Information Sciences 515 (2020) 1–25

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Formal approach for discovering work transference networks from workflow logs

Hyun Ahn, Kwanghoon Pio Kim*

Department of Computer Science and Engineering, Kyonggi University, Seoul, South Korea

ARTICLE INFO

Article history:
Received 12 February 2018
Revised 25 September 2019
Accepted 24 November 2019
Available online 6 December 2019

Keywords:
Workflow model
Information control net
Workflow intelligence
Workflow event log
Workflow discovery
Workflow rediscovery
Work transference network
Workflow-supported organizational social network

ABSTRACT

This paper proposes a formal principle for discovering work transference networks of workflow-supported organizational employees from workflow enactment histories contained in event logs. This originates from the strong belief that those work transference networks, hidden in workflow enactment activities and histories, can both connote the degree of work sharing and work relevancy between workflow performers and denote their degree of work intensity. In this paper, we devise a series of formal definitions and algorithms for discovering a work transference network from a workflow procedure, and from its enactment histories in event logs. The final goal of the paper is to theoretically build the principle of fidelity into workflow human resource planning and its performance, via a novel concept of work transference networks that can be discovered from a workflow model and, moreover, rediscovered from its enactment history. In deploying the proposed principle, we base the formal representation on information control net theory, which can graphically and mathematically represent workflow procedures. We apply directed graph theory to formally and graphically define the work transference network model proposed in this paper. For the sake of verifying and validating the proposed concepts and algorithms, we implement a work transference network rediscovery system, and apply it to a workflow enactment event log dataset, in an experimental study. Finally, we describe the implications of discovering and rediscovering work transference networks, as a human resource management and evaluation technique, in workflow-supported enterprises and organizations.

© 2019 Elsevier Inc. All rights reserved.

1. Introduction

A workflow[1] (or business process) management system is able to support two fundamental functionalities|modeling functionality and enacting functionality. The modeling functionality allows modelers to define, analyze, and maintain workflow procedures by associating all entities essential to workflow—such as activities, roles, performers, relevant data, and invoked applications—with the corresponding procedures. In contrast, the enacting functionality enables performers to play the essential roles of invoking, executing, and monitoring the instances of the workflow procedures. The logical foundation of such a workflow management system is based upon its underlying workflow model, which implies that the system is able to automate the definition, creation, execution, and management of workflow procedures according to the internal principles and

* Corresponding author.
E-mail address: kwang@kgu.ac.kr (K.P. Kim).

[1] The term *w*orkflow can be used interchangeably with the term *b*usiness process. We prefer the former to the latter in this paper.

https://doi.org/10.1016/j.ins.2019.11.036
0020-0255/© 2019 Elsevier Inc. All rights reserved.

structure of the underlying workflow model. Several workflow models [1] have been proposed in the workflow literature, and almost all of them employ the five essential entity types—such as activity, role, performer, repository, and application—to represent organizational workers and their procedural collaborations. We turn our attention to the performer entity type in this paper.

In recent years, the workflow literature has started to be focused on "people" working in workflow-supported organizations, because it is widely accepted for a workflow system to be a "people system". By analyzing the social relationships and collaborative behaviors between the people who are involved in enacting workflow models, we are able to measure and estimate their overall performance in the real organization, and their working productivity as well. The authors' research group has proposed the research and development issues of applying the concept of social networks, and their analysis methods, to human-centered workflow knowledge discovery [2,3] and analysis [4–6]. In this paper, we are particularly interested in the work transference relationships between the performers who participate in the enactment of a specific workflow procedure, which can be formally and graphically represented by a work transference network model.

There exist two main branches of research on the work transference network model. One branch involves *discovery* issues, and the other involves *rediscovery* issues. The former aims to discover a work transference network by analyzing a specific workflow model, whereas the latter is concerned with mining a work transference network from workflow event logs from the model. More specifically, we can differentiate them as follows: the former is to construct a planned work transference network, whereas the latter is to construct an enacted work transference network. This paper directly concerns the work transference network rediscovery issue, which means that the proposed approach can discover an enacted work transference network from a specific workflow's enactment event logs. In conclusion, the purpose of this paper is to originate a fundamental principle for rediscovering a work transference network from a specific workflow's event logs, through which we are able to graphically visualize, and mathematically measure, the human-centered organizational work transference relationships and their intensities that are formed by enacting the corresponding workflow model.

2. Information control nets

The theoretical background is the information control net (ICN) methodology [1], which is a typical workflow modeling approach supporting graphical and formal representations. In defining a workflow procedure, the methodology uses the basic workflow entity types—activity, role, actor/performer, invoked application, and transition condition—to represent the procedural properties of workflow, such as control flow and data flow, as well as the associative properties of workflow—such as the activity-to-role, role-to-performer, activity-to-condition/rule, and activity-to-application associations. In this section, we define the formal representation of the ICN of a workflow model, through Definition 1.

Definition 1. *ICN*. A basic **ICN** is an 8-tuple $\Gamma = (\delta, \rho, \lambda, \varepsilon, \pi, \kappa, \mathbf{I}, \mathbf{O})$ over a set $\mathbf{A}$ of activities[2] (including a set of group activities), a set $\boldsymbol{T}$ of transition conditions, a set $\boldsymbol{D}$ of data repositories, a set $\boldsymbol{G}$ of invoked application programs, a set $\boldsymbol{R}$ of roles, and a set $\boldsymbol{P}$ of performers (including a set of performer groups), where the following hold.

- External Data Properties
  - $\boldsymbol{I}$ is a finite set of initial input repositories, which is assumed to be loaded from some external workflow before execution.
  - $\boldsymbol{O}$ is a finite set of final output repositories, which is assumed to be transferred to some external workflow after execution.
- Procedural Properties
  - $\delta = \delta_i \cup \delta_o$,
    where δ_o: $\mathbf{A} \longrightarrow \wp(\mathbf{A})$[3] is a multi-valued function mapping an activity to its set of (immediate) successors, and δ_i: $\mathbf{A} \longrightarrow \wp(\mathbf{A})$ is a multi-valued function mapping an activity to its set of (immediate) predecessors.
  - $\rho = \rho_i \cup \rho_o$,
    where ρ_o: $\mathbf{A} \longrightarrow \wp(\mathbf{D})$ is a multi-valued function mapping an activity to its set of output repositories, and ρ_i: $\mathbf{A} \longrightarrow \wp(\mathbf{D})$ is a multi-valued function mapping an activity to its set of input repositories.
- Associative Properties
  - $\lambda = \lambda_a \cup \lambda_g$,
    where λ_g: $\mathbf{A} \longrightarrow \mathbf{G}$ is a single-valued function mapping a task-type activity to its invoked application program, and λ_a: $\mathbf{G} \longrightarrow \wp(\mathbf{A})$ is a multi-valued function mapping an invoked application program to its set of associated task-type activities.
  - $\varepsilon = \varepsilon_a \cup \varepsilon_r$,
    where ε_r: $\mathbf{A} \longrightarrow \mathbf{R}$ is a single-valued function mapping a task-type activity to a role, and ε_a: $\mathbf{R} \longrightarrow \wp(\mathbf{A})$ is a multi-valued function mapping a role to its set of associated task-type activities.
  - $\pi = \pi_r \cup \pi_p$,
    where π_p: $\mathbf{R} \longrightarrow \wp(\mathbf{P})$ is a multi-valued function mapping a role to its set of associated performers, and π_r: $\mathbf{P} \longrightarrow \wp(\mathbf{R})$ is a multi-valued function mapping a performer to its set of associated roles.

[2] It is assumed that activities are classified into three activity types: events, gateways, and tasks.

[3] $\wp(\mathbf{A})$ is the powerset of $\mathbf{A}$.

- $\kappa = \kappa_i \cup \kappa_o$,
  where κ_i: $(\mathbf{A} \times \mathbf{A}) \longrightarrow \wp(\mathbf{T})$ is a multi-valued function mapping a set of control-transition conditions, $\boldsymbol{T}$, on directed edges (ordered pairs), $(\delta_i(\alpha), \alpha \in \mathbf{A})$, from $\delta_i(\alpha)$ to α; and κ_o: $(\mathbf{A} \times \mathbf{A}) \longrightarrow \wp(\mathbf{T})$ is a multi-valued function mapping a set of control-transition conditions, $\boldsymbol{T}$, on directed edges (ordered pairs), $(\alpha \in \mathbf{A}, \delta_o(\alpha))$, from α to $\delta_o(\alpha)$.

Starting and Terminating Nodes. The execution of a workflow model commences with a single χ transition condition. Therefore, we always assume without the loss of generality that there is a single starting event node (α_I), which has no source node (that is, it has no incoming edges). At the commencement, it is also assumed that all input repositories in the set I have been initialized with relevant data from the external workflow model:

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

The execution is terminated with any single λ outgoing transition condition. In addition, we assume, without loss of generality, that there is a single terminating event node (α_F), which has no destination node (that is, it has no outgoing edges). A set of output repositories O is a group of data holders that are transferred to the external workflow model after termination:

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

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

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

which can be interpreted as follows:

- Upon the completion of an activity, α, with a single incoming transition, it simultaneously initiates all of the activities β_{i1} through $\beta_{im(i)}$; in this case, all of the initiated transitions are called parallel (conjunctive) transitions.
- After completing an activity, α, with a single incoming transition, only one value out of n with $[m(i) = 1 \wedge 1 \leq i \leq n]$ is selected as the result of a decision made; in this case, the selected transition is called a decision (disjunctive or exclusive-OR) transition.
- After completing an activity, α, with a single incoming transition, if $n = 1 \wedge m(n) = 1$, then neither decision nor parallelism is needed, and the transition is called a sequential transition.
- After completing an activity, α, with more than one incoming transition, if $1 \leq n \leq 2 \wedge m(n) = 1$, then only one value out of $i(1 \leq i \leq n)$ is selected as the result of a decision made; in this case, the selected transition is called a loop-initiation (repetitive) transition.

The backward temporal ordering of activities in the corresponding workflow model can also be formally represented. In this paper, we will not describe this formal definition in detail, because it can be straightforwardly represented according to the forward temporal ordering of activities.

Graphical Formation. Based on the interpretation, we graphically define several primitive transition types. The conjunctive (or parallel) outgoing transitions are connected by a solid dot ($\bullet$) with a single incoming transition, and a disjunctive (or decision) outgoing transition is connected by a hollow dot ($\circ$) with a single incoming transition. These special types of nodes are called gateway nodes in the workflow literature. The starting and terminating event nodes are a medium-sized circle with a thin line and a medium-sized circle with a thick line, respectively. Naturally, activities and roles are represented by large circles and diamonds, respectively. The gateway nodes need to be formed in matched pairs of split and join types. In addition, multiple pairs of split and join gateway nodes should be kept in a properly nested form, to syntactically support safety and soundness.

The example in Fig. 1 depicts an ICN for the library book acquisition workflow procedure, which was first introduced in [7]. The left-hand side of the figure is the control flow aspect of the ICN, which consists of a starting event node, a terminating event node, 12 task-type activities, a pair of split and join parallel gateway-type nodes, a pair of split and join decision gateway-type nodes, and a pair of loop-initiation and loop-termination gateway-type nodes.

Assigning Roles and Performers. For a specific task-type activity α, $\varepsilon_r(\alpha) = \{\eta\}$ means that the task-type activity α is associated with the role, η. In addition, $\varepsilon_a(\eta) = \{\alpha_1, \alpha_2, \ldots, \alpha_m\}$, where m is the number of task-type activities associated with the role, means that a specific role η can be associated with several task-type activities in a workflow model. In terms of the role-to-performer association, for any role η, $\pi_p(\eta) = \{p_1, p_2, \ldots, p_n\}$, where n is the number of performers assigned to the role, means that one or more performers (participants) should be assigned to perform a task-type activity via a role. A role is a named designator for one or more participants, who share work skills, access controls, execution controls, and

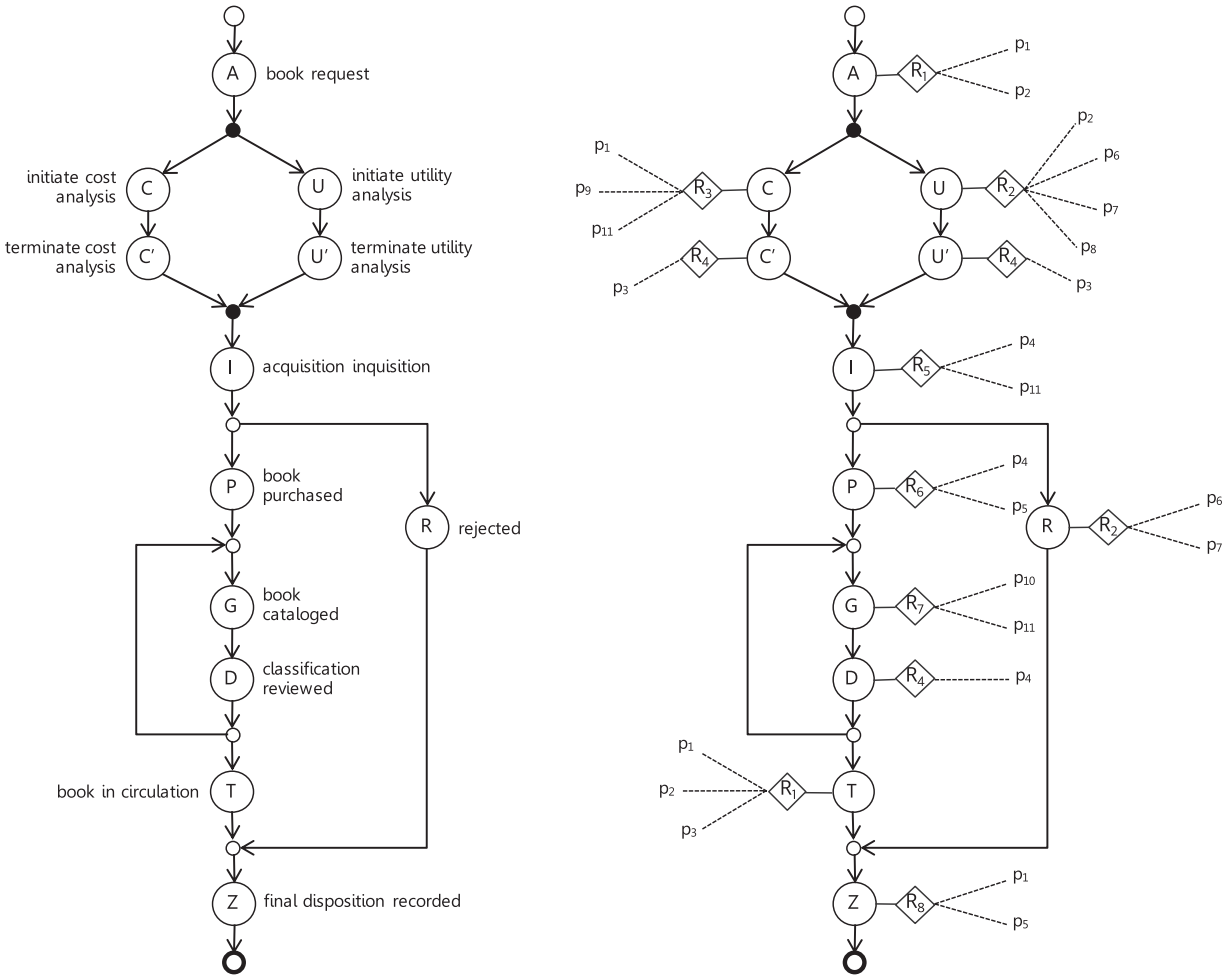

Fig. 1. An example ICN for the library book acquisition workflow model.

authority/responsibility. In addition, for any performer p, $\pi_r(p) = \{\eta_1, \eta_2, \ldots, \eta_m\}$, where m is the number of roles assigned to the performer, means that a specific performer can be assigned to several roles at the same time, and he/she fulfills the roles via their associated task-type activities in a workflow model.

The right-hand side of Fig. 1 shows a graphical formation, representing the role and performer assignment status, in the example workflow model. As the figure shows, the following are some examples in the formal representations for role r_2 and performer p_1 assignments: $\varepsilon_r(\mathrm{U}) = \{r_2\}$, $\varepsilon_a(r_2) = \{\mathrm{U}, \mathrm{R}\}$, $\pi_p(r_2) = \{p_2, p_6, p_7, p_8\}$, and $\pi_r(p_1) = \{r_1, r_3, r_8\}$. In this paper, we particularly focus on these activity-to-role and role-to-performer associative properties, which are directly related to both forming a work transference network in a workflow procedure and discovering the work transference network from the enactment event logs of the corresponding workflow procedure.

3. Work transference networks

In this section, we formally describe the basic concept and definition of work transference networks, which are formed by modeling artifacts of workflow procedures. We know that the procedural activities in a workflow procedure eventually trigger work transferences between the performers who are involved in the workflow procedure. In general, analyzing work transferences between performers is one of the essential planning activities in human resource management and decision-making support management [8]. In particular, the work transference relationships in performing a workflow procedure capture the essential knowledge and criteria for evaluating and validating the human resource performance of the corresponding workflow procedure.

As a formalism to represent such knowledge about work transferences, we formally define a buildtime work transference network model[4], in Definition 2. A work transference network is the formal and graphical structure of a directed graph

[4] The model is discovered from a workflow model defined at build time (the time when the workflow model is constructed).

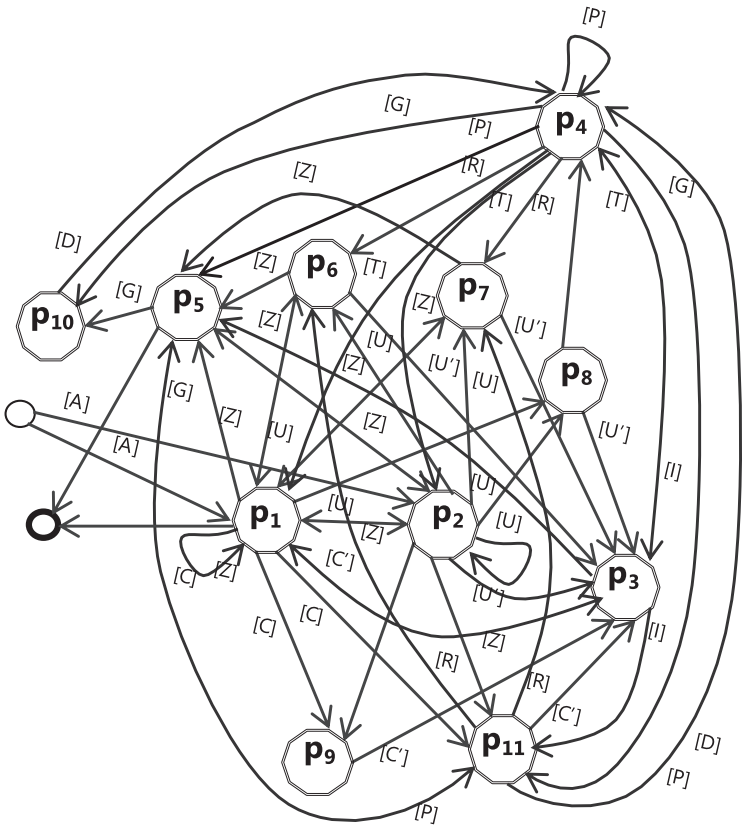

Fig. 2. A buildtime work transference network from the library book acquisition workflow model.

(or digraph) model to represent the relationship of work (activity) transferences, and their associated workitems, between performers who are involved in a corresponding workflow procedure. Each node represents a performer, and each ordered pair of nodes (or directed edge) represents a work transference relationship. Through the formal notation $\sigma\,(=\sigma_i \cup \sigma_o)$, we define the work transference relationships, in which the source and destination nodes of a directed edge represent the transferrer and receiver of work. We also define the formal notation $\psi\,(=\psi_i \cup \psi_o)$ for work association relationships, by labeling each directed edge with the names of workitems that are transferred to the destination node from the source node of the corresponding work transference relationship.

Definition 2. *(Buildtime) Work Transference Network Model.* A buildtime work transference network model is formally defined as $\Lambda^B = (\sigma, \psi, F_r^B, T_o^B)$, over a set **P** of performers, and a set **A** of activities in a workflow model, where the following hold.

- F_r^B is a finite set of coordinators, or coordinator groups, connected from an external buildtime model of the work transference network.
- T_o^B is a finite set of coordinators, or coordinator groups, connected to an external buildtime model of the work transference network.
- $\sigma = \sigma_i \cup \sigma_o$ /* Work (Activity) Transferences */
  - $\sigma_o : \mathbf{P} \longrightarrow \wp(\mathbf{P})$ is a multi-valued function mapping a performer to its set of (immediate) work (activity) transferrers.
  - $\sigma_i : \mathbf{P} \longrightarrow \wp(\mathbf{P})$ is a multi-valued function mapping a performer to its set of (immediate) work (activity) receivers.
- $\psi = \psi_i \cup \psi_o$ /* Work (Activity) Associations */
  - $\psi_i : (\mathbf{P} \times \mathbf{P}) \longrightarrow \wp(\mathbf{A})$ is a multi-valued function returning a set of receiving workitems (activities) on ordered pairs of performers, $(\sigma_i(o), o)$, $o \in \mathbf{P}$, from $\sigma_i(o)$ to o.
  - $\psi_o : (\mathbf{P} \times \mathbf{P}) \longrightarrow \wp(\mathbf{A})$ is a multi-valued function returning a set of transferring workitems (activities) on ordered pairs of performers, $(o, \sigma_o(o))$, $o \in \mathbf{P}$, from o to $\sigma_o(o)$.

Fig. 2 is a buildtime work transference network that graphically illustrates the work transference and work association relationships of the library book acquisition example from the previous section. The network is composed of 11 performers, p_1–p_{11}, and their ordered pairs, labeled with the associated workitems in the activity set, $\mathbf{A} =$

{A, C, C′, U, U′I, P, G, D, R, T, Z}, of the workflow procedure. By convention[5], we adopt a bidirectional arrow to draw a pair of *strongly connected* nodes—such as (p_5, p_{11}), (p_1, p_2), (p_1, p_3), (p_1, p_6), (p_2, p_5), and (p_3, p_4)—and we locate each edge label graphically near to the location of the source node. We define the formal implications of strongly connected nodes, and a strongly connected digraph, in Definition 3.

Definition 3. *Strongly Connected Nodes and Digraph.*

- Nodes u and v are *strongly connected* if there is both a directed edge, (u, v), and a directed edge, (v, u), in a work transference network.
- The digraph of a work transference network, Λ, is *strongly connected* if every pair of nodes is *strongly connected*.

Fortunately, we are able to automatically construct a work transference network from the ICN of a workflow procedure. By revising the algorithm [9] for generating an actor-oriented workflow model from an ICN, we develop an algorithm that is able to build a formal model of the work transference network from a formal model of the ICN. Therefore, we apply the newly revised algorithm to an ICN and automatically discover a work transference network model of the corresponding workflow procedure. Algorithm 1 is the newly revised algorithm to discover a formal model of the work transference network from a formal model of the ICN.

Algorithm 1 BUILDTIME WTN DISCOVERY ALGORITHM: An Algorithm for Discovering a Buildtime Work Transference Network from an ICN.

Require: An ICN, $\Gamma = (\delta, \rho, \lambda, \varepsilon, \pi, \kappa, \mathbf{I}, \mathbf{O})$;
Ensure: A buildtime work transference network model, $\Lambda^B = (\sigma, \psi, F_r^B, T_o^B)$;

procedure MAIN ▷ Discovering a work transference network model
  for $\forall \alpha \in \mathbf{A}$ **do**
    ▷ $\sigma = \sigma_i \cup \sigma_o$: Discovering the work (activity) transferences
    $\sigma_i(\textit{each performer of } \pi_p(\varepsilon_r(\delta_o(\alpha)))) \leftarrow \textit{all of } \pi_p(\varepsilon_r(\alpha))$;
    $\sigma_o(\textit{each performer of } \pi_p(\varepsilon_r(\alpha))) \leftarrow \textit{all of } \pi_p(\varepsilon_r(\delta_o(\alpha)))$;
    ▷ $\psi = \psi_i \cup \psi_o$: Discovering the work (activity) associations
    $\psi_i(\textit{each performer of } \pi_p(\varepsilon_r(\delta_o(\alpha)))) \leftarrow \textit{all in } (\alpha, o), \forall o \in \pi_p(\varepsilon_r(\alpha))$;
    $\psi_o(\textit{each performer of } \pi_p(\varepsilon_r(\alpha))) \leftarrow \textit{all in } (\alpha, o), \forall o \in \pi_p(\varepsilon_r(\delta_o(\alpha)))$;
  end for
end procedure

4. A formal discovery approach

In this section, we formally describe an approach, and its core concepts and algorithms, to discovering a work transference network from a collection of event traces in the execution log of a workflow procedure. To deploy the conceptual formalization of the discovery approach, we axiomatically define a series of formal concepts, such as event trace, temporal workcase, loopcase, and temporal transference. A single event trace is the execution log of a workflow instance, and its named formalism is a temporal workcase model, which was first introduced in [10]. Each event trace has its own performer trace, which is formally named a temporal transference model. Together with these formal concepts, we need to revise the formal definition of the work transference network model given in the previous section, because duplication of work associations on a single directed edge must be allowed in representing a discovered work transference network. In conclusion, the work transference networks discovered from a collection of event traces will differ slightly from those built from ICNs by the algorithm. Incidentally, a discovered work transference network itself provides a very valuable intuition about the strength of the work transference relationship between every pair of (transferring and receiving) performers, in performing the corresponding workflow procedure.

4.1. Temporal workcases and temporal worktransferences

As a workflow instance is executed, the logging and auditing component of the workflow enactment engine records its activity execution events in a log, and these logged events are arranged in a temporal sequence. This execution sequence of a workflow instance forms a *workflow instance event trace*, from which we can extract a *workflow instance activity trace*; its formal representation is specified by a *temporal workcase* model. The execution sequence of a workflow instance also produces a performing sequence of the performers (a *workflow instance performer trace*) who are in charge of executing the workitems in the corresponding workflow instance. We can also extract a workflow instance performer trace from a workflow instance event trace; its formal representation is specified by a *temporal worktransference* model. Here, we formally describe the discovery approach, by defining a workflow activity event log, in Definition 4.

[5] However, this is only for visual simplicity in the graphical representation of work transference networks.

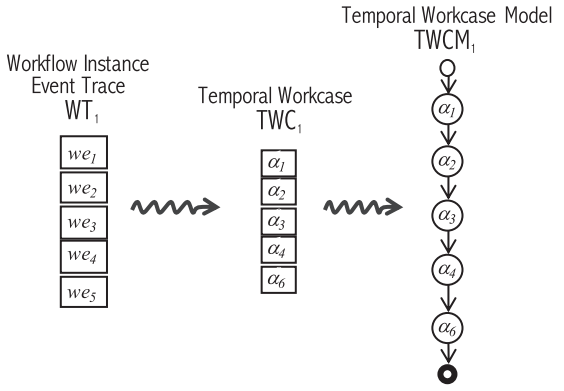

Fig. 3. Formal components for discovering temporal workcases.

Definition 4. *Workflow Activity Event Log.* Let ***we*** = (*wi, pc, wf, wc, ac, p**, *t, s*) be a workflow activity (workitem) event stored in a log, where

- *wi* is a workitem (activity instance) identifier,
- *pc* is a workflow package identifier,
- *wf* is a workflow process identifier,
- *wc* is a workflow instance (workcase) identifier,
- *ac* is an activity identifier,
- *p*[6] is a participant (or performer) identifier,
- *t* is a timestamp, and
- *s* is a workitem's current state, such as *ready, assigned, reserved, running, completed,* or *cancelled*.

In practice, we assume the workflow activity event log to be stored in the format of a tag-based language (XML schema). An XML-based workflow activity event log format, XWELL[7] [11], for the purpose of workflow mining, has been studied and proposed by the authors' research group, and the WfMC has released the standardized audit and log specification, BPAF[8] [12]. In recent years, IEEE has released a standard tag-based language, XES[9] [13], whose aim is to provide designers of information systems with a unified and extensible methodology for capturing the behavior of systems by means of event logs and event streams. As the format of the workflow activity event log structure, we can use the "XES Schema", to describe the structure of an XES event log or stream, and the "XES extension" to describe the structure of an extension of such a log or stream. Based upon one of the formats, we simply define the XML schema with the essential attributes in the workflow activity event log, as follows:

- The EVENT attribute is used to specify an event identifier, which is assigned by the workflow enactment engine.
- The WORKITEM attribute of a workflow activity event represents a workitem identifier that is uniquely assigned by using combined identifiers, such as `PACKAGE ID`, `WORKFLOW ID`, `ACTIVITY ID`, and `INSTANCE ID`.
- The PARTICIPANT attribute is used to specify the performer who is in charge of enacting the workitem.
- The TIMESTAMP attribute specifies the time when the event occurred.
- Finally, the STATE attribute represents the workitem's runtime state, maintained by the engine. Whenever the workitem's state is changed, it is logged with the eventcode `WMChanged WorkitemState`. This should be a state such as `READY`, `ASSIGNED`, `RESERVED`, `RUNNING`, `COMPLETED`, and `CANCELLED`.

Definition 5. *Workflow Instance Event Trace.* Let WT(***c***) be the workflow instance event trace of a workcase, ***c***, where $WT(c) = (we_1, \ldots, we_n)$ and $\{we_i \mid we_i.wc = \mathbf{c} \wedge we_i.t \leq we_j.t \wedge we_i.pc = we_j.pc \wedge we_i.wf = we_j.wf \wedge we_i.wc = we_j.wc \wedge i < j \wedge 1 \leq i, j \leq n\}$. This formally represents a temporally ordered workflow activity event sequence of a specific workcase, which is constructed by preprocessing the workflow instance event logs using the `TIMESTAMP` and the `STATE` attributes.

From the formal definition of a workflow instance event trace, WT(***c***) (where ***c*** is a workcase identifier), in Definition 5, we construct a temporal workcase model, TWCM(***c***), and a temporal worktransference model, TWTM(***c***), as shown in Figs. 3 and 4, respectively. A meaningful temporal order in managing workflow instances (workcases) should be based upon one of the following instantaneous points of time, each of which is named a *timestamp origin*. Accordingly, we need to discover a series of workflow instance event traces from the workflow event log histories and, from each of the discovered traces, build four species of temporal workcases and temporal worktransferences, each based on one of the timestamp origins.

[6] * indicates multiplicity.

[7] XWELL stands for XML-based workflow execution logging mechanism and language.

[8] BPAF stands for business process audit format, which is supported by the Workflow Management Coalition (WfMC).

[9] XES stands for extensible event stream, which is supported by the extensible event stream working group of the IEEE standards committee.,xes2016

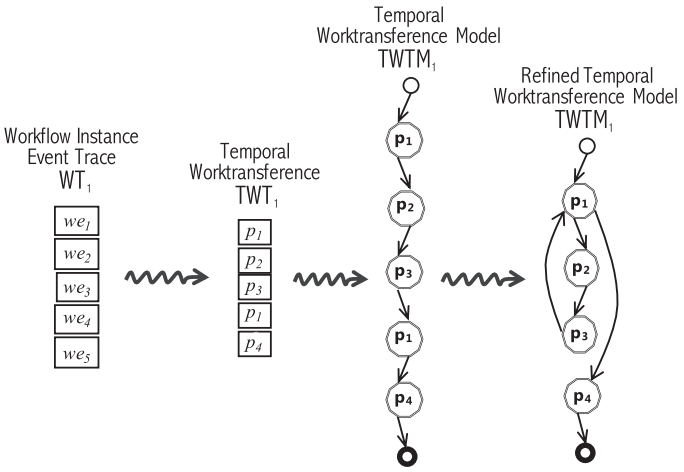

Fig. 4. Formal components for discovering temporal worktransferences.

- The Scheduled Point of Time: the event's timestamp is the time when the state of the workitem is changed from READY[10] to ASSIGNED[11]: *a workflow activity event log with scheduledTimestamp,* $we^{t.s} \Rightarrow (t = we.t \wedge s = we.s \wedge s =$ '*assigned*').
- The Assessed Point of Time: the event's timestamp is the time when the state of the workitem is changed from ASSIGNED to RESERVED[12]: *a workflow activity event log with assessedTimestamp,* $we^{t.e} \Rightarrow (t = we.t \wedge e = we.s \wedge e =$ '*reserved*').
- The Started Point of Time: the event's timestamp is the time when the state of the workitem is changed from RESERVED to RUNNING[13]: *a workflow activity event log with runningTimestamp,* $we^{t.u} \Rightarrow (t = we.t \wedge u = we.s \wedge u =$ '*running*').
- The Completed Point of Time: the event's timestamp is the time when the state of the workitem is changed from RUNNING to COMPLETED[14]: *a workflow activity event log with completedTimestamp,* $we^{t.o} \Rightarrow (t = we.t \wedge o = we.s \wedge o =$ '*completed*').

Fig. 3 illustrates four principal components, and their conceptual relationships, to be used for discovering a formal concept of the temporal workcases (defined in Definition 6) represented by a temporal workcase model (defined in Definition 7). A workflow instance event trace containing a specific workcase identifier, WT($\boldsymbol{c}$), comprises all of the workflow activity event logs that contain the same workcase identifier and are temporally ordered by one of the timestamp origins. Assume that these workflow activity event logs are extracted from a collection of workflow enactment event logs and histories. We can straightforwardly organize a temporal workcase, TWC($\boldsymbol{c}$), by removing all activity identifiers from the workflow activity event logs that comprise the corresponding workflow instance event trace. Based upon the temporal workcase TWC($\boldsymbol{c}$), we also construct its temporal workcase model, TWCM($\boldsymbol{c}$).

Definition 6. *Temporal Workcase*. Let TWC($\boldsymbol{c}$) be the workflow instance activity trace of a workcase (temporal workcase), $\boldsymbol{c}$:

- $TWC(c) = (we_{\alpha_1}^{\tau[.\phi]}, \ldots, we_{\alpha_m}^{\tau[.\phi]})$,
  where $\{we_{\alpha}^{\tau[.\phi]} \mid \alpha = we.ac \wedge \tau = we.t \wedge \phi \in \{s, e, u, o\} \wedge we_{\alpha}.wc = \mathbf{c} \wedge (we_{\alpha_i}^{\tau_i} \prec we_{\alpha_j}^{\tau_j})$[15] $\wedge\ \tau_i < \tau_j \wedge i < j \wedge 1 \le i, j \le m\}$,

which is a temporally ordered workflow activity event sequence along with the timestamp origins. In particular, a temporal workcase is formally defined as a temporal workcase model, and it is also assumed that all of the workitems in the corresponding workcase are successfully completed, and their executions are running without being suspended.

Based on Definition 6, we can interpret the formal definition as the conceptual implication that all of the workflow activity events with the same INSTANCE ID are ordered in a workflow instance activity trace, along with the timestamp origins. Consequently, from a workflow instance event trace, we produce a workflow instance activity trace by extracting

[10] The READY state of a workitem indicates that the workitem is ready to be processed but has not been assigned to any participant.
[11] The ASSIGNED state of a workitem indicates that the workitem has been assigned to a role (potentially a group of participants), but work has not started yet.
[12] The RESERVED state of a workitem indicates that the workitem has been assigned to a named user (a single participant), but work has not started yet.
[13] The RUNNING state of a workitem indicates that the workitem is actively being worked on, and time spent in this state would be recorded as processing time or work time.
[14] The COMPLETED state of a workitem indicates that the workitem has been fully executed and completed, with either success or failure.
[15] $we_{\alpha_i}^{\tau_i}$ is the predecessor of $we_{\alpha_j}^{\tau_j}$ in the list of a temporal workcase.

the activity identifiers and their timestamps, and we name it a temporal workcase, along with the timestamp origins (such as scheduled time, assessed time, started time, and completed time). For the sake of the formal representation, a temporal workcase is defined as a temporal workcase model, as formally described in Definition 7. All of the event logs in a workflow instance activity trace may have the same timestamp origin. Accordingly, there are four possible species of temporal workcase, and corresponding models, as follows:

- ScheduledTime Temporal Workcase Species and Model.
- AssessedTime Temporal Workcase Species and Model.
- StartedTime Temporal Workcase Species and Model.
- CompletedTime Temporal Workcase Species and Model.

Definition 7. *Temporal Workcase Model.* A temporal workcase model is formally defined as a 3-tuple $TWCM = (\omega, F_r^c, T_o^c)$ over a set **A** of activity trace nodes, $\forall \eta_\alpha^{\tau[.\phi]}$, of a temporal workcase, $TWC(\mathbf{c})$, of a workflow instance (workcase), **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 workcase model;
- T_o^c is an activity or an activity group linked to an external temporal workcase model;
- $\omega = \omega_i \cup \omega_o$ on $\forall \eta_\alpha^{\tau[.\phi]} \in \mathbf{A}$,
  - $\omega_o : \mathbf{A} \longrightarrow \wp(\mathbf{A})$ is a single-valued mapping function of an activity trace node, $\eta_\alpha^{\tau[.\phi]} = we_\alpha^{\tau[.\phi]} \wedge \phi \in \mathbf{K}$, to its (immediate) successor in a temporal workcase;
  - $\omega_i : \mathbf{A} \longrightarrow \wp(\mathbf{A})$ is a single-valued mapping function of an activity trace node, $\eta_\alpha^{\tau[.\phi]} = we_\alpha^{\tau[.\phi]} \wedge \phi \in \mathbf{K}$, to its (immediate) predecessor in a temporal workcase.
- The species of temporal workcase models: $TWCM^\phi$
  - ScheduledTime Temporal Workcase Model: $\phi =$'s' *in* $\forall \eta_\alpha^{\tau.\phi}$ of a temporal workcase model.
  - AssessedTime Temporal Workcase Model: $\phi =$ 'e' *in* $\forall \eta_\alpha^{\tau.\phi}$ of a temporal workcase model.
  - StartedTime Temporal Workcase Model: $\phi =$ 'u' *in* $\forall \eta_\alpha^{\tau.\phi}$ of a temporal workcase model.
  - CompletedTime Temporal Workcase Model: $\phi =$ 'o' *in* $\forall \eta_\alpha^{\tau.\phi}$ of a temporal workcase model.

Fig. 4 illustrates three principal components, and their conceptual relationships, to be used for discovering a formal concept of the temporal worktransferences (defined in Definition 8) represented by a temporal worktransference model (defined in Definition 9). A workflow instance event trace containing a specific workcase identifier, WT(***c***), comprises all of the workflow activity event logs that contain the same workcase identifier and are temporally ordered by one of the timestamp origins. Assume that these workflow activity event logs are extracted from a collection of workflow enactment event logs and histories. We can straightforwardly organize a temporal worktransference, TWT(***c***), by removing the performer identifiers from each of the workflow activity event logs making up the corresponding workflow instance event trace. The temporal worktransference is transformed to a temporal worktransference model, TWTM(***c***); in addition, we need to graphically refine the model by rearranging the duplicated performers.

Definition 8. *Temporal Worktransference.* Let TWT(***c***) be the workflow instance performer trace of a workcase (temporal worktransference), ***c***:

- $TWT(c) = (we_{p_1}^{\tau[.\phi]}, \ldots, we_{p_m}^{\tau[.\phi]})$,
  where $\{we_p^{\tau[.\phi]} \mid p = we.p \wedge \tau = we.t \wedge \phi \in \{e, u, o\} \wedge we_p^{\tau[.\phi]}.wc = \mathbf{c} \wedge (we_{p_i}^{\tau} \prec we_{p_j}^{\tau})$[16] $\wedge\ \tau_i < \tau_j \wedge i < j \wedge 1 \le i, j \le m\}$,

which is a temporally ordered workflow performer event sequence along with the timestamp origins. The ScheduledTimestamp origin is not available in the workflow performer event sequence. In particular, a temporal worktransference is formally defined by a temporal worktransference model, and it is also assumed that all of the workitems in the corresponding workcase are successfully completed, and their executions are running without being suspended.

Based on Definition 8, we can interpret the formal definition as the conceptual implication that all of the workflow activity events with the same `INSTANCE ID` are ordered in a workflow instance performer trace, along with the timestamp origins. Consequently, from a workflow instance event trace, we produce a workflow instance performer trace by extracting the performer identifiers and their timestamps, and we name it a temporal worktransference, along with the timestamp origins (such as assessed time, started time, and completed time). For the sake of the formal representation, a temporal workcase is defined as a temporal workcase model, as formally described in Definition 7. All of the event logs in a workflow instance performer trace may have the same timestamp origin. Accordingly, there are three possible species of temporal worktransference and their corresponding models, as follows:

- AssessedTime Temporal Worktransference Species and Model.
- StartedTime Temporal Worktransference Species and Model.
- CompletedTime Temporal Worktransference Species and Model.

[16] $we_{p_i}^{\tau_i}$ is the predecessor of $we_{p_j}^{\tau_j}$ in the list of a temporal worktransference.

Definition 9. *Temporal Worktransference Model.* A temporal worktransference model is formally defined as a 3-tuple $TWTM = (\chi, F_r^t, T_o^t)$ over a set **P** of performer trace nodes, $\forall \eta_p^{\tau[.\phi]}$, of a temporal worktransference, $TWT(\mathbf{c})$, of a workflow instance (workcase), **c**, and a species **K** $(= \{e, u, o\})$ of the timestamp origins, where

- F_r^t is a coordinator or a coordinator group linked from an external temporal worktransference model;
- T_o^t is a coordinator or a coordinator group linked to an external temporal worktransference model;
- $\chi = \chi_i \cup \chi_o$ on $\forall \eta_p^{\tau[.\phi]} \in \mathbf{P}$,
  - $\chi_o : \mathbf{P} \longrightarrow \wp(\mathbf{P})$ is a single-valued mapping function of a performer trace node, $\eta_p^{\tau[.\phi]} = we_p^{\tau[.\phi]} \wedge \phi \in \mathbf{K}$, to its (immediate) successor in a temporal worktransference;
  - $\chi_i : \mathbf{P} \longrightarrow \wp(\mathbf{P})$ is a single-valued mapping function of a performer trace node, $\eta_p^{\tau[.\phi]} = we_p^{\tau[.\phi]} \wedge \phi \in \mathbf{K}$, to its (immediate) predecessor in a temporal worktransference.
- The species of temporal worktransference models: $TWTM^{\phi}$
  - AssessedTime Temporal worktransference Model: $\phi =$ '*e*' *in* $\forall \eta_p^{\tau.\phi}$ of a temporal worktransference model.
  - StartedTime Temporal worktransference Model: $\phi =$ '*u*' *in* $\forall \eta_p^{\tau.\phi}$ of a temporal worktransference model.
  - CompletedTime Temporal worktransference Model: $\phi =$ '*o*' *in* $\forall \eta_p^{\tau.\phi}$ of a temporal worktransference model.

To conclude this subsection, we know that so-called *workflow warehouses*, based on temporal workcases and temporal worktransferences, are needed to build a workflow mining system for discovering the work transference networking knowledge from the workflow enactment event logs. A workflow warehouse should be shaped as a cube with three axes: workflow procedures (*workflow models*), workflow instances (*temporal workcases* and *temporal worktransferences*), and workflow workitems (*workflow activity event logs*). To construct a workflow warehouse, we have to perform a preprocessing phase that extracts a series of temporal workcases with temporal worktransferences from a collection of workflow enactment event logs, and forms two cubes: one for temporal workcases and one for temporal worktransferences. These two cubes eventually become the input to a workflow work transference network discovery algorithm, which will be developed in the next section.

4.2. A formal discovery algorithm

Before developing a formal discovery algorithm, we start by formally defining a runtime model of a work transference network that is formed by fulfillment of a workflow procedure, as described in Definition 10. This runtime model reveals the real form of the worktransference relationships between the performers during the time period that has elapsed since deploying the corresponding workflow model. It is discovered from the workflow enactment event history in a workflow warehouse, such as the temporal workcase warehouse and temporal worktransference warehouse presented in the previous subsection. We strongly emphasize that analyzing and comparing the two models of work transferences, discovered from a buildtime model and a runtime history, are among the essential activities in human resource management and decision-making support management [8]. Consequently, the differentiation between these two types (planned and enacted) of work transference relationships in performing a workflow procedure should provide essential knowledge for evaluating and validating the human resource performance of the corresponding workflow procedure.

Definition 10. *(Runtime) Work Transference Network Model.* A runtime work transference network model is formally defined as $\Lambda^R = (\sigma, \psi, F_r^R, T_o^R)$, over a set **P** of performer trace nodes, $\eta_p^{\tau[.\phi]}$, and a set **A** of activity trace nodes, $\eta_\alpha^{\tau[.\phi]}$, in a collection of workflow instance event traces logged from enacting a specific workflow model, where

- F_r^R is a coordinator, or coordinator group, linked from some external runtime work transference networks;
- T_o^R is a coordinator, or coordinator group, linked to some external runtime work transference networks;
- $\sigma = \sigma_i \cup \sigma_o$ /* Work (Workitem) Transferences */
  - $\sigma_o : \mathbf{P} \longrightarrow \wp(\mathbf{P})$ is a multi-valued function mapping a performer to its set of (immediate) work (workitem) transferrers;
  - $\sigma_i : \mathbf{P} \longrightarrow \wp(\mathbf{P})$ is a multi-valued function mapping a performer to its set of (immediate) work (workitem) receivers;
- $\psi = \psi_i \cup \psi_o$ /* Work (Workitem) Associations */
  - $\psi_i : (\mathbf{P} \times \mathbf{P}) \longrightarrow \wp(\mathbf{A})$ is a multi-valued function returning a **bag**[17] of receiving workitems on ordered pairs of performers, $(\sigma_i(p), p)$, $p \in \mathbf{P}$, from $\sigma_i(p)$ to p;
  - $\psi_o : (\mathbf{P} \times \mathbf{P}) \longrightarrow \wp(\mathbf{A})$ is a multi-valued function returning a **bag** of transferring workitems on ordered pairs of performers, $(p, \sigma_o(p))$, $p \in \mathbf{P}$, from p to $\sigma_o(p)$.

Based upon the runtime work transference network model, we develop an algorithm that is able to discover a work transference network from a workflow instance event trace of a single workflow instance, which is preprocessed to form both a temporal workcase and a temporal worktransference. The core idea of the algorithm is to amalgamate the two temporal models, of a workcase and a worktransference, to discover a work transference network. The algorithm discovers the workitem association knowledge from the temporal workcase model, and discovers the workitem transference knowledge from the temporal worktransference model. Algorithm 2 takes a temporal workcase model and a temporal worktransference

[17] A bag (or a multiset) is a generalization of the concept of set that allows multiple instances of elements.

Algorithm 2 RUNTIME WTN DISCOVERY ALGORITHM: An Algorithm for Discovering a Work Transference Network Model from a Workflow Instance Event Trace Based upon AssessedTime ($\phi =' e'$).

Require: (1) A Temporal Workcase Model, $TWCM^e = (\omega, F_r^c, T_o^c)$, over all activity trace nodes, $\forall \eta_\alpha^{\tau.e} \in \mathbf{A}$;(2) A Temporal Worktransference Model, $TWTM^e = (\chi, F_r^t, T_o^t)$, over all performer trace nodes, $\forall \eta_p^{\tau.e} \in \mathbf{P}$;

Ensure: (1) A Runtime Work Transference Network Model, $\Lambda^R = (\sigma, \psi, F_r^R, T_o^R)$;

procedure MAIN ▷ Mining a work transference network model
  for $\forall p \in \mathbf{P} \wedge \forall \alpha \in \mathbf{A}$ **do** ▷ $p = \eta_p^{\tau.e}$; $\alpha = \eta_\alpha^{\tau.e}$
    ▷ $\sigma = \sigma_i \cup \sigma_o$: Discovering the work (workitem) transferences
    $\sigma_i(p) \leftarrow \sigma_i(p) + \chi_i(p)$; ▷ $p = \eta_p^{\tau.e}$
    $\sigma_o(p) \leftarrow \sigma_o(p) + \chi_o(p)$; ▷ $p = \eta_p^{\tau.e}$
    ▷ $\psi = \psi_i \cup \psi_o$: Discovering the work (workitem) associations
    if $\alpha.t = p.t \wedge \alpha.p = p \wedge \alpha.\alpha = p.\alpha$ **then**
      $\psi_i((\sigma_i(p), p)) \leftarrow \psi_i((\sigma_i(p), p)) + \alpha$; ▷ $\sigma_i(p) = \eta_{\sigma_i(p)}^{\tau.e}$
    end if
    if $\omega_o(\alpha).t = \sigma_o(p).t \wedge \omega_o(\alpha).p = \sigma_o(p) \wedge \omega_o(\alpha).\alpha = \sigma_o(p).\alpha$ **then**
      $\psi_o((p, \sigma_o(p))) \leftarrow \psi_o((p, \sigma_o(p))) + \omega_o(\alpha)$; ▷ $\sigma_o(p) = \eta_{\sigma_o(p)}^{\tau.e}$
    end if
  end for
end procedure

model as input and produces a single runtime work transference network model as output. In particular, the algorithm assumes that the temporal workcase model ($TWCM^e$) and the temporal worktransference model ($TWTM^e$) contain the same number of workitems and the same origin for their logging timestamps (such as the AssessedTime timestamp origin). The algorithm also exploits the fact that the temporal workcase model and the temporal worktransference model are built from a single workflow instance event trace and the fact that they can possibly be formed with the same timestamp origin. The overall time complexity of the algorithm is $O(|A| + |P|)$ as its time complexity, and so it becomes $O(N)$, where N is the number of workflow workitem event logs in a workflow instance event trace of a workflow instance.

The semantical structure of the algorithm is composed of two logical routines. One is for discovering the work transference knowledge, which is formally represented as $\sigma = \sigma_i \cup \sigma_o$, from a temporal worktransference model. The other is for discovering the work association knowledge, which is formally represented as $\psi = \psi_i \cup \psi_o$, from a temporal workcase model. The former is a straightforward and simple logical routine, because a temporal worktransference model is a partial subset of the work transference network model that is an eventual outcome of the discovery approach proposed in this paper. The latter is a slightly more complex logical routine, because the syntactical structure of a temporal workcase model is completely different from that of the work transference network model. Accordingly, the algorithm reasonably uses the internal properties of each workitem's event log, which is defined in Definition 4, to discover an associated workitem with a corresponding work transference edge (an ordered pair of transferring performer node and receiving performer node). The following are the crucial parts of the latter routine to discover the work association knowledge. They comprise a mapped association of a predecessor of a performer (p) with a workitem (α) and an incoming edge ($(\sigma_i(p), p)$), and another mapped association of a successor of the performer with a workitem ($\omega_o(\alpha)$) and an outgoing edge ($(p, \sigma_o(p))$), in a corresponding work transference network:

- α *iff* $[\alpha.t = p.t \wedge \alpha.p = p \wedge \alpha.\alpha = p.\alpha]$
- ω_o (α) *iff* $[\omega_o(\alpha).t = \sigma_o(p).t \wedge \omega_o(\alpha).p = \sigma_o(p) \wedge \omega_o(\alpha).\alpha = \sigma_o(p).\alpha]$

Fig. 5 illustrates some operational examples of the algorithm. We apply the algorithm to the workcase event traces of four different workflow instances that can be instantiated from the library book acquisition workflow model introduced in the previous section. Each workcase event trace is preprocessed and transformed to a temporal workcase (e.g., $\mathsf{ACC'UU'IP[GD]^2TZ}$) and a temporal worktransference (e.g., $p_1p_1p_3p_2p_3p_4p_4[p_{10}p_4]^2p_1p_1$). The algorithm is gradually discovering a work transference network by being applied to the workitem's event logs for these two temporal models, incrementally. It finally constructed a work transference network model, which is depicted as a directed graph model in the figure. In particular, in listing the workitems in a temporal workcase, we use brackets with a superscripted number (e.g., $[\mathsf{GD}]^2$) to emphasize the concept of a bag (not a set), to formally represent the multiplicity of associated workitems of a single worktransference edge.

Fig. 6 depicts a conceptual principle of the discovery approach proposed in this paper, which is the ultimate goal of the research and development of the authors' research group. It shows a stepwise procedure for the discovery of a work transference network, which individually amalgamates the work transference networks of four different workcases, which are discovered by applying the algorithm to each of them, as depicted in Fig. 5. Through the illustration of the conceptual discovery approach, we can reasonably imagine how to design and develop its formal algorithm. In this paper, we omit the details of the conceptual principle and its implementation, because of page limitations.

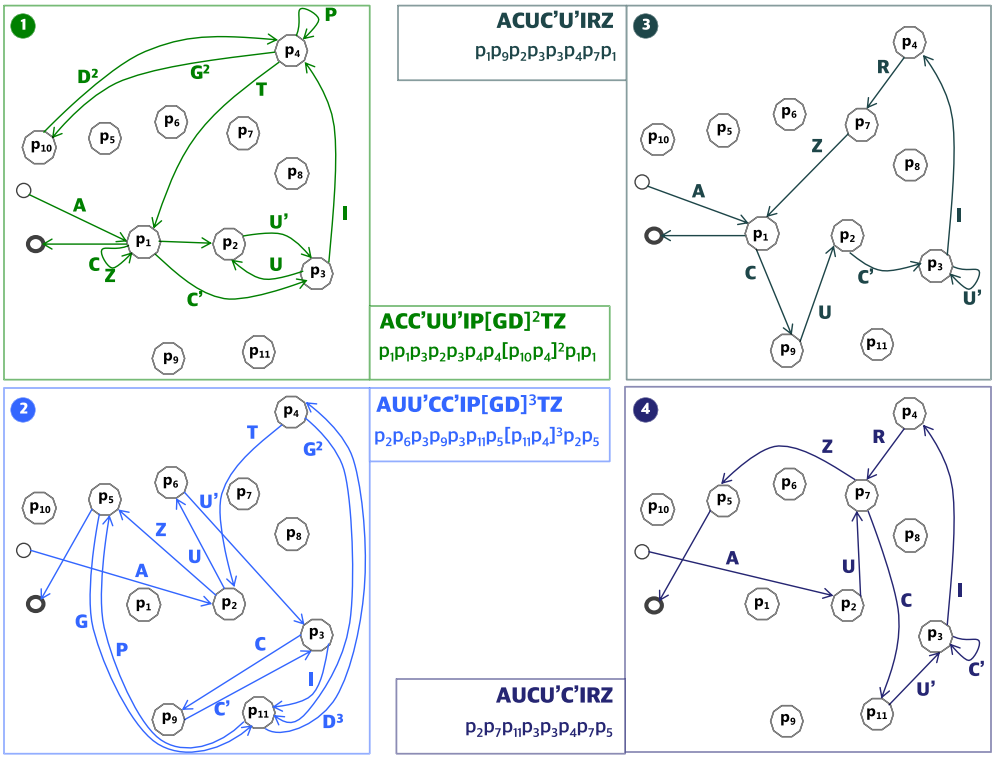

Fig. 5. Runtime work transference networks discovered from four traces of the LBA workflow model.

4.3. Handling loops in workflow instance event traces

In this subsection, we extend the conceptual formalisms defined in the previous subsection to properly handle the event logs of iterative activities (*e.g.*, G and D in the library book acquisition workflow model of Fig. 1) in a workflow instance event trace that is spawned from a workcase instantiated from a workflow procedure with loop constructs. In particular, the conceptual names of the repeated event traces and their repeated work transferences are called temporal loopcases and temporal looptransferences, respectively. Their formal representations are defined by the temporal loopcase model and temporal looptransference model. We propose a series of formal definitions reflecting the concepts of temporal loopcases and temporal looptransferences. Before proceeding to formulate the loop discovery formalisms further, we need to characterize the loop constructs, which we assume to be used in modeling such iterative activities in an ICN of a workflow model, as follows:

- A simple loop construct is constructed from a gateway-type activity of an OR-split node (loop termination) that has one of its two outgoing edges pointing back to another gateway-type activity of an OR-join node (loop initiation) that has two incoming edges and one outgoing edge.
- A compound loop construct is constructed from several loop constructs, each of which is formed by a pair of loop-initiation and loop-termination gateway-type nodes, and its inner loop constructs are properly nested inside its outer loop constructs.
- A workflow model may be constructed from several loop constructs, which are sequentially ordered.

In general, all loop bodies of the outer loop constructs, and their inner loop constructs, can be constructed from combinations of sequential, disjunctive, conjunctive, and iterative transition activities, to define an ICN of a workflow procedure. We emphasize that the structures of loop bodies can be complex and that correctly handling these complex loop bodies is a very challenging research topic in the workflow discovery and mining field. Therefore, in this paper, we attempt to formalize only a simplified type of loop body: a sequential transition structure of loop bodies, and a single-level (non-nested) repetition of outer loop constructs. We leave the most challenging issues to future work. This will be able to properly handle the complex structure of loop bodies and the nested repetition of outer loop and inner loop constructs, in developing workflow discovery and mining formalisms and algorithms.

Fig. 7 illustrates two types of conceptual deployments in handling a looping portion in a workflow instance event trace, logged from three iterative enactments of a single-level outer loop construct that contains a sequential transition loop body with three sequential activities, α_4, α_6, and α_9. On the left-hand side, the figure shows a conceptual deployment for formal-

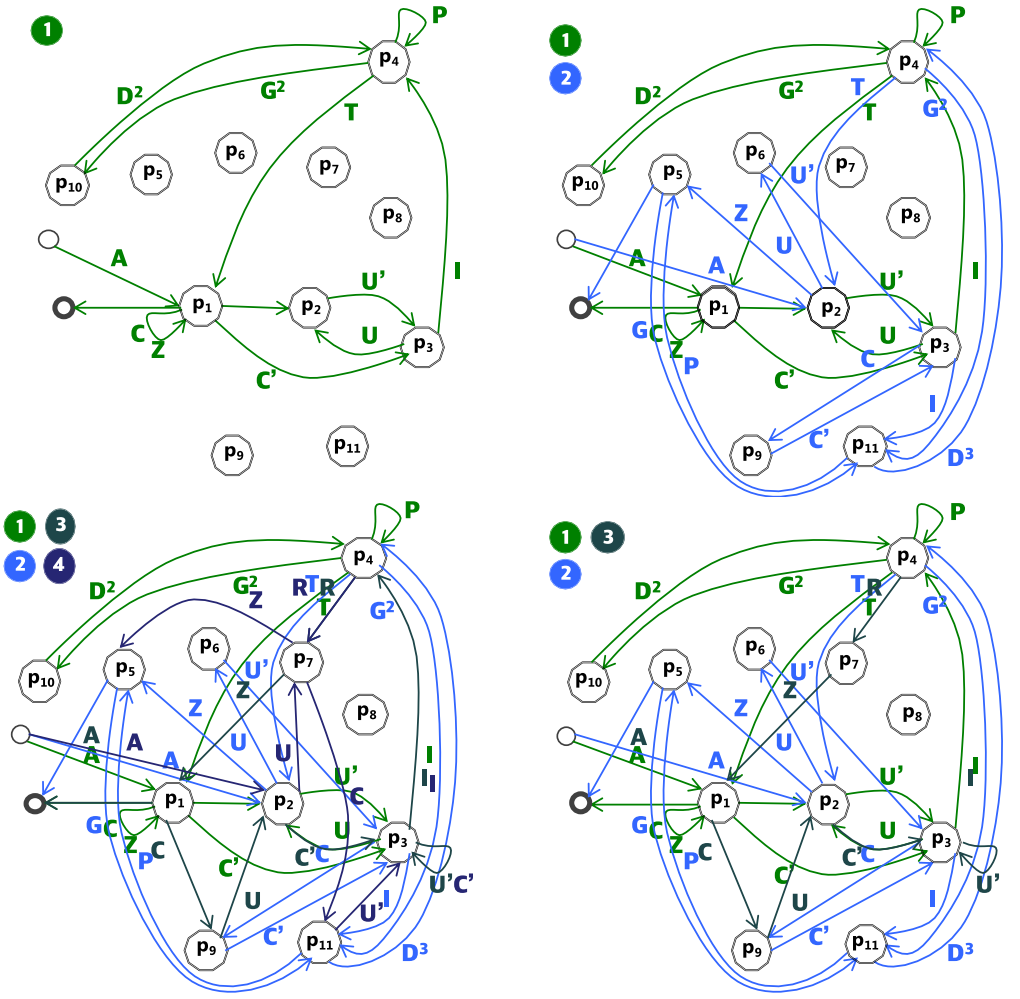

Fig. 6. A Conceptual principle of discovering a work transference network by amalgamating trace logs incrementally.

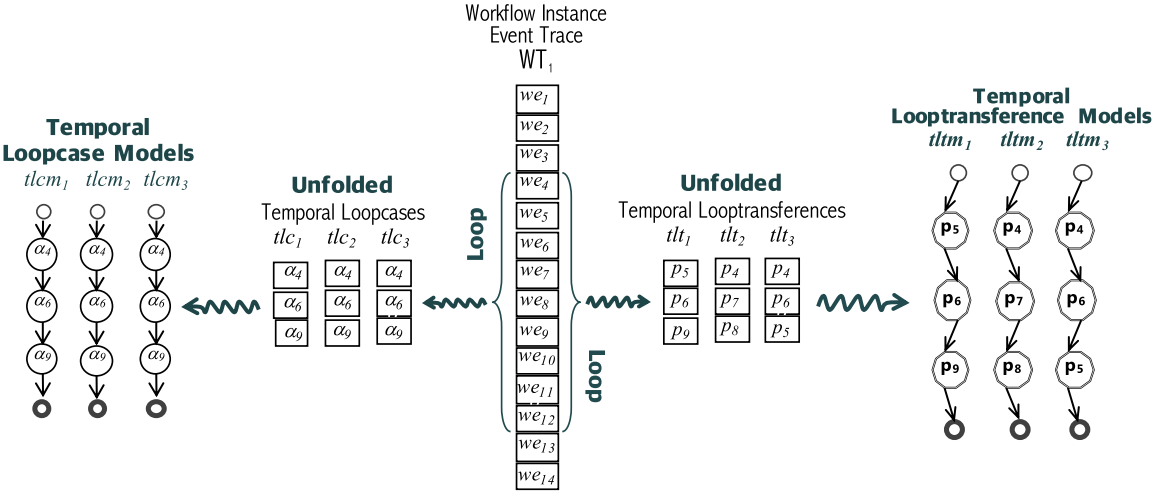

Fig. 7. Formal components of unfolded temporal loopcases and unfolded temporal looptransferences from a workflow instance event trace of a single-level (non-nested) outer loop construct.

izing the three temporal loopcases and their models, each of which is extracted by unfolding each iteration out of the loop portion ($we_4.ac$–$we_{12}.ac$) in a corresponding workflow instance event trace that is a sequence of workflow activity event logs, such as we_1–we_{14}, ordered by a timestamp origin. The right-hand side provides a conceptual deployment for formalizing three temporal looptransferences and their models, each of which corresponds to a temporal loopcase. By taking each performer identifier from the event logs of the temporal loopcases ($we_4.p$–$we_{12}.p$), we obtain three temporal looptransferences and their models. Through Definitions 11–14, we formalize components such as temporal loopcase, temporal loopcase model, temporal looptransference, and temporal looptransference model, for reasonably handling iterative workflow instance event traces logged from enacting sequential transition loop body constructs in a workflow model.

Definition 11. *Temporal Loopcases*. Let $TLC^{lc}(\boldsymbol{c})$ be a series of loopcases (temporal loopcases) of a loop construct identifier, lc, in a workflow instance event trace, $\boldsymbol{c}$:

$$
\begin{aligned}
TLC^{lc}(c) = \{ & \\
& we^{\tau[.\phi]}_{1,\alpha_1}, \ldots, we^{\tau[.\phi]}_{1,\alpha_m}\}, \\
& \{we^{\tau[.\phi]}_{2,\alpha_1}, \ldots, we^{\tau[.\phi]}_{2,\alpha_m}\}, \\
& \ldots, \\
& \{we^{\tau[.\phi]}_{k,\alpha_1}, \ldots, we^{\tau[.\phi]}_{k,\alpha_m}\} \\
& \},
\end{aligned}
$$

where

- k is the loopcase identifier, which depends on the number of repetitive enactments of a non-nested loop construct (with identifier lc) that contains sequential transition activities, $\{\alpha_1, \ldots, \alpha_m\}$;
- $\{we^{\tau[.\phi]}_{u,\alpha} \mid \alpha = we.ac \wedge \tau = we.t \wedge \phi \in \{s, e, u, o\} \wedge we_\alpha.wc = \mathbf{c} \wedge (we^{\tau_i}_{u,\alpha_i} \prec we^{\tau_j}_{u,\alpha_j})$[18] $\wedge\ \tau_i < \tau_j \wedge i < j \wedge 1 \le i, j \le m \wedge 1 \le u \le k\}$.

Each of the serialized loopcases is a temporally ordered workflow activity event sequence, along with the timestamp origins, for each iteration of a sequential transition loop body. In particular, a temporal loopcase is formally defined by a temporal loopcase model, and it is assumed that all of the workitems in the corresponding loopcase are successfully completed, and their executions are running without being suspended.

Based on Definition 11, we can interpret the formal definition as the conceptual implication that we can discover a series of *iterated loop instance event traces* from a workflow instance event trace. We name each of them a temporal loopcase, along with the timestamp origins (such as the scheduled time, assessed time, started time, and completed time). For the sake of the formal representation, a temporal loopcase is defined by a temporal loopcase model, as formally described in Definition 12. Note that all of the event logs in a loop instance event trace, originated from a loop construct with an identifier (lc), have to contain the same timestamp origin. Accordingly, there exist four species of the temporal loopcase, and their models, as follows:

- ScheduledTime Temporal Loopcase Species and Model.
- AssessedTime Temporal Loopcase Species and Model.
- StartedTime Temporal Loopcase Species and Model.
- CompletedTime Temporal Loopcase Species and Model.

Definition 12. *Temporal Loopcase Model*. A temporal loopcase model is formally defined as a 5-tuple $TLCM^{lc}_k = (\varpi^{lc}, F^{lc}_r, T^{lc}_o, I, O)$ over a set $\mathbf{A}$ of loop-activity event trace nodes, $\forall \eta^{\tau[.\phi]}_{k,\alpha}$, on a temporal loopcase, $TLC^{lc}_k(\mathbf{c})$, of a single workflow loop iteration (loopcase identifier), $\mathbf{k}$, a loop construct identifier, $\mathbf{lc}$, and a workcase identifier, $\mathbf{c}$, with a species $\mathbf{K}$ $(= \{s, e, u, o\})$ of timestamp origin, where

- F^{lc}_r is an activity or activity group linked from its loop construct in a corresponding temporal workcase model;
- T^{lc}_o is an activity or activity-group linked to its loop construct in a corresponding temporal workcase model;
- I is a set of input repositories coming from its loop construct in a corresponding temporal workcase model;
- O is a set of output repositories going to its loop construct in a corresponding temporal workcase model;
- $\varpi^{lc}_k = \varpi^{lc}_{k,i} \cup \varpi^{lc}_{k,o}$ on $\forall \eta^{\tau[.\phi]}_{k,\alpha} \in \mathbf{A}$,
  - $\varpi^{lc}_{k,o} : \mathbf{A} \longrightarrow \wp(\mathbf{A})$ is a single-valued mapping function of a loop-activity event trace node, $\eta^{\tau[.\phi]}_{k,\alpha} = we^{\tau[.\phi]}_{\alpha} \wedge \phi \in \mathbf{K}$, to its (immediate) successor in a temporal loopcase with identifier, k, and loop construct identifier, lc;
  - $\varpi^{lc}_{k,i} : \mathbf{A} \longrightarrow \wp(\mathbf{A})$ is a single-valued mapping function of a loop-activity trace node, $\eta^{\tau[.\phi]}_{k,\alpha} = we^{\tau[.\phi]}_{k,\alpha} \wedge \phi \in \mathbf{K}$, to its (immediate) predecessor in a temporal loopcase with identifier, k, and loop construct identifier, lc.

[18] $we^{\tau_i}_{u,\alpha_i}$ is the predecessor of $we^{\tau_j}_{u,\alpha_j}$ in the list of a temporal loopcase.

- The species of temporal loopcase models: $TLCM_k^{lc.\phi}$
  - ScheduledTime Temporal Loopcase Model: ϕ = '*s*' *in* $\forall\eta_{k,\alpha}^{\tau.\phi}$ of a temporal loopcase model.
  - AssessedTime Temporal Loopcase Model: ϕ = '*e*' *in* $\forall\eta_{k,\alpha}^{\tau.\phi}$ of a temporal loopcase model.
  - StartedTime Temporal Loopcase Model: ϕ = '*u*' *in* $\forall\eta_{k,\alpha}^{\tau.\phi}$ of a temporal loopcase model.
  - CompletedTime Temporal Loopcase Model: ϕ = '*o*' *in* $\forall\eta_{k,\alpha}^{\tau.\phi}$ of a temporal loopcase model.

Definition 13. *Temporal Looptransferences.* Let $TLT^{lc}(\mathbf{c})$ be the repeated workflow instance performer traces of a loop construct identifer, *lc*, on a workcase identifier, $\mathbf{c}$:

$$TLT^{lc}(c) = \{ \{we_{1,p_1}^{\tau[.\phi]}, \ldots, we_{1,p_m}^{\tau[.\phi]}\}, \{we_{2,p_1}^{\tau[.\phi]}, \ldots, we_{2,p_m}^{\tau[.\phi]}\}, \ldots, \{we_{k,p_1}^{\tau[.\phi]}, \ldots, we_{k,p_m}^{\tau[.\phi]}\} \},$$

where

- k is the looptransference identifier, which depends on the number of repetitive enactments of a non-nested loop construct (lc) that contains sequential transition performers, $\{p_1, \ldots, p_m\}$;
- $\{we_{u,p}^{\tau[.\phi]} \mid p = we.p \wedge \tau = we.t \wedge \phi \in \{e, u, o\} \wedge we_p^{\tau[.\phi]}.wc = \mathbf{c} \wedge (we_{u,p_i}^{\tau} \prec we_{u,p_j}^{\tau})$[19] $\wedge\ \tau_i < \tau_j \wedge i < j \wedge 1 \le i, j \le m \wedge 1 \le u \le k\}$.

This is a temporally ordered workflow performer event sequence, along with the timestamp origins. The Scheduled-Timestamp origin is not available in the workflow performer event sequence. In particular, a temporal worktransference is formally defined by a temporal worktransference model, and it is assumed that all of the workitems in the corresponding workcase are successfully completed, and their executions are running without being suspended.

Based on Definition 13, we can interpret the formal definition as the conceptual implication that all of the workflow activity events with the same `INSTANCE ID` are ordered in a workflow instance performer trace, along with the timestamp origins. Consequently, we produce a series of iterated workflow performer event traces by extracting the performer identifiers and their timestamps from a workflow instance event trace and its iterated temporal loopcases. We name each of the workflow performer event traces a temporal worktransference, along with the timestamp origins (such as the assessed time, started time, and completed time). For the sake of the formal representation, a temporal looptransference is defined as a temporal looptransference model, as formally described in Definition 14. All of the event logs in a workflow performer event trace may have the same timestamp origin. Accordingly, there exist three species of temporal looptransferences, and their models, as follows:

- AssessedTime Temporal Looptransference Species and Model.
- StartedTime Temporal Looptransference Species and Model.
- CompletedTime Temporal Looptransference Species and Model.

Definition 14. *Temporal Looptransference Model.* A temporal looptransference model is formally defined as a 3-tuple $TLTM_k^{lc} = (\theta_k^{lc}, F_r^{lc}, T_o^{lc})$ over a set $\mathbf{P}$ of loop performer event trace nodes, $\forall\eta_{k,p}^{\tau[.\phi]}$, on a temporal looptransference, $TLT_k^{lc}(\mathbf{c})$, of a loop construct identifier, *lc*, on a workflow instance (workcase), $\mathbf{c}$, with a species $\mathbf{K}$ ($= \{e, u, o\}$) of the timestamp origins, where

- F_r^{lc} is a performer or performer group linked from its loop construct in a corresponding temporal worktransference model;
- T_o^{lc} is a performer or performer group linked to its loop construct in a corresponding temporal worktransference model;
- $\theta_k^{lc} = \theta_{k,i}^{lc} \cup \theta_{k,o}^{lc}$ on $\forall\eta_{k,p}^{\tau[.\phi]} \in \mathbf{P}$,
  - $\theta_{k,o}^{lc} : \mathbf{P} \longrightarrow \wp(\mathbf{P})$ is a single-valued mapping function of a performer trace node, $\eta_{k,p}^{\tau[.\phi]} = we_{k,p}^{\tau[.\phi]} \wedge \phi \in \mathbf{K}$, to its (immediate) successor in a temporal looptransference with identifier, *k*, and loop construct identifer, *lc*;
  - $\theta_{k,i}^{lc} : \mathbf{P} \longrightarrow \wp(\mathbf{P})$ is a single-valued mapping function of a performer trace node, $\eta_{k,p}^{\tau[.\phi]} = we_{k,p}^{\tau[.\phi]} \wedge \phi \in \mathbf{K}$, to its (immediate) predecessor in a temporal looptransference with identifier, *k*, and loop construct identifer, *lc*.
- The species of temporal looptransference models: $TLTM_k^{lc.\phi}$
  - AssessedTime Temporal looptransference Model: ϕ = '*e*' *in* $\forall\eta_{k,p}^{\tau.\phi}$ of a temporal looptransference model.
  - StartedTime Temporal looptransference Model: ϕ = '*u*' *in* $\forall\eta_{k,p}^{\tau.\phi}$ of a temporal looptransference model.

[19] $we_{u,p_i}^{\tau_i}$ is the predecessor of $we_{u,p_j}^{\tau_j}$ in the list of a temporal looptransference.

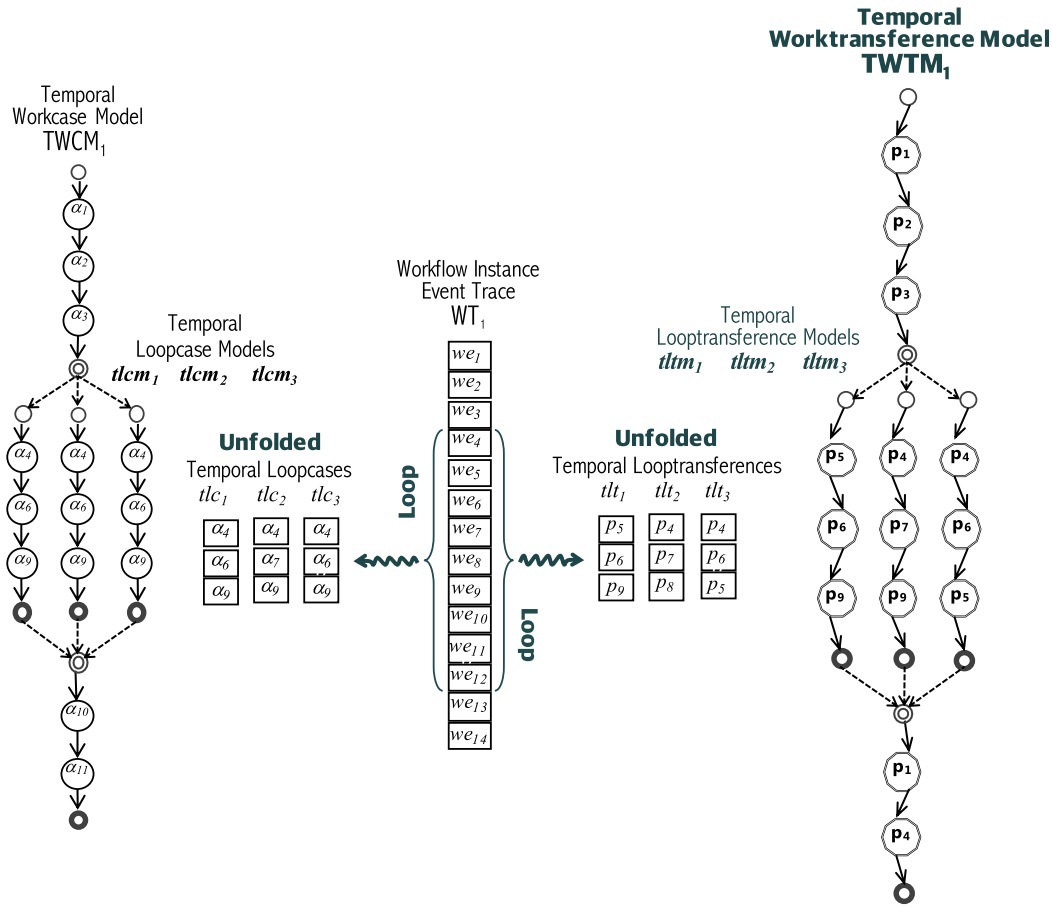

Fig. 8. A Complete temporal workcase model and its temporal worktransference model with unfolded loopcases and unfolded looptransferences.

- CompletedTime Temporal looptransference Model: ϕ = 'o' *in* $\forall \eta_{k,p}^{\tau.\phi}$ of a temporal looptransference model.

In summary, through Fig. 8 we finalize the matter of the loop handling in discovering a work transference network from a workflow instance event trace. The temporal loopcase models of a loop construct are merged into its corresponding temporal workcase model, and the temporal looptransference models of the loop construct are merged into its corresponding temporal worktransference model. We emphasize that the temporal workcase model and worktransference model (containing loopcases and looptransferences) can be represented by their formal definitions with no modifications. So far, we have assumed that the structure of loop bodies is a simple formation of sequential transitions and that each loop body is a non-nested loop construct. As stated previously, this paper will not consider the handling of more complex loop bodies, such as conjunctive and disjunctive transition structures with nested loop bodies.

4.4. A formal discovery framework

Based upon those formal components (such as temporal workcases, temporal loopcases, temporal worktransferences, and temporal looptransferences) and their formal models, we complete this formal discovery approach with a formal discovery framework, illustrated in Fig. 9. The framework performs the whole task of discovery, from archiving the workflow enactment event histories as logs of big data in a workflow-supported organization. To build the logs of big data, a workflow enactment event logging mechanism [14,15] collects event logs through the following three types of functional components:

- Event triggering component: Requester and worklist handler.
- Event formatting component: Workcase pool.
- Event logging component: Log agent and big data storage.

The event triggering component fulfills the role of workflow enactment services being requested from the workflow runtime clients. The services are classified into three classes—workflow-level, workcase-level, and workitem-level. The event formatting component composes workflow activity event logs, according to the service class, after performing the requested

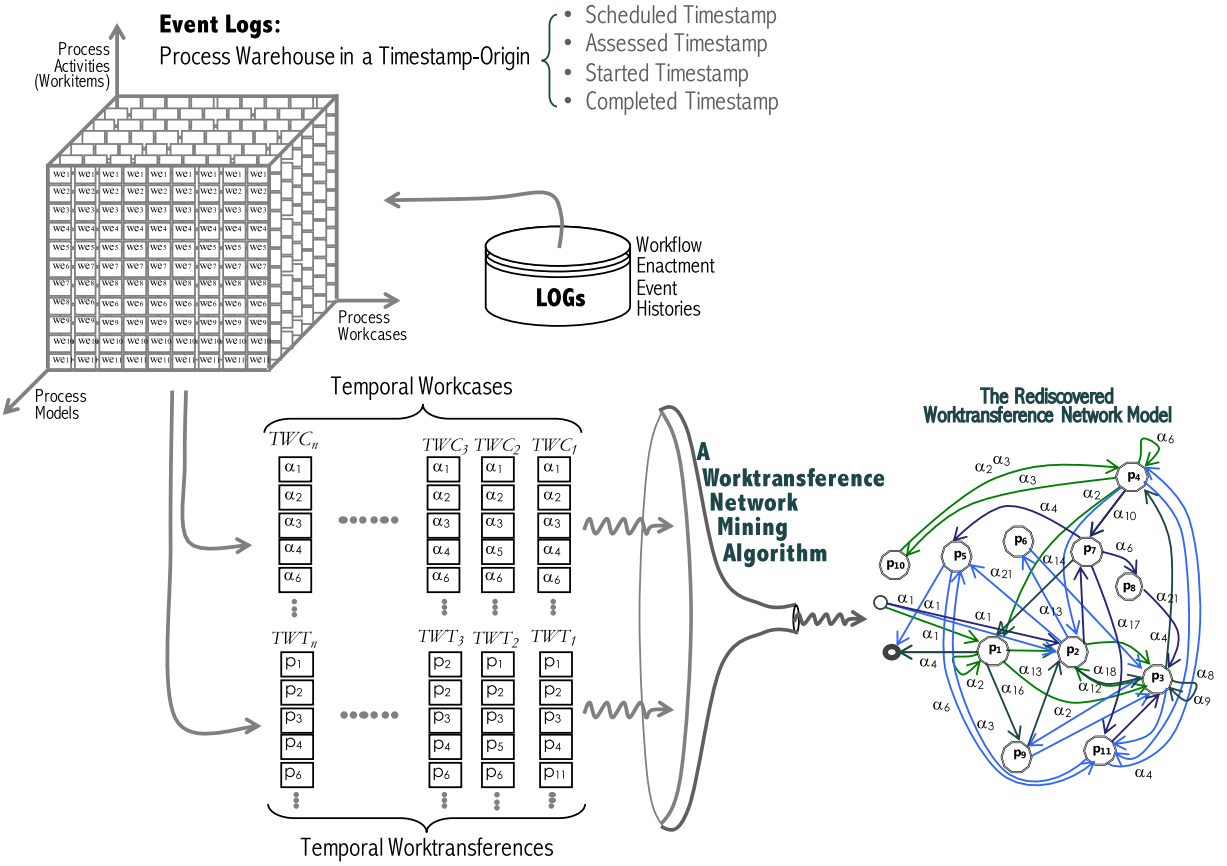

Fig. 9. A formal framework for discovering work transference networks.

services. Finally, the event logging component, specifically the log agents, is responsible for the workflow enactment event logging mechanism. When the log agent receives event logs, it transforms them to a standardized log format in an XML-based event log language, such as XES [13], BPAF [12], or XWELL [11].

Definition 15. Workflow Warehouse (or workitem event logs). Let $I = \{c_1, \ldots, c_m\}$ be a group of workflow instances that are instantiated from a workflow model, where **c** is the workflow instance identifier, and **m** is the number of workflow instances.

- $WL^{\tau[.\phi]}(\mathbf{w})$ is a workflow enactment event log, which comprises a group of workflow instance event traces with a species of the timestamp origin of a corresponding workflow model, **w**, where $WL^{\tau[.\phi]}(\mathbf{w}) = \{\mathrm{WT}(c_1), \ldots, \mathrm{WT}(c_m)\}$;
- $WW^{\tau[.\phi]}(\mathbf{pc})$ is a workflow warehouse of a workflow package, **pc**, with a species of the timestamp origins, where $WW^{\tau[.\phi]}(\mathbf{pc}) = \{WL^{\tau[.\phi]}(w_1), \ldots, WL^{\tau[.\phi]}(w_n)\}$;
  - **pc** is a workflow package identifier;
  - $\tau[.\phi]$ is a species of timestamp origin, $\phi = \{\mathrm{s}, \mathrm{e}, \mathrm{u}, \mathrm{o}\}$, with the timestamp property, τ, of the workflow activity event log format;
  - n is the number of workflow models belonging to a corresponding workflow package, **pc**;
- *WW* is a set of all of the workflow warehouses of a workflow-supported organization with the timestamp origins, where $\mathrm{WW} = \{WW^{\tau[.\phi]}(\boldsymbol{pc_1}), \ldots, WW^{\tau[.\phi]}(\boldsymbol{pc_a})\}$ with $\phi = \{s, e, u, o\}$, and **a**, which is the number of workflow packages in the corresponding workflow-supported organization.

From a collection of workflow enactment event histories stored in big data log storage (LOGs), we can construct a workflow warehouse as a cube, with its three axes being the core workflow components, ordered by one of the timestamp origins. That is, the formal definition of a workflow warehouse is given in Definition 15. It is formed as a conceptual cube with three dimensions: workflow procedures, workflow workcases, and workflow activities' workitems. From the cube (workitem event logs), we extract a stream of temporal workcases (including temporal loopcases) and another stream of temporal worktransferences (including temporal looptransferences) for a specific workflow model. These temporal workcases and temporal worktransferences become the input to a work transference network mining algorithm, such as the

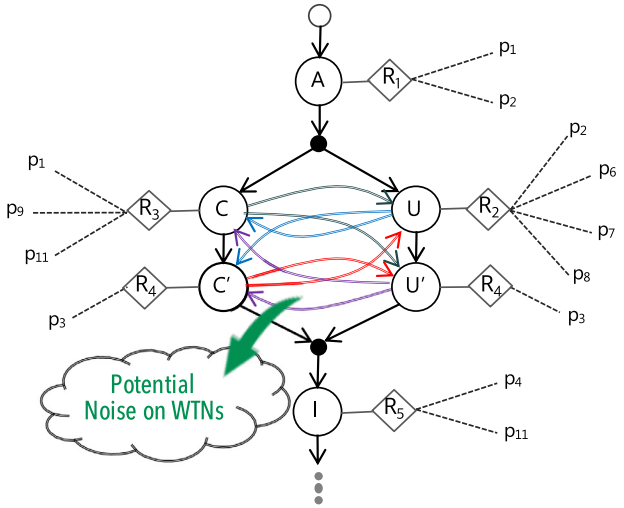

Fig. 10. Potential event log noise from the bodies of AND-gateway process patterns.

discovery algorithm introduced in Algorithm 2, which is able to discover a work transference network for the corresponding workflow model.

4.5. Deliberate noise in rediscovered work transference networks

The work transference network model rediscovered by the formal framework of Fig. 9 might not be a complete and exact model, unfortunately, because of the conceptual models of temporal workcases and temporal worktransferences. Recall that, in these conceptual models, all ordered pairs of performers and of activities form corresponding directed edges, and these directed edges are used to construct a work transference network model. We found that some of the directed edges did not originate from the original workflow process model. Therefore, in this subsection, we need to conceive an approach to handle these nonexistent directed edges when discovering a work transference network model from the large dataset of workflow process event log histories.

Fig. 10 illustrates a possible situation in which the formation of nonexistent directed edges stems from the conjunctive-AND process patterns in the original workflow process model, and these will become potential noise edges in the corresponding work transference network model. This type of nonexistent directed edge must inevitably be formed, because of the nature of our work transference network discovery algorithms, and so we name those nonexistent directed edges *deliberate noise*. We need to remove this deliberate noise from the outcome of the discovery algorithm. We describe how the internal structure of temporal workcases and temporal worktransferences inevitably lead to the creation of deliberate noise, and we simply devise an algorithm for removing all of the deliberate noise from the rediscovered work transference networks.

Fig. 11 illustrates the formation of deliberate noise in a work transference network model. The deliberate noise edge from a transferring performer, P_3, to a receiving performer, P_2, and the value of its associated activity, U, are created from both a temporal workcase model and a temporal worktransference model formed from the enactment event log trace of an example workflow instance. Although there are two temporally ordered activities, $\alpha_{C'}$ and α_U, in the specific temporal workcase, no edge between them exists in the original workflow process model. These abnormal event logs of two activities can be generated during the enactment of a workflow process instance that includes parallel-AND process patterns in its original workflow process model.

As an additional step of the work transference network discovery framework, we need to discover and exclude all such deliberate noise. We devise Algorithm 3 for this purpose. The algorithm requires two models as the input arguments: one is an original ICN model and the other is the incomplete rediscovered model of the work transference network. As the output argument, the algorithm constructs the complete model of the work transference network by eliminating all of the deliberate noise. The main body of the algorithm consists of two essential routines for eventually constructing the complete work transference network model: one is to place the directed edges of work transferences between performers and the other is to place the transferred activities of work associations on the corresponding directed edges. Fig. 12 illustrates the results from the four workflow instance enactment event log traces of the example LBA workflow model, as the results of the algorithm. The overall time complexity of the algorithm is $O(N^2)$, where N is the number of performers in the incomplete model of the work transference network rediscovered from the workflow and workitem event logs.

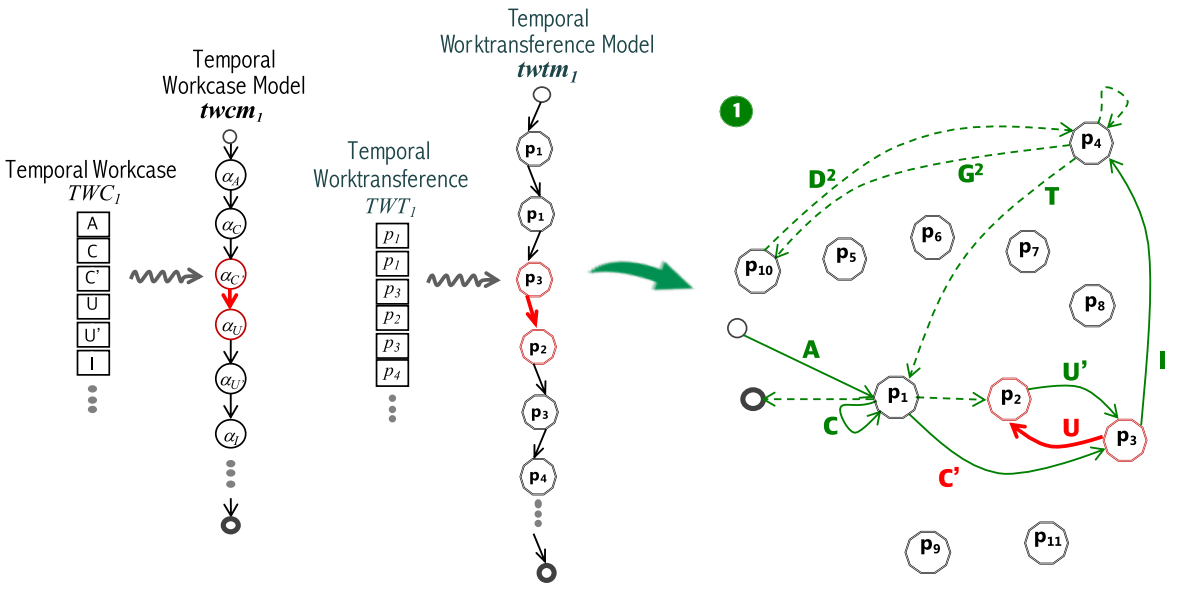

Fig. 11. Deliberate noise in a discovered work transference network originating from temporal workcases and worktransferences.

Algorithm 3 DELIBERATE NOISE DISCOVERY AND REMOVAL ALGORITHM: An Algorithm for Discovering and Removing Deliberate Noise from a Rediscovered Work Transference Network Model.

Require: (1) An ICN, $\Gamma = (\delta, \rho, \lambda, \varepsilon, \pi, \kappa, \mathbf{I}, \mathbf{O})$;(2) A Rediscovered Runtime Work Transference Network Model, $\Lambda^R = (\sigma^I, \psi^I, F_r^R, T_o^R)$;

Ensure: A Complete Runtime Work Transference Network Model, $\Lambda^R = (\sigma^C, \psi^C, F_r^R, T_o^R)$;

procedure MAIN ▷ Discovering and excluding deliberate noise from a rediscovered work transference network model
  for $\forall p \in \mathbf{P} \wedge \forall \alpha \in \mathbf{A}$ **do** ▷ **P** = Performer Set; **A** = Activity Set
    for $\forall q \in \sigma_i^I(p) \wedge \forall \beta \in \delta_o(\alpha) \wedge \forall \gamma \in \sigma_o^I(p)$ **do**
      ▷ Discovering deliberate noise from $\sigma^I = \sigma_i^I \cup \sigma_o^I$
      if *edge of* $(\psi_i((q, p)), \psi_o((p, \gamma)))$ ***is equal to*** *edge of* (α, β) **then** ▷ If TRUE, then the edge is not deliberate noise.
        ▷ $\sigma^C = \sigma_i^C \cup \sigma_o^C$: Building the work (workitem) transferences
        $\sigma_i^C(p) \leftarrow \sigma_i^C(p) + q$; ▷ Not deliberate noise: incoming performer of p
        $\sigma_o^C(p) \leftarrow \sigma_o^C(p) + \gamma$; ▷ Not deliberate noise: outgoing performer of p
        ▷ $\psi^C = \psi_i^C \cup \psi_o^C$: Building the work (workitem) associations
        $\psi_i^C((q, p)) \leftarrow \psi_i^C((q, p)) + \alpha$; ▷ Not deliberate noise: workitem on the incoming edge
        $\psi_o^C((p, \gamma)) \leftarrow \psi_o^C((p, \gamma)) + \beta$; ▷ Not deliberate noise: workitem on the outgoing edge
      end if
    end for
  end for
end procedure

5. Experimental verification and validation

In this section, we conduct an experimental analysis to verify and validate the correctness of the formal framework for rediscovering a work transference network model from a workflow event log dataset. Additionally, the experiment verifies the operational correctness of the algorithm for deliberate noise discovery and removal. For the purpose of this experiment, we have successfully implemented all of the algorithms, and related functions, of the proposed framework. Because of the scope of this paper and the page limitations, we omit the details of the implementation. In this experiment, we apply the implemented framework to a workflow event log dataset from the 4TU.Centre for Research Data [16].

5.1. Preparation of the workflow enactment event log dataset

As a workflow instance is executed, the logging and auditing component of the workflow enactment engine records its workitem execution events in a log, and these logged events are arranged in a temporal sequence. This execution sequence

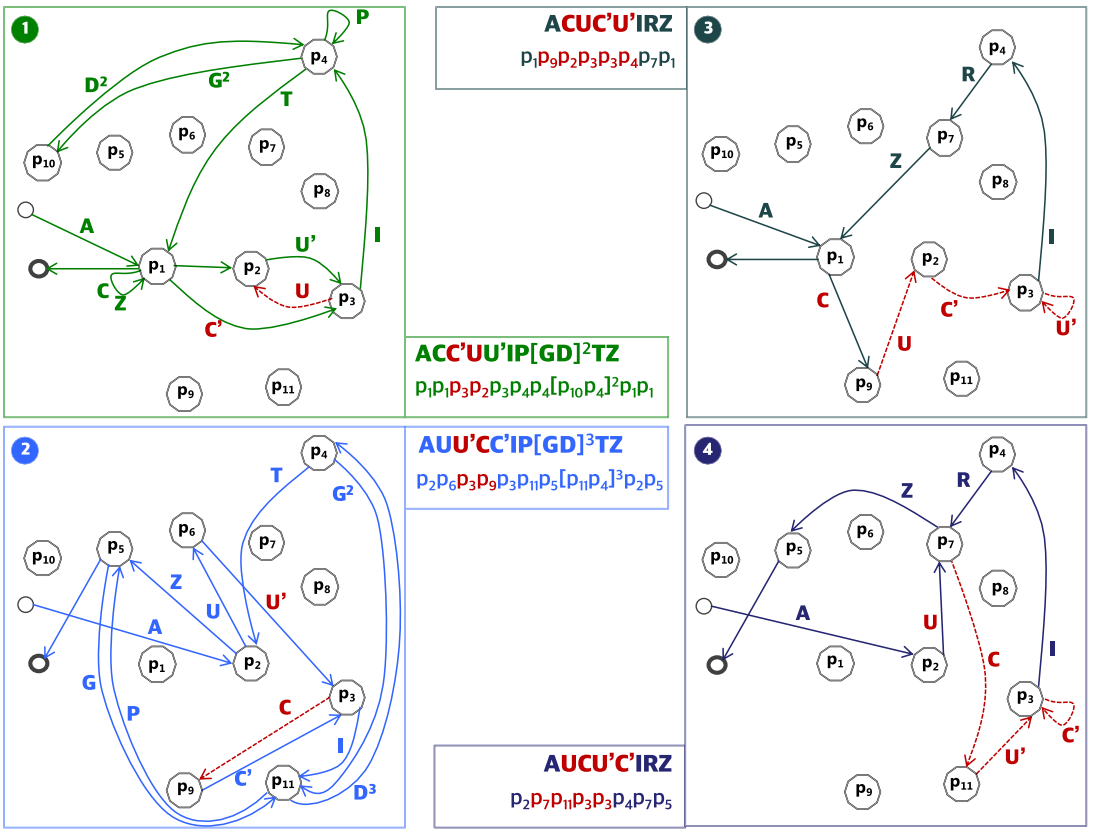

Fig. 12. Discovering deliberate noise from the four workflow instances enactment event log traces of the lba workflow model.

of a workflow instance forms a workflow instance event trace, from which we can extract the workflow instance's workitem event trace. Its formal representation is specified by the concept of temporal workcases (and its workcase model) and the concept of temporal worktransference (and its worktransference model), as described in the previous sections. In general, the log formats for the workflow enactment event log used to be a tag-based language like XWELL [11], BPAF [12], and XES [13]. In recent years, IEEE has released a standard tag-based language, XES (extensible event stream), whose aim is to provide designers of information systems with a unified and extensible methodology for capturing the behavior of systems by means of event logs and event streams. For the purpose of the experimental analysis, we prepared a process enactment event log dataset that was available from the 4TU.Centre for Research Data [16] and was well suited and appropriate for this experiment. The name of the dataset is "Review-Example-Log.xes," and it is formatted in the IEEE XES standard [13]. By experimental verification and validation, we expect to prove the functional correctness of the devised algorithms and the feasibility of the formal framework proposed in this paper.

5.2. The first round experiment: Validating the work transference network rediscovery framework

In this experiment, it is necessary to check the correctness of the work transference network discovery functionality for the dataset prepared in the previous subsection. We discovered that the dataset contains a total of 10,000 workflow instance event traces, with 14 activities and 11 performers, involving all types of primitive process patterns, including exclusive-OR, parallel-AND, and iterative-LOOP constructs. In addition, the characteristics of event logs in the dataset are real (and not noise), according to the description of the dataset posted on the 4TU.Centre website. Fig. 13 shows the work transference network model with embedded deliberate noise, finally being rediscovered from the prepared dataset. The number on each edge shows the number of occurrences of worktransference rediscovered from the dataset. We know that the rediscovered model should be incomplete, because it might embed a certain amount of deliberate noise. In the next round of this experiment, we apply the implemented function to discover and remove deliberate noise in this incomplete work transference network model. Through the implemented framework and its experimental verification results, we were able to prove that the formal framework is perfectly operable and feasible.

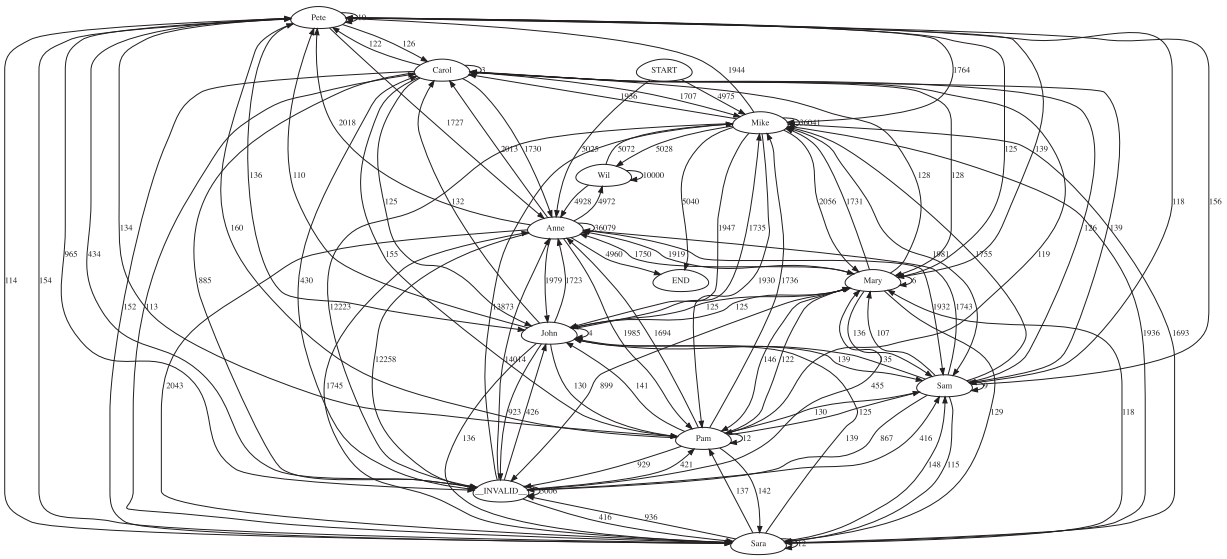

Fig. 13. A work transference network model embedding deliberate noise, rediscovered from the workflow event log dataset.

5.3. The second round experiment: validating the deliberate noise discovery and removal algorithm

Additionally, we implemented the algorithm for discovering and excluding deliberate noise from the rediscovered work transference network model. This was for both verifying the theoretical principle and validating the functional correctness of the algorithm. As the second round of the experiment, we applied the implemented algorithm to the rediscovered work transference network model produced in the first round of the experiment. Ultimately, we were able to find that a large amount of deliberate noise was embedded in the rediscovered model. Therefore, all of the deliberate noise was discovered and excluded when constructing the complete work transference network model. Fig. 14 shows two graph models generated by the implemented algorithm. The upper graph model visualizes (using the graphViz open source library) the work transference network model after eliminating the deliberate noise, and the lower graph model visualizes (using the GraphStream dynamic graph library) the complete work transference network. In the complete model, the performer nodes are represented by avatar-style faces, to help users to intuitively grasp the discovered work transference network models. There are 11 performers[20] who were involved in enacting the corresponding workflow model, and they were transferring 14 activities[21] to each other, to fulfill 10,000 workflow instances. In particular, from the complete work transference network model, we can easily recognize that the performers *Anne* and *Mike* were the most influential workers in performing the 10,000 workflow instances (*i.e.*, those paper review processing activities) of the corresponding workflow model.

6. Related work and summaries

Recently, there has been substantial newly emerging research and development issues, trials, and experiments in the workflow literature. One of the most recent issues and experiments is the issue of workflow mining and knowledge discovery, and its experiments, which are related to collecting runtime event log data in workflow logs, filtering out and forming workflow warehouses from the logs, and discovering knowledge from the workflow warehouses. Almost all of the recent workflow management systems provide their own logging mechanism [11,15] for organizing those workflow logs and warehouses. In particular, workflow logging mechanisms tend to have completely distributed architectures to support very large-scale workflow applications based upon object-oriented and internet-based infrastructures. Accordingly, the key paradigm for organizing workflow logs and warehouses is characterized by the properties of active objects, distributed architectures, and human-centered, intellectual, and large-scaled applications. This makes the workflow mining and discovery functionality, to reflect these paradigmatic properties, essential features of workflow management systems.

In terms of collecting, filtering, and discovering activities, using workflow logs and warehouses, the workflow literature to date has been focused mainly on two types of workflow discovery activities: workflow process discovery [10,17–20] and workflow knowledge discovery [2,3,8,21–26]. Workflow process discovery is directly concerned with redesigning and reengineering the process aspect (control flow) of workflow models and applications by rediscovering temporal workcases from the

[20] The names of the performers involved are Mike, Mary, Anne, Pete, Carol, John, Sam, Pam, Wil, Sara, and Unknown (INVALID).

[21] The titles of the associated activities are: invite reviewers, get review 1, get review 2, get review 3, time-out 1, time-out 2, time-out 3, collect reviews, decide, invite additional reviewer, get review X, time-out X, accept, and reject.

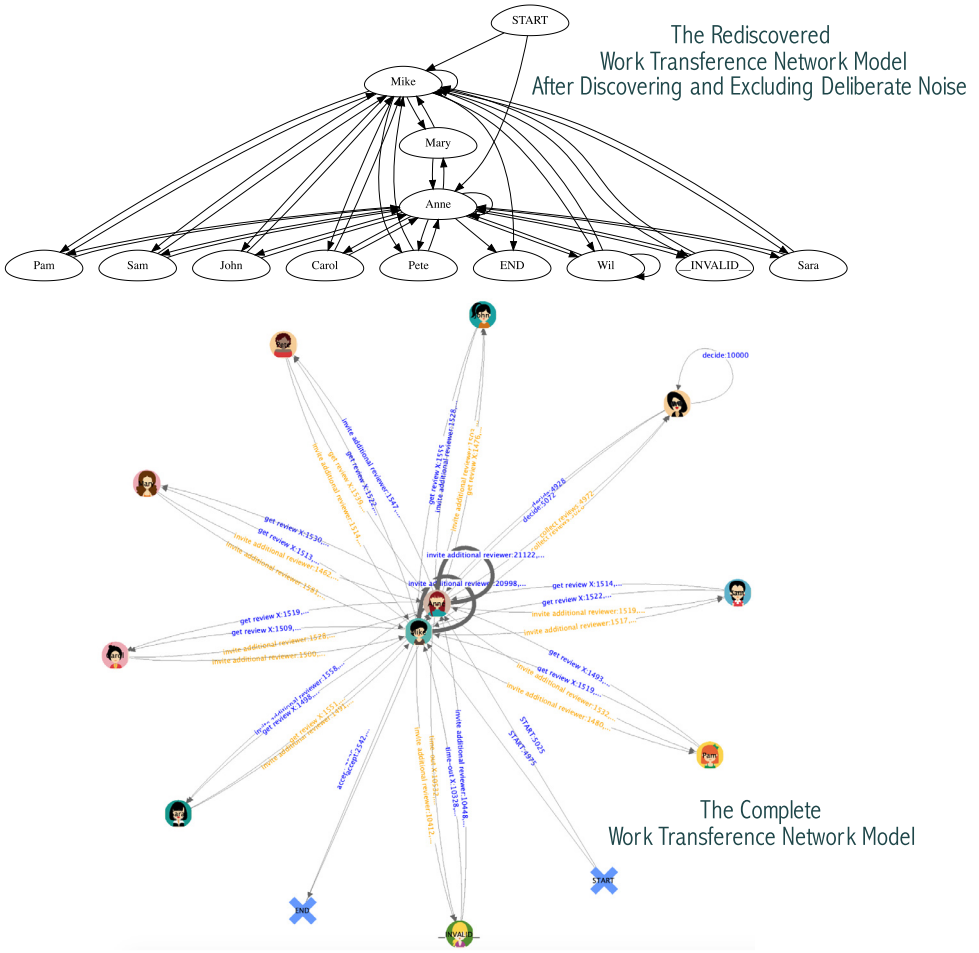

Fig. 14. The complete work transference network model ultimately rediscovered from the workflow event log dataset.

workflow logs and warehouses, whereas workflow knowledge discovery is closely connected with replanning and realigning the organizational resource aspect (such as data flow, performer, role, and program) of workflow models and applications by rediscovering enacting and binding histories (*e.g.*, temporal worktransferences) from the workflow logs and warehouses. The eventual goal of workflow discovery is to measure, evaluate, control, and predict the degree of workflow fidelity in a workflow-supported organization. Through the novel concept of workflow fidelity, we are able to achieve the managerial decision-making goal of minimizing the discrepancies between the planned workflow models, as estimated at build time, and their enacted workflow models, as actually performed at run time. In this paper, we have addressed the buildtime and runtime types of workflow discovery approaches by the model of temporal workcases and the model of temporal worktransferences, respectively. The typical previous work related to these types of workflow discovery are summarized as follows:

(1) Workflow process discovery. In the introductory paper [17] of a special issue[22] on workflow and process mining, the authors first initiated the topic of workflow process discovery and mining approaches, and discussed the main issues around the topic by introducing the context of the papers carefully selected for the special issue. In this paper, they defined the concept of process mining as extracting information from event logs to capture the workflow process as it is being executed and defined the scope of the operational workflow processes as ranging from simple internal back-office processes to complex inter-organizational processes. In [18], the authors characterized the definition of workflow models as a very time consuming and error-prone task and understood the concept of workflow process mining as a reasonable approach for supporting the task of defining workflow models. Therefore, they gave an overview of the algorithms that were implemented within the InWoLvE workflow mining system and summarized the most important results of their experimental evaluation that was performed in the first industrial application of InWoLvE. In [10], the authors understood workflow systems as people systems that must be designed, deployed,

[22] The special issue was drawn up by a leading professional journal, Computers in Industry.

and understood within social and organizational contexts. Therefore, they asserted that the workflow discovery topics need to encompass behavioral (process/control flow), social, informational (data flow), and organizational perspectives. As an algorithmic approach for mining the behavioral perspective (*e.g.*, a workflow process) from the enactment event logs of workflow models, they proposed the so-called σ-algorithm, which is able to discover an ICN of a structured workflow process model by incrementally amalgamating a series of temporal workcases (workflow instance event traces) according to three types (sequential, selective, and parallel) of basic merging principles conceived in this paper. In [27], the authors proposed a conceptual approach to discover a proportional information control net from the enactment event histories of the corresponding business process and a series of procedural frameworks and operational mechanisms formally and graphically supporting the proposed approach. In [19], the authors proposed the so-called time interval approach, comprising two algorithms, which could synthesize workflow process models from sets of workflow enactment and audit logs, each of which is represented by a directed graph annotated with activity names. In addition, they compared the workflow process model synthesized by the proposed approach to the mining results of others, and proved that their approach is able to discover a much more accurate workflow process model. Finally, the exactness of the discovered workflow process model was investigated by [20] and [28], who emphasized the importance of mining exact workflow process models from workflow enactment event logs. In [20], the authors outlined and specified a clear representation for workflow process models as the input of their mining approach and precisely described a complete workflow mining procedure by a stepwise illustration with an example. In [28], the authors proposed a discovery approach that deals with a specific process pattern of redo-activities and its performer involvement knowledge.

(2) Workflow knowledge discovery. The topic of workflow knowledge discovery stems from the concepts of business process intelligence [23] and workflow intelligence [29]. In [23], the authors claimed business process intelligence to be a suite of integrated software tools aimed at managing workflow execution quality by providing several features, such as analysis, prediction, monitoring, control, and optimization. Through the business process intelligence suite, we are able to accomplish a higher level of enhancements, such as automating prevention of exceptions, refining workflow data preparations, and integrating other data mining techniques, in managing workflow execution quality. [23] is a conceptual contribution, whereas [29] is a much more theoretical and concrete contribution. In [29] and [26], the authors proposed a framework of control path-oriented workflow intelligence and quality improvement, to achieve a higher degree of workflow traceability and rediscoverability, and devised an efficient approach to control path analysis, through the concept of the minimal workflow model. In particular, the discovered knowledge in the proposed framework is a quantitative measure of runtime enactments, according to each of the control paths generated from a workflow model. The control paths and reachable paths, with their runtime enactment occurrences, become extremely valuable knowledge for redesigning and reengineering the corresponding workflow model. In particular, from the perspective of workflow intelligence, the workflow knowledge discovery topic needs to combine all of the perspectives, such as behavioral, social, informational, historical, performative, temporal, resource management, and organizational perspectives. [9] first initiated the human-centered knowledge discovery issue on workflows, by observing the collaborative behavior between workflow performers. In this paper, the authors proposed a formal approach, with an algorithm that is able to discover a procedural collaboration network connecting the workflow performers who are involved in enacting a specific workflow model. Subsequently, the research work of [21] became the major turning point in the human-centered workflow knowledge discovery topic. In the research work, the authors understood the workflow performers' network as the relationships of work hand-over between the workflow performers. That is, they interpreted the concept of workflow performer networks using the concept of social networks, discovered workflow social networks from workflow logs, and analyzed them by using social network analysis techniques. In addition, [3] developed another approach and a system for modeling, discovering, analyzing, and visualizing the novel concept of workflow-supported social networks. Moreover, [4] and [5] developed a theoretical framework and visualization framework for supporting the theoretical analysis approach and the graphical representation approach, respectively, of the workflow performer network. This mainly focused on closeness centrality measurements, which are among the most important social network analysis techniques. In [2,25,30], the authors found another aspect of workflow knowledge to be used for supporting the activities of resource management and decision-making, which can be formally represented by the concept of workflow affiliation networks. They developed a discovery algorithm [30], which is able to extract the so-called workflow performer–role affiliation networking knowledge from a workflow model, and formalized a series of approaches [25] for modeling, discovering, and visualizing the knowledge. These research results related to integrating the concepts of workflows and social networks. In addition, handling behavioral knowledge for concurrent workflows, discovering temporal workflow patterns, and mining event logs for workflow resource allocations were addressed by Cook et al. [22], Hwang et al. [24], Liu et al. [8] and Pham et al. [28], respectively.

In summary, the workflow literature has started to be interested in repositioning workflow systems as a tool of business and organizational knowledge and intelligence. It began from a strong belief that social relationships and collaborative behavior between workflow performers may affect the overall performance and success in real businesses and working productivity. Some typical outcomes of this repositioning work include [3,9,21,25], in which the authors formalized models, discovery algorithms, and analysis techniques for several types of human-centered networking knowledge, such as workflow-supported social networking knowledge and affiliation networking knowledge, in workflow models and their en-

actment event histories. In this paper, we focused on a new type of human-centered networking knowledge hidden inside workflow procedures, namely, the work transference networking knowledge that we have proposed. By discovering the work transference networking knowledge, we are able to quantitatively measure the degrees of work sharing and work relevancy between workers, and qualitatively estimate the work intensity of workers in a workflow-supported organization. Based upon the discovered work transference networking knowledge, we can measure and estimate how close the performers are to each other, with respect to workflow-supported activities. In more detail, it provides analytical work transferences, by finding answers to the following questions:

- How many others can a performer interact with, via very few intermediary activities, in enacting workflow procedures?
- How closely can a performer connect to others, via very few intermediary performers, in enacting workflow procedures?
- Which activities does a specific performer transfer to which performers, in enacting workflow procedures?
- How many groups of work transference paths are there, in enacting each of the workflow procedures?

7. Conclusion

In this paper, we proposed a formal approach for discovering work transference networks from workflow enactment event histories and logs. A concept of work transference networking knowledge can be discovered at the build time and run time of a workflow procedure. Accordingly, we proposed two types of formal discovery approach, supported by models and algorithms. One is a method of discovering a work transference network at build time from an ICN of a workflow procedure; the other is a method of discovering a work transference network at run time from a collection of workflow instance event traces. In particular, we developed a series of formal models and algorithms, such as the temporal workcase model and the temporal worktransference model, that cover the formal discovery approach of this paper. We demonstrated the proposed formal approach by illustrating its application to four different workcases spawned from the example workflow model. As a consequence of the paper, we devised a formal framework for rediscovering work transference networks by a stepwise discovery procedure that amalgamates the trace logs incrementally. This was the final goal of our study on the issue of discovering work transference networking knowledge. Additionally, we finalized the formal framework through the concept of deliberate noise, and an algorithm for its discovery and elimination. Finally, we conducted an experimental study to verify and validate the proposed concepts, algorithms, and formal framework by implementing the framework and applying it to a workflow event log dataset. As future work, we plan to extend the formal framework and its implemented system, to develop workflow and business process intelligence and knowledge discovery solutions.

Declaration of Competing Interest

None.

Acknowledgement

This research was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF), funded by the Ministry of Science, ICT & Future Planning (Grant No. 2017R1A2B2010697). Additionally, we would like to express our special thanks to Editage (www.editage.co.kr) for English language editing.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.ins.2019.11.036.

References

[1] K. Kim, C.A. Ellis, ICN-based workflow model and its advances, Handbook of Research on Business Process Modeling (first ed.). IGI Global, Chapter 7, 2009, doi:10.4018/978-1-60566-288-6.
[2] H. Kim, H. Ahn, H. Jeong, K.P. Kim, A workflow performer-role affiliation networking knowledge discovery algorithm, ICIC Express Lett. 8 (3) (2014) 717–722.
[3] M. Park, H. Ahn, K.P. Kim, Workflow-supported social networks: discovery, analyses, and system, J. Netw. Comput. Appl. 75 (2016) 355–373, doi:10.1016/j.jnca.2016.08.014.
[4] M.-J. Kim, H. Ahn, M. Park, A graphML-based visualization framework for workflow-performers closeness centrality measurements, KSII Trans. Internet Inf. Syst. 9 (8) (2015) 3216–3230, doi:10.3837/tiis.2015.08.028.
[5] M.-J. Kim, H. Ahn, M. Park, A theoretical framework for closeness centralization measurements in a workflow-supported organization, KSII Trans. Internet Inf. Syst. 9 (9) (2015) 3611–3634, doi:10.3837/tiis.2015.09.018.
[6] K.P. Kim, An XPDL-based workflow control-structure and data-sequence analyzer, KSII Trans. Internet Inf. Syst. 13 (3) (2019) 1702–1721, doi:10.3837/tiis.2019.03.034.
[7] C. Ellis, A. Rembert, J. Wainer, K.P. Kim, Investigations on stochastic information control nets, Inf. Sci. 194 (2012) 120–137, doi:10.1016/j.ins.2011.07.031.
[8] T. Liu, Y. Cheng, Z. Ni, Mining event logs to support workflow resource allocation, Knowl.-Based Syst. 35 (2012) 320–331, doi:10.1016/j.knosys.2012.05.010.
[9] K. Kim, Actor-oriented workflow model, in: Proceedings of the 2nd International Symposium on Cooperative Database Systems for Advanced Applications, Wollongong, Australia, 1999, pp. 150–164.
[10] K. Kim, C.A. Ellis, σ-algorithm:structured workflow process mining through amalgamating temporal workcases, Lect. Notes Artif. Intell. 4426 (2007) 119–130, doi:10.1007/978-3-540-71701-0_14.

[11] M. Park, K. Kim, XWELL: a xml-based workflow event logging mechanism and language for workflow mining systems, Lect. Notes Comput. Sci. 4707 (2007) 900–909, doi:10.1007/978-3-540-74484-9_76.
[12] M. zur Muehlen, K.D. Swenson, BPAF: a standard for the interchange of process analytics data, Lect. Notes Bus. Inf. Proecss. 66 (2011) 170–181, doi:10.1007/978-3-642-20511-8_15.
[13] IEEE Standard for eXtensible Event Stream (XES) for Achieving Interoperability in Event Logs and Event Streams, 1849–2016 - IEEE. doi:10.1109/IEEESTD.2016.7740858
[14] K. Kim, An xml-based workflow event logging mechanism for workflow mining, Lect. Notes Comput. Sci. 3842 (2006) 132–136, doi:10.1007/11610496_17.
[15] M. Park, K.P. Kim, A workflow performer-centered runtime state model, ICIC Express Lett. 8 (2) (2014) 505–510.
[16] T.C. for Research Data, Bpi challenges, 2012, 2013, 2014, 2015, 2016, 2017, 2018, (BPM).
[17] W. van der Aalst, A. Weijters, Process mining: a research agenda, J. Comput. Ind. 53 (2004) 231–244, doi:10.1016/j.compind.2003.10.001.
[18] J. Herbst, D. Karagiannis, Workflow mining with Inwolve, J. Comput. Ind. 53 (2004) 245–264, doi:10.1016/j.compind.2003.10.002.
[19] S.S. Pinter, M. Golani, Discovering workflow models from activities' lifespans, J. Comput. Ind. 53 (2004) 283–296, doi:10.1016/j.compind.2003.10.004.
[20] G. Schimm, Mining exact models of concurrent workflows, J. Comput. Ind. 53 (2004) 265–281, doi:10.1016/j.compind.2003.10.003.
[21] W.M.P. van der Aalst, H.A. Reijers, M. Song, Discovering social networks from event logs, Comput. Supported Coop. Work 14 (6) (2005) 549–593, doi:10.1016/j.compind.2003.10.005.
[22] J.E. Cook, Z. Du, C. Liu, A.L. Wolf, Discovering models of behavior for concurrent workflows, J. Comput. Ind. 53 (2004) 249–319, doi:10.1016/j.compind.2003.10.005.
[23] D. Grigori, F. Casati, M. Castellanos, U. Dayal, M. Sayal, M.-C. Shan, Business process intelligence, J. Comput. Ind. 53 (2004) 321–343, doi:10.1016/j.compind.2003.10.006.
[24] S.-Y. Hwang, C.-P. Wei, W.-S. Yang, Discovery of temporal patterns from process instances, J. Comput. Ind. 53 (2004) 345–364, doi:10.1016/j.compind.2003.10.007.
[25] H. Kim, H. Ahn, K.P. Kim, Modeling, discovering, and visualizing workflow performer-role affiliation networking knowledge, KSII Trans. Internet Inf. Syst. 8 (2) (2014) 689–706, doi:10.3837/tiis.2014.02.022.
[26] K. Kim, C.A. Ellis, Workflow reduction for reachable-path rediscovery in workflow mining, in: T. Young Lin, S. Ohsuga, C.J. Liau, X. Hu (Eds.), Foundations and Novel Approaches in Data Mining, 9, 2006, pp. 289–310, doi:10.1007/11539827-17.
[27] K. Kim, M. Yeon, B. Jeong, K. Kim, A conceptual approach for discovering proportions of disjunctive routing patterns in a business process model, KSII Trans. Internet Inf. Syst. 11 (2) (2017) 1148–1161, doi:10.3837/tiis.2017.02.030.
[28] D. Pham, H. Ahn, K.P. Kim, Discovering redo-activities and performers' involvements from XES-formatted workflow process enactment event logs, KSII Trans. Internet Inf. Syst. 13 (8) (2019) 4108–4122, doi:10.3837/tiis.2019.08.016.
[29] M. Park, K. Kim, Control-path oriented workflow intelligence analyses, J. Inf. Sci. Eng. 24 (2) (2008) 343–359, doi:10.1688/JISE.2008.24.2.2.
[30] K.P. Kim, Discovering activity-performer affiliation knowledge on ICN-based workflow model, J. Inf. Sci. Eng. 29 (2013) 79–97. http://www.iis.sinica.edu.tw/page/jise/2013/201301_06.pdf.