applied sciences

MDPI

Article

Mining Shift Work Operation from Event Logs

Nur Ichsan Utama [1], Riska Asriana Sutrisnowati [2], Imam Mustafa Kamal [3], Hyerim Bae [1,*] and You-Jin Park [4]

1 Department of Industrial Engineering, Pusan National University, Busan 46241, Korea; nichsan@pusan.ac.kr
2 IOChord Inc., Busan 48059, Korea; riska@iochord.com
3 Department of Big Data, Pusan National University, Busan 46241, Korea; imamkamal@pusan.ac.kr
4 Department of Industrial Engineering and Management, National Taipei University of Technology, Taipei 10608, Taiwan; yjpark@mail.ntut.edu.tw
* Correspondence: hrbae@pusan.ac.kr; Tel.: +82-51-510-2733 or +82-51-512-7603

Received: 30 August 2020; Accepted: 9 October 2020; Published: 15 October 2020

Abstract: Event logs are records of events that are generally used in process mining to determine the manner in which various processes are practically implemented. Previous studies on process mining attempted to combine the results based on different perspectives such as control flow, data, performance, and resources (organizational) to create a simulation model. This study focuses on the resource perspective. A prior study from the resource perspective focused on clustering the resources into organizational units. Implementing the results of the above study in a simulation model will yield inaccurate results because the resources are assumed to always be available if no task is performed. In a practical scenario, resources (particularly humans) tend to work based on shifts. Thus, we propose mining the shift work operation of resources from event logs to tackle this issue. We utilized a self-organizing map and *k*-means clustering to incorporate the shift work information from the event logs into the simulation model. Moreover, we introduce a distance function and weight-centroid updating rule in the clustering technique to realize our objective. We conducted extensive experiments with artificial data sets to assess the effectiveness of the proposed method. The simulation shows that introducing the shift work operation time of resources can yield more accurate results. Furthermore, the proposed distance function can capture the shift work operation of the resources more precisely compared with the general distance function.

Keywords: event logs; resource behaviour; simulation; clustering

1. Introduction

Nowadays, with the capability of computer hardware, every company can record any events of its operational activities in the system. Recorded events can be used by the company for further analysis. The possible use of recorded event data is to create a data-driven simulation model. The simulation model derived from the event data directly is very useful for users who are unfamiliar with simulations. Discrete-event simulation (DES) is one of the most popular existing simulation types [1]. In DES, an event is typically defined as a specific change in the state of the system at a particular time [1]. Another possible use of the recorded event data is to analyze the discrepancy between the guided plan of operational activity and the real activity in the field. A possible technique to conduct this type of analysis is process mining.

Process mining is both data-driven and process-centric [2]. Although process mining is generally used for automated process discovery, it can be used for other purposes such as checking compliance, diagnosing deviations (base and real), highlighting bottlenecks, improving performance, predicting flow times, and recommending action [2]. The input for process mining is an event log [2]. This event log may contain all the events related to operational activity. In process mining, each event generally

refers to an activity that occurs at a particular time for specific cases [3]. Evidently, the definition of events in process mining and DES is related. Moreover, event data can be generated as in process mining based on the rules defined in the simulation model. Hence, combining process mining and simulation such that the results of the two can complement each other is reasonable, as suggested by Aalst [3].

To the best of our knowledge, the first practical approach that combined process mining and simulation was proposed in 2009 [4]. The author used a process model based on a Petri net representation discovered from an event log using a process discovery algorithm [4]. This Petri net comprises places, transitions, and arcs. The process discovery algorithm used in this study is an alpha algorithm used to mine the workflow process from the event log [5]. The discovered process model represents a control-flow perspective, combined with other analysis techniques such as decision point analysis, organizational miner, and statistical analysis [4]. Decision point analysis is used to mine decisions based on the attribute data that influence the choices made in the process based on past process executions [6]. The decision is triggered when the next execution occurs at a place where more than one possible output (activities) can be obtained. An organizational miner is conducted to mine resource allocations in a specific activity. Several algorithms such as default mining, doing similar task, and agglomerative hierarchical clustering can be used to perform this resource allocation [7]. This aspect is also important in simulation because the execution of an activity in simulation is generally constrained by the resources. Statistical analysis is used to analyze the performance based on the event log. This analysis can also be used to fit the distribution of several parameters for simulation such as the execution time of activity and the arrival time of cases [4]. The results from the organizational miner, decision point analysis, and statistical analysis are then merged and transformed into a colored Petri net (CPN) representation. If a considerable amount of data in the event log required to determine the parameters of the simulation is missing, the missing data can be predicted using machine learning by first changing the representation of the event log into an image [8]. Related with mining the event log, another interesting research in process mining by implementing a verification method for the message passing behavior of IoT systems is proposed to extract event relations from the log and check for any minor deviations that exist in the system [9]. Visual filtering approached for event logs that make the process analysis tasks more feasible and tractable are conducted and analyzed with the event log data from Business Process Intelligence (BPI) challenge 2017 [10].

Although the idea of creating a CPN model based on process mining is promising, the first practical implementation still uses general/incorrect assumptions—for instance, all the resources are available if no task is being performed, and the processing time of the resources when executing tasks is always based on a specific underlying distribution in any condition [4]. Several studies have attempted to mitigate this limitation. In particular, the resource behavior in process mining has been analyzed to explore the effect of workload on service times and subsequently incorporate it into the simulation model [11]. This study assumed that a worker under time pressure may become more efficient and can consequently finish tasks faster [11]. Another study calculated the batch process and resource availability for subsequent incorporation into the CPN model [12]. The event logs were analyzed to calculate the percentage of resource availability in a specific period [12]. Our study attempts to complement the previous studies on resource availability by incorporating shift time. The previous study did not set resource availability at specific times, i.e., it did not consider that resources, particularly humans, generally work based on shift times. Although the simulation model incorporates resource availability based on percentages, it still has a gap from reality, as it does not consider shift times. Without knowing the information related to the shift time, the simulation model generated could still not imitating reality. We attempt to use an event log to analyze and cluster the shift times of resources so that this information can be used in the simulation model and could improve the accuracy of the model.

The main contribution of our research is as follow:

- We proposed the clustering method of mining the shift work operation of resources that can be put in the simulation model to generate a more accurate model.
- We propose the new distance function to handle the problem that can be happened when clustering the time data.
- We propose the new representation of activity that can handle the resource shift work rule in CPN.

The remainder of this paper is structured as follows. Section 2 discusses some terminology, provides an overview of the related work, and presents a running example. Section 3 outlines the data-driven method to retrieve shift work information. Section 4 will explain about how we incorporate this shift work information in the simulation model. Section 5 describes the explanation about the mechanism of how we generate the artificial event logs. The result from the method in Section 3 is empirically evaluated in Section 6. The paper ends with a discussion in Section 7 and a conclusion in Section 8.

2. Preliminary

In this section, we describe the preliminary of our study. The definition and overview of process mining, event logs, process model, Petri net, CPN, resource allocation, and resource shift time work of operation will be described bellow.

2.1. Process Mining

Process mining has been used in the field of process management to support the analysis of business process based on the event logs. During process mining, specific data mining algorithms are applied to event log data in order to identify trends, patterns, and details contained in event logs recorded by an information system. Process mining aims to improve process efficiency and understanding of processes.

There are three classes of process mining techniques. This classification is based on whether there is a prior model and, if so, how the prior model is used during process mining [2].

- Discovery
  The discovery technique is used in process mining, usually when a prior model does not exist. Based on the event log, a new model is constructed based on low-level events.
- Conformance Checking
  Conformance checking is used when there is a prior model. The existing model is compared with the process event log. Discrepancies between the log and model are analyzed.
- Performance Mining
  The process model based on a prior model or generated from a discovery technique is extended with performance information such as processing times, cycle times, waiting times, etc.

2.2. Event Logs

This section formally defines an event log describing an event, event properties, and traces. Let E be the event universe, i.e., the set of all possible event identifiers, A be a set of all possible activity names, R be a set of all possible resource names, and T be the time domain. An event $e \in E$ has a number of properties, and we can define functions assigning properties to events:

- act $\in E \rightarrow A$ assigning activities to events,
- type $\in E \rightarrow$ {schedule, assign, start, suspend, resume, abort activity, complete} assigning event types to the events,
- time $\in E \rightarrow T$ assigning timestamps to events, and
- $res \in E \rightarrow R$ assigning resources to events.

An event *e* is described by some unique identifier and can have several properties. An event log is a set of events. Each event in the log is linked to a particular trace which represents a process instance and is globally unique, i.e., the same event cannot occur twice in a log [11].

2.3. Process Model

A process model is a graphical representation of a process. The model or modeling can be based on various notations and standards, such as Business Process Management Notation (BPMN) 2.0, Petri net, Event Driven Process Chain (EPC), or Unified Modeling Language (UML) activity diagrams, etc. A process model can be created manually or discovered from the event logs using several process discovery algorithm based on the process mining technique.

2.4. Petri Net

A (classical) Petri net is a directed bipartite graph with two types of nodes called places and transitions. The places and transitions are connected to each other with directed arcs. Arcs connecting two nodes of the same type to each other are forbidden. In a Petri net, places are graphically represented by circles, and transitions are represented by rectangles. The elements of places and transitions are referred to as nodes. A node is an input node of a second node if and only if an arc connects this node to the second node. The representation of a Petri Net is shown in Figure 1 below.

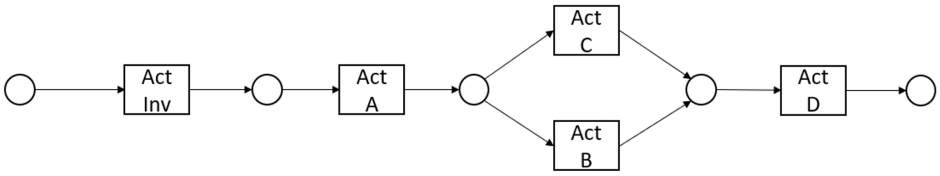


Figure 1. Example of a Petri Net representation.

2.5. Colored Petri Nets

Colored Petri nets (CPN) provide a modeling language that combines Petri nets with the functional programming language [13]. Previously, the functional programming language used for executing CPN is based on Standard ML, but currently, several tools' executions of CPN are based on other different programming languages. Petri nets model the basic behavior of a process while the programming language handles the definition and manipulation of data values. In a CPN model, all the tokens are distinguishable by assigning to each token a certain value, referred to as a color. Figure 2 shows a fragment of a CPN model, which consists of two places and one transition. The circles in the model indicate the places in a Petri net, and the boxes indicate the transitions. Each place in a colored Petri net has a certain type. Each transition can have certain properties such as guard, action, and priority. Guard is represented as the conditional execution of the transition.

CPN is a discrete-event modeling language that combines the capabilities of Petri nets with those of a high-level programming language [13]. CPN belongs to the class of high-level Petri nets owing to the inclusion of several aspects, such as time, colored (token attributes), and hierarchical representation. CPN models are executable; hence, they can be used to simulate different scenarios of the system. A computer tool for CPN modeling that can perform simulation-based performance analysis is the CPN tool. Dealing with CPN from the process mining perspective (process model on Petri nets) is easy because the foundation of the representation is the same. Moreover, hierarchical representation in the CPN can omit the details of the model in the specific module and provide the detailed implementation in another module. Therefore, it is conceivable to create a module in CPN based on the Petri nets generated from process discovery and set the detailed implementation of each transition in a separate module. An example representation of CPN is shown in Figure 2 below.

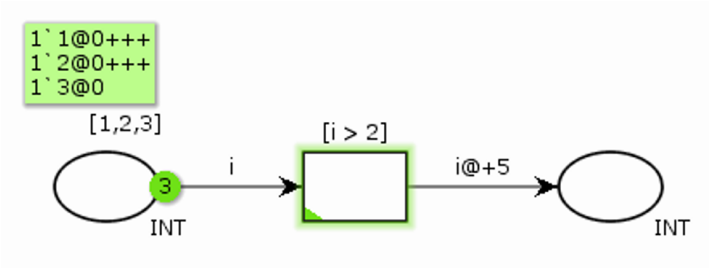


Figure 2. Example of Colored Petri Nets model.

2.6. Resource Allocation

By considering the resource assignment, resource allocation determines which specific employee or resource will execute a work item. From the process mining perspective, this execution of work items can be related to the activity.

Resource allocation has also been extensively studied. This involves investigating the allocation of work items to specific staff members while taking into account, for example, relations between resources and emergency circumstances. In the process mining field, the organizational miner is one of the algorithms that try to conduct this kind of approach by clustering resources into several organizational units and then assigning the organization unit to specific activities.

2.7. Resource Shift Time Work of Operation

Usually, humans work daily based on shift time. Shift work for humans basically can be divided into two categories, which are fixed or rotational shift work. Fixed shift work follows the same pattern every day at a fixed time, whereas rotational shift work changes the working time of resource at regular intervals. It is important to distinguish between the two types of shift work because their characteristics are distinctly different. For example, fixed shift workers should be able to keep their daily rhythms as consistent as possible, so the resource will work based on this pattern consistently.

This information is somehow very important to be considered when we want to simulate the work operation of the process. Without considering this information in the model when performing the simulation can lead to generating less useful results because the model is not reflecting the reality. This is the reason we tried to conducting mining the shift work of human resources based on the event log.

3. Shift Time Clustering on Event Logs

To the best of our knowledge, clustering shift times based on event logs has not been studied thus far. Although regular event logs may not contain data about resources, the event log in this study is required to include data about resources. Before conducting clustering, we first need to preprocess the event log data. This preprocessing mechanism is based on the assumption that resources, i.e., humans, will have a certain amount of rest time between consecutive shift time working. This assumption is based on data from the International Labor Organization, which released a report and standard guideline for the working times of a worker employed by a company [14]. The guideline states not only the limitations of daily and weekly working times but also the rest period after work [15]. It states that the rest period after work must follow the following rules [15]:

- After a working day, you may not work for at least 11 consecutive hours. However, this may be reduced once every seven days to eight hours if the nature of the work or the company circumstances make this necessary.
- After a working week of five days, you may not work for at least 36 consecutive hours.
- • A longer working week is also possible, provided you have a rest period of at least 72 h once every 14 days. This 72 h period may be split into two separate periods, neither of which may be shorter than 32 h.

We need to find a rest period between the working times of each resource to cluster the shift time of an event log. In the event logs, data are grouped based on case ID. Here, we need to rearrange the event logs so that we can easily map each resource across multiple cases. Based on the rules defined in [15], we preprocess the event log to merge and split the row of each resource in the event logs. First, we group the event log by each resource and sort the row of resources of each group by the start time of the event in ascending order. Subsequently, we calculate the gap between the completion time of the previous event and the starting time of the subsequent event. We split two events if the calculated gap is equal to or longer than eight hours. This split creates another subgroup consisting of rows of events. Subsequently, we take the starting time of the first event, the completion time of the last event, and the distance between the two for each subgroup consisting of rows of events. The data here are not represented by the shift time of the resource directly because the recorded event of each resource may not consist of a complete event from the start to the end of the shift time. Hence, we use clustering for this problem, not only to cluster the resources into a specific group of shift times but also to be more certain of the shift time range. The preprocessing mechanism is shown in Algorithm 1.

Algorithm 1: Event logs preprocessing.

Input : Event Logs
Output: Array of object

1 Read event logs data.
2 Group rows of an event by resource.
3 Order rows in each group by start time attribute ascendingly.
4 Calculate the gap (completion of the first event and starting time of the second event) between two consecutive event.
5 Split two consecutive event if the gap $\geq$ 8 h. This split will create new subgroup.
6 For each subgroup, take the first event starting time and the last event completion time, calculate the duration between the two.
7 From the previous step, take the only time of the starting time and the completion time of each subgroup without considering the date.
8 Make an object data consist of the starting time (without date info), the completion time (without date info), and the duration.
9 Flatten each object into one collection of array.

The formalized representation of Algorithm 1 can be described by the following mathematical definitions:

- Event logs
  Based on the definition of event logs in the Preliminary section, an event is an atomic representation to an activity instance. It thus contains a single timestamp. However, in Algorithm 1, we refer to the event with start and complete lifecycle. An event $e \in E$ has a number of properties such as *caseid*, *activity*, *resource*, *start*, and *complete timestamp*.
- Resource
  R be a set of all possible resource names, where $res \in E \rightarrow R$ assigning resources to events. $R = \{r_1, r_2, r_3, \ldots, r_n\}$.
- Group rows of an event by resource and order by start time
  G is a set of subset event group by the resources.
  $G = \{\{e_1, e_2, e_3, \ldots, e_x\} \rightarrow r(e) = r1, \{e_5, e_6, e_7, \ldots, e_y\} \rightarrow r(e) = r2, \ldots\}$, each subset of events is ordered by start time attribute.
- Splitting G of set into another subset SG
  SG is a set of subset event group after splitting.

$SG = \{\{e_1, e_2, \ldots, e_{(x-a)}\} \rightarrow r(e) = r1, \{e_{(x-a+1)}, e_{(x-a+2)}, e_{(x-a+3)}, \ldots, e_x\} \rightarrow r(e) = r1, \ldots\}$, where e_{x-a} and e_{x-a+1} have the gap ≥ 8 h

- The array of object data
  SOB is a set of object data (array of object data).
  $SOB = \{OB_1, OB_2, OB_3, \ldots, OB_n\}$, where each OB_i have $\{start, completion, duration\}$ information. Start and completion time in OB_i is contained only time (hour/minutes/seconds) without any date information. The start and the completion in OB_i are gathered from the subset event explained previously. Take the start time of the first event in each subset as a start for OB_i and take the end time of the last event in each subset as a completion.

As explained in the above algorithm, we consider only the time without the date to cluster the data. The date is neglected because, in shift time clustering, working hours that start today at time, for example, 08:00 should be considered the same as other working hours that started yesterday or last week also at the time 08:00. This is reasonable for shift time clustering because the critical aspect is the time, but not the date. In addition to time information, we include the distance (duration of working hours) in the input. Clustering the data considering only the time but not the date is a trivial task but should be performed carefully. The distance between two different times could not be calculated using only a general distance function, as shown in Figure 3. The distance between time 23 and time 18 is 5 based on Figure 3 and based on normal calculation $|23 - 18| = 5$. However, the distance between time 23 and time 1 is 2 based on Figure 3, but if we calculate the distance normally, it becomes 21, $|23 - 2| = 21$ (which is not correct). We need to modify the distance function between times t_i and t_j to calculate the distance accurately. The symbol tr here refers to the time threshold for a specific time unit. For example, if the time unit is hours, tr is 24, and if the time unit is minutes, tr is 1440. The distance function of time, which always generates a positive value, can be shown in the following equation:

$$D(i,j) = \min(|t_i - t_j|, tr + \min(t_i, t_j) - \max(t_i, t_j)) \quad (1)$$

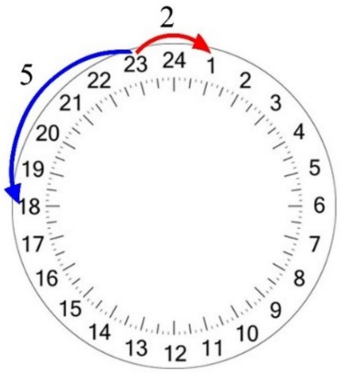


Figure 3. Description of distance based on time.

Theorem 1. $D(i,j)$ in Equation (1) is a distance metric. The Equation (1) is a distance metric if it is a non-negative function $D(i,j)$ and satisfies several properties based on the theorem of metrics as follow [16]:

- The non-negativity $D(i,i) \geq 0$,
- The identity property $D(i,i) = 0$,
- The symmetry property, $D(i,j) = D(j,i)$, and
- The triangle inequality, $D(i,j) + D(j,k) \geq D(i,k)$

$D(i,j)$ in Equation (1) is a non-negative function because no input combination can yield a negative result. $D(i,i) = \min(|t_i - t_i|, tr + \min(t_i, t_i) - \max(t_i, t_i)) = 0$. Therefore, D satisfies the identity property. As we use the absolute value and the minimum and maximum values between two

input variables, changing the order of the input will not yield different results. This indicates that D also satisfies the symmetry property. We now prove that D satisfies the triangle inequality as well. Assume that

- $D(i,j) = \min(|t_i - t_j|, tr + \min(t_i, t_j) - \max(t_i, t_j))$
- $D(j,k) = \min(|t_j - t_k|, tr + \min(t_j, t_k) - \max(t_j, t_k))$
- $D(i,k) = \min(|t_i - t_k|, tr + \min(t_i, t_k) - \max(t_i, t_k))$

Evidently, the above equation satisfies the triangularity based on Figure 4 below.

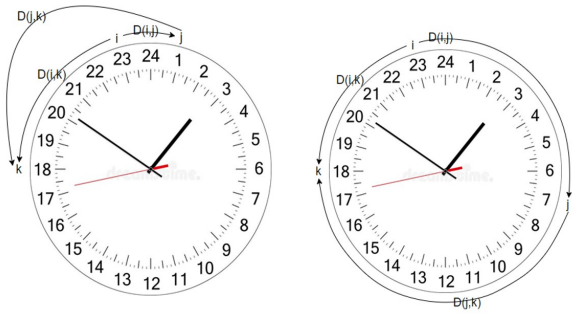


Figure 4. Illustration of the equation for satisfying the triangle inequality.

We use the distance function described in Equation (1) in the self-organizing map (SOM) and k-means algorithms to cluster the resources based on shift work. k-means is used because of its simplicity and its speed computation. SOM is used because then the number of clusters does not have to be determined. This algorithm can help realize dimensionality reduction, which can be used to determine the number of clusters automatically. k-means is more restricted in that the number of clusters needs to be defined. However, k-means can be used to determine the number of clusters automatically by incorporating the elbow method. The basic idea of SOM and k-means is to group the data by calculating the Euclidian distance between several attributes of data and the groups (weight in SOM or centroid in k-means). To measure the Euclidian distance correctly, the distance between the group and each attribute of data should be on the same scale to avoid bias toward a specific attribute. Thus, researchers generally normalize and standardize the attributes to ensure that all the attribute data are on the same scale. In this study, we use data normalization to [0, 1] using the equation given in [16]. The limitation of the normalization based on the equation in [16] is that, if we consider the input as the time value (without date), the result will not be useful, and an incorrect distance may be generated. The distance between two different time values is calculated by considering the threshold value, as shown in Equation (1). To cope with this problem, we modified the value for the time attribute and normalized it after the distance between two different time values was calculated. The equation is as follows, where $D(i,j)$ is the distance function from Equation (1), $\max(D(i,j))$ is the maximum distance among the data (it is similar to $x_{\max} - x_{\min}$, but the formula is different for time values), and $Dn(i,j)$ is the distance value after it is normalized. For example, if the data are 1 and 23, then $\max(D(i,j))$ is 2, but if the data are 1, 13, 23, then $\max(D(i,j))$ is 12. Here, the largest $\max(D(i,j))$ that could be obtained for any combination of the time value data is 12.

$$Dn(i,j) = \frac{D(i,j)}{\max(D(i,j))} \tag{2}$$

SOM is a neural-network-based clustering technique widely used for visualization and data reduction of high-dimensional data [17,18]. SOM comprises m neurons located at a low-dimensional map representation that are generally spread across a 2-D map in x and y coordinates [19]. The two common types of topologies generally used in SOM are rectangular and hexagonal topologies [19].

Each neuron i has a p-dimensional weight vector $w_i = (w_{i1}, w_{i2}, \cdots, w_{ip})$, where $i = 1, 2, 3, ..., m$, which has the same dimension as the input space.

If we have N input data with a p-dimensional attribute vector, then the following steps describe the modified SOM algorithm based on the conventional SOM algorithm [20]:

a. Set the maximum number of iterations, s.

b. Initialize the weight vectors w_i for each neuron.

c. Randomly select an input vector x_j among n data and check its distance from each weight $w_i(s)$ of neurons. We modified the Euclidian distance by separating the calculation of the data of each attribute based on its characteristics. The attribute data that need to be evaluated with time (time-dependent) are grouped together (start from $p = 1$ until $dmTm$), whereas the attribute data that do not need to be evaluated with time (not time-dependent) are grouped into different groups (start from $p = dmTm + 1$ until dm). Here, Dn_p and $X_{j,p}$ are the distance function normalized in Equation (2) and the input vector x_j for a specific attribute p based on the p-dimensional attribute vector of data, respectively. $W_{i,p(s)}$ is the weight vector w_i for a specific attribute p based on the p-dimensional attribute vector of data in a specific iteration s.

$$||x_j - w_i(s)|| = \sqrt{\sum_{p=1}^{dmTm} Dn_p(i,j)^2 + \sum_{p=dmTm+1}^{dm} (x_{j,p} - w_{i,p}(s))^2} \quad (3)$$

d. Determine the best matching unit (BMU) to find the winner neuron c by comparing the previously calculated Euclidian distances.

$$c = \text{argmin}_{1 \le i \le m}\{||x_j - w_i(s)||\}$$

e. Calculate the learning rate and neighborhood function. Several learning rates and neighborhood functions have been proposed in the literature. The learning rate $\alpha(s)$ and neighborhood function $\alpha(s)$ are functions that decrease with the iteration s. The symbol α_0 and $\sigma(s)$ indicate the initial learning rate and initial neighborhood function, respectively. In this study, we use the learning rate $\alpha(s)$ and neighborhood function $\sigma(s)$ based on [20,21] as follows:

$$\alpha(s) = \alpha_0 \cdot exp(-\frac{s}{s_max}), s = 1, 2, 3, \cdots$$

$$\sigma(s) = \sigma_0 \cdot exp(-\frac{s}{s_max}), s = 1, 2, 3, \cdots$$

f. Calculate the neighborhood distance weights $h_{c,i}(s)$ between the winner neuron c and every other node neuron i, where the *coor* mean coordinate of the neuron in the lattice or map is given as follows:

$$h_{c,i}(s) = exp(-\frac{||coor_c - coor_i||}{2 \cdot \sigma(s)^2})$$

g. Update the weight vectors of the neurons within the local neighborhood of the BMU (including the winner neuron c). We modified the updating weight mechanism as shown in Equation (4) to satisfy the time value constraint. $wn_{i,p}(s)$ in Equation (5) is the new weight of attribute p, where p is the non-time value at iteration s. $wntmp_i(s)$ in Equation (6) is the new temporary weight of the time value at iteration s. We use the term "temporary" because this value is not the final value of the new weight (we need to evaluate this value with the threshold first). $disp_{x_{j,p}, w_{i,p}}$ in Equation (7) is the displacement value between $x_{j,p}$ and $w_{i,p}(s)$. The terms displacement and distance are not the same. The displacement value can be negative, but the distance value is always positive.

$$w_{i,p}(s+1) = \begin{cases} wn_{i,p}(s), & p \text{ is not time-dependent} \\ wntmp_{i,p}(s) - tr, & p \text{ is time-dependent} \wedge wntmp_{i,p}(s) \geq tr \\ tr - wntmp_{i,p}(s), & p \text{ is time-dependent} \wedge wntmp_{i,p}(s) \leq 0 \end{cases} \quad (4)$$

$$wn_{i,p}(s) + \alpha(s) \cdot h_{c,i}(s) \cdot (x_{j,p} - w_{i,p}(s)) \quad (5)$$

$$wntmp_{i,p}(s) + w_{i,p}(s) + \alpha(t) \cdot h_{c,i}(s) \cdot disp_{x_p, w_{i,p}(s)} \quad (6)$$

$$disp_{x_{j,p}, w_{i,p}} = \begin{cases} x_{j,p} - w_{i,p}(s), & 0 \leq |x_{j,p} - w_{i,p}| \leq 12 \\ -D(x_{j,p}, w_{i,p}(s)), & x_{j,p} - w_{i,p} > 12 \\ D(x_{j,p}, w_{i,p}(s)), & x_{j,p} - w_{i,p} < -12 \end{cases} \quad (7)$$

h. Repeat steps c to g until the maximum number of iterations is reached or other stopping criteria are found.

We should first determine the number of neurons in the SOM to execute it. We determine the number of neurons in the SOM using Equation (8) based on a previous study [22]. The equation calculates the number of neurons from the observations in the training data, where M is the number of neurons, and N is the number of observations in the training data.

$$M \approx 5\sqrt{N} \quad (8)$$

To compare the effect of the modified distance function of time value for different algorithms, we also implement it using k-means clustering. k-means is a centroid-based partitioning technique that uses the centroid of a cluster, C_i, to represent that cluster [16]. Conceptually, the centroid of a cluster is its center point. The quality of cluster C_i can be measured using the within-cluster variation, which is the sum of squared errors (SSEs) between all the objects in C_i and the centroid c_i, defined as follows:

$$SSE = \sum_{i=1}^{k} \sum_{x \in C_i} (||x - c_i||)^2 \quad (9)$$

If the data consist of a dataset containing N observations and k is the number of clusters, the k-means algorithm operates based on the following procedure, where x^j is each datum contained in the dataset and c^j is the centroid label cluster for datum x^j:

a. Arbitrarily choose k objects from the data as the initial cluster centers.
b. Assign each object to the cluster to which the object is the most similar, based on the mean value (centroid) of the objects in the cluster. We use the same distance function described in Equation (3), but the term weight (w) is changed to mean (μ).

$$c^j = \text{argmin}_i \{||x^j - \mu_i||\}$$

c. Update the cluster mean—that is, calculate the mean value of the objects for each cluster. For each i cluster, we modified the cluster mean updating by incorporating an incremental average. The incremental average is used because, by averaging the time value, we could not compute the mean directly for all the values. We must evaluate the average value one by one for each element, as shown in Equations (10) and (11), where A_n refers to the nth element among the elements that will be averaged.

$$\mu_{i,p} = \begin{cases} \frac{\sum_{j=1}^{n} 1\{c^j=i\} \cdot x_p^j}{\sum_{j=1}^{n} 1\{c^j=i\}}, & p \text{ is not time-dependent} \\ \mu inc_{i,p} - tr, & p \text{ is time-dependent} \wedge \mu inc_{i,p}(s) \geq tr \\ tr - \mu inc_{i,p}, & p \text{ is time-dependent} \wedge \mu inc_{i,p}(s) \leq 0 \end{cases} \quad (10)$$

$$disp_{x_{j,p}, w_{i,p}} = \begin{cases} A_{n-1} + \frac{x_p^n - A_{n-1}}{n}, & 0 \leq |x_p^n - A_{n-1}| \leq 12 \\ A_{n-1} - \frac{D(x_p^n - A_{n-1})}{n}, & x_p^n - A_{n-1} > 12 \\ A_{n-1} + \frac{D(x_p^n - A_{n-1})}{n}, & x_p^n - A_{n-1} < -12 \end{cases} \quad (11)$$

d. Repeat steps b to c until the stopping criteria are found (for example, the cluster member is not changed).

Several methods to define the number of clusters in *k*-means exist, such as the elbow method, gap statistic, silhouette coefficient, canopy, Akaike's information criterion, and the Monte Carlo technique [23,24]. As mentioned previously, we employed the elbow method to determine the number of cluster *k* in *k*-means clustering. Generally, the elbow method is based on the subjective visualization of the user. However, in this study, we performed the elbow method without subjective visualization. The elbow method used in this study is based on the method described in [24], using the square of the distance between the sample points in each cluster and the centroid of the cluster. In our case, the square of the distance between the sample points in each cluster and the centroid of the cluster is based on the SSE formula in Equation (9). Our program will monitor the SSE value for each increment of the *k* cluster and detect the place at which a rapid decline occurs, similar to the paper [25], but by using Heron's formula.

The shift work generated based on our approach is conducted based on the clustering method. There is also interesting research conducted by other researchers that tried to mine the resource availability by considering the temporal perspective [26]. Our approach is different from the existing method in that our method is using not only specific on the single resource but aggregating the resource shift work using clustering. Our resource focuses on humans who usually work based on shift time. We are using a clustering method, so by incorporating the shift mining, the resource that only recorded a short time in the event log because of the limited job assigned to him/her will be automatically grouped into the quite similar cluster. Nakatumba [12] and Martin [26] are not using this kind of mechanism but focus on every single resource, and Nakatumba and Martin also use a statistical analysis approached, whereas we use a machine learning technique using a clustering algorithm. We also represent the proposed representation activity that can incorporate the shift work rule in the simulation model, whereas Nakatumba approached used resource availability percentage.

4. Incorporating Shift Work in the CPN Model

We modified a part of the CPN model used by Rozinat in the study [4] to incorporate shift work based on a predefined rule or the result of the clustering event logs in the CPN model. The part of the CPN model modified in this study is the activity subpage, which represents the execution of activity by resources [4]. In this representation, the resource is always assumed to be available if no activity is performed.

We modified the above CPN model to incorporate the shift work rule by adding two more places and activities. The two activities and places operating based on the shift work rule are incorporated into the example CPN model, as shown in Figure 5. The two additional places are used to store the data of the starting and completion of shift work and the current availability time of the resource based on the simulation time unit. The two additional activities are used to update the state of the token at the place containing the current availability time of the resource explained previously.

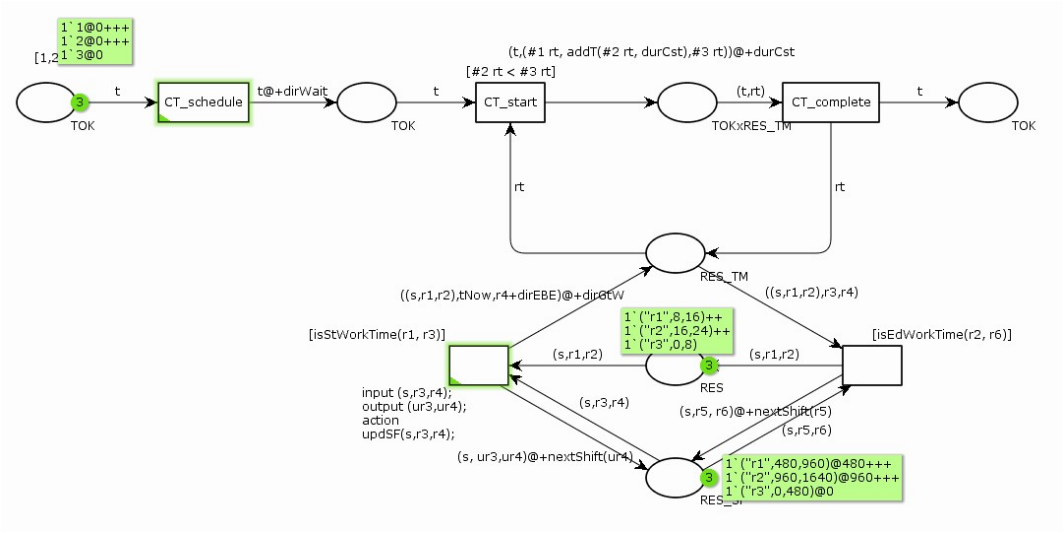
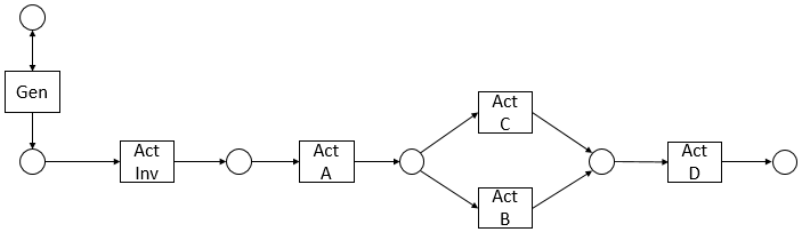


Figure 5. Part of the CPN model that incorporates the shift work rule.

5. Artificial Event Logs with Shift Work Rule

An artificial event log generator is created to generate artificial event logs based on the predefined shift work rule of a resource. We use the shift work rule based on two scenarios. Scenario 1 consists of two shifts that are divided into 00:00–12:00 and 12:00–24:00. Scenario 2 consists of three shifts that are divided into 00:00–08:00, 08:00–16:00, and 16:00–24:00. After defining this rule, we create a simple simulation model using this shift work rule and adding a token generator that will generate a token. In this simulation model, we assume that each day token arrives in a batch at the beginning of the day. We use this assumption to show the benefit of using shift work mining in the simulation model by analyzing the flow time of the token in the system. The simulation model can be described based on Petri net representation from Figure 6 below (we simplified the CPN representation into Petri nets just to showing the flow of the process). Each transition in the Petri net representation (except the Gen transition) is a high-level activity that can be breakdown into a low-level representation of activity, as shown in Figure 5 above. Gen transition is a transition that is used as a token generator. Several parameters used in the simulation model, including the token generator mechanism, is shown in Figure 7. We used the assumption that for each day, the number of incoming tokens is fixed (we set 90 tokens per day), and the simulation stops after 270 tokens are generated (three days of operation). We set this stopping criterion as a guard in the Gen transition.




```
String MATH_ROUND_GEN = "if(b.tid.get % 90 == 0) { "
        + "(b.tid.get/90)*3600*24-cgraph.getGlobalTime().getTime() } "
        + "else Math.round(Gaussian(60, 10).draw())";
String MATH_ROUND_INV = "Math.round(Gaussian(60, 10).draw())";
String MATH_ROUND_A = "Math.round(Gaussian(864, 10).draw())";
String MATH_ROUND_B = "Math.round(Gaussian(864, 10).draw())";
String MATH_ROUND_C = "Math.round(Gaussian(864, 10).draw())";
String MATH_ROUND_D = "Math.round(Gaussian(864, 10).draw())";
String MATH_WAITING = "Math.round(Gaussian(300, 50).draw())";
```

Figure 6. Based model to generate an artificial event log.

For scenario 1, we put one resource in Act Inv and two resources in each remaining activities (each resource in the remaining activities is assigned based on predefined shift time). For scenario 2, we put one resource in Act Inv and three resources in each remaining activities (each resource in the remaining activities is assigned for each shift time). The resource allocation for each activity explained above can be described in Figure 7 below.

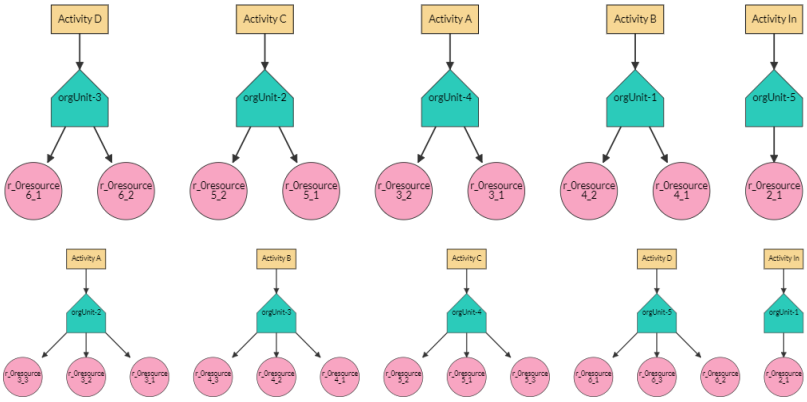


Figure 7. Resource allocation of each activity for scenario 1 and scenario 2.

6. Results

As explained previously, we used an artificial data set generated based on simulation using the predefined rule of shift work of resources. The rule is embedded in the CPN model, as shown in Figure 6. From this CPN model representation, we create a matching model in the Scala programming language for generating the output more flexibly. Using this model, we performed the simulation based on a three-day time window, as explained in Section 5, and obtained the result by marking a token that transformed into a comma-separated values (CSV) event log containing life cycle information (start and completion times). In the token generator explained in the previous section, we set the number of cases arriving each day to 90. These 90 cases arrive at the beginning of the day, and they will be processed later by the resources. This condition is handled by the arc function `MATH_ROUND_GEN` in the activity Gen, as shown in Figure 7.

Then, the CSV event log is uploaded into the database and later used as an input to resource mining. The log data already existing in the database will be preprocessed based on the algorithm described in Figure 1 of Section 3. The screenshot of the data after preprocessing is shown in Figure 8.

caseid	activity	resource	start	completion
1	Activity In	r_0resource2_1	2020-01-01 00:05:36	2020-01-01 00:06:36
2	Activity In	r_0resource2_1	2020-01-01 00:07:00	2020-01-01 00:07:55
3	Activity In	r_0resource2_1	2020-01-01 00:07:55	2020-01-01 00:08:52
5	Activity In	r_0resource2_1	2020-01-01 00:08:52	2020-01-01 00:09:31
4	Activity In	r_0resource2_1	2020-01-01 00:09:31	2020-01-01 00:10:35
6	Activity In	r_0resource2_1	2020-01-01 00:10:58	2020-01-01 00:11:54
7	Activity In	r_0resource2_1	2020-01-01 00:12:07	2020-01-01 00:12:59
2	Activity A	r_0resource3_1	2020-01-01 00:12:32	2020-01-01 00:26:54
8	Activity In	r_0resource2_1	2020-01-01 00:12:59	2020-01-01 00:14:14
9	Activity In	r_0resource2_1	2020-01-01 00:14:14	2020-01-01 00:15:15
10	Activity In	r_0resource2_1	2020-01-01 00:15:15	2020-01-01 00:16:14
13	Activity In	r_0resource2_1	2020-01-01 00:16:14	2020-01-01 00:17:10
11	Activity In	r_0resource2_1	2020-01-01 00:17:10	2020-01-01 00:18:16
12	Activity In	r_0resource2_1	2020-01-01 00:18:32	2020-01-01 00:19:27
15	Activity In	r_0resource2_1	2020-01-01 00:19:27	2020-01-01 00:20:26

start	completion	resource	gap	gapdt
2020-01-02 19:10:35	2020-01-02 18:56:03	r_0resource3_2	14 minute	0:14
2020-01-02 19:25:04	2020-01-02 19:10:35	r_0resource3_2	14 minute	0:14
2020-01-02 19:39:22	2020-01-02 19:25:04	r_0resource3_2	14 minute	0:14
2020-01-02 19:53:39	2020-01-02 19:39:22	r_0resource3_2	14 minute	0:14
2020-01-02 20:08:07	2020-01-02 19:53:39	r_0resource3_2	14 minute	0:14
2020-01-02 20:22:21	2020-01-02 20:08:07	r_0resource3_2	14 minute	0:14
2020-01-02 20:36:51	2020-01-02 20:22:21	r_0resource3_2	14 minute	0:14
2020-01-02 20:50:59	2020-01-02 20:36:51	r_0resource3_2	14 minute	0:14
2020-01-02 21:05:36	2020-01-02 20:50:59	r_0resource3_2	14 minute	0:14
2020-01-03 00:12:09	2020-01-02 21:05:36	r_0resource3_2	186 minute	3:06
2020-01-03 12:00:00	2020-01-03 00:12:09	r_0resource3_2	707 minute	11:47
2020-01-03 12:14:28	2020-01-03 12:00:00	r_0resource3_2	14 minute	0:14
2020-01-03 12:28:49	2020-01-03 12:14:28	r_0resource3_2	14 minute	0:14
2020-01-03 12:43:20	2020-01-03 12:28:49	r_0resource3_2	14 minute	0:14
2020-01-03 12:57:46	2020-01-03 12:43:20	r_0resource3_2	14 minute	0:14

Figure 8. Preprocessing the event logs of our resource mining algorithm.

We conducted the *k*-means clustering experiment using the elbow method. As explained in the previous section, the elbow method automatically detects the best number of clusters k without subjective visualization. This step involves the mechanism to determine the best number of clusters, k, based on the data shown in Figure 7. First, we set the value of k_{max} to 8. The k_{max} in this experiment is the maximum number of k that will be analyzed. We set the value of k_{max} to 8 because the number of

different operational shifts in a day is generally relatively small. For each k, we evaluate the SSE based on Equation (13). The SSEs per k are listed in Table 1 below for scenario 1 and scenario 2.

Table 1. Sum of Squared Errors of the data per k in k-means algorithm.

k	SSE of Scenario 1	SSE of Scenario 2
$k = 1$	2736.342	3863.116
$k = 2$	479.879	1452.482
$k = 3$	51.433	137.117
$k = 4$	51.433	38.089
$k = 5$	51.433	38.089
$k = 6$	51.433	38.089
$k = 7$	51.433	38.089
$k = 8$	51.433	38.089

Based on this SSE, we can determine the furthest line marked with the red line in Figure 9 to decide the best value of k. The furthest line can be determined easily using Heron's formula if the information of each SSE in each k is known.

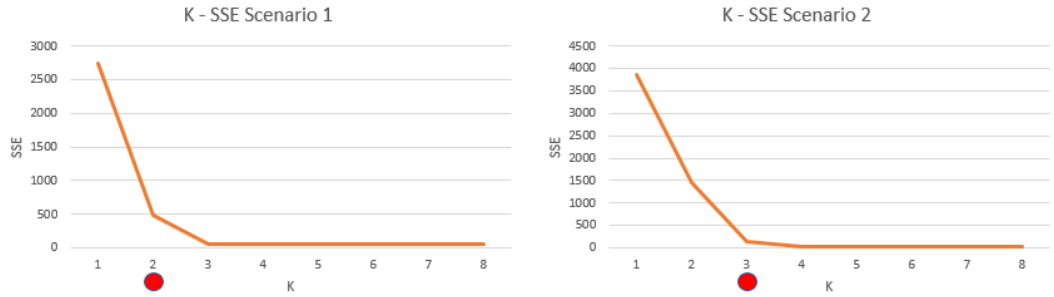


Figure 9. k to SSE graph visualization.

We conducted an experiment with both algorithms, SOM and k-means, using the data described above to compare their results when mining the shift work. Implementing the new distance function in the k-means algorithm always yields better results than those of the SOM. Based on the experiment, k-means using the new distance always divided the cluster into two groups for scenario 1 and three groups for scenario 2, which matched the baseline rule used in the simulation model. SOM tends to divide clusters into more than three groups. The experiment showed that SOM divided the cluster into six groups for scenario 1 and seven groups for scenario 2 using the number of neuron settings based on Equation (8), which was far from the expected result. Figures 10 and 11 are the results of k-means without the new distance function and k-means with the new distance function, respectively. The left side of Figure 10 shows that the k-means generate three groups of resources using the data from scenario 1, and the right side of Figure 11 shown that the k-means also generate three groups of resources using the data from scenario 2. Each cluster is separated by an empty space in the horizontal axis. Different colors in the chart represent different resources. Each vertical line in the chart that spreads along the horizontal axis is represented the object data explained in Algorithm 1. The start time of object data and completion time of object data is represented by the bottom and top side of each vertical line. The vertical axis on the left side of the chart visualizes the time information that is separated into the current day and the next day time. We use current day and next day time to handle visualization of data that can have the start time bigger than the completion time because we do not consider the date (for example, the start time of object data is 21, but the completion time is 9; this is like the representation of the shift time work that started at 9 p.m. and ended at 9 a.m.). We can see in the left side of Figure 10 that although the start time of several object data is the same (hour is 12), if there is any completion time of object data is below and above h hour 00/24 on the next day, those

data will be clustered into a different group. This happened because the distance between time above hour 24 and time below hour 24 is far enough if calculated using a normal distance function.

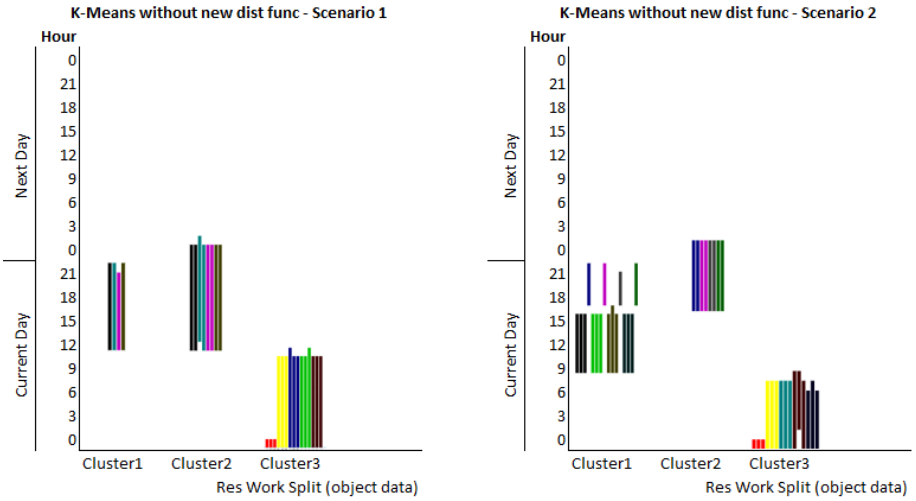


Figure 10. The comparison of clustering *k*-means without the new distance function.

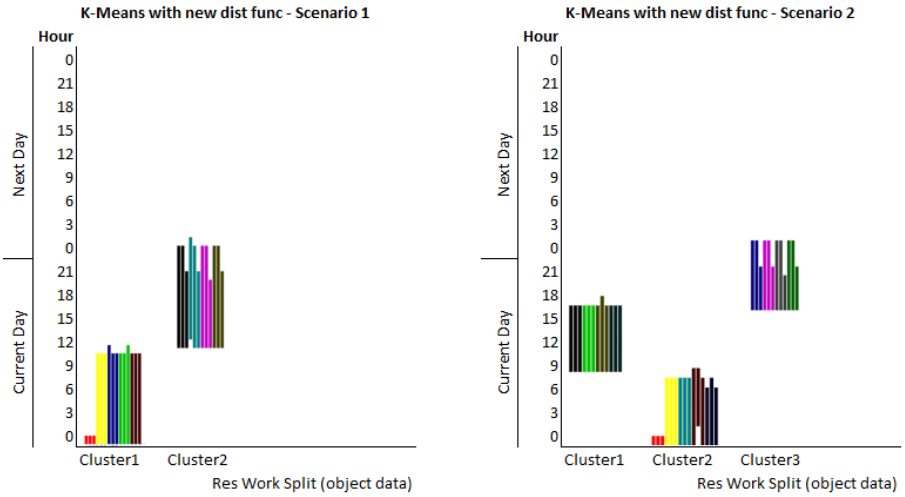


Figure 11. The comparison of clustering by incorporating the new distance function.

The previous simulation model to generate the event log is used again (all of the parameters are kept), but this time, the rule of predefined resource shift work is removed from the simulation model. We later used the minimum and the maximum time in each cluster generated from the clustering algorithm as the start time and end time of the shift work, respectively. This information of resource shift work from the clustering algorithm will then be embedded in the simulation model. The other simulation is run using the same simulation model but without any rule of shift work embedded inside. To compare the effectiveness of this approached to generate a better simulation model, we used basic Key Performance Indicator (KPI) that is usually used to measure the simulation accuracy of the process, which is flow time and throughput. Evidently, the KPI generated from the simulation model that incorporates shift work from the clustering algorithm fits better with the KPI from the baseline event log. If the shift work is not incorporated into the simulation model, all the resources are assumed to be available since the start of the simulation, thus leading to generation of inaccurate KPIs. This indicates that all the resources can perform work at the same time, which will tend to make the average flow time cases to finish shorter. The flow time used here is measured by calculating the duration of each from the generating time to finishing time in the process. The throughput is calculated by dividing the

number of tokens finished by the time of the simulation. The comparison of simulation performance results using the configuration of scenarios 1 and 2 is shown in Tables 2 and 3, respectively.

Table 2. Performance of the model when using shift work vs. not using shift work of nine resources.

KPI	Shift Work	No Shift Work	Baseline
Average flow time	11.1552 h	5.64 h	10.882 h
Throughput	3.80/h	4.525/h	3.844/h

Table 3. Performance of the model when using shift work vs. not using shift work of 13 resources.

KPI	Shift Work	No Shift Work	Baseline
Average flow time	11.258 h	3.83 h	10.831 day
Throughput	3.79/h	4.813/h	3.85/h

7. Discussion

In this study, we attempted to mine the shift work information from event logs. We then incorporated this information into the simulation model to obtain results that represented the operational event log more accurately. Based on our experiment, implementing this method could improve the results of the simulation model. However, several limitations should be noted, as the clustering algorithm in this article assumes that the data are separable. If the event log data about the shift work of the resources are not quite separable, obtaining good results could be difficult. We observed that the result from the SOM algorithm tends to generate too many clusters, even after the new distance function is incorporated.

A more thorough analysis of the result indicated that this phenomenon occurs because of the formula for determining the size of the neuron map in Equation (8) and the method for initializing the weight of neurons at the beginning of execution. The size of the data determines the size of the neuron map in Equation (8). However, if the number of data is small—for example, less than 25—a neuron map larger than the size of the data themselves will be generated.

Applying this method for generating a more accurate simulation model can be done by combining process model discovery, decision point analysis, organizational miner, and our proposed method. The result of our proposed method is to mine the shift work for the resources. We later can use the mined result of shift work resources from the clustering algorithm to be embedded in the simulation model.

Basically, to apply the proposed method in the real world, we first should find any event log with human resource information that exists in the event. Using that event log, we can generate the simulation model. The step to generate the simulation model is started by the process discovery algorithm. The process model from the process discovery algorithm will be used as a baseline of the process for the simulation. After this step, we mined the organizational miner to allocate resources into activities. Later, we performed the decision point analysis using input from the process model and the event log. Finally, we use shift work mining to get information about shift time of the resources. We then put this information as a rule for the resource in the simulation model. This shift work rule will add the constraint of the resource in the simulation model. By adding more constraints in the simulation model that reflect the real constraints that happened in reality, we can generate a more accurate simulation model. However, adding more constraints in the simulation model could make the running time of the simulation become longer.

There are several possibilities for future work from this research that can be conducted. Combining the proposed method with another result of the resource mining approached, for example, resource percentage availability explained by Nakatumba [12], could possibly be useful. It is also possible to improve the currently proposed method by combining it with another clustering algorithm or another distance function to handle inseparable data explained previously in the limitation part of

this section. Another interesting possible future work from this research is how to perform automatic optimization of resource allocation and execution based on event log automatic generated simulation.

8. Conclusions

The experiment conducted in this study indicated that shift work could be mined from an event log. Incorporating the shift work information into the simulation model could improve the simulation results that are more similar to reality. So, in this case, we can generate a more accurate simulation model. Limitations that should be noted include the clustering algorithm in this research study assuming that the data are separable. If the event log data about the shift work of the resources are not quite separable, obtaining good results could be difficult.

Author Contributions: Conceptualization, N.I.U. and R.A.S.; methodology, N.I.U.; software, N.I.U. and R.A.S.; validation, H.B.; writing—original draft preparation, N.I.U.; writing—review and editing, I.M.K., R.A.S., Y.-J.P., and H.B.; visualization, N.I.U.; supervision, H.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the MSIT(Ministry of Science and ICT), Korea, under the Grand Information Technology Research Center support program(IITP-2020-2016-0-00318) supervised by the IITP(Institute for Information & communications Technology Planning & Evaluation).

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

References

1. Robinson, S. Discrete-event simulation: From the pioneers to the present, what next? *J. Oper. Res. Soc.* **2005**, *56*, 619–629. [CrossRef]
2. Van der Aalst, W.M.P. *Process Mining: Data Science in Action*, 2nd ed.; Springer: Berlin/Heidelberg, Germany, 2016. [CrossRef]
3. Van der Aalst, W.M.P. Process Mining and Simulation: A Match Made in Heaven! In Proceedings of the 50th Computer Simulation Conference, SummerSim '18, Bordeaux, France, 9–12 July 2018.
4. Rozinat, A.; Mans, R.S.; Song, M.; van der Aalst, W.M.P. Discovering Simulation Models. *Inf. Syst.* **2009**, *34*, 305–327. [CrossRef]
5. Van der Aalst, W.; Weijters, T.; Maruster, L. Workflow mining: discovering process models from event logs. *IEEE Trans. Knowl. Data Eng.* **2004**, *16*, 1128–1142. [CrossRef]
6. Rozinat, A.; van der Aalst, W.M.P. Decision Mining in ProM. In *Business Process Management*; Dustdar, S., Fiadeiro, J.L., Sheth, A.P., Eds.; Springer: Berlin/Heidelberg, Germany, 2006; pp. 420–425.
7. Song, M.; van der Aalst, W.M.P. Towards comprehensive support for organizational mining. *Decis. Support Syst.* **2008**, *46*, 300–317. [CrossRef]
8. Kamal, I.M.; Bae, H.; Utama, N.I.; Yulim, C. Data pixelization for predicting completion time of events. *Neurocomputing* **2020**, *374*, 64–76. [CrossRef]
9. Ahmadon, M.A.B.; Yamaguchi, S. Verification Method for Accumulative Event Relation of Message Passing Behavior with Process Tree for IoT Systems. *Information* **2020**, *11*, 232. [CrossRef]
10. Silva, L.J.S.; Campagnolo, L.Q.; Fiol-González, S.; Rodrigues, A.M.B.; Schardong, G.G.; França, R.; Lana, M.; Barbosa, S.D.J.; Poggi, M.; Lopes, H. Visual Support to Filtering Cases for Process Discovery. In Proceedings of the 20th International Conference on Enterprise Information Systems, ICEIS 2018, Funchal, Madeira, Portugal, 21–24 March 2018; Hammoudi, S., Smialek, M., Camp, O., Filipe, J., Eds.; SciTePress: Setúbal, Portugal, 2018; Volume 1, pp. 38–49. [CrossRef]
11. Nakatumba, J.; van der Aalst, W.M.P. Analyzing Resource Behavior Using Process Mining. In *Business Process Management Workshops*; Rinderle-Ma, S., Sadiq, S., Leymann, F., Eds.; Springer: Berlin/Heidelberg, Germany, 2010; pp. 69–80.
12. Nakatumba, J. Resource-Aware Business Process Management: Analysis and Support. Ph.D. Thesis, Department of Mathematics and Computer Science, Eindhoven, The Netherlands, 2013. [CrossRef]
13. Jensen, K.; Kristensen, L.M. *Coloured Petri Nets: Modelling and Validation of Concurrent Systems*, 1st ed.; Springer: Berlin/Heidelberg, Germany, 2009.

14. International Labour Organization. *Working Time in the Twenty-First Century*. Available online: https://www.ilo.org/wcmsp5/groups/public/---ed_protect/---protrav/---travail/documents/publication/wcms_161734.pdf (accessed on 14 October 2020).
15. International Labour Organization. *Q+A Working Hours*. Available online: https://www.ilo.org/dyn/travail/docs/2576/q-aworking-hours (accessed on 14 October 2020).
16. Janeja, V.P.; Atluri, V.; Vaidya, J.; Adam, N.R. Collusion Set Detection through Outlier Discovery. In Proceedings of the 2005 IEEE International Conference on Intelligence and Security Informatics, Atlanta, GA, USA, 19–20 May 2005; Springer: Berlin/Heidelberg, Germany, 2005; pp. 1–13. [CrossRef]
17. Han, J.; Kamber, M.; Pei, J. *Data Mining: Concepts and Techniques*, 3rd ed.; Morgan Kaufmann Series in Data Management Systems; Morgan Kaufmann: Amsterdam, The Netherlands, 2011.
18. Kohonen, T. *Self-Organizing Maps*, 3rd ed.; Springer series in information sciences, 30; Springer: Berlin, Germany, 2001.
19. Jin, C.H.; Pok, G.; Lee, Y.; Park, H.W.; Kim, K.D.; Yun, U.; Ryu, K.H. A SOM clustering pattern sequence-based next symbol prediction method for day-ahead direct electricity load and price forecasting. *Energy Convers. Manag.* **2015**, *90*, 84–92. [CrossRef]
20. Chaudhary, V.; Bhatia, R.; Ahlawat, A.K. A novel Self-Organizing Map (SOM) learning algorithm with nearest and farthest neurons. *Alex. Eng. J.* **2014**, *53*, 827–831. [CrossRef]
21. Riese, F.M.; Keller, S.; Hinz, S. Supervised and Semi-Supervised Self-Organizing Maps for Regression and Classification Focusing on Hyperspectral Data. *Remote Sens.* **2020**, *12*, 7. [CrossRef]
22. Tian, J.; Azarian, M.H.; Pecht, M. Anomaly Detection Using Self-Organizing Maps-Based K-Nearest Neighbor Algorithm. In Proceedings of the European Conference of the Prognostics and Health Management Society, Nantes, France, 8–10 July 2014.
23. Pham, D.T.; Dimov, S.S.; Nguyen, C.D. Selection of K in K-means clustering. *Proc. Inst. Mech. Eng. Part J. Mech. Eng. Sci.* **2005**, *219*, 103–119. [CrossRef]
24. Yuan, C.; Yang, H. Research on K-Value Selection Method of K-Means Clustering Algorithm. *Multidiscip. Sci. J.* **2019**, *2*, 226–235. [CrossRef]
25. Satopaa, V.; Albrecht, J.; Irwin, D.; Raghavan, B. Finding a "Kneedle" in a Haystack: Detecting Knee Points in System Behavior. In Proceedings of the 2011 31st International Conference on Distributed Computing Systems Workshops, Minneapolis, MN, USA, 20–24 June 2011; pp. 166–171.
26. Martin, N.; Depaire, B.; Caris, A.; Schepers, D. Retrieving the resource availability calendars of a process from an event log. *Inf. Syst.* **2020**, *88*, 101463. [CrossRef]