

RESEARCH ARTICLE

Discovering Structural Formations of Multi-Level and Compound Control-Flow Gateways From Process Logs

THANH-HAI NGUYEN¹ AND KWANGHOON PIO KIM², (Member, IEEE)

¹Department of Computer Science, Graduate School, Kyonggi University, Suwon, Gyeonggi 16227, Republic of Korea

²Division of AI Computer Science and Engineering, Kyonggi University, Suwon, Gyeonggi 16227, Republic of Korea

Corresponding author: Kwanghoon Pio Kim (kwang@kgu.ac.kr)

This work was supported by the National Research Foundation of Korea (NRF) Grant funded by the Korean Government, Ministry of Science and ICT (MSIT), Republic of Korea, under Grant NRF-2022R1A2C2093002; additionally, this work was supported by the Kyonggi University's Graduate Research Assistantship Program awarded in 2021.

ABSTRACT In the field of process mining, the discovery of process patterns from event logs remains a challenging topic and has always interested many researchers. Exploiting the process model remains a major challenge and is highly dependent on event log characteristics, such as dataset size, the completeness of the event trace, and especially the complexity of the process model structure. The ρ (rho)-algorithm is a powerful process mining algorithm that can mine all the structured information control net (SICN), such as sequential, selective, parallel, and iterative. The ρ -algorithm also faces challenges when mining complex process patterns, consisting of many SICN primitive patterns combined at multiple levels that satisfy properties matched pairing and the proper nesting. This paper presents a new approach, defines the formalization of the multi-level and compound control-flow gateways (ML-CCFG) and a series of rules for decision-making these gateways, and proposes an algorithm extending from ρ -algorithm (we named it ρ MC-algorithm), that can efficiently discover the SICN-oriented process model containing ML-CCFG from process instances event log. We developed an implementation system ρ MC-algorithm to discover and visualize in a graphical form SICN-oriented process models from the datasets of the IEEE XES-formatted process enactment event logs. We also perform a series of experimental analyses using the implemented system on the process enactment event log datasets to verify the proposed algorithm.

INDEX TERMS Process mining algorithm, structured information control nets (SICN), workflow and business process model, multi-level and compound control-flow gateways.

I. INTRODUCTION

Process mining allows us to know and better understand workflow processes from historical records [1]. Using process mining techniques, we capture more information about what happens in the organization to analyze and visualize the entire system, comparing it with the predefined workflow model. Therefore, we can make appropriate adjustments to improve the efficiency of the business process using the results from the process mining. In the over twenty-year

history of process mining, uncovering process patterns from event logs has always been challenging.

Since 2004, many algorithms have been proposed after Professor W.M.P. van der Aalst et al. proposed the α (alpha) algorithm [2]. We can talk about the typical process mining algorithms include those the α family of algorithms with extended versions α^+ [3]; α^{++} [4]; $\alpha^\#$ [5], σ (sigma)-algorithm [6], ρ (rho)-algorithm [7], ILP mining [8], Fuzzy mining [9], Heuristic miner [10] and Inductive visual miner (IDM) [11]. Recently, modern systems for managing processes have encountered various challenges related to their scalability, including sustainability issues such as process lifecycle management issues and fidelity issues such

The associate editor coordinating the review of this manuscript and approving it for publication was Mansoor Ahmed¹.

as process redesign and refactoring, and process performance measurement issues [12], as well as addressing problems and engaging in predictive capacity planning [13]. In this dynamic landscape, the evolution of process discovery algorithms remains essential to continually adapt to changing data and the growing intricacy and diversity of workflows. Currently, most algorithms can discover structures common in workflow models, such as sequential, selection, parallelism, and somewhat loop structures [14]. However, process model mining highly depends on event log characteristics, such as the data set's size, the process model structure's complexity, and the event trace's completeness [15]. The research in [16] attempted to propose a method to discover hierarchical complex models into multiple sub-process levels and achieved some positive results. The ρ -algorithm is a powerful process mining algorithm to discover SICN-oriented process model from process event log dataset [7]. The ρ -algorithm also faces challenges in discovering complicated processes, consisting of many primitive patterns combined at multiple levels that satisfy properties matched pairing and the proper nesting follow the definition of the SICN model. This paper presents an approach that can efficiently discover complex SICN patterns extended from the ρ -algorithm (we call it ρ MC) to adapt to the evolution of diverse, complex, and large-scale business processes. We also developed a tool to extract the process from the event log based on the recommended algorithm and perform experimental verification on real data sets. The contributions of this paper are threefold:

- 1) Firstly, we formally generalize the six types of ML-CCFG in the SICN-oriented process model.
- 2) Second, we extend the processing rules for the ρ -algorithm to accurately discover the SICN-oriented process model containing ML-CCFG from the process enactment event log.
- 3) Finally, we conduct experimental verification and validation of the ρ MC-algorithm with expanded rules on the event log dataset to discover the SICN-oriented process model contains ML-CCFG. We concurrently compare the results of discovering the process model by our algorithm to some other algorithms to show the effectiveness of the proposed solution.

The rest of the paper is organized as follows: Section II discusses the work and the background involved, such as the SICN-oriented process model, the ρ -algorithm, trace, events, and temporal work cases. We describe the concept of ML-CCFG, which mainly motivated us to devise the ρ MC-algorithm in Section III. Section IV describes the experimental verification and validation of the ρ MC-algorithm on discovering the SICN-oriented process model on the event log dataset. Finally, we conclude this work and discuss future research directions in section V.

II. BACKGROUND

A. PROCESS, EVENT LOG, TRACE, EVENT AND ACTIVITY

In the context of process mining, a process is a series of steps or stages from start to end to complete a particular

task. The process enactment event log is an activity log created during the operation of a process-aware information system [17]. Each event in an event log refers to an activity possibly executed by a resource (performer) for a specific process instance (trace) at a specific time [18]. Each event has many attributes, e.g., transaction information, costs, customer, location, unit, etc. In this paper, we focus on control-flow perspective to discover the sequence of activities in a business process [19]. Thus, we just consider the activity and the ordering of events within the trace. Since 2016, event log files have been uniformly stored in a standard format called IEEE XES [20]. Table 1 shows an example of information commonly found in the event log cutting out from the Review Example Large even log [21], where each recorded activity is related to the execution of a process instance.

TABLE 1. A cutting out of review example large event log.

Trace ID	Activity	Resource	Timestamp
102	invite reviewers	Mike	2007-04-17 01:00
103	invite reviewers	Mike	2007-04-17 01:00
102	get review 3	Sara	2007-04-21 01:00
103	get review 1	Pam	2007-04-24 01:00
...

B. STRUCTURE INFORMATION CONTROL NETS

Our research based on the structured information control networks (SICN) to represent business process models (workflow model) [22]. We rewrite the definitions as follows:

Definition 1 (SICN): A basic SICN is 8-tuple $\Gamma = (\delta, \rho, \lambda, \varepsilon, \pi, \kappa, I, O)$ over a set of \mathbf{A} activities (including a set of group activities), a set \mathbf{T} of transition conditions, a set \mathbf{D} of data repositories, a set of \mathbf{G} of invoked application programs, a set \mathbf{R} of roles, and a set \mathbf{P} of performers (including a set of performer groups), where

- \mathbf{I} is a finite set of initial input repositories, which is assumed to be loaded from some external workflow before execution;
- \mathbf{O} is a finite set of final output repositories, which is assumed to be transferred to some external workflow after execution;
- $\delta = \delta_i \cup \delta_o$ where, $\delta_o: \mathbf{A} \rightarrow \wp(\mathbf{A})^1$ is a multi-valued function mapping an activity to its sets of (immediate) successors, and $\delta_i: \mathbf{A} \rightarrow \wp(\mathbf{A})$ is a multi-valued function mapping an activity to its sets of (immediate) predecessors;
- $\rho = \rho_i \cup \rho_o$ where, $\rho_o: \mathbf{A} \rightarrow \wp(\mathbf{D})$ is a multi-valued function mapping an activity to its set of output repositories, and $\rho_i: \mathbf{A} \rightarrow \wp(\mathbf{D})$ is a multi-valued function mapping an activity to its set of input repositories;
- $\lambda = \lambda_a \cup \lambda_g$ where, $\lambda_a: \mathbf{A} \rightarrow \wp(\mathbf{G})$ is a single-valued function mapping a task-type activity to its invoked application program, and $\lambda_g: \mathbf{G} \rightarrow \wp(\mathbf{A})$ is a

¹ $\wp(\mathbf{A})$ is the powerset of \mathbf{A} .

multi-valued function mapping an invoked application program to its set of associated task-type activities;

- $\varepsilon = \varepsilon_a \cup \varepsilon_r$ where, $\lambda_a: \mathbf{A} \rightarrow \wp(\mathbf{R})$ is a single-valued function mapping a task-type activity to a role, and $\lambda_r: \mathbf{R} \rightarrow \wp(\mathbf{A})$ is a multi-valued function mapping a role to its sets of associated task-type activities;
- $\pi = \pi_r \cup \pi_p$ where, $\pi_r: \mathbf{R} \rightarrow \wp(\mathbf{P})$ is a multi-valued function mapping a role to its sets of associated performers, and $\pi_p: \mathbf{P} \rightarrow \wp(\mathbf{R})$ is a multi-valued function mapping a performer to its sets of associated roles;
- $\kappa = \kappa_i \cup \kappa_o$ where, $\kappa_i: (\mathbf{A} \times \mathbf{A}) \rightarrow \wp(\mathbf{T})$ is a multi-valued function mapping a set of control-transition conditions, \mathbf{T} , on directed arcs (ordered pairs), $(\delta_i(\alpha); \alpha \in \mathbf{A})$, from $\delta_i(\alpha)$ to α ; and $\kappa_o: (\mathbf{A} \times \mathbf{A}) \rightarrow \wp(\mathbf{T})$ is a multi-valued function mapping a set of control-transition conditions, \mathbf{T} , on directed arcs (ordered pairs), $(\alpha \in \mathbf{A}, \delta_o(\alpha))$, from α to $\delta_o(\alpha)$.

1) STARTING AND TERMINATING NODES

Additionally, the execution of a workflow model commences by a single χ transition-condition. So, we always assume without loss of generality that there is a single starting node(α_I). At the commencement, it is assumed that all input repositories in the set \mathbf{I} have been initialized with data by the external system:

$$\alpha_I \in A | \delta_i(\alpha_I) = \{\emptyset\} \wedge \kappa_o(\alpha_I) = \{\{\chi\}\}.$$

The execution is terminated with any one λ outgoing transition-condition. Also we assume without loss of generality that there is a single terminating event node (α_F) that has no to-node without any outgoing arc. A set of the output repositories \mathbf{O} is a group of data holders that is transferred to the external workflow model after termination:

$$\alpha_F \in A | \delta_o(\alpha_F) = \{\emptyset\} \wedge \kappa_i(\alpha_F) = \{\{\lambda\}\}.$$

2) CONTROL FLOW: TEMPORAL ORDERING OF ACTIVITIES

Given a formal definition, the forward temporal ordering of activities (event-type, gateway-type, task-type activities) in a workflow model can be formally represented as follows: For any type of activity α , in general:

$$\delta(\alpha) = \{ \begin{aligned} &\{\beta_{11}, \beta_{12}, \dots, \beta_{1m(1)}\}, \\ &\{\beta_{21}, \beta_{22}, \dots, \beta_{2m(2)}\}, \\ &\dots, \\ &\{\beta_{n1}, \beta_{n2}, \dots, \beta_{nm(n)}\}, \end{aligned} \}$$

means that upon the completion of activity α , a transition that simultaneously initiates all of the activities β_{i1} through $\beta_{im(i)}$ occurs, which is called a parallel transition; otherwise only one value out of $i(1 \leq i \leq n)$ is selected as the result of a decision made within activity α , which is called a decision transition. Note that if $n = 1 \wedge m = 1$, then neither decision nor parallel

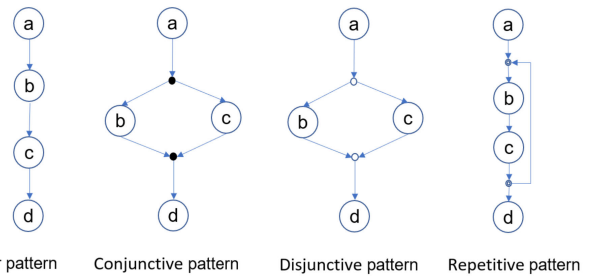


FIGURE 1. The SICN control-flow primitive patterns.

is needed after completion of activity α , which means that the transition is a sequential transition. Additionally stating to make sure, if $m(i) = 1$ for all i , then no parallel processing is initiated by completion of α .

3) GRAPHICAL FORMATION

Based on the interpretation, we graphically define these primitive transition types as shown in Figure 1. There are four types of transition (Control-flow) in the SICN-Based process model. The circles with the labels inside represent the activity (for example, here a, b, c, d), and the nodes we call Gateway-Transition, including the OPEN-Transition (Open or Split Gateway) and the corresponding CLOSE-Transition (Close or Join Gateway). In the linear pattern, the activity follows the activity in sequential order. There is no Gateway-Transition in the linear pattern. In the conjunctive pattern, an activity that has a conjunctive (or parallel) Gateway-Transition (AND-Gateway) that is graphically represented by a black dot, followed by two or more activities, those following activities will be performed. In the disjunctive pattern, an activity has a disjunctive (or decision) Gateway-Transition (OR-Gateway) is represented graphically by a white dot followed by two or more activities. Only one of the following activities will be performed next. In the repetitive pattern, an activity has an iterative Gateway-Transition (LOOP-Gateway) represented graphically by a double empty dot. The activities inside the loop area will be performed repeatedly. In the SICN process modeling methodology, the definition of structural formation means the SICN model is satisfied by keeping the model matched pairing property and the proper nesting property.

C. THE ρ -ALGORITHM

The ρ -Algorithm is a proportional process mining algorithm [23], [24] that can discover process mining SICN-oriented from process enactment event log dataset [7]. The ρ -Algorithm consists of three steps as following:

- Step 1: Mining groups of temporally ordered Adjacent Activity pairs. This step mines a chronologically ordered group of adjacent activity pairs from temporal work-cases of the process instance event logs.
- Step 2: Building up a Weighted Adjacent-Activity Set and its Weighted Process Pattern Graph. This step constructs all chronologically ordered groups of

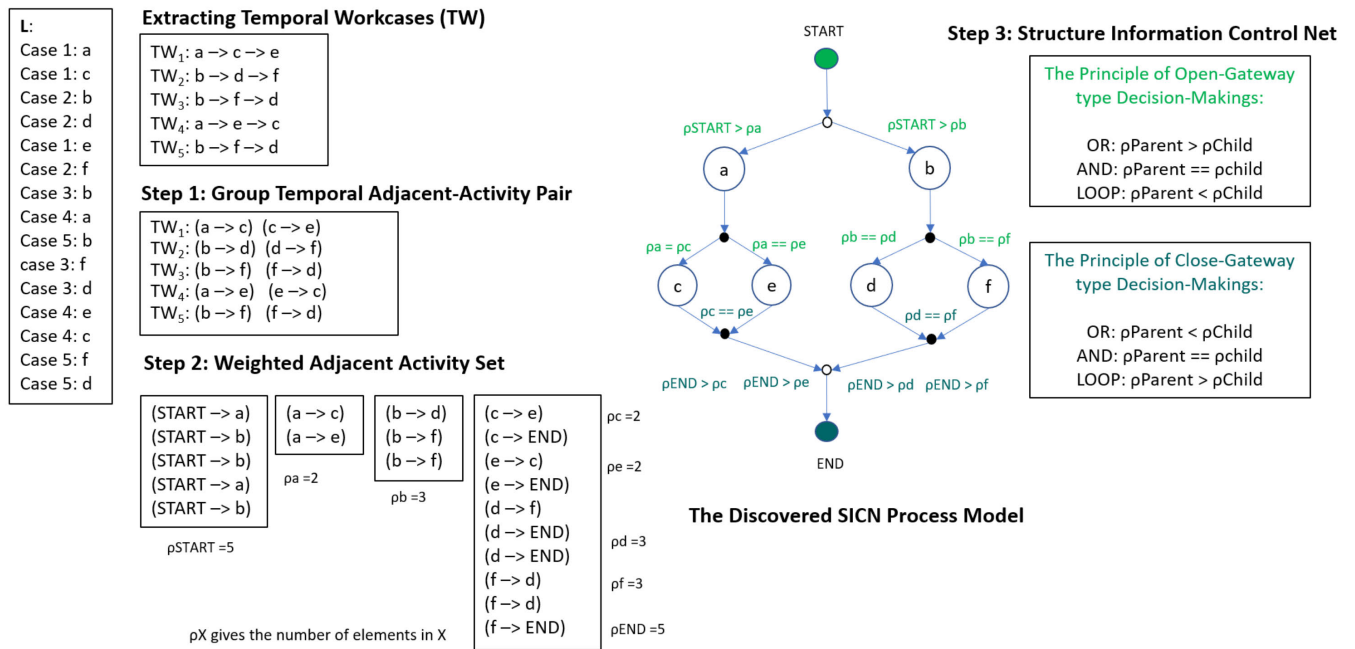


FIGURE 2. The stepwise of implementing the ρ-algorithm.

adjacent active pairs, each corresponding to an event trace process instance. The final output of this step is a weighted contiguous set of operations represented as a Weighted Process Pattern Graph.

- Step 3: Discovering All the Structural Process Patterns and building up a SICN. This step discovers SICN-Oriented process model from all groups of contiguous activity pairs ordered by time.

Figure 2 depicts each step of the ρ-algorithm to discover SICN-oriented process model from the event log dataset (L). Step-1, the algorithm extracts five temporal workcases and group temporal Adjacent-Activity pair, each corresponding to each process instance (trace) from the process enactment event log dataset (L); Step 2, the algorithm fragment each temporary workcase into a chronologically ordered group of contiguous activity pairs and convert into a weighted adjacent activity set by ρX operation [7]; Step-3, the algorithm finally discovers a SICN-oriented process model based on the decision-makings patterns for open and closed gateway respectively.

III. APPROACH

In the reality of business operations, the workflow processes can be highly diverse and change frequently. In process discovery SICN-oriented, one of the significant problems is how to mine four primitive control flows [22] and their classification (i.e., Linear, Disjunctive OR control, Conjunctive AND control, Repetitive LOOP control). Furthermore, these control flows will not simply appear separate but be combined or nested. The SICN-oriented process must not only ensure strict properly but also show the exact story

of workflow that event log data gives, thereby helping the manager make decisions to improve the model and improve the business performance. Explicitly uncovering parallel, selection, or redo flow contributes to constructing clearly control flows in the SICN-oriented process model and will be a base support for making decisions and improving the business operations. In the next subsection, we describe the concept of complex process models and approaches and an algorithm to discover it from the event log.

A. ML-CCFG

When building a SICN-oriented process model, we use four control-flow primitives for representing the model. As mentioned in the previous section, the discovered process model can be simple with control-flow primitives or very complex depending on workflow characteristics and event log data. The SICN-oriented process model becomes complex when combining many transaction gateway types at multi-levels. The concept of *Multi-Level and Compound Control-flow Gateways* in this paper means a group of SICN Control-flow primitive patterns created in the SICN-oriented process model includes two layers of meaning. The first layer is *Multi-level*, and the second is *Compound*. The former means that there are multiple levels of SICN Control-flow primitive patterns in the group, and the latter indicates that there is a combination of SICN control-flow multiple primitive patterns in the group. Figure 3 depicts the six types of ML-CCFG, including the Parallel-AND open ML-CCFG, Parallel-AND close ML-CCFG, Exclusive-OR open ML-CCFG, Exclusive-OR close ML-CCFG, Iterative-LOOP

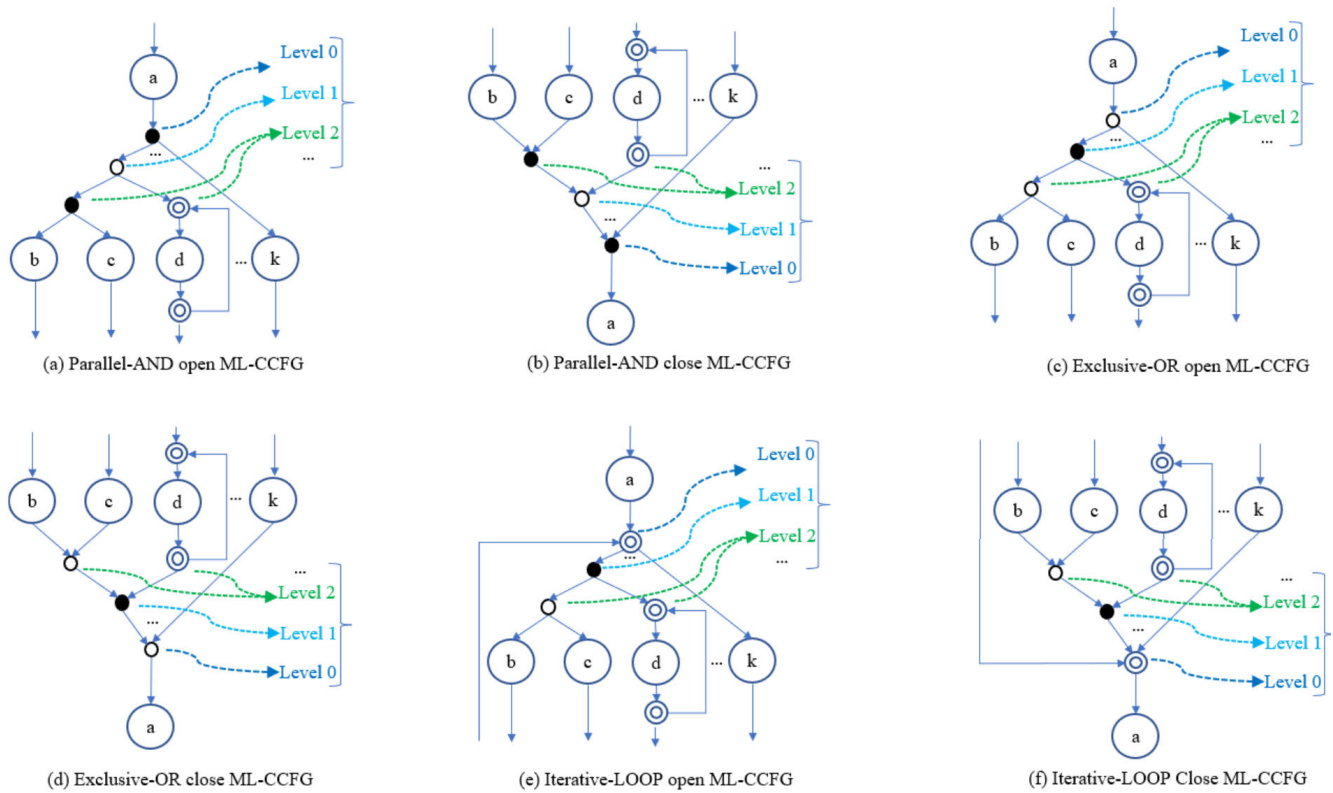


FIGURE 3. The six types of ML-CCFG.

open ML-CCFG, and Iterative-LOOP close ML-CCFG respectively.

Figure 3(a) illustrates a graphical formation of a Parallel AND open ML-CCFG with three levels from level 0 to level 2. The root level, i.e., level 0, is the AND open gateway. The next level, i.e., level 1, includes an OR open gateway and an activity (k). Finally, level 2 includes AND open gateway and LOOP open gateway. Figure 3(b) illustrates a formal generalization of a Parallel AND close ML-CCFG with three levels from level 0 to level 2. The highest level, i.e., level 0, is the AND close gateway. The next level, i.e., level 1, includes an OR close gateway and an activity (k). Finally, level 2 includes AND close gateway and LOOP close gateway. Similarly, Figure 3 (c, d, e, and f) illustrates a formal generalization for Exclusive-OR open ML-CCFG, Exclusive-OR close ML-CCFG, Iterative-LOOP open ML-CCFG, and Iterative-LOOP close ML-CCFG respectively. Note that the dots (...) implies that, depending on the complexity of the process, they can have more control-flow primitive patterns at deeper levels, i.e., levels 4, 5, etc, as well as more control-flow primitive patterns in a level. For the sake of brevity, we will sometimes refer to six types of ML-CCFG respectively: AND open *i*L-CCFG, AND close *i*L-CCFG, OR open *i*L-CCFG, OR close *i*L-CCFG, LOOP open *i*L-CCFG, and LOOP close *i*L-CCFG, where *i* is the number of level in the ML-CCFG. For example, an OR open 4L-CCFG is an Exclusive-OR ML-CCFG with three levels from level 0 to level 3.

B. DISCOVERING STRUCTURAL FORMATIONS OF ML-CCFG

This subsection proposes additional rules extending from the ρ -Algorithm to discover the SICN-oriented process model containing ML-CCFG. The motivation for our work arises from the increasing need for process mining algorithms that can handle more and more complex patterns in real-world event log datasets. To build a SICN-oriented process model, it is essential to have rules to decide when to open-gateways and close-gateways. These rules were introduced in [22] and were executed at STEP 3-2 of the ρ -Algorithm. We consider the following example to see how the ρ -Algorithm has not effectively coped with some specific consist of many primitive processes at multi-levels.

1) RUNING EXAMPLE

Consider the sample event log with six temporal workcases as follows $L = (aeed, ade, acdb, abcd, acbd, abcd)$, in which a, b, c, d, e are activities. Figure 4 (a). depicts the process pattern after removing deliberate-noise [7], and Figure 4 (b). depicts the SICN-oriented process model discovered by implementing ρ -Algorithm from the sample event log. The problems that appear here are: (i) the SICN model is not reasonable when it violates the rules of opening and closing the corresponding gateway, i.e., matched pairing property [22], (ii) the one gateway has only one child node (activity)

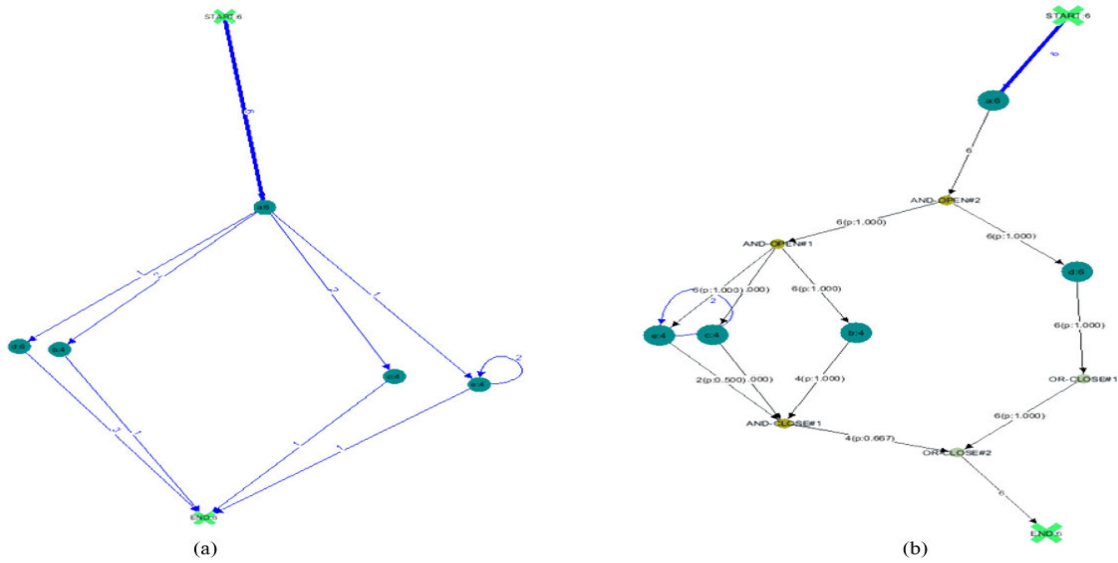


FIGURE 4. The weighted process pattern graph (a) and the SICN-oriented process model discovered (b) from the Sample event log.

TABLE 2. Rules for decision making ML-CCFG.

Type of ML-CCFG	Rule name	Rule for decision making
Exclusive-OR open ML-CCFG	Rule 1	1. $Outgoing(crN) \geq 3$ 2. $\rho(crN) > \rho(u)$ with all child node u of crN 3. Existing at least 2 child nodes u and v of crN : $\rho(u) = \rho(v)$
Exclusive-OR close ML-CCFG	Rule 2	1. $Incoming(crN) \geq 3$ 2. $\rho(crN) > \rho(u)$ with all parent nodes u of crN 3. Existing at least 2 parent nodes u and v of crN : $\rho(u) = \rho(v)$
Parallel-AND open ML-CCFG	Rule 3	1. $Outgoing(crN) \geq 3$ 2. Existing child nodes u_1, u_2, \dots, u_m ($m < outgoing(crN) $): $\sum_i^m \rho(u_i) = \rho(crN)$ 3. Existing at least 1 child node v of crN : $\rho(v) = \rho(crN)$
Parallel-AND close ML-CCFG	Rule 4	1. $Incoming(crN) \geq 3$ 2. Existing child nodes u_1, u_2, \dots, u_m ($m < outgoing(crN) $): $\sum_i^m \rho(u_i) = \rho(crN)$ 3. Existing at least 1 parent node v of crN : $\rho(v) = \rho(crN)$
Iterative-LOOP open ML-CCFG	Rule 5	1. $Incoming(crN) \geq 2$ 2. $Outgoing(crN) \geq 2$ 3. $\rho(crN) > \rho(u)$ with all parent nodes u of crN 4. $\rho(crN) > \rho(v)$ with all child nodes v of crN 5. Existing a control-path from one of parent nodes to crN
Iterative-LOOP close ML-CCFG	Rule 6	1. $Incoming(crN) \geq 2$ 2. $Outgoing(crN) \geq 2$ 3. $\rho(crN) > \rho(u)$ with all parent nodes u of crN 4. $\rho(crN) > \rho(v)$ with all child nodes v of crN 5. Existing a control-path from crN to one of child nodes

violates the rules of control-flow primitive pattern, (iii) and the model decide the wrong gateway for the combination multi-level of primitive patterns of nodes leads to not being able to play-out temporal workcases correctly. To solve the problems the ρ -algorithm encountered, we needed to add rules to help the ρ -algorithm deal with these complex cases as subsection following.

2) THE RULES FOR DECISION-MAKING A ML-CCFG

Let a weighted process pattern graph $G = (V, E)$, where V is the set of nodes/vertices(activities), and E is the set of directed arcs connecting the vertices. Let ρX be the list of

ρ values of all nodes in V , calculated by the ρ -function [7]. Our key idea is to traverse depth-first to build the lowest level gateway, i.e., the primitive patterns, and traverse breadth-first to combine the primitive pattern to build the complex gateway based on the main idea of the ρ -Algorithm. To detect and construct SICN compound patterns from G , along with the original rules of the ρ -Algorithm, we are adding a set of rules to decide on a current node(crN) that can be an ML-CCFG. Considering the case of open gateway, the ρ values of the parent nodes satisfy simultaneously two of these conditions: (i) the ρ values of the child nodes are equal of each other but different from the ρ value of parent node; (ii) the ρ values of

Algorithm 1 Discovery Loop Gateways From Weighted Process Pattern Graph

Require: A Weighted Process Pattern Graph $G = (V, E)$
Require: A dictionary of rho values of nodes ρ
Ensure: A Weighted Process Pattern Graph with Loop Gateways G , A updated dictionary ρ

```

1: OpenIdx  $\leftarrow 0$ 
2: CloseIdx  $\leftarrow 0$ 
3: for ( $\forall v \in G$ ) do
4:   if  $v.edgeInComing > 1$  then
5:     for ( $\forall f \in v.fathers$ ) do
6:       if  $(\rho(f) < \rho(v)$  and  $hasCycle(v, f)$  then
7:         OpenIdx  $\leftarrow$  OpenIdx + 1
8:          $\rho[LOIdx] \leftarrow \rho(v) - |(f, v)|$ 
9:         V.add(LOIdx)
10:        for ( $\forall f \in v.fathers$ ) do
11:          E.add((LOIdx, v))
12:          E.add((f, LOIdx))
13:          E.remove((f, v))
14:        end for
15:      end if
16:    end for
17:  end if
18:  if  $v.edgeOutGoing > 1$  then
19:    for ( $\forall s \in v.sons$ ) do
20:      if  $(\rho(s) < \rho(v)$  and  $hasCycle(v, s)$  then
21:        CloseIdx  $\leftarrow$  CloseIdx + 1
22:         $\rho[LCIdx] \leftarrow \rho(v) - |(v, s)|$ 
23:        V.add(LCIdx)
24:        for ( $\forall s \in v.sons$ ) do
25:          E.add((s, LCIdx))
26:          E.add((LCIdx, s))
27:          E.remove((v, s))
28:        end for
29:      end if
30:    end for
31:  end if
32: end for

```

some child nodes are equal to the ρ value of parent node, but the rest of the nodes do not; (iii) existing several child node pairs (or triplets, quadruplets and so on) whose sum of ρ value of them is equal to ρ value of parent node. These conditions are similar for close gateways. We summarized the rules of decision-making for discovering ML-CCFG specified in Table 2.

3) THE ρ MC-ALGORITHM

The ρ MC-algorithm is divided into two phases. In the first phase, we create all the LOOP gates for the process model. In the next phase, we will create AND/OR gateway at the primitive patterns and combine them based on the previous decision-making rules. The first phase is performed by **Algorithm 1**. As you can see in the pseudocode, we look at the current node with its edgeInComing/edgeOutGoing

Algorithm 2 Create And/OR Gateway for SICN Model From Weighted Process Pattern Graph

Require: A Weighted Process Pattern Graph $G=(V, E)$
Require: A dictionary of rho values of nodes ρ
Require: A type of gate t , a vertex v , a list of node Ln
Ensure: A Weighted Process Pattern Graph with SICN-Gateway G^{SICN}

```

1: procedure CreateGate( $V, E, \rho, t, v, Ln$ )
2:   V.add( $t$ )
3:   if "OR" in  $t$  then
4:      $\rho[t] = \sum_{k \in Ln} \rho(k)$ 
5:   end if
6:   if "AND" in  $t$  then
7:      $\rho[t] = \rho(Ln[0])$ 
8:   end if
9:   if "Open" in  $t$  then ▷ Create OpenGate
10:    E.add( $(t, v)$ )
11:    for ( $\forall n \in Ln$ ) do
12:      E.add( $(t, n)$ )
13:    end for
14:  end if
15:  if "Close" in  $t$  then ▷ Create CloseGate
16:    E.add( $(v, t)$ )
17:    for ( $\forall n \in Ln$ ) do
18:      E.add( $(n, t)$ )
19:    end for
20:  end if
21: end procedure

```

(parent/child) and whether existing a cycle between them to create a LOOP Open/Close gateway. Then algorithm calculates the ρ value for the added LOOP gateway, excluding the number of redo times. This will form the correct basis for comparing proportional ρ values to decide on a ML-CCFG according to the proposed rules in the next phase.

To do the second phase, we build a procedure to create a gateway for transforming the block process pattern in the Weighted Process Pattern Graph into a SICN-oriented model by **Algorithm 2**. The main part of the proposed approach is **Algorithm 3** to identify and transform the process pattern graph to the SICN-oriented process model containing ML-CCFG. **Algorithm 3** will travel in the graph to create AND/OR gateway at primitive patterns and combine them based on the previous rules. Once a node satisfies the rules constituting the primitive pattern is identified, the CREATGATE procedure in **Algorithm 2** is called to add the gateway into the graph.

IV. EXPERIMENTAL RESULTS

A. DATASETS

We implemented a series of experiments to evaluate the effectiveness of the proposed ρ MC-algorithm and applied it to three process logs. Table 3 shows a summary of the

Algorithm 3 Discovery SICN-Oriented Process Model**Require:** A Weighted Process Pattern Graph $G = (V, E)$, A dictionary of rho values of nodes ρ **Ensure:** A Structure Information Control Net Graph G^{SICN}

```

1: OpenIdx  $\leftarrow$  0
2: CloseIdx  $\leftarrow$  0
3: for ( $\forall v \in G$ ) do
4:   if  $v.edgeOutGoing > 2$  then ▷ Checking the rule for AND/OR compound open gateways
5:     sonsOfAndOpen  $\leftarrow$  []
6:     sonsOfOrOpen  $\leftarrow$  []
7:     for ( $\forall son \in v.sons$ ) do ▷ Checking the rule to create primitive AND/OR OpenGate
8:       if ( $\rho(son) == \rho(v)$ ) then
9:         sonsOfAndOpen.add(son)
10:      end if
11:      if ( $\rho(son) < \rho(v)$  and ( $\sum_{\forall v \in sonsInOrOpenGate} \rho(v) + \rho(son) \leq \rho(v)$ )) then
12:        sonsOfOrOpen.add(son)
13:      end if
14:    end for
15:    if len(sonsOfAndOpen) > 1 then
16:      OpenIdx  $\leftarrow$  OpenIdx + 1
17:      tgate  $\leftarrow$  "AND-Open" + str(OpenIdx)
18:      CreateGate(V,E, $\rho$ , tgate, v, sonsOfAndOpen) ▷ Call procedure CreateGate in Alogrithm 1
19:    end if
20:    if len(sonsOfOrOpen) > 1 then
21:      OpenIdx  $\leftarrow$  OpenIdx + 1
22:      tgate  $\leftarrow$  "OR-Open" + str(OpenIdx)
23:      CreateGate(V,E, $\rho$ , tgate, v, sonsOfOrOpen)
24:    end if
25:  end if
26:  if  $v.edgeInComing > 2$  then ▷ Checking the rule for AND/OR compound close gateways
27:    fathersOfAndClose  $\leftarrow$  []
28:    fathersOfOrClose  $\leftarrow$  []
29:    for ( $\forall father \in v.fathers$ ) do ▷ Checking the rule to create primitive AND/OR CloseGate
30:      if ( $\rho(father) == \rho(v)$ ) then
31:        fathersOfAndClose.add(father)
32:      end if
33:      if ( $\rho(father) < \rho(v)$  and ( $\sum_{\forall v \in fathersInOrCloseGate} \rho(v) + \rho(father) \leq \rho(v)$ )) then
34:        fathersOfOrClose.add(father)
35:      end if
36:    end for
37:    if len(fathersOfAndClose) > 1 then
38:      CloseIdx  $\leftarrow$  CloseIdx + 1
39:      tgate  $\leftarrow$  "AND-Close" + str(CloseIdx)
40:      CreateGate(V,E, $\rho$ , tgate, v, fathersOfAndClose)
41:    end if
42:    if len(fathersOfOrClose) > 1 then
43:      CloseIdx  $\leftarrow$  CloseIdx + 1
44:      tgate  $\leftarrow$  "OR-Close" + str(CloseIdx)
45:      CreateGate(V,E, $\rho$ , tgate, v, fathersOfOrClose)
46:    end if
47:  end if
48: end for

```

basic information of these three datasets. The first dataset is the Sample log dataset mentioned in the running example above. The second dataset is the review-example-large

event logs [21]. This process log contains 10,000 traces, 236360 events, and 14 activities with several iterative constructs. This dataset contains process instances about the

TABLE 3. Summary of the event log datasets.

Dataset	Number of traces	Number of events	Number of activities
Sample log	6	24	5
Review example large	10000	236360	14
Large bank transaction	10000	678864	113

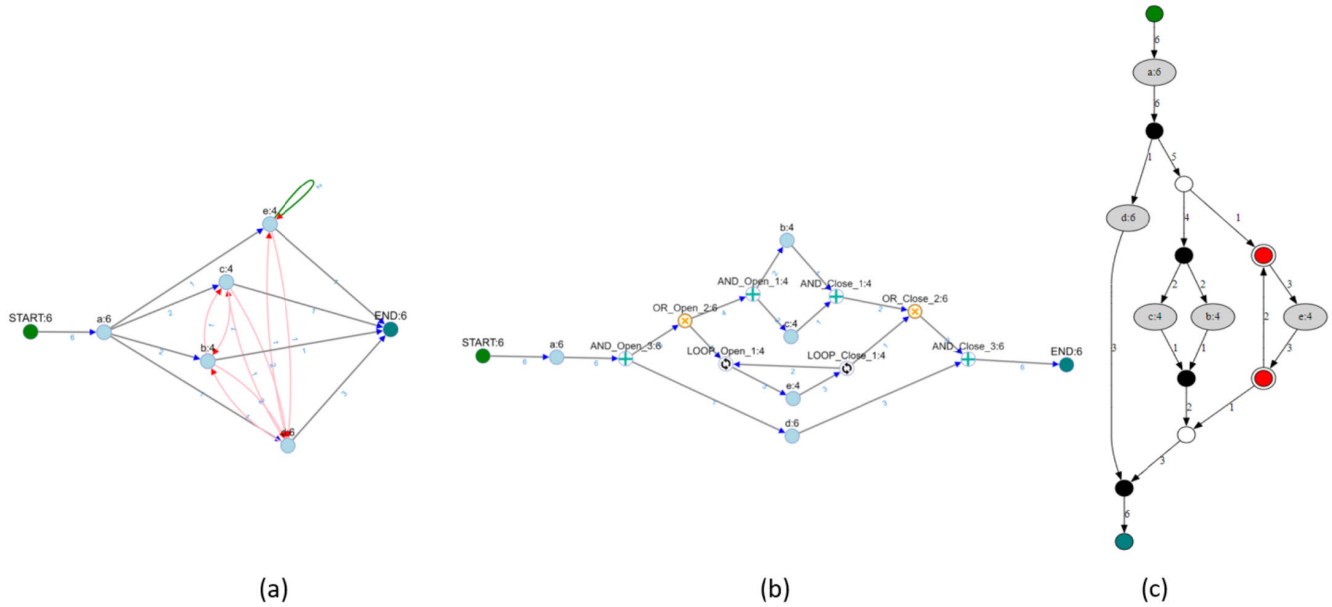


FIGURE 5. The SICN-oriented process model discovered from the sample log by implemented ρ MC-algorithm.

review process for a journal. For review, a paper will be sent to three different reviewers. Reviewers are invited to write a review. In practice, reviewers often do not respond for various reasons. Therefore, it is not always possible to decide on the first round of review. When a paper has not received sufficient reviews, it is necessary to invite additional reviewers. This process is repeated until a final decision (accept or reject) can be made. Therefore, this workflow will contain several multiple and compound patterns. The third dataset is the large bank transaction dataset [21]. This is a non-noise and synthetic process enactment event log dataset, which is very appropriate and ideal to be used for validating the functional correctness of a certain process mining algorithm. There are 113 activities associated with the total number of 10,000 traces and 678864 events in this log dataset. Thus, the SICN-oriented process model from these datasets potentially contains ML-CCFG, making them suitable for testing the ability of the ρ MC-algorithm to discover a process model. The following subsection will present the experiments and results of discovering structural formations of ML-CCFG on three of the given datasets.

B. RESULTS

To prove the proposed algorithm can work effectively, we built a system to implement it and conducted verification

experiments. We use Cytoscape² and Graphviz³ libraries to visualize and manipulate the process patterns.

In the first experiment, we mined the process model on the Sample Log dataset. Figure 5 depicts the results of the visualized modeling. Specifically, Figure 5(a) is the process pattern, including deliberate noises (which were visualized in pink color), and Figure 5(b) is the constructed SICN-oriented process model visualized by Cytoscape. Figure 5(c) is the SICN-oriented process model displayed in Graphviz formatted. Applying rules 1, 2, 3, and 4 for the entire model, we successfully transform the weighted process pattern graph to SICN-oriented process model with created ML-CCFG are Parallel-AND open 3L-CCFG (AND_Open_3); Parallel-AND close 3L-CCFG (AND_Close_3), Exclusive-OR open 2L-CCFG (OR_Open_3), and Exclusive-OR close 2L-CCFG (OR_Close_3). It can be seen that the process model discovered was a perfect fit according to the SICN rules, i.e., matched pairing property as well as the proper nesting property, correctly uncovering the ML-CCFG for the event log. All disjunctive/ conjunctive/ repetitive OPEN gateways and CLOSE gateways are created and paired strictly, and all gateways are created and coupled tightly. Simultaneously, it is easily confirmed that the process model discovered can play back all the temporary workcases correctly.

²<https://cytoscape.org/>

³<https://graphviz.org/>

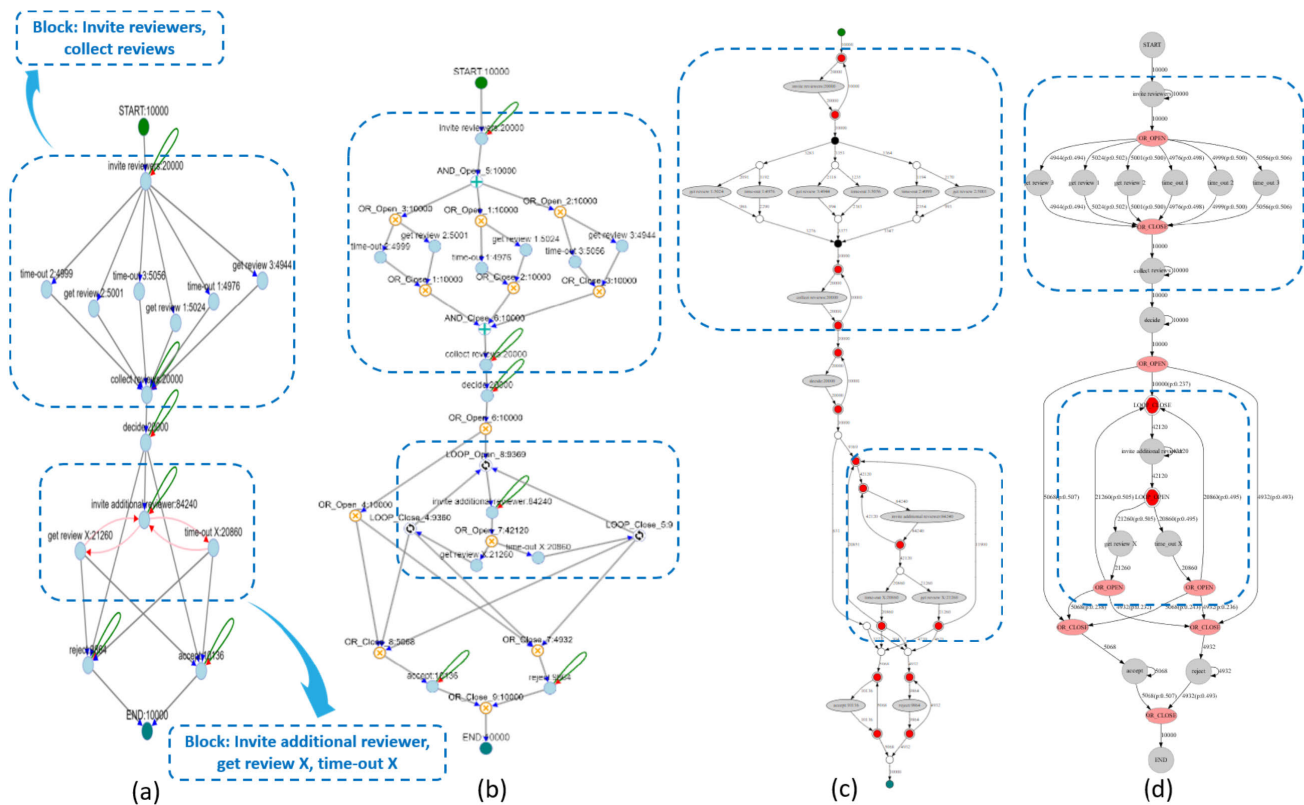


FIGURE 6. The SICN-oriented process model discovered from the review example large dataset by implemented ρ MC-algorithm.

We discovered the SICN-oriented process model from a large-scale process dataset in the second experiment. Figure 6 Depicts the results of discovered SICN-oriented process model from the review example large data set. We consider two blocks, the *Invite reviewers, collect reviews* block and the *Invite additional reviewer, get review X, time-out X* block (visualized as blue rounded dotted rectangles). These blocks consist of many nodes (activities) and arcs (control-path) inside are potential blocks to form the ML-CCFG when building the SICN-oriented process model. Figure 6(a) shows the weighted process pattern graph visualized using Cytoscape. Figure 6(b) shows the SICN-oriented process model discovered and visualized using Cytoscape. Figure 6(c) is the discovered SICN-oriented process model displayed in Graphviz format. Meanwhile, Figure 6(d) results from discovered SICN-oriented process model visualized in Graphviz format implemented by the mining system in [25]. There are two significant differences in the model discovered by the expansion algorithm and the original ρ -Algorithm, as seen in Figure 6(c) and Figure 6(d), respectively. First, applying rules 3 and 4 for the *Invite reviewers, collect reviews* block, our newly discovered model in Figure 6(c) shows that we have correctly discovered the ML-CCFG for this block. A review submission will be made with three reviewers simultaneously, i.e., a Conjunctive (Parallel-AND) open gateway must be created following

Invite reviewers activity, and a Conjunctive (Parallel-AND) close gateway must be created preceding *collect reviews* activity. Moreover, for each reviewer, the editorial board will either get their review or will wait a period (time-out) with no response, so that each pair works (*get review 1, time-out 1*), (*get review 2, time-out 2*), (*get review 3, time-out 3*) must be divided into disjunctive (Exclusive-OR) primitive pattern, each group corresponding to the receiving back or not from the corresponding reviewer 1, reviewer 2, and review 3. As a result, we have a Parallel-AND open 2L-CCFG, a Parallel-AND close 2L-CCFG discovered from the *Invite reviewers, collect reviews* block. This resulting model can be done more perfectly than the SICN-oriented model discovered by the original ρ -algorithm depicted in Figure 6(d). Second, we apply rule 5 and rule 6 for the *Invite additional reviewer, get review X, time-out X* block. When the editorial board has not collected enough three reviews, they will have to repeat the process of sending a review invitation to another reviewer and may also receive a response *get review X* or not after a period *time-out X*. Similar to the analysis for the review collection activity from reviews 1, 2, and 3 mentioned previously, in this block, *get review X, time-out X* must be into a disjunctive (Exclusive-OR) primitive pattern. Moreover, this block is carried out by repeating the operation iteratively until a paper receives enough review from three reviewers and is reached the conclusion of either accepting or rejecting the

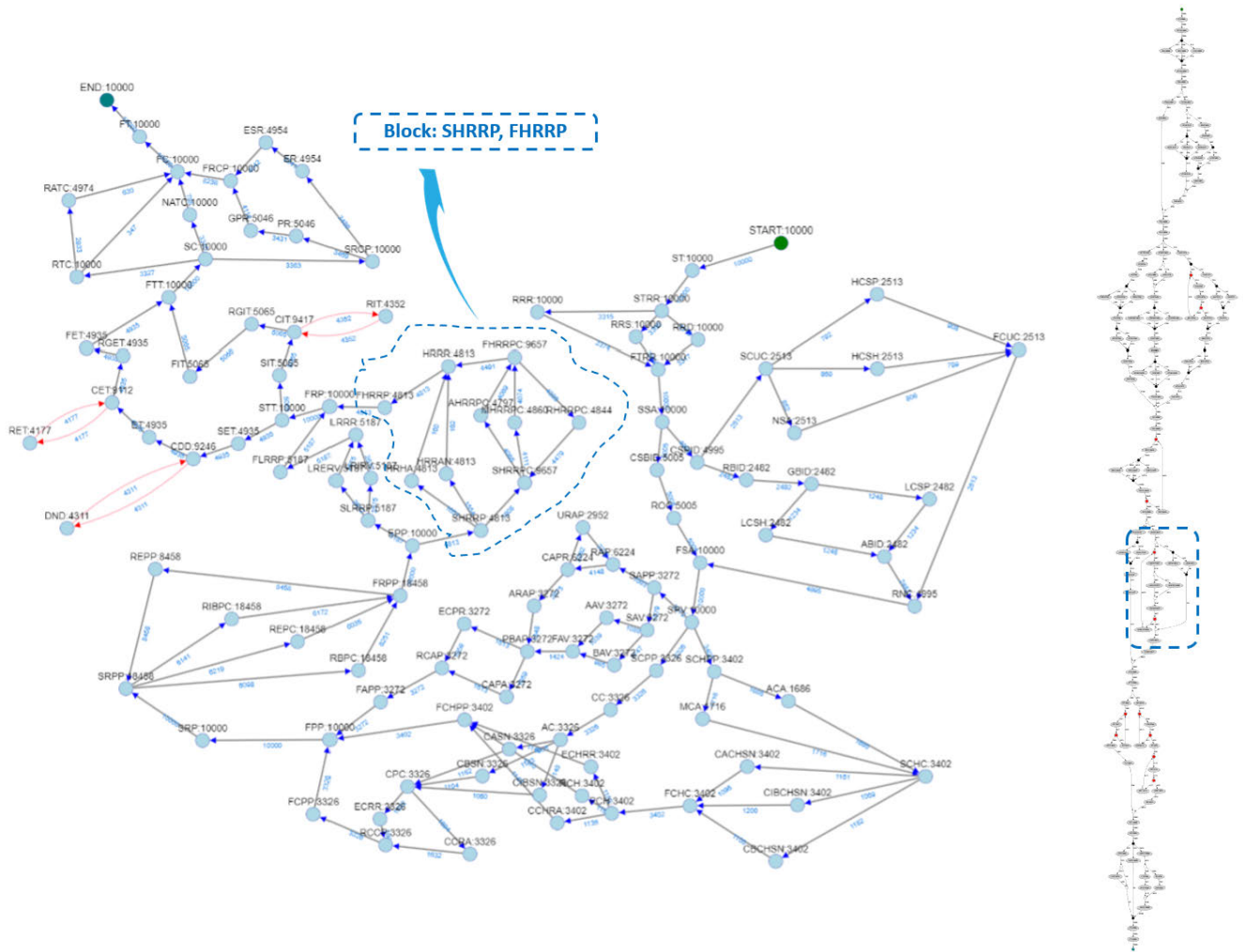


FIGURE 7. The SICN-oriented process model discovered from the large bank transaction dataset by implemented ρ MC-algorithm.

paper. Therefore, our discovered model can clearly identify a Iterative-LOOP open 2L-CCFG and a Iterative-LOOP close 2L-CCFG discovered from the *Invite additional reviewer, get review X, time-out X* block, as shown in Figure 6(c). The discovered SICN-oriented process model in Figure 6(d) did not deal with this block precisely.

In the third experiment, we performed mining ML-CCFG from the Large Bank Transaction process dataset. Figure 7. shows the result of this experiment, the left side is the process pattern graph, and the right side is discovered SICN-oriented process model in Graphviz format. We consider the *SHRRP, FHRRP* block (visualized as blue dotted shape). Applying rule 1 for node *SHRRP*, an Exclusive-OR open 3L-CCFG following *SHRRP* was created. Applying rule 2 for node *FHRRP*, an Exclusive-OR close 3L-CCFG preceding *FHRRP* was created. Applying rule 5 for node *SHRRPC*, an Iterative-LOOP open 2L-CCFG preceding *FHRRPC* was created. Applying rule 6 for node *FHRRPC*, an Iterative-LOOP compound 2L-CCFG following *FHRRPC* was created.

So far, by conducting three experiments, we show that our algorithm, with the added extending rules along with the implemented system, efficiently discovered the SICN-oriented process model containing ML-CCFG strictly complies with matched pairing property and the proper nesting property in each compound gateway. Finally, the results of the experiments mining ML-CCFG were summarized in Table 4. In the table, the first column consists of the name of the event log dataset, and the entire or a block of the process pattern graph contains ML-CCFG. The second column describes the rules applied to mining the ML-CCFG. The third and fourth columns are images of the entire or partial process patterns and the result of transforming them to ML-CCFG implemented by the proposed algorithm. Our results show that the algorithm discovered the SICN-oriented process model in a rigorous, reasonable, and detailed. In the following subsection, we provide details of the results of experiments on three datasets and compare them to the algorithms ρ -algorithm original [7], $\alpha^{\#}$ [5], ILP [8], and

TABLE 4. Summary of experimental verification cases applied rules, partial process patterns, and ML-CCFG discovered results.

Dataset, Block, type of ML-CCFG	Rules applied	Partial pattern graph	Partial SICN-oriented model
<p>Sample log</p> <p>Entire model</p> <p>AND open 3L-CCFG</p> <p>AND close 3L-CCFG</p> <p>OR open 2L-CCFG</p> <p>OR close 2L-CCFG</p>	<p>Rule 1</p> <p>Rule 2</p> <p>Rule 3</p> <p>Rule 4</p>		
<p>Review example large</p> <p>Block:</p> <p>invite reviewers,</p> <p>collect reviews</p> <p>AND open 2L-CCFG</p> <p>AND close 2L-CCFG</p>	<p>Rule 3</p> <p>Rule 4</p>		
<p>Review example large</p> <p>Block:</p> <p>Invite additional reviewer,</p> <p>get review X,</p> <p>time-out X</p> <p>LOOP open 2L-CCFG</p> <p>LOOP close 2L-CCFG</p>	<p>Rule 5</p> <p>Rule 6</p>		
<p>Large bank transaction</p> <p>Block:</p> <p>SHRRP, FHRRP</p> <p>OR open 3L-CCFG</p> <p>OR close 3L-CCFG</p> <p>LOOP open 2L-CCFG</p> <p>LOOP close 2L-CCFG</p>	<p>Rule 1</p> <p>Rule 2</p> <p>Rule 5</p> <p>Rule 6</p>		

IDM [26]. We used the ProM 6.12 tool [27] to implement the $\alpha^{\#}$, ILP, and IDM algorithms because these algorithms are recently developed and are freely available in the toolkit.

Specifically, in Table 5, the process model discovered by the ρMC on the Sample log is equivalent to the discovered model by the IDM algorithm and more reasonable than all other algorithms. We can see that $\alpha^{\#}$ and ILP algorithms fail to discover loop patterns (activity e) in the process model. Meanwhile, the ρ -algorithm violates the match pairing

property and forms a gateway as the definition of the SICN model.

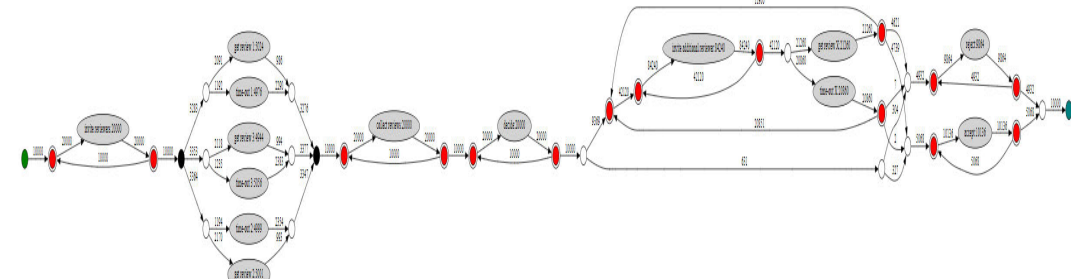
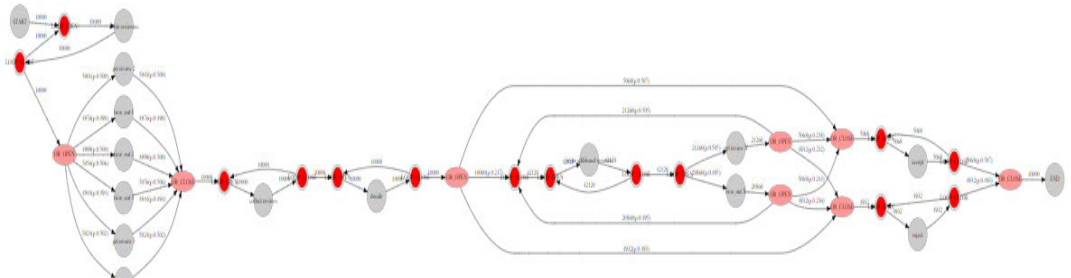
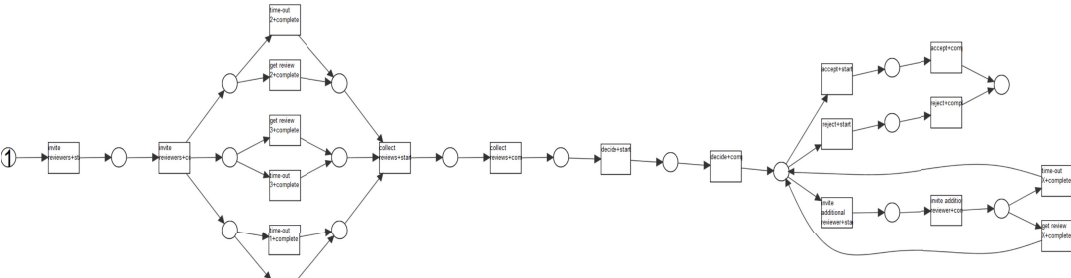
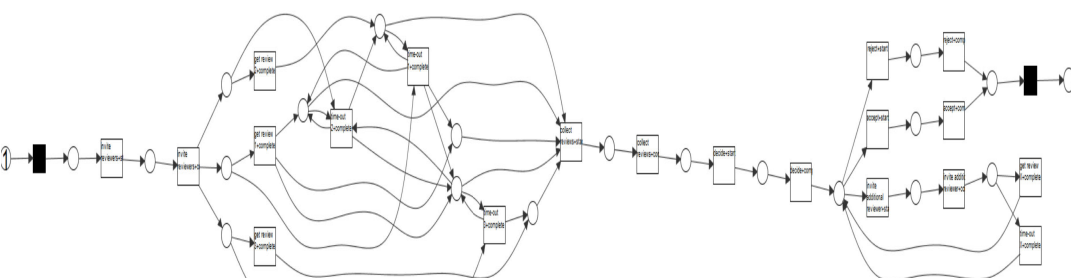
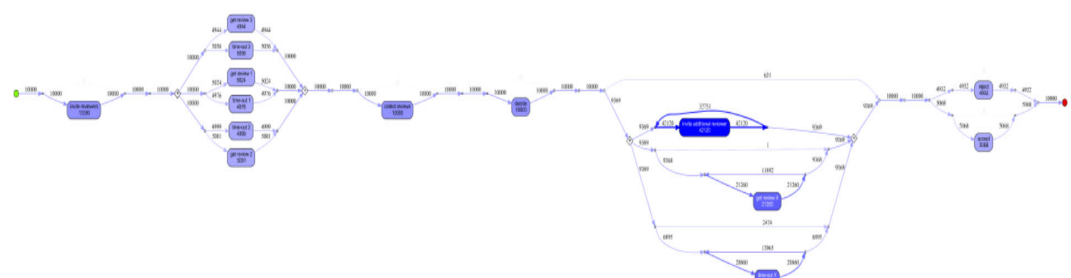
In Table 6, our result on the review example large dataset shows that our approach deals well with both blocks (i.e. *Invite reviewers, collect reviews* block and the *Invite additional reviewer, get review X, time-out X* block), while the rest of the algorithms cannot explore both blocks well.

In Table 7, the results of the discovered process model implemented by our algorithm on the large bank transaction

TABLE 5. Process model discovered from sample log.

Algorithm	Discovered model
ρMC	
ρ	
$\alpha\#$	
ILP	
IDM	

TABLE 6. Process model discovered from review example large log.

Algorithm	Discovered model
ρMC	
ρ	
$\alpha\#$	
ILP	
IDM	

set, while the remaining algorithms cannot discover the model (time-out). This also depicts another aspect of discovering models on very large-scale datasets. The result shows that our proposed algorithm is extremely powerful when coping well with a large-scale process containing more than 600,000 events. This shows the scalability potential of the proposed method in handling event logs in today's growing big-data environment.

V. CONCLUSION

In this paper, we proposed an extending algorithm from the ρ -Algorithm, called the ρ MC-Algorithm, capable of discovering multiple and compound SICN process patterns from event logs. Our experimental verification of the datasets shows that the ρ MC-Algorithm effectively discovers SICN process patterns from event logs. Consequently, our work in this paper makes several contributions to the field of process mining:

- We defined and introduced the concept of ML-CCFG, which may be encountered in real-world event logs.
- Proposed the ρ MC-algorithm by extending the processing rules of the ρ -algorithm to discover the SICN-oriented process model containing ML-CCFG from the process logs.
- Our experimental evaluation proved that the proposed approach could discover the ML-CCFG from the process event log datasets. Moreover, the experimental results also show that the proposed approach not only achieves more reasonable results compared to other approaches (Tables 5, 6) but also can discover process models on very large datasets (Table 7).

We believe that the proposed algorithm has the potential to advance the field of process mining and provide valuable insights into complex processes in real-world business operations. In the future, we plan to integrate this research with artificial intelligence to solve real-world problems, such as using deep learning to support functions of learning management systems [28] or reinforcement learning-based intelligent decision-making [29].

REFERENCES

- [1] W. M. P. van der Aalst and A. J. M. M. Weijters, "Process mining: A research agenda," *Comput. Ind.*, vol. 53, no. 3, pp. 231–244, Apr. 2004.
- [2] W. van der Aalst, T. Weijters, and L. Maruster, "Workflow mining: Discovering process models from event logs," *IEEE Trans. Knowl. Data Eng.*, vol. 16, no. 9, pp. 1128–1142, Sep. 2004.
- [3] A. K. A. de Medeiros, B. F. van Dongen, W. M. P. van der Aalst, and A. J. M. M. Weijters, "Process mining: Extending the algorithm to mine short loops," in *Proc. BETA Work. Paper Ser.*, vol. 113, 2004, pp. 1–25.
- [4] L. Wen, W. M. P. van der Aalst, J. Wang, and J. Sun, "Mining process models with non-free-choice constructs," *Data Mining Knowl. Discovery*, vol. 15, no. 2, pp. 145–180, Oct. 2007.
- [5] L. Wen, J. Wang, W. M. P. van der Aalst, B. Huang, and J. Sun, "Mining process models with prime invisible tasks," *Data Knowl. Eng.*, vol. 69, no. 10, pp. 999–1021, Oct. 2010.
- [6] K. P. Kim and C. A. Ellis, " σ -Algorithm: Structured workflow process mining through amalgamating temporal workcases," in *Proc. Pacific-Asia Conf. Knowl. Discovery Data Mining*, 2007, pp. 119–130.
- [7] K. S. Kim, D. L. Pham, and K. P. Kim, " ρ -Algorithm: A SICN-oriented process mining framework," *IEEE Access*, vol. 9, pp. 139852–139875, 2021.
- [8] S. J. van Zelst, B. F. van Dongen, W. M. P. van der Aalst, and H. M. W. Verbeek, "Discovering workflow nets using integer linear programming," *Computing*, vol. 100, no. 5, pp. 529–556, May 2018.
- [9] C. W. Günther and W. M. P. van der Aalst, "Fuzzy mining—adaptive process simplification based on multi-perspective metrics," in *Business Process Management*. G. Alonso, P. Dadam, and M. Rosemann, Eds. Berlin, Germany: Springer, 2007, pp. 328–343.
- [10] A. Weijters, W. A. van der, and A. A. De Medeiros, *Process Mining With the Heuristics Miner Algorithm (BETA)*. Eindhoven, The Netherlands: Technische Universiteit Eindhoven, 2006.
- [11] S. J. Leemans, *Robust Process Mining With Guarantees*, vol. 2196. Technische Universiteit Eindhoven, 2018. [Online]. Available: <https://research.tue.nl/en/publications/robust-process-mining-with-guarantees>
- [12] X. Wang, Z. Cao, R. Zhan, M. Bai, Q. Ma, and G. Li, "Density-based outlier detection in multi-dimensional datasets," *KSII Trans. Internet Inf. Syst.*, vol. 16, no. 12, pp. 3815–3835, 2022.
- [13] S. Liu, H. Qiao, L. Yuan, Y. Yuan, and J. Liu, "Research on data augmentation algorithm for time series based on deep learning," *KSII Trans. Internet Inf. Syst.*, vol. 17, no. 6, pp. 1530–1544, 2023.
- [14] H. Sun, W. Liu, L. Qi, Y. Du, X. Ren, and X. Liu, "A process mining algorithm to mixed multiple-concurrency short-loop structures," *Inf. Sci.*, vol. 542, pp. 453–475, Jan. 2021.
- [15] N. Martin, D. A. Fischer, G. D. Kerpedzhiev, K. Goel, S. J. J. Leemans, M. Röglinger, W. M. P. van der Aalst, M. Dumas, M. La Rosa, and M. T. Wynn, "Opportunities and challenges for process mining in organizations: Results of a delphi study," *Bus. Inf. Syst. Eng.*, vol. 63, no. 5, pp. 511–527, Oct. 2021.
- [16] X. Lu, A. Gal, and H. A. Reijers, "Discovering hierarchical processes using flexible activity trees for event abstraction," in *Proc. 2nd Int. Conf. Process Mining (ICPM)*, Oct. 2020, pp. 145–152.
- [17] W. M. Van Der Aalst, "Process-aware information systems: Lessons to be learned from process mining," *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 5460. Springer, May 2009, pp. 1–26, doi: 10.1007/978-3-642-00899-3_1.
- [18] W. M. Van Der Aalst, "Relating process models and event logs 21 conformance propositions," in *Proc. CEUR Workshop*, vol. 2115, no. 9, 2018, pp. 56–74.
- [19] M. Park and K. Kim, "Control-path oriented workflow intelligence analyses," *J. Inf. Sci. Eng.*, vol. 24, no. 2, pp. 343–359, 2008.
- [20] *IEEE Standard for Extensible Event Stream (XES) for Achieving Interoperability in Event Logs and Event Streams*, IEEE Standard 1849-2016, 2016, pp. 1–50.
- [21] BC 4TU ResearchData. (2023). *Event Log Datasets for Research From 4TU*. ResearchData. [Online]. Available: <https://data.4tu.nl/>
- [22] K. H. Kim and C. A. Ellis, "ICN-based workflow model and its advances," *Handbook Res. Bus. Process Model.*, pp. 142–171, 2009.
- [23] K. Kim, M. Yeon, B. Jeong, and K. Kim, "A conceptual approach for discovering proportions of disjunctive routing patterns in a business process model," *KSII Trans. Internet Inf. Syst.*, vol. 11, no. 2, pp. 1148–1161, 2017.
- [24] K. Kim, Y.-K. Lee, H. Ahn, and K. P. Kim, "An experimental mining and analytics for discovering proportional process patterns from workflow enactment event logs," *Wireless Netw.*, vol. 28, no. 3, pp. 1211–1218, Apr. 2022.
- [25] K.-S. Kim, D.-L. Pham, Y.-I. Park, and K. P. Kim, "Experimental verification and validation of the SICN-oriented process mining algorithm and system," *J. King Saud Univ. Comput. Inf. Sci.*, vol. 34, no. 10, pp. 9793–9813, Nov. 2022.
- [26] S. J. J. Leemans. (2017). *Inductive Visual Miner Manual*. [Online]. Available: <http://promtools.org>
- [27] W. M. P. van der Aalst, A. Bolt, and S. J. van Zelst, "RapidProm: Mine your processes and not just your data," 2017, *arXiv:1703.03740*.
- [28] J. Yun and T. Park, "An Analysis of university students' needs for learning support functions of learning management system augmented with artificial intelligence technology," *KSII Trans. Internet Inf. Syst.*, vol. 17, no. 1, pp. 1–15, 2023.
- [29] X. Xie, Z. Dou, and Y. Zhang, "Reinforcement learning-based intelligent decision-making for communication parameters," *KSII Trans. Internet Inf. Syst.*, vol. 16, no. 9, pp. 2942–2960, 2022.



process-aware information systems, predictive process monitoring, deep learning, and applications supporting crime prevention and prediction.

THANH-HAI NGUYEN received the B.S. and M.Sc. degrees in computer science from Thai Nguyen University, Vietnam, in 2008 and 2012, respectively. He is currently pursuing the Ph.D. degree with Kyonggi University, South Korea. Since 2008, he has been an IT Engineer with Thai Nguyen University. He has taught various courses for undergraduate students and deployed several projects. His research interests include process mining, business process management systems,



2007, he has been the Founder and the Director of the Contents Convergence

KWANGHOON PIO KIM (Member, IEEE) received the B.S. degree in computer science from Kyonggi University, South Korea, in 1984, the M.S. degree in computer science from Chung-Ang University, South Korea, in 1986, and the M.S. and Ph.D. degrees in computer science from the University of Colorado Boulder, Boulder, CO, USA, in 1994 and 1998, respectively. From 2017 to 2021, he was the Dean of the Computerization and Informatics Institute, Kyonggi University. Since

Software Research Institute, Kyonggi University, where he has been leading and fulfilling a multi-million dollars research project that will be continuously supported and funded by the National Research Foundation of Korea, from 2020 to 2029, ever since the institute was officially designated as the National Science and Engineering Research Institute by the Ministry of Education, South Korea, in 2020. He was a Researcher and a Developer with Aztek Engineering, American Educational Products Inc., and IBM, USA, and a Research Member of Staff with the Electronics and Telecommunications Research Institute (ETRI), South Korea. He is currently a Full Professor with the Division of AI Computer Science and Engineering and the Founder and the Supervisor of the Data and Process Engineering Research Laboratory, Kyonggi University. His research interests include groupware, workflow and business process management systems, BPM, CSCW, collaboration theory, Grid/P2P distributed systems, process warehousing and mining, predictive process monitoring, workflow-supported social networks discovery and analysis, process-aware information systems, data-intensive workflows, process-aware Internet of Things, predictive process modeling, process deep-learning, active contents big data engineering, and applications supporting crime prevention and prediction. He is the Vice Chair of the BPM Korea Forum. He has been in charge of the Country Chair, South Korea, and an ERC Vice Chair of the Workflow Management Coalition. He has also been on the editorial board of the KSII journal and a committee member of several conferences and workshops. He became the Chairperson of the Korea Internet Information Society as of 2023.

• • •