improves the apprehensibility if the discarded patterns only contain information irrelevant to the practitioners. Therefore, we are only interested in constraints that preserve relevant information, i.e., that are able to adapt to the application context of the practitioners. To facilitate the analysis of the constraints from the literature, we propose to classify them in two categories that we name: (1) Global constraints, i.e., constraints that relate to the characteristics of a set of sequential patterns. (2) Local constraints, i.e., constraints that relate to the intrinsic characteristics of each sequential pattern.

2.1. Global constraints

Global constraints [43, 39, 19] are, for example, the well-known maximal, generator and closedness constraints. With Global constraints, the reduction of set of frequent patterns may be reached at the cost of a part of the information. However, it is impossible to control both the amount and the nature of the information lost, this loss can thus be prejudicial for many applications. The maximal constraint [21, 16] mines only the frequent sequential patterns that are not included in any other frequent sequential pattern. Therefore, the set of maximal sequential patterns is significantly smaller than the complete set of sequential patterns. However, the information (especially the support value) of the sequential patterns included in maximal patterns is lost, which often represents a non-negligible part of the information. The generator constraint [53, 14, 33] mines sequential patterns that do not contain any other sequential pattern with the same support. The set of generator sequential patterns is also significantly smaller than the set of sequential patterns. The information loss is about all the sequential patterns that contain the generator sequential patterns, especially about those that are not frequent. The closedness constraint [57] mines sequential patterns that have no super-pattern with the same support. The set of closed sequential patterns is often larger than both maximal and generator sequential patterns sets, and remains smaller than the set of sequential patterns. Moreover, the closedness constraint does

not cause any loss of information. Closedness is by far the most popular Global constraint and numerous algorithms [43, 51, 34, 23, 13, 33, 45] focus on the mining of closed patterns.

The Clospan [51] algorithm is one of the first algorithms dedicated to the mining of closed sequential patterns. Clospan mines and stores a set of candidate patterns, based on PrefixSpan, then a post processing phase filters out the non closed ones. Storing the set of candidates is expensive in terms of memory. To cope with this issue, the BIDE algorithm [47] proposes to not memorize any candidate pattern. BIDE is also based on PrefixSpan, but uses a BI-Directional closedness checking scheme to not generate any set of candidates, hence BIDE improves greatly both execution time and memory usage. More recent algorithms such as Clasp [23] and CloFast [17] adopt a vertical IDLists representation to outperform Clospan and BIDE in terms of execution time, but at the expense of a higher memory usage due to the candidate generation phase. CM-Clasp [13] and FCloSM [33] algorithms both use a co-occurrence matrix to reduce the number of candidates and improve both execution time and memory usage.

A remaining major issue in the mining of closed sequential patterns lies in the low apprehensibility of the set of mined patterns in large databases. This is especially true for vertical IDLists-based algorithms, which are not efficient due to their high memory usage. The pattern-growth philosophy appears to be more suited. However, the issue gets worse when mining noisy data [2].

In a nutshell, on one hand, the maximal and the generator constraints allow to significantly reduce the set of mined patterns. On the other hand, these constraints also cause a significant and unpredictable loss of information, which is often unacceptable for practitioners. The closedness constraint seems more relevant as no information is lost. However, its issue with large and noisy databases needs to be addressed.

2.2. Local constraints

Local constraints [46, 40, 55] regroup numerous constraints such as monotonic or anti-monotonic constraints: presence of an item or a pattern, maximum or minimum length, duration, etc. [44], regular expression constraints [22], gap constraints [34], including contiguity constraint, etc. Local constraints are practitioner-dependent: they are oriented towards the application context, according to the practitioner interests. One of the very first sequential pattern mining algorithm that has integrated Local constraints is GSP [46]. GSP allows to use duration and gap constraints on all the sequential patterns that it extracts.

These two constraints are widely used in the literature [29, 28, 15]. The SPIRIT algorithm is the first algorithm that integrates regular expression constraints [22], it converts the constraints into an automaton that prunes the patterns during the mining. The monotonic and anti-monotonic constraints allow the pattern-growth philosophy to identify in an early stage if the patterns formed, when extending a prefix, will satisfy or not a set of constraints [44]. For this reason, several algorithms like PG [44], CloSPEC [8], global-p.f [32], CCPM [2], based on the pattern-growth philosophy, are able to mine patterns satisfying an aggregate of monotonic or anti-monotonic constraints.

A commonly used gap constraint is the contiguity constraint ($gap = 0$) [18, 10], especially used for natural language processing or document classification. Recently, [58] has shown that the information contained in the set of contiguous sequential patterns is almost equivalent to the information contained in the set of sequential patterns. The contiguity constraint is therefore a Local constraint that can be used to reduce the number of mined patterns. Consequently, the use of the contiguity constraint improves both the efficiency of the mining and the apprehensibility of the results. However, the mining of contiguous patterns has the drawback of being unable to mine noisy data, without a significant loss of information.

2.3. Combination of constraints

Both Global and Local constraints are used to reduce the number of sequential patterns mined, while retaining either the complete information, or at least the most important information for the practitioner. In addition to improving the apprehensibility of the results, these constraints also improve the efficiency of the algorithm since the decrease in the number of sequential patterns mined has a very favorable impact on execution time and memory usage [47, 44]. Therefore, both the ability of algorithms to integrate some constraints and the possibility for the practitioner to specify them, are critical.

Several algorithms allow to use multiple constraints during the mining. For instance, the PG algorithm [44] uses rules to integrate a set of monotonic and anti-monotonic constraints during the mining of sequential patterns. Many algorithms like CloSPEC [8] and CCPM [2] use such rules in addition to the closedness constraint. Unfortunately, although Local constraints are often effective to improve the efficiency of the mining process and the apprehensibility of its result, they also depend on the application context of the practitioner and thus are not always specifiable.

Recently the CCSpan [58] algorithm introduced closed

contiguous sequential pattern (*CICoSP*) mining. Frequent *CICoSP* are far fewer and shorter than frequent closed sequential patterns, while containing the same information [58] than closed sequential patterns. CC-Span outperforms several closed sequential pattern mining algorithms like Clospan and BIDE, but scales rather poorly on datasets with long sequences [2]. The CCPM algorithm [2] also mines *CICoSP* based on the same pattern-growth philosophy than BIDE and PG. CCPM outperforms and scales better than CCSpan, while allowing to use in addition monotonic or anti-monotonic constraints. However, although *CICoSP* mining is an effective combination of constraints to reduce the size of the resulting set of patterns, it has the noise resistance limit of the contiguity constraint.

2.4. Pattern mining in noisy data

As many real-world applications generate noisy or uncertain data, more and more works focus on the mining of sequential patterns in such data [3, 38, 54, 35, 37]. In these works, data is qualified as either noisy data [38, 54] or uncertain data [37, 34]. The common approach adopted by these works notices that it is impossible to count the frequency of patterns deterministically, so do adopt a probabilistic approach. In addition, they assume statistical independence between the different items in terms of their uncertain probability behavior. Thus they evaluate the probabilistic support of each itemset or pattern in each sequence to determine if the appearance of an itemset in a pattern is likely to be an error or not.

The approach proposed in [38] adopts a constraint-based approach, the model proposed places constraints on the fraction of errors permitted in each item column and the fraction of errors permitted in a supporting transaction. Taken together, these constraints make the patterns that contain systematic errors being discarded.

The major drawback of the probability-based approach is its high computational complexity, it is thus not efficient and does not meet our requirements.

2.5. Summary

In summary, the literature highlights that a wide range of sequential pattern mining algorithms mine a (too) high number of patterns. This set is not apprehensible. Two main directions are adopted to solve this issue. The first direction aims at decreasing the set of patterns while maximizing the amount of information, without depending on the application context of the practitioner. Like the well-known closed sequential pattern mining or contiguous sequential pattern mining. However, none of them is able to

manage large and noisy databases, which is common in many real-world applications [38]. The second direction aims at decreasing the set of patterns while extracting only the information relevant to the practitioner, by using Local constraints. A local constraint is practitioner-dependent, and significantly improves the apprehensibility and the efficiency of the mining. Although many algorithms (BIDE, PG, CCPM) based on the pattern-growth philosophy allow the combination of several constraints, no noise-resistant algorithm aggregates as many constraints as monotonic, anti-monotonic, closedness and contiguity constraints. In other words, most algorithms lack genericity. As a consequence, there is a critical need for a generic algorithm able to manage large and noisy databases.

The issues raised by the state-of-the art make us consider the following questions:

- 1) How to efficiently mine an apprehensible set of sequential patterns, including in the frame of noisy data?
- 2) How to design a sequential pattern mining algorithm that aggregates several constraints such as monotonic, anti-monotonic, closedness and contiguity constraints and thus can be used with several combinations of constraints, whatever are the needs of practitioners are?

3. Preliminaries

In this section, we introduce some definitions and notations that will be further used in this paper.

Let $I = \{i_1, i_2, \dots, i_n\}$ be a set of distinct items. An itemset $I_m \subseteq I$ with $|I_m|$ being the length of I_m , is an unordered set of distinct items. Without loss of generality, we assume that items in itemsets are sorted according to a total order, such as the lexicographic order to facilitate the reading and for further processing. A sequential pattern P is an ordered list of itemsets, denoted by $P = \langle E_1, E_2, \dots, E_j \rangle$ where $E_k \subseteq I$ with $1 \leq k \leq n$. In this paper, an itemset is always written with an uppercase letter, while an item is always written with a lowercase letter.

Let e be an item of I , $P \diamond e$ means P concatenates with e . Concatenation can be an I-extension (I for item), $P \diamond_i e = \langle E_1, E_2, \dots, E_m \cup \{e\} \rangle$ or an S-extension (S for sequence), $P \diamond_s e = \langle E_1, E_2, \dots, E_m, \{e\} \rangle$. We expand the definition of *item extension* to *sequential pattern extension*. Given $P = \langle E_1, E_2, \dots, E_n \rangle$ and $P' = \langle E'_1, E'_2, \dots, E'_m \rangle$ two sequential patterns, $P \diamond P'$ means P concatenates with P' . When it is a I-extension, $P \diamond_i P' = \langle E_1, E_2, \dots, E_n \cup \{E'_1, E'_2, \dots, E'_m\} \rangle$ and when it is a S-extension, $P \diamond_s P' = \langle E_1, E_2, \dots, E_n, E'_1, E'_2, \dots, E'_m \rangle$.

436 An input sequence database SDB is a set of tuples 484
 437 (sid, S), where sid is a sequence id, and S an input se- 485
 438 quence. A sequence can also be considered as a pattern, 486
 439 thus it shares the same properties. The number of 487
 440 tuples in SDB is called the size of SDB and is denoted by 488
 441 $|SDB|$. A wildcard is a symbol that matches any itemset 489
 442 of the sequence database. Aside from the above nota- 490
 443 tions, we further present some definitions.

444 **Definition 3.1** (Contiguous sub-pattern). Given $P =$ 491
 445 $\langle E_1, E_2, \dots, E_n \rangle$ and $P' = \langle E'_1, E'_2, \dots, E'_m \rangle$ two sequential 492
 446 patterns, P is a contiguous sub-pattern of P' , denoted as 493
 447 $P \sqsubseteq P'$, if and only if $n \leq m$ and there exist consecutive 494
 448 integers j_1, j_2, \dots, j_n such that: (1) $1 \leq j_1 < j_2 < \dots <$
 449 $j_n \leq m$; and (2) $E_1 \subseteq E'_{j_1}, E_2 \subseteq E'_{j_2}, \dots, E_n \subseteq E'_{j_n}$. P' is 495
 450 called a contiguous super-pattern of P , and P' is said to 496
 451 contain P .

452 **Definition 3.2** (Absolute, Relative and Universal sup- 497
 453 port). The absolute support of a sequential pattern P 498
 454 in a sequence database SDB is the number of tuples in 499
 455 SDB that contain P , denoted by $sup_A^{SDB}(P)$. The rela- 500
 456 tive support of P in SDB is the proportion of tuples in 501
 457 SDB that contain P , denoted by $sup_R^{SDB}(P)$. The univer- 502
 458 sal support of P in SDB is the number of occurrences 503
 459 of P in SDB , denoted by $sup_U^{SDB}(P)$.

460 Without loss of generality, we use the absolute sup- 505
 461 port for describing the C3Ro algorithm and use the rela- 506
 462 tive support to present the experimental results in the 507
 463 remaining of the paper.

464 **Definition 3.3** (Frequent closed contiguous sequential 509
 465 pattern). Given min_sup a support threshold, a con- 510
 466 tiguous sequential pattern P is frequent in SDB if 511
 467 $sup_A^{SDB}(P) \geq min_sup$. P is a frequent closed con- 512
 468 tiguous sequential pattern if there exists no sequen- 513
 469 tial pattern P' such that: $P \sqsubset P'$, and $sup_A^{SDB}(P) =$ 514
 470 $sup_A^{SDB}(P')$.

471 We now present some definitions related to the con- 517
 472 tiguous projected items of a prefix sequential pattern, in 518
 473 the context of contiguous sequential pattern mining. 519
 474 Given an input database SDB , an input sequence 520
 475 $S = \langle E'_1, E'_2, \dots, E'_m \rangle$ and a sequential pattern $P =$ 521
 476 $\langle E_1, E_2, \dots, E_n \rangle$ with $P \sqsubseteq S$. Therefore, there exist con- 522
 477 secutive integers j_1, j_2, \dots, j_n such that: (1) $1 \leq j_1 <$ 523
 478 $j_2 < \dots < j_n \leq m$; and (2) $E_1 \subseteq E'_{j_1}, E_2 \subseteq E'_{j_2}, \dots, E_n \subseteq$ 524
 479 E'_{j_n} .

480 **Definition 3.4** (Contiguous projected items of a sequen- 527
 481 tial pattern). The contiguous projected items of P in S 528
 482 are the union of two sets: (1) the set of i-extensions e_i of 529
 483 each occurrence of P in S such that $E_n \diamond_i e_i \subseteq E'_{j_n}$, (2) if 530

$m > n$, the set of s-extensions e_s of each occurrence of 484
 P in S such that $e_s \in E'_{j_{n+1}}$.

486 **Definition 3.5** (Contiguous projected sequence). The 487
 488 set of contiguous projected items of P in S is called the 489
 489 contiguous projected sequence of P in S .

490 **Definition 3.6** (Prefix contiguous sequential pattern). P 491
 491 is called the prefix contiguous sequential pattern.

492 **Definition 3.7** (Contiguous projected database). The set 493
 493 of contiguous projected sequences of P in SDB is called 494
 494 the contiguous projected database of P in SDB , denoted 495
 495 P_cSDB .

4. Introduction of two New Constraints: Robustness and Extended-Closedness

4.1. CCoSP mining in noisy data

498 CCoSP mining is currently one of the most efficient 499
 499 way to mine sequential patterns in data [58, 2]. In our 500
 500 review of the literature, we have raised the issue of 501
 501 mining closed contiguous sequential patterns in noisy 502
 502 data. As mentioned in the previous section, the noise 503
 503 in data represents a part of data that does not bring any 504
 504 additional information. In practice, the noise in data 505
 505 corresponds to three types of disturbances: loss, substi- 506
 506 tution or addition of items in the data (notice that most 507
 507 of the works of the literature only consider addition of 508
 508 items). The noise is therefore a random disturbance, 509
 509 poorly reproducible and infrequent, generally not found 510
 510 in frequent patterns. However, the noise decreases the 511
 511 support of some patterns, and can even make them not 512
 512 being part of the resulting set of patterns. For example, 513
 513 in a text mining context, noise may correspond to 514
 514 word omission (loss), spelling errors (substitution) or 515
 515 the use of irrelevant words (addition). The sentence 516
 516 “Elementary, my dear Watson” can thus become, in 517
 517 noisy data, “Elementary, my Watson” or “Elementary, 518
 518 my deer Watson” or even “Elementary, my dear old 519
 519 Watson” in case of imperfect typing of a user. However, 520
 520 this is in fact, three times the same pattern “Element- 521
 521 ary, my dear Watson” with different types of noise. 522
 522 Let us set up a closed contiguous sequential pattern 523
 523 mining with a minimum support $min_sup = 2$ in the 524
 524 sequence database *Noise* (presented in Table 1) made 525
 525 up of 5 sequences: two occurrences are the sequence 526
 526 $\langle Elementary, my, dear, Watson \rangle$ without noise and 527
 527 three occurrences occur with noise.

528 With the purpose of illustrating the impact of noise 529
 529 in closed contiguous sequential pattern mining, we 530
 530 exceptionally look for a specific pattern in the resulting

Table 1: Example of noisy sequence database *Noise*

sid	sequence	<i>CiCoSP</i>
#1	Elementary my dear Watson	$\langle \textit{Elementary}, \textit{my} \rangle : 5, \langle \textit{Elementary}, \textit{my}, \textit{dear} \rangle : 3,$
#2	Elementary my Watson	$\langle \textit{Elementary}, \textit{my}, \textit{dear}, \textit{Watson} \rangle : 2,$
#3	Elementary my deer Watson	$\langle \textit{Watson} \rangle : 5$
#4	Elementary my dear old Watson	
#5	Elementary my dear Watson	

set. Indeed, the pattern $\langle \textit{Elementary}, \textit{my}, \textit{dear}, \textit{Watson} \rangle$ represents the information we are looking for. In the context of sequential pattern mining, we define the information contained in a pattern as the tuple (pattern, support of the pattern).

From this mining, the pattern $\langle \textit{Elementary}, \textit{my}, \textit{dear}, \textit{Watson} \rangle$ has a support of 2 in this base of five sequences. This low frequency is partly due to the low support of the item *dear*: 3, which is substituted once (“deer”) and omitted once. The low support of the item *dear* causes a split of the pattern: $\langle \textit{Elementary}, \textit{my}, \textit{dear}, \textit{Watson} \rangle$ in two more frequent sub-patterns: $\langle \textit{Elementary}, \textit{my} \rangle, \langle \textit{Watson} \rangle$. The set of closed contiguous sequential patterns extracted then lacks irreversibly part of the information. In order to make *CiCoSP* relevant when mining noisy data, the three types of disturbances have to be handled. All the conclusions that will be drawn on the closed contiguous sequential pattern mining *CiCoSP* remain valid in contiguous sequential pattern mining *CoSP*.

On the one hand, wildcards [34, 49, 50] can be used to address the addition of items issue [2]. On the other hand, their use may lead to the mining of patterns that may not be relevant in the frame of contiguous sequential pattern mining. Indeed, a contiguous sequential pattern that always occurs with wildcards is unlikely to be a contiguous sequential pattern that occurs with “noise” items. It is more likely that it is a semi-contiguous sequential pattern, i.e. a contiguous sequential pattern whose all occurrences use a limited number of wildcards. For example, in the *Noise* database, the pattern $\langle \textit{Elementary}, \textit{my}, \textit{dear}, \textit{Watson} \rangle$ occurs twice contiguously (#1, #5) and once with one wildcard (#4). It can thus be considered as a contiguous pattern. However, the pattern $\langle \textit{Elementary}, \textit{my}, \textit{Watson} \rangle$ occurs contiguously only once (#2) and four times with one or two wildcards (#1, #3, #4, #5). This pattern is more likely to be a semi-contiguous pattern.

We now introduce some definitions related to the usage of wildcards in *CoSP* mining. Let $k \in \mathbb{N}$ be the number of wildcards authorized.

Definition 4.1 (*k*-contiguous sub-pattern). Given $P =$

$\langle E_1, E_2, \dots, E_n \rangle$ and $P' = \langle E'_1, E'_2, \dots, E'_m \rangle$ two sequential patterns, P is a *k*-contiguous sub-pattern of P' , denoted as $P \sqsubseteq_k P'$, if and only if $n < m$ and there exist integers j_1, j_2, \dots, j_n such that: (1) $1 \leq j_1 < j_2 < \dots < j_n < m$; and (2) $E_1 \subseteq E'_{j_1}, E_2 \subseteq E'_{j_2}, \dots, E_n \subseteq E'_{j_n}$; and (3) $j_n - j_1 - n \leq k$. P' is called a *k*-contiguous super-pattern of P , and P' *k*-contains P ($j_n - j_1 - n$ is the number of wildcards actually used).

Definition 4.2 (Absolute, Relative, Universal and Wildcard *k*-support). The absolute *k*-support of a sequential pattern P in a sequence database *SDB* is the number of tuples in *SDB* that *k*-contain P , denoted by $sup_{AW_k}^{SDB}(P)$. The relative *k*-support of P in *SDB* is the proportion of sequences in *SDB* that *k*-contain P , denoted by $sup_{RW_k}^{SDB}(P)$. The universal *k*-support of P in *SDB* is the number of occurrences with *k* wildcards of P in *SDB*, denoted by $sup_{UW_k}^{SDB}(P)$. The wildcard *k*-support of a sequential pattern P in a sequence database *SDB* is the number of tuples that *k*-contain P using at least 1 wildcard, denoted by $sup_{W_k}^{SDB}(P)$.

The subtraction of the wildcard *k*-support from the *k*-support of a pattern makes it possible to obtain the number of sequences where the pattern appears strictly respecting the contiguity constraint.

We now present in Table 2 the mining of closed 1-contiguous sequential patterns (with $k = 1$ wildcard) on the previous basis (Table 1), specifying the 1-support and wildcard 1-support of each pattern. Patterns impacted by the use of a wildcard are in marked red. The purpose of this new mining is still to extract the pattern $\langle \textit{Elementary}, \textit{my}, \textit{dear}, \textit{Watson} \rangle$, which represents the information. We see that using $k = 1$ wildcard allows to extract a new pattern: $\langle \textit{Elementary}, \textit{my}, \textit{Watson} \rangle$. This pattern does not contain all the information we are looking for, this is explained by the fact that this pattern is mined three times out of four times thanks to the use of a wildcard. Therefore, in the context of a closed contiguous sequential pattern mining, we consider the pattern $\langle \textit{Elementary}, \textit{my}, \textit{Watson} \rangle$ as not being a contiguous pattern.

As expected, the wildcard allows to

Table 2: Use of one wildcard in the mining of *CICoS P* in the *Noise* database

sid	sequence	<i>CICoS P</i> with one wildcard: support(wildcard 1-support)
#1	Elementary my dear Watson	$\langle Elementary, my \rangle : 5 (0), \langle Elementary, my, dear, Watson \rangle : 3 (1),$
#2	Elementary my Watson	$\langle Elementary, my, Watson \rangle : 4 (3),$
#3	Elementary my deer Watson	$\langle Watson \rangle : 5 (0)$
#4	Elementary my dear old Watson	
#5	Elementary my dear Watson	

614 find additional occurrences of the pattern 652
615 $\langle Elementary, my, dear, Watson \rangle$, here one addi- 653
616 tional occurrence in the sequence #4 and so allows 654
617 to overcome the presence of the “noise” item *old*. 655
618 This increase of 1 of the 1-support allows the pattern 656
619 $\langle Elementary, my, dear, Watson \rangle$ to have the same 657
620 1-support as the pattern $\langle Elementary, my, dear \rangle$, 658
621 and as $\langle Elementary, my, dear \rangle \sqsubseteq_1$
622 $\langle Elementary, my, dear, Watson \rangle$, it is no longer 659
623 closed and therefore is removed from the resulting set.
624 In conclusion, the use of $k = 1$ wildcard makes it
625 possible to alleviate the appearance of additional items
626 and thus to extract information not mined so far. In
627 return, the use of wildcards mines patterns that do not
628 necessarily respect the contiguity constraint. These
629 patterns are therefore not relevant for the extraction
630 of information in the frame of closed contiguous
631 sequential pattern mining. The problem is thus to limit
632 the use of wildcards.

633 4.2. Robustness Constraint

634 In order to answer this problem, we introduce a new
635 constraint: the robustness constraint. This constraint is
636 based on an adjustable parameter, the robustness ratio,
637 to limit the proportion of patterns in the resulting set
638 that use wildcards, without compromising on the use
639 of wildcards to mitigate the disturbance of additional
640 items during the mining process. This constraint can be
641 linked to the one proposed in [38] that aim at discarding
642 patterns that contain systematic errors.

643 **Definition 4.3** (k -contiguous δ -robust sequential pat- 680
644 tern). Given $\delta \in [0; 1]$ a ratio and *SDB* a sequence 681
645 database, a k -contiguous sequential pattern P is δ -robust 682
646 if and only if $\frac{sup_{W_k}^{SDB}(P)}{sup_{AW_k}^{SDB}(P)} \leq \delta$. 683

647 This definition reflects the fact that k -contiguous 686
648 δ -robust sequential patterns stand on the proportion of 687
649 contiguous occurrences of a pattern against occurrences 688
650 that use wildcards. This proportion is set by the value 689
651 of the robustness ratio δ , which is fixed by the user, 690

depending on the estimated amount of noise in the
data. The mining of k -contiguous δ -robust sequential
patterns, thus relies on the use of wildcards and a
robustness ratio (see next section).

In our running example (Table 1), for
both patterns mined with $k = 1$ wildcard:
 $\frac{sup_{W_k}^{Noise}(\langle Elementary, my, Watson \rangle)}{sup_{AW_k}^{Noise}(\langle Elementary, my, Watson \rangle)} = \frac{3}{4} = 0.75$ et
 $\frac{sup_{W_k}^{Noise}(\langle Elementary, my, dear, Watson \rangle)}{sup_{AW_k}^{Noise}(\langle Elementary, my, dear, Watson \rangle)} = \frac{1}{3} \approx 0.33$.
Thus, the mining of closed sequential patterns 1-
contiguous δ -robust when $\delta \in [0; 0.74]$ allows to
remove the pattern $\langle Elementary, my, Watson \rangle$ of the
resulting set and thus contributes to the improvement
of the apprehensibility of the set of patterns. For
 $\delta \in [0.34; 0.74]$, the set of extracted patterns becomes:
 $\langle Elementary, my \rangle : 5, \langle Elementary, my, dear, Watson \rangle : 3, \langle Watson \rangle : 5$.

To summarize, regular sequential pattern mining of-
ten produces an unmanageable set and regularly faces
issues related to both execution time and memory us-
age. Contiguous sequential pattern mining allows to im-
prove both the apprehensibility of the resulting set and
the efficiency of the mining. However, this mining loses
information when the data is noisy. The k -contiguous
 δ -robust sequential pattern mining is a solution that pre-
vents the mining of unexpected additional items in noisy
data and that guarantees, to a certain extent, the conti-
guity of patterns.

However, the other two disturbances that are item loss
and substitution, remain an issue when mining closed
contiguous sequential patterns in noisy data. Indeed,
in our running example, with $\delta \in [0.34; 0.74]$, the re-
sulting set is still made up of 3 patterns (see Table 2),
whereas we only look for the information of the pat-
tern $\langle Elementary, my, dear, Watson \rangle$. Therefore, this
set contains redundant information, which leads to a
loss of apprehensibility for the user. The closedness
constraint allows to remove the patterns having a super-
pattern of the same support.

4.3. Extended-Closedness Constraint

The problem raised by the loss and substitution of items in data is also the decrease of the support of the associated patterns. This decrease leads to the presence of sub-patterns in the resulting set that cannot be removed by the closedness constraint. Indeed, in the set of *CICoSP* from the *Noise* database, the pattern $\langle \text{Elementary}, \text{my}, \text{dear}, \text{Watson} \rangle$ has only a support of 2, while the sub-patterns $\langle \text{Elementary}, \text{my} \rangle$ and $\langle \text{Watson} \rangle$ are also *CICoSP* with a support of 5. Therefore, the closedness constraint cannot remove these sub-patterns. The resulting set thus becomes less apprehensible. As the loss and substitution of items is irreversible when mining noisy data, our goal is to mitigate the increase of the resulting set of patterns by removing some of the sub-patterns to improve the apprehensibility, at the price of the information regarding the support of these new sub-patterns. To achieve this goal, we propose to relax the closedness constraint to form a new constraint. The resulting extended-closedness constraint does not require that a sub-pattern has the same support than its super-pattern to be removed. The extended ratio λ , associated to this constraint, allows to set to what extent the support of a super-pattern can be lower than the support of its sub-patterns for them to be removed. We rely on the definition 3.3 to introduce the following definition:

Definition 4.4 (Extended-closed sequential pattern). Given $\lambda \in [0; 1]$ a ratio, SDB a sequence database, and P a closed sequential pattern. P is a λ -closed sequential pattern if and only if there exists no super-pattern P' verifying: $\frac{sup_A^{SDB}(P')}{sup_A^{SDB}(P)} \geq \lambda$.

If P' exists and is a λ -closed sequential pattern, the reference closed sequential pattern of P' is the shortest closed sequential pattern $P_c \subset P'$ verifying: $\frac{sup_A^{SDB}(P')}{sup_A^{SDB}(P_c)} \geq \lambda$.

We now present in Table 3 the use of the extended-closedness constraint depending on the value of the extended ratio. This new constraint will contribute to the reduction of the number of mined patterns. The configuration where $\lambda = 1$ is similar to the closed contiguous sequential pattern mining like in Table 1. In the configuration where $\lambda = 0.6$, the pattern $\langle \text{Elementary}, \text{my}, \text{dear} \rangle$ disappears from the set of extracted patterns, as: $\langle \text{Elementary}, \text{my}, \text{dear} \rangle \sqsubset_c \langle \text{Elementary}, \text{my}, \text{dear}, \text{Watson} \rangle$ and $\frac{sup_{AW_k}^{Noise}(\langle \text{Elementary}, \text{my}, \text{dear}, \text{Watson} \rangle)}{sup_{AW_k}^{Noise}(\langle \text{Elementary}, \text{my}, \text{dear} \rangle)} \simeq 0.66$.

Since $0.66 \geq \lambda = 0.6$, we consider that both patterns are considered to contain the same information, so $\langle \text{Elementary}, \text{my}, \text{dear} \rangle$ is not a 0.6-closed contiguous sequential pattern. In the configuration where $\lambda = 0.4$:

$$\frac{sup_{AW_k}^{Noise}(\langle \text{Elementary}, \text{my}, \text{dear}, \text{Watson} \rangle)}{sup_{AW_k}^{Noise}(\langle \text{Elementary}, \text{my} \rangle)} = \frac{sup_{AW_k}^{Noise}(\langle \text{Elementary}, \text{my}, \text{dear}, \text{Watson} \rangle)}{sup_{AW_k}^{Noise}(\langle \text{Watson} \rangle)} = 0.4.$$

Since $0.4 \geq \lambda = 0.4$, the only 0.4-closed contiguous sequential pattern is $\langle \text{Elementary}, \text{my}, \text{dear}, \text{Watson} \rangle$. A configuration with an extended ratio as low as $\lambda = 0.4$ allows a potential large loss of the support information as a 0.4-closed sequential pattern only needs to have 40% of the support of its sub-patterns to remove them.

As a λ -closed sequential pattern is defined as a particular closed sequential pattern, the complete set of λ -closed sequential patterns is a subset of the set of closed sequential patterns. Therefore, the complete set of λ -closed k -contiguous sequential patterns can be mined while mining closed k -contiguous sequential patterns.

4.4. Notations of the Patterns of Interest

To clarify the set of sequential patterns we are interested in, Table 4 presents the complete list, and the associated abbreviations.

4.5. Summary on the Running Example

We now exhaustively list the various sets of sequential patterns that can be mined from the sequence database *Noise* (Table 1) with $min_sup = 2$, $k = 1$ wildcard, $\delta = 0.6$ and $\lambda = 0.6$. The complete list of patterns is presented in Table 5.

The set of frequent contiguous sequential patterns *CoSP* is made up of 10 patterns and decreases to 4 patterns in the set of closed contiguous sequential patterns *CICoSP*. Both sets (grayed out in Table 5) are mined using the contiguity and closedness constraints of the literature.

The use of $k = 1$ wildcard (*CICo1* mining) results in a set of the same size (4 patterns). However, as seen previously, the 1-support of several patterns is increased:

- $\langle \text{Elementary}, \text{my}, \text{Watson} \rangle$, increases from 1 to 4. It thus becomes not only frequent, but also closed.
- $\langle \text{Elementary}, \text{my}, \text{dear}, \text{Watson} \rangle$, increases from 2 to 3.

In addition, $\langle \text{Elementary}, \text{my}, \text{dear} \rangle$ and $\langle \text{Elementary}, \text{my}, \text{dear}, \text{Watson} \rangle$ now have the same support, so $\langle \text{Elementary}, \text{my}, \text{dear} \rangle$ is no more closed.

Table 3: Impact of the extended-closedness constraint in the mining in *Noise* database

λ	$Cl_\lambda CoSP$
1	$\langle Elementary, my \rangle : 5, \langle Elementary, my, dear \rangle : 3, \langle Elementary, my, dear, Watson \rangle : 2, \langle Watson \rangle : 5$
0.6	$\langle Elementary, my \rangle : 5, \langle Elementary, my, dear, Watson \rangle : 2, \langle Watson \rangle : 5$
0.4	$\langle Elementary, my, dear, Watson \rangle : 2$

784 $\langle Elementary, my, Watson \rangle$ is mined three times out 824
785 of four thanks to the use of a wildcard, $\frac{3}{4} = 0,75 >$
786 $\delta = 0.6$. Thus, this pattern does not respect the 0.6- 826
787 robustness constraint. The set of $ClCo_1R_{0.6}$ patterns is 827
788 thus reduced to 3 patterns. 828

$$789 \frac{\langle Watson \rangle \sqsubset_1 \langle Elementary, my, dear, Watson \rangle \text{ and } 829 \\ \text{sup}_{AW_k}^{Noise}(\langle Elementary, my, dear, Watson \rangle)}{\text{sup}_{AW_k}^{Noise}(\langle Watson \rangle)} = 0.6. \quad 830$$

$$791 \text{Likewise,} \quad 832 \\ 792 \langle Elementary, my \rangle \sqsubset_1 \langle Elementary, my, dear, Watson \rangle \quad 833 \\ \text{and } \frac{\text{sup}_{AW_k}^{Noise}(\langle Elementary, my, dear, Watson \rangle)}{\text{sup}_{AW_k}^{Noise}(\langle Elementary, my \rangle)} = 0.6. \quad 834$$

793 The 0.6-closedness allows to remove both patterns 835
794 $\langle Watson \rangle$ and $\langle Elementary, my \rangle$. 836
795 837

796 The final set of patterns $Cl_{0.6}Co_1R_{0.6}$ is reduced 839
797 to one single pattern $\langle Elementary, my, dear, Watson \rangle$ 840
798 with a 1-support of 3, which greatly increases the 841
799 apprehensibility of the set of patterns, in comparison to 842
800 the set of $ClCoSP$. 843

801 In the 5 sequences from the *Noise* database, 2 844
802 sequences contain a contiguous occurrence of 845
803 $\langle Elementary, my, dear, Watson \rangle$ without noise and 846
804 3 of them contain it with disturbances of items of 847
805 type loss, substitution or addition (actually on of 848
806 each). We can conclude that the use of the robustness 849
807 constraint makes it possible to overcome the addition 850
808 of items by finding one additional occurrence of 851
809 $\langle Elementary, my, dear, Watson \rangle$ and by removing the 852
810 semi-contiguous sequential patterns appeared thanks 853
811 to the use of a wildcard. We can say that the use of 854
812 the extended-closedness constraint makes it possible to 855
813 disregard the patterns appearing due to the loss and the 856
814 substitution of items in the *Noise* database. 857
815 858

816 In conclusion, both new robustness and extended- 859
817 closedness constraints improve the apprehensibility of 860
818 the set of $ClCoSP$ mined from noisy data. Indeed, on 861
819 the one hand, they mitigate the addition of items in data, 862
820 while preserving the contiguity constraint as much as 863
821 possible. On the other hand, they increase the apprehen-
822 sibility of the resulting set, by avoiding segmentations in
823 patterns.

4.6. Problem statement

As part of our problem, which is to improve the ef-
ficiency of the pattern mining and the apprehensibility
of its results, we want to propose a sequential pattern
mining algorithm that natively integrates λ -closedness,
 k -contiguity, δ -robustness constraints, and allows for an
easy integration of a set of monotonic or anti-monotonic
constraints. Finally, the algorithm must answer the
following problem: given *SDB* a sequence database,
 min_sup a minimum support threshold, k a number of
wildcards, δ a robustness ratio, λ an extended ratio and
 ζ a set of monotonic or anti-monotonic constraints, how
to mine $Cl_\lambda Co_k R_\delta C_\zeta SP$, the complete set of λ -closed k -
contiguous δ -robust sequential patterns satisfying a set
of constraints ζ ?

5. $Cl_\lambda Co_k R_\delta C_\zeta SP$ Mining

In this section, we introduce the C3Ro algorithm
dedicated to $Cl_\lambda Co_k R_\delta C_\zeta SP$ mining. The name C3Ro
stands for **C**losed **C**ontiguous **R**obust **C**onstrained se-
quential pattern mining algorithm. C3Ro is based on
the pattern-growth philosophy, used in many sequen-
tial pattern mining algorithms, such as PrefixSpan [27],
BIDE [47], PG [44], CCPM [2], etc. C3Ro natively in-
tegrates closedness (as Clospan [51]) and contiguity (as
CCSpan [58]) constraints, in addition to any monotonic
or anti-monotonic constraints (as PG [44]). All these
constraints can be easily managed thanks to the pattern-
growth philosophy.

The main idea behind C3Ro is to provide a toolbox al-
lowing the final user to mine multi-constraints sequen-
tial patterns that fit his/her needs, through the use of sev-
eral meaningful parameters:

- k a maximum number of wildcards per pattern oc-
currence.

When $k = 0$, C3Ro mines contiguous sequential
patterns and when $k = \infty$, it mines regular se-
quential patterns. More importantly, the number
of wildcards is a way to mine semi-contiguous se-
quential patterns or to improve the mining of con-
tiguous sequential patterns in noisy data.

864 • δ a robustness ratio, to mine the newly introduced
 865 k -contiguous δ -robust sequential patterns.
 866 It allows C3Ro to limit the impact of wildcards on
 867 the set of occurrences of each pattern, thus on the
 868 resulting set of patterns, by adjusting the robust-
 869 ness ratio on the estimated amount of noise in the
 870 data.

871 • λ an extended ratio, to mine the newly introduced
 872 λ -closed sequential patterns.
 873 It allows C3Ro to alleviate the closedness con-
 874 straint with the extended ratio, and thus to extract
 875 a more compact set of patterns than the traditional
 876 set of closed sequential patterns. When $\lambda = 0$
 877 C3Ro mines maximal sequential patterns and when
 878 $\lambda = 1$, C3Ro mines closed sequential patterns.

879 • ζ a set of monotonic or anti-monotonic constraints.

880 In Table 6, we can see the adjustment of the extended
 881 ratio λ and the maximum number of wildcards k allows
 882 the C3Ro algorithm to extract numerous types of pat-
 883 tern. We do not make the robustness ratio vary ($\delta = 1$)
 884 as it does not impact the type of the patterns mined.

Table 6: Types of sequential patterns mined by C3Ro with $\delta = 1$

$k \backslash \lambda$	0]0; 1[1
0	maximal $CoSP$	$Cl_\lambda CoSP$	$ClCoSP$
]0; ∞ [maximal Co_kSP	$Cl_\lambda Co_kSP$	$ClCo_kSP$
∞	maximal SP	$Cl_\lambda SP$	$CISP$

885 C3Ro is able to mine a wide variety of patterns,
 886 depending on the number wildcards, the robustness
 887 ratio, the extended ratio and a set of monotonic or
 888 anti-monotonic constraints integrated in the mining.
 889 All these parameters have a non negative impact on
 890 the efficiency of C3Ro (time and memory), therefore
 891 the efficiency of C3Ro is at worst the one of BIDE.
 892
 893

894 We introduce the sequence database SDB (Table 7) to
 895 illustrate, throughout this section, how C3Ro algorithm
 896 works. The structure of C3Ro is the following:
 897 First, C3Ro uses a framework that enumerates frequent
 898 k -contiguous δ -robust sequential patterns satisfying a
 899 set of monotonic or anti-monotonic constraints. This
 900 framework is guided by the one of PrefixSpan to enu-
 901 merate frequent sequential patterns.
 902 Second, C3Ro uses a new extended-closedness check-
 903 ing scheme, a pruning technique and a Backscan

Table 7: Sequence database SDB

sid	sequence
#1	$\langle \{a\}\{b\}\{c\}\{bd\} \rangle$
#2	$\langle \{a\}\{b\}\{c\} \rangle$
#3	$\langle \{ab\}\{d\}\{b\}\{c\}\{d\} \rangle$

904 method, guided by the BIDE algorithm. We now present
 905 both elements.

906 5.1. Framework to Enumerate Frequent k -Contiguous 907 δ -Robust Sequential Patterns.

908 We start by introducing some definitions related to
 909 the projected database with k wildcards (in line with the
 910 definition of a contiguous projected database) and the
 911 prefix-monotone constraint, required by the framework
 912 we propose. Then we explain how to enumerate
 913 frequent k -contiguous δ -robust sequential patterns and
 914 illustrate it on our running example (Table 7).
 915

916 5.1.1. Preliminaries

917 Given SDB an input database, $S = \langle E'_1, E'_2, \dots, E'_m \rangle$
 918 an input sequence and $P = \langle E_1, E_2, \dots, E_n \rangle$ a sequential
 919 pattern with $P \sqsubseteq_k S$. By definition, there exist integers
 920 j_1, j_2, \dots, j_n such that: (1) $1 \leq j_1 < j_2 < \dots < j_n < m$
 921 ; and (2) $E_1 \subseteq E'_{j_1}, E_2 \subseteq E'_{j_2}, \dots, E_n \subseteq E'_{j_n}$; and (3)
 922 $j_n - j_1 - n \leq k$ (see definition 4.3):

923 **Definition 5.1** (Number of wildcards available). To ex-
 924 tend the k -prefix P in the sequence S , at least one
 925 wildcard has to be available. This number is equal to
 926 $k - (j_n - j_1 - n)$.

927 **Definition 5.2** (Projected k -items of a sequential pat-
 928 tern). The projected k -items of P in S are the union
 929 of two sets 1) the set of i -extensions e_i of each occur-
 930 rence of P in S such that $E_n \diamond_i e_i \subseteq E'_{j_n}$; 2) the set of
 931 s -extensions e_s of each occurrence of P in S such that
 932 $e_s \in E'_{j_{n+1}} \cup E'_{j_{n+2}} \dots \cup E'_{\min(j_n+(k-j_n-j_1-n)+1, m)}$.

933 **Definition 5.3** (Projected k -sequence). The set of pro-
 934 jected k -items of P in S is called the projected k -
 935 sequence of P in S .

936 **Definition 5.4** (k -prefix sequential pattern). P is called
 937 the k -prefix sequential pattern.

938 **Definition 5.5** (Projected k -database). The set of pro-
 939 jected k -sequences of P in SDB is called the projected
 940 k -database of P in SDB , denoted $P_{-k}SDB$.

941 **Definition 5.6** (Usage of wildcards). For each j_i with 990
 942 $i \in \llbracket 1; n \rrbracket$, if $j_{i+1} - j_i > 1$, we write $P =$ 991
 943 $\langle E_1, E_i, E_{i+1}^*, \dots, E_n \rangle$ to identify itemsets found with the 992
 944 use of a wildcard in a projected k -database.

945 In our running example, in the sequence $(a)(b)(c)$, 994
 946 the set of contiguous projected items (with no 995
 947 wildcards) of the prefix sequential pattern (a) is: 996
 948 $\{\{(b)\}, \{(b)\}, \{(_b), (d)\}\}$. 997

949 If a wildcard is used to find an item, a $*$ is denoted 998
 950 on the item it corresponds to. In our example, when 999
 951 $k = 1$, and no wildcard is used yet, all the first items of 1000
 952 second adjacent itemset after (a) are projected items too: 1001
 953 $\{\{(c)^*\}, \{(c)^*\}, \{(b)^*\}\}$. To summarize, the set of projected 1002
 954 1-items of the prefix sequential pattern (a) in $(a)(b)(c)$ 1003
 955 is $\{\{(b)\}, (c)^*\}, \{(b), (c)^*\}, \{(_b), (d), (b)^*\}\}$. 1004

956 **Definition 5.7** (Wildcard support in a projected 1005
 957 k -database). Given P a k -prefix sequential pattern and e 1006
 958 an item of its projected k -database. The wildcard sup- 1007
 959 port of e is the number of sequences in which e occurs, 1008
 960 using at least one wildcard and following an occurrence 1009
 961 of P that has used no wildcard.

962 We illustrate the wildcard support in a projected 1010
 963 k -database on our running example, with $k = 2$ and the 1011
 964 pattern $(a)(b)(d)$: 1012

965 In sequence sid #1, the occurrence of $(a)(b)$ has not 1013
 966 used any wildcard. One wildcard has to be used to find 1014
 967 (d) and form $(a)(b)(d)$. The used wildcard increases 1015
 968 the 2-support by 1. In sequence id #3, the 2-prefix 1016
 969 sequential pattern $(a)(b)$ has already used one wildcard 1017
 970 and needs to use a second wildcard to find (d) . Thus, 1018
 971 this wildcard does not increase the wildcard support 1019
 972 of (d) in the projected 2-database of $(a)(b)$. Therefore, 1020
 973 $sup_{W_2}^{(a)(b), SDB}((d)) = 1$ 1021

974 The following definition and lemma come from [44]. 1022

976 **Definition 5.8** (Prefix-monotone constraint). A con- 1023
 977 straint C is called prefix anti-monotonic if for each se- 1024
 978 quential pattern P satisfying the constraint, every prefix 1025
 979 of P also satisfies C . A constraint C is called prefix- 1026
 980 monotonic, if for each sequential pattern P satisfying 1027
 981 the constraint C , each sequence having P as a prefix, 1028
 982 also satisfies C . A constraint is called prefix-monotone 1029
 983 if it is either prefix anti-monotonic or prefix monotonic.

984 **Lemma 5.1** (Prefix-monotone constraint principle). 1030
 985 Given a prefix-monotone constraint C and a sequential 1031
 986 pattern P . 1032

987 1. When C is prefix anti-monotonic, if $C(P) = false$, 1033
 988 then there exists no sequential patterns P' that has P as 1034
 989 a prefix with: $C(P') = true$. 1035

2. When C is prefix monotonic, if $C(P) = true$, then ev-
 ery sequential pattern P' having P as a prefix verifies:
 $C(P') = true$.

993 **Theorem 5.2** (Pruning in a projected k -database).
 994 Given a k -prefix sequential pattern P in a sequence
 995 database SDB and e an item of the projected k -database
 996 of P . If an item e' always appears after e in the pro-
 997 jected k -database of P , e' can be safely removed from
 998 the projected k -database.

999 *Proof.* Given e and e' two items, P a k -prefix sequen-
 1000 tial pattern and SDB a sequence database. If each oc-
 1001 currence of an itemset e' always appears after an oc-
 1002 currence of an itemset e in each projected k -sequence
 1003 of P , it means that there exists a sequential pattern
 1004 $Q = \langle P \diamond e \diamond e' \rangle$ with $sup_{UW_k}^{SDB}(Q) = sup_{UW_k}^{SDB}(\langle P \diamond e' \rangle)$.
 1005 Thus, each occurrence of the sequential pattern $\langle P \diamond e' \rangle$ is
 1006 included in an occurrence of Q and so are its extensions.
 1007 Therefore, it is useless to extend P with e' , only with e
 1008 and e' can be pruned from the projected k -database of
 1009 P . \square

1010 **Theorem 5.3** (k -contiguous δ -robust sequential pat-
 1011 tern projection). Given $\delta \in [0; 1]$ the robustness ra-
 1012 tio, P a k -contiguous δ -robust sequential pattern and
 1013 e a frequent item from its projected k -database. $\langle P \diamond e \rangle$
 1014 is a k -contiguous δ -robust sequential pattern if
 1015 $\frac{sup_{W_k}^{SDB}(P) + sup_{W_k}^{P, SDB}(e)}{sup_{AW_k}^{P, SDB}(e)} \leq \delta$.

1016 *Proof.* Given P a k -contiguous δ -robust sequential pat-
 1017 tern and e a frequent item from its projected database
 1018 with $sup_{W_k}^{SDB}(P) + sup_{W_k}^{P, SDB}(e) \leq sup_{AW_k}^{P, SDB}(e) * \delta$.
 1019 Building on the definition 5.7, $sup_{W_k}^{SDB}(P) +$
 1020 $sup_{W_k}^{P, SDB}(e) = sup_{W_k}^{SDB}(\langle P \diamond e \rangle)$. As $sup_{AW_k}^{P, SDB}(e) =$
 1021 $sup_{AW_k}^{SDB}(\langle P \diamond e \rangle)$, if $sup_{W_k}^{SDB}(\langle P \diamond e \rangle) \leq sup_{AW_k}(e) * \delta$ then
 1022 $\langle P \diamond e \rangle$ is a k -contiguous δ -robust sequential pattern. \square

5.1.2. Enumeration of Frequent $Co_k R_\delta C_\zeta S P$.

In C3Ro, the complete search space of k -contiguous
 sequential patterns forms a sequential pattern tree, as
 in [7]. The root node of the tree is at the top level
 and is labeled \emptyset . To form the tree, C3Ro recursively
 extends a node (referred to as a k -prefix sequential
 pattern) at a certain level in the tree by adding at most
 $k + 1$ contiguous itemsets per occurrence of the k -prefix
 sequential pattern.

As C3Ro is designed to mine patterns satisfying ζ a
 set of monotonic and/or anti-monotonic constraints, the
 framework only extends the k -prefix sequential pattern
 with frequent items according to Lemma 5.1, depending

1036 on the given constraint(s) in the ζ set. Applying this 1085
 1037 Lemma greatly reduces the search space, hence the 1086
 1038 efficiency of C3Ro. It also reduces the resulting set, 1087
 1039 by removing irrelevant patterns. The Lemma is used 1088
 1040 in C3Ro in the following way: when all frequent 1089
 1041 items satisfying ζ are identified, each one becomes 1090
 1042 a k -prefix sequential pattern to be extended (at this 1091
 1043 step the algorithm is at the first level of the tree). The 1092
 1044 framework then builds the projected k -database of each 1093
 1045 k -prefix. The wildcard support of all the items must be 1094
 1046 evaluated in order to apply Theorem 5.3. Finally, only 1095
 1047 the frequent items in the projected k -database with a 1096
 1048 wildcard support verifying Theorem 5.3 can extend the 1097
 1049 k -prefix in k -contiguous δ -robust sequential patterns. 1098
 1050 This process is recursively repeated for each k -prefix to 1099
 1051 mine the resulting set of $Co_kR_\delta C_\zeta SP$. 1100

1052
 1053 Now we rely on our running example to describe 1101
 1054 how to enumerate frequent closed k -contiguous sequen- 1102
 1055 tial patterns with $k = 1$ wildcard. The 1-prefix sequen- 1103
 1056 tial pattern $P = (a)$ is the first one studied (due 1104
 1057 to the lexicographic order). Its projected 1-database 1105
 1058 is: $\{(b), (c)^*\}, \{(b), (c)^*\}, \{(-b), (d), (b)^*\}$. In this pro- 1106
 1059 jected 1-database, (b) has an 1-support of 3 and (c) 1107
 1060 has an 1-support of 2. As both 1-support exceed 1108
 1061 min_sup , they can extend (a) . However, in the pro- 1109
 1062 jected 1-database, (b) always occurs before $(c)^*$. Ac- 1110
 1063 cording to Theorem 5.8, (a) can be extended with (b) , 1111
 1064 while (c) can be removed. Then, given the 1-prefix 1112
 1065 sequential pattern $(a)(b)$, its projected 1-database is: 1113
 1066 $\{(c), (b)^*\}, \{(c)\} \{(c)\}$. (c) has a 1-support of 3 and can 1114
 1067 extend $(a)(b)$. The projected 1-database of $(a)(b)(c)$ is: 1115
 1068 $\{((b), (d)), \{\emptyset\} \{(d)\}$. (d) is the last item that can ext- 1116
 1069 end $(a)(b)(c)$. The 1-prefix sequential pattern (a) is 1117
 1070 now fully extended. The same method can be used with 1118
 1071 $P = (b)$, $P = (c)$ and $P = (d)$. 1119

1072 5.2. BI-Directional λ -closedness Checking with Wild- 1120 1073 cards. 1121

1074 Before explaining the two main steps that are 1122
 1075 forward-extensions and backward-extensions, we start 1123
 1076 by explaining global strategy adopted by C3Ro for 1124
 1077 checking closedness and λ -closedness. Then, we detail 1125
 1078 the Backscan pruning method. 1126

1079 5.2.1. Closedness and λ -closedness checking strategy 1127

1080 The previous framework mines k -contiguous δ -robust 1128
 1081 frequent sequential patterns, satisfying prefix-monotone 1130
 1082 constraints. To get the set of frequent λ -closed 1131
 1083 sequential patterns, an λ -closedness checking has to be 1132
 1084 performed. C3Ro λ -closedness checking scheme is 1133

based on the same efficient BI-Directional design than BIDE to check closed sequential patterns.

As the definition 4.4 of $Cl_\lambda Co_k SP$ is based on the definition 3.3 of $CISP$, we first must identify them to be able to identify $Cl_\lambda Co_k SP$. We introduce and redefine here several concepts and theorems of the BIDE [47] algorithm in order to use them in the context of $Cl_\lambda Co_k R_\delta C_\zeta SP$ mining:

According to the definition 4.4, if $P = \langle E_1, E_2, \dots, E_n \rangle$ a $Co_k SP$, is not closed, it means that there exists at least one item e , which can be used to extend P to get P' a $Co_k SP$, with $sup_{AW_k}(P) = sup_{AW_k}(P')$. In line with BIDE, P can be extended into P' in three ways: (1) $P' = \langle E_1, E_2, \dots, E_n \rangle \diamond e$ (2) $P' = e \diamond \langle E_1, E_2, \dots, E_n \rangle$ and (3) $\exists i, 1 < i < n, P' = \langle E_1, E_2, \dots, E_i \rangle \diamond e \diamond_s \langle E_{i+1}, \dots, E_n \rangle$. (1) e occurs after the itemset E_n , BIDE challenges e a forward-extension item of P . (2) and (3) e occurs before itemset E_n , BIDE calls e a backward-extension item of P .

Theorem 5.4 (BI-Directional extended-closedness checking). *P a sequential pattern is closed, if and only if P has no backward-extension and no forward-extension items.*

The closedness checking strategy BIDE relies on forward-extension and backward-extension detection. We use the same strategy in the context of $Cl_\lambda Co_k R_\delta C_\zeta SP$.

Similarly to $CISP$, if a closed sequential pattern $P = \langle E_1, E_2, \dots, E_n \rangle$, is not an $Cl_\lambda Co_k SP$, it means that there exists at least one item e , which can be used to extend P to get a new sequential pattern P' , with $sup_{AW_k}(P_c) * \lambda \leq sup_{AW_k}(P')$ (where P_c is the reference closed sequential pattern of P' with $P_c \sqsubseteq_k P$). The closed sequential pattern P can be extended into P' in three ways: (1) $P' = \langle E_1, E_2, \dots, E_n \rangle \diamond e$ (2) $P' = e \diamond \langle E_1, E_2, \dots, E_n \rangle$ and (3) $\exists i, 1 < i < n, P' = \langle E_1, E_2, \dots, E_i \rangle \diamond e \diamond_s \langle E_{i+1}, \dots, E_n \rangle$. (1) e occurs after the itemset E_n , we call e a λ -forward-extension item of P . (2) and (3) e occurs before itemset E_n , we call e a λ -backward-extension item of P . Given the above elements, the following theorem becomes obvious, according to the definition of a frequent λ -closed sequential pattern with wildcards (definition 4.4).

Theorem 5.5 (BI-Directional extended-closedness checking). *P a $Co_k SP$ is a $Cl_\lambda Co_k SP$, if and only if P has no λ -backward-extension and no λ -forward-extension items.*

We have just shown that thanks to the four types of extension we can check if a pattern is closed and then if it is λ -closed.

We propose to follow the procedure below to identify the λ -closed sequential patterns when enumerating the $Co_kR_\delta C_\zeta SP$. We evaluate each k -prefix $Co_kR_\delta C_\zeta SP$ that has just been validated by the enumeration framework to first determine if this k -prefix is closed. Then the k -prefix is kept if λ -closed, if not, a super-pattern necessary is. Thus, the support of the reference closed sequential pattern is kept to find the λ -closed sequential pattern in a future extension of this k -prefix.

5.2.2. Forward-extensions and λ -forward-extensions checking

We now introduce the lemmas to identify the four types of extension and evaluate the closedness and the λ -closedness of a pattern. We start with both lemmas to check forward-extension and λ -forward-extension:

Lemma 5.6 (Forward-extension item checking). *Given P a $ClCo_kSP$ and SDB a sequence database, its complete set of forward-extensions items is the set of items e in its projected k -database that verifies: $sup_{AW_k}^{P \diamond SDB}(e) = sup_{AW_k}^{SDB}(P)$.*

Proof. For each item e in the projected k -database of P , a new Co_kSP $P' = \langle P \diamond e \rangle$ can be created, if $sup_{AW_k}^{P \diamond SDB}(e) = sup_{AW_k}^{SDB}(P)$ thus $sup_{AW_k}^{SDB}(P') = sup_{AW_k}^{SDB}(P)$ then P is not a $ClCo_kSP$ and $\langle e \rangle$ is a forward-extension of P . \square

Lemma 5.7 (λ -forward-extension item checking). *Given P a $ClCo_kSP$, P_c its reference closed sequential pattern and SDB a sequence database, its complete set of λ -forward-extensions items is the set of items e in its*

projected k -database that verifies: $\frac{sup_{AW_k}^{P \diamond SDB}(e)}{sup_{AW_k}^{SDB}(P_c)} > \lambda$.

Proof. For each item e in the projected k -database of P , a new Co_kSP $P' = \langle P \diamond e \rangle$ can be created, if $\frac{sup_{AW_k}^{P \diamond SDB}(e)}{sup_{AW_k}^{SDB}(P_c)} > \lambda$ thus $\frac{sup_{AW_k}^{P \diamond SDB}(P')}{sup_{AW_k}^{SDB}(P_c)} > \lambda$ then P is not a $ClCo_kSP$ and $\langle e \rangle$ is a λ -forward-extension of P . \square

These lemmas show that the evaluation of the k -support of items in the projected k -database of a k -prefix allows to determine if this k -prefix has both a forward-extension and a λ -forward-extension.

5.2.3. Backward-extensions and λ -backward-extensions checking

We now present how to identify backward-extensions and λ -backward-extensions. These extensions combine two ways to extend the k -prefix: either the item appears between the first and last itemset of the k -prefix, or the item appears before the first itemset of the k -prefix. We introduce a theorem allowing a pruning of extensions between the first and the last itemset of the k -prefix, during the enumeration phase.

Theorem 5.8. *Given P a $Co_kR_\delta C_\zeta SP$, SDB a sequence database and e_1 an item of the projected k -database of P . If an item e_2 always appears after e_1 in each projected k -sequence of P then e_2 can be removed from the projected k -database of P .*

Proof. Given P a $Co_kR_\delta C_\zeta SP$, SDB a sequence database and e_1, e_2 two items of the projected k -database of P . If e_2 always appears after e_1 in each projected k -sequence of P , then it means that there exists P' a $Co_kR_\delta SP$ such as $P' = \langle P \diamond e_1 \diamond e_2 \rangle$ with $sup_{UW_k}^{SDB}(P') = sup_{UW_k}^{SDB}(\langle P \diamond e_2 \rangle)$. Each occurrence of $\langle P \diamond e_2 \rangle$ is included in an occurrence of P' , thus all its extensions are also included the extensions of P' . Therefore, it is useless to extend P with e_2 and e_2 can be removed from the projected k -database of P . \square

Thanks to this theorem, λ -backward-extensions between the first and last itemset of the k -prefix are pruned during the enumeration phase, so we can focus only on identifying backward-extensions and λ -backward-extensions when the item appears before the first itemset of the k -prefix. Given SDB a sequence database, S a sequence $\langle E'_1, E'_2, \dots, E'_m \rangle$ from SDB and P a $Co_kR_\delta C_\zeta SP$ $\langle E_1, E_2, \dots, E_n \rangle$ in SDB such as $P \sqsubseteq_k S$. It means there exists integers j_1, j_2, \dots, j_n such as: (1) $1 \leq j_1 < j_2 < \dots < j_n \leq m$; (2) $E_1 \subseteq E'_{j_1}, E_2 \subseteq E'_{j_2}, \dots, E_n \subseteq E'_{j_n}$; (3) $j_n - j_1 - n \leq k$. The number $k - (j_n - j_1 - n)$ represents the number of wildcards available for P .

Definition 5.9 (Backward k -sequence). The backward k -sequence of P in S is the union of the set of i -extensions e_i such as $e_i \diamond_i E_1 \subseteq E'_{j_1}$ and the set of s -extensions e_s such as $e_s \in E'_{j_1-1} \cup E'_{j_1-2} \dots \cup E'_{\max(j_1 - (k - (j_n - j_1 - n)) - 1, 1)}$, denoted $P_{-B_k}S$.

Definition 5.10 (Backward k -database). The set of backward k -sequences of P in SDB is called the backward k -database of P in SDB , denoted $P_{-B_k}SDB$.

We introduce the two lemmas to identify the backward-extensions and λ -backward-extensions:

Lemma 5.9. Given P a $ClCo_kSP$ and SDB a sequence database, its complete set of forward-extensions items is the set of items e in its projected k -database that verifies: $sup_{AW_k}^{P-B_kSDB}(e) = sup_{AW_k}^{SDB}(P)$.

Proof. For each item e in the backward k -database of P , a new Co_kSP $P' = \langle e \diamond P \rangle$ can be created, if $sup_{AW_k}^{P-B_kSDB}(e) = sup_{AW_k}^{SDB}(P)$ thus $sup_{AW_k}^{SDB}(P') = sup_{AW_k}^{SDB}(P)$ then P is not a $ClCo_kSP$ and $\langle e \rangle$ is a backward-extension of P . \square

Lemma 5.10. Given a $ClCo_kSP$ P and its closed sequential pattern P_c in a sequence database SDB , its complete set of λ -backward-extensions items is the set of items e in its backward k -database that verifies:

$$\frac{sup_{AW_k}^{P-B_kSDB}(e)}{sup_{AW_k}^{SDB}(P_c)} > \lambda.$$

Proof. For each item e in the backward k -database of P , a new Co_kSP $P' = \langle e \diamond P \rangle$ can be created, if $sup_{AW_k}^{P-B_kSDB}(e) > \lambda$ thus $\frac{sup_{AW_k}^{P-B_kSDB}(P')}{sup_{AW_k}^{SDB}(P_c)} > \lambda$ then P is not a $ClCo_kSP$ and $\langle e \rangle$ is a λ -backward-extension of P . \square

The theorem 5.8 and the lemmas 5.9 and 5.10 show that the evaluation of the k -support of items in the backward k -database of a k -prefix allows to identify its backward-extensions and λ -backward-extensions. Therefore, the strategy we propose to verify the closedness and the λ -closedness of a pattern is based on the evaluation of the k -support of the items in the backward k -database of each k -prefix.

5.2.4. Backscan pruning method

A new Backscan pruning method has been designed in order to avoid mining non-closed sequential patterns. The idea behind the Backscan pruning is to identify items that always appear contiguously with the k -prefix in each backward sequence.

Theorem 5.11 (Backscan pruning). Given a k -prefix P in a sequence database SDB . If an item e is always the first item before each occurrence of P , then P can be safely stopped to be extended.

Proof. Given an item e , a k -prefix P and a sequence database SDB . If an item e is always the first item before each occurrence of P , it means that there exists a Co_kSP $Q = \langle e \diamond P \rangle$ using the same number of wildcards as P with $sup_{UW_k}^{SDB}(Q) = sup_{UW_k}^{SDB}(P)$. Thus, the sequential pattern P is always included in Q and so are its extensions and it is useless to extend P . \square

5.3. Illustration of $ClCo_kR_\delta C_\zeta SP$ mining with C3Ro

In the previous section, we have introduced the definitions and theorems necessary to enumerate the $Co_kR_\delta C_\zeta SP$ (section 5.1.2) and then to evaluate their λ -closedness (section 5.1.3). Before presenting Algorithm C3Ro, we first illustrate its process. We rely on SDB sequences (Table 7) to extract the $Cl_{0.65}Co_1R_{0.6}$ patterns with $min_sup = 2$.

Items $a : 3, b : 3, c : 3$ and $d : 2$ are frequent. In alphabetical order, the 1-prefix $P = \langle \{a\} \rangle$ is the first pattern to be extended. The first step is the enumeration of the $Co_1R_{0.6}SP$ of 1-prefix P . The projected 1-database of P is $\{\langle \{b\}, \{c^*\} \rangle, \langle \{b\}, \{c^*\} \rangle, \langle \{b\}, \{d\}, \{b^*\} \rangle\}$. Items b and c are frequent in the projected 1-database of $\langle \{a\} \rangle$. We notice that item c is always preceded by item b in the projected 1-database of $\langle \{a\} \rangle$. In accordance with Theorem 5.8, item c can therefore be removed from the projected 1-database. We also note that $\frac{sup_{W_k}^{SDB}(\langle \{a\} \rangle) + sup_W^{M-kSDB}(c)}{sup_{AW_k}^{M-1SDB}(c)} = \frac{0+2}{2} = 1 > 0.6$, according to Theorem 5.3 item c can also be removed from the projected 1-database of $P = \langle \{a\} \rangle$. Theorem 5.8 allows pruning by identifying if one item always precedes another, while Theorem 5.3 evaluates the k -support and wildcard k -support of each item and prunes items that cannot form δ -robust sequential patterns. The second step is the evaluation of the 0.65-closedness of the 1-prefix P . Knowing that item b has a 1-support of 3, just like the 1-prefix $P = \langle \{a\} \rangle$, according to Lemma 5.6, this 1-prefix is neither closed nor 0.65-closed.

We now consider the 1-prefix $P = \langle \{a\}, \{b\} \rangle$ with the two same steps: enumeration of the $Co_1R_{0.6}SP$ of 1-prefix P and evaluation of the 0.65-closedness of the 1-prefix P . The projected 1-database of P is: $\{\langle \{c\}, \{bd^*\} \rangle, \langle \{c\} \rangle, \langle \{c\}, \{d^*\} \rangle\}$. Items $\{c\}, \{d\}$ are frequent. Item d is always preceded by item c in the projected 1-database, so it is useless to keep d in the projected 1-database. Item c has a 1-support of 3, just like P , the pattern $P = \langle \{a\}, \{b\} \rangle$ is neither closed nor 0.65-closed.

We consider the 1-prefix $P = \langle \{a\}, \{b\}, \{c\} \rangle$ and repeat the same two steps, its projected 1-database is: $\{\langle \{bd\} \rangle, \langle \{d\} \rangle\}$. Item d is frequent, so it can extend the prefix. There are no items of the same 1-support as the 1-prefix in the projected 1-database and therefore no forward-extensions. Since the backward 1-database of the 1-prefix is empty because it has no item before item a , therefore P has no backward-extensions either. According to Theorem 5.4, the $\langle \{a\}, \{b\}, \{c\} \rangle$ pattern is closed. According to Lemma 5.7, d is a 0.65-forward-extension of $\langle \{a\}, \{b\}, \{c\} \rangle$ be-

1317 cause the 1-support of d in the projected 1-database

1318 checks: $\frac{sup_{AW_k}^{\langle\{a\},\{b\},\{c\}\rangle_{-B_1}SDB}(d)}{sup_{AW_k}^{SDB}(\langle\{a\},\{b\},\{c\}\rangle)} = \frac{2}{3} = 0.66 \geq \lambda = 0.65$

1319 Therefore, $\langle\{a\},\{b\},\{c\}\rangle$ is not 0.65-closed.

1320 The 1-prefix $P = \langle\{a\},\{b\},\{c\},\{d\}\rangle$ is now considered.

1321 This 1-prefix has neither a backward 1-database nor
1322 a projected 1-database, because no item occur before
1323 or after P . Thus, P has no extension. According to
1324 Theorem 5.5, $\langle\{a\},\{b\},\{c\},\{d\}\rangle$ is therefore the only
1325 0.65-closed sequential pattern having as first item a .

1326 The same method can be applied with the 1-prefixes
1327 $P = \langle\{b\}\rangle$, $P = \langle\{c\}\rangle$ and $P = \langle\{d\}\rangle$. Finally,
1328 $Cl_{0.65}Co_1R_{0.6}SP$ is made up of only one pattern:
1329 $\{\langle\{a\},\{b\},\{c\},\{d\}\rangle\}$ compare to the 9 $CoSP$.

1330

1331 To summarize, in the frame of a $ClCoSP$ mining, the
1332 use of $k = 1$ wildcard allows to discover new patterns
1333 that may not appear in the resulting set due to noisy
1334 data (2 patterns). The 0.6-robustness constraint sets the
1335 limit between contiguous sequential patterns and semi-
1336 contiguous sequential patterns. In this configuration,
1337 if a pattern occurs more than 4 times out of 10 with
1338 a wildcard, it is considered semi-contiguous and thus
1339 not relevant (1 pattern). The 0.65-closedness constraint
1340 allows to remove sub-patterns having a support 35
1341 percent higher than a super-pattern and thus to remove
1342 more patterns than the closedness constraint (1 more
1343 pattern).

1344 We have here an illustration of the capacity of the
1345 robustness constraint and the extended-closedness
1346 constraint to reduce the resulting.

1347

1348 The Table 8 is a summary of the various frequent se-
1349 quential patterns with $min_sup = 2$, which we have pre-
1350 sented throughout this section.

1351 5.4. The C3Ro Algorithm.

1352 In this section we present the C3Ro algorithm (Algo-
1353 rithm 1), which integrates the framework that searches
1354 frequent k -contiguous δ -robust sequential patterns sat-
1355 isfying a set of prefix-monotone constraints ζ (section
1356 5.1), before checking the λ -closedness (section 5.2).

1357 The input parameters of C3Ro are SDB , a sequence
1358 database, min_sup a minimum support, ζ a set of prefix-
1359 monotones constraints, δ a robustness ratio, λ an ex-
1360 tended ratio and k a number of wildcards. C3Ro starts
1361 with an evaluation of the k -support of each item to
1362 extract $F1$, the set of frequent items satisfying the
1363 set of constraints ζ , according to the rules introduced
1364 in [44], via the function **Enumeration_items** (line 2).
1365 The Backscan function (line 4) checks if $f1$ can be

Algorithm 1 C3Ro($SDB, min_sup, \zeta, \delta, \lambda, k$)

Input: SDB a sequence database, min_sup a minimum support, ζ a set of prefix-monotones constraints, δ a robustness ratio, λ an extended ratio and k a number of wildcards.

Output: F , the $Cl_\lambda Co_k R_\delta C_\zeta SP$

```

1:  $F = \{\emptyset\}$ 
2:  $F1 = \mathbf{Enumeration\_items}(SDB, min\_sup, \zeta)$ ;
3: for each  $f1$  in  $F1$  do
4:   if Backscan( $f1$ )
5:      $f1\_SDB = \mathbf{P\_kdatabase}(SDB, f1, k)$ ;
6:      $\mathbf{pExt}(f1\_SDB, f1, min\_sup, \zeta, \delta, \lambda, k, -1, F)$ ;
7:   return  $F$ ;
```

Algorithm 2 $\mathbf{pExt}(P_kSDB, P, min_sup, \zeta, \delta, \lambda, k, cl_sup, F)$

Input: P_kSDB the projected k -database of k -prefix P , P a k -prefix, min_sup a minimum support, ζ a set of prefix-monotones constraints, δ a robustness ratio, λ an extended ratio and k a number of wildcards, cl_sup the support the previous $ClSP$ and F the previous set of $Cl_\lambda Co_k R_\delta C_\zeta SP$.

Output: F , the current set of $Cl_\lambda Co_k R_\delta C_\zeta SP$

```

1:  $FI = \mathbf{Enumeration}(P\_SDB, min\_sup, \zeta, \delta, k)$ ;
2:  $BE = \mathbf{Backward\_extension}(P, k)$ 
3:  $FE = \left\{ \left\{ e \in FI \mid sup_{AW_k}^{P\_SDB}(e) = sup^{SDB}(P) \right\} \right\}$ ;
4: if  $(\{BE \cup FE\} == \{\emptyset\})$ 
5:   if  $cl\_sup == -1$ 
6:      $cl\_sup = sup_{AW_k}^{SDB}(P)$ ;
7:    $\lambda BE = \lambda\text{-Backward\_extension}(P, k)$ 
8:    $\lambda FE = \left\{ \left\{ e \in FI \mid sup_{AW_k}^{P\_SDB}(e) \geq cl\_sup * \lambda \right\} \right\}$ ;
9:   if  $(\{\lambda BE \cup \lambda FE\} == \{\emptyset\})$ 
10:     $cl\_sup = -1$ ;
11:   if  $\mathbf{Check\_C}(P, \zeta)$ 
12:      $F = F \cup \{P\}$ ;
13: for each  $i$  in  $FI$  do
14:    $P_i = \langle P \diamond i \rangle$ ;
15:   if Backscan( $P_i$ )
16:      $P_i\_SDB = \mathbf{P\_kdatabase}(P\_SDB, P_i, k)$ ;
17:      $\mathbf{pExt}(P_i\_SDB, P_i, min\_sup, \zeta, \delta, \lambda, k, cl\_sup, F)$ ;
```

1366 pruned according to Theorem 5.11. The projected k - 1416
1367 database of each item in FI is created with the function 1417
1368 **P.kdatabase** (line 5). Then, the sub-algorithm **pExt** is 1418
1369 called with each frequent items satisfying all the prefix- 1419
1370 monotonic constraints ζ (line 5), its goal is the recursive 1420
1371 path of the projected k -databases of each k -prefix to ex- 1421
1372 tract the complete set of patterns $Cl_\lambda Co_k R_\delta C_\zeta$. In the 1422
1373 sub-algorithm **pExt**, the function **Enumeration** allows 1423
1374 the enumeration of the $Co_k R_\delta C_\zeta$ patterns in the pro- 1424
1375 jected k -database of the k -prefix P (line 1). The function 1425
1376 **Backward_extension** obtains the BE set of backward- 1426
1377 extensions of a the k -prefix P according to the lemma 1427
1378 5.9 (line 2). FE is the set of forward-extensions of the k - 1428
1379 prefix P obtained according to the lemma 5.6 (line 3). If 1429
1380 sets BE and FE are empty, then the k -prefix P is closed 1430
1381 and its k -support must therefore be kept for the check- 1431
1382 ing of the λ -closedness constraint (lines 4,5,6). In the 1432
1383 case where the k -prefix P is closed, the λ -backward- 1433
1384 extensions and λ -forward-extensions are searched ac-
1385 cording to the lemmas 5.10 and 5.7 (lines 9,10). If the
1386 k -prefix has no extension and satisfies all the constraints 1434
1387 ζ (line 11), then it is added to the F set of $Cl_\lambda Co_k R_\delta C_\zeta$
1388 (line 12) patterns. The sub-algorithm **pExt** is called re- 1435
1389 cursively (line 16) with a new k -prefix: the P pattern 1436
1390 extended with each item in the FI set, created during 1437
1391 the enumeration phase (line 1). When all k -prefixes 1438
1392 have been extended, the F set is the complete set of 1439
1393 $Cl_\lambda Co_k R_\delta C_\zeta$. 1440

1394 C3Ro is thus an answer to the second scientific ques- 1441
1395 tion: How to design a sequential pattern mining algo- 1442
1396 rithm that aggregates several constraints such as mono- 1443
1397 tonic, anti-monotonic, closedness and contiguity con- 1444
1398 straints and thus can be used with several combinations 1445
1399 of constraints, whatever the needs of a user are? 1446

1400 6. Discussion 1447

1401 The review of the literature of sequential pattern min- 1450
1402 ing highlighted that mining large and noisy databases 1451
1403 remains an important issue although such databases are 1452
1404 common in many practitioners' application contexts. 1453
1405 Using constraints during the mining process is a promis- 1454
1406 ing direction to alleviate this issue as they improve both 1455
1407 the apprehensibility of the set of patterns and the effi- 1456
1408 ciency of the mining process. However, no algorithm 1457
1409 in the literature integrates constraints dedicated to noisy 1458
1410 databases. 1459

1411 C3Ro alleviates these limits. It is a generic sequential 1460
1412 pattern mining algorithm, able to mine both large and 1461
1413 noisy databases. It relies on two newly defined con- 1462
1414 straints and on the use of wildcards. Given that mem- 1463
1415 ory is the most limiting resource when mining large 1464

databases, C3Ro is designed to limit memory usage. As
a consequence, C3Ro may be less efficient in terms of
execution time in comparison to recent algorithms that
use vertical IDLists representation.

Furthermore, C3Ro is designed to be fully practitioner-
oriented, and uses understandable parameters that make
sense in practitioners' application context. Neverthe-
less, these parameters are numerical parameters that
need to be set by the user. To ensure that the best set
of patterns is mined, the practitioner might have to ex-
periment several combinations of values.

In conclusion, C3Ro does not guarantee to be the most
efficient mining algorithm for each individual case, es-
pecially in terms of execution time. However, it is a
generic algorithm, designed to be the only input (al-
gorithm) that a practitioner uses, whatever are his/her
needs, constraints and data characteristics. C3Ro fulfils
a wide range of needs and is extremely practical.

7. Experiments

In this section, we conduct a comprehensive perfor-
mance study to evaluate C3Ro in terms of efficiency,
noise resistance and apprehensibility.

First, we aim at evaluating the efficiency of C3Ro
when mining $CICo_kSP$ (closed k -contiguous sequential
patterns) in terms of execution time, memory usage
and scalability. As C3Ro is able to mine from closed
contiguous sequential patterns ($CICoSP$) to closed
regular sequential patterns ($CISP$), depending on the
value of k (from 0 to ∞), we start by comparing C3Ro
to CCSpan [58], a $CICoSP$ mining algorithm, and
to CM-Clasp [13], one of the most efficient vertical
 $CISP$ mining algorithm, in terms of both execution
time and memory usage. Then, we evaluate the impact
of each additional wildcard both on the execution
time and on the number of mined patterns by C3Ro.
For clarity, in this experiment, we use the notation
 $C3_kRo$ when C3Ro is in the configuration of min-
ing $CICo_kSP$. The evaluation of the efficiency of
 $C3_kRo$ is further detailed by an evaluation of its scal-
ability on several datasets, including very large datasets.

Second, C3Ro is evaluated in terms of noise resis-
tance and apprehensibility of the set of patterns, when
mining $Cl_\lambda Co_k R_\delta C_\zeta$ (λ -closed k -contiguous δ -robust se-
quential patterns satisfying a set ζ of prefix-monotones
constraints). As the employment sector is our main ap-
plication field of interest, we conduct this second exper-
iment on a dataset made up of job offers.

1465 *7.1. Datasets and Environment.*

1466 The experiments are performed on a i7-7700HQ 1496
 1467 2.8GHz with 16GB memory on Windows 10. 1497

1468 In order to evaluate the efficiency of C3Ro (first part 1498
 1469 of the experiments), we use three reference datasets 1499
 1470 (see Table 9), traditionally used to evaluate sequen- 1500
 1471 tial pattern mining algorithms [51, 47, 13, 58, 2]. 1501
 1472 *BMS1 Gazelle*, *Kosarak* and a light version *LKosarak*. 1502
 1473 *BMS1 Gazelle* and *Kosarak* are available online on the 1503
 1474 SPMF website¹. 1504

1475 The *BMS1 Gazelle* dataset contains clickstream and 1505
 1476 purchase data from Gazelle.com, a legwear and legcare 1506
 1477 web retailer, it was used in the KDD-CUP 2000 1507
 1478 competition. The second dataset, *Kosarak*, is a very 1508
 1479 large dataset containing almost 1 million sequences 1509
 1480 of clickstream data from a Hungarian news portal. 1510
 1481 The light version (*LKosarak*) contains about a third 1511
 1482 of the sequences of the original, with almost the same 1512
 1483 characteristics. 1513

1484 The second part of the experiments is conducted on 1514
 1485 *Jobs60K* [2]. This dataset is made up of 60,000 job 1515
 1486 offers, collected from specialized websites between 1516
 1487 January 2016 and June 2016. *Jobs60K* is noisy and 1517
 1488 very sparse, it reflects the kind of textual data that can 1518
 1489 be found on the Web. 1519

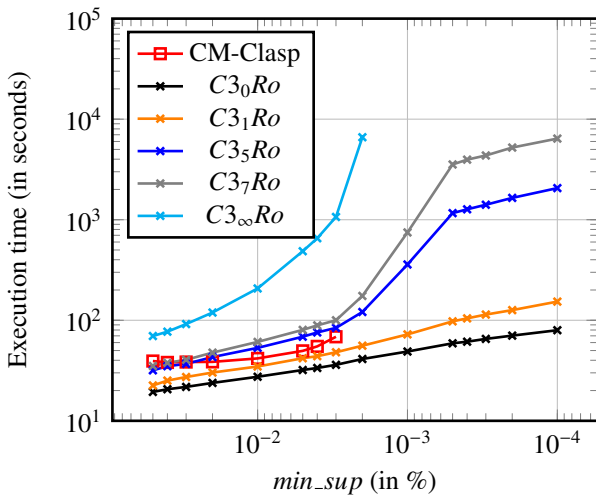


Figure 2: Execution time on Kosarak

1491 *7.2. Efficiency Evaluation*

1492 *7.2.1. C3Ro vs CM-Clasp vs CCSpan*

1493 This section is interested in the execution time (Fig- 1543
 1494 ures 1 and 2) and the memory usage (Table 10) of the 3 1544

¹<http://www.philippe-fournier-viger.com/spmf/index.php>

1495 algorithms on the 3 datasets.

1496 We first focus on the comparison of *C3_0Ro* and 1496
 1497 *CCSpan*, as they both mine *CICoSP*, in terms of ex- 1497
 1498 ecution time, with various relative support thresholds, 1498
 1499 on *BMS1 Gazelle*. The execution time of both *CCSpan* 1499
 1500 and *C3_0Ro* on *BMS1 Gazelle*, according to *min_sup*, 1500
 1501 is shown in Figure 1a. *C3_0Ro* always has an execution 1501
 1502 time below *CCSpan*, up to a maximum of almost 1502
 1503 700 times with the lowest support. In addition, the 1503
 1504 decrease of the support impacts twice less *C3_0Ro* than 1504
 1505 *CCSpan*. In addition, even with *k* ranging from 1 to ∞ , 1505
 1506 *C3_kRo* remains faster than *CCSpan* on *BMS1 Gazelle*. 1506
 1507 The execution time of *CCSpan* on both *Kosarak* and 1507
 1508 *LKosarak* is not shown due to an extremely high 1508
 1509 execution time. 1509

1520 Second, we focus on the mining of *CIS P* by compar- 1520
 1521 ing *C3_kRo* and *CM-Clasp* with a number of wildcards *k* 1521
 1522 ranging from 0 to ∞ (Figures 1 and 2). *CM-Clasp* could 1522
 1523 not be executed on any of the 3 datasets with a support 1523
 1524 lower than 10^{-3} , due to an extreme memory usage. We 1524
 1525 observe that when the number of wildcards is below 1525
 1526 5 and the support is higher than 10^{-3} , *C3_kRo* and 1526
 1527 *CM-Clasp* are very similar in terms of execution time 1527
 1528 on the three datasets. Therefore, we can conclude that, 1528
 1529 when *k* is ranging from 0 to 5, *C3_kRo* competes with 1529
 1530 one of the most efficient *CloSP* mining algorithms, 1530
 1531 *CM-Clasp*, while being able to run with very low 1531
 1532 support values on very large datasets such as *Kosarak*. 1532
 1533 Let us recall that *C3Ro* has been designed to improve 1533
 1534 the efficiency of sequential pattern mining and the 1534
 1535 apprehensibility of its results. Thus, even if *C3Ro* can 1535
 1536 mine *CIS P*, it is rather intended to mine contiguous and 1536
 1537 semi-contiguous sequential patterns, so with a limited 1537
 1538 number of wildcards, in order to achieve its objective 1538
 1539 regarding efficiency and apprehensibility. 1539

1540 Regarding the memory usage (Table 10), let us first 1540
 1541 focus on the comparison of *CCSpan* and *C3_0Ro* that 1541
 1542 mine *CICoSP*. On *BMS1 Gazelle*, they are both consum- 1542
 1543 ing less than 300MB, so are comparable. Let us recall 1543
 1544 that *CCSpan* could not be executed on both 1544
 1545 *LKosarak* and *Kosarak* due to execution time issues. 1545

1546 Let us now focus on the comparison of *CM-Clasp* 1546
 1547 and *C3_infinityRo* that mine *CloSP*. On *BMS1 Gazelle*, 1547
 1548 *CM-Clasp* consumes at least 4,200MB and explodes 1548
 1549 regarding memory usage when is *min_sup* lower than 1549
 1550 $5 \cdot 10^{-3}$. *C3_infinityRo* consumes less than 250MB, whatever 1550
 1551 is *min_sup*, which is similar to *C3_0Ro*. 1551

1552 Concerning *LKosarak* and *Kosarak*, we observe a 1552
 1553 similar behavior. *C3_infinityRo* has a low memory usage, i.e. 1553
 1554 between 500MB and 1,300MB according to *min_sup* 1554

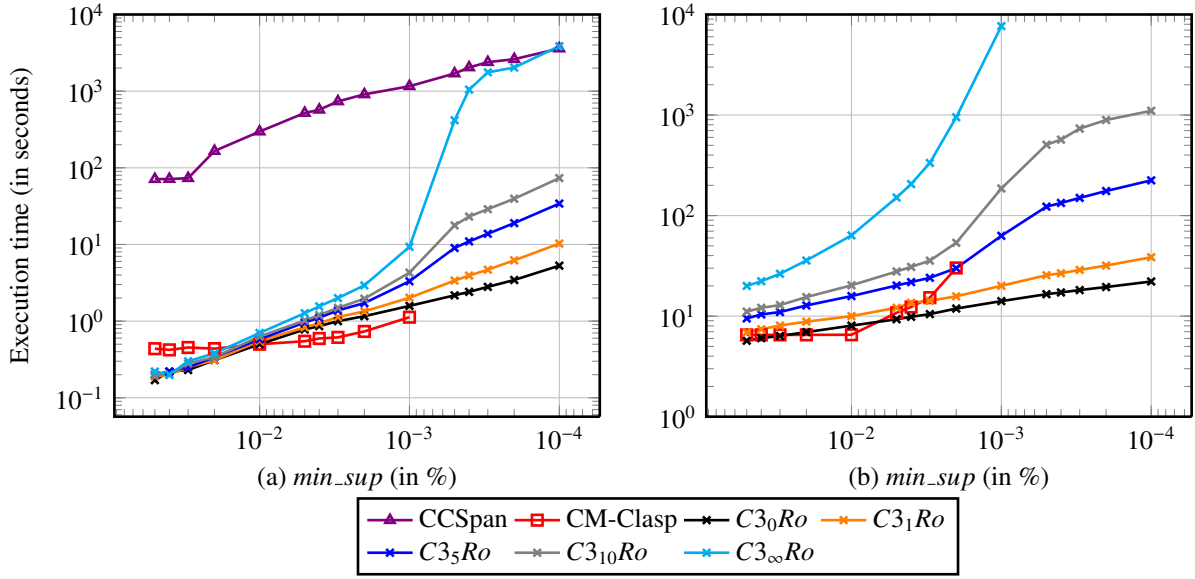


Figure 1: Execution time on Gazelle(a) and LKosarak(b)

1547 and CM-Clasp has a very high memory usage, between 1574
 1548 6,700MB and 10,600MB with a high *min_sup* value, 1575
 1549 and explodes when the support is lower than $2 \cdot 10^{-2}$. 1576
 1550 This high memory usage is a drawback of the vertical 1577
 1551 representation when the number of frequent patterns 1578
 1552 mined increases. This makes CM-Clasp not executable 1579
 1553 on any dataset with a low *min_sup* value. We observe 1580
 1554 that the number of wildcards has no impact on the mem- 1581
 1555 ory usage of $C3_kRo$, probably due to the pattern growth 1582
 1556 representation.

1557 With a limited number of wildcards, $C3_kRo$ has an 1585
 1558 execution time similar to CM-Clasp and becomes slower 1586
 1559 when the number of wildcards is above 5. Let us recall 1587
 1560 that, even if $C3_kRo$ is able to mine *CISP*, it has been 1588
 1561 designed to mine closed contiguous or semi-contiguous 1589
 1562 sequential patterns, therefore the number of wildcards is 1590
 1563 not supposed to be high. In the frame of *CICoSP* min- 1591
 1564 ing, $C3_0Ro$ is up to 700 times faster than CCSpan. At 1592
 1565 last, $C3_kRo$ has a very low memory usage, contrary to 1593
 1566 CM-Clasp that cannot use a low *min_sup* value, due an 1594
 1567 excessive memory usage of the vertical IDLists repre- 1595
 1568 sentation. 1596

1569 We can conclude that among CCSpan, CM-Clasp and 1599
 1570 $C3_kRo$, $C3_kRo$ is the only sequential pattern mining 1600
 1571 algorithm able to mine large datasets in a reasonable time, 1601
 1572 including with low *min_sup* values, due to its low mem- 1602
 1573 ory usage. 1603

7.2.2. Impact of the number of wildcards

Let us now focus on the impact of the number of wildcards on $C3_kRo$ execution time, on the 3 datasets (Figures 1 and 2). We also propose to study the impact of the number of wildcards on the size of the set of mined patterns.

First, we focus on the configuration where the highest support is used. On the *BMS1 Gazelle* dataset, each additional wildcard increases the execution time by 2% on average. On the *LKosarak* and *Kosarak* datasets, this increase is higher, on average 7% and 11% respectively. When focusing on the associated number of mined patterns, it is unexpectedly slightly impacted. This small impact may be due to the support value that may be too high.

Second, we focus on the configuration where the lowest support is used. In terms of execution time, unlike in the previous configuration, each wildcard has a significant impact. Indeed, regardless of the rank of the wildcard, the impact on the execution time is about 30% on *BMS1 Gazelle* and 100% on *Kosarak*. In terms of number of patterns, at the opposite of the previous configuration, each wildcard allows to mine a significant additional number of patterns (on average twice more), whatever is the dataset used.

We can conclude that the impact of the use of one

wildcard, on both execution time and number of patterns, highly depends on its rank, the minimum support value and the dataset characteristics. The largest impact is achieved when the lowest support is used. This impact is explained by the large number of additional patterns provided by each wildcard. This result supports the fact that the number of wildcards is a critical parameter that greatly impacts the execution time and the number of mined patterns.

7.2.3. Scalability

We now propose to evaluate the scalability of $C3_kRo$, through the use of various number of sequences in datasets. We use both *LKosarak* and *Kosarak*, which have similar characteristics, except the number of sequences that is three times smaller in *LKosarak*. With the lowest support, which represents the configuration where the execution time and the number of mined patterns are the highest, $C3_0Ro$ runs in 22 seconds on *LKosarak* and in less than 80 seconds on *Kosarak* (4 times faster on *LKosarak*). It is worth noting that CCSpan (which mines the same patterns than $C3_0Ro$) cannot be executed in a decent amount of time on both datasets, whatever is the minimum support value used. $C3_5Ro$ runs 10 times faster on *LKosarak* (200 seconds) than on *Kosarak* (2,000 seconds). We can say that the use of wildcards only slightly impacts the scalability of $C3Ro$. Let us recall that CM-Clasp, one of the most efficient *CloSP* mining algorithm, cannot be executed on any of the three datasets when the support is under 5.10^{-3} . Nevertheless, $C3_\infty Ro$ can be executed with a very low memory usage with such low supports. These elements highlight the high scalability of $C3_kRo$.

Two more arguments are put forward to illustrate the scalability of $C3_kRo$:

To the best of our knowledge, no sequential pattern mining algorithms in the literature can be executed on both *LKosarak* and *Kosarak*, in a decent amount of time, when the support is lower than 5.10^{-3} . The fact that $C3_0Ro$ and $C3_7Ro$ can be executed in a reasonable time on *Kosarak* with such a low support, is a first indicator of its scalability.

At last, the configuration of $C3Ro$ used was dedicated to the mining of $Cl_1Co_kRo_1C_0SP$, for the sake of comparison with CCSpan and CM-Clasp. If different values of λ and δ are used, the computation time will be even more lower. We can thus conclude that the mining of $Cl_\lambda Co_k Ro_\delta C_\zeta SP$ does not impact the scalability of $C3Ro$ in terms of execution time.

To summarize, this first set of experiments has shown that $C3Ro$ is always significantly faster than CCSpan and similar to CM-Clasp, when mining closed contiguous or semi-contiguous sequential patterns. Moreover, the number of wildcards has an impact on the execution time of $C3Ro$, with an average increase of 33% per wildcard on each of the three datasets. On these three datasets, the use of wildcards only slightly impacts the scalability. Furthermore, this increase in the execution time was expected due to the large number of additional patterns mined when using wildcards (40% for each wildcard) on the three datasets.

This experiment has also shown that $C3Ro$ has a very low memory usage regardless of the support, the dataset or the number of wildcards, while the memory usage is the main flaw of CM-Clasp due to the vertical IDLists representation.

$C3Ro$ has thus proven to be a scalable algorithm with a really small execution time on a very large datasets, such as *Kosarak*, especially in comparison to CCSpan and CM-Clasp, thus to closed sequential pattern mining algorithms.

7.3. Noise resistance of $C3Ro$ and apprehensibility of the set of patterns

In this section, we show how the extended-closedness and the robustness constraints that we have introduces impact the number and the relevance of the patterns mined in noisy data, especially in a contiguous sequential pattern mining frame, so the apprehensibility of the set of patterns.

This experiment is conducted on the *Jobs60K* dataset, which contains 60K job offers with an average length of 77.0 on 92,394 distinct items (see Table 9).

In the employment sector, activities are at the core of job offers. An activity is a coherent set of completed tasks, organized toward a predefined objective with a result that can be measured. An activity is usually formalized by action verbs [1]. In this context, we define activities as patterns starting with a verb and with a length no less than 3. We view these characteristics as constraints. Thus, we set two prefix-monotone constraints: C_1 as a pattern having a first item of the *verb* category (V): $C_1(P) = P[1] \in V$, and C_2 on the length of the pattern that should be no less than 3: $C_2(P) = len(P) > 2$. “Inform clients by explaining procedures”, “Manage business by performing related duties” or “Start operations by entering commands” are examples of activities. Experts agree that the length of an activity is never less than 3 and is on average made up of between 3 and 6 words.

We set a relative support of $15.10^{-4}\%$ in order to mine

Table 11: $Cl_{0.8}Co_1R_{0.5}C_{\{C_1,C_2\}}SP$ in *Jobs60K*

id	configuration	activities
#1	$Cl_1Co_0R_1C_{\{C_1,C_2\}}$	352
#2	$Cl_1Co_1R_1C_{\{C_1,C_2\}}$	698
#3	$Cl_1Co_1R_{0.5}C_{\{C_1,C_2\}}$	429
#4	$Cl_{0.8}Co_1R_1C_{\{C_1,C_2\}}$	601
#5	$Cl_{0.8}Co_1R_{0.5}C_{\{C_1,C_2\}}$	384

a significant amount of patterns and perform a relevant analysis. An experiment conducted in [2] on the same dataset has shown that $k = 1$ wildcard is the adequate number to mine activities. Indeed, when more than one wildcard are allowed, the patterns mined become too long (average length above 11) and are therefore often meaningless.

We will not study the impact of the variation of the values of δ and λ , as it would only give information on the dataset regarding activities. We set the robustness ratio to $\delta = 0.5$ and the extended ratio to $\lambda = 0.8$. These values have been determined experimentally during several runs to reduce the impact of the noise on the mined activities, while preventing at best the mining of irrelevant activities in the final set. For example, $\delta = 0.5$ means that an activity occurring with a wildcard more than 50% of the cases does not represent a real activity.

Let us first focus on the number of mined activities according to the parameters of C3Ro (table 11): with a wildcard ($k = 1$) or without any ($k = 0$), with a robustness ratio ($\delta = 0.5$) and without any ($\delta = 1$), with an extended ratio ($\lambda = 0.8$) or without any ($\lambda = 1$). We refer to the different configurations of C3Ro by using the id indicated in Table 11. We start by evaluating the impact of the use of $k = 1$ wildcard on the number of mined activities by comparing the results of #1 and #2. #1 mines 352 activities, while #2 mines 698 activities. The use of one wildcard allows to find 346 additional activities, which doubles the number of activities. This increase suggests that the dataset could be noisy and that the use of a wildcard is relevant. However, this configuration cannot differentiate activities that are frequently contiguous, thus that were not mined by #1 due the noise, from the activities that are frequently not contiguous, thus artificially created by the wildcard. Thus, this configuration does not give any information about the relevance of the newly mined activities.

To tackle this issue, #3 mines 429 activities, meaning

that 269 activities, among the 346 added by #2 are mined more than half of the cases thanks to the use of the wildcard. We consider these activities as irrelevant. The robustness ratio (#3) filters out 78% of the activities added by #2, while mining 22% more activities than #1. This last number shows that many activities were not mined due to the noise in the dataset and that the 0.5-robustness constraint allows to mine part of these activities in noisy data. These figures confirm that this web job offers dataset is noisy. Let us notice that, without the 0.5-robustness constraint, which comes down to semi-contiguous sequential pattern mining, none of these 269 activities would have been removed.

When using the extended ratio $\lambda = 0.8$, #4 mines 601 activities. The extended ratio $\lambda = 0.8$ decreases the number of mined activities in #2 by 14% (97 activities), which are considered as useless as they are included in another “super-activity” (super-pattern) with a similar support that is part of the resulting set of patterns. The extended-closedness constraint allows to remove more patterns than the traditional closedness constraint, decreasing the size of the resulting set and thus increasing its apprehensibility.

When combining $\delta = 0.5$ and $\lambda = 0.8$, #5 mines 384 activities, which is 11% lower than the set mined by #3, without any loss of useful information. Indeed, the use of the 0.5-robustness only removes semi-contiguous sequential patterns that are not relevant in activity mining and the 0.8-closedness removes activities included in other “super-activity” with a similar support (20% lower at maximum). Therefore, the combination of both new constraints allows to improve even more the apprehensibility of the resulting set.

The comparison between the set of mined activities by #5 and the one mined by #1 shows that 280 activities are common to both sets. We now focus on the evaluation of the relevancy of the activities specific to each set. The 72 remaining activities of the set mined by #1 were assessed by an expert of the field, who concluded that more than 50% of these 72 activities are either irrelevant or redundant. At the opposite, the analysis, by the same expert, of the 104 activities specific to #5, highlighted that only 10% of these activities are either irrelevant or redundant. This analysis confirms that 0.8-closed 1-contiguous 0.5-robust sequential patterns allow to reduce the final set of patterns by removing useless patterns, while increasing the relevance of the patterns mined. In a nutshell, #5 allows to mine around 20% more relevant activities than #1 in noisy data, while only increasing by 9% the resulting set size and thus improve the apprehensibility in the mining of activities.

1797
1798 To conclude, this experiment shows that the use of
1799 the two newly introduced constraints (robustness and
1800 extended-closed) is a lever to improve the apprehensibility
1801 of the results, with no impact on the execution
1802 time. In addition, the use of these new constraints in
1803 C3Ro is flexible through the use of a robustness and
1804 extended ratios allowing C3Ro to be adjustable to the
1805 final user needs.
1806 C3Ro is thus an answer to the first scientific question:
1807 How to efficiently mine an apprehensible set of sequen-
1808 tial patterns, including the frame of noisy data?

1809 7.4. Expert feedback

1810 The mining of activities in job offers is a key com-
1811 ponent of the job market’s analysis. Indeed, the mined
1812 activities provide a relevant picture of the needs of com-
1813 panies, which is essential in many areas such as consult-
1814 ing, recruitment and training. In consulting, it provides
1815 relevant factual elements to support companies in ex-
1816 pressing their own needs in terms of recruitment.

1817 Table 12 illustrates how the use of C3Ro significantly
1818 improves the set of mined activities in *Jobs60K*. The
1819 activity “managing sales teams” is no longer mined with
1820 C3Ro as it is included in the activity “managing sales
1821 teams effectively”, which is 12% more frequent. In that
1822 pattern, the word “effectively” is as important as the rest
1823 of the activity. This fact has been validated by a HR ex-
1824 pert. In addition, C3Ro discovers a new activity “man-
1825 aging sales teams distribution channels”, which was not
1826 frequent with traditional algorithms, and is now discov-
1827 ered thanks to the use of the two new constraints. This
1828 activity has also been validated by the HR expert. More
1829 generally, the set of mined activities is more apprehen-
1830 sible. Not only its size is decreased but the information
1831 it contains is more valuable.

1832 8. Conclusion

1833 While sequential pattern mining has been widely
1834 studied for many years, it still faces several challenges.
1835 First, many studies have focused on increasing the effi-
1836 ciency of the mining process and on decreasing the
1837 size of the set of patterns, but mining large databases re-
1838 mains a challenge. Second, although noise is a common
1839 feature in real-world databases, it has not faced much
1840 interest in the pattern mining literature. Third, the re-
1841 duction of the size of the set of patterns is often at the
1842 expense of an uncontrolled loss of information, which
1843 may make the set of patterns useless, especially when it
1844 is intended to practitioners.

1845 This paper aimed at proposing a sequential pattern min-
1846 ing algorithm that addresses these three challenges. It is
1847 designed to mine large and noisy databases, while dis-
1848 covering an apprehensible set of patterns, i.e. a set of
1849 patterns of reduced size, with a controlled information
1850 loss.

1851 To reach this goal, we introduced two new con-
1852 straints, namely the *robustness constraint* that allows
1853 the mining of contiguous sequential patterns in noisy
1854 data and the *extended-closedness constraint*, which
1855 represents a trade-off between the full information ex-
1856 tracted with the closedness constraint and the reduced
1857 set mined with the maximal constraint. In addition,
1858 we proposed a generic algorithm C3Ro, based on
1859 the pattern-growth philosophy that manages several
1860 parameters. These parameters have been chosen to
1861 mine a large range of patterns: going from closed
1862 contiguous sequential patterns to maximal regular
1863 sequential patterns. C3Ro is thus thought to be the
1864 single pattern mining algorithm that can fit the final
1865 users’ specific needs, whatever they are, thanks to these
1866 parameters.

1867 The experiments we conducted showed that C3Ro is
1868 scalable with a very low memory usage compared to
1869 well-known algorithms such as CCSpan and CM-Clasp.
1870 An expert has validated that the use of the two new
1871 constraints allows to mine more relevant patterns
1872 compared to the use of the standard contiguity or
1873 closedness constraints in noisy data.

1874 Based on this work, we intend to work on four future
1875 directions. First, although C3Ro is designed for practi-
1876 tioners’ use, the required parameters need to be fixed by
1877 these practitioners, especially those associated to both
1878 constraints (δ and λ). Their optimal value can only be
1879 found by iteratively running C3Ro. We would like to
1880 work on the automatic determination of these values, at
1881 least of a limited range of possible values. It will rely
1882 on an analysis of the type and amount on noise in data.
1883 Limiting this possible range of values will be of high
1884 utility for practitioners.

1885 Second, still with the objective of satisfying practition-
1886 ers, we plan to focus on new quality criteria. Indeed,
1887 support is the traditionally used as the evaluation cri-
1888 terion, but unexpectedness, periodicity, coherence, etc.
1889 also deserve to be studied for mining patterns that may
1890 fit better practitioners’ expectations.

1891 Third, we intend to focus on noise, especially on study-
1892 ing the precise position of noise within patterns (in what
1893 places does noise occur). Not only the identification of
1894 this noise will help quantify and qualify it, but its man-
1895 agement will contribute to represent patterns more pre-
1896

- cisely and thus increase even more the impact of both new constraints on apprehensibility.
- Last, we will use the output of the third goal as a first base to study pattern drift. We view the management of the position of noise as a possibility to differentiate between noise and new forms of patterns (loss, substitution or addition of items in existing patterns). This last goal is of high utility for many real-world applications, as it is common that data evolve and this evolution has to be managed.
- [1] Abboud, Y., Boyer, A., and Brun, A. (2015). Predict the emergence: Application to competencies in job offers. In *Tools with Artificial Intelligence (ICTAI), 2015 IEEE 27th International Conference on*, pages 612–619. IEEE.
 - [2] Abboud, Y., Boyer, A., and Brun, A. (2017). CCPM: A Scalable and Noise-Resistant Closed Contiguous Sequential Patterns Mining Algorithm. In *13th International Conference on Machine Learning and Data Mining MLDM 2017*, volume 89, page 15. New York, United States.
 - [3] Aggarwal, C. C., Li, Y., Wang, J., and Wang, J. (2009). Frequent pattern mining with uncertain data. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 29–38. ACM.
 - [4] Aggarwal, C. C., Ta, N., Wang, J., Feng, J., and Zaki, M. (2007). Xproj: a framework for projected structural clustering of xml documents. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 46–55. ACM.
 - [5] Agrawal, R., Imieliński, T., and Swami, A. (1993). Mining association rules between sets of items in large databases. In *Acm sigmod record*, volume 22, pages 207–216. ACM.
 - [6] Agrawal, R. and Srikant, R. (1995). Mining sequential patterns. In *Data Engineering, 1995. Proceedings of the Eleventh International Conference on*, pages 3–14. IEEE.
 - [7] Ayres, J., Flannick, J., Gehrke, J., and Yiu, T. (2002). Sequential pattern mining using a bitmap representation. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '02*, pages 429–435. New York, NY, USA. ACM.
 - [8] Béchet, N., Cellier, P., Charnois, T., and Crémilleux, B. (2015). Sequence mining under multiple constraints. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, pages 908–914. ACM.
 - [9] Chen, J. (2008). Contiguous item sequential pattern mining using updown tree. *Intelligent Data Analysis*, 12(1):25–49.
 - [10] Chen, J. and Cook, T. (2007). Mining contiguous sequential patterns from web logs. In *Proceedings of the 16th International Conference on World Wide Web, WWW '07*, pages 1177–1178. New York, NY, USA. ACM.
 - [11] Djenouri, Y., Belhadi, A., and Fournier-Viger, P. (2017). Extracting useful knowledge from event logs: A frequent itemset mining approach. *Knowledge-Based Systems*.
 - [12] Fischer, J., Heun, V., and Kramer, S. (2006). Optimal string mining under frequency constraints. In *PKDD*, volume 4213, pages 139–150. Springer.
 - [13] Fournier-Viger, P., Gomariz, A., Campos, M., and Thomas, R. (2014a). Fast vertical mining of sequential patterns using co-occurrence information. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 40–52. Springer.
 - [14] Fournier-Viger, P., Gomariz, A., Šebek, M., and Hlosta, M. (2014b). Vgen: fast vertical mining of sequential generator patterns. In *International Conference on Data Warehousing and Knowledge Discovery*, pages 476–488. Springer.
 - [15] Fournier-Viger, P., Nkambou, R., and Nguifo, E. M. (2008). A knowledge discovery framework for learning task models from user interactions in intelligent tutoring systems. In *Mexican International Conference on Artificial Intelligence*, pages 765–778. Springer.
 - [16] Fournier-Viger, P., Wu, C.-W., Gomariz, A., and Tseng, V. S. (2014c). Vmsp: Efficient vertical mining of maximal sequential patterns. In *Canadian Conference on Artificial Intelligence*, pages 83–94. Springer.
 - [17] Fumarola, F., Lanotte, P. F., Ceci, M., and Malerba, D. (2016). Clofast: closed sequential pattern mining using sparse and vertical id-lists. *Knowledge and Information Systems*, 48(2):429–463.
 - [18] Fürnkranz, J. (1998). A study using n-gram features for text categorization. *Austrian Research Institute for Artificial Intelligence*, 3(1998):1–10.
 - [19] Gao, C., Wang, J., He, Y., and Zhou, L. (2008). Efficient mining of frequent sequence generators. In *Proceedings of the 17th international conference on World Wide Web*, pages 1051–1052. ACM.
 - [20] García, S., Luengo, J., and Herrera, F. (2015). *Data preprocessing in data mining*. Springer.
 - [21] García-Hernández, R. A., Martínez-Trinidad, J. F., and Carrasco-Ochoa, J. A. (2006). A new algorithm for fast discovery of maximal sequential patterns in a document collection. In *CICLing*, pages 514–523. Springer.
 - [22] Garofalakis, M. N., Rastogi, R., and Shim, K. (1999). SPIRIT: Sequential pattern mining with regular expression constraints. In *Proceedings of the 25th International Conference on Very Large Data Bases, VLDB '99*, pages 223–234. San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.
 - [23] Gomariz, A., Campos, M., Marin, R., and Goethals, B. (2013). Clasp: an efficient algorithm for mining frequent closed sequences. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 50–61. Springer.
 - [24] Grossi, R., Iliopoulos, C. S., Liu, C., Pisanti, N., Pissis, S. P., Retha, A., Rosone, G., Vayani, F., and Versari, L. (2017). On-line pattern matching on similar texts. In *LIPICs-Leibniz International Proceedings in Informatics*, volume 78. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.
 - [25] Hahsler, M. and Karpienko, R. (2017). Visualizing association rules in hierarchical groups. *Journal of Business Economics*, 87(3):317–335.
 - [26] Han, J., Pei, J., Mortazavi-Asl, B., Chen, Q., Dayal, U., and Hsu, M.-C. (2000). Freespan: frequent pattern-projected sequential pattern mining. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 355–359. ACM.
 - [27] Han, J., Pei, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U., and Hsu, M. (2001). Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In *Proceedings of the 17th international Conference on Data Engineering*, pages 215–224.
 - [28] Hirate, Y. and Yamana, H. (2006). Generalized sequential pattern mining with item intervals. *JCP*, 1(3):51–60.
 - [29] Ho, J., Lukov, L., and Chawla, S. (2005). Sequential pattern mining with constraints on large protein databases. In *Proceedings of the 12th International Conference on Management of Data (COMAD)*, pages 89–100.
 - [30] Kang, T. H., Yoo, J. S., and Kim, H. Y. (2007). Mining frequent contiguous sequence patterns in biological sequences. In *2007 IEEE 7th International Symposium on Bioinformatics and BioEngineering*, pages 723–728.
 - [31] Karim, M., Rashid, M., Jeong, B.-S., Choi, H.-J., et al. (2012). An efficient approach to mining maximal contiguous frequent pat-

- terns from large dna sequence databases. *Genomics & informatics*, 10(1):51–57.
- [32] Kemmar, A., Loudni, S., Lebbah, Y., Boizumault, P., and Charnois, T. (2015). Prefix-projection global constraint for sequential pattern mining. In *International Conference on Principles and Practice of Constraint Programming*, pages 226–243. Springer.
- [33] Le, B., Duong, H., Truong, T., and Fournier-Viger, P. (2017). Fclosm, fgensm: two efficient algorithms for mining frequent closed and generator sequences using the local pruning strategy. *Knowledge and Information Systems*, pages 1–37.
- [34] Li, C. and Wang, J. (2008). *Efficiently mining closed subsequences with gap constraints*, pages 313–322.
- [35] Li, H., Zhang, N., Zhu, J., Wang, Y., and Cao, H. (2018). Probabilistic frequent itemset mining over uncertain data streams. *Expert Systems with Applications*.
- [36] Liao, V. C.-C. and Chen, M.-S. (2014). DFSP: a depth-first spelling algorithm for sequential pattern mining of biological sequences. *Knowledge and information systems*, 38(3):623.
- [37] Liu, H., Wang, L., Liu, Z., Zhao, P., and Wu, X. (2018). Efficient pattern matching with periodical wildcards in uncertain sequences. *Intelligent Data Analysis*, 22(4):829–842.
- [38] Liu, J., Paulsen, S., Wang, W., Nobel, A., and Prins, J. (2005). Mining approximate frequent itemsets from noisy data. In *Data Mining, Fifth IEEE International Conference on*, pages 4–pp. IEEE.
- [39] Luo, C. and Chung, S. M. (2005). Efficient mining of maximal sequential patterns using multiple samples. In *Proceedings of the 2005 SIAM International Conference on Data Mining*, pages 415–426. SIAM.
- [40] Mannila, H., Toivonen, H., and Inkeri Verkamo, A. (1997). Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, 1(3):259–289.
- [41] Matsui, T., Uno, T., Umemori, J., and Koide, T. (2013). *A New Approach to String Pattern Mining with Approximate Match*, pages 110–125. Springer Berlin Heidelberg, Berlin, Heidelberg.
- [42] Pasquier, N., Bastide, Y., Taouil, R., and Lakhal, L. (1999). Discovering frequent closed itemsets for association rules. In *Proceedings of the 7th International Conference on Database Theory, ICDT '99*, pages 398–416, London, UK, UK. Springer-Verlag.
- [43] Pei, J., Han, J., Mao, R., et al. (2000). CLOSET: An efficient algorithm for mining frequent closed itemsets. In *ACM SIGMOD workshop on research issues in data mining and knowledge discovery*, volume 4, pages 21–30.
- [44] Pei, J., Han, J., and Wang, W. (2007). Constraint-based sequential pattern mining: the pattern-growth methods. *Journal of Intelligent Information Systems*, 28(2):133–160.
- [45] Rodríguez-González, A. Y., Lezama, F., Iglesias-Alvarez, C. A., Martínez-Trinidad, J. F., Carrasco-Ochoa, J. A., and de Cote, E. M. (2017). Closed frequent similar pattern mining: Reducing the number of frequent similar patterns without information loss. *Expert Systems with Applications*.
- [46] Srikant, R. and Agrawal, R. (1996). Mining sequential patterns: Generalizations and performance improvements. *Advances in Database Technology EDBT'96*, pages 1–17.
- [47] Wang, J. and Han, J. (2004). BIDE: Efficient mining of frequent closed sequences. In *Proceedings of the 20th International Conference on Data Engineering, ICDE '04*, pages 79–, Washington, DC, USA. IEEE Computer Society.
- [48] Wang, J., Han, J., and Pei, J. (2003). CLOSET+: Searching for the best strategies for mining frequent closed itemsets. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '03*, pages 236–245, New York, NY, USA. ACM.
- [49] Wu, X., Zhu, X., He, Y., and Arslan, A. N. (2013). Pmbc: Pattern mining from biological sequences with wildcard constraints. *Computers in biology and medicine*, 43(5):481–492.
- [50] Xie, F., Wu, X., and Zhu, X. (2017). Efficient sequential pattern mining with wildcards for keyphrase extraction. *Knowledge-Based Systems*, 115(Supplement C):27 – 39.
- [51] Yan, X., Han, J., and Afshar, R. (2003). Clospan: Mining: Closed sequential patterns in large datasets. In *Proceedings of the 2003 SIAM international conference on data mining*, pages 166–177. SIAM.
- [52] Yang, J., Wang, W., Yu, P. S., and Han, J. (2002). Mining long sequential patterns in a noisy environment. In *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, pages 406–417. ACM.
- [53] Yi, S., Zhao, T., Zhang, Y., Ma, S., and Che, Z. (2011). An effective algorithm for mining sequential generators. *Procedia Engineering*, 15:3653–3657.
- [54] Yun, U. and Ryu, K. H. (2011). Approximate weighted frequent pattern mining with/without noisy environments. *Knowledge-Based Systems*, 24(1):73–82.
- [55] Zaki, M. J. (2000). Sequence mining in categorical domains: Incorporating constraints. In *Proceedings of the Ninth International Conference on Information and Knowledge Management, CIKM '00*, pages 422–429, New York, NY, USA. ACM.
- [56] Zaki, M. J. (2001). SPADE: An efficient algorithm for mining frequent sequences. *Machine learning*, 42(1):31–60.
- [57] Zaki, M. J. and Hsiao, C.-J. (2002). CHARM: An efficient algorithm for closed itemset mining. In *Proceedings of the 2002 SIAM international conference on data mining*, pages 457–473. SIAM.
- [58] Zhang, J., Wang, Y., and Yang, D. (2015). CCSpan: Mining closed contiguous sequential patterns. *Knowledge-Based Systems*, 89:1–13.
- [59] Zhang, M., Kao, B., Cheung, D. W., and Yip, K. Y. (2007). Mining periodic patterns with gap requirement from sequences. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(2):7.

Table 4: Abbreviations

abbreviations	type of patterns
SP	Sequential Pattern
$CISP$	Closed Sequential Pattern
$CoSP$	Contiguous Sequential Pattern
$CICoSP$	Closed Contiguous Sequential Pattern
$CICo_kR_\delta SP$	Closed k-Contiguous δ-Robust Sequential Pattern
$Cl_\lambda Co_k SP$	λ - Closed k-Contiguous Sequential Pattern
$Cl_\lambda Co_k R_\delta C_\zeta$	$Cl_\lambda Co_k R_\delta$ with a set of Constraints ζ

Table 5: Type of sequential patterns mined in the *Noise* database

type of patterns	patterns
$CoSP$	$\langle \text{dear} \rangle : 3, \langle \text{dear, Watson} \rangle : 2, \langle \text{Elementary} \rangle : 5, \langle \text{Elementary, my} \rangle : 5, \langle \text{Elementary, my, dear} \rangle : 3, \langle \text{Elementary, my, dear, Watson} \rangle : 2, \langle \text{my} \rangle : 5, \langle \text{my, dear} \rangle : 3, \langle \text{my, dear, Watson} \rangle : 2$
$CICoSP$	$\langle \text{Elementary, my} \rangle : 5, \langle \text{Elementary, my, dear} \rangle : 3, \langle \text{Elementary, my, dear, Watson} \rangle : 2, \langle \text{Watson} \rangle : 5$
$ClCo_1 SP$	$\langle \text{Elementary, my} \rangle : 5, \langle \text{Elementary, my, dear, Watson} \rangle : 3, \langle \text{Elementary, my, Watson} \rangle : 4, \langle \text{Watson} \rangle : 5$
$ClCo_1 R_{0.6} SP$	$\langle \text{Elementary, my} \rangle : 5, \langle \text{Elementary, my, dear, Watson} \rangle : 5, \langle \text{Watson} \rangle : 5$
$Cl_{0.6} Co_1 R_{0.6} SP$	$\langle \text{Elementary, my, dear, Watson} \rangle : 3$

Table 8: Types of frequent sequential patterns in *SDB*

type of patterns	patterns
Co	$\{a\} : 3, \{a\}\{b\} : 2, \{a\}\{b\}\{c\} : 2, \{b\} : 3, \{b\}\{c\} : 3, \{b\}\{c\}\{d\} : 2, \{c\} : 3, \{c\}\{d\} : 2, \{d\} : 2$
$Co_1 R_{0.6}$	$\{a\} : 3, \{a\}\{b\} : 3, \{a\}\{b\}\{c\} : 3, \{a\}\{b\}\{c\}\{d\} : 3, \{b\} : 3, \{b\}\{c\} : 3, \{b\}\{c\}\{d\} : 2, \{c\} : 3, \{c\}\{d\} : 2, \{d\} : 2$
$CICo$	$\{a\}\{b\}\{c\} : 2, \{b\}\{c\} : 3, \{b\}\{c\}\{d\} : 2$
$CICo_1$	$\{a\}\{b\}\{c\} : 3, \{a\}\{b\}\{c\}\{d\} : 2, \{b\}\{b\} : 2$
$CICo_1 R_{0.6}$	$\{a\}\{b\}\{c\} : 3, \{a\}\{b\}\{c\}\{d\} : 2$
$Cl_{0.65} Co_1 R_{0.6}$	$\{a\}\{b\}\{c\}\{d\} : 2$

Table 9: Datasets characteristics

dataset	#seq.	#items	avg.len.	max.len.	std.dev.len.
<i>BMS1 Gazelle</i>	59,601	499	6.0	535	9.7
<i>LKosarak</i>	305,281	32,138	8.1	2,498	23.6
<i>Kosarak</i>	990,002	41,270	8.1	2,498	23.7
<i>Jobs60K</i>	60,000	92,394	77.0	1,667	56.8

Table 10: Memory usage on the three datasets w.r.t min_sup (MB)

dataset	Minsupp(%)	FI	CCSpan	CM-Clasp	$C3_0Ro$	$C3_\infty Ro$
<i>BMS1 Gazelle</i>	0.05	4	250	4,200	<50	<50
	0.005	150	250	4,500	200	200
	0.0005	374	300	-	200	250
<i>LKosarak</i>	0.05	10	-	5,000	500	500
	0.005	158	-	6,700	500	500
	0.0005	2303	-	-	550	550
<i>Kosarak</i>	0.05	10	-	10,600	1,200	1,200
	0.005	156	-	10,600	1,200	1,200
	0.0005	2297	-	-	1,300	1,300

Table 12: mined activities with or without C3Ro in *Jobs60K*

activity id	configuration	activities	absolute support
#1	without C3Ro	managing sales teams	1712
#2	without C3Ro	managing sales teams effectively	1352
#2	with C3Ro	managing sales teams effectively	1536
#3	with C3Ro	managing sales teams distribution channels	384