Mining Common Morphological Fragments from Process Event Logs

Asef Pourmaoumi Hasankiyadeh[a], Mohsen Kahani[a], Ebrahim Bagheri[b], Mohsen Asadi[c]

[a] Web Technology Lab, Ferdowsi University of Mashhad, Iran

[b] Department of Electrical and Computer Engineering, Ryerson University, Canada

[c] School of Interactive Arts and Technology, Simon Fraser University Surrey, Canada

asef.pourmasoumi@stu-um.ac.ir, kahani@um.ac.ir, bagheri@ee.ryerson.ca, masadi@sfu.ca

Abstract

Many organizations have implemented their organizational processes within integrated information systems using formal process models. These processes, which have been implemented in different organizations can share significant amount of similarities. Analysis and mining of these processes for identifying similarities can lead to valuable insight for the organizations. There has already been work on mining process models from event logs for an individual organization. The objective of this paper is, however, to detect and extract common process fragments from a family of processes that may not have been executed within the same application/organization. These identified common fragments can be used as building blocks of future applications or be used for refactoring existing applications. To this end, we first provide a precise definition of process fragments. We define morphological fragments as operationally identical fragments. We then propose an algorithm for extracting morphological fragments from process event logs. We discuss the relative performance of our proposed algorithm and its applicability in practice.

Keywords: process model fragmentation, event logs, common fragments

1 Introduction

Many organizations have replaced their traditional processes, which would be executed and monitored manually, with the so-called Process Aware Information Systems (PAIS) [1]. Business Process Management (BPM) technology lies at the core of PAIS and promotes effectiveness and efficiency of business processes [6]. In the relative short evolution lifetime of BPM technology, the number of already deployed process models within organizations has grown exponentially [22].

With the growth in the formal deployment of business process models, new challenges are faced. One of the challenges that is of interest to our work in this paper is the co-existence of multiple variants of the same business process fragment in different peer organizations [14]. For example, in municipalities, many of the processes are driven by legislation, e.g., determining the property sales tax rate. Therefore, the processes that are executed in the different departments of these organizations are extensively regulated and therefore quite similar. However, while legislation is establishing the important fundamentals, some degree of freedom is given to the execution units regarding the concrete implementation of such processes. So, different departments are following very similar processes that have been slightly adapted based on their local needs and preferences, e.g. depending on demographics and political choices, the size of the organization [11].

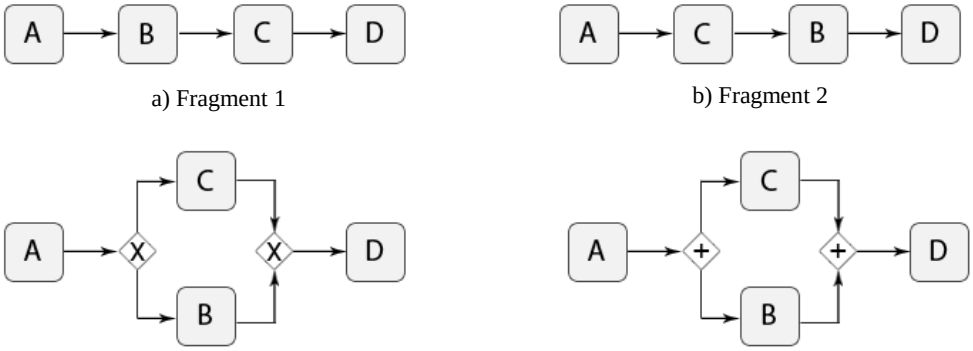


Fig. 1: Four operationally identical fragments with different structures.

Given one could find a significant amount of similarity between a large number of process fragments within similar organizations, consolidating families of process variants can assist organizations in improving their operations and to design more efficient new process models.

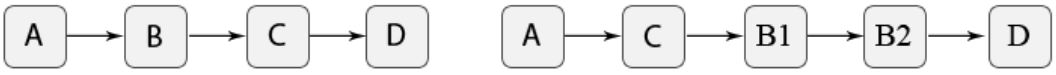
Process similarity and process merging are two concepts, which are commonly used for detecting commonality and variability between processes and for effectively configuring them [20][17]. In process similarity, the distance between two process models is calculated [7]. In process merging, a consolidated solution is offered by combining two or more process models [11]. In process configuration, a consolidated model representing the family of process variants is specialized for a given purpose [5].

Many existing work in the literature identify process fragments and variabilities through comparison of structural similarities of business process models [12] [13]. In our work, we will focus on the harder problem of identifying process fragments by only analyzing process event logs. This is mainly driven by the fact that the process models that are initially designed are often different from the ones that are actually being executed due to the gradual changes that are made on the models under execution [18].

In this paper, we propose a new process fragmentation algorithm, which take as input a collection of event logs of process variants' execution and extracts common reusable fragments from them. Our method can have many applications in process configuration and process optimization. For instance, by employing our method, configuration can be done at the level of fragments instead of activities; therefore, significantly reducing the complexity of the consolidated model. The higher the number of detected fragments is, the higher complexity reduction would be. The identification of the common process fragments from a family of process variants can enable the detection of the best fragments based on the users' needs. To the best of our knowledge, although there are various techniques for process fragmentation, there are no techniques that fragment a family of process models based on a collection of event logs. In this paper, we propose a new approach for extracting common fragments from a set of process variants based on a collection of event logs.

As support for our method, we provide the following contributions:

- First, we provide a precise definition for process fragments. This detailed definition of process fragments will be the basis for our work.
- Second, our fragmentation method considers a collection of processes together as a whole instead of only one process model at a time. We contend that in this case, the proposed fragmentation method collectively considers the characteristics of the whole collection of process models. Considering each process separately may overlook similarities [8].



a) Fragments 1 b) Fragments 2

Fig. 2: Two operationally identical fragments with varying degrees of granularity.

— Third, the proposed approach directly captures process fragments based on a collection of event logs. This leads to a lower error rate compared to the techniques, which rely on the structure of process models.

The rest of the paper is organized as follows. Section 2 establishes the basic terminology and definitions that are adopted in this work. In Section 3, we discuss related work on process fragmentation. In Section 4, the proposed approach is described. In Section 5, we apply the proposed approach on a real-life event log collection and Section 6 concludes the paper and suggests directions for future work.

2 Definitions

Process fragmentation (also known as *process decomposition* or *process modularization* [15]) is the basis for techniques supporting management, reusability and analysis of process models [13]. Although there are existing works on the fragmentation problem, there is yet to be a precise definition for *process fragments* [15]. In [13], a *process fragment* is defined as "any arbitrary subset of the process elements comprised within a process model". In other words, a process fragment can be thought of as any selection of the process elements, even empty set. This definition is very generic. In [15], some guidelines are provided on how to select parts of a process model as fragments. According to these guidelines, good candidates for process fragments are components with a single input and single output control flow arc (also referred as single entry and single exit, SESE). Also, depending on the intended usage of process fragments, some structural constraints may be added to the definition [13]. It should be noted that all publications in the fragmentation domain, assume that there is only one process model to be considered as input [13]. In this paper, we propose a new fragmentation approach, which takes any number of processes as input and extracts common fragments between them.

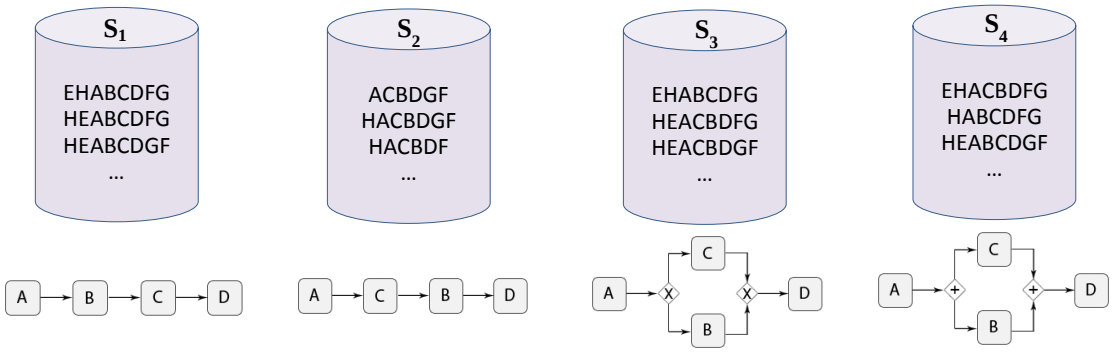
We start by providing a definition for process fragments.

Definition 1: A *process fragment F* is a directed graph described as $F(A, G, R, n_s, n_e)$ where A is the set of activities, G is gateways and $R \sqsubseteq (A \times G) \cup (G \times A) \cup (G \times G)$ is the set of control flow relations. A process fragment F is a subset of the process model elements, which have single input n_s and single output n_e nodes. These input/output nodes can be an activity or an operator or start/end nodes.

Based on this definition, we can fragmentize a process model, but it would not be satisfactory to be used for identifying process similarity or multi-process analysis in our case. In multi-process analysis (e.g., a family of process variants) finding common fragments between process variants can be very valuable. Using this fragment definition, two fragments f_1 and f_2 within two process variants p_1 and p_2 are shared, if and only if they are identical. For example in Figure 1, fragment 1 and 2 are not identical, so they cannot be captured as common fragments between two process models, while they perform operationally equivalent tasks. In the following we further define morphological fragments to address this issue:

Definition 2: Two fragments $F_1(A_1, G_1, R_1, n_{s1}, n_{e1})$ and $F_2(A_2, G_2, R_2, n_{s2}, n_{e2})$ are *morphologically* identical if they perform a similar task but in different ways. These fragments must have the same start and end nodes ($n_{s1} = n_{s2}\ and\ n_{e1} = n_{e2}$). Intermediate nodes must also be identical ($A_1 = A_2$), but they can have different order or different relations.

In Definition 2, the relations between any intermediate activities ($R_1\ and\ R_2$) within a fragment are ignored. The reason is that two fragments that have identical enter/exit nodes and identical intermediate activities, and are perform-



a) Organization s_1 b) Organization s_2 c) Organization s_3 d) Organization s_4

Fig. 3: Event logs from four different organizations.

ing a similar task, would be considered to be very similar fragments that only have structural variances, hence the term *morphological fragment*.

For instance in Figure 1, four operationally identical fragments are shown. All four fragments try to determine the maximum allowable amount of loan that can be given to a person; however, each achieves this in a different way. Activity `A` sends a request to the bank for getting a loan. All fragments start with this activity. Activity `B` is the computation of the applicant's loan capacity and activity `C` is a check on the applicant's credit history. Finally, activity `D` is about making a decision. As seen in Figure 1, activities `B` and `C` can be placed in different relations to each other. This difference is only probably due to various branch managers' choices.

Furthermore, we assume in Definition 2 that two fragments that have the same start and end nodes and have the same or similar activities (regardless of the logical relationship between them), are operationally similar. We consider two activities similar if they perform the same set of operations. In some process variants, one activity may be divided into two or more activities. For example, in Figure 2 activity `B1` sends a *credit history check request*, activity `B2` is *processing and responding to the check credit history request*. In fact, activity `B` that is described in Figure 1 is divided into two activities `B1` and `B2` in this example, which need to be identified as being the same as activity `B`. It should be noted that similar to existing work in the literature, we assume that event labels used in different event logs for the same family of processes are the same for similar events. In other words, we will not deal with scenarios such as the case where `B1` and `B2` collectively represent `B`.

3 Proposed Approach

In this section, we present a new approach for mining common morphological fragments in a process variant family from a collection of event logs. As will be covered in the related works section, all existing fragmentation algorithms are based on the structure of the process models and not based on event logs. In this paper, we are interested in identifying fragments directly form a collection of event logs. Our goal is twofold. First, we are interested in reducing the detection error rate, i.e., the number of misidentified fragments. It has already been discussed that algorithms dealing with event logs are prone to such errors [2] [3]. Second, we would like to improve the fragmentation identification speed of the algorithm. Given intermediate node structures is not important in the morphological fragment definition, if fragments could be directly extracted from unstructured event logs, there is no need to consider activity relationships and therefore it can significantly speed up the process.

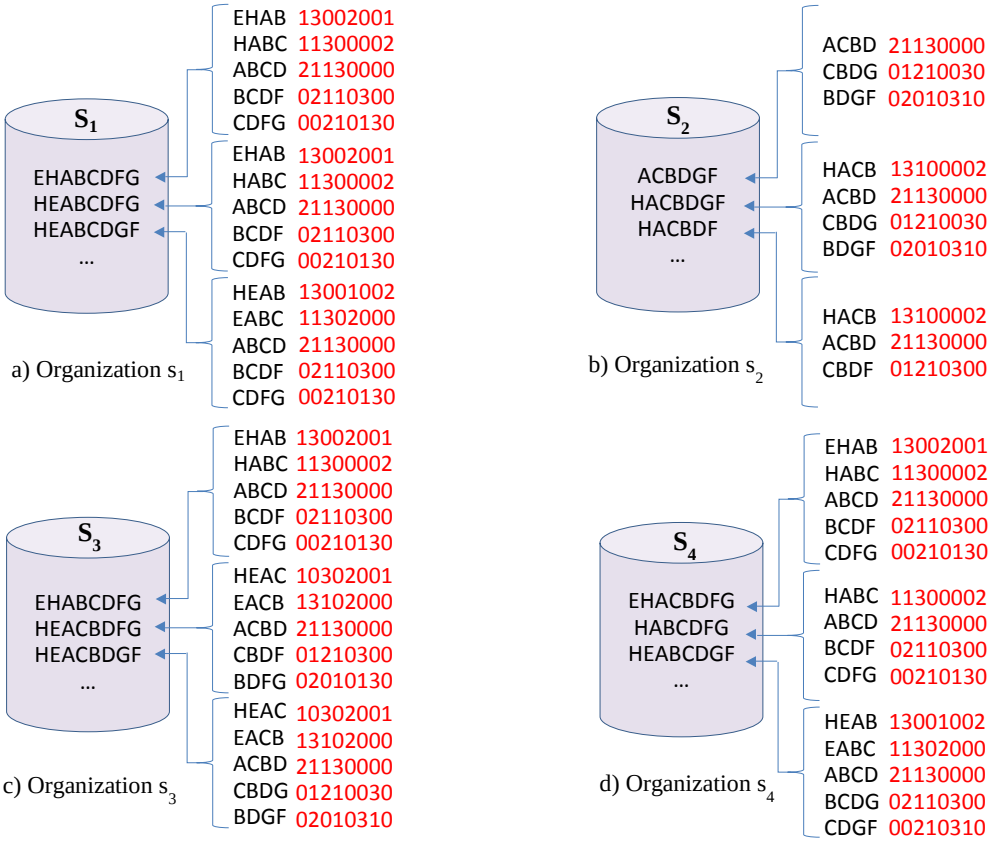


Fig. 4: Generating 4-grams for traces existing in organization event logs set. Red numbers are numerical value which have been mapped to each 4-gram.

We divide the proposed approach into two main phases: *i*) generating *L*-grams (successive L-grams) from each set of event logs and *ii*) finding common *L* -grams between organizations event logs. In the next section, a detailed description of these phases is given.

3.1 Generating *L*-grams from each set of event logs

For the sake of simplicity, we will first assume that we intend to identify fragments with length *L*. To this end, we generate all *L*-grams (successive *L*-grams) from each set of event logs. For better understanding, we explain the method with an example.

Imagine there are four organizations each of which executes a loan application process. Each organization has its event log dataset. This is shown in Figure 3. For instance, we assume that organizations have operationally identical fragments that check the amount of possible loan that can be given to any person, but this is achieved in different ways. The corresponding fragments of each event log are shown below them in the figure. We will first generate all *L*-grams. To this end, we take a window of size *L* around all event traces and generate all possible *L*-grams. This is shown in Figure 4. In this figure, three traces of S_1 are shown: EHABCDFG, HEABCDFG and HEABCDGF. Taking a window of 4 around EHABCDFG, we can generate five 4 -grams: EHAB, HABC, ABCD, BCDF and CDFG. In a similar vein, we can generate five 4-grams for HEABCDFG: HEAB, EABC, ABCD, BCDF and CDFG. Likewise, five 4-grams for HEABCDGF: HEAB, EABC, ABCD, BCDF and CDFG. All of the 4-grams for the four organizations event logs are shown in Figure 4. It should be noted that since the relations between the intermediate ac-

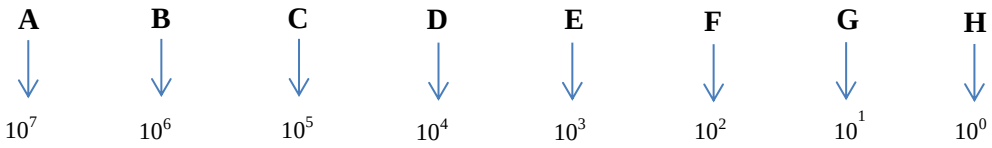


Fig. 5: Mapping of activities to a unique value as a power of 10.

L	O_1	O_2	….	O_N	L-gram number

Fig. 6: Data structure for storing L-grams

4	35	0	….	8	21130000

Fig. 7: A populated example of the data structure for storing a 4-gram

tivities do not matter, we do not need to calculate 4-combinations or 4-permutations in our work. With this approach we are able to generate L-grams with any size L.

3.2 Detecting common L-grams between organizations' event logs

After generating all *L*-grams ($L = 1$ to maximum length of all traces), we will now need to detect shared *L*-grams between organizations that respect the properties outlined in Definition 2. We present a fast and accurate method for identifying such common *L*-grams. For the sake of simplicity, we will follow the previous running example. Assume that there are 8 activities in all organizations named: A, B, C, D, E, F, G and H. We map each activity to a numerical representation, which is a power of 10 as shown in Figure 5. Now, we assign each trace T to a numerical value (T_{Value}) using the following formula:

$$T_{Value} = \sum_{j \neq I,\, j \neq O} A_j + \varepsilon \times A_I + \kappa \times A_O \qquad (1)$$

where A_I is the numerical representation of the first activity of the trace and A_o is the numerical representation of the last activity and $A_j\,(j \neq I, J \neq O)$ is the numerical representation of other activities in the trace. Using this formula, we are able to develop a unique numerical representation for each *L*-gram. In light of this, ABCD and ACBD traces, which are operationally identical according to Definition 2, will have the same value. The reason is that each activity regardless of its position in the trace has a unique value (a power of 10). We have added two different coefficients for the first and last activities, which distinguish the first and last positions in the trace. The important point is that the coefficients should make a difference between first and last positions and the other positions in a trace. An example of mapping traces to numerical values is shown in Table 1 by assuming ε is 2 and κ is 3. The two constants ε and κ can be any value as long as they distinguish between the first and last positions and the rest of the positions and they should not be a power of 10.

As shown in Table 1, ABCD and ACBD traces have equal numerical values. DBCA has different value compared to ABCD because their first and last activities are different. The red numbers in Figure 4 are numerical values, which have been assigned to each 4-gram.

Table 1- An example of mapping traces to numerical values

Trace	Mapping Calculation	Numerical Value
ABEG	$2\times10^7+10^6+10^3+3\times10^1$	21,001,030
ABCD	$2\times10^7+10^6+10^5+3\times10^4$	21,130,000
ACBD	$2\times10^7+10^5+10^6+3\times10^1$	21,130,000
DBCA	$2\times10^4+10^6+10^5+3\times10^7$	31,120,000

If an L-gram contains a loop (at least one activity is repeated more than once), duplicated activities will be removed and the L-gram will be converted to an $(L-k)$-gram (k is the number of removed activities).

3.3 Proposed Algorithm

Algorithm 1 shows the pseudo-code of the algorithm that extracts morphological fragments from multiple event logs. It consists of two functions called: a) *getMorphologicalFragment*, which is the main function and returns the best-detected fragments and b) *extracL-Fragment*, which extracts fragments with length L. For the sake of efficiency, we have defined a new data structure, which is shown in Figure 6. In the *L-gram number* cell, numerical value of the fragment is shown. Oi $(1 < i < N)$ cell contains frequency of *L-gram number* in Oi organization and cell L shows the length of the fragment. Therefore, Figure 7 shows a fragment with numerical value of *21,130,000* and length 4, which has occurred 35 times in O_1 and 8 times in O_N and never in O_2.

Algorithm1

```
1   var f-array ;
2   var fragments;
3   var maxIndex :=-1;
4   function getMorphologicalFragment()
5   begin
6     maxFragNo := 0;
7     for L:=T to 1 do
8       fc := extracL-Fragment (L,0);
9       if fc > maxFrag
10        maxFrag := fc;
11        maxIndex := L;
12      end
13    end
14  end
15
16  function extracL-Fragment(L, var fragNo)
17    if f-array(L) is fragment
18      fragments[L] := getFragments();
19      removeFromTraces(f-array(L));
20      updateL-Gram(size(f-array(L)));
21    else
22      return 0;
23    end
24    fragNo ++;
25    extractFragment(T-L, fragNo);
26    return fragNo;
27  end
```

Fig. 8- The proposed algorithm for extracting common morphological fragments

We can index an array of this data structure based on the *L-gram number*. In Algorithm 1, *f-array* is of this new data structure type and contains all L-grams. In *extracL-Fragment* function if an L-gram is detected as a fragment, it will be removed from all traces (using *removeFromTraces* function) and the remaining parts of the traces will be updated (using *updateL-Gram*).

These operations are repeated till all fragments are identified. In Equation (2), the *isfragment* function is described.

$$isF(t) = \left\lceil \frac{\left(\sum_{i=1}^{S} \left\lceil \frac{Fr_i^t}{N_i} - \alpha \right\rceil \right)}{S} - \beta \right\rceil \qquad (2)$$

$$,0 \le \alpha \le 1$$

$$,0 \le \beta \le 1$$

Equation (2) defines function $isF(t)$, which determines whether t is a morphological fragment, or not. In this equation, S is the organization number, t is a candidate L-gram, Fr_i^t is the number of occurrence of t in organization i, N_i is the number traces in organization i. Here, α is a locality factor and β is a global factor. We employ the locality factor for removing noise or low frequency patterns. That means if a candidate L-gram has low frequency in each organization, even if a large number of organizations include it, it cannot be selected as a good morphological fragment, because it may be noise or an exception in these organizations. So, we calculate relative frequency of t and subtract α form it. If the ceiling function is equal to 1, it means that t has a good frequency in the organization. The constant β is then used to ensure that the identified morphological fragment has been at least seen in a minimum number of organizations.

3.4 Computational Complexity Analysis

The proposed approach is straightforward to implement, has polynomial time complexity and converges to a global solution. In this section, the time complexity of the proposed approach for the

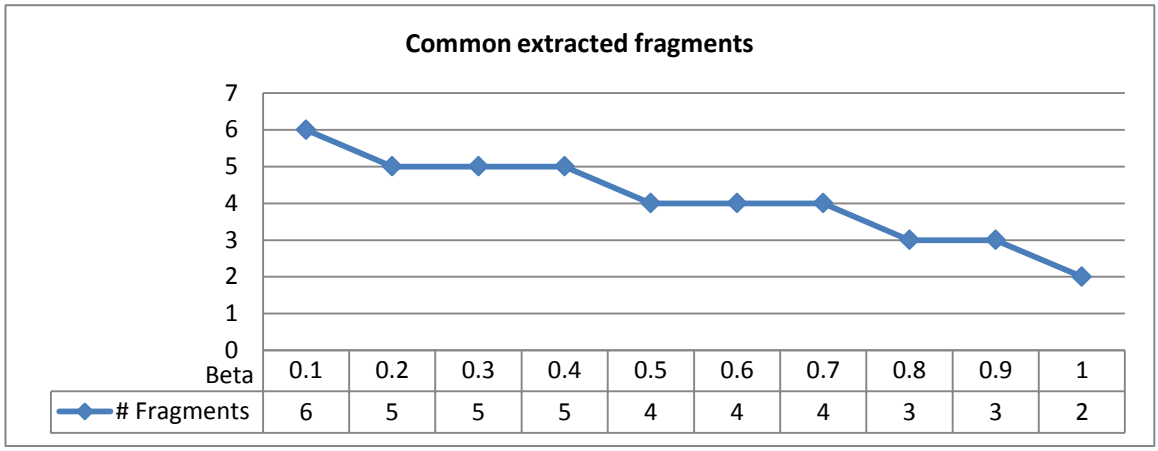


Fig. 9 - Different number of morphological fragments extracted in our case study.

main parts of the algorithm is explained. This time complexity is determined independent of the implementation technique.

3.4.1 Time complexity of generating L-grams

Assume that we have S organizations, each of which contains N traces of event logs. Also, assume that each trace has a maximum size of T. We can calculate the number of created L-grams as follows:

$$L-gram_{\#} = \sum_{i=1}^{T} S \times N \times (T-i+1)$$

$$= S \times N \sum_{i=1}^{T} (T-i+1)$$

$$= S \times N \times (\frac{T^2+T}{2}) \quad (3)$$

As seen in Equation (3), the time complexity of generating L-grams is $O(SNT^2)$. As the number of organization is limited, the values of S can be ignored. Thus, the time complexity for this part is $O(NT^2)$.

3.4.2 Time complexity of detecting common L-grams

The time complexity for calculating numerical value corresponding to each trace is constant C. Hence, the time complexity for calculating numerical value of all traces is equal to the number of them. Likewise, the time complexity for detecting common fragments is in the order of the number of traces. Therefore, the time complexity of detecting common L-grams and the proposed method as a whole is $O(NT^2)$.

4 Case Study

In this section we give a real case on how we used the proposed approach to extract the common fragments in workflow management system in municipality of *Mashhad*[1]. This system is an electronic organization system and a large number of processes are implemented in it such as *vacation request process*, *settlements process*, *services purchase*, and others.

4.1 Background

Mashhad is the second most populous city in *Iran* and is the capital of *Khorasan Razavi Province*. *Mashhad municipality* has 13 regions and 16 organizations. In this case study we selected the *purchasing* process event logs for extracting common morphological fragments. The purchasing process is executed in most regions and organizations of *Mashhad municipality*, but is customized differently for each of them.

When an organization intends to place an order, the responsible employee fills in the required information, including the general information, goods information (type, numbers, code, date etc.) and some extra specifications. Then a *purchasing* process is started. Several checks and confirmations may be required during the process. For example, the sales department can check the order

[1] - gam.mashhad.ir

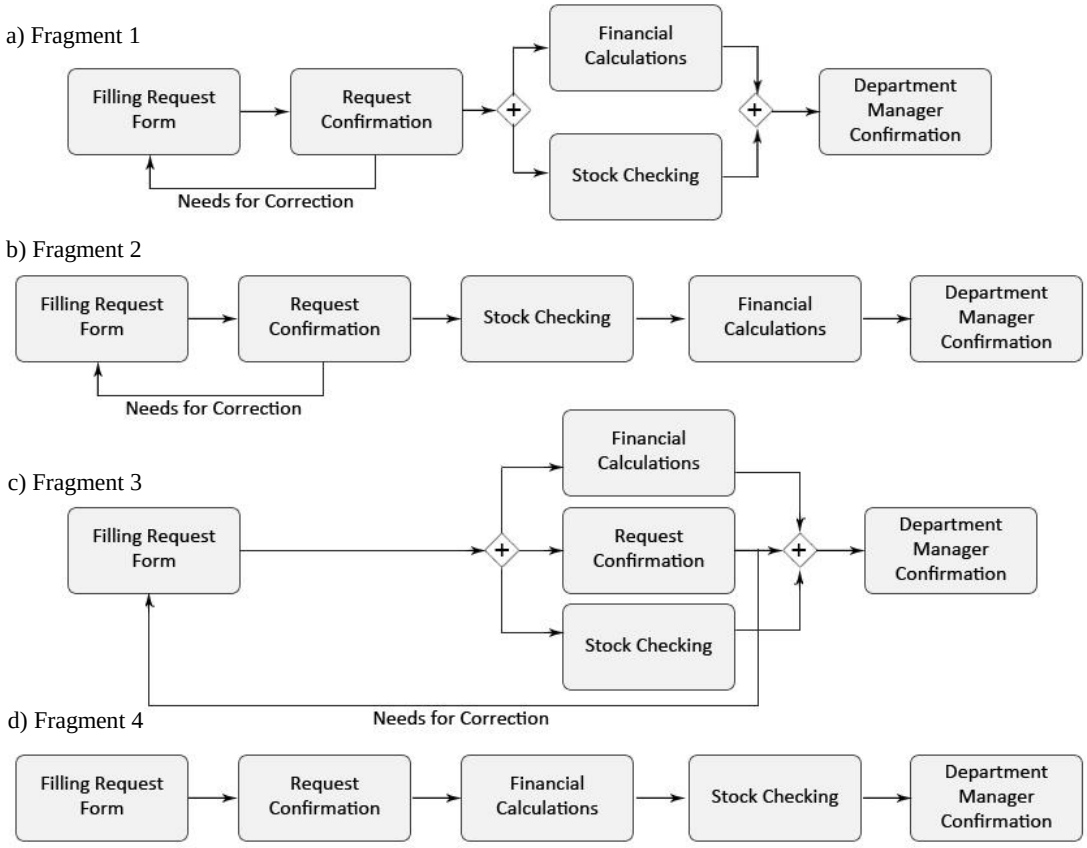


Fig. 10: Morphological fragments extracted in our experiments.

and complete any missing fields; the stock department checks the stock; and the financial department calculates the price. Depending on the specific organizations, these sub-processes might be executed with different orderings. For instance, the checks performed by the stock and the financial departments can be done in parallel or in sequence.

It should be noted that given Mashhad municipality had not logged the event data in a suitable format such as XES[2], we had to manually make some changes to the event log data. In Table 2, some properties of this event data are shown.

Table 2- Statistics about purchasing process models and its event logs

Feature	Value
Number of organizations	18
Average number of process model elements	64
Average number of selected event log records form each data set	1000

4.2 Finding Common Morphological Fragments

The result of the proposed fragmentation technique on the *purchasing* event log is shown in Figure 9. There were 18 organizations and regions that had different *purchasing* processes. In addition to executing our fragmentation process, we also investigated the values of β that was defined

[2] - www.xes-standard.org

in Section 4. Changing β can results in higher or lower number of common morphological fragments. In Figure 9 the result of the fragmentation process with different values of β are shown. With value of 0.5 for β, 4 common fragments among fifty percent of organizations and regions are detected. Maximum number of common fragments is achieved when β is 0.1. In this case, 6 common fragments are detected. Our experimental observations show that 4 of these 6 fragments are good and 2 of them are small and are not too meaningful. Our investigation shows that lower values for β do not produce meaningful fragments and higher values for it will identify more accurate fragments. Our understanding is that the value of 0.6 is a good choice for β. In other words, if a fragment is common in more than sixty percent of organizations, it can be considered to be a suitable morphological fragment that can be reliably extracted.

Another interesting finding from our case study is the following: If the difference between the number of morphological fragments identified by lower beta values is close to the number of fragments identified by higher beta values, it can be inferred that the process variance between organizations is very low. In other words, the organizations have implemented their processes quite similarly.

In Figure 10, four variants of a morphological fragment that have been extracted from the purchasing processes of 18 different organizations are shown. These fragments perform an equivalent task in different ways and therefore have been selected as morphological fragments by our approach.

5 Discussion

In Section 4, we presented a novel approach for extracting common morphological fragments. As described previously, each activity in event logs is uniquely represented. So one might think that techniques for finding sub-string methods such *Longest Common Subsequent* (LCS) [16] or some modern computational approaches in genomic [23] can be used for extracting common fragments. In this section, we want to analyze LCS as one of the famous string processing methods and discuss its pros and cons.

LCS is a classic computer science problem and its goal is to find the longest common subsequence among the set of sequences and has many application in bioinformatics [16]. To describe LCS, let $S_1 = a_1 a_2 \dots a_m$ and $S_2 = b_1 b_2 \dots b_n \ \ (m \leq n)$ be two sequences. A sequence that can be obtained by deleting some symbols of another sequence is referred to as a subsequence of the original sequence. A common subsequence of S_1 and S_2 is a subsequence of both S_1 and S_2. A longest common subsequence is a common subsequence of greatest possible length.

For the general case of an arbitrary number of input sequences, the problem is NP-hard [4]. When the number of sequences is constant, the problem is solvable in polynomial time through dynamic programming. For the case of two sequences of n and m length, time complexity of the dynamic programming algorithm is $O(n \times m)$. For an arbitrary number of input sequences, time complexity of dynamic programming approach is:

$$O\left(N \prod_{i=1}^{N} n_i\right) \quad (4)$$

where N is the number of sequences. There exist algorithms with lower time complexity [4].
In comparison to LCS, our proposed approach has polynomial time complexity versus exponential complexity of LCS for the variable number of $L - gram$. More importantly, there are constraints on morphological fragments such as insensitivity of activity sequences as long as they are within the same fragment that cannot be satisfied by LCS. Therefore, it is not possible to employ LCS directly for extracting common fragments in process logs.

6 Related Work

The contribution of our work is related to a stream of research, referred to as process fragmentation. A survey of different fragmentation (also referred to as modularization, see e.g. [15], or decomposition, e.g. [21]) algorithms can be found in [13]. In [13], a classification framework investigates the characteristics of fragmentation techniques for process models and the properties of the resulting process fragments. The classification criteria presented in [13] provides (1) a basis for classifying existing fragmentation techniques, and (2) a "check-list" of what authors should consider in their fragmentation approaches.

Reijers *et.al.* [15] propose three types of criteria for fragmentation namely: a) the block-structuredness of the subprocess, b) the connectedness of nodes in the subprocess, and c) the similarity of the labels of the nodes in the subprocess. A process is called block-structuredness if it has a single *entry* and a single *exit* (SESE). A collection of nodes is connected if the nodes in the collection are more strongly connected by arcs to each other than to nodes outside this collection.

In [9], a fragment identification approach based on sharing analysis is presented. They have used Horn clauses designed to adequately enforce sharing between inputs and outputs of the workflow activities. Nevertheless, it is not interesting in fragmentation because describing the data objects does not influence the sharing analysis processing, and consequently, the obtained fragments are the same as when data objects are taken without description.

The idea of distributed processes execution has been tackled in [19]. The authors dynamically separate one integrated workflow model into small partitions at process runtime and allocate them to different servers to be executed. The focus of this work is on distributed execution of one process model.

To the best of our knowledge, all fragmentation techniques receive one process as input. In [13], it has been noted that fragmentation is the act of creating process fragments out of *one process* model by applying a fragmentation technique. There has only been the work in [8] that processes a collection of process models as input and extracts common sub-process using process similarity methods. Although in [8] a collection of process models is received as input, but the proposed fragmentation approach only considers one process model at a time instead of the collection together as a whole. Also, all fragmentation algorithms that are available in the literature are based on the process model structure and not their event logs.

7 Conclusion

In this paper, we presented a new definition for operationally identical sub-processes called morphological fragments. Based on this definition, we presented a new method for extracting common fragments from a collection of event logs. This algorithm extracts morphological fragments directly from a collection of event logs in a polynomial time complexity. In our work, the fragmentation algorithm considers a collection of process models as a whole. There are some directions for future work which are summarized as follows:

- Extracted common fragments from various organizations can be used in many applications. We intend to use these fragments in evaluating process models and extracting best recommendable process model among a family of process models. We would like to investigate the possibility of identifying and using the most suitable fragments to generate the best process model according to user preferences.
- We have defined two local and global parameters in Section 4. There is the need to analyze these parameters for different values to identify their impact on the quality of the extracted morphological fragments.
- In our work, we have used a top-down method for extracting common fragments. In this approach, we extract the fragments, which have prescribed constraints. For future work, we would like to use statistical analysis in a bottom-up approach to reach a new definition for operationally identical fragments, which would include a wider range of fragments with fewer limitations.

References

[1] Aalst, W. Van Der. *Process-aware information systems: Lessons to be learned from process mining.* T. Petri Nets and Other Models of Concurrency, 2:1-26, 2009.

[2] Aalst, W. Van Der. *Process Mining: Discovery, Conformance and Enhancement of Business Processes*. Springer-Verlag, Berlin, 2011.

[3] Aalst, W. Van Der. *Process Mining: Overview and Opportunities*. ACM Transactions on Management Information Systems, Vol. 99, No. 99, Article 99, 2012.

[4] Bergroth, L., Hakonen, H., and Raita , T. *A Survey of Longest Common Subsequence Algorithms*. SPIRE (IEEE Computer Society), 2000.

[5] Buijs J.C.A.M., Dongen B.F. VAN, and Aalst, W. Van Der. *Mining Configurable Process Models from Collections of Event Logs*. In Proceedings of the 11th International Conference on Business Process Management (BPM. Lecture Notes in Computer Science Series, vol. 8094. Springer-Verlag, Berlin, 33–48, 2013.

[6] Dumas M., Rosa M. La, Mendling J., Reijers H.A. *Fundamentals of Business Process Management.* Springer-Verlag, Berlin, 414pages, 2013.

[7] Dijkman, R.M., Dongen, B.F., Dumas, M., Garcia-Banuelos, L., Kunze, M., Leopold, H., Mendling, J., Uba, R., Weidlich, M., Weske, M., Yan, Z. *A short survey on process model similarity*. In: Bubenko, J., Krogstie, J., Pastor, O., Pernici, B., Rolland, C., Solvberg, A. (eds.) Seminal Contributions to Information Systems Engineering, pp. 421–427. Springer, Heidelberg, 2013.

[8] Gao, X., Chen, Y., Ding, Z., Wang, M., Zhang, X., Yan, Z., Wen, L., Guo, Q., Chen, R. *Process Model Fragmentization, Clustering and Merging: An Empirical Study*. Lecture Notes in Business Information Processing Volume 171, 2014, pp 405-416, 2014.

[9] Dragan Ivanovic , Manuel Carro, and Manuel Hermenegildo. *Automatic Fragment Identification in Workflows based on Sharing Analysis*. In Mathias Weske, Jian Yang, Paul Maglio, and Marcelo Fantinato, editors, *Servíce-Oríented Computíng –ICSOC 2010*, number 6470 in LNCS. Springer Verlag, 2010.

[10] Kaufman, L., Rousseeuw, P.J. *Finding Groups in Data: An Introduction to Cluster Analysis.* Wiley, New York ,1990.

[11] Larosa, M., Dumas, M., Uba, R.,and Dijkman, R. M. *Business Process Variability Modeling: A Survey* .ACM Transactions on Software Engineering Methodology 22,2,11, 2013.

[12] Larosa, M., Dumas, M., Uba, R.,and Dijkman, R. M. *Business process model merging: An approach to business process consolidation*. ACM Transactions on Software Engineering Methodology 22,2,11, 2013.

[13] Mancioppi M., Danylevych O. *Towards classification criteria for process fragmentation techniques.* in: Proc. BPM Work., Springer Berlin Heidelberg, pp. 1–12, 2012.

[14] Milani F, Dumas M, Matulevičius R. *Decomposition Driven Consolidation of Process Models, Advanced Information Systems Engineering.* Lecture Notes in Computer Science Volume 7908, pp 193-207, 2013.

[15] Reijers, H.A.,Mendling, J., Dijkman, R.M. *Human and automatic modularizations of process models to enhance their comprehension.* Inf. Syst. 36(5), 881–897, 2011.

[16] Ronald I. *Bounds on the Number of Longest Common Subsequences*. Greenberg, 2003.

[17] Schunselaar, D., Verbeek, E., Aalst, W.M.P. van der, and Reijers. H. *Creating Sound and Reversible Configurable Process Models Using CoSeNets.* In W. Abramowicz, D. Kriksciuniene, and V. Sakalauskas, editors, Business Information Systems (BIS 2012), volume 117 of LNBIP, pages 24–35. Springer, 2012.

[18] TFPM – IEEE TASK FORCE ON PROCESS MINING. *Process Mining Manifesto.* In BPM Workshops. Lecture Notes in Business Information Processing Series, vol. 99. Springer-Verlag, Berlin, 2012.

[19] W. Tan and Y. Fan. *Dynamic workflow model fragmentation for distributed execution*. Computers in Industry, vol. 58, no. 5, pp. 381–391, 2007.

[20] Valenca, G., Alves, C., Alves, V., and Niu, N. *A systematic mapping study on business process variability*. Int. Journal of Computer Science & Information Technology 5,1, 2013.

[21] Vanhatalo, J., Volzer, H., Leymann, F. *Faster and more focused control-flow analysis for business process models through sese decomposition.* in : B. Kramer, K.-J. Lin, P. Narasimhan (Eds.), Proceedings of the 5th International Conference on Service-Oriented Computing—ICSOC 2007, Vienna, Austria, September 17–20, Lecture Notes in ComputerScience,vol.4749, Springer 2007, pp.43–55, 2007.

[22] Weske, M. *Business Process Management Architectures*. in Business Process Management , Springer Berlin Heidelberg, pp. 333 –371, 2012.

[23] Xuhua, X., *Bioinformatics and the Cell: Modern Computational Approaches in Genomics*. Proteomics and Transcriptomics. New York: Springer. p. 24, 2007.