applied sciences

MDPI

Article

A Novel Process Recommendation Method That Integrates Disjoint Paths and Sequential Patterns

Danni Han [1,*], Chaoxue Wang [1], Genqing Bian [1], Bilin Shao [2] and Tengteng Shi [1]

[1] School of Information and Control Engineering, Xi'an University of Architecture and Technology, Xi'an 710055, China
[2] School of Management, Xi'an University of Architecture and Technology, Xi'an 710055, China
* Correspondence: hdn@xauat.edu.cn

Abstract: As the primary means of modern enterprise management, business process management (BPM) technology has become the mainstream development trend of modern enterprise management. The efficient and accurate establishment of business processes is essential for effective BPM. However, the traditional manual-based modeling approach is time-consuming and error-prone. To overcome this, process recommendation technology can improve the intelligence and efficiency of modeling to a certain extent. However, existing process modeling recommendation methods suffer from the problem of low accuracy and neglecting short-process models. Therefore, a novel process modeling recommendation method that integrates disjoint paths and sequential patterns was proposed. This method uses edge-disjoint paths for the first time to represent the behavioral semantics of processes, and an improved contiguous sequential pattern mining algorithm was proposed to mine the contiguous path sequential patterns (CPSPs) of edge-disjoint paths. In the process modeling recommendation stage, the k CPSPs with the highest matching degree with the current reference model process were calculated, and the last node in these CPSPs was used as the set of recommendation nodes. In cases with CPSPs with the same matching degree, the one with the higher value was recommended according to their corresponding lift, confidence, and support degrees. Through experimental evaluation and comparison, it was shown that the proposed method effectively improved the accuracy of the recommendation of both short-process and long-process models while ensuring effectiveness and time efficiency.

Keywords: process recommendations; process modeling; sequential pattern mining; edge-disjoint path; business process management

Citation: Han, D.; Wang, C.; Bian, G.; Shao, B.; Shi, T. A Novel Process Recommendation Method That Integrates Disjoint Paths and Sequential Patterns. *Appl. Sci.* **2023**, *13*, 3894. https://doi.org/10.3390/app13063894

Academic Editors: Chaogang Tang and Dong Zeng

Received: 25 February 2023
Revised: 14 March 2023
Accepted: 16 March 2023
Published: 19 March 2023

1. Introduction

To meet the increasing business demands, modern enterprises have raised the requirements for process modeling. The efficient and accurate establishment of business processes is a critical guarantee for BPM in modern enterprises and is one of the vital issues for business management software providers to accurately capture the flow of customers. However, process modeling is a time-consuming process that may require modeling a large combination of task nodes in the process, making it difficult for users to identify the appropriate node for forming a business diagram. Traditional human-based modeling approaches are time-consuming, imprecise, and require the modeler to have a high level of professionalism. Therefore, process modeling recommendation technology, which automatically recommends the next process node or the entire process based on the process nodes entered by the modeler, has been gaining more and more attention and research in the process management of enterprises.

In the study of process modeling recommendations, one class of methods is based on log data [1,2]. However, due to the high requirements of the dataset and the limited application scope, process modeling recommendations based on the original process library data

can better satisfy most application scenarios. Currently, the relevant mainstream research is primarily divided into two categories. The process modeling recommendation based on graph structures [3–9] received more attention via related work, but it results in a loss of semantic information and a higher mining complexity. In contrast, the process recommendation method based on behavioral semantics [10–13] simulates the execution behavior of the process through the process model structure and has a higher reference value.

Therefore, this study proposed a process modeling recommendation method that combines disjoint paths and sequential patterns from the perspective of process behavior. It addresses the issues of poor accuracy and neglect of short-process models that are common in the two methods above. The main contributions of this study are as follows:

1. A new edge-disjoint path extraction algorithm (EPE), which extracts the behavioral semantics of process library models based on the task-based process structure tree (TPST) [14], was proposed. This solves the problem of ignoring the semi-ordered structure characteristics of the TPST in the existing literature.
2. A contiguous path sequential pattern mining (CPSPan) algorithm for process recommendation was proposed; this algorithm mines frequent and continuous subsequences of edge-disjoint paths, increasing the number of process fragments to be matched in the reference model. This provides a greater range of options and more dependable guidance for process recommendations.
3. A new process recommendation strategy was proposed, in which the node recommendation degree was introduced to measure the influence of the node position on the recommendation. The lift, confidence, and support degrees were introduced as the basis for measuring the importance of the process fragments to be matched.
4. An experimental analysis demonstrated that the proposed method could effectively recommend the next node in the process modeling with an accuracy of 89.67% and 90.57% on the real and simulated datasets, respectively. Additionally, the method met the time requirements.

2. Related Work

Although the field of BPM has evolved, generally accepted definitions of their primary terms, business process, and workflow are still lacking in the scientific literature [15]. In this study, a business process was described as an orderly work process consisting of process nodes and execution methods. A business process model with a sequential, selective, parallel structure is shown in Figure 1.

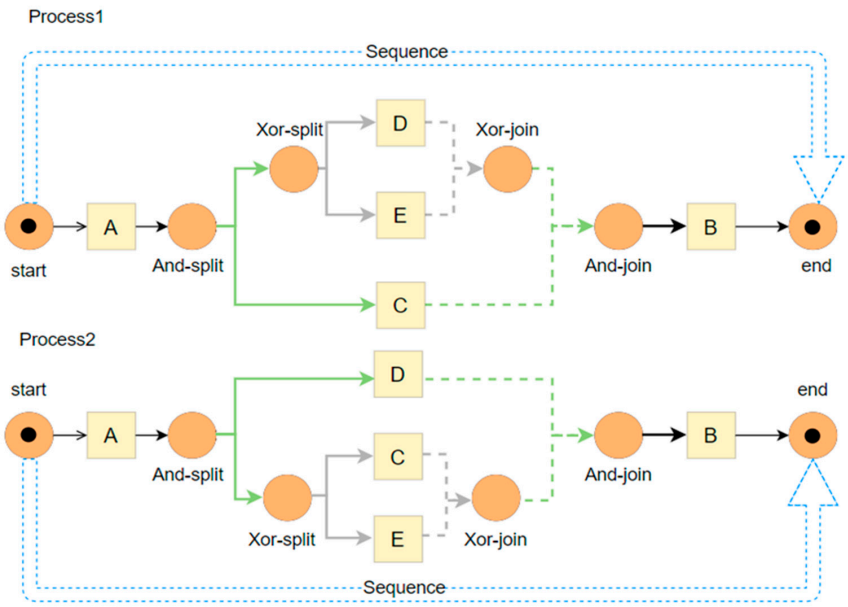

Figure 1. Business process models represented using Petri nets.

In the current era of data and information domination, companies require new BPM tools and optimization solutions to cope with the increasing complexity of industrial operations. Gavvala et al. proposed a business process optimization method based on natural observation in [16], which first conducts extensive exploration based on an eagle search, and then uses a whale optimization algorithm to find the optimal solution within the narrowed area. Zhu et al. [17] used process mining to extract fine-grained service orchestration knowledge from the event logs of edge intelligence services to improve business processes. Represa et al. [18] examined and compared various solutions for workflow management and execution to support decision-making in a variety of scenarios. However, as mentioned in the text, the feature that seems less relevant and lacking in most solutions is support for users or manual tasks. Instead, the focus was on software systems that can be designed by software systems that work autonomously. However, this can neglect the individual needs of the modeler when undertaking business process modeling. Process recommendation technology, which is an essential part of BPM, was designed to analyze the event logs of a process or raw process library data to get the next recommended process node or complete process from the modeling node being entered by the modeler.

For the recommendation problem in the business process modeling stage addressed in this study, Wang et al. [19] proposed a process recommendation method based on cost constraints to solve the difficulty of distinguishing similar processes. Meanwhile, Luo et al. [20] proposed a collaborative filtering-based business process recommendation method that mines user behavior preferences based on the historical trajectories of executed processes by users to select or recommend current users' activities and achieve the automatic construction of the whole process. Both methods rely on event log data for process recommendation, which requires a significant amount of data to ensure accuracy. However, the accuracy of these methods cannot be guaranteed when the amount of data is insufficient. Additionally, when applied to the field of process recommendation, if there are nodes in the process fragment inputted by the modeler that are not available in the process repository, the accuracy of the recommended methods cannot be guaranteed.

Based on the original process database data, Deb et al. [9] proposed a business process design method that quantifies and qualifies service quality to obtain a better path for business process models. Ye et al. [21] proposed a process recommendation technique based on a process regularization matrix that supports complex business process structures and fuzzy recommendations in practical applications. Fellmann et al. [22] proposed a data model that analyzes each process that supports process modeling. These methods are based on the graph structure of the process model for process recommendation, providing high similarity accuracy and more relevant research. However, there is a loss of semantic information, and the mining complexity is slightly higher. Furthermore, most graph-structure-based methods rely on the "distance" of the graph to compare the similarity of processes, which may not effectively reflect the similarity between processes containing a loop structure.

A behavior-based business process recommendation is an approach that is closer to the real-life situations of process operation. Wang et al. [23] proposed an independent path-based process recommendation algorithm to improve biomedical process modeling. This algorithm calculates the similarity scores between the reference process and any biomedical process in the repository based on their independent paths. The top k most similar biomedical processes are then recommended for the reference process. Jalali et al. [24] developed a hybrid business-process-oriented modeling approach based on declarative rules that link mandatory cross-domain concerns and imperative business process models. Li et al. [25] established the target profile of the reference model and compared it with the behavior profile extracted from the process library to find the process model with the same goal. Gui et al. [26] used the process structure tree and the vertex-disjoint path of the reference model and compared this with the vertex-disjoint path of the process library model for similarity calculation, thus completing the recommendation. Behavior-based process recommendation algorithms focus on the execution behavior of the process to

restore the execution semantics of the process, which can be more complex. Additionally, as these algorithms do not concentrate on mining the process structure, the recommendation process relies on the existing process model, which lacks the ability to recommend more related nodes for the business process.

In addition, current process recommendation technology primarily focuses on the accuracy of long process models, and there is limited research on process recommendations when the reference models and process library models are short. The experimental section of [25] examined the accuracy of short-process models. However, their experiments showed that the accuracy was below 65% when the number of nodes currently modeled was either 4 or 5, which resulted in some instability in the modeling process.

Aiming to address the aforementioned issues, the EPE and CPSPan were developed to conduct offline pre-processing and mining of process library data. The process node recommendation algorithm was devised to provide online recommendations, which combine lift, confidence, support, and process-matching degrees. This method was validated through experiments using both real and extended simulation datasets to enhance the recommendation accuracy of both short- and long-process models while ensuring time efficiency.

3. Proposed Method

This section proposes the specific process modeling recommendation method, the basic flow of which is shown in Figure 2.

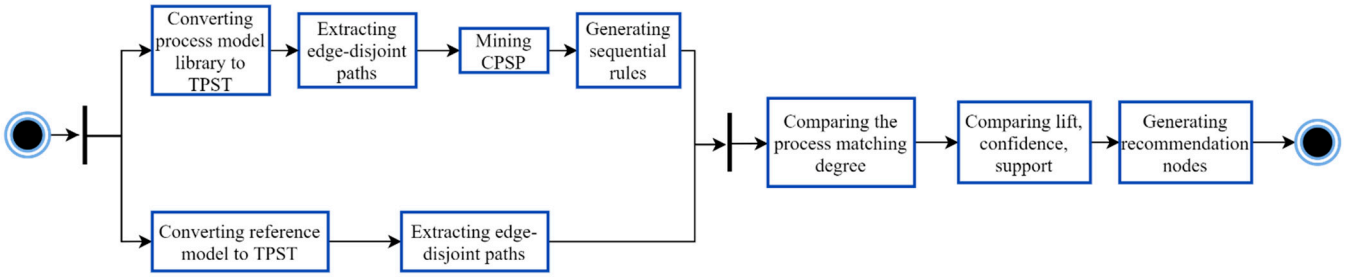

Figure 2. The basic flow of the proposed method as represented using a UML activity diagram.

The method aims to assist modelers in rapid modeling by extracting the edge-disjoint paths of the process model and mining all the CPSPs. The process-matching degree between these sequential patterns and the edge-disjoint paths extracted from the reference model is computed. The top k sequential patterns in the process library with the highest process-matching degree with the reference model are selected, and the last node among them is used as the recommended node set.

The overall framework of business process recommendation using the method in this study is shown in Figure 3, which is divided into three modules: preprocessing, CPSP mining, and node recommendation.

1. The preprocessing module is divided into two parts. The first part analyzes the process model library in the TPST and saves it in the TPST library. Figure 4, for example, shows a TPST converted from the business process model in Figure 1. In the second part, all models in the TPST library are extracted according to the different gateway nodes and stored in the edge-disjoint path library.
2. The CPSP mining module mines CPSPin the edge-disjoint path library, and a sequential rule is generated according to the excavated CPSP and its corresponding calculated degree of lift, confidence, and support. All the sequential rules are saved in the sequential rule library for the subsequent recommendation.
3. The node recommendation module first calculates the process-matching degree between the edge-disjoint path and each CPSP in the sequential rule library excavated by the reference model. It then selects the appropriate recommendation result based on the process-matching degree, lift degree, confidence degree, and support degree. The specific recommended steps are as follows:

(i) The edge-disjoint path is extracted from the TPST that was transformed by the process model library in the preprocessing module using the EPE.

(ii) The frequent CPSP mining is carried out using CPSPan on the edge-disjoint path library. The corresponding degrees of lift, confidence, and support of each CPSP are calculated, and the results are saved into the sequential rule library T.

(iii) The corresponding TPST is generated by the process decomposition of the reference model, and its edge-disjoint paths are extracted.

(iv) All sequential rules in the sequential rule library are circled to calculate the process-matching degree between each sequential rule's CPSP and the edge-disjoint path in the reference model.

(v) At the end of the cycle, the nodes are sorted according to the calculation result of the process-matching degree, and the recommended node set is the last node of the CPSP in the top k sequential rules after sorting. If there are CPSPs with the same matching degree, the nodes are ranked according to the degrees of lift, confidence, and support, and the larger value is recommended first.

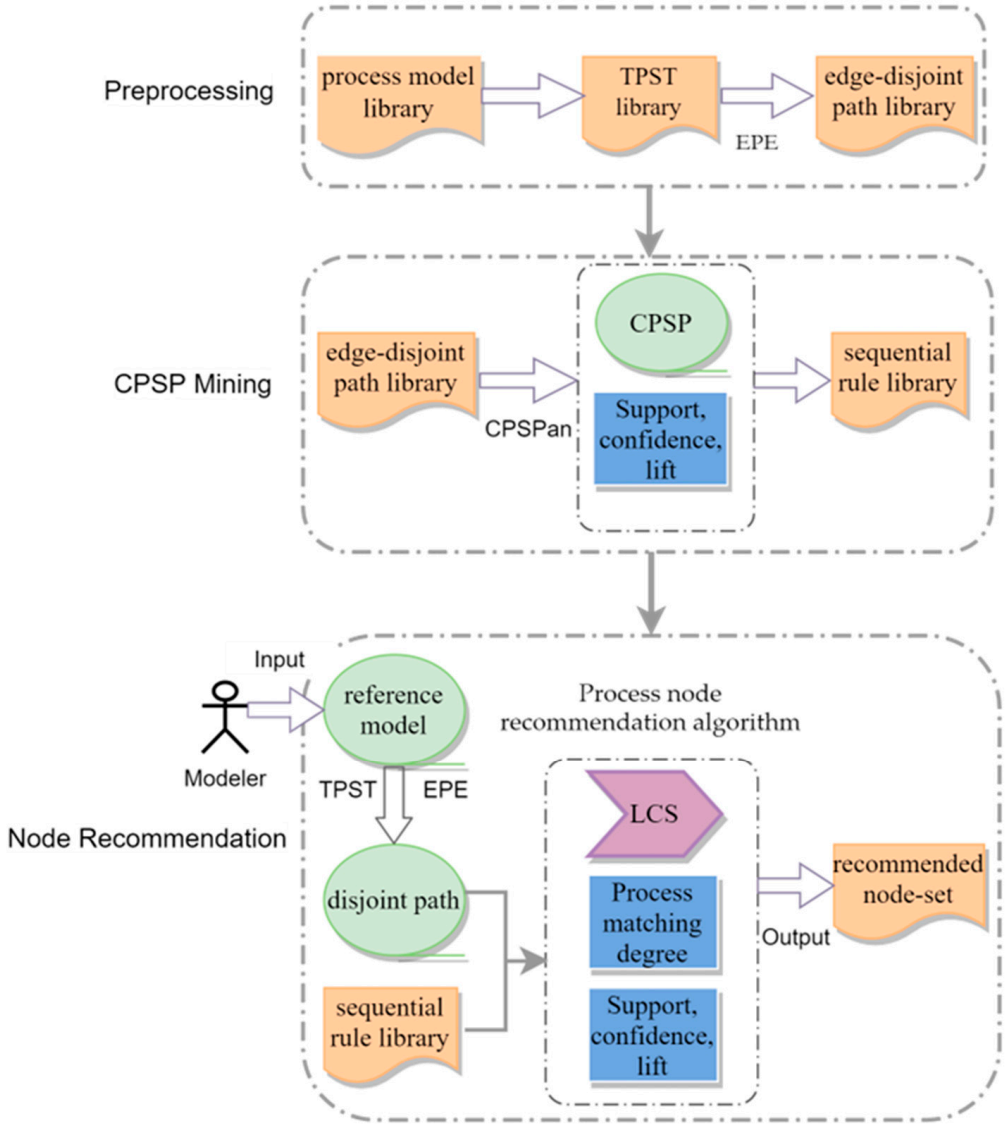

Figure 3. Business process recommendation framework.

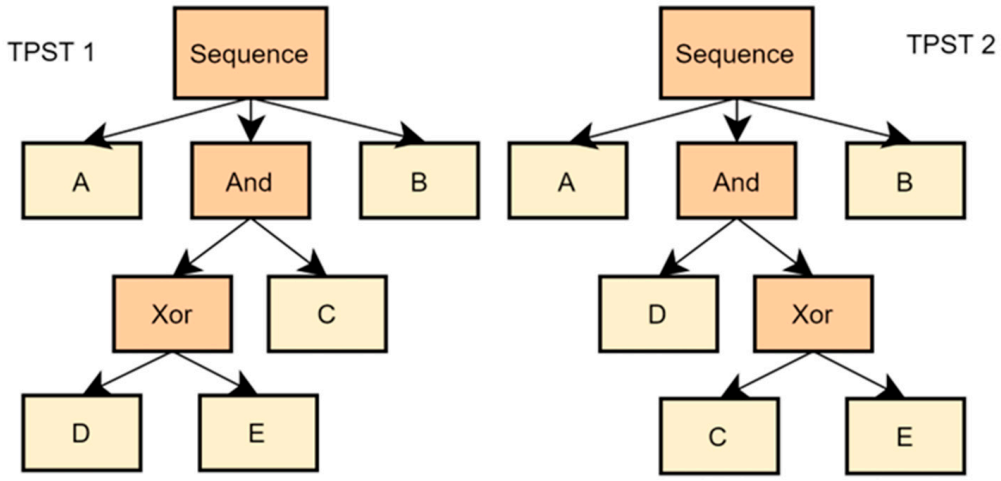

Figure 4. TPST structure corresponding to the business process in Figure 1.

3.1. Edge-Disjoint Path Extraction Method

The disjoint paths in a process model represent one of the execution paths, and thus, a process's behavior can be expressed by its set of disjoint paths. While process similarity comparison methods based on disjoint paths [2,26] are highly interpretable and easy to implement, all current methods rely on vertex-disjoint paths. However, as per the literature [27,28], each execution of a vertex-disjoint path has at least one unexecuted node, which ignores the disorder of 'And' structures whose child nodes are disordered.

For example, consider the TPST1 shown in Figure 4, which includes an 'And' structure with child nodes 'C' and 'Xor', where the child nodes 'D' and 'E' of 'Xor' select one to execute. When using the vertex-disjoint path to express the behavioral semantics of a process, the resulting vertex-disjoint path extraction are <A,C,D,B> and <A,C,E,B>, indicating that the execution order of the process is A → C → D → B or A → C → E → B. This approach cannot distinguish the semantics of the 'And' structure or the 'Sequence' structure since the execution order of node 'C' always comes before nodes 'D' and 'E'. In contrast, edge-disjoint paths allow nodes to be repeated, and thus, this study introduced edge-disjoint paths to express the behavioral semantics of processes. The corresponding edge-disjoint path extraction results in Figure 1 are <A,C,D,B>, <A,C,E,B>, <A,D,C,B>, and <A,E,C,B>, which meet the requirements for process recommendation.

Inspired by the vertex-disjoint path extraction algorithm (VPE) [26] and based on the definition of TPST structure and edge-disjoint path, this study proposed the EPE for the process model. A breadth-first search traverses each node of the TPST. The edge-disjoint path extraction is carried out according to the parent nodes of different types of current nodes and the characteristics of each gateway node. In addition to the above improvement for the disorderly characteristics of the child nodes in the 'And' structure, the problem that occurs when the original algorithm has infinite disjoint paths in the loop structure is also solved.

The implementation of the EPE is performed in the following steps shown in Figure 5.

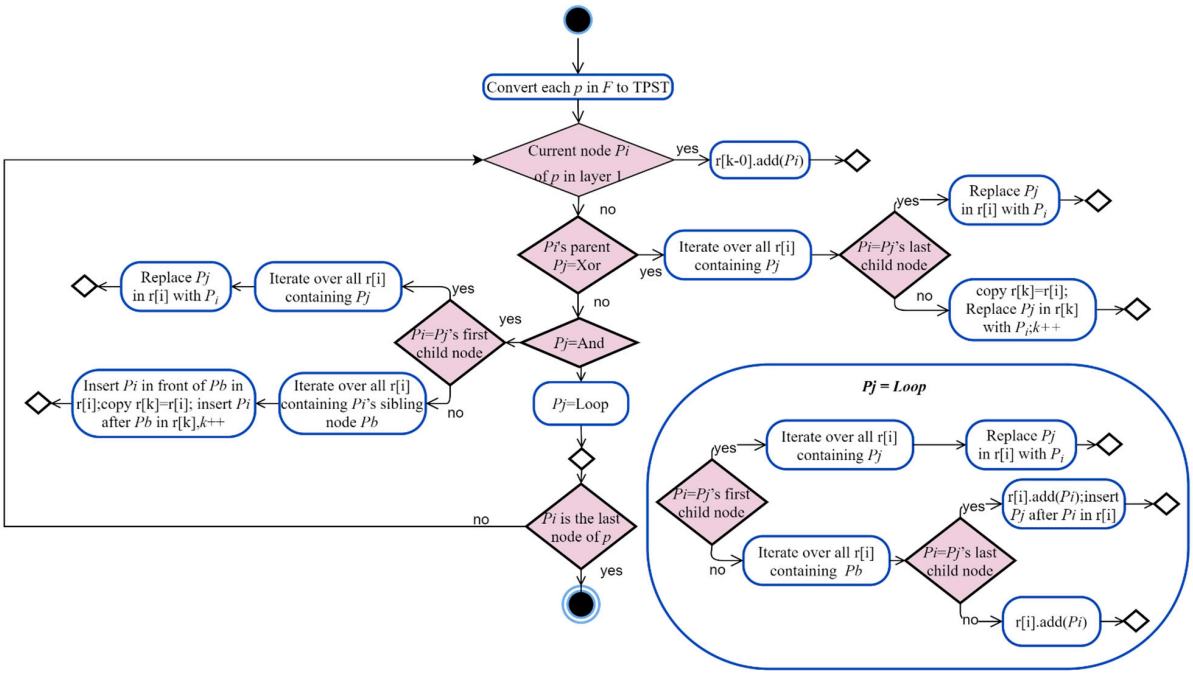

Figure 5. The EPE algorithm framework.

Algorithm 1 uses BFS to parse the TPST into edge-disjoint paths. The input for Algorithm 1 is a process library model, and the output is an edge-disjoint path table.

Algorithm 1. EPE for the process model

Input: process model library *F*
Output: edge-disjoint path table *R*
FOR (each process *p* in *F*) do{
Initialize set of edge-disjoint path *r*; *k* = 1; convert *p* to TPSP;
FOR *t* do BFS{
IF current node *Pi* in layer1 {r[k-0].add(*Pi*)
ELSE IF *Pi*'s parent node *Pj* is Xor{
FOR (each r[i] that contains *Pj*) do{
IF Pi = Pj's last child node {replace *Pj* in r[i] with P_i}
ELSE {copy r[k] = r[i]; replace *Pj* in r[k] with P_i; *k*++}}
ELSE IF *Pj* is And{
IF *Pi* = *Pj*'s first child node {FOR (each r[i] that contains *Pj*) do{replace *Pj* in r[i] with P_i}}
ELSE FOR (each r[i] that contains *Pi*'s sibling node *Pb*) do{
Insert *Pi* in front of *Pb* in r[i]; copy r[k] = r[i]; insert *Pi* after *Pb* in r[k]; *k*++}}
ELSE IF Pj is Loop{
IF *Pi* = *Pj*'s first child node {FOR (each r[i] that contains *Pj*) do{ Replace *Pj* in r[i] with P_i}}
ELSE FOR (each r[i] that contains *Pb*) do{
IF *Pi* = *Pj*'s last child node{r[i].add(*Pi*); insert *Pj* after *Pi* in r[i]}
ELSE {r[i].add(*Pi*)}}}}
Return *R* → r[i],r[k]

First, the top-level nodes are directly placed into the corresponding edge-disjoint path r[k-1]. Then, for the second level, if the current visiting node's parent node is 'Xor', for each r[i] that contains its parent node, a new path r[k] = r[i] is copied, and the current node replaces its parent node to form a new path. Because 'Xor' represents choice, multiple choices indicate multiple different edge-disjoint paths, and thus, multiple path collections need to be generated by copying. Only when the current visiting node's parent node is 'Xor' and the node is the last node of its parent node, a new path does not need to be copied, and the node can directly replace the parent node in the original r[i] path.

If the current visiting node's parent node is 'And', since 'And' represents parallelism and to show the characteristic of the child nodes being unordered, a new path needs to be copied. If the current node is the first child node of its parent node, a new edge-disjoint path does not need to be copied, and for each r[i] that contains its parent node, the current node replaces its parent node. If the current node is not the first child node, we need to consider the order between it and its sibling nodes in the sequence. We use the method of an alternating front to represent unorderedness. For each path r[i] containing sibling nodes, the current node is inserted in front of its sibling nodes in the original r[i] path. Then, r[k] is copied as the current r[i], and the current node is inserted after its sibling nodes in the new path r[k].

If the current visiting node's parent node is 'Loop', this means that the current node only has one edge-disjoint path. If the node is the first child node of its parent node, for each path r[i] containing its parent node, the current node replaces its parent node. If it is not the first child node of its parent node, that is, it is not the starting node of the loop structure, for each path r[i] containing its sibling nodes, we add the current node to the end of r[i]. Only when the current node is the last child node of its parent node, that is, the end node of the loop structure, do we add the current node to the end of r[i] and insert it after its parent node too.

Then, following the same procedure as for the second level, each level from the first to the nth level is visited in turn, and finally, a set of edge-disjoint paths R is obtained.

Take TPST1 in Figure 4 as an example. First, BFS is carried out to obtain 'A', which is placed in r[0]. Similarly, the nodes 'And' and 'B' of the first layer are added to r[0], and r[0] is [A,And,B]. The second layer is then traversed, and the 'Xor' node is discovered first. Since the parent node of 'Xor' belongs to type 'And' and is the first child node of the 'And' node, for each path r[i] that contains its parent node 'And', 'And' is replaced with 'Xor', where r[0] is [A,Xor,B]. Then, for the next node 'C', for each path r[i] that contains its sibling node 'Xor', node 'C' is inserted before the sibling node. Copy the new path r[1] and insert 'C' after the sibling node. Here r[0] is [A,C,Xor,B] and r[1] is [A,Xor,C,B]. Traversing the third layer, the first node is 'D', whose parent node is type 'Xor' and not the last child node. For each r[i] that contains the parent 'Xor', the new path is copied and the 'Xor' in r is replaced with 'D'. For r[0], the copy r[2] = r[0] takes place and the replacement node gets r[2] = [A,C,D,B]; likewise, the copy r[3] = r[1] takes place and the replacement node gets r[3] = [A,D,C,B]. Then, the next node 'E' is accessed because its parent is 'Xor' and is the last child node; for each path r[i] that contains the parent, replace the parent with 'E', and obtain r[0] = [A,C,E,B] and r[1] = [A,E,C,B]. The final output is [A,C,E,B], [A,E,C,B], [A,C,D,B], and [A,D,C,B].

Through experimental comparison and analysis with the VPE, the recommended hit rate of the EPE proposed in this study was improved by 24% on the real dataset, which verified the effectiveness of this proposed algorithm.

3.2. CPSP Mining Method

Sequential pattern mining focuses on mining the correlation between sequences. Therefore, to extract more correlation knowledge between nodes in the edge-disjoint path sequences, this study further mined the edge-disjoint paths to obtain CPSPs.

Definition 1. *Contiguous path sequential pattern (CPSP): Given an edge-disjoint path* $L =< l_1, l_2, \ldots l_i >$ *and a sequence* $\beta =< \beta_1, \beta_2, \ldots \beta_m > (m \leq i)$, *$\beta$ is a contiguous path suffix in L concerning the prefix* $\alpha =< \alpha_1, \alpha_2, \ldots \alpha_k > (k \leq i, \alpha_1 = l_i, \alpha_k = l_k)$ *that satisfies the following condition:* $\beta_1 = l_{k+1}$, $\beta_2 = l_{k+2}$, *and* $\beta_m = l_{k+m}$.

CPSP refers to a frequent and positionally contiguous subsequence with support greater than a threshold in an edge-disjoint path sequence. A CPSP $S =< s_1, s_2, \ldots s_n >$ *can be represented by a binary group (α,β), where*

(1) $n = k + m$;

(2) $s_1 = \alpha_1$, $s_2 = \alpha_2$, $s_k = \alpha_k$, $s_{k+1} = \beta_1$, $s_{k+2} = \beta_2$, $s_n = \beta_m$.

CPSPs are all subsequences of frequent and position-continuous edge-disjoint paths, thus providing more reference knowledge to the process recommendation.

Most current advanced contiguous sequential pattern mining algorithms [29–31] further constrain the data based on closed pattern mining algorithms [32,33], thus mining non-complete sets of contiguous sequential patterns without the same degree of support.

Since sequences with the same support may contain different nodes, which cannot be ignored in the calculation of the matching degree between the reference model and the process model library, this study proposed a CPSP that is non-closed for process recommendation. The algorithm mines the complete contiguous non-jumping CPSP, excluding frequent 1-sequences in the extracted edge-disjoint path sequences, to effectively cooperate with the recommendation module.

Inspired by the algorithm of [34,35], the improved idea of CPSpan is as follows:

1. Recursive mining is performed only on the first item of local suffixes with support greater than a threshold, thus reducing the projection size and ensuring the continuity of the path sequence.
2. It further reduces the size of the projection database by converting the item set expansion into sequence expansion for the case where the modeler does not click on two process nodes simultaneously during the process recommendation.
3. It removes frequent 1-sequences that are not meant for process recommendation.

As there may be sequence patterns that are not strongly correlated but match well with the processes of the reference model, the lift, confidence, and support thresholds were set to 1 in this study.

The implementation of CPSPan is performed in the following steps shown in Figure 6.

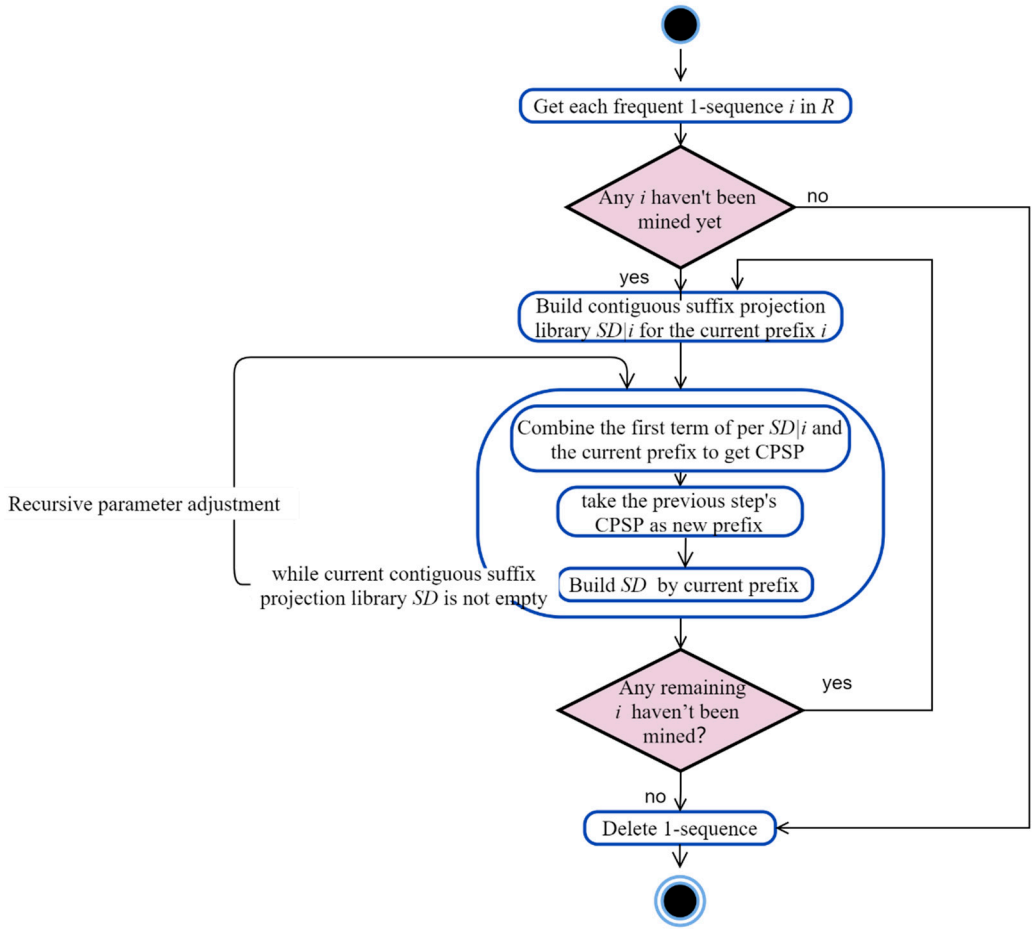

Figure 6. The CPSPan algorithm framework.

The input for Algorithm 2 consists of the edge-disjoint path table obtained from the result of Algorithm 1 and the support threshold, and the output is a set of CPSPs.

Algorithm 2. CPSPan for process recommendation

Input: *R*, the threshold of support *threshold*
Output: set of CPSPs *patterns*
Get all frequent 1-sequences *i* in *R*
Build contiguous suffix projection library *SD* | *i* for the current prefix *i*, and remove sequences that are discontiguous with *i*
FOR (each suffix *j* in *SD* | *i*) do{
Combine the first term of *j* and the current prefix as a new prefix, obtain frequent 2-sequences
Build *SD* | *j* prefixed by frequent 2-sequences in *SD* | *i*}
Repeat (3)~(5), recursively mine *patterns* of *i*, returning when the current contiguous suffix projection library *SD* is empty
Repeat (2)~(6), mining *patterns* of the remaining 1-sequences
Delete 1-sequences
Return *patterns*

Taking the two edge-disjoint path sequences <A,C,D,B> and <A,D,C,B> obtained in Section 3.1 as an example. The occurrence of each item was counted and the frequent 1-sequence <A><C><D><B> was obtained by selecting the items that met the minimum support threshold. The CPSP was then recursively mined. First, we traversed the database and obtained the continuous path suffix projection database $SD|_A$ = {<C,D,B><D,C,B>} for the prefix <A>. For each suffix in $SD|_A$, such as <C,D,B>, only the first item <C> was mined to ensure the continuity of the sequential pattern. The frequent 2-sequence <A,C> was obtained and used as a new prefix. Then, the suffix projection database $SD|_{A,C}$ = {<D,B>}, was built and further mined to obtain the frequent 3-sequence <A,C,D>. <A,C,D> was used as a new prefix, and the suffix projection database became $SD|_{A,C,D}$ = {<B>}. <A,C,D,B> was mined from it. As the suffix projection database was empty, mining for the frequent 1-sequence <A> was finished and all its CPSPs were obtained: <A><A,C><A,C,D><A,C,D,B> <A,D><A,D,C><A,D,C,B>.

Using the same approach, CPSPs were mined with other frequent 1-sequences <C>, <D>, and <B> as prefixes. The set of CPSPs was then reduced by deleting the frequent 1-sequences that were not meaningful for the recommendation.

Table 1 compares the mining results of three sequential pattern mining algorithms for <A,C,D,B> and <A,D,C,B>. The results show that Prefixspan was too redundant in its mining results due to its lack of contiguous sequence constraint, resulting in discontiguous sequential patterns that did not follow the execution sequence of path sequences in the process library model and had a low reference. CCSpan [29] is a closed contiguous mining algorithm that generates only superset sequences with different degrees of support, excluding many sequence patterns with the same degree of support but the same recommendation value. In contrast, the proposed CPSPan can mine each sequential pattern of consecutive positions in the knowledge discovery process of edge-disjoint path sequences without disrupting the execution order of the process model, providing a greater range of options and more dependable guidance for process recommendation. This advantage becomes more apparent when mining long edge-disjoint path sequences.

Table 1. Comparison of sequential pattern mining results.

Algorithm	Sequential Pattern
PrefixSpan	A:2,AC:2,AD:2,AB:2,ACD:1,ACB:2,ADC:1,ADB:2,ACDB:1,ADCB:1,B:2,C:2,CD:1,CB:2,CDB:1,D:2,DB:2,DC:1,DCB:1
CCSpan	ACDB:1,ADCB:1,CB:1,CDB:1,DB:1,DCB:1
CPSPan	AC:1,AD:1,ACD:1,ADC:1,ACDB:1,ADCB:1,CD:1,CB:1,CDB:1,DB:1,DC:1,DCB:1

After mining the CPSP, the support, confidence, and lift of the CPSP were calculated according to the following formulas:

$$Sup(S) = Sup(\alpha \to \beta) = num(\alpha \cup \beta) \tag{1}$$

$$Conf(S) = Conf(\alpha \to \beta) = P\left(\frac{\beta}{\alpha}\right) = \frac{num(\alpha \cup \beta)}{num(\alpha)} \tag{2}$$

$$Lift(S) = Lift(\alpha \to \beta) = \frac{P(\beta/\alpha)}{P(\beta)} = \frac{P(\alpha, \beta)}{P(\alpha) \times P(\beta)} = \frac{num(\alpha \cup \beta) \times num(D)}{num(\alpha) \times num(\beta)} \tag{3}$$

In process recommendation, the recommended node is always the last node of a CPSP. Therefore, when calculating the support, confidence, and lift of the CPSP $S =< s_1, s_2, \ldots s_n >$, the contiguous path suffix β is always the last node s_n and the prefix α is always $< s_1, s_2, \ldots s_{n-1} >$.

Definition 2. *Sequential rule: We define sequence rules as quadruples T = (S, Lift(S), Conf(S), Sup(S)), where S is a CPSP, and Lift(S), Conf(S), and Sup(S) are its lift, confidence, and support, respectively.*

Save all sequential rules in the sequential rule library for the subsequent process node recommendation.

3.3. Process Node Recommendation

To recommend the best next node for the current node 'A' of the reference model, first, all edge-disjoint paths of the reference model are extracted. Then, the edge-disjoint path L whose last node is 'A' is searched to find the longest common substring between L and the CPSP S of each sequence rule. The similarity of L and S is calculated using the following formula:

$$LCS\_Sim(L, S) = \frac{|LCS(L, S)| \times 2}{(|L| + |S|)} \tag{4}$$

where *LCS_Sim* denotes the similarity and *LCS(L,S)* denotes the longest common subsequence owned by L and S [26,36].

To measure the importance of nodes in different positions in edge-disjoint paths, we proposed the concept of node recommendation degree. As an example, given an edge-disjoint path <A,B>, there exist two sequential rules <A,C,1.5,1.5,2> and <B,D,1.5,1.5,2> with a similarity of 0.5. However, we considered that <B,D> has a greater impact on the recommended node than <A,C>. Thus, the node recommendation degree was introduced. For the two nodes in <A,B>, the node recommendation degree of node 'A' was 0.9, and the node recommendation degree of node 'B' was 1. Therefore, the position importance of node 'B' was considered higher than that of node 'A'.

Definition 3. *Node Recommendation Degree: Let $L =< l_1, l_2, \ldots l_n >$ be the edge-disjoint path of the reference model and S be the CPSP, then the node recommendation of $l_i (1 \le i \le n)$ is shown in Equation (5):*

$$REC\_Node(l_i, S) = \begin{cases} 0, & l_i \in S \\ 1, & l_i \in S \cap S_{index} = -1 \\ 1.1 - \left(\frac{len(L) - L_{index}}{10}\right), & l_i \in S \cap S_{index} \neq -1 \end{cases} \tag{5}$$

where *REC_Node* is the node recommendation degree, S_{index} is the index of l_i in S, L_{index} is the index of l_i in L, and *len* is the sequence length.

The process-matching degree is then calculated using the following formula:

$$P\_m(L, S) = \lambda LCS\_Sim(L, S) + \varphi REC(L, S) \tag{6}$$

where P_m is the process-matching degree; REC is the maximum value of node recommendation degrees for all nodes in the edge-disjoint path; and $\lambda + \varphi = 1$, where $\lambda = 0.5$ and $\varphi = 0.5$ were selected in this study.

As shown in Table 1, when mining the CPSP of edge-disjoint paths of length n, a great deal of knowledge of process nodes is discovered, and CPSPs of length i $(2 \leq i \leq n)$ will all be mined as process segments to be matched. Therefore, we set the following recommended node of node 'A' to the last node in the CPSP with the highest process-matching degree, thus avoiding interference with duplicate nodes in the CPSP. Finally, the recommended node was added to the execution sequence of the reference model.

When recommending k candidates for the next node for node 'A', the CPSPs in the sequential rule were first sorted according to the process-matching degree. When there were CPSPs with the same process-matching degree, the lift, confidence, and support degrees were introduced to measure the sequential rules with greater recommendation values.

The lift degree, confidence degree, and support degree serve as the basis for determining the behavioral association relationship between the prefix and the last node in a CPSP. Among them, the support degree of a CPSP indicates the frequency of simultaneous occurrence of the prefix and the last node in this sequence pattern. The confidence degree indicates the probability of the last node's simultaneous occurrence when this prefix occurs, which is more reliable than the support degree. However, since the support degree of the last node is not calculated, it may misestimate the sequential rule's importance. The lift degree best reflects the correlation between the prefix and the last node in that sequence pattern, and a higher lift indicates a higher positive correlation [37,38].

The following principles should be followed when recommending k nodes:

1. Recommend sequential rules with a high process-matching degree first.
2. Recommend sequential rules with a high lift under the same conditions first.
3. Recommend sequential rules with high confidence under the same conditions first.
4. Recommend sequential rules with high support under the same conditions first.

The process-matching degree is calculated as a weighted combination of similarity and node recommendation. When the similarity equals 1, it indicates that the edge-disjoint path is the same as the CPSP, and thus, has no recommendation significance and it will not be recommended. The recommended top k node set is the last node of the CPSP in the recommended top k sequential rule.

The implementation of the process node recommendation algorithm is performed in the following steps shown in Figure 7.

The input for Algorithm 3 consists of a reference model and a table of sequential rules, and the output is a set of recommended nodes.

Algorithm 3. Process node recommendation algorithm

Input: reference model F, sequential rules table T^*
Output: recommended node set *Node*
Initialize candidate recommendation sequential rules dictionary P
Call Algorithm1 to get edge-disjoint path R of F
FOR (each edge-disjoint path L in R where the last node is the current node) do{
FOR (each sequential rule T in T^*) do{
Sequential pattern S and its corresponding *Lift, Conf, Sup* = T[0], T[1], T[2], T[3]
Call LCS(L,S) to get the longest common subsequence owned by L and S
Calculate *LCS_Sim(L,S)* according to Equation (4)
Calculate *P_m(L,S)* according to Equations (5) and (6)
P+ = {S:[*P_m, Lift, Conf, Sup*]}}}
Get top k P sorted by *P_m*; if the same, then sorted by *Lift*; and so on
Get *Node*→the last node of S in top k P
Return *Node*

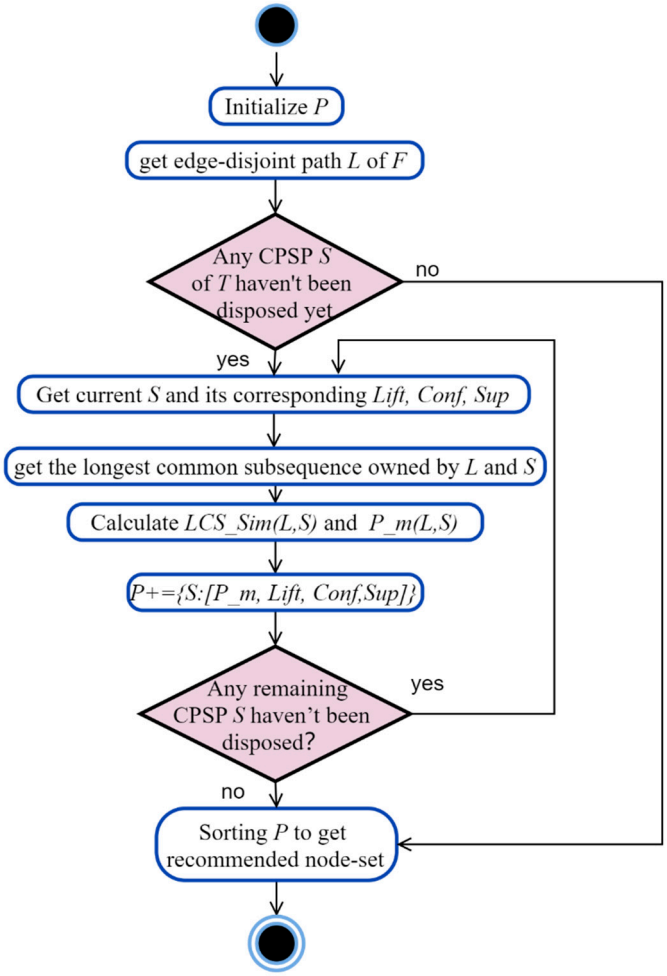

Figure 7. The process node recommendation algorithm framework.

Taking the real datasets as an example, suppose the reference model's edge-disjoint path entered by the modeler is <OpenAPI Message, Wiretap1>. There are two sequential rules in the sequential rule base, 1: <OpenAPIMessage, Wiretap1, Wiretap, 1.22, 1.0, 49> and 2: <OpenAPIMessage, Wiretap1, Terminal, 1.43, 1.0, 42>. First, the process-matching degree of both CPSPs was calculated to be 0.9. Then, the lift degree was compared, and it was found that the lift degree of sequential rule 2 was greater than that of sequential rule 1. Therefore, the recommended next node for "Wiretap1" was the last node of the CPSP in sequential rule 2.

The top five recommendation results of the edge-disjoint path in the whole real datasets are shown in Table 2.

Table 2. Recommended results of real data sets.

Process-Matching Degree	Lift	Confidence	Support	Recommended Node
0.9	1.43	1.0	42	Terminal
0.9	1.22	1.0	49	Wiretap
0.7	5.45	4.36	11	ToStringConverter
0.7	5.45	0.27	11	SAPCPEMProducer
0.7	5.45	0.18	11	Input

4. Experimental Results and Analysis

The experiments were conducted on real and randomly expanded simulation datasets, and the proposed methods' effectiveness, accuracy, and time efficiency were evaluated

through comparative experiments. The experimental environment was the coding language Python3, an Intel(R) Core(TM) i5-8300H CPU @ 2.30 GHz processor, an NVIDIA GeForce GTX 1050 Ti GPU, 8 GB RAM, Windows 10, and a 64-bit OS.

4.1. Datasets Introduction

In order to validate the effectiveness and applicability of the proposed method, experiments were conducted on both real and simulation datasets. The real dataset consisted of various business process models collected from the Modeler application of SAP, which is a provider of enterprise management solutions, where these process models contained complex selection, loop, sequential, and parallel structures. The process models in the real dataset were short in length and, therefore, suitable for evaluating the performance of the proposed approach on short-process models. The simulation dataset was obtained by randomly expanding nodes in the process model of the real dataset and was used to evaluate the performance of the proposed method on long process models. The statistical information for both datasets is provided in Table 3.

Table 3. Process dataset.

Dataset	Experimental Process Model	Process Nodes	Shortest/Longest Model Nodes	Sequential Rules	Complex Structures
Real dataset	337	276	2/8	4268	√
Simulation dataset	75	222	8/18	10,833	√

4.2. Validity Test

The EPE and CPSPan proposed in this study directly impact the effectiveness of the process recommendation. Therefore, to verify the effectiveness of the proposed algorithms, this experiment compared the VPE and Prefixspan with the methods proposed in this study to examine the enhanced effect of the proposed algorithms in the process recommendation.

The experiment consisted of three stages: preprocessing, sequential pattern mining, and node recommendation. In the VPE experiment, the VPE was applied in the preprocessing stage to obtain the vertex-disjoint path of the process model, and CPSPan was utilized in the sequential pattern mining stage. In the Prefixspan method experiments, the EPE was employed in the preprocessing stage, and Prefixspan was utilized in the sequential pattern mining stage. This study used the EPE in the preprocessing stage and CPSPan was employed in the sequential pattern mining stage. For the node recommendation stage, all three methods utilized Algorithm 3 in Section 4.3. The effectiveness of the two proposed algorithms was verified by comparing the VPE and Prefixspan methods with the method proposed in this study.

In the experiment, a five-fold cross-validation strategy was employed. The real dataset was randomly divided into five subsets of similar size, namely, *G1*, *G2*, *G3*, *G4*, and *G5*. Four subsets were selected in turn as training sets for disjoint path extraction and sequential pattern mining, and the remaining subset was used as a cross-validation test set. Finally, the five experimental results were used to evaluate the recommendation effect.

For each process model in the test set, the recommendation started from the second node because there was no recommendation basis for the starting node. The correct recommendation node was determined as the next node of the process model in the original model. If the recommendation result included the correct recommendation node, it was considered a recommended hit. The recommended hit rate was calculated using the following equation:

$$HitRate = \frac{hit\_times}{rec\_times} \tag{7}$$

where *hit_times* denotes the number of hits and *rec_times* denotes the number of recommendations.

HitRate verified the recommended effectiveness of the experiment. Suppose the recommended *HitRate* increased, and the application of the algorithm was confirmed to be effective. The effectiveness of the algorithm was examined by calculating the *HitRate* for different numbers of recommendations (i.e., top k).

The comparison results in Figure 8 show that the *HitRate* for all three methods increased as the number of recommended results (*k*) increased. When *k* was greater than 3, the *HitRate* of the proposed method exceeded 95%.

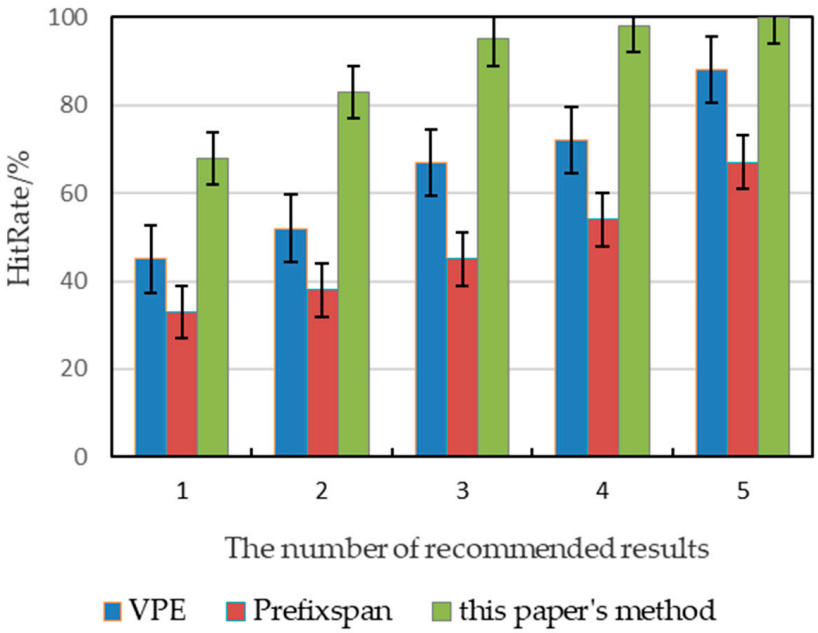

Figure 8. Effectiveness test of the proposed algorithm for the process recommendation.

Specifically, for $k = 1$, the *HitRate* of the VPE method was 45%, while it was 68% for the proposed method; for $k = 5$, the *HitRate* of the VPE method was 88%, while the proposed method reached 100%, demonstrating the effectiveness of the proposed EPE. These results demonstrate that the EPE outperformed the VPE regarding process recommendation, as the former could extract a larger set of disjoint paths, which could better capture the behavioral semantics of a process. In contrast, the vertex-disjoint paths extracted by VPE were a subset of the edge-disjoint paths, and there were an infinite number of vertex-disjoint paths under the 'Loop' structures.

In the Prefixspan method experiments, the *HitRate* was 33% for $k = 1$, which was 35% lower than for the proposed method; for $k = 5$, it was 37% lower than the *HitRate* of the proposed method, confirming the effectiveness of CPSPan in the proposed method.

Furthermore, when comparing the VPE method experiments with the Prefixspan method experiments, it was evident that CPSPan was more effective than the EPE regarding process recommendation.

4.3. Accuracy Test

In order to demonstrate the accuracy of the proposed method more intuitively, this study compared it with the maximum common sub-graph and minimum common graph method (MCS) proposed in [4], the matrix distance similarity method (MDS) proposed in [12], the target profile similarity method (TPS) proposed in [25], and the vertex-disjoint path similarity method (VPS) proposed in [26].

When the number of fixed recommendation nodes was five, ten reference models with *n* nodes were randomly extracted from the process library separately (where *n* ranged from 1 to 6 in the real dataset and from 7 to 13 in the simulation dataset), and the correct recommendation rate accuracy was calculated using each of the five methods. To more

accurately examine the recommendation accuracy, accuracy was achieved by applying different weight signatures to different bit orders of the correct recommendation node in the recommendation results.

The accuracy was defined as follows:

1. When the correct recommendation node appeared in the first position of the recommendation node set (i.e., the top 1 recommendation node), the accuracy was 100%.
2. When the correct recommendation node was in the second position, the accuracy was 80%, and so on.
3. If the correct recommendation node was in the fifth position of the recommendation node set, the accuracy was 20%.
4. If the recommended node set did not include the correct recommended node, the accuracy rate was 0.

Based on the real dataset, this study evaluated the recommended accuracy of the short-process models, divided into two dimensions: the reference model and the process library model. The process library model of the real dataset was short. As shown in Figure 9a, the proposed method generally had a higher accuracy than the other four methods. Meanwhile, when the number of reference model nodes was only 1 or 2, the accuracy of this method remained stable at above 70%. This finding demonstrates that the modeling of a shorter reference model during process recommendation could provide a useful reference for pre-modeling.

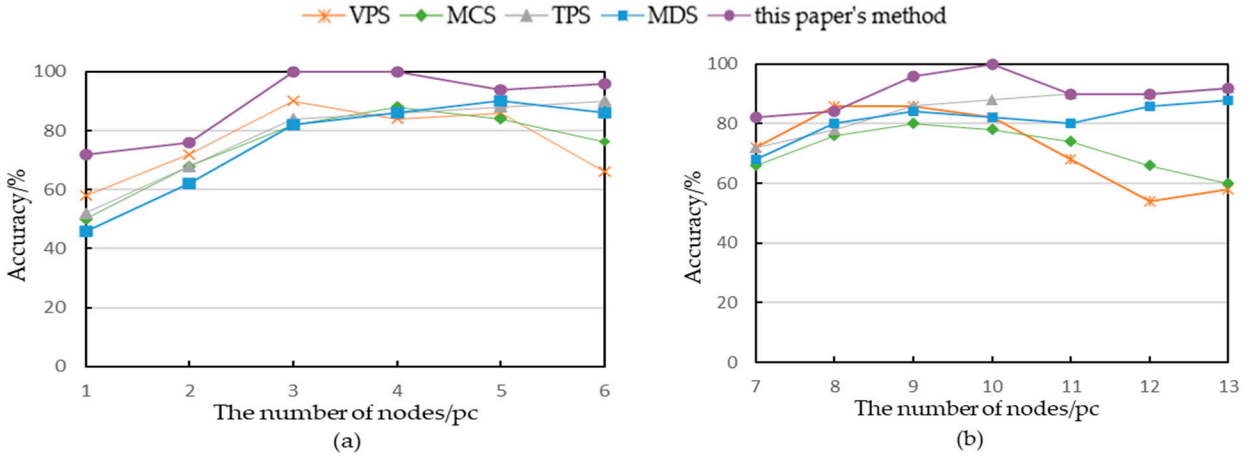

Figure 9. Accuracy comparison of five methods: (**a**) real dataset; (**b**) simulation dataset.

As shown in Figure 9b, the method proposed in this study also performed better in terms of *Accuracy* when the input reference model was longer, or the process library model was longer.

The proposed method's average *Accuracy* was 89.67% for 60 reference models with 6 different numbers of nodes in the real dataset, which was higher than the VPS (76%), MCS (74.67%), TPS (78%), and MDS (75.33%). Similarly, for 70 reference models with 7 different numbers of nodes in the simulated dataset, the average accuracy was 90.57%, which was higher than the VPS (72.29%), MCS (71.43%), TPS (85.14%), and MDS (81.14%). These results demonstrate the effectiveness of the proposed method in enhancing the accuracy of short-process models and its superiority over the compared methods in terms of accuracy.

4.4. Time Efficiency Test

From the user's perspective, the offline processing phase does not affect their experience, and thus, only the online recommendation phase was tested for time efficiency. In the real dataset, 50, 100, 150, 200, 250, and 300 process models were randomly selected as new process library models. In the simulated dataset, 15, 30, 45, 60, and 75 process models were randomly selected as new process library models to observe the changes in data.

From Figure 10, it can be observed that the response time of the five methods was positively correlated with the number of models in the process library. The proposed method exhibited a higher time efficiency compared with the TPS, MCS, and MDS, as the MCS is calculated based on graph structure and has a complexity of O(n), where *n* is the number of subgraphs; TPS and MDS have a maximum complexity of $O(n^3)$, where *n* is the number of process library models, and are calculated based on a matrix. On the other hand, the proposed method has a complexity of O(m*n), where *m* is the number of edge-disjoint paths of the reference model and *n* is the number of CPSPs of the process library model, and is based on an array, making it more advantageous in terms of time efficiency.

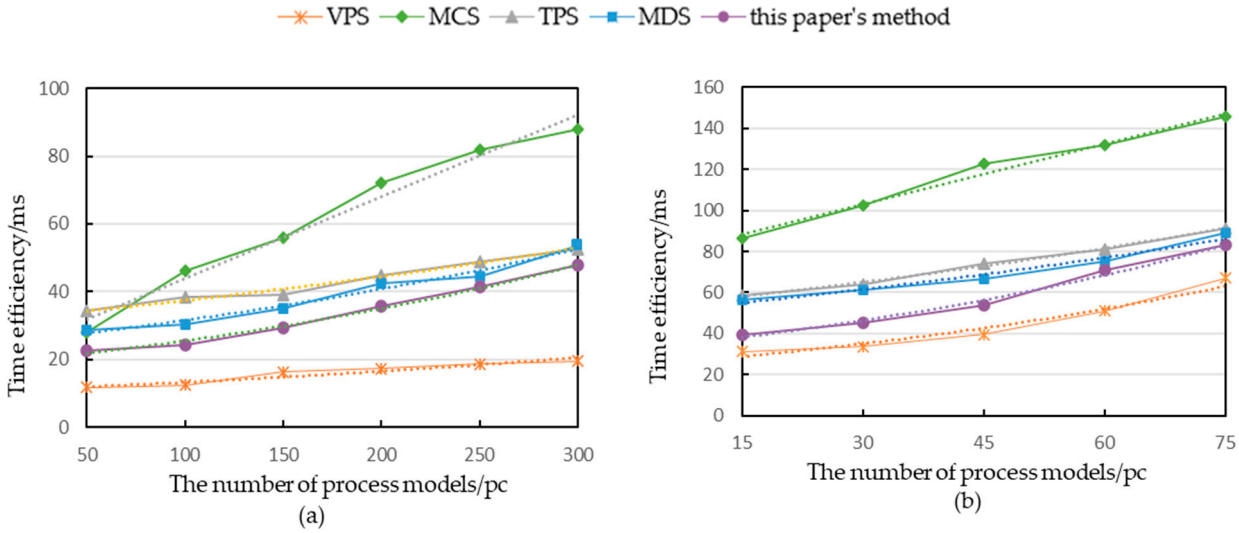

Figure 10. Time comparison of five methods: (**a**) real dataset; (**b**) simulation dataset.

Furthermore, the time efficiency of this method is only slightly lower than that of the VPS, which has a complexity of O(m*n), where *m* is the number of vertex-disjoint paths of the reference model and *n* is the number of vertex-disjoint paths of the process library model because the number of vertex-disjoint paths extracted using the VPS is smaller than the number of CPSPs mined using this method, resulting in fewer traversals. However, the experimental results show that the difference between the time spent using this method and the VPS was only a few milliseconds, which had no significant impact on the user experience. Therefore, considering both the accuracy and time efficiency, this method outperformed the other methods.

5. Visual Prototyping System

To demonstrate the real-life application of the proposed method, we designed and proposed a visual prototyping system that automatically recommends the next node based on the process node entered by the modeler through the process recommendation method that combines disjoint path and sequential patterns, thus better assisting the modeler in rapid modeling. As illustrated in Figure 11, the system comprises two modules: the upper and lower modules.

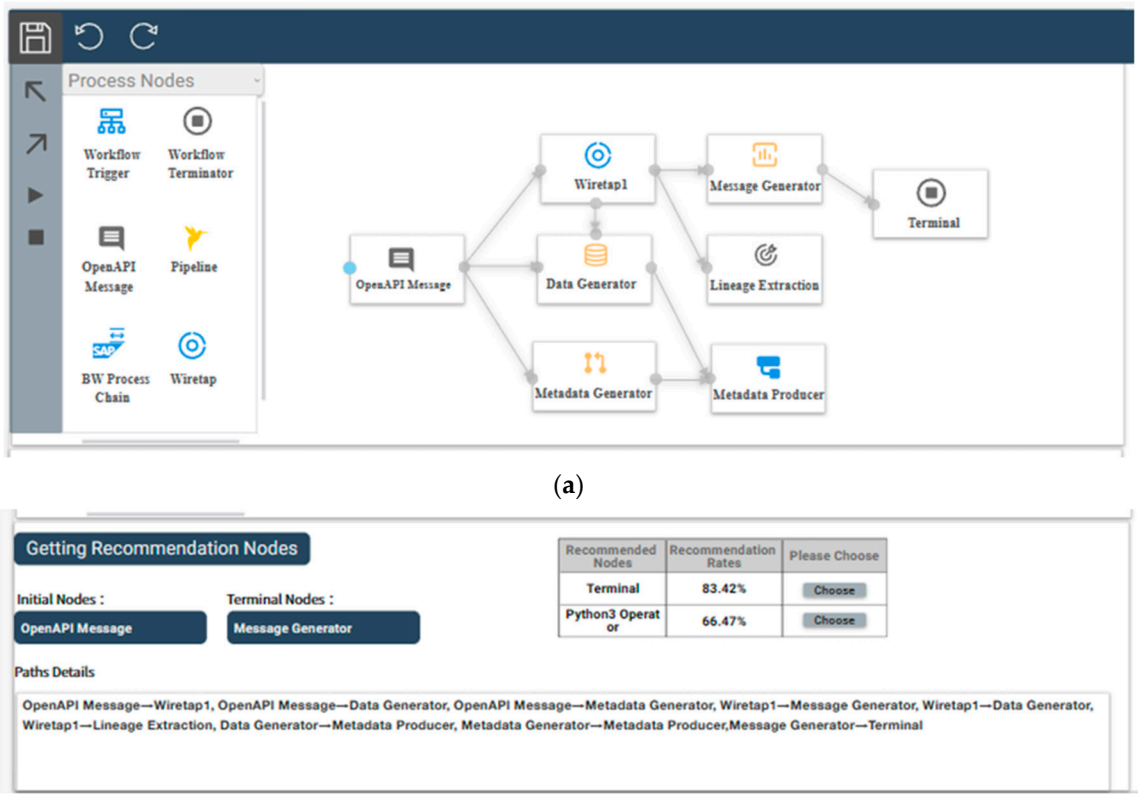

Figure 11. Visual prototyping system: (**a**) process builder module; (**b**) recommendation and data interaction module.

1. The upper module, shown in Figure 11a, is a process builder that offers the following features:
   (i) Enabling users to add process nodes and build process diagrams freely;
   (ii) Enabling users to recommend, add, edit, delete, and drag and drop each node while connecting them;
   (iii) Enabling users to undo modifications and reverse undo modifications when accidental deletions or other errors occur;
   (iv) Enabling users to save the process with one click once it is built, without any other unnecessary operations.
2. The lower module, shown in Figure 11b, is the recommendation and data interaction module.

After the user finishes building the current reference model and selects the next node, clicking on the "Get recommendation node" button prompts the program to parse the reference model built on the process builder, analyze and match the data in the background, and display the corresponding results in the recommendation module, with support for limiting the maximum number of recommendation nodes displayed.

When the user selects a recommendation node as a hit, the process builder automatically builds the next process node based on the user's selected results without requiring the user to build it manually.

As shown in Figure 10b, when ''Message Generator" is taken as the termination node, the system recommends "Terminal" and "Python3 Operator" as the recommendation nodes after the matching is completed and displayed on the bottom right side. The system will automatically build the "Terminal" node after clicking the "choose" button.

6. Discussion and Limitations

BPM plays a crucial role in the digitalization and informatization processes of enterprises. The ultimate goal of BPM is to achieve the optimal realization of business processes. However, several factors can affect the performance of these processes. Previous research predominantly focused on optimizing business processes in terms of process efficiency, resource utilization, customer experience, and cost reduction, yielding significant results. Process recommendation technology can also be viewed as a means of optimizing the customer experience to a certain extent, involving the consideration of factors such as user modeling requirements, offline and online processing techniques, and time constraints. By analyzing the model structure of the business process and current business requirements, process recommendation technology can suggest the most suitable process execution path, thereby enhancing the efficiency and accuracy of the business process.

Our contribution to the academic and practical community is a novel process recommendation technique that combines disjoint path and sequential patterns. This technique automatically recommends the next process node during the process modeling based on the current process node entered by the modeler.

In order to address the issues of low accuracy and neglect of short-process models, which were the main focus of this study, we conducted experiments on both real and simulated datasets to evaluate the effectiveness of the proposed method. The "Experimental Results and Analysis" section demonstrates that our process recommendation technique achieved a higher hit rate and accuracy and met the time response requirements. Specifically, the comparison method MCS proposed in [4] uses graph structures for process recommendation, but the proposed graph-matching method can have a large variance in results due to overly complex graphs. The MDS proposed in [12] measures the similarity between processes by distance, but the significance of the "distance" value does not reflect the similarity between processes that contain a loop structure well. The TPS proposed in [25] requires the modeler to develop the target profile for each reference model being modeled, which has high professional requirements for the modeler and is challenging to execute. The VPS proposed in [26] uses the vertex-disjoint paths to express behavioral semantics, which cannot fully reflect the absence of order among the child nodes of the parallel and selective structure of the process. Moreover, this method ignores the fact that there may be multiple paths to complete the same business goal. Furthermore, [25,26] ignored the existence of duplicate nodes in the process model.

In summary, the method proposed in this study has the following advantages:

1. This study proposed a method that focuses on the behavioral semantics of the process and uses edge-disjoint path parsing, which offers a more significant efficiency advantage and smaller result differences compared with graph parsing.
2. The proposed method uses the EPE to more accurately represent the behavioral semantics of parallel and selective structures.
3. The method also introduces a node recommendation degree to measure the importance of different node positions in the paths and selects the last node of the CPSP as the recommendation result, thus avoiding the influence of duplicate nodes on the result.
4. Unlike previous methods (such as [12] and [26]) that only consider the frequency of the node appearance in the process model library, this study introduced lift and confidence measures to evaluate nodes that are more closely related to the recommended fragment, thereby improving the accuracy.
5. CPSPan was utilized to perform further knowledge discovery of edge-disjoint paths and obtain a wider range of options and more reliable guidance for process recommendations.

In addition, we developed a visual prototype system that includes features such as recommendation, addition, deletion, editing, and dragging of nodes. When a user selects a recommended node displayed in the system as the hit, the system automatically constructs the next process node based on the user's selection, eliminating the need for manual

construction. This demonstrates the effectiveness of the proposed method in supporting practical application scenarios.

Our conclusion is that in the new era of digitization and informatization, efficient and accurate process modeling technology will become the core value of BPM applications. By extracting the edge-disjoint paths of the process through the EPE and further mining the behavioral knowledge of the process using CPSPan, recommending process nodes based on their process-matching, confidence, lift, and support degrees can effectively promote efficient and accurate modeling.

However, despite our efforts, our technology still has certain limitations:

1. Our approach recommends the next best node, which provides the greatest flexibility for the modeler to choose the next process activity. However, recommending process segments or complete process models can in some way facilitate the modeler to model more easily and quickly.
2. Our method is based on the complex structures of process library models for process recommendation. Although event logs do not contain information such as complex structures, event logs contain a large amount of abstract information such as the frequency of process branch directions, which can also serve as a powerful basis for process recommendation.
3. From a business process optimization perspective, our approach meets the time requirements and provides highly accurate recommended results. However, it only focuses on the next best node in the business process modeling design, taking into account both the process model and the business requirements, without considering the reliability of the services/components involved in each process node during the modeling process. For instance, we did not consider factors such as process execution time, task execution time, the number of exception events, timeouts, failures, and retries associated with each service/component. However, as this study aimed to address the problem of accuracy in short-process models, there are few business process datasets that contain both short-process models and reliability metrics in the current business process domain.

7. Conclusions

Starting from the behavioral semantics of business processes, this study proposed a novel process recommendation method that integrates disjoint paths and sequential patterns. The EPE and CPSPan were introduced to calculate the process-matching degree using the edge-disjoint paths of the reference model and the CPSP of the process library model. Lift, confidence, and support were combined to provide more of a reference basis for the recommendation results. Comparison experiments with existing methods demonstrated that the proposed method effectively improved the process recommendation effect, performed better in terms of accuracy, and could meet the demand for time efficiency. In addition, this study presented a visual prototyping system that demonstrated the effectiveness of the methodology proposed in this study for supporting practical applications. In order to further enhance the contribution of the proposed method to both the research and industrial communities, the following future work is proposed: (1) To recommend a required number of process segments or the entire process model, a possible enhancement would be to include the currently recommended node as the next modeling node by invoking the process recommendation algorithm recursively until the desired number of process segments or the whole process model is recommended. (2) To better assist modelers in modeling and optimizing the execution performance of business processes, real business process event logs containing reliability metrics, node execution frequency, etc., can be collected and a process recommendation method combining the log data and model structure can be designed to better predict node execution times and execution outcomes.

Author Contributions: Conceptualization, D.H. and C.W.; methodology, D.H.; validation, D.H. and T.S.: writing—original draft preparation, D.H.; visualization, D.H. and T.S.; writing—review and editing C.W., G.B., and B.S.; funding acquisition B.S. and G.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China (No.62072363) and Shaanxi Provincial Natural Science Foundation (S2019-JC-YB-1191).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Yu, F.; Guo, L.P.; Zhang, L. Process Modeling Leveraged by Workflow Fragments and Logs Analysis. *J. Chin. Comput. Syst.* **2017**, *38*, 664–670.
2. Cai, Q.M.; Zhang, L.; Xu, C.H. Research of Process Similarity Based on Single-layer Neural Network. *Comput. Eng. Appl.* **2022**, *58*, 295–302.
3. Li, Y.; Cao, B.; Xu, L.; Yin, J.; Deng, S.; Yin, Y.; Wu, Z. An Efficient Recommendation Method for Improving Business Process Modeling. *IEEE Trans. Ind. Inform.* **2013**, *10*, 502–513. [CrossRef]
4. Wang, D.G.; Deng, S.G.; Cao, B. Efficient and reliable process recommendation system-JTangWFR. *Comput. Integr. Manuf. Syst.* **2013**, *19*, 1883–1890.
5. Deng, S.; Wang, D.; Li, Y.; Cao, B.; Yin, J.; Wu, Z.; Zhou, M. A Recommendation System to Facilitate Business Process Modeling. *IEEE Trans. Cybern.* **2017**, *47*, 1380–1394. [CrossRef]
6. Bobek, S.; Baran, M.; Kluza, K. Application of bayesian networks to recommendations in business process modeling. In Proceedings of the CUER Whokshop, Online, 25–27 November 2013.
7. Cao, B.; Yin, J.; Deng, S.; Wang, H.; Zaki, M.J. Graph-based workflow recommendation: On improving business process modeling. In Proceedings of the 21st ACM International Conference on Information and Knowledge Management, New York, NY, USA, 29 October–2 November 2012.
8. Fradi, A.; Louhichi, B.; Ali, M.; Eynard, B. 3D Object Retrieval Based on Similarity Calculation in 3D Computer Aided Design Systems. In Proceedings of the 14th IEEE/ACS International Conference on Computer Systems and Applications (AICCSA), Hammamet, Tunisia, 30 October–3 November 2017; IEEE: Piscataway, NJ, USA, 1963; pp. 160–165.
9. Deb, D.; Ghose, A.; Chaki, N.A. Framework for business process modeling by QoS-based pruning. *Innov. Syst. Softw. Eng.* **2017**, *13*, 1–8. [CrossRef]
10. Matsuda, T.; Kitajo, K.; Yamaguchi, Y.; Yamaguchi, Y.; Komaki, F. A point process modeling approach for investigating the effect of online brain activity on perceptual switching. *NeuroImage* **2017**, *152*, 50–59. [CrossRef]
11. Alfegadhi, S. Business process modeling:blueprinting. *Int. J. Comput. Sci. Inf. Secur.* **2017**, *15*, 286–291.
12. Hu, H.; Qiao, J.; Hu, H.Y. Process recommendation based on probabilistic and time Petri net. *Appl. Res. Comput.* **2018**, *35*, 62–68.
13. Wang, H.; Wen, L.; Lin, L.; Wang, J. RLRecommender: A Representation-Learning-Based Recommendation Method for Business Process Modeling. In Proceedings of the International Conference on Service Oriented Computing, Online, 7 November 2018.
14. Cao, B.; An, W.S.; Wang, J.X.; Fan, J. A difference detection algorithm for process models based on process structure tree. *Acta Electron. Sin.* **2018**, *46*, 97–105.
15. Russell, N.; Van Der Aalst, W.M.; Ter Hofstede, A.H. *Workflow Patterns: The Definitive Guide*; MIT Press: Cambridge, MA, USA, 2016.
16. Gavvala, S.; Jatoth, V.C.; Gangadharan, G. QoS-aware cloud service composition using eagle strategy. *Future Gener. Compute. Syst.* **2019**, *90*, 273–290. [CrossRef]
17. Zhu, Y.; Hu, Z.; He, Z. Edge Intelligence Service Orchestration with Process Mining. *Appl. Sci.* **2022**, *12*, 10436. [CrossRef]
18. Represa, J.G.; Larrinaga, F.; Varga, P.; Ochoa, W.; Perez, A.; Kozma, D.; Delsing, J. Investigation of Microservice-Based Workflow Management Solutions for Industrial Automation. *Appl. Sci.* **2023**, *13*, 1835. [CrossRef]
19. Wang, Q.; Shao, C.; Fang, X.; Zhang, H. Business process recommendation method based on cost constraints. *Connect. Sci.* **2022**, *34*, 2520–2537. [CrossRef]
20. Luo, W.; Peng, Z.; Deng, A.; Bi, X. Toward business process recommendation-based collaborative filtering. *Int. J. Internet Manuf. Serv.* **2019**, *6*, 389–402. [CrossRef]
21. Ye, Y.M.; Yin, J.W.; Cao, B. Process warping matrix based business process recommendation technique. *Comput. Integr. Manuf. Syst.* **2013**, *19*, 1868–1875.
22. Fellmann, M.; Metzger, D.; Jannaber, S.; Zarvic, N.; Thomas, O. Process modeling recommender systems. *Bus. Inf. Syst. Eng.* **2018**, *60*, 21–38. [CrossRef]
23. Wang, J.; Tan, D.; Cao, B.; Fan, J. Samundra Deep. Independent path-based process recommendation algorithm for improving biomedical process modelling. *Electron. Lett.* **2020**, *56*, 531–533. [CrossRef]

24. Jalali, A.; Maria, F.; Reijers, H.A. A hybrid approach for aspect-oriented business process modeling. *J. Softw. Evol. Process* **2018**, *30*, 1–21. [CrossRef]
25. Li, D.Q.; Fang, X.W. Process modeling recommendation method based on behavioral profile definition target rules. *J. Comput. Appl.* **2022**, *42*, 223–229.
26. Gui, S.C.; Wang, J.X.; Hong, F.; Winckler, M.; Trætteberg, H. Behavior- based automated process modeling method using recommendation. *Comput. Integr. Manuf. Syst.* **2020**, *26*, 1500–1509.
27. Sun, Z.S.; Xie, Z.; Chen, Z. Algorithm design of independent path problem. *Comput. Eng.* **2013**, *39*, 148–152.
28. Joseph, M.; Roby, T. Toggling independent sets of a path graph. *Electron. J. Comb.* **2018**, *25*, 1–18. [CrossRef] [PubMed]
29. Zhang, J.; Wang, Y.; Yang, D.Y. CCSpan: Mining closed contiguous sequential patterns. *Knowl. Based Syst.* **2015**, *89*, 1–13. [CrossRef]
30. Abboud, Y.; Boyer, A.; Brun, A. CCPM: A Scalable and Noise-Resistant Closed Contiguous Sequential Patterns Mining Algorithm. In Proceedings of the 13th International Conference on Machine Learning and Data Mining, New York, NY, USA, 15–20 July 2017; pp. 147–162.
31. Abboud, Y.; Boyer, A.; Brun, A. C3Ro: An Efficient Mining Algorithm of Extended-Closed Contiguous Robust Sequential Patterns in Noisy Data. *Expert Syst. Appl.* **2019**, *131*, 172–189. [CrossRef]
32. Wang, J.; Han, J. BIDE: Efficient Mining of Frequent Closed Sequences. In Proceedings of the 20th International Conference on Data Engineering, New York, NY, USA, 14–17 December 2004.
33. Yan, X.; Han, J.; Ramin, A. CloSpan: Mining: Closed Sequential Patterns in Large Datasets. In Proceedings of the 2003 SIAM International Conference on Data Mining, Minneapolis, MS, USA, 27–29 April 2003; pp. 176–184.
34. Pei, J.; Han, J.; Mortazavi, A.; Pinto, H.; Chen, Q.M.; Dayal, U.; Hsu, M.C. *PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth. Proceedings of the 17th International Conference on Data Engineering, Heidelberg, Germany, 2–6 April 2001*; IEEE: Piscataway, NJ, USA, 1963; pp. 215–224.
35. Peng, M.J. *A Study on Parallel Minging of Contiguous Sequential Pattern*; Hubei University: Wuhan, China, 2015.
36. Hirschberg, D.S. Algorithms for the Longest Common Subsequence Problem. *J. ACM JACM* **1977**, *24*, 664–675. [CrossRef]
37. Agrawal, R.; Imielinski, T.; Swami, A. Mining association rules between sets of items in large databases. In Proceedings of the ACM SIGMOD Conference on Management of Data, Washington, DC, USA, 26–28 May 1993; pp. 207–216.
38. Brin, S.; Motwani, R.; Silverstein, C. Beyond market baskets: Generalizing association rules to correlations. In Proceedings of the 1997 ACM SIGMOD international conference on Management of data, Tucson, AZ, USA, 13–15 May 1997; pp. 265–276.