

Received August 15, 2021, accepted October 6, 2021, date of publication October 8, 2021, date of current version October 19, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3119011

# $\rho$ -Algorithm: A SICN-Oriented Process Mining Framework

KYOUNG-SOOK KIM<sup>1</sup>, DINH-LAM PHAM<sup>1,2</sup>, AND KWANGHOON PIO KIM<sup>1,3</sup>, (Member, IEEE)

<sup>1</sup>Contents Convergence Software Research Institute, Kyonggi University, Suwon, Gyeonggi 16227, Republic of Korea

<sup>2</sup>Information Technology Institute, Vietnam National University Hanoi, Hanoi 100000, Vietnam

<sup>3</sup>Data and Process Engineering Research Laboratory, Division 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 Basic Science Research Program through the National Research Foundation of Korea (NRF) by the Ministry of Education under Grant NRF-2020R1A6A1A03040583.


**ABSTRACT** This paper devises an algorithmic process mining framework characterized by the mathematical process model of structured information control nets (SICN) and the concept of mass-driven  $\rho$ -function as a decision-making criterion of structural process patterns. In order to prove the functional correctness of the proposed algorithmic framework, this paper also implements all the related algorithms as a process mining system and carries out an operational experiment on a typical synthetic dataset of process enactment event logs prepared and released in the 4TU Centre for Research Data. The core contribution of the paper is just the algorithmic framework development named as the  $\rho$ -Algorithm, which ought to be a novel approach not only for mining all the primitive process patterns, such as linear (sequential), disjunctive (selective-OR), conjunctive (parallel-AND), and repetitive (iterative-LOOP) process patterns, with perfectly keeping the structural properties of matched pairing and proper nesting, but also for reasonably discovering structured (even unstructured) information control nets from such IEEE XES-formatted datasets of process enactment event logs. The mining functionality of the  $\rho$ -Algorithm is made up of three stepwise algorithms: STEP-1, STEP-2 and STEP-3 algorithms, and these algorithms are formally described as an algorithmic framework supported by the conceptual process mining architecture with a series of theoretical concepts with the temporal work-case model and the temporal loop-case model. Finally, we validate the functional correctness as well as the discovery perfectness of the proposed algorithmic framework named as  $\rho$ -Algorithm by deploying the implemented  $\rho$ -Algorithm on a synthetic, non-noise and IEEE XES-formatted dataset of process enactment event logs recorded from the 10,000 work-cases with 113 activities of the Petrinet-oriented process model named as the Large Bank Transaction Process Model.

**INDEX TERMS** Structured information control net, process mining, process reengineering, process analyzing, process discovery and rediscovery, process enactment event log datasets.

## I. INTRODUCTION

A workflow process (or business process, from now on, which is named as process) management system (WPMS or BPMS) is defined as a system that partially or fully automates the definition, creation, execution, and management of work procedures through the use of software that is able to interpret the procedure definition, interact with business-task participants, and invoke their uses of IT tools and applications. Steps of a work procedure are called activities, and jobs or transactions that flow through the system are called work-cases or process

instances. One of the recent research issues in the process management literature is the process intelligence coping with the process mining [1]–[4] and its related process-aware knowledge [5]–[10] *discovery* and *rediscovery*. The motivation of the paper is to develop an algorithmic process rediscovery framework (which is called a SICN-oriented process mining framework in this paper), in particular, that becomes an effective catalytic means for successfully accomplishing not only the organizational change management works but also the process automation works in all the phases of the process life-cycle, continuously. At this moment, we have to emphasize that the rediscovered process model ought not to be equivalent to its original process model in terms of its

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

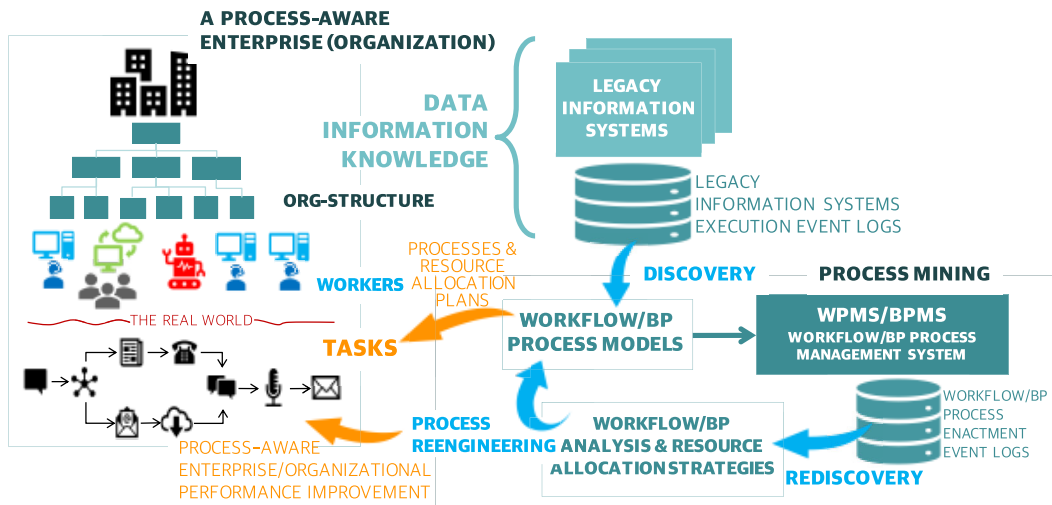


FIGURE 1. A conceptual definition of process-aware knowledge discovery and rediscovery.

process aspect itself as well as almost all the other aspects like process resource allocation aspect, performer behavioral aspect and process relevant data aspect.

Figure 1 is to illustrate a situational view of a certain process-aware enterprise (or organization) to define the conceptual difference between process-aware knowledge discovery and rediscovery. As you see, we assume that the enterprise has been supported from a variety of information systems that manage all the valuable assets of data, information and knowledge available and produced in the business and organizational environments. One of the recent trends in the traditional business and managerial information systems ought to be the introduction of process management systems and the deployment of the process automation methodologies as a means of monitoring and controlling business-activities and managements. We call those enterprises and organizations fortified with process automation and management methodologies, tools and systems as the process-aware enterprises and organizations, in general. The core competitiveness of the process-aware enterprises and organizations ought to be on the process mining functionality that is able to continuously manage as well as repeatedly evolve all the processes deployed in the corresponding enterprises and organizations. The process mining functionality is mainly composed of the process discovery [7], [11], [12] functionality and the process rediscovery [1], [5], [13], [14] functionality. The former is to discover business-activity processes from the event logs of the executions of the traditional information systems, while the later is to rediscover the enacted business-activity processes from the event logs stored whenever the corresponding business-activity processes are enacted and executed by their process management system. Especially, the rediscovered processes can be re-engineered and re-designed according to their performances measured and mined from the event logs by the process knowledge mining tools [15] and the

process intelligence solutions as well. Note that we won't, from now on, differentiate the terminology of discovery from the rediscovery in this paper.

Particularly, the paper proposes an algorithmic process mining framework for discovering structured process patterns from process enactment event logs and validates the proposed algorithmic framework by implementing all the related stepwise algorithms. Also, it carries practically out an experimental analysis by deploying the implemented framework onto the dataset provided by the BPI challenges of 4TU.Centre for Research Data [16] and formatted in the IEEE XES standardized format [17]. The following are the characteristics of the algorithmic process mining framework proposed in the paper. First, the algorithmic framework is able not only to discover the process patterns [18] but also to discover the enactment occurrences [19], [39] of the process patterns from a dataset of the IEEE XES-formatted enactment event logs of a corresponding process model. Second, the algorithmic framework is theoretically supported by the information control nets modeling methodology [20] of process models. Third, the essential algorithm of the algorithmic framework named as  $\rho$ -Algorithm (rho-Algorithm) is able to discover a structured information control net model with the enactment occurrences of the activities associated with an underlying process model. Fourth, the  $\rho$ -Algorithm is firstly developed in the process management and mining literature as the process mining algorithm that discovers a structured process model theoretically supported by the structured information control net modeling methodology. Fifth, the  $\rho$ -Algorithm is able to discover all the process patterns such as linear (sequential), conjunctive (parallel-AND), disjunctive (exclusive-OR), and repetitive (iterative-LOOP) process patterns and discover the enactment occurrences and proportions of each branch of the process patterns.

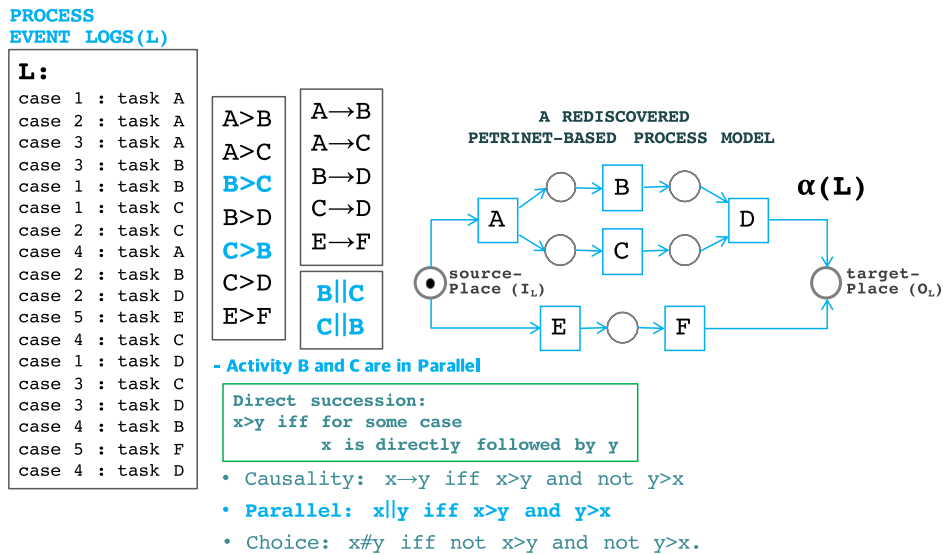


FIGURE 2. An operational verification of discovering the petrinet-oriented process model by the  $\alpha$ -algorithm.

As the validation study of the proposed algorithmic framework ( $\rho$ -Algorithm), the paper applies the implemented framework and system to a real dataset that contains non-noise and synthetic event logs from the enactment history of 10,000 process instance event traces of the Large Bank Transaction Process Model [16]. Through the feasibility study, the paper verifies the functional correctness of the  $\rho$ -Algorithm in discovering all the types of process patterns from the real dataset formatted in the IEEE XES standardized specification [17].

In terms of organizing the paper with the sections, the next section summarizes the related research works and scope as the algorithmic challenges of the conventional process discovery algorithms done in the business and workflow process intelligence literature. In the consecutive section, we describe the theoretical and algorithmic details of devising the  $\rho$ -Algorithm with three stepwise algorithms under the name of the SICN-oriented process mining framework. Subsequently, the implementation and operational details of validating the core functions of the  $\rho$ -Algorithm are explicated with focusing on how it works for discovering all the structural process patterns and finally building up a structured information control net from an exemplary dataset as well as the synthetic and non-noise dataset with the 10,000 instances' enactment histories of the Large Bank Transaction Process Model, respectively. Finally, the last section summarizes our research outcomes and experimental validation results with concluding our future works and remarks.

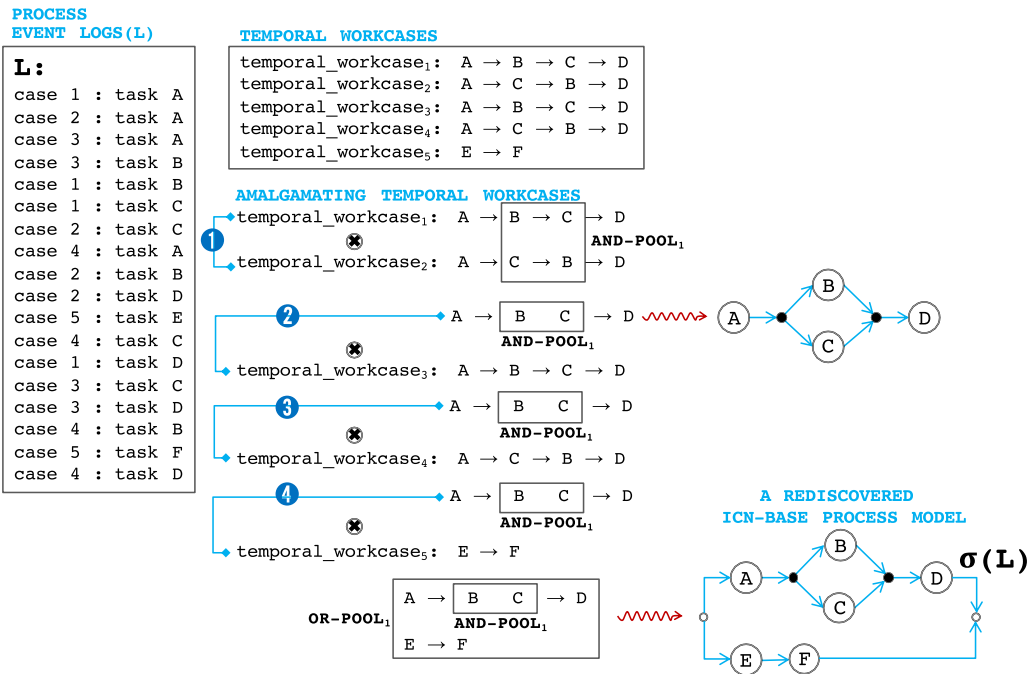
## II. RELATED WORKS AND MOTIVATION

The main research challenge of this paper is to devise an algorithmic process mining framework for discovering a model of structured information control nets from a process enactment event log dataset. Therefore, the literature surveys of the

challenge are summarized in this section of related works. The most popular approaches of the theoretical modeling methodologies to formally define and graphically represent business process models are the Petrinet-oriented model [21]–[23] of process modeling methodology and the information control net model [10], [24]–[26] of process modeling methodology. Both of them are based upon the mathematical representation and the graphical representation at the same time. The Petrinet-oriented process model has a strong advantage in terms of the mathematical and analytical power, whereas the information control net process model has a much stronger merit in terms of the expressiveness of the process domain. So far, there have been several process mining algorithms in the literature. One of the typical process mining algorithms for discovering the Petrinet-oriented process models is the  $\alpha$ (alpha)-Algorithm [13], [27], whereas the typical process mining algorithm for discovering the information control net process models is the  $\sigma$ (sigma)-Algorithm [1]. Note that the name of the process mining algorithm proposed in the paper is the  $\rho$ (rho)-Algorithm, and the naming reason of the  $\rho$ -Algorithm will be explained later. The crucial idea and characteristics of these algorithms and their comparisons with the  $\rho$ -Algorithm are arranged in this section.

### A. THE $\alpha$ -ALGORITHM FOR DISCOVERING THE PETRINET-ORIENTED PROCESSES

Through publishing the  $\alpha$ -Algorithm, Aalst *et al.* [13] firstly addressed the process mining problem. This problem was formulated as follows: Find a mining algorithm able to discover a large class of sound Petrinet-oriented process model on the basis of complete process event logs. The  $\alpha$ -Algorithm is able to discover a large and relevant class of structured processes. Through examples they showed that the algorithm



**FIGURE 3.** An operational verification of discovering the structured information control net process model by the  $\sigma$ -algorithm.

provides interesting analysis results for processes and tackled the problem of short loops with focusing on hidden tasks, duplicate tasks, and advanced routing constructs. The discovery problem is not a goal by itself. They also assumed that the overall goal is to be able to analyze any process event logs without any knowledge of the underlying process and in the presence of noise. In a common sense in the literature, their works were accepted as a stepping stone for good and robust process mining techniques. Figure 2 shows the three types of Petrinet-oriented process patterns that can be discovered from the process event logs by the  $\alpha$ -Algorithm, and the log-based ordering relations, such as causality, parallel and choice, that are used for deciding the type of process patterns by the  $\alpha$ -Algorithm, respectively. Based upon these ordering relation operators, they applied the  $\alpha$ -Algorithm to an exemplary dataset of the process enactment event logs as depicted in Figure 2, and proved that the  $\alpha$ -Algorithm is reasonable and applicable in the real process intelligence arena.

**B. THE  $\sigma$ -ALGORITHM FOR DISCOVERING THE INFORMATION CONTROL NET PROCESSES**

Through publishing the  $\sigma$ (sigma)-Algorithm, Kim and Ellis [1] proposed a mining algorithm for discovering process models from the underlying process enactment event logs. Differently from the  $\alpha$ -Algorithm dealing with the Petrinet-oriented process modeling methodology as introduced in the previous section, the  $\sigma$ -Algorithms theoretical basis is the structured information control net process modeling methodology. They understood process management systems as process-aware information systems that help

to execute, monitor and manage work process flow and execution. These systems, as they are executing, keep a record of who does what and when (e.g. log of events). Also they defined the basic concept of the process mining as those activities of using computer software to examine these records and deriving various structural data results. They thought that the process mining activity needs to encompass behavioral (process/ control-flow), social, informational (data-flow), and organizational perspectives. Especially, they were insisted that process management systems are people systems that must be designed, deployed, and understood within their social and organizational contexts. The process mining activity ought to be planned and fulfilled under the preconditions of the people systems, too. They especially focused on the behavioral perspective of a structured process model that preserves the proper nesting and the matched pair properties and proposed a process mining algorithm that is able to discover a structured information control net process model, which is named as  $\sigma$ -Algorithm. The main reason of naming the  $\sigma$ -Algorithm is because it is incrementally amalgamating a series of temporal work-cases (process instance event traces) according to three types of basic merging principles. Figure 3 shows how the  $\sigma$ -algorithm works with temporal work-cases by illustrating an example of the information control net process discovery using the  $\sigma$ -Algorithm.

**C. THE MOTIVATION OF THE  $\rho$ -ALGORITHM**

The essential motivation of the paper is about the proof of concept and feasibility with the SICN-oriented process mining framework that is able to discover a process model

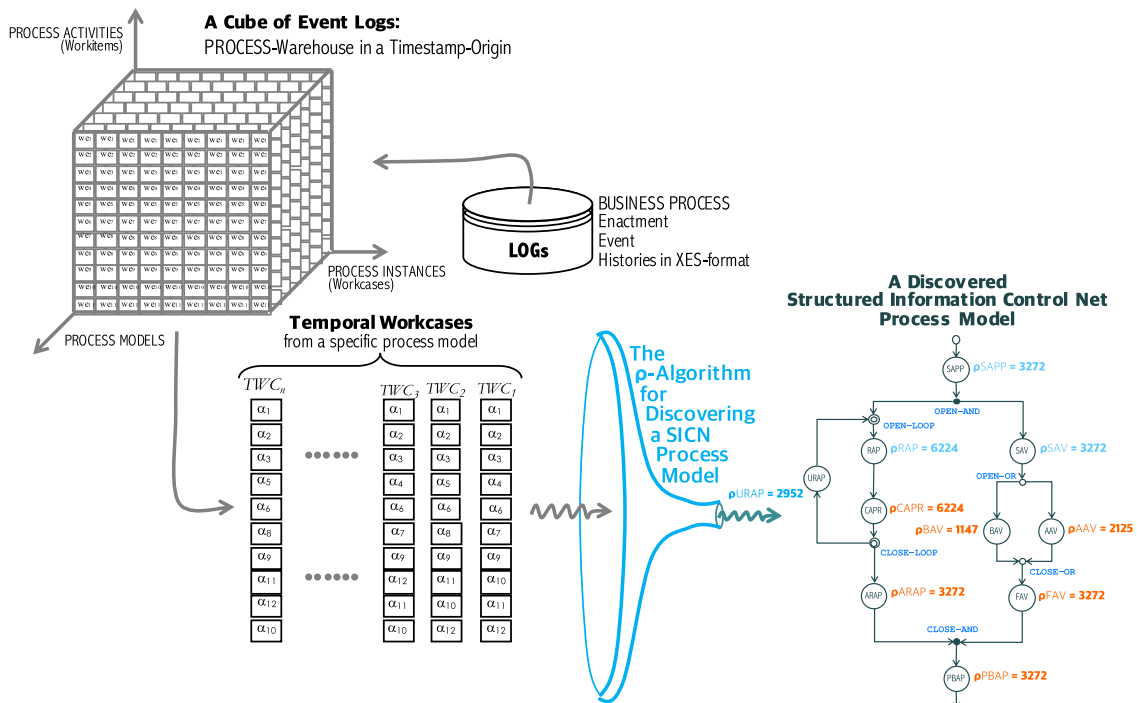


FIGURE 4. The conceptual process mining architecture for discovering structured information control net process models.

of structured information control net from such a dataset of process enactment event logs formatted in IEEE XES standardized specifications. As stated in the previous subsections, for the sake of eventually supporting the model-log comparison, which is the process fidelity issue [13], [22], [28]–[30], the literature has produced so far the two concepts and theories; one is the Petri net-oriented process modeling methodology and the other is the information control net process modeling methodology. Also, the literature has published the  $\alpha$ -Algorithm [2], [30] and the  $\sigma$ -Algorithm [1] for discovering Petri net-oriented process models and information control net process models, respectively. However, all of these discovery approaches have the limitations that they are able to deal with only sequential, parallel and selective process patterns out of all the types of the process patterns such as sequential, parallel, selective and repetitive process patterns. In other words, both of the discovery algorithms have a limitation in dealing with discovering the repetitive-LOOP process patterns; the  $\alpha$ -Algorithm, in principle, is implicitly dealing with the repetitive-LOOP process pattern as the selective process pattern. In recent, some [25], [30]–[33] of the research groups tried to not only extend the  $\alpha$ -Algorithm and the  $\sigma$ -Algorithm so as to explicitly deal with the repetitive-LOOP process patterns, but also expand the process mining usages to the predictive process managing and monitoring area.

### III. DEVISING THE $\rho$ -ALGORITHM

In this section, a SICN-oriented process mining framework is proposed and formalized with a series of formal concepts

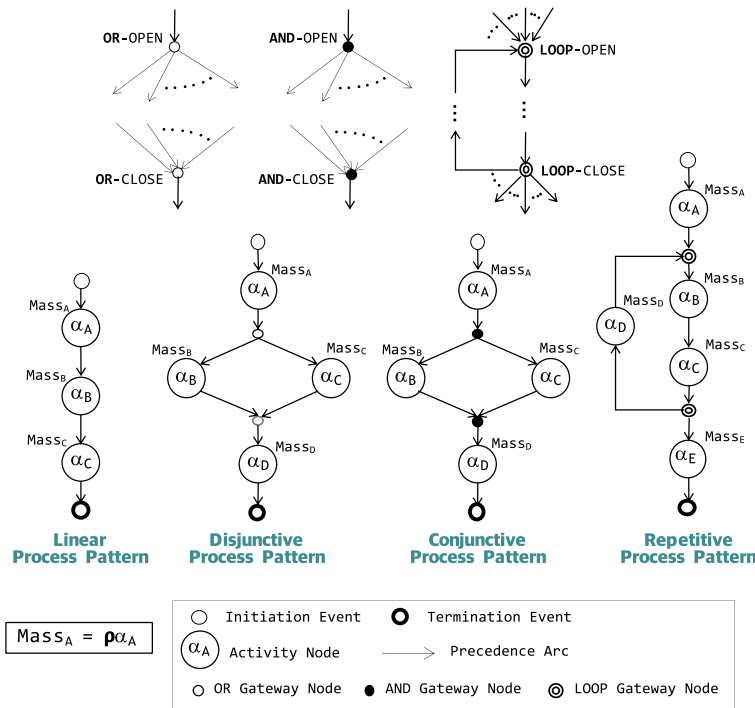
and models. Especially the proposed algorithmic framework is characterized by the concept of mass-driven  $\rho$ -function as a decision-making criterion of process patterns. In other words, the proposed algorithmic framework is able to eventually discover all the four types of primitive process patterns in a structured information control net process model from a process enactment event log dataset by applying the mass-driven  $\rho$ -function with their enactment event log occurrences. In this section we formally describes a series of conceptual components of the conceptual process mining architecture at first. Next, we also devise an algorithmic process mining framework with a series of algorithms for transforming and discovering all the process patterns and their enactment event log occurrences. Especially, the core component of the devised algorithmic framework is the  $\rho$ -Algorithm,<sup>1</sup> which is firstly introduced and so named by this paper, and the  $\rho$ -operator of which gives the masses (occurrences) of all the activities in the event log dataset of a corresponding process model.

#### A. THE CONCEPTUAL PROCESS MINING ARCHITECTURE

In this section, we explicate the conceptual approach with formal notations for devising the  $\rho$ -Algorithm proposed in the paper, which is to formally define a conceptual process

<sup>1</sup>In the  $\rho$ -Algorithm, the symbol and name of rho ( $\rho$ ) comes from the A programming language (APL) firstly released in 1960s. The function rho, coded like  $\rho X$  in APL, implies that it gives the number of elements in X, from which the concept of *mass* comes. The central idea of the discovery algorithm of the framework is exactly same to the implication of the APL function, rho ( $\rho$ ).





**FIGURE 5.** Four types of process patterns in the structured information control net modeling methodology.

mining architecture to discover a structured information control net process model from its enactment event histories logged by executing the corresponding process model built from any combinations of all the four types of primitive process patterns. Figure 4 illustrates a series of formal components of the conceptual process mining architecture, which is revised from the original architecture [1] to emphasize the proposed algorithm’s conceptual requirement with discovering all the possible combinations of the four types of primitive process patterns [18], [34], such as linear, disjunctive (exclusive-OR), conjunctive (parallel-AND) and repetitive (iterative-LOOP) process patterns, with their graphical representations given in Figure 5.

The concrete algorithm is theoretically supported by a series of functional routines that are formalized by a series of formal concepts and their related algorithms, such as process enactment event log datasets, process warehouses, temporal work-cases, temporal loop-cases, temporal work-case model, temporal loop-case model and a process pattern rediscovery algorithm named as the  $\rho$ -Algorithm. Especially, the functional core of the  $\rho$ -Algorithm is to receive a bunch of temporal work-cases as inputs and to discover a structured information control net as an output. At the moment of composing a structured information control net as the output, the  $\rho$ -Algorithm has to precisely situate three types of special nodes, such as selective-XOR, parallel-AND and iterative-LOOP nodes, between those activity nodes having multiple incoming or outgoing transition relationships, as shown in Figure 5. Besides, these three types of special

nodes are called gateway activity nodes; each type of the special nodes has to match a pair of OPEN and CLOSE gateway activities; simultaneously, the rule of properly nested formation must be kept so that multiple sets of the OPEN and CLOSE gateway activities are theoretically satisfied with the principles of safeness and soundness. Eventually, the discovered process model is graphically represented by the control flow aspect of the information control net that is built from a starting node, a terminating node, the enacted activities with their nodes, and the edges combining all these nodes with a set of sequential node, a set of pairs of OPEN and CLOSE parallel gateway nodes, a set of pairs of OPEN and CLOSE selective gateway nodes, and a set of pairs of OPEN and CLOSE iterative gateway nodes. Likewise, it is necessary to be graphically visualized as a graphical process model with keeping the structural properties of matched-pairing and proper-nesting according to the following graphical composition rules:

- Primitive OPEN-Transition (Open or Split Gateway Activity) Types: The conjunctive (or parallel) open gateway activity is graphically represented by a solid dot (●) with a single incoming transition and multiple outgoing transitions; the disjunctive (or selective-decision) open gateway activity is graphically represented by a hollow dot (○) with a single incoming transition and multiple outgoing transitions; the repetitive open gateway activity is graphically represented by a double-hollow dot (⊙) with multiple incoming transitions and a single outgoing transition.

- Primitive CLOSE-Transition (Close or Join Gateway Activity) Types: The primitive Close-transition types have the same graphical notations, but their connections *vice versa*.
- The starting and the terminating event nodes are medium-sized circle with thin line and medium-sized circle with thick line, respectively.

The conceptual architecture is concretized by a series of formal concepts and their algorithms, such as process enactment event log datasets, process warehouses, temporal work-cases, temporal loop-cases, temporal work-case model, temporal loop-case model and a process pattern discovery algorithm named as the  $\rho$ -Algorithm. Especially, the figure highlights the conceptual role of the  $\rho$ -Algorithm that is the core component of the conceptual architecture. This section formally defines these formal components.

### 1) PROCESS ENACTMENT EVENT LOGS

On the conceptual approach of Figure 4, the starting point is the process enactment event logs that are organized into a cubic structure of the *process enactment event logs* of each process package, the accumulated *process instance enactment event traces* of each process model and the *process work-item enactment events* of each process instance. In other words, the event logs imply the enactment histories of all the process models (or packages) managed by an underlying process management system. According as an instance of a process model executes, its temporal execution sequence is produced and logged into the backup databases or files; this temporal execution sequence is called a process enactment event trace. The process enactment event trace is made up of a temporal sequence of the activity event logs associated with a specific process instance. According as a process instance is executed, the logging and auditing component of the process enactment engine records its work-item (activity instance) execution events on a log repository, and those logged events are arranged in a form of temporal sequence of events. This execution sequence of a process instance is forming a process instance event trace. Here, we start to formally describe the conceptual architecture from defining a formal structure of the process enactment event logs with its three dimensional cube as defined in the following consecutive definitions from *Definition 1* through *Definition 4*.

*Definition 1: Process Work-Item Enactment Event.* Let  $pe = (\alpha, pc, wf, wc, ac, p^*, t, s)$  be a process work-item enactment event stored as logs, where

- $\alpha$  is a work-item (activity instance) identifier,
- $pc$  is a package identifier,
- $wf$  is a process identifier,
- $wc$  is a process instance (work-case) identifier,
- $ac$  is an activity identifier,
- $p^2$  is a participant (or performer) identifier,
- $t$  is a timestamp, and

<sup>2</sup>Note that \* indicates multiplicity.

- $s$  is a work-item's current state, which is one of the states such as *ready*, *assigned*, *reserved*, *running*, *completed*, and *cancelled*.

*Definition 2: Process Instance Enactment Event Trace.* Let  $PET(c)$  be the process instance event trace of a work-case,  $c$ , where  $PET(c) = (pe_1, \dots, pe_n)$ , where  $\{pe_i \mid pe_i.wc = c \wedge pe_i.t \leq pe_j.t \wedge pe_i.pc = pe_j.pc \wedge pe_i.wf = pe_j.wf \wedge pe_i.wc = pe_j.wc \wedge i < j \wedge 1 \leq i, j \leq n\}$ , which formally represents a temporally ordered work-activity event sequence of a specific work-case, which is built through preprocessing the process enactment event logs by considering the `TIMESTAMP` and the `STATE` attributes.

*Definition 3: Process Enactment Event Log.* Let  $PL(I_i)$ ,  $I_i = \{c_1^i, \dots, c_m^i\}$ , be a process enactment event log with a set of completed process instances ( $m$  is the number of the process instances) that have been instantiated from a process model,  $I_i$ .

*Definition 4: Process Package Cubic Warehouse.* Let  $PW$  be a process package cubic warehouse consisting of a set of process enactment event logs,  $PL(I_1), \dots, PL(I_n)$ , where  $PL(I_i) = \forall PET(c^i \in I_i)$ , and  $n$  is the number of process models in a process package managed in a system.

Basically, we build a series of the formal models of a process package cubic warehouse,  $PET$ ,  $PL$  and  $PW$ , so far. Importantly, we remind that a process instance enactment event trace is composed of a temporally ordered group of the work-item enactment events having the same process instance identifier. Note also that the meaningful temporal order in managing process instances (work-cases) ought to be based upon one of the following instantaneous points of time, each of which is named as a *timestamp-origin*. Accordingly, we necessarily build several different types of process package cubic warehouses based upon the type of timestamp-origins as follows:

- The Scheduled Point of Time: the event's timestamp is taken at when the state of a work-item is changed from `READY`<sup>3</sup> to `ASSIGNED`<sup>4</sup>: a work-activity event log with *scheduledTimestamp*,  $we^{t.s} \Rightarrow (t = we.t \wedge s = we.s \wedge s = \text{"assigned"})$
- The Assessed Point of Time: the event's timestamp is taken at when the state of a work-item is changed from `ASSIGNED` to `RESERVED`<sup>5</sup>: a work-activity event log with *assessedTimestamp*,  $we^{t.e} \Rightarrow (t = we.t \wedge e = we.s \wedge e = \text{"reserved"})$
- The Started Point of Time: the event's timestamp is taken at when the state of a work-item is changed from `RESERVED` to `RUNNING`<sup>6</sup>: a work-activity event log

<sup>3</sup>The `READY` state of a work-item implies that the work-item is ready to be processed but has not been assigned to a particular participant.

<sup>4</sup>The `ASSIGNED` state of a work-item implies that the work-item has been assigned to a role (potentially a group of participants), but work has not started yet.

<sup>5</sup>The `RESERVED` state of a work-item implies that the work-item has been assigned to a named user (a single participant), but work has not started yet.

<sup>6</sup>The `RUNNING` state of a work-item implies that the work-item is actively being worked on, and time spent in this state would be recorded as processing time or work time.

with *runningTimestamp*,  $we^{t,u} \Rightarrow (t = we.t \wedge u = we.s \wedge u = \text{"running"})$

- The Completed Point of Time: the event's timestamp is taken at when the state of a work-item is changed from RUNNING to COMPLETED<sup>7</sup>: a work-activity event log with *completedTimestamp*,  $we^{t,o} \Rightarrow (t = we.t \wedge o = we.s \wedge o = \text{"completed"})$

## 2) TEMPORAL WORK-CASES AND MODELS

In order to develop a formal algorithm for discovering a structured information control net model from the process enactment event logs, we need to build a formal concept and model of temporally ordered activity event sequences of process instances, which can be extracted from the formal model of process instance enactment event traces, *PET*(*c*). The formal model of temporally ordered activity event sequences is named as *temporal work-cases*. At this moment, we have to remind that the meaningful temporal order in managing process instances ought to be based upon one of the *timestamp-origins* like scheduled, assessed, started and completed timestamps. Accordingly, we necessarily build four species of temporal work-cases with holding one of the timestamp-origins. Figure 6 illustrates the formal concepts and models of the temporal work-cases including the temporal loop-cases associated with the repetitive-LOOP process patterns. The formal definition of the temporal work-cases is as follows:

*Definition 5: Temporal Work-case.* Let *TWC*(*c*) be a temporal work-case of the activity event sequence of a specific process instance, *c*:

- $TWC(c) = (we_{\alpha_1}^{\tau[\phi]}, \dots, we_{\alpha_m}^{\tau[\phi]})$ ,  
where  $\{we_{\alpha}^{\tau[\phi]} \mid \alpha = we.ac \wedge \tau = we.t \wedge \phi \in \{s, e, u, o\} \wedge we_{\alpha}.wc = c \wedge (we_{\alpha_i}^{\tau_i} < we_{\alpha_j}^{\tau_j})^8 \wedge \tau_i < \tau_j \wedge i < j \wedge 1 \leq i, j \leq m\}$ ,

which is a temporally ordered work-activity event sequence along with one of the timestamp-origins. Especially, each temporal work-case is formally defined as a temporal work-case model, and it is assumed that all the work-items (work-activity instances) in the corresponding temporal work-case are successfully completed, and their executions are running without being suspended, as well.

Based on *Definition 5*, we can interpret the formal definition as the conceptual implication that all of the work-activity events holding an identical INSTANCE ID are lined up in a temporal work-case along with its timestamp-origin. Consequently, from a process instance event trace, we produce a corresponding temporal work-case by extracting the activity identifiers and their timestamps, and we name it as a temporal work-case along with one of the timestamp-origins like the scheduled time, assessed time, started time, and completed time. For the sake of the formal representation, a temporal work-case is defined as a temporal work-case model as

<sup>7</sup>The COMPLETED state of a work-item implies that the work-item has been fully executed and completed with either success or failure.

<sup>8</sup> $we_{\alpha_i}^{\tau_i}$  is the predecessor of  $we_{\alpha_j}^{\tau_j}$  in the list of a temporal work-case.

formally described in *Definition 6*. Accordingly, there possibly exist four species of the temporal work-cases and their models, as follows:

- ScheduledTime Temporal Work-case Species and Model
- AssessedTime Temporal Work-case Species and Model
- StartedTime Temporal Work-case Species and Model
- CompletedTime Temporal Work-case Species and Model

*Definition 6: Temporal Work-case Model.* A temporal work-case model is formally defined through 3-tuple  $TWCM = (\omega, F_r^c, T_o^c)$  over a set **A** of activity trace-nodes,  $\forall \eta_{\alpha}^{\tau[\phi]}$ , on a temporal work-case, *TWC*(*c*), of a process instance, *c*, and a species **K** ( $= \{s, e, u, o\}$ ) of the timestamp-origins, where

- $F_r^c$  is an activity or an activity-group linked from an external temporal work-case model;
- $T_o^c$  is an activity or an activity-group linked to an external temporal work-case model;
- $\omega = \omega_i \cup \omega_o$  on  $\forall \eta_{\alpha}^{\tau[\phi]} \in \mathbf{A}$ ,
  - $\omega_o : A \rightarrow \wp(A)$  is a single-valued mapping function of an activity event node,  $\eta_{\alpha}^{\tau[\phi]} = we_{\alpha}^{\tau[\phi]} \wedge \phi \in \mathbf{K}$ , to its (immediate) successor in a temporal work-case;
  - $\omega_i : A \rightarrow \wp(A)$  is a single-valued mapping function of an activity event node,  $\eta_{\alpha}^{\tau[\phi]} = we_{\alpha}^{\tau[\phi]} \wedge \phi \in \mathbf{K}$ , to its (immediate) predecessors in a temporal work-case.
- The species of temporal work-case models:  $TWCM^{\phi}$ 
  - ScheduledTime Temporal Work-case Model:  $\phi = 's'$  in  $\forall \eta_{\alpha}^{\tau[\phi]}$  of a temporal work-case model
  - AssessedTime Temporal Work-case Model:  $\phi = 'e'$  in  $\forall \eta_{\alpha}^{\tau[\phi]}$  of a temporal work-case model
  - StartedTime Temporal Work-case Model:  $\phi = 'u'$  in  $\forall \eta_{\alpha}^{\tau[\phi]}$  of a temporal work-case model
  - CompletedTime Temporal Work-case Model:  $\phi = 'o'$  in  $\forall \eta_{\alpha}^{\tau[\phi]}$  of a temporal work-case model

The formal concept and the theoretical model of a temporal loop-case are given in *Definition 7* and *Definition 8*, respectively. As shown in Figure 6, those temporal loop-cases are extracted from their corresponding process enactment events, too. For instance,  $\{\alpha_{11}, \alpha_{12}, \dots, \alpha_{1s(1)}\}$  is a temporal loop-case having an identical work-case identifier with others of the *bag* and being temporally ordered activity events with their timestamp-origin properties, because each temporal loop-case represents each of the iterative work-activities involved in a loop transition construct. Therefore, two or more identical temporal loop-cases can be iterated in a temporal work-case according to the number of iterations of a corresponding loop transition. In addition, we have to consider a work-case of nested loop transitions, and then its temporal loop-cases have to be unfolded with multiple temporal loop-cases from its inner loop transitions, recursively.

*Definition 7: Temporal Loop-case.* Let *TLC*(*c*) be the temporal loop-case of one of the work-activity iteration event



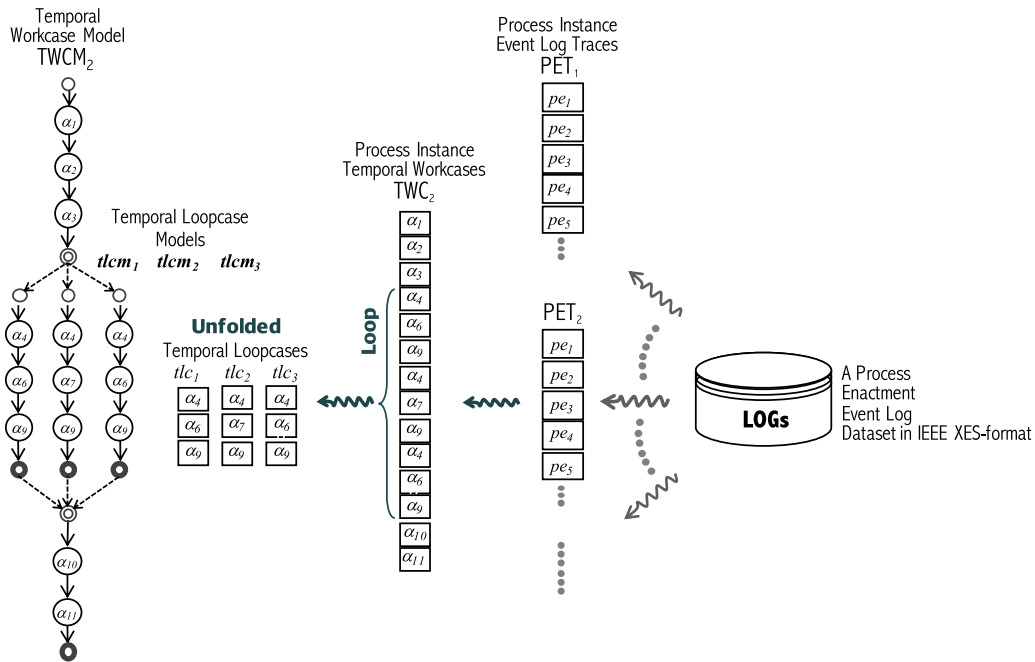


FIGURE 6. The formal concepts and models of the temporal work-cases and temporal loop-cases.

traces in the temporal work-case of a corresponding process instance,  $c$ :

$$TLC(c) = \{ \{ \alpha_{11}, \alpha_{12}, \dots, \alpha_{1s(1)} \}, \{ \alpha_{21}, \alpha_{22}, \dots, \alpha_{2s(2)} \}, \dots, \{ \alpha_{k1}, \alpha_{k2}, \dots, \alpha_{ks(k)} \} \}^*$$

where,  $\{ \alpha_{ij} \mid \alpha_{ij}.c = c \wedge \alpha_{ij}.t < we_{mn}.t \wedge (i < m \vee (i = m \wedge j < n)) \wedge 1 \leq i, m \leq k \wedge 1 \leq j, n \leq \forall [s(1), \dots, s(k)] \}$ ,

which is a bag of temporally ordered sub-sequences of work-activity events according to the timestamp-origin property. A work-activity iteration event trace is called a temporal loop-case.

**Definition 8: Temporal Loop-case Model (TLCM).** A temporal loop-case model is formally defined through 3-tuple  $L = (\omega, P, S)$  over a bag of iterative activities' ( $\forall \alpha \in A$ ) event logs on a process trace, where

- $P$  is a predecessor activity of the first activity of the loop-case model folded out from the first iteration;
- $S$  is a successor activity of the last activity of the loop-case model folded out from the last iteration;
- $\theta = \theta_i \cup \theta_o$ ,

where,  $\theta_o : A \rightarrow \wp(\alpha \in A)$  is a single-valued mapping function of a work-activity event log to its (immediate) successor in a work-activity iteration trace with one of the timestamp-origin types, and  $\theta_i : A \rightarrow \wp(\alpha \in A)$  is a single-valued mapping function of a work-activity event log to its (immediate) predecessors in a work-activity iteration trace with one of the timestamp-origin types.

In the formal approach of Figure 4, there are four different process patterns expected to be discovered from a bunch of temporal work-cases associated with a specific process model through the formal discovery algorithm of the  $\rho$ -Algorithm. Also, there are four different temporal work-case types forming the temporal work-cases that are characterized by the types of timestamp-origins such as scheduled, assessed, started, and completed timestamps held by their work-activity event logs. The types of temporal work-cases are differentiated from each other according to such temporal information, *i.e.* the activity execution (work-item) events' timestamps, logged at the time when the corresponding activities work-items were performed. The enactment engine of a process management system maintains the four types of timestamp-origins when it stores the enactment event timestamps of activity execution work-items. Accordingly there possibly exist four types of temporal work-cases in the process enactment event log traces at the same time, and from which there also possibly exist four types of process warehouses, too. The concepts and their related algorithms in this paper do not differentiate from these specific types of the temporal work-case models in detail, and assume that the type of the completed timestamp-origin is the most appropriate timestamp type in terms of forming a dataset of process enactment event log traces.

### B. $\rho$ -ALGORITHM: THE PROPOSED ALGORITHMIC FRAMEWORK FOR DISCOVERING STRUCTURED INFORMATION CONTROL NETS OF PROCESS MODELS

The algorithmic process mining framework is based mainly upon the detailed implementation of the  $\rho$ -Algorithm simply stated in the conceptual architecture in the previous section.

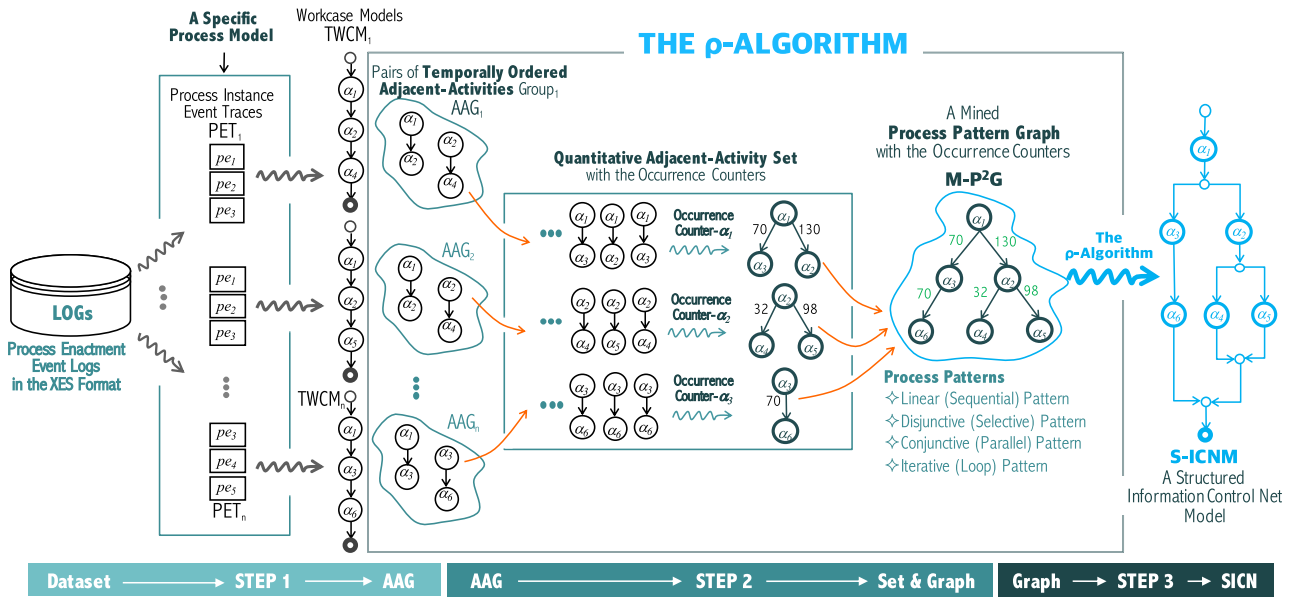


FIGURE 7.  $\rho$ -algorithm: the algorithmic process mining framework for discovering sicn-oriented process models.

As shown in Figure 7, the algorithmic framework is composed of seven procedural concepts and six transformation algorithms, each of which conducts a transformation from one concept to another concept, one after another, to support the procedural process discovery and process-aware knowledge mining experiments. These concepts are procedurally started from process enactment event log histories formatted in the IEEE XES standard: process instance event traces, temporal work-case models, pairs of temporally ordered adjacent-activity groups, weighted adjacent-activity set, weighted process pattern graph, and structured information control net process model, whereas the transformation algorithms are listed as a series of the functional analytics algorithms: the event trace mining algorithm, temporal work-case composing algorithm, adjacent-activity fragmentizing algorithm, adjacent-activity quantifying algorithm, process pattern discovering algorithm, and the structured information control net discovering algorithm. In this section, we axiomatically formalize the algorithmic framework with its core concepts and algorithms related with the  $\rho$ -Algorithm in particular.

### 1) DATA FORMATS FOR THE $\rho$ -ALGORITHM

In terms of the event format, we consider the work-item event being stored in a tag-based language (XML schema). An XML-based process work-item event format, XWELL<sup>9</sup> [35], for the purpose of process mining had been studied and proposed by the authors' research group, and the WfMC (Workflow Management Coalition) has released

<sup>9</sup>XWELL stands for XML-based workflow execution logging mechanism and language.

the standardized audit and log specification, BPAF<sup>10</sup> [36]. In recent, IEEE has released a standard tag-based language, XES,<sup>11</sup> which aims at providing the designers of information systems with a unified and extensible methodology for capturing systems' behaviors by means of event logs and event streams. As the format of the process work-item event structure, we can use the "IEEE XES Schema" describing the structure of an XES event log/stream and the "XES extension" describing the structure of an extension of such a log/stream. Note that all the formal concepts and algorithms devised in the paper are assumed that the format of the process enactment event logs is the IEEE XES standardized format. We simply define the XML schema with the essential attributes as the process work-item event structure, as follows:

- The **EVENT** attribute is used to specify an event identifier, which is assigned by the process enactment engine.
- The **WORK-ITEM** attribute of a process activity-level event represents a work-item identifier that is uniquely assigned by using those combined identifiers of PACKAGE ID, PROCESS ID, ACTIVITY ID, and INSTANCE ID.
- The **PARTICIPANT** attribute is used to specify the performer who is in charge of enacting the work-item.
- The **TIMESTAMP** attribute specifies the time stamp of the event occurred.
- Finally, the **STATE** attribute represents the work-item's runtime state maintained by the engine. Whenever the work-item's state is changed, it is logged with

<sup>10</sup>BPAF stands for business process audit format.

<sup>11</sup>XES stands for extensible event stream, which is supported from the extensible event stream working group of the IEEE standard committee, ieeexes.

the event-code of WMChanged WorkitemState. It ought to be one of those states such as READY, ASSIGNED, RESERVED, RUNNING, COMPLETED, and CANCELLED.

## 2) PROCEDURAL COMPONENTS OF THE $\rho$ -ALGORITHM

The overall algorithmic framework, as illustrated in Figure 7, is a stepwise mining procedure with the functional components to be used for discovering all the types of the primitive process patterns and their enactment occurrences that eventually build a discovered process model. The algorithmic framework is supported by a series of stepwise transformation algorithms for discovering an enacted process model from a dataset of process enactment event logs. The first transformation algorithm is to discover the enacted work-cases from the dataset, each of which is modeled by a temporal work-case model. At the same time, it is necessary to count the number of occurrences of each temporal work-case in the dataset. The second transformation algorithm, which is just right the  $\rho$ -Algorithm, is to discover a process model, which is formally modeled by using the structured information control net methodology and the number of occurrences of the temporal work-cases. In terms of discovering the process model, the  $\rho$ -Algorithm is able to deal with any combinational number of AND/OR/LOOP process patterns in the structured information control net methodology. The figure shows an exemplary case of the discovered process model represented by a structured information control net consisting of two Exclusive-OR (disjunctive) process patterns.

### a: GROUPS OF TEMPORALLY ORDERED ADJACENT-ACTIVITY PAIRS

From now on, the first step of the  $\rho$ -Algorithm in the algorithmic framework is to develop an algorithm that is able to mine a group of temporally ordered adjacent-activities pairs from a temporal work-case and its formal work-case model corresponding to each of the process instance event traces. Also, each of the temporal work-cases is formally represented by one of the work-case model types formalized in the previous section. Figure 8 depicts a conceptual procedure of the STEP-1 algorithm of the  $\rho$ -Algorithm, which is related with a single process instance event trace ( $PET_1$ ). That is, a temporal work-case represents an ordered enactment sequence of work-activity event logs, each of which is formed with its work-activity identifier and its time-stamp extracted from its corresponding process enactment event log. The extracted temporal work-case ( $TWC_1$ ) is used for formally defining its work-case model ( $TWCM_1$ ), from which the STEP-1 algorithm is able to mine a group ( $AAG_1$ ) of temporally ordered adjacent-activity pairs belonging to a corresponding process instance event trace ( $PET_1$ ). Each of the adjacent-activity pairs is formally represented according to the formal definition of *Definition 9*. Finally, **Algorithm 1** in the next page algorithmically describes the pseudo-codes of the STEP-1 of the  $\rho$ -Algorithm.

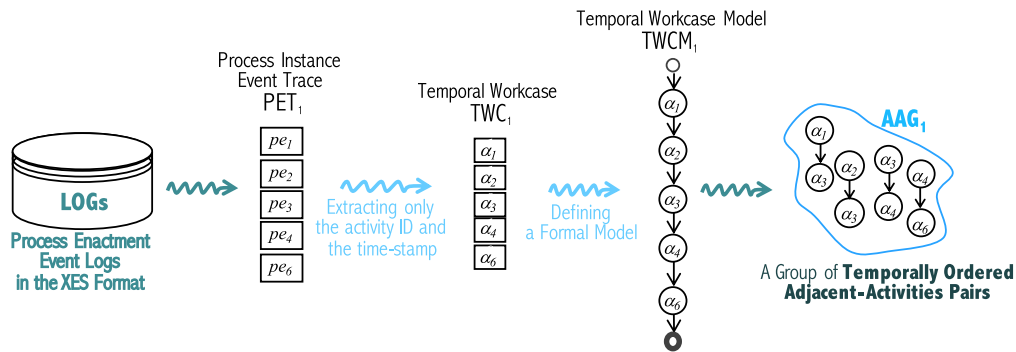
*Definition 9: Group of Adjacent-Activity Pairs.* Let  $AAG^T(c)$ ,  $s, b, o \in T$ , be a group of all the pairs of adjacent-activities in a work-case model of the process instance,  $c$ , where  $AAG^T(c) = (aa^T p_1, \dots, aa^T p_m)$ . Each adjacent-activity pair,  $aa^T p$ , is defined in to be  $(\alpha_a, \alpha_s)$ ,  $\forall \alpha \in A$  in PET, where  $\alpha_a$  and  $\alpha_s$  are adjacent,  $\{\alpha_s \in \omega_o^T(\alpha_a) \wedge \alpha_a \in \omega_i^T(\alpha_s)\}$ , each other, in a work-case model. There are three types of adjacent-activity pair according to the time-stamp type:

- ScheduledTime Adjacent-Activity Pair ( $\mathbf{T} = s$ )  
 $\{aa^s p = (a_a, a_s) \mid a_a.c = c \wedge a_s.c = c \wedge a_a.e = 'ScheduledTime' \wedge a_s.e = 'ScheduledTime' \wedge a_a.t \leq a_s.t \wedge a < s \wedge 1 \leq a, s \leq n\}$
- StartedTime Adjacent-Activity Pair ( $\mathbf{T} = b$ )  
 $\{aa^b p = (a_a, a_s) \mid a_a.c = c \wedge a_s.c = c \wedge a_a.e = 'StartedTime' \wedge a_s.e = 'StartedTime' \wedge a_a.t \leq a_s.t \wedge a < s \wedge 1 \leq a, s \leq n\}$
- CompletedTime Adjacent-Activity Pair ( $\mathbf{T} = o$ )  
 $\{aa^o p = (a_a, a_s) \mid a_a.c = c \wedge a_s.c = c \wedge a_a.e = 'CompletedTime' \wedge a_s.e = 'CompletedTime' \wedge a_a.t \leq a_s.t \wedge a < s \wedge 1 \leq a, s \leq n\}$

As you can read, **Algorithm 1** needs a dataset of the process enactment event logs as inputs. Assume that the dataset is formatted in the IEEE XES (extensible event stream) format [17], the schema structure of which has a hierarchical inclusion relationship among Log, Trace and Event classes. Accordingly, the Log class is containing the Trace class of the process instance event traces in an XML tag form of ( $\langle trace \rangle \dots \langle /trace \rangle$ ), and the Trace class contains the Event class of the process enactment event logs in an XML tag form of ( $\langle event \rangle \dots \langle /event \rangle$ ). The STEP-1 algorithm is to conceptually extract a process instance event log ( $PET_1$ ) from the XES-formatted dataset, and to conceptually fragmentize a temporal work-case model into a fragment-group of temporally ordered adjacent-activity pairs. At the same time, the implemented STEP-1 algorithm can extract practically an XES-formatted process instance event trace, and fragmentize the tagged trace into a fragment-group of event-level adjacent-activity pairs. As you can see in Figure 8, the conceptual procedure illustrates the detailed internal transformations of the **Algorithm 1**, which is a conceptual mining procedure eventually being connected to the STEP-2 algorithm.

### b: WEIGHTED ADJACENT-ACTIVITY SET AND WEIGHTED PROCESS PATTERN GRAPH

The STEP-2 of the  $\rho$ -Algorithm is to build all the groups of temporally ordered adjacent-activity pairs, each of which corresponds to a process instance event trace. The eventual output of this algorithm is a weighted adjacent-activity set named as *adjacencyList* ( $\beta$ ). This set is built from all the groups of temporally ordered adjacent-activity pairs through an internal transformation procedure. Especially, the concept of weights on the edge and its activities of each pair implies the number of occurrences of the corresponding pair and



**FIGURE 8.** The conceptual procedure of the STEP-1 of the  $\rho$ -algorithm: from dataset to process enactment event traces with adjacent-activity pair groups.

its activities. Also, the set is used to produce a weighted process pattern graph with all the edges weights and their activities weights through an internal transformation procedure. The middle part of Figure 7 depicts the conceptual procedure of the STEP-2 algorithm. Finally, **Algorithm 2** algorithmically describes the pseudo-codes of the STEP-2 of the  $\rho$ -Algorithm.

#### c: DISCOVERING STRUCTURED INFORMATION CONTROL NET MODEL

The final step (STEP-3) of the  $\rho$ -Algorithm is to build a structured information control net model from the weighted process pattern graph mined from all the groups of temporally ordered adjacent-activity pairs. The eventual goals of the  $\rho$ -Algorithm are to discover a structured information control net model as the process discovery aspect and to discover proportions of the process patterns as the process knowledge discovery aspect. Note that the structured information control net model must be satisfied with the proper nesting as well as the matched pairing properties in forming gateway activities in each process pattern. The right-most part of Figure 7 depicts the conceptual procedure of the STEP-3 algorithm that is able to completely transform from a weighted process pattern graph as the outcome of the STEP-2 algorithm to a structured information control net model on every branch of the associated gateway activities. Finally, **Algorithm 3** algorithmically describes the pseudo-codes of the STEP-3 of the  $\rho$ -Algorithm.

**Algorithm 3** performs two essential functions. One (STEP-3.1) is a graphical visualization function [27], [37], [38] to visualize the weighted process pattern graph in a form of graphical viewer using the Graph Stream Library. The other (STEP-3.2) is a visual transformation function to transform the weighted process pattern graph into a graphical viewer of the structured information control net model formed by a certain combination of the four types of primitive process patterns shown in Figure 5. As described in the previous subsection, the weighted process pattern graph is built by performing the internal transformations from *adjacencyList* ( $\beta$ ). In terms of visualizing the weighted process patterns and

its information control nets, the Graph Stream Library is used. The core part of the  $\rho$ -Algorithm is the STEP-3.2 function that decides open-gateways and close-gateways of each process patterns in the graph by using the concept of rho ( $\rho$ ). That is, the rho ( $\rho$ ) function gives the number of occurrences of its associated activity, and the principles of decision-makings are followings:

- The Linear (Sequential) Process Pattern
  - $\rho\text{ONLYONE\_ParentNode} == \rho\text{ONLYONE\_ChildNode}$
- The Disjunctive (Exclusive-OR) Process Pattern
  - Open-gateway:  $\rho\text{ParentNode} > \rho\text{ALL\_ChildNodes}$
  - Close-gateway:  $\rho\text{ALL\_ParentNodes} < \rho\text{ChildNode}$
- The Conjunctive Process (Parallel-AND) Pattern
  - Open-gateway:  $\rho\text{ParentNode} == \rho\text{ALL\_ChildNodes}$
  - Close-gateway:  $\rho\text{ALL\_ParentNodes} == \rho\text{ChildNode}$
- The Repetitive (Iterative-LOOP: DO-WHILE) Process Pattern
  - Open-gateway:  $\rho\text{ParentNode} < \rho\text{ONE\_ChildNode}$
  - Close-gateway:  $\rho\text{ParentNode} > \rho\text{ONE\_ChildNode}$

#### C. SUMMARY ANALYSIS AND COMPLEXITY OF THE $\rho$ -ALGORITHM

The eventual goal of the algorithmic process mining framework proposed in this paper is to discover a structural information control net model being composed of a certain combination of all the process patterns from a dataset of process enactment event logs through deploying the  $\rho$ -Algorithm with a series of concepts and their related algorithms like STEP-1 (**Algorithm 1**), STEP-2 (**Algorithm 2**) and STEP-3 (**Algorithm 3**) algorithms. Until now, the paper has formally described those conceptual and procedural formal definitions, such as the group of temporally ordered adjacent-activity pairs, the weighted adjacent-activity set, the weighted process pattern graph and the structured information control net model. Therefore, it is necessary to summararily show that the major goals of the paper have been achieved by not only discovering a structural information control net model as one goal of the process mining aspect but also discovering its occurrences on each of the process patterns as the

**Algorithm 1** STEP-1 of the  $\rho$ -Algorithm**Require:** A Dataset of Process Enactment Event Logs in XES-Format,  $a.xes$ **Ensure:** A List of Tagged Traces of XES Event Logs,  $ListXES$ 

```

1: procedure STEP-1 of  $\rho$ -Main( $a.xes$ )                                ▷ Forming a list of process instance event traces
2:   openfile  $a.xes$ ;
3:   while (!EOF in  $a.xes$ ) do
4:     Line  $\leftarrow$  currentline;
5:     if (currentline == "< trace >") then                            ▷ Checking out the beginning trace-tag of a single process instance
6:       startTrace  $\leftarrow$  true;
7:     else
8:       if (currentline == "< event >") then                            ▷ Checking out the beginning event-tag of a single work-item
9:         startEvent  $\leftarrow$  true;
10:        endEvent  $\leftarrow$  false;
11:      else
12:        if (currentline == "< /event >") then                            ▷ Checking out the ending event-tag of the work-item
13:          startEvent  $\leftarrow$  false;
14:          endEvent  $\leftarrow$  true;
15:        else
16:          if (currentline == "< /trace >") then                            ▷ Checking out the ending trace-tag of the process instance
17:            startTrace  $\leftarrow$  false;
18:            ListXES.add(currentTrace);                                ▷ Forming a single process instance event trace (PET)
19:            currentTrace  $\leftarrow$  "";
20:          end if
21:        end if
22:      end if
23:    end if
24:    if (startTrace && startEvent) then
25:      store all contents of the event to currentTrace;                ▷ Store all the information of activities, performers and
timestamp inside each event to currentTrace
26:    end if
27:    seek to the next line;
28:  end while
29:  closefile  $a.xes$ ;
30:  return ListXES;                                                  ▷ Returning the formed list of all the process instance event traces
31: end procedure

```

other goal of the process-aware knowledge discovery aspect. An operational example, the  $\rho$ -Algorithm is deployed on the same exemplary dataset in Figure 2 and Figure 3 that were used for proving the  $\alpha$ -Algorithm and  $\sigma$ -Algorithm, respectively. Figure 9 illustrates all the intermediary results produced from the dataset (L) by applying all the subroutines of the  $\rho$ -Algorithm, like the STEP-1, STEP-2 and the STEP-3 algorithms. The **Algorithm 1** extracts the 5 temporal work-cases, each of which is corresponding to each of the process instances, from the process enactment event logs (L); the **Algorithm 2** fragmentizes each temporal work-case into a group of temporally ordered adjacent-activity pairs and internally transforms all the groups into a weighted adjacent-activity set; the **Algorithm 3** algorithm finally discovers a structured information control net model,  $\rho(L)$ , based upon the principles of decision-making type of the process patterns and their open-gateways and close-gateways. Through this operational example on the exemplary dataset,

we were able to manually verify that the  $\rho$ -Algorithm ought to be theoretically correct and functionally reasonable.

Finally, the time complexity of the **Algorithm 1** and the **Algorithm 2** in the  $\rho$ -Algorithm is  $O(N \times M)$ , respectively, where  $N$  is the number of temporal work-cases, which implies the number of process instance event traces in a dataset of process enactment event logs, and  $M$  is the number of activities associated with the corresponding process model. However, the time complexity of the **Algorithm 3** in the  $\rho$ -Algorithm is  $O(M)$ , where  $M$  is the number of activities associated with the weighted process pattern graph discovered from the underlying dataset. Consequently, in comparing with these conventional process mining algorithms, the functional merits and restrictions of the  $\rho$ -Algorithm proposed in the paper, which also imply the functional goals of the  $\rho$ -Algorithm, can be summarized as follows:

- The  $\rho$ -Algorithm is able to explicitly deal with discovering not only the three types of the primitive process



**Algorithm 2** STEP-2 of the  $\rho$ -Algorithm**Require:** A List of Tagged Traces of XES Event Logs, *ListXES***Ensure:** A Weighted Adjacent-Activity Set, *adjacencyList* ( $\beta$ )**Ensure:** A Weighted Process Pattern Graph, *G*

```

1: procedure STEP-2 of  $\rho$ -Main(ListXES) ▷ Fragmentizing a process enactment event trace into a set of temporally ordered
   adjacent-activity pairs and building a weighted adjacent-activity set as well as its process pattern graph
2:   initialize  $\beta$  and G to Null-value;
3:   openfile ListXES;
4:   for ( $\forall$  tracei  $\in$  ListXES) do                                     ▷ For each process enactment event trace
5:     for ( $\forall$  eventj  $\in$  tracei) do                                     ▷ For each process enactment event in the trace
6:       currentActivity = eventj.GetActivityName();
7:       if (currentActivity  $\notin$   $\beta$ ) then                                     ▷ Checking out the existence of the current activity in the weighted
   adjacent-activity set
8:          $\beta$ .add(currentActivity);                                       ▷ Building the weighted adjacent-activity set
9:       else
10:        listOfVertex  $\leftarrow$   $\beta$ .add(currentActivity);
11:        nextActivity  $\leftarrow$  eventj+1.GetActivityName();
12:       end if
13:       if (nextActivity  $\notin$  listOfVertex) then                               ▷ Checking out the existence of the next activity as a node in the
   weighted process pattern graph
14:         listOfVertex.add(nextActivity);                                   ▷ Building the weighted process pattern graph
15:       else
16:        listOfVertex.update(nextActivity, +1);
17:       end if
18:     end for
19:   end for
20:   G = CreateGraph(listOfVertex,  $\beta$ );
21:   closefile ListXES;
22:   return  $\beta$  and G;                                     ▷ Returning the weighted adjacent-activity set and its process pattern graph
23: end procedure

```

patterns supported by the conventional algorithms but also the most challenging type of the repetitive-LOOP process pattern. In principle, its discovered model is in a graphical form of structured information control net that is satisfied with keeping the matched-pairing as well as the proper-nesting properties in forming all the primitive process patterns made up of the corresponding structured information control net process model.

- The  $\rho$ -Algorithm is theoretically supported by the information control net modeling methodology; its final output model discovered is in a mathematical graph of the information control net process model, even though its input dataset is not originated from the information control net process model.
- The  $\rho$ -Algorithm deals with discovering only the DO-WHILE type of the repetitive-LOOP process patterns, in principle.
- The next node(s) selection rule of the iterative-LOOP CLOSE-gateway node in the discovered process model is the same as the inclusive-OR OPEN-gateway node's selection rule.
- Finally, the discovered information control net process model by the  $\rho$ -Algorithm may be unstructured only if the original process model of the input dataset

is unstructured, and it may also be unreasonably and improperly constructed only if the input dataset is not a non-noise dataset without any types of control-flow-related and timestamp-related noises.

#### IV. VALIDATING THE $\rho$ -ALGORITHM

As the final stage of designing the algorithmic process mining framework, it is necessary to validate the functional correctness and feasibility of the  $\rho$ -Algorithm by implementing the  $\rho$ -Algorithm and deploying onto a process enactment event log dataset especially formatted in a form of the IEEE-XES<sup>12</sup> standardized log format [17]. The authors' group has successfully implemented a process mining system<sup>13</sup> theoretically supported by the  $\rho$ -Algorithm. By using the implemented system, we carry out an experimental analysis on the synthetic dataset of process enactment event logs, which was released to the public by the 4TU.Centre for Research Data [16] and named as the Large Bank Transaction Process

<sup>12</sup>In recent, IEEE has released a standard tag-based language, XES (XML Event Stream), whose aim is to provide designers of information systems with a unified and extensible methodology for capturing systems' behaviors by means of event logs and event streams.

<sup>13</sup>In this paper, we won't describe the detailed implementation of the SICN-oriented process mining system.

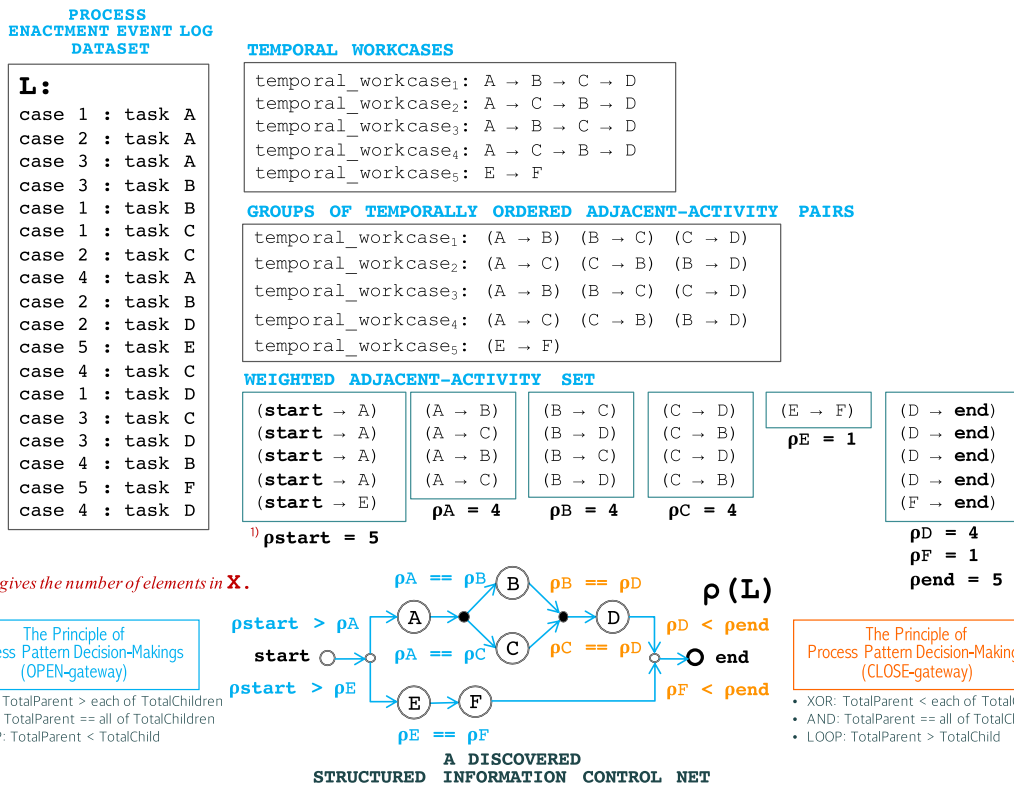
**Algorithm 3** STEP-3 of the  $\rho$ -Algorithm

**Require:** A Weighted Process Pattern Graph,  $G$

**Ensure:** A Structured Information Control Net Graph,  $G^{SICN}$

```

1: procedure STEP-3 of  $\rho$ -Main(  $G$  )                                ▷ Mining the structured information control net process model
2:   openfile  $G$ ;
3:    $G.removePhantomEdge()$ ;                                       ▷ Removing the phantom edges on  $G$ 
4:   for  $\forall \alpha \in G$  do                                           ▷  $\alpha$  is a member node of the node-set in  $G$ .
5:      $\alpha.listEdgeOutGoing \leftarrow G.getEdgeOutGoing(\alpha)$ ;
6:     if  $\alpha.listEdgeOutGoing.size() > 1$  then
7:        $G^{open} \leftarrow ProcessForTheOpenGate(G, \alpha, \alpha.listEdgeOutGoing)$ ;   ▷ Discovering the open-gateways in  $G$ 
8:     end if
9:      $\alpha.listEdgeInComing \leftarrow G.getEdgeInComing(\alpha)$ ;
10:    if  $\alpha.listEdgeInComing.size() > 1$  then
11:       $G^{close} \leftarrow ProcessForTheCloseGate(G, \alpha, \alpha.listEdgeInComing)$ ;   ▷ Discovering the close-gateways in  $G$ 
12:    end if
13:  end for
14:   $G^{SICN} \leftarrow ProcessForTheLoopGate(G^{open}, G^{close})$ ;       ▷ Discovering the loop-gateways from  $G^{open}$  and  $G^{close}$ 
15:  closefile  $G$ ;
16:  return  $G^{SICN}$                                                ▷ Discovered a structured information control net process model,  $G^{SICN}$ 
17: end procedure
  
```



**FIGURE 9.** An operational verification on the exemplary dataset for discovering structured information control net process model by the  $\rho$ -algorithm.

Dataset. This validation with an experimental analysis aims at proving that the  $\rho$ -Algorithm is able to discover all the four types of primitive process patterns [39] such as linear, disjunctive, conjunctive and repetitive process patterns and their occurrences.

**A. IMPLEMENTATION OF THE  $\rho$ -ALGORITHM**

For the sake of confirming the feasibility and the applicability of the devised  $\rho$ -Algorithm, it ought to be implemented as a specific and tangible process mining system supporting the structured information control net modeling methodology.

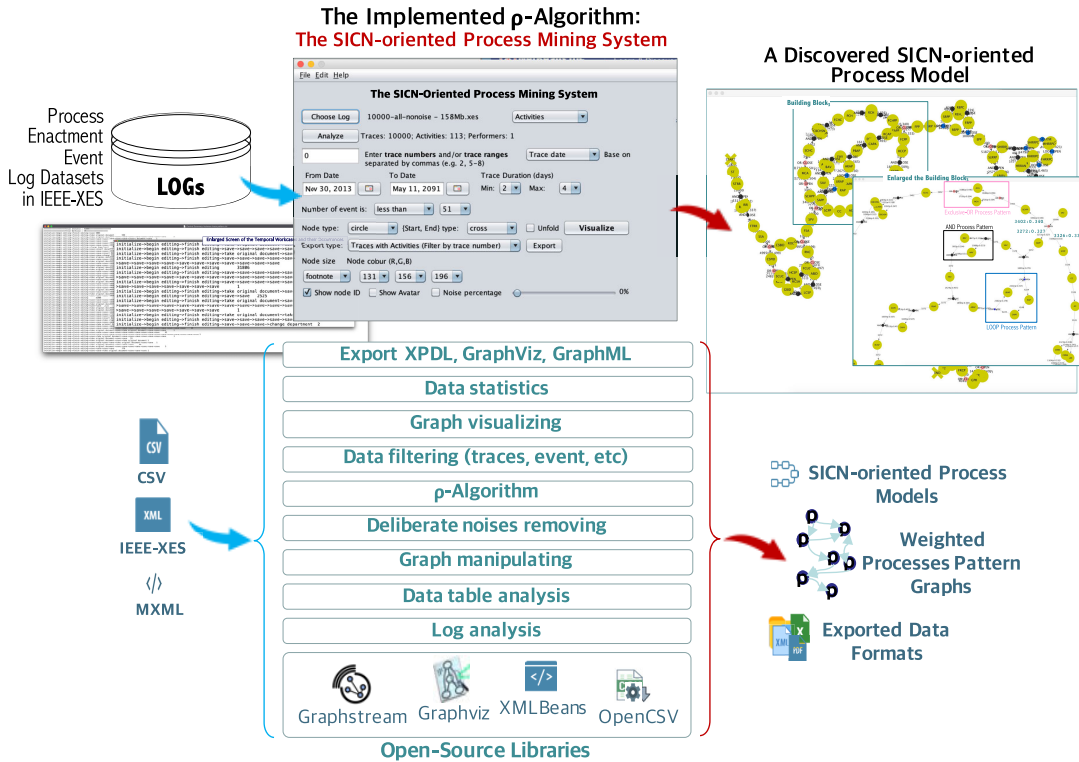


FIGURE 10. Functional components of the SICN-oriented process mining system implementing the  $\rho$ -algorithm.

Consequently, we have successfully implemented the  $\rho$ -Algorithm, which is named as the SICN-oriented process mining system and extended from the process mining system [40] previously developed by the authors' research group, and the system's dashboard with a list of core-functional components are illustrated in Figure 10, on which the inline and dashboard prompts are displaying the experimental processing status like choosing the dataset file-name of the Large Bank Transaction process dataset, inputting the event-trace identifiers, clicking the functional operations to get the eventual discovery result of the SICN-oriented Large Bank Transaction process model, and so on. In this subsection, we briefly introduce the functional descriptions of all the essential components of the implemented  $\rho$ -Algorithm, as followings:

- **Log analysis:** This function is operated as the entrance gate of the system. The raw input dataset, formed in the different formats (e.g., XES, MXML, etc.), will be read using this function. Depending on the different input data types, the log analysis function analyzes their structures and entities. It then stores the analyzed information to the system's memory and represents in two different types of data formation: *Table* and *Graph*. The key objective is to transform the raw event log data into the efficient managing form of *Table* with traces, events, and event's properties. In the *Table* structure, each row is assigned for an event's properties in each event-trace through the Trace ID and Event ID. Therefore, a complete event-trace holds a temporal sequence

of associated events as each row of the Table through the same Trace ID. In the *Graph* structure, each vertex represents an event holding information with activity, performer, timestamp, and other additional information. An edge connecting two vertices represents the directed transition between the associated two events. The weights of vertex and edge represent the numbers of occurrences that the events and their transitions were operated, respectively.

- **Data table analysis:** This function analyzes data about traces and events stored in the table after being imported from the raw data. Each row in the table contains information about event-operated time, information about the activity, performer, and other related information. Different rows are differentiated by Trace ID and Event ID, and hence we can perform different aspects of the data aggregation functions. For example, we can summarize how many times a specific activity occurred in a trace, and how many times it was operated by filtering the table with the activity among different Trace IDs. Similarly, we may produce some statistical information about a specific performer from the input dataset, like how many times the performer was participated in the instances' operations.
- **Graph manipulating:** This function is to manipulate the elements on a graph. That is, the graph manipulation operations are provided not only for adding, editing, and deleting vertices and edges, but also for updating

the weights of vertices and edges on a graph. Additionally, this function also includes a series of operations like searching and traversing graphs with breadth-first search algorithm, spanning tree algorithm, binary search tree algorithms, and so on. Using the graph manipulating function, moreover, we are able to visualize whatever actually happened during the discovery algorithm's operations and whichever different control-flow paths identified from all the traces in the event log dataset. This function's main task is to support visualizing all the graph-formed outcomes, such as the adjacent-activity pair sets, the weighted process pattern graph, and the information control net graph, to be produced through the stepwise discovery operations being conducted with the implemented system.

- *Deliberate noises removing*: In forming the weighted process pattern graph as one of the implemented system's outcomes, we realize that there exist some phantom edges among the activities involved in the conjunctive transitions, which is named as deliberate noises. This function is so for removing all the phantom edges existing among the activities involved in a single conjunctive (parallel-AND) transition group. The implemented system executes this function on every group of the conjunctive transitions and produces a weight process pattern graph satisfying the completeness and safeness requirements.
- $\rho$ -Algorithm: This function is to implement the  $\rho$ -Algorithm devised in the previous section and also extended from the proportional process mining algorithm proposed in [25]. The algorithm's idea is to discover all the disjunctive (exclusive-OR), conjunctive (parallel-AND), and repetitive (iterative-LOOP) process patterns from a dataset of process enactment event logs by considering the occurrences of activities and their control-transition relationships. This function is completed with the supports of the three steps of the algorithms (STEP-1, STEP-2, and STEP-3 Algorithms) and the graph manipulating and deliberate noises removing functions as described above.
- *Data filtering*: In the historical event logs of the business process management system, the implemented  $\rho$ -Algorithm needs to analyze thousands of traces enacted, hundreds of thousands of events operated, and even a much larger number of the event properties' values. Therefore, the data filtering function ought to be a vital component in managing and manipulating the huge-sized dataset in the most detailed and effective way. This function allows the users to filter the input dataset out onto tables or graphs according to the intentioned criteria made by using such events, activities, performers, and/or relevant information like timestamps.
- *Graph visualizing*: Graph visualization brings knowledge of what occurred inside the log in an intuitive manner. This function uses the Graphstream library [41] for visualizing and analyzing the graph. By using

this library, not only we can generate, measure, visualize all the graph-formed outcomes of the implemented  $\rho$ -Algorithm, but also we can provide the users very useful and easy ways of interactive user-interfaces.

- *Data statistics*: This function performs and calculates data statistics based on the information managed on the implemented  $\rho$ -Algorithm. By combining the Data table analysis and Data filtering functions, we can perform statistical measurements and generate reports related with traces, events, activities, and the others inside the historical event log dataset.
- *Export XPDL, GraphViz, GraphML*: The SICN-oriented process mining system need to provide the exporting function that transforms the discovered processes and process models into any other different data formats to be used in different scenarios. Currently, the users can simply export the discovered SICN-oriented process model to any one of the XPDL, GraphViz, and GraphML formats.

## B. DATASET OF THE PROCESS ENACTMENT EVENT LOGS

For carrying out the validation analysis of the  $\rho$ -Algorithm, we chose a synthetic dataset from the 4TU Centre's archive, the name of which is 10000 – all – nonoise – 150MB.xes [16], [39] containing 10,000 process instance event traces and all the 113 activities are involved in the enactment of the corresponding process model, which is named as Large Bank Transaction Process Model and released in a graphical form of the Petri-net-oriented process model. The following are the description of the dataset posted in the 4TU Centre's website:

- title: Large Bank Transaction Process
- creator: Munoz-Gama.J (Jorge)
- data accepted: 2014-08-21
- data published: 2014
- description: "Synthetic Bank Transaction Process Models: Petri net, Large, Stand-alone and SESE-aided Decomposed Logs: Large, with and without noise, two particular scenarios. Additional: Model diagram, decomposition diagram, activity re-naming."
- keyword: Bank Transaction
- language: en
- publisher: Universitat Politecnica de Catalunya (Barcelonatech)
- subject: 1503 - Business and Management
- in collection: Synthetic Event Logs

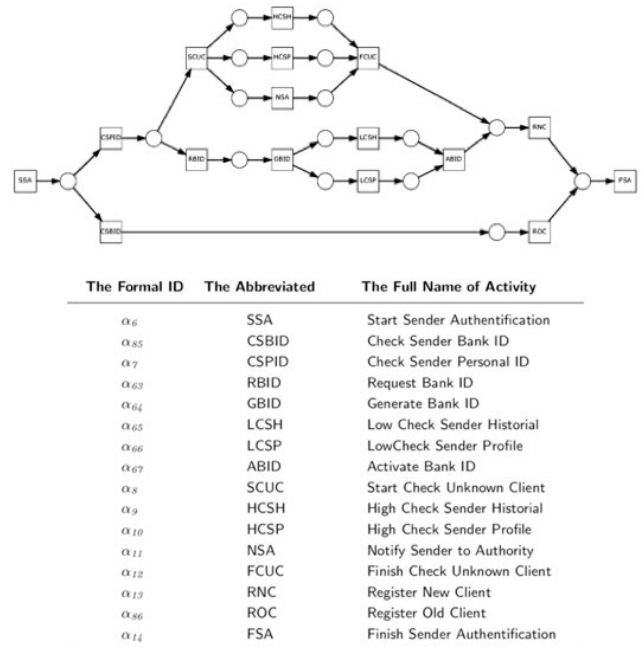
This dataset ought to be a typical dataset to usefully validate the functional correctness of such a process mining algorithm, because it is a synthetic as well as non-noise dataset, and it is also built from enacting a large-enough number of activities with 113 different activities, and their enactment event histories embedding all the possible process enactment sequences. Finally, the ultimate advantage of the dataset is based upon the various and different control-flow patterns with combining all the four types of primitive process patterns. Therefore, by using this dataset we can effectively



and efficiently verify and validate the functional correctness and the operational performance of the  $\rho$ -Algorithm. Fortunately for the experimental validation, the original process model of the dataset is released in a graphical form of the Petrinet-oriented process model with those names of the involved activities. Figure 11 is illustrated with a Petrinet-oriented graphical form of a process building block and a table form listing the abbreviated and full names of those 16 activities involved. Additionally, the formal ID of each activity is listed in the table and these are arranged for depicting the formal graph models to be produced by the  $\rho$ -Algorithm. As you see, the process building block, which is chosen for validating the functional correctness of the disjunctive and conjunctive process pattern discovery, is made up of two pairs of Exclusive-OR gateway nodes and two pairs of Parallel-AND gateway nodes and these gateway nodes are properly nested. And moreover, Figure 14 shows a Petrinet-oriented graphical form of another process building block that is projected from the Large Bank Transaction process model and chosen for validating the repetitive process pattern discovery functionality of the  $\rho$ -Algorithm. Note that a Petrinet graph model is built from a group of rectangular transitions and a group of circled places and the directed edges connecting transitions and places; in a Petrinet-oriented process model, transitions and places imply activities and pre-/post-conditions (before/after enacting the activities), respectively.

### C. DISCOVERING THE DISJUNCTIVE (EXCLUSIVE-OR), CONJUNCTIVE (PARALLEL-AND) AND REPETITIVE (ITERATIVE-LOOP) PROCESS PATTERNS

This section deploys the implemented process mining system, which is based upon all the algorithmic functions (STEP-1, STEP-2 and STEP-3) of the  $\rho$ -Algorithm, onto the typical dataset that is prepared in the previous section, and describes its outcomes and artifacts that are produced from this validation and feasibility analysis. Fortunately, the provider of the dataset released a supplementary file containing the graphical representation of the Petrinet-oriented process model as the original process model of the Large Bank Transaction process dataset, which ought to be very useful to check up on verifying and validating the functional correctness of the process mining algorithm and system proposed in this paper. Accordingly, in this section we carry out two cases of experimental analyses applying each step of the algorithmic process mining framework and validating their experimental outcomes by comparing this supplementary model with the discovered SICN-oriented process model by the implemented  $\rho$ -Algorithm. In other words, the first experimental analysis is focusing on the discovery experiment of disjunctive and conjunctive process patterns with a building block of the Sender Authentication Subprocess Model introduced in Figure 11, and the second experimental analysis is concentrating on the discovery experiment of repetitive process pattern with another building block of the Account Payment Processing Subprocess Model shown in Figure 14.



**FIGURE 11.** The disjunctive and conjunctive process patterns discovery target of the  $\rho$ -algorithm: a petrinet-oriented process model of the sender authentication subprocess model.

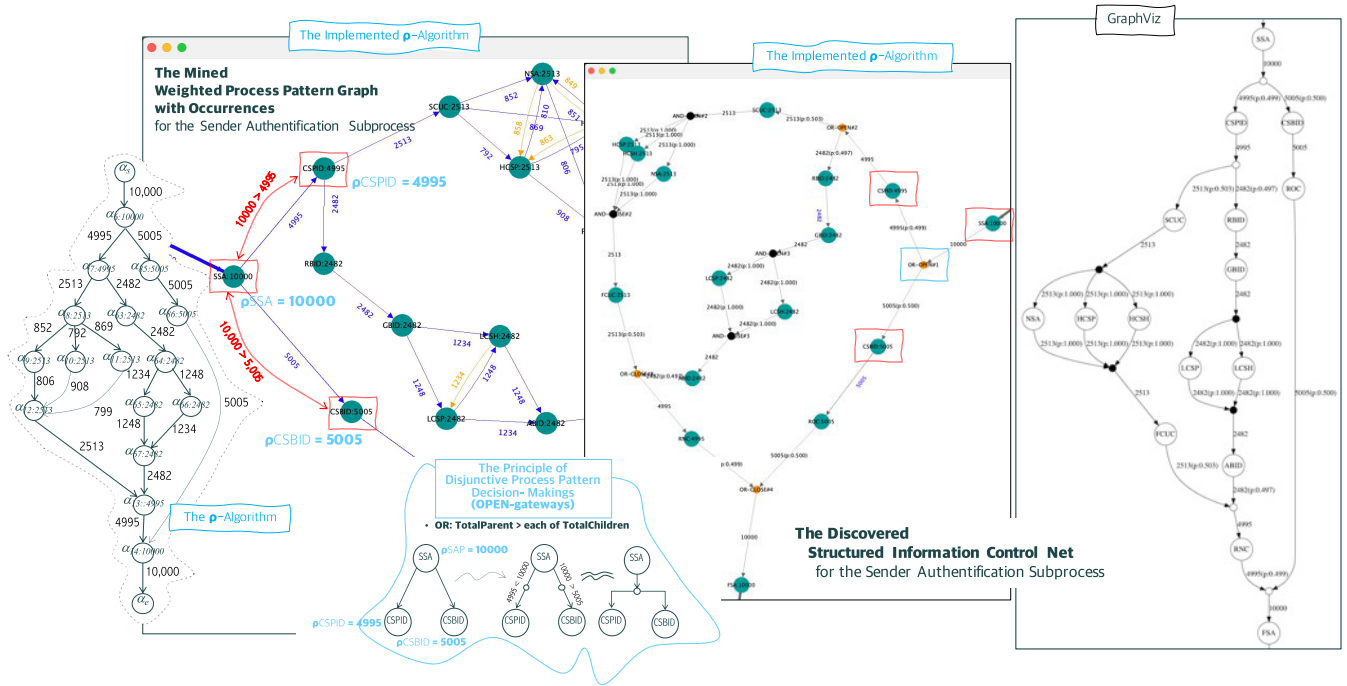
#### 1) MINING GROUPS OF ADJACENT-ACTIVITY PAIRS BY THE STEP-1 ALGORITHM

From now on, the STEP-1 algorithm launches to mine all the groups of temporally ordered adjacent-activity pairs from all the event traces on the Large Bank Transaction process dataset. We probably suppose that the dataset is recorded by enacting the 10,000 process instances of the Large Bank Transaction process model. In the previous section, we have already defined a formal concept of the temporal work-case model that is the formal representation of an event trace of a process instance. From all the temporal work-case models, the algorithm mined a large number of groups (10,000 AAGs) of temporally ordered adjacent-activity pairs through a series of the internal transformations of the STEP-1 algorithm.

#### 2) MINING A WEIGHTED ADJACENT-ACTIVITY SET AND ITS WEIGHTED PROCESS PATTERN GRAPH BY THE STEP-2 ALGORITHM

The first internal transformation of the STEP-2 algorithm is to build a weighted adjacent-activity set from all the groups (10,000 AAGs) of temporally ordered adjacent-activity pairs. This internal transformation algorithm fulfills integrating all of the groups and regrouping same adjacent-activity pairs with calculating their occurrences. To clearly validate this algorithm, we also scrutinize every single-case in the output and confirmed that the implemented  $\rho$ -Algorithm works correctly. The second internal transformation of the STEP-2 algorithm is to build a weighted process pattern graph by combining all the elements in the weighted adjacent-activity set. The devised and implemented functions of this combining step are properly connecting the source-nodes and the





**FIGURE 12.** The SICN-oriented sender authentication subprocess model successfully discovered from the dataset by the implemented  $\rho$ -algorithm.

destination-nodes in the weighted adjacent-activity set along with the matched activity ID. Right after building the weighted process pattern graph, it is necessary to eliminate all the deliberate noises in the graph, which are naturally created in forming the temporal work-cases and their models from all the event traces.

Figure 12 and Figure 13 depict the situational experiments with the discovered outcomes of the STEP-2 and the STEP-3 algorithms in the  $\rho$ -Algorithm and the implemented  $\rho$ -Algorithm, as well. The captured screens in the lefthand side of Figure 12 and Figure 13 are the projected outcomes of the STEP-2 algorithm of the implemented system, which are the weighted process pattern graphs corresponding to the building block of Figure 11 and the building block of Figure 14, respectively. Moreover, the hand-made graph in the leftmost of Figure 12 shows the weighted process pattern graph and the occurrence numbers of activity nodes and edges after removing the deliberate noises, which is the eventual weighted process pattern graph generated by the STEP-2 algorithm, and the hand-made graph-fragments and activities' total occurrences in the leftmost of Figure 13 represent the weighted adjacent-activity set with occurrences generated by the STEP-2 algorithm and corresponding to the subprocess model of Figure 14.

3) DISCOVERING A STRUCTURED INFORMATION CONTROL NET BY THE STEP-3 ALGORITHM

The last step of the validation is to discover a structured information control net from the weighted process pattern graph with the STEP-3 algorithm. The captured screens in

the right-hand side of Figure 12 and Figure 13 display the discovered structured information control nets generated by the STEP-3 algorithm of the implemented system, which are transformed from the weighted process pattern graphs corresponding to Figure 11 and Figure 14, respectively. Finally, both of the structured information control nets discovered by the implemented system are neatly visualized by the graph visualization software, GraphViz, as attached on the rightmost part of Figure 12 and Figure 13, respectively. The detailed decision-making procedure of the STEP-3 algorithm is applied to the weighted process pattern graph with activities' enactment occurrences to finally discover all the primitive process patterns and form a structured information control net, as follows:

- *Discovering the Disjunctive (Exclusive-OR) and Conjunctive (Parallel-AND) Process Patterns:* First of all, on Figure 12 we elucidate all the stepwise outcomes on discovering the disjunctive and the conjunctive process patterns in the Sender Authentication Subprocess Model that is graphically represented in a form of the Petri-net-oriented process model as shown in Figure 11. Additionally, Figure 12 illustrates the detailed outcomes of the decision-making principle for disjunctive and conjunctive process patterns and displays the final outcome of the SICN-oriented process model and its GraphViz visualization. As you can see, the implemented  $\rho$ -Algorithm operates perfectly the decision-making principles of the open-gateways and the close-gateways with the disjunctive and the conjunctive process patterns. In the figure, the final outcome of

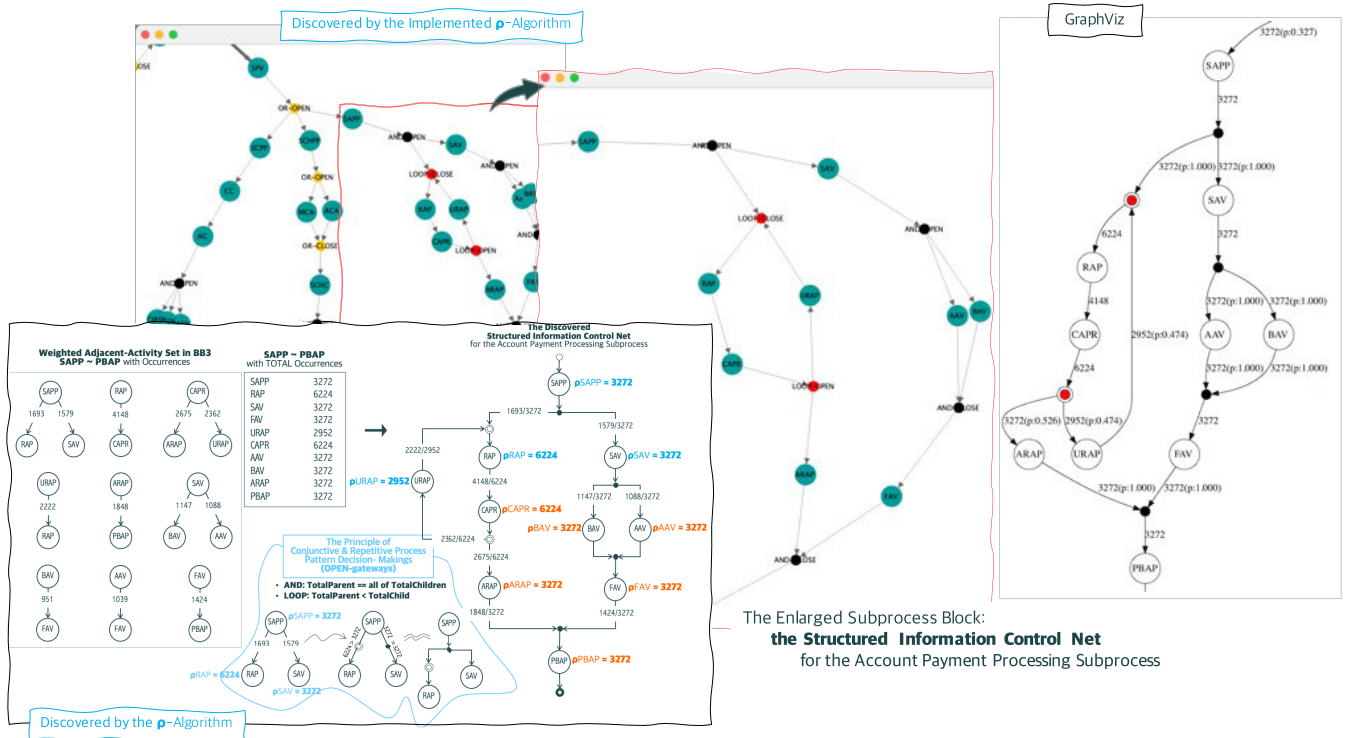


FIGURE 13. The SICN-oriented account payment processing subprocess model successfully discovered from the dataset by the implemented  $\rho$ -algorithm.

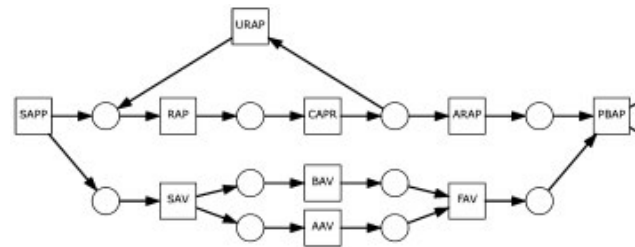


FIGURE 14. The repetitive process pattern discovery target of the  $\rho$ -algorithm: a Petri net-oriented process model of the account payment processing subprocess model.

the SICN-oriented process model holds two disjunctive pairs of open-gateway nodes and close-gateway nodes and two conjunctive pairs of open-gateway nodes and close-gateway nodes, and all of these disjunctive and conjunctive pairs are in keeping the matched pairing and proper nesting properties. Consequently, we can clearly declare that the functional correctness of the  $\rho$ -Algorithm and its implemented system in discovering disjunctive and conjunctive process patterns is successfully verified as well as validated.

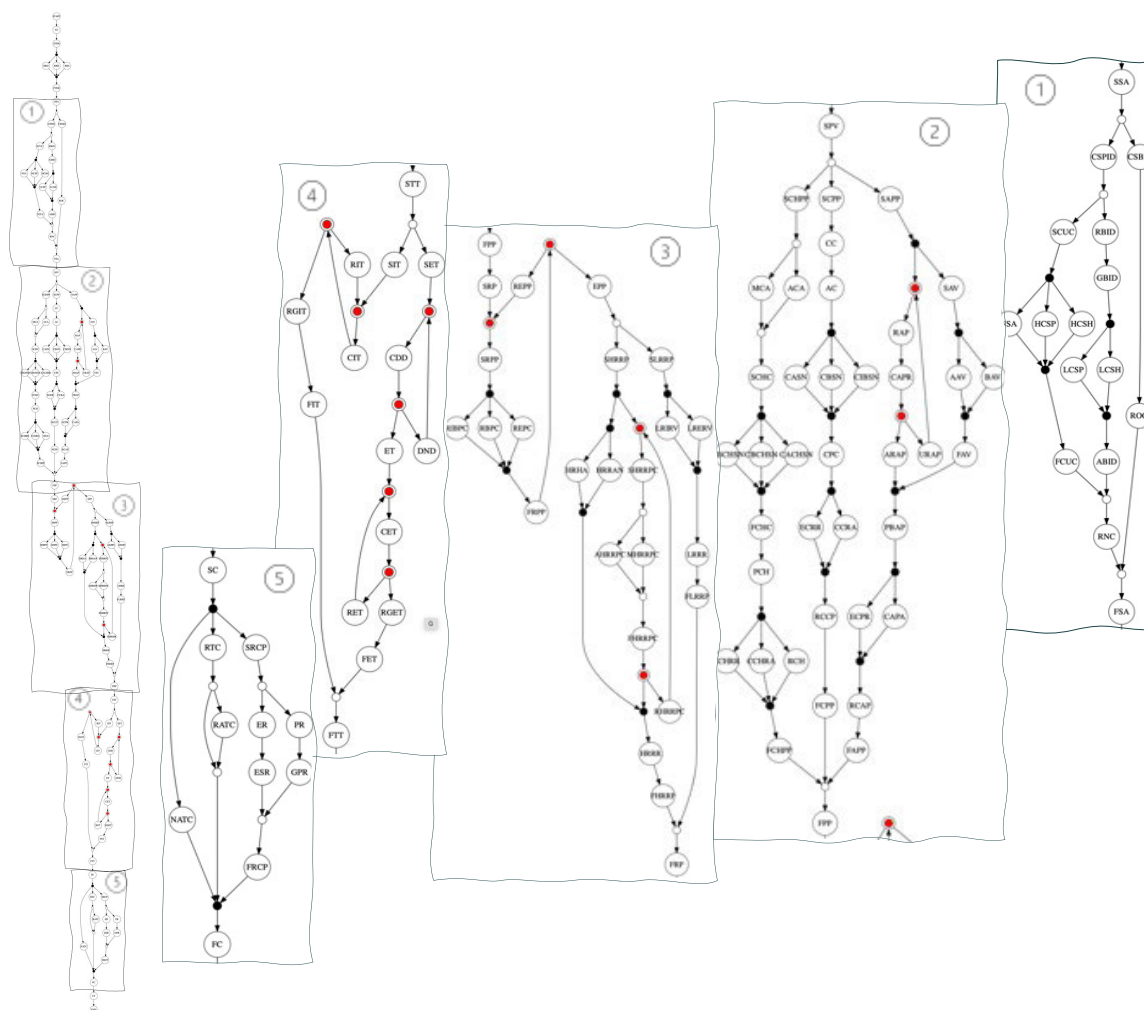
- *Discovering the Repetitive (Iterative-LOOP) Process Patterns:* The second experimental analysis is done for validating the functional correctness of the  $\rho$ -Algorithm and its implemented system in terms of the discovering perfectness of repetitive process patterns with keeping the matched pairing and proper nesting properties. In this second experimental analysis, we use a specific Petri net-oriented process model

of the Account Payment Processing Subprocess containing a single iterative-LOOP construct and two parallel-AND constructs, as shown in Figure 14. On Figure 13, we illustrate all the outcomes of the detailed decision-making procedure of the STEP-3 algorithm and its eventual formation of the discovered structured information control net containing a repetitive pair of open-LOOP gateway and close-loop gateway nodes and two conjunctive pairs of open-AND gateway and close-AND gateway nodes with keeping the matched pairing and proper nesting properties. Additionally, we attach the GraphViz visualization of the discovered one to the rightmost part of Figure 13. As you can see, the discovered SICN-oriented process model looks exactly like the structural formation of the Petri net-oriented process model of the Account Payment Processing Subprocess, and it is also satisfied with the structural requirement of the match pairing as well as proper nesting properties. Through this experimental analysis, we can conclude that the STEP-3 algorithm and its implemented system work perfectly in terms of discovering the repetitive (iterative-LOOP) process patterns with keeping the structural properties.

#### D. SUMMARY ANALYSIS OF THE $\rho$ -ALGORITHM VALIDATION

These two cases of experimental analysis for validating the  $\rho$ -Algorithm and its implemented system have been successfully done as presented in the previous section.





**FIGURE 17.** The GraphViz visualization form of the SICN-oriented process model discovered from the large bank transaction process dataset.

In the first experimental analysis, we have proved that the  $\rho$ -Algorithm is able to discover the disjunctive process pattern type and the conjunctive process pattern type; in the second experimental analysis, we have also confirmed that the  $\rho$ -Algorithm can discover the repetitive process pattern type, too; eventually, we have so corroborated that the  $\rho$ -Algorithm works the discovery perfectness on all the primitive process patterns with keeping the structural requirement of matched pairing and proper nesting properties. Actually, we carried out a single experiment on the synthetic dataset for validating the STEP-1, STEP-2 and STEP-3 algorithms of the  $\rho$ -Algorithm. Through this experiment, we found out, as shown in Figure 15, that the dataset contains all the enactment event histories recorded from enacting 10,000 process instances of the Large Bank Transaction Process Model with the total number of activities involved in the enactments is 113 different activities, and that the largest number of activity enactment occurrences is 18,458 times. Note that the titles of the activities having the largest occurrences are RBPC (Receiver Bank Profiling Checking Activity), REPC (Receiver External

Profiling Checking Activity), SRPP (Start Receiver Pre Profiling Activity), RIBPC (Receiver Inter-Bank Profiling Checking Activity) and FRPP (Finish Receiver Pre Profiling Activity), and all of which are associated with the Receiver Processing Subprocess Model of the Original Model of the Large Bank Transaction Process Model.

Figure 16 displays two of the captured-screens from the experimental results of the implemented  $\rho$ -Algorithm: one is the weighted process pattern graph from the STEP-1 and STEP-2 algorithms and the other is the structured information control net from the STEP-3 algorithm. In addition, in order to emphasize the functional correctness and perfectness of the proposed algorithm, we display the enlarged building block with being indicated by three boxes, each of which is marked with the corresponding primitive process patterns. All of these graph models in the captured-screens are in a visual expression of the  $\rho$ -Algorithm and mined from all the 10,000 temporal work-cases (process instance event traces) in the enactment event log dataset of the Large Bank Transaction Process Model. Finally, Figure 17 is the



GraphViz visualization form of the SICN-oriented process model discovered from the IEEE XES-formatted synthetic dataset, at last. Based upon these results of the experiment, the  $\rho$ -Algorithm of the conceptual and theoretical approach and its implemented process mining system proposed in the paper ought to be reasonable and feasible in terms of their deployments and applications in the real world.

## V. CONCLUSION

So far, this paper has proposed the SICN-oriented process mining framework and its related stepwise algorithms, such as the STEP-1, STEP-2 and STEP-3 algorithms, under the name of  $\rho$ -Algorithm. To validate the proposed algorithmic framework, we successfully implemented all the stepwise algorithms and carried out an experiment based upon the IEEE XES-formatted synthetic dataset of process enactment event logs recorded from simulating and enacting the Large Bank Transaction Process Model. The theoretical background of the proposed algorithmic framework stems from the conceptual discovery approach dealing with the SICN-oriented process patterns, such as linear, disjunctive, conjunctive and repetitive process patterns, whereas the implementable background of the proposed algorithmic framework is supported by a series of the procedural mining functions being concretized as the **Algorithm 1**, **Algorithm 2** and **Algorithm 3**, which are the stepwise algorithmic components of the  $\rho$ -Algorithm, for discovering the structured information control nets from an IEEE XES-formatted dataset of process enactment event logs. Based upon these theoretical and algorithmic approaches, the paper devised, implemented and developed these concepts, algorithms and systems, respectively. By using the implemented  $\rho$ -Algorithm, the paper carried out two cases of experimental analysis to help for validating the discovery perfectness of the  $\rho$ -Algorithm, effectively and efficiently, and both cases of which are related with a specific subprocess building block with the disjunctive and conjunctive process pattern types as well as another subprocess building block with the repetitive process pattern type chosen from the Large Bank Transaction Process Model, respectively.

Summarily, the characteristics of the SICN-oriented process mining framework proposed and implemented in the paper are as followings: First, the proposed algorithmic framework is able not only to discover the SICN-oriented process patterns but also to mine the enactment occurrences of the process patterns from an IEEE XES-formatted dataset of process enactment event logs. Second, the proposed algorithmic framework is theoretically supported by the information control nets modeling methodology of workflow and business process models. Third, the essential algorithm of the proposed algorithmic framework is named as  $\rho$ -Algorithm that is able to properly deal all the structural process patterns, such as linear (sequential), conjunctive (parallel-AND), disjunctive (exclusive-OR), and repetitive (iterative-LOOP) process patterns, with keeping the structural properties of matched-pairing and proper-nesting by using the

concept of  $\rho$ -function returning the enactment occurrences of the associated activities. Fourth, the  $\rho$ -Algorithm is the first SICN-oriented process mining framework successfully developed and experimentally proved in the workflow and business process management and mining literature. Fifth, the  $\rho$ -Algorithm is also working the discovery perfectness on such datasets recorded from the unstructured process models. As the future work, we need to develop a kind of data preprocessing techniques that are able to handle various types of noises embedded in process enactment event log datasets.

## REFERENCES

- [1] K. Kim and C. A. Ellis,  $\sigma$ -Algorithm: Structured Workflow Process Mining Through Amalgamating temporal workcases (Lecture Notes in Computer Science), vol. 4426. Berlin, Germany: Springer, 2007, pp. 119–130.
- [2] W. M. van der Aalst, H. A. Reijers, A. J. Weijters, B. F. van Dongen, A. A. D. Medeiros, M. Song, and H. Verbeek, "Business process mining: An industrial application," *Inf. Syst.*, vol. 32, no. 5, pp. 713–732, 2007.
- [3] W. Van Der Aalst, A. Adriansyah, A. K. A. De Medeiros, F. Arcieri, T. Baier, T. Blicke, J. C. Bose, P. Van Den Brand, R. Brandtjen, J. Buijs, and A. Burattin, *Process Mining Manifesto* (Lecture Notes in Business Information Processing), vol. 99. Berlin, Germany: Springer, 2011, pp. 169–194.
- [4] W. M. van der Aalst, *Process Cubes: Slicing, Dicing, Rolling Up and Drilling Down Event Data for Process Mining* (Lecture Notes in Business Information Processing), vol. 159. Cham, Switzerland: Springer, 2013, pp. 1–22.
- [5] 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*. [Online]. Available: <http://arxiv.org/abs/1703.03740>
- [6] 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," in *Proc. 9th EAI Int. Conf. Big Data Technol. Appl. (BDTA)*, Exeter, U.K., Sep. 2018, pp. 1–8.
- [7] C. A. Ellis, A. J. Rembert, K. H. Kim, and J. Wainer, *Beyond Workflow Mining* (Lecture Notes in Computer Sciences), vol. 4102. Berlin, Germany: Springer, 2006, pp. 49–64.
- [8] J. Won, "A framework: Organizational network discovery on workflows," Ph.D. dissertation, Dept. Comput. Sci., Kyonggi Univ., Suwon, South Korea, Feb. 2008.
- [9] D. Grigori, F. Casati, M. Castellanos, U. Dayal, M. Sayal, and M.-C. Shan, "Business process intelligence," *J. Comput. Ind.*, vol. 53, no. 3, pp. 321–343, 2004.
- [10] H. Ahn and K. P. Kim, "Organizational closeness centralities of workflow-supported performer-to-activity affiliation networks," *IEEE Access*, vol. 9, pp. 48555–48582, 2021.
- [11] M. Park and K. Kim, "Control-path oriented workflow intelligence analyses," *J. Inf. Sci. Eng.*, vol. 24, no. 2, pp. 343–359, 2008.
- [12] H. Ahn and K. P. Kim, "Formal approach for discovering work transference networks from workflow logs," *Inf. Sci.*, vol. 515, pp. 1–25, Apr. 2020. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0020025519310813>
- [13] 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.
- [14] M. Park and K. Kim, "A workflow event logging mechanism and its implications on quality of workflows," *J. Inf. Sci. Eng.*, vol. 26, no. 5, pp. 1817–1830, 2010.
- [15] H. Ahn, D.-L. Pham, and K. P. Kim, "An experimental analytics on discovering work transference networks from workflow enactment event logs," *Appl. Sci.*, vol. 9, no. 11, pp. 2368–2390, Jun. 2019.
- [16] T. C. for Research Data, "BPI challenges," BPM, 2012, 2013, 2014, 2015, 2016, 2017, 2018.
- [17] *IEEE Standard for Extensible Event Stream (XES) for Achieving Interoperability in Event Logs and Event Streams*, Standard IEEE 1849-2016, 2016.
- [18] 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.



- [19] K. Kim, Y.-K. Lee, M. Jin, H. Ahn, and K. P. Kim, "An experiment on mining proportions of disjunctive process patterns from workflow logs," in *Proc. 13th Asia Pacific Int. Conf. Inf. Sci. Technol. (APIC-IST)*, Nhatrang, Vietnam, Jun. 2018, pp. 265–268.
- [20] K. Kim and C. A. Ellis, *Section II / Chapter VII. An ICN-based Workflow Model and Its Advances, Handbook of Research on BP Modeling*. Hershey, PA, USA: IGI Global, ISR, 2009.
- [21] B. F. Van Dongen, A. K. A. de Medeiros, H. Verbeek, A. Weijters, and W. M. Van Der Aalst, *The Prom Framework: A New Era in Process Mining Tool Support* (Lecture Notes in Computer Science), vol. 3536. Berlin, Germany: Springer, 2005, pp. 444–454.
- [22] A. Polyvyanyy, W. van der Aalst, A. H. M. ter Hofstede, and M. T. Wynn, "Impact-driven process model repair," *ACM Trans. Softw. Eng. Methodol.*, vol. 25, no. 4, p. 28, 2017.
- [23] A. Pika, M. Leyer, M. T. Wynn, C. J. Fidge, A. H. M. T. Hofstede, and W. M. P. V. D. Aalst, "Mining resource profiles from event logs," *ACM Trans. Manage. Inf. Syst.*, vol. 8, no. 1, pp. 1–30, May 2017.
- [24] C. A. Ellis, K. Kim, A. Rembert, and J. Wainer, "Investigations on stochastic information control nets," *Inf. Sci.*, vol. 194, pp. 120–137, Jul. 2012.
- [25] K.-S. Kim, "Proportional process pattern discovery and its experimental analyses," Ph.D. dissertation, Dept. Comput. Sci. Eng., Kyunghee Univ., Suwon, South Korea, Feb. 2019.
- [26] D.-L. Pham, H. Ahn, and K. P. Kim, "Discovering redo-activities and performers' involvements from XES-formatted workflow process enactment event logs," *KSII Trans. Internet Inf. Syst.*, vol. 13, no. 8, pp. 4108–4122, Aug. 2019.
- [27] W. Z. Low, W. M. P. van der Aalst, A. H. M. ter Hofstede, M. T. Wynn, and J. D. Weerd, "Change visualisation: Analysing the resource and timing differences between two event logs," *Inf. Syst.*, vol. 65, pp. 106–123, Apr. 2017.
- [28] L. Märušter and N. R. van Beest, "Redesigning business processes: A methodology based on simulation and process mining techniques," *Knowl. Inf. Syst.*, vol. 21, no. 3, p. 267, 2009.
- [29] A. A. Kalenkova, W. M. P. van der Aalst, I. A. Lomazova, and V. A. Rubin, "Process mining using BPMN: Relating event logs and process models," *Softw. Syst. Model.*, vol. 16, no. 4, pp. 1019–1048, 2017.
- [30] 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. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0020025520306666>
- [31] I. Yürek, D. Birant, Ö. E. Yürek, and K. U. Birant, "Time-oriented interactive process miner: A new approach for time prediction," *TURKISH J. Electr. Eng. Comput. Sci.*, vol. 29, no. 1, pp. 122–137, Jan. 2021.
- [32] S.-H. Ham, H. Ahn, and K. P. Kim, "LSTM-based business process remaining time prediction model featured in activity-centric normalization techniques," *J. Internet Comput. Services*, vol. 21, pp. 83–92, Mar. 2020.
- [33] D.-L. Pham, H. Ahn, K.-S. Kim, and K. P. Kim, "Process-aware enterprise social network prediction and experiment using LSTM neural network models," *IEEE Access*, vol. 9, pp. 57922–57940, 2021.
- [34] K. P. Kim, "An XPDL-based workflow control-structure and data-sequence analyzer," *KSII Trans. Internet Inf. Syst.*, vol. 13, no. 3, pp. 1702–1721, 2019.
- [35] M. Park and K. Kim, *XWELL: A XML-Based Workflow Event Logging Mechanism and Language for Workflow Mining Systems* (Lecture Notes in Computer Science), vol. 4707. Berlin, Germany: Berlin, 2007, pp. 900–909.
- [36] M. Z. Muehlen and K. D. Swenson, *BPAF: A Standard for the Interchange of Process Analytics Data* (Lecture Notes in Business Information Processing), vol. 66. Berlin, Germany: Springer, 2011, pp. 170–181.
- [37] M.-J. Kim, H. Ahn, and M.-J. Park, "A GraphML-based visualization framework for workflow-performers' closeness centrality measurements," *KSII Trans. Internet Inf. Syst.*, vol. 9, no. 8, pp. 3216–3230, 2015.
- [38] M.-J. Kim, H. Ahn, and M.-J. Park, "A theoretical framework for closeness centralization measurements in a workflow-supported organization," *KSII Trans. Internet Inf. Syst.*, vol. 9, no. 9, pp. 3611–3634, 2015.
- [39] K. Kim, Y. Lee, H. Ahn, and K. P. Kim, "An experimental mining and analytics for discovering proportional process patterns from workflow enactment event logs," *Wireless Netw.*, pp. 1–8, Dec. 2018.
- [40] D.-L. Pham, H. Ahn, M.-J. Park, K.-S. Kim, and K. P. Kim, "A proportional process mining system," in *Proc. 19th Ind. Conf. Data Mining (ICDM)*, New York, NY, USA, Jul. 2019, p. 58.
- [41] L. Computer Science Laboratory. (Apr. 2016). *Graphstream—A Dynamic Graph Library*. [Online]. Available: <http://graphstream-project.org/>



**KYOUNG-SOOK KIM** received the B.S. degree in computer science from Kyonggi University, South Korea, in 1984, and the M.S. and Ph.D. degrees in computer science and engineering from Kyunghee University, South Korea, in 2004 and 2019, respectively. She is currently a Lecturer with the Division of AI Computer Science and Engineering, Kyonggi University. She is also a Research Member of the Contents Convergence Software Research Institute, Kyonggi University. Her research interests include data and knowledge engineering, process-aware enterprise information systems, business process intelligence, process mining, and deep learning and knowledge engineering using active contents big data.



**DINH-LAM PHAM** received the B.S. and M.S. degrees in computer science from Thai Nguyen University, Vietnam, in 2008 and 2010, respectively, and the Ph.D. degree in computer science from Kyonggi University, South Korea, in 2021. He is currently a Research Professor with the Contents Convergence Software Research Institute and a Research Member with the Data and Process Engineering Research Laboratory, Kyonggi University. His research interests include process mining, deep learning-based process predicting, workflow systems, workflow-supported social and affiliation networks discovery and analysis, and process-aware Internet of Things architecture and services.



**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 Chungang University, South Korea, in 1986, and the M.S. and Ph.D. degrees in computer science from the University of Colorado Boulder, USA, in 1994 and 1998, respectively. He had been the Dean of the Computerization and Informatics Institute, Kyonggi University, from 2017 to 2021. Since 2007, he has been the Founder and the Director in charge of the Contents Convergence 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 had worked as a Researcher and a Developer at Aztek Engineering, American Educational Products Inc., and IBM, USA, and a Research Member of Staff at 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, South Korea. 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 journal of KSII and the committee member of the several conferences and workshops.

...