ISSN 1870-4069

Identificación de procesos de eventos discretos cíclicos temporizados

Marina Montes-Partida, Ernesto López-Mellado

Centro de Investigación y de Estudios Avanzados,
Unidad Guadalajara,
México

{marina.montes, e.lopez}@cinvestav.mx

Resumen. Esta investigación se inscribe en el contexto de la minería de procesos; en particular, en el descubrimiento de procesos de eventos discretos, donde los modelos obtenidos, expresados mediante redes de Petri temporizadas (RPT), son obtenidos a partir de una secuencia S de eventos fechados. El presente artículo propone un método de descubrimiento de procesos temporizados, donde los modelos obtenidos son RPT. La temporización de las transiciones es determinística, la cual se expresa mediante dos parámetros: duración media e intervalo de los tiempos de disparo. La propuesta extiende un método de descubrimiento básico mediante de una técnica de refinamiento estructural y temporal a la RPT *N* descubierta para mejorar su precisión. Se analiza la distribución de las duraciones calculadas en cada transición para determinar si hay más de un grupo (clúster) de duraciones alrededor de un valor; en tal caso se refina la transición en dos o más transiciones según el número de clústeres. Posteriormente, las nuevas transiciones T' se reemplazan en S de acuerdo a su clúster de duraciones y se determina su patrón de ocurrencia; éste se expresa como una red de Petri N' con nuevos lugares, la cual se fusiona con **N** usando T' para obtener la RPT refinada.

Palabras clave: Minería de procesos, descubrimiento de procesos temporizados, refinamiento estructural y temporal, redes de Petri temporizadas.

Discovering Timed Cyclic Discrete Event Processes

Abstract. This research is in the scope of process mining, focusing on discrete event process discovery, where the discovered models are expressed by timed Petri nets (TPN) from a dated event sequence S. This paper presents a timed process discovery method that builds transition-TPN, where the timing is expressed in two deterministic parameters: average and interval firing durations. This proposal extends a former discovery method through a structural and temporal refining technique applied to a discovered TPN *N* to improve its precision. For each transition, the distribution of computed firing delays is analyzed to determine if there is more than one durations cluster; in such a case, the transition is refined into two or more new transitions according to the number of clusters. Afterwards, the set of new transitions T' are replaced in S; then, their occurrence pattern is synthesized as a PN *N'* using new places; the *N'* is merged with N through *T'* to obtain the refined TPN.

Keywords: Process mining, timed process discovery, structural and temporal refining, timed Petri nets.

1. Introducción

La identificación o descubrimiento de procesos es una forma de extracción de conocimiento sobre un proceso, donde el comportamiento observado, en la forma de secuencias de ejecución de eventos, tareas, o actividades, es representado por modelos formales.

Procesos de eventos discretos. En el caso de procesos de eventos discretos, los formalismos describen relaciones estado-eventos; los más usados son los autómatas finitos (AF) y las redes de Petri (RP). En el ámbito de los procesos de flujo de trabajo, las secuencias que describen el comportamiento son capturadas desde el inicio hasta el fin de una ejecución (cases); éstas son en gran cantidad.

En cambio, en el ámbito de los procesos industriales, las secuencias son pocas y muy largas; la ejecución de los procesos (jobs) se registra de manera continua y no se conoce la delimitación de dichos procesos. En ambos contextos los eventos, representados por símbolos, pueden tener información adicional relativa a la actividad, los recursos asignados y el instante de ejecución.

Un tipo de procesos de eventos discretos, es el de los sistemas de manufactura automatizados. Éstos, en general son estructurados como un sistema compuesto de un controlador y una planta interactuando en un ciclo cerrado, los cuales intercambian señales para llevar a cabo la ejecución de las tareas; el controlador envía comandos y la planta informa su estado a través de las señales de sensores.

Identificación/descubrimiento. El comportamiento del proceso puede ser registrado por la observación de las señales intercambiadas cada vez que ocurra un cambio en ellas. A partir de estas observaciones se genera una secuencia de vectores de entrada/salida, las cuales son la entrada a un sistema de descubrimiento o identificación; el sistema convierte la secuencia de señales a una secuencia de eventos la cual se procesa para construir un modelo que expresa el comportamiento del proceso desde el punto de vista del controlador. Este enfoque, ilustrado en la Figura 1, ha sido abordado en [1, 2, 3, 4]. El método de identificación se presenta en dos pasos.

En el primer paso se descubre el comportamiento observable generando los eventos de entrada asociados a las transiciones que producen cambios en los lugares asociados a las salidas (marcado de la red de Petri). Posteriormente, estas transiciones forman una secuencia de eventos S que replican el comportamiento funcionamiento del sistema.

El segundo paso, presentado por [3, 5] descubre el comportamiento no observable a partir de la secuencia S. Primero se determinan las relaciones causales y concurrentes observadas en los eventos, es decir, en el disparo de cada transición en S.

Finalmente, los modelos observable y no observable son fusionados para conseguir el modelo final. En estos métodos el modelo construido no contiene información relativa a las duraciones de las tareas o eventos.

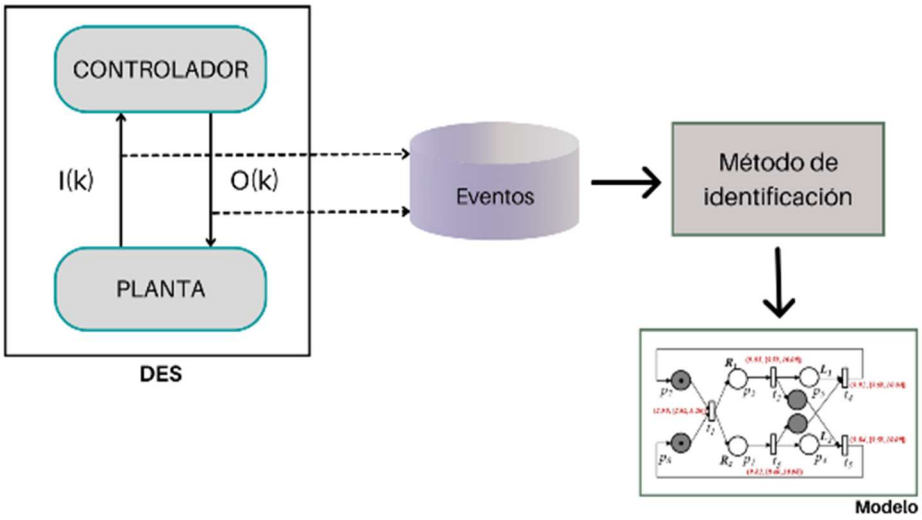

Fig. 1. Proceso de identificación para un DES.

Modelos temporizados. De acuerdo con [6], el tiempo en una red de Petri puede ser asociado con la duración de una operación o con el tiempo esperado previo a la ocurrencia de un evento. Trabajos relevantes sobre este tema se presentan en [7] donde se aborda la identificación del sistema basado en programación lineal entera, y en [8] donde se propone un algoritmo de aprendizaje que calcula RP ordinarias de acuerdo con la medición de flujos de salida cíclicos.

Propuesta. Este artículo se enfoca en el aspecto del tiempo; los modelos a descubrir capturan esa información con RP temporizadas (RPT) en las transiciones. Se presenta un método de identificación/descubrimiento de procesos temporizados donde el modelo obtenido es una RPT. La propuesta extiende un método previo [4] mediante de una técnica de refinamiento a una primera RPT identificada, la cual puede agregar nuevas transiciones con temporización más precisa.

Organización. El artículo está organizado como sigue. La sección 2 presenta los conceptos básicos de RP y el planteamiento del problema. La sección 3 describe la primera parte del método. La sección 4 presenta la fase de refinamiento. La Sección 5 ilustra la propuesta mediante un caso de estudio de tipo académico.

2. Preliminares y planteamiento del problema

2.1. Redes de Petri

Definición 1. (Red de Petri). Una estructura de red de Petri ordinaria G es un grafo bipartita representado por la tupla $G = (P, T, F)$ donde: $P = \{p_1, p_2, \dots, p_{|P|}\}$ y $T = \{t_1, t_2, \dots, t_{|T|}\}$ son conjuntos finitos de vértices denominados lugares y transiciones respectivamente; $F \subseteq P \times T \cup T \times P$ es una relación que representa los arcos que van de lugares a transiciones y viceversa. [9].

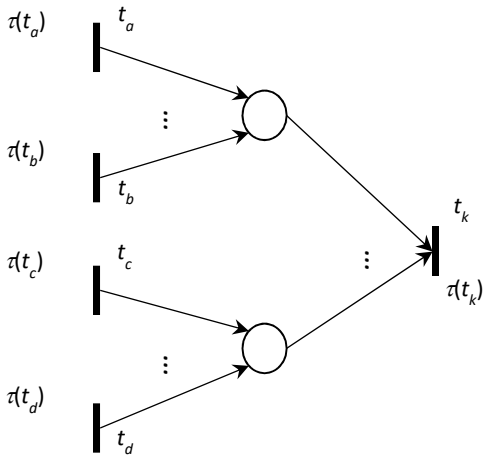

Fig. 2. Posibles transiciones precedentes a t_k.

Definición 2. (Red de Petri temporizada). Una transición temporizada es la tupla $N_\delta = (G, M_0, Tim)$, donde G es una PN ordinaria, M_0 es el marcado inicial; Tim es una función que asocia el tiempo a las transiciones; ésta puede ser de dos tipos:

- $\delta: T \rightarrow \Re^{\geq 0}$: una función que asigna un valor no negativo a cada $t_j \in T$; tal valor representa el tiempo de disparo de t_j contado a partir de su habilitación [10].
- $\iota: T \rightarrow \Re^{\geq 0} \times \Re^{\geq 0}$: una función de intervalo (ventana) de tiempo $[l_j, u_j]$ de disparo de la transición $t_j \in T$; t_j debe ser disparada después de l_j o antes de u_j unidades de tiempo contadas a partir del instante en que se habilita t_j [11].

Cuando $\delta(tj) = 0$ ó $\iota(tj) = [0, 0]$, t_j se dispara en cuanto está habilitada; a t_j se le llama transición inmediata. Las transiciones temporizadas e inmediatas suelen representarse por rectángulos vacíos y llenos, respectivamente.

Definición 3. (Registro de eventos temporizado). Un Registro de eventos temporizado (fechado) $\lambda_t = \{\sigma_{ti}\}$ es un conjunto de trazas $\lambda_t = \{\sigma_{ti} | \sigma_{ti} = (A_1, d_1) \ldots (A_i, d_i) \ldots (A_K, d_K)\}$, donde $A_j \in \boldsymbol{\Sigma}$ representa la tarea en la posición j y $d_j \in \Re^{\geq 0}$ es el instante en que se registra A_j. Cada traza se registra desde el tiempo $\tau = 0$.

2.2. Planteamiento del problema

El problema de identificación de procesos de eventos discretos temporizados se puede enunciar como sigue.

Definición 4. Sea λ_t un registro de eventos temporizado generado por un proceso de eventos discretos funcionando normalmente. El problema de identificación consiste en obtener una RPT la cual describa el comportamiento del proceso registrado en λ_t.

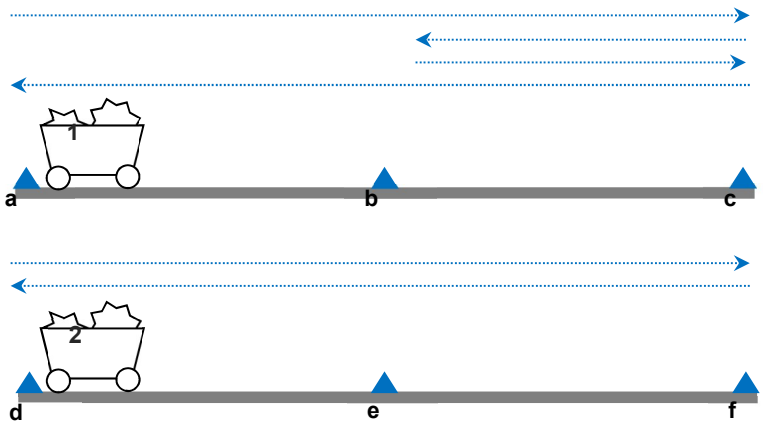

Fig. 3. Representación de proceso a ser modelado.

En [4] se presentó un método de identificación que aborda el problema anterior, el cual propone una estrategia en dos etapas. En la primera etapa, se procesa λ_t sin tomar en cuenta las fechas para obtener una RP cíclica no temporizada *N*.

En la segunda etapa, se determina la temporización de las transiciones; para ello se ejecuta λ_t en *N* determinando las duraciones de cada t_j cada vez que se dispara en la secuencia. Al final se tiene un conjunto de duraciones calculadas para cada transición; estos datos se procesan para obtener $\delta(t_j)$ y $\iota(t_j)$.

Los valores de los conjuntos de duraciones pueden ser dispersos, por lo que los parámetros $\delta(t_j)$ y $\iota(t_j)$ no representarían cercanamente el comportamiento temporal del proceso. Por esta característica, en el presente artículo se propone una extensión al método referido para mejorar la exactitud (*accuracy*) de la RPT descubierta.

A partir del análisis de los datos sobre las duraciones, se determina si existe más de una agrupación (clúster) de datos; en tal caso la transición es remplazada por un número de transiciones igual al número de clústeres. Posteriormente, se determina una sub-red con dichas transiciones, la cual ordena el disparo en las secuencias $\sigma_{t\,i}$. A este problema se le llama Refinamiento, sobre el cual se enfoca el presente artículo.

Definición 5. Sea N una RPT obtenida a partir de λ_t. El problema de refinamiento de una RPT consiste en extender N agregando un conjunto de subredes N_r tales que la composición $N||N_r$ ejecute λ_t y las temporizaciones de las transiciones en se aproximen al comportamiento observado.

3. Identificación de procesos temporizados

En el contexto de procesos temporizados, en [4] se presenta una extensión a los métodos en [12, 13] donde a partir de una secuencia S_τ de eventos con sus tiempos de ejecución (eventos fechados) se analiza cada ocurrencia de las distintas transiciones

Algoritmo 1. Refinamiento de TPN.

Entrada: $N=(P,T,F)$, S, $\gamma(T)$, Tim
Salida: N_R, Tim

1. $R(t_j) \leftarrow \varnothing$
2. $\forall t_j \in T$:
  If EMgetComponents ($\gamma(t_j)$) > 1
  then
    lab ← GetLabel($\gamma(t_j)$); // obtiene el conj nombres de las tr replicadas
    $R(t_j)$ ← Rename (t_j, lab); // obtiene las tr replicadas
    $\forall t_r \in R(t_j)$
    $\gamma(t_r) \leftarrow$ GetTime($R(t_j)$); //obtiene los tiempos para cada t_r de $R(t_j)$
3. $S_P \leftarrow Pr(S', R(t_j))$ // proyección
  N' ← IdentPN(S_P)
  $N_R \leftarrow$ Merge(N, N') // composición síncrona
4. $\forall\ t_k \in$ lab
  $\delta(t_k) \leftarrow$ average($\gamma(t_k)$);
  $\iota(t_k) \leftarrow$ [min($\gamma(t_k)$), max($\gamma(t_k)$)];
  $Tim_R(t_k) \leftarrow (\delta(t_k), \iota(t_k))$;
  Tim ← Tim ∪ Tim_R
5. Return N_R, Tim

para determinar el tiempo ocurrido entre ellas y finalmente asignar valores de tiempo a cada transición.

La obtención de los valores de tiempo para una PN en [4] se determina una función Tim, a partir de una secuencia de eventos temporizados S_τ y la estructura de la red de Petri.

La estrategia consiste en analizar S_τ comparando cada transición t_k en S_τ(k) con una o varias transiciones previas en la secuencia y determinando el tiempo que ocurre entre $\tau(t_k)$ de S_τ(k) y la $\tau(t_k)$ correspondiente en la secuencia S_τ(k-i).

Estas comparaciones se realizan en base a la semántica de las redes de Petri en que el tiempo transcurrido asociado a una transición t_k representa la duración máxima de la permanencia de una marca que habilita t_k

De esta manera, cada vez que aparece t_k en la secuencia S_τ, se analiza la sub-secuencia anterior a S_τ(k), hasta una aparición previa de t_k o hasta que se haya llegado al inicio de la secuencia, es decir, S_τ(1). En esta sub-secuencia, se analiza la ocurrencia de las transiciones inmediatamente anteriores a t_k en el modelo; la Figura 2 ilustra los elementos a considerar para la obtención de tiempos referentes a t_k.

Para cada lugar previo a t_k deberá detectarse una de sus transiciones de entrada en la sub-secuencia previa a t_k en S_τ. Entre esas transiciones, siendo t_r la transición cuyo instante de registro es el último, el tiempo máximo de residencia de las marcas en el marcado que habilita t_k es $\delta = \tau(t_k) - \tau(t_r)$, lo que es el tiempo transcurrido asociado a t_k para esta sub-secuencia. El algoritmo descrito obtiene para cada transición t_j, el promedio de duración del marcado que habilita t_j, y sus valores mínimo y máximo.

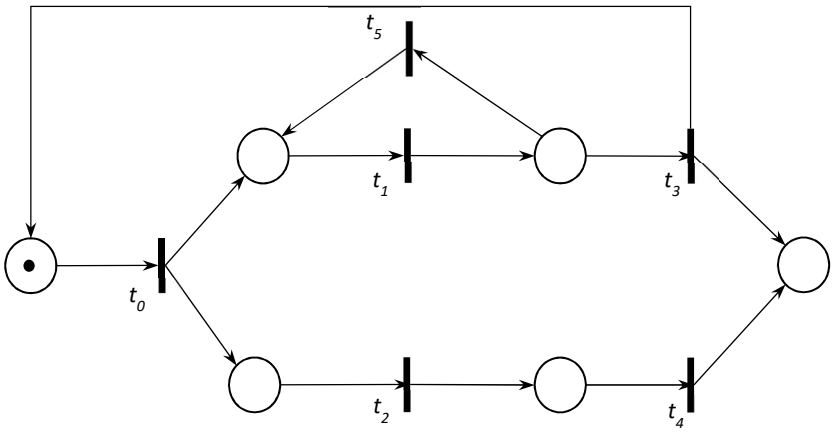

Fig. 4. Modelo que representa proceso de Fig. 3.

Tabla 1. Cálculo de tiempos asociados a un proceso.

Transición	**Tim(t_j)**	**$\gamma(t_j)$**
t_0	(4.017, [3.90, 4.15])	3.93, 3.95, 4.03, 4.04, 3.97, 3.93, 4.06, 3.97, 4.14, 4.05, 3.93, 3.90, 4.02, 3.91, 4.02, 4.14, 4.14, 4.01, 4.15, 4.06
t_1	(7.488, [4.90,10.09])	9.92, 5.01, 10.04, 4.92, 9.94, 4.91, 10.08, 4.94, 9.92, 5.17, 10.01, 4.90, 9.92, 4.93, 10.01, 4.91, 10.02, 5.03, 10.05, 5.01, 10.09, 4.91, 9.94, 5.06, 10.03, 5.11, 9.94, 4.95
t_2	(9.884, [9.67,10.06])	10.02, 9.96, 10.06, 9.88, 9.94, 9.99, 9.93, 9.80, 9.99, 9.76, 10, 9.73, 9.88, 9.80, 9.83, 9.93, 9.67, 9.71, 9.88, 9.93
t_3	(9.901, [9.80,10.06])	9.87, 9.91, 9.94, 9.92, 9.80, 9.80, 9.80, 9.85, 9.83, 10.05, 9.91, 9.93, 9.81, 9.98, 9.98, 9.84, 10.06, 10.01, 9.83, 9.91
t_4	(9.954, [9.82,10.10])	9.82, 10, 10.02, 9.90, 10.04, 10.10, 9.91, 10.07, 9.87, 10.05, 9.94, 9.86, 9.91, 10.06, 9.98, 9.86, 9.93, 9.92, 10.01, 9.84
t_5	(4.809,[4.47,5.10])	4.73, 5.06, 4.7, 4.86, 4.69, 4.66, 4.49, 4.74, 4.91, 5.1, 4.96, 5, 5.09, 4.69, 4.47, 4.91, 4.60, 4.91, 5.08, 4.53

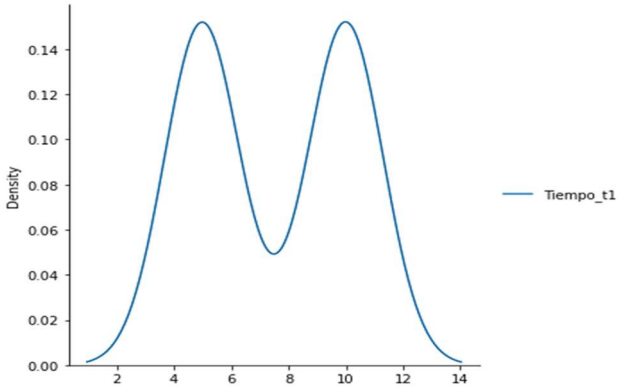

Fig. 5. Clústeres obtenidos para una transición.

4. Refinamiento de redes de Petri temporizadas

4.1. Enfoque

El método propuesto aborda el caso en que se cuenta con un modelo obtenido mediante algún método de identificación, así como una secuencia de eventos con sus tiempos de ejecución. Siguiendo la propuesta de [4], se generan conjuntos de valores de tiempo de disparo para cada transición; estos tiempos son analizados para identificar si los valores de duración pueden agruparse en dos o más conjuntos de valores (clúster).

Si se detecta más de una agrupación, es necesario modificar localmente el modelo agregando transiciones para hacerlo más exacto. En este trabajo se utiliza el algoritmo esperanza-maximización (EM) [14], dado que permite obtener el número de clústeres alrededor de diversos valores medios sin necesidad de especificarlos previamente, a diferencia del algoritmo k-means [15, 16].

Al aplicar el algoritmo EM para obtener los clústeres y valores medios del conjunto de datos, se obtienen etiquetas asignadas a cada elemento para identificar a que clúster pertenece cada valor en la secuencia.

Con esta información, una transición se sustituirá por tantas transiciones como clústeres se encuentren a fin de ajustar el modelo inicial. En particular para este trabajo se ha utilizado la biblioteca de Python scikit-learn [18], específicamente el objeto GaussianMixture que permite la implementación del algoritmo EM.

4.2. Método de refinamiento

El método propuesto se puede dividir en cinco etapas:

1. Aplicación del algoritmo EM al conjunto de duraciones $\gamma(t_j)$ calculadas para cada transición t_j, para obtener el número de componentes con sus valores medios respectivos.

Tabla 2. Tiempos de t_1 agrupados por clúster.

Transición	$\gamma(t_j)$
t_1	9.92, 10.04, 9.94, 10.08, 9.92, 10.01, 9.92, 10.01, 10.02, 10.05, 10.09, 9.94, 10.03, 9.94
t_1',	5.01, 4.92, 4.91, 4.94, 5.17, 4.90, 4.93, 4.91, 5.03, 5.01, 4.91, 5.06, 5.11, 4.95

2. Obtener las etiquetas correspondientes a cada una de las medias. Al aplicar el algoritmo EM se obtiene, además del número de clústeres existente en los datos, las etiquetas que permiten identificar a que clúster pertenece cada uno de los elementos analizados. Esta información es útil ya que ayudará a agrupar los valores pertenecientes a cada una de las transiciones replicadas.
3. A partir de las etiquetas obtenidas se replican las transiciones t_j que tiene más de un clúster. En el modelo *N*, t_j y sus réplicas tendrán los mismos $^\bullet t_j$ y $t_j^\bullet$. Las nuevas transiciones (T') se renombran y se actualizan en la secuencia original S; se obtiene una secuencia S'.
4. Hacer una proyección de S' sobre las nuevas T' renombradas Pr (S', T'). Sobre esa proyección, se aplica el método de identificación a S' para obtener una subred N' con T' y nuevos lugares P'.
5. Combinar la subred N' recién creada con la red de Petri original N para generar el modelo refinado (N||N').

El procedimiento se repite para cada transición t_j que tenga más de un clúster en el análisis de $\gamma(t_j)$ del Paso 1. Cada paso del método se describe con más detalle a continuación.

4.2.1 Análisis de las duraciones calculadas

Dado que se pueden encontrar variaciones de los tiempos en $\gamma(t_j)$, estos conjuntos se analizan con el algoritmo EM. En una primera instancia, se le especifica un rango definido de componentes para probar.

El algoritmo entrega el valor 1 (número de componentes) cuando los valores de tiempo están alrededor de una sola media; si este es el caso, esa transición no requerirá ajustes y la siguiente transición podrá ser procesada. Cuando el algoritmo devuelva un valor de 2 o más componentes para una transición, se aplicarán los pasos consecuentes.

4.2.2 Replicar transiciones con más de un clúster

Si un $\gamma(t_j)$ tiene dos o más componentes se deben crear nuevas transiciones. Entonces se utiliza un método que proporciona las etiquetas para las muestras de datos; cada dato de entrada tendrá asociado un número de etiqueta y habrá tantas etiquetas diferentes como clústeres en la transición. En el modelo N todas las transiciones replicadas de t_j se agregan usando los mismos lugares de entrada y de salida de t_j. Las transiciones creadas son renombradas para ser incluidas en S.

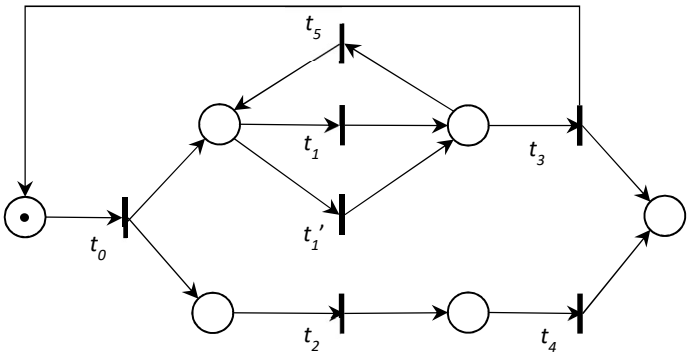

Fig. 6. Subred creada a partir de proyección en S.

4.2.3 Renombrar y sustituir las transiciones en S

Analizando las etiquetas, la transición se replica y se deben renombrar. Así, por ejemplo, t_j se puede convertir en t_j, y t_j' dependiendo de las etiquetas correspondientes. Las transiciones replicadas se incluyen en un conjunto T_j. Con la información proporcionada por las etiquetas, se relaciona cada valor de tiempo de la transición que se está tratando con el clúster al que pertenece.

A partir de esta relación se determina el número de transiciones nuevas representan ahora a t_j; para cada una se generan nuevos nombres y se determina la inclusión de las transiciones replicadas en la secuencia original. La secuencia actualizada se denomina S'.

4.2.4 Obtención de una subred con las transiciones replicadas

Luego de la sustitución, deberá modificarse la red original para incluir las nuevas transiciones. Para ello cada transición replicada tendrá los mismos lugares de entrada t_j y de salida t_j que t_j.

Adicionalmente, se realiza una proyección de S' sobre las transiciones replicadas T' $(Pr(S', T'))$; esto genera una secuencia S_P, formada por transiciones de T', a la cual se le aplicará un método de identificación [5] para obtener una RP N' con las transiciones de T' y nuevos lugares P'.

4.2.5 Fusionar la subred creada con el modelo original

Finalmente, se fusiona la subred creada N' con el modelo N de la TPN con las transiciones replicadas de acuerdo al método propuesto en [17]. La nueva red resultado de la composición síncrona N||N' y las temporizaciones correspondientes a las transiciones en T', mejora su exactitud. Los pasos anteriores se resumen en el Algoritmo 1.

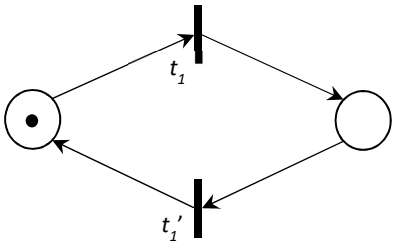

Fig. 7. Subred creada a partir de proyección en S.

5. Caso de estudio

A continuación, se presenta un ejemplo de aplicación del algoritmo propuesto. En la Figura 3 se muestra un proceso que puede ser modelado por una TPN. Se trata de un proceso con dos dispositivos móviles (carros) que se desplazan en vías separadas. Cada carro tiene dos tipos de movimiento, hacia la derecha y hacia la izquierda; en la figura se indica la secuencia de desplazamientos que tienen que hacer cada carro.

El carro 1 realiza una secuencia que toma más tiempo; es claro ver que los desplazamientos del carro 1, tanto a la derecha como a la izquierda, tienen duración diferente. Si en el modelo el movimiento hacia la derecha se representa mediante una transición, las duraciones calculadas tendrán variaciones.

Al aplicar el método propuesto, se deberá encontrar dos transiciones distintas que representen el movimiento a la derecha, pero con diferentes tiempos asignados muy similares.

Considere que en el proceso previo se definieron las siguientes actividades $T = \{t_0, t_1, t_2, t_3, t_4, t_5\}$, donde t_1 y t_2 representan el desplazamiento a la derecha de los carros 1 y 2 respectivamente; t_3 y t_5 representan los desplazamientos largo y corto, respectivamente, a la izquierda del carro 1; t_0 es el evento de inicio. Las secuencias registradas de la ejecución del proceso son:

S = *t0, t1, t2, t5, t1, t4, t3, t0, t2, t1, t5, t4, t1, t3, t0, t1, t2, t5, t1, t4, t3, t0, t2, t1, t5, t4, t1, t3, t0, t1, t2, t5, t1, t4, t3, t0, t2, t1, t5, t1, t4, t3, t0, t1, t2, t5, t1, t4, t3, t0, t2, t1, t5, t1, t4, t3, t0, t1, t5, t2, t4, t1, t3, t0, t1, t5, t2, t1, t4, t3, t0, t2, t1, t4, t5, t1, t3, t0, t1, t2, t4, t5, t1, t3, t0, t1, t2, t5, t4, t1, t3, …*

S_τ = *(t0, 3.93), (t1, 13.85), (t2, 13.95), (t5, 18.58), (t1, 23.59), (t4, 23.77), (t3, 33.64), (t0, 37.59), (t2, 47.55), (t1, 47.63), (t5, 52.69), (t4, 57.55), (t1, 57.61), (t3, 67.52), (t0, 71.55), (t1, 81.49), (t2, 81.61), (t5, 86.19), (t1, 91.1), (t4, 91.63), (t3, 101.57), (t0, 105.61), (t2, 115.49), (t1, 115.69), (t5, 120.55), (t4, 125.39), (t1, 125.49), (t3, 135.41), (t0, 139.38), (t1, 149.3), (t2, 149.32), (t5, 153.99), (t1, 159.16), (t4, 159.36), (t3, 169.16), (t0, 173.09), (t2, 183.08), (t1, 183.1), (t5, 187.76), (t1, 192.66), (t4, 193.18), (t3, 202.98), (t0, 207.04), (t1, 216.96), (t2, 216.97), (t5, 221.45), (t1, 222.38), (t4, 226.88), (t3, 236.68), (t0, 240.65), (t2, 250.45), (t1, 250.66), (t5, 255.4), (t1, 260.31), (t4, 260.52), (t3, 270.37), …*

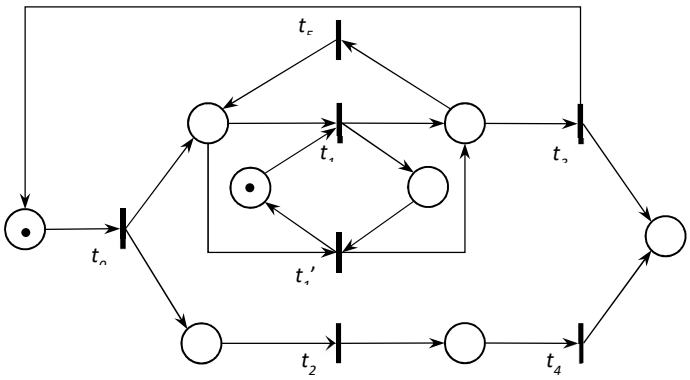

Fig. 8. RPT obtenida después de refinamiento.

A partir de la secuencia no fechada S, un método de identificación obtendrá la RP mostrada en la Figura 4. En cuanto a la determinación de los tiempos de ejecución, en la Tabla 1 se muestra un listado de tiempos que pueden ser obtenidos mediante el método descrito en [4] en base a S_τ y la RP de la Figura 4.

Analizando la tabla, se puede observar que el tiempo promedio para la t_1 es de 7.488, con un intervalo de tiempo entre 4.90 y 10.09. Esto se debe a que la transición se ejecuta algunas veces con valores de tiempo alrededor de 5, y otras veces alrededor de 10, lo que genera un intervalo amplio y una media no muy exacta.

Siguiendo los pasos descritos del algoritmo propuesto, en primer lugar, se realiza una búsqueda de clústeres en los tiempos de cada transición. Se obtiene un clúster para t_0, t_2, t_3, t_4, t_5 y dos clústeres para t_1 como se muestra en la figura 5. Luego, el resto del algoritmo se aplica a esta transición. Al aplicar el algoritmo EM, además del número de clústeres, se obtienen los valores de las medias y una etiqueta para cada elemento, que identifica a que clúster pertenece dicho elemento.

De esta manera los valores de las medias son 4.98 y 9.99. Por su parte, los elementos de tiempo que anteriormente pertenecían a t_1 hora se dividen como se muestra en la tabla 2. Una vez teniendo los clústeres, las transiciones son renombradas como t_1 y t_1' (T'). La RP *N* se refina de acuerdo a manera descrita anteriormente quedando como se muestra en la Figura 6. Por otro lado, T' = {t_1 y t_1'} son incluidas en la secuencia original S para obtener S' de la siguiente manera:

S' = $t_0, t_2, t_1, t_5, t_1', t_4, t_3, t_0, t_1, t_2, t_5, t_1', t_4, t_3, t_0, t_1, t_5, t_2, t_4, t_1', t_3, t_0, t_1, t_5, t_2, t_1', t_4, t_3, t_0, t_2, t_1, t_4, t_5, t_1', t_3, t_0, t_1, t_2, t_4, t_5, t_1', t_3, t_0, t_2, t_1, t_5, t_4, t_1', t_3, t_0, t_1, t_2, t_5, t_4, t_1', t_3, \ldots$

Haciendo la proyección de S'en t_1 y t_1' se obtiene:

$$S_P = Pr(S', T') = t_1, t_1', t_1, t_1', t_1, t_1', t_1, t_1', t_1, t_1', t_1, t_1', t_1, t_1', t_1, t_1'$$

Aplicando el método de identificación a S_P se obtiene la RP mostrada en la Figura 7. Las nuevas transiciones obtenidas t_1 y t_1' serán agregadas a la red de Petri inicial respetando el Pre y el Post de la transición original t_1. Esto se ilustra en la figura 6. En el ejemplo, se realiza la proyección de S en las transiciones recién etiquetadas y se obtiene una PN como la que se muestra en la Figura 7.

Finalmente, la subred creada N' es fusionada con el modelo N de la TPN (N_R=N||N') siguiendo los pasos indicados en [17]; ésta red es mostrada en la Figura 8. Los parámetros de tiempo correspondientes a las transiciones en T' son Tim(t_1) = (9.99, [9.92, 10.09]) y Tim(t_1') = (4.982, [4.90, 5.17]).

6. Conclusión

Se ha presentado un método para identificar procesos temporizados donde los modelos usados son redes de Petri temporizadas. La propuesta extiende un algoritmo básico existente que determina parámetros de tiempo para una red de Petri.

El nuevo método permite abordar casos en que los valores de tiempo calculados para algunas transiciones son muy variables. A partir de un primer modelo N obtenido y el cálculo de las temporizaciones, el algoritmo propuesto determina un refinamiento estructural de N y un aumento de la exactitud de las temporizaciones. Cuando se presentan estos casos de dispersión, el método aumenta la exactitud del modelo.

Esta propuesta permite obtener modelos más exactos en la temporización; además, su desempeño es bastante eficiente respecto a los trabajos publicados sobre identificación de procesos temporizados.

Actualmente se estudian situaciones en las que los valores de tiempo para una transición no sigan una distribución normal o en las que la media no represente fielmente el comportamiento del tiempo del sistema. Se analizan medidas de dispersión para cubrir más escenarios y describir de manera adecuada el comportamiento de los parámetros temporales.

También se trabaja en la definición de medidas para evaluar la exactitud de los modelos con respecto a los registros de eventos temporizados, tomando en consideración las desviaciones del instante de ocurrencia de los eventos con respecto al disparo esperado de las transiciones en el modelo.

Referencias

1. Meda-Campaña, M., Ramirez-Treviño, A., López-Mellado, E.: Asymptotic identification of discrete event systems. In: Proceedings of the 39th IEEE Conference on Decision and Control, pp. 2266–2271 (2000) doi: 10.1109/CDC.2000.914135
2. Meda-Campaña, M., López-Mellado, E.: Identification of concurrent discrete event systems using Petri nets. In: Proceedings of the 17th IMACS World Congress on Computational and Applied Mathematics, pp. 11–15 (2005)
3. Estrada-Vargas, A. P., López-Mellado, E., Lesage, J. J.: A black-box identification method for automated discrete-event systems. IEEE Transactions on Automation Science and Engineering, vol. 14, no. 3, pp. 1321–1336 (2017) doi: 10.1109/TASE.2015.2445332
4. Rodríguez-Pérez, E., Tapia-Flores, T., López-Mellado, E.: Identification of timed discrete event processes. Building input-output petri net models, vol. 16, pp. 153–167 (2016)
5. Tapia-Flores, T., López-Mellado, E., Estrada-Vargas, A. P., Lesage, J. J.: Discovering Petri net models of discrete-event processes by computing T-invariants. IEEE Transactions on Automation Science and Engineering, vol. 15, no. 3, pp. 992–1003 (2017) doi: 10.1109/ TASE.2017.2682060

6. David, R., Alla, H.: Discrete, continuous, and hybrid Petri nets. Springer (2010) doi: 10.1007/978-3-642-10669-9
7. Basile, F., Chiacchio, P., Coppola, J.: Identification of time Petri net models. IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 47, no. 9, pp. 2586–2600 (2016) doi: 10.1109/TSMC.2016.2523929
8. Meda-Campaña, M. E., Medina-Vazquez, S.: Synthesis of timed Petri net models for on-line identification of discrete event systems. In: 9th IEEE International Conference on Control and Automation, pp. 1201–1206 (2011) doi: 10.1109/ICCA.2011.6137968
9. Murata, T.: Petri nets: Properties, analysis and applications. In: Proceedings of the IEEE, vol. 77, no. 4, pp. 541–580 (1989) doi: 10.1109/5.24143
10. Ramchandani, C.: Analysis of asynchronous concurrent systems by timed Petri nets. Ph. D. Thesis, Massachusetts Institute of Technology (1973) http://hdl.handle.net/1721.1/13739
11. Merlin, P. M.: A study of the recoverability of computing systems. University of California, Irvine (1974)
12. Estrada-Vargas, A. P., López-Mellado, E., Lesage, J. J.: Input–output identification of controlled discrete manufacturing systems. International Journal of Systems Science, vol. 45, no. 3, pp. 456–471 (2014) doi: 10.1080/00207721.2012.724098
13. Estrada-Vargas, A. P., Lesage, J. J., López-Mellado, E.: A stepwise method for identification of controlled discrete manufacturing systems. International Journal of Computer Integrated Manufacturing, vol. 28, no. 2, pp. 187–199 (2015) doi: 10.1080/0951192X.2013.874591
14. Moon, T. K.: The expectation-maximization algorithm. IEEE Signal Processing Magazine, vol. 13, no. 6, pp. 47–60 (1996) doi: 10.1109/79.543975
15. Alldrin, N., Smith, A., Turnbull, D.: Clustering with EM and K-means. University of San Diego, California, Tech Report, pp. 261–295 (2003)
16. Li, M. J., Ng, M. K., Cheung, Y. M., Huang, J. Z.: Agglomerative fuzzy k-means clustering algorithm with selection of number of clusters. IEEE Transactions on Knowledge and Data Engineering, vol. 20, no. 11, pp. 1519–1534 (2008) doi: 10.1109/TKDE.2008.88
17. López-Mellado, E., Flores-Tapia, T.: Refining discovered Petri nets by sequencing repetitive components. In: International Workshop on Algorithms and Theories for the Analysis of Event Data Conference (2017)
18. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot M., Duchesnay, E.: Scikit-learn: Machine learning in python. Journal of Machine Learning Research, vol. 12, pp. 2825–2830 (2011)