Detección de concept drift en minería de procesos basado en agrupamiento de trazas

Víctor José Gallego Fontenla, Juan Carlos Vidal Aguiar, Manuel Lama Penín

Centro Singular de Investigación en Tecnoloxías da Información
Universidade de Santiago de Compostela
{victorjose.gallego, juan.vidal, manuel.lama}@usc.es

Resumen En este artículo se presenta un método para la detección y localización de *concept drift* en minería de procesos, que, a diferencia del resto de propuestas del estado del arte, combina técnicas de agrupamiento de trazas y descubrimiento de modelos para realizar una clasificación de las trazas de ejecución contra una serie de modelos que constituyen el *ground truth* de nuestro sistema. Esta aproximación permite detectar, localizar y caracterizar los cambios y evaluar la evolución sufrida por el proceso. El algoritmo ha sido validado con un registro de eventos sintético que presenta puntos de *concept drift*, demostrando que la aproximación tomada es válida a la hora de detectar cuándo tiene lugar un cambio en la estructura del modelo de proceso.

1. Introducción

El campo de la inteligencia de negocio que busca extraer conocimiento útil relacionado con los procesos de negocio a partir de registros de actividad se conoce como *minería de procesos*, y trata de resolver tres problemas bien diferenciados [1]: *(i)* el descubrimiento de modelos de proceso [2–4], donde se obtiene el modelo que describe el comportamiento real que ha tenido lugar; *(ii)* la verificación de la conformidad [5, 6], cuyo objetivo consiste en comprobar en qué medida el modelo de proceso se ajusta a la realidad; y *(iii)* la mejora de modelos de proceso [7, 8], donde se extrae conocimiento sobre lo que sucede o puede suceder en el proceso.

Ahora bien, los procesos suelen evolucionar con el tiempo para adecuarse a su entorno, por lo que la detección y caracterización de cambios en la estructura del proceso resulta de vital importancia a la hora de obtener unos resultados adecuados cuando se aplican cualquiera de las tareas principales de minería de procesos. Así, se conocen como técnicas de detección del *concept drift* a las técnicas que permiten manejar los cambios producidos en los datos a lo largo del tiempo, de forma que se pueda adaptar el modelo aprendido a los datos más recientes. Por lo general, se distinguen tres tipos de problemas en *concept drift* [9]: (i) *detección del cambio*, que constituye el problema principal a tratar y que consiste en detectar si ha ocurrido o no un cambio; (ii) *localización y caracterización del cambio*, o lo que es lo mismo, detección del cómo y el cuándo del cambio ocurrido; y (iii) *descripción de la evolución del proceso*, una vez se dan

los cambios detectados y localizados anteriormente. En este problema podemos, por ejemplo, analizar cada cuánto tiempo se repite el cambio en caso de ser estacional o mostrar cómo evoluciona el proceso con los sucesivos cambios.

En minería de procesos la detección del cambio se enfrenta a un problema adicional: la complejidad estructural inherente de los procesos. En muchos casos las técnicas de descubrimiento de modelos a partir de registros de eventos muy complejos pueden generar modelos con muchas relaciones entre las tareas, derivando en los conocidos como modelos *spaghetti*. Este tipo de modelos no resultan útiles a las organizaciones, ya que no reflejan la estructura real del proceso o reflejan una representación tan especifica que resulta demasiado compleja. A esto hay que añadir que la obtención de modelos válidos a partir de registros complejos suele tener asociado un elevado coste computacional.

En este artículo presentamos una aproximación que permite *detectar* los cambios estructurales en un proceso, procurando revelar los cambios con el menor retardo posible una vez se hayan dado, *localizar* la traza concreta del registro que ha desencadena la detección, y describir la *evolución* del proceso, proporcionando una representación previa al cambio y otra posterior a éste. Además, aunque en el artículo nos centramos en un escenario *offline*, la aproximación algoritmo sería fácilmente extensible a un escenario *online* incluyendo una fuente de eventos que vaya añadiendo las trazas más recientes resultado de las sucesivas ejecuciones del proceso. Para evaluar la aproximación se han realizado una serie de pruebas empleando registros de eventos que presentan *concept drift* generados sintéticamente.

El resto de este artículo se estructura como sigue: en la Sección 2 se realiza un breve análisis del estado del arte en cuanto a la detección de *concept drift* en minería de procesos; en la Sección 3 se presenta el algoritmo propuesto para la detección de los cambios; en la Sección 4 se detallan los resultados experimentales del algoritmo propuesto; y, para finalizar, en la Sección 5 se presentan las conclusiones y el trabajo futuro.

2. Trabajos relacionados

A pesar de que la minería de procesos y el tratamiento del *concept drift* constituyen dos campos de estudio muy activos en los últimos años, la combinación de ambos todavía se encuentra en una fase inicial. Así, algunos de los trabajos más relevantes son:

- En [9], los autores proponen un método basado en la extracción de características de los registros y el uso de test de hipótesis para determinar si ha habido un cambio significativo o no en éstas. Los principales inconvenientes de este método son que *(i)* únicamente trata la detección *offline*; *(ii)* las características propuestas son demasiado simples como para permitir representar todas las modificaciones que se pueden dar en el proceso; *(iii)* el tamaño de las características definidas tienen una elevada dimensionalidad para registros complejos, lo que convierte su tratamiento en un problema

intratable para procesos reales; y *(iv)* no evalúa la evolución del proceso una vez que se detecta el *concept drift*.

- En [10], los autores proponen una aproximación basada en representar un subconjunto de trazas mediante una interpretación abstracta, empleando un poliedro; y posteriormente emplean esta representación para evaluar si el modelo cambia a lo largo del tiempo, mediante el uso de una ventana de tamaño adaptativo. El principal problema de este método es que el elevado nivel de abstracción que se hace al intepretar las trazas puede dar lugar a una perdida de información estructural del proceso que puede provocar falsos negativos a la hora de detectar cambios.
- En [11], los autores proponen una aproximación basada en la extracción de clústeres presentes en el registro en diferentes instantes de tiempo y su posterior comparación. Sus principales desventajas son que *(i)* sigue una aproximación únicamente *offline*; *(ii)* no trata el problema de la localización y caracterización del cambio; *(iii)* no estudia como evoluciona el proceso una vez se ha dado el cambio; y *(iv)* mezcla características de la estructura del proceso con otros atributos de la ejecución, como pueden ser propiedades relativas a los recursos, lo que puede dar lugar a confusión entre *concept drift* real y *concept drift* virtual.

Es esta carencia de una solución completa, que trate todos los problemas propuestos y sea aplicable a la realidad, lo que nos lleva a buscar una aproximación diferente a la detección y tratamiento de *concept drift* en minería de procesos.

3. Detección de concept drift basado en agrupamiento de trazas

Dado un registro de eventos, el objetivo es detectar el momento en que un porcentaje de casos mayor que un umbral dado se desvía del proceso subyacente hasta el momento. El método propuesto en este artículo se encuentra representado en la Figura 1. En un primer lugar se realiza un agrupamiento de un subconjunto representativo de las trazas presentes en el registro con el fin de obtener modelos más simples, y a continuación se genera un modelo para cada uno de los clústeres resultantes. A continuación, se desliza una ventana de un tamaño fijo sobre el registro, comprobando, de forma secuencial, si las trazas que se van añadiendo a ésta se adaptan a alguno de los modelos generados, y en caso de que no sea así se determina si se trata de un *outlier* o por el contrario estamos ante un suceso de *concept drift*. En los siguientes apartados se describirán en mayor detalle cada una de las partes que forman parte del algoritmo.

3.1. Agrupamiento de trazas

El primer paso del método de detección de *concept drift* planteado consiste en la realización de un agrupamiento de las trazas en clústeres, de forma que las trazas más parecidas entre ellas se encuentren en el mismo grupo, generando así

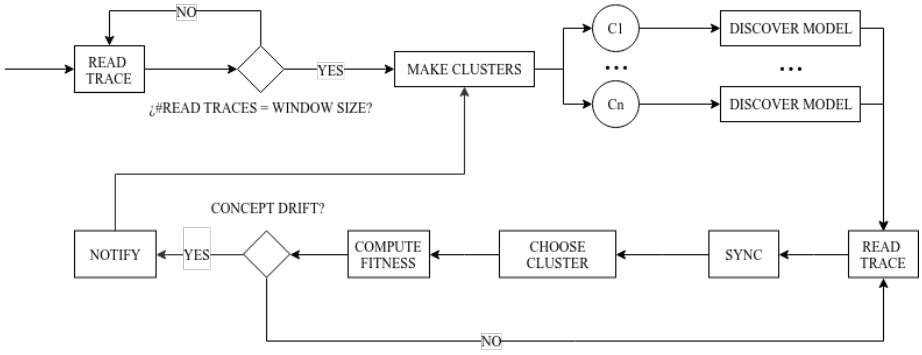


Figura 1. Estructura básica del algoritmo planteado.

modelos más simples. Para ello, se ha empleado un algoritmo de agrupamiento jerárquico aglomerativo [12].

El funcionamiento de este tipo de algoritmos es simple: se empieza considerando los datos de entrada como clústeres individuales, y se van agrupando de forma que la *distancia* entre elementos de un mismo clúster sea mínima, hasta que queda un único clúster. Debido a su funcionamiento, este tipo de algoritmos no devuelven una lista de clústeres, sino que crean una jerarquía de clústeres. Para obtener un número de clústeres k basta con cortar dicha jerarquía en el nivel adecuado.

Para calcular las distancias entre cada una de las trazas del conjunto de datos se debe definir una medida que permita calcular cuán diferentes son. La mayoría de las aproximaciones utilizan distancias de edición de textos (Hamming, Levenshtein o la distancia de edición) para medir la similaridad entre dos trazas. Esta elección se fundamenta en que las trazas se representan mediante cadenas de texto, donde cada símbolo representa la ejecución de una tarea. No obstante, a pesar de que estas medidas pueden ser aplicadas al calculo de distancias entre trazas, no proporcionan resultados óptimos, pues hay ciertos aspectos que no pueden detectar, como es el caso de los bucles o de las ramas paralelas, proporcionando resultados poco correctos. Por ejemplo: si calculamos la distancia de Levenshtein entre las trazas $t_1 = aaabcd$ y $t_2 = aaaaaaaaaaabcd$ obtendremos un valor de disimilitud de 9, pues se deben insertar 9 caracteres en t_1 para obtener t_2 (o eliminar el mismo número de caracteres de t_2), lo que indicaría que ambas trazas son muy diferentes. En cambio, lo más probable es que ambas trazas provengan del mismo proceso, donde nos encontramos con un bucle en la tarea a y a continuación las tareas b, c y d en secuencia (Fig. 2).

Para evitar este tipo de situaciones hemos optado por construir el grafo que representa a cada traza y utilizar una *distancia de edición de grafos*, que se basa en la distancia de edición de textos pero permite reducir en gran medida los errores en el cálculo de distancias entre dos trazas provenientes del mismo proceso, pues tiene en cuenta la posible presencia de bucles, tanto de una única

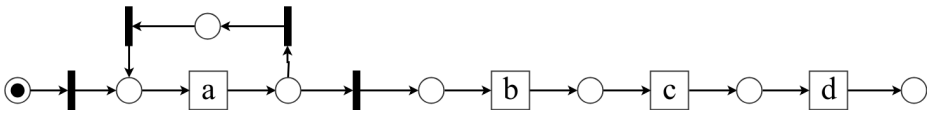


Figura 2. Red de Petri ejemplo para representar las trazas *aaabcd* y *aaaaaaaaaaaabcd*.

tarea como de múltiples. La formulación matemática de esta medida, dados dos grafos g_1 y g_2 es la siguiente:

$$GED(g_1, g_2) = \min_{e_1..e_k \in P(g_1,g2)} \sum_{i=1}^{k} c(e_i) \tag{1}$$

donde $P(g_1, g_2)$ representa todos los conjuntos de ediciones posibles para transformar g_1 en $g2$, y $c(e_i) \geq 0$ el coste de aplicar la operación e_i sobre el grafo.

Una vez decidida la medida de distancia a emplear para comparar trazas se debe definir como se computarán las distancias entre los distintos clústeres. Para ello, se ha usado en *enlace promedio*, donde la distancia entre dos clústeres viene determinada por la media de las distancias entre cada par de trazas de cada uno de los clústeres. Esta elección es una solución de compromiso entre el enlace simple, que funciona mejor en conjuntos de datos razonablemente limpios, y el enlace completo, que es más propenso a generar clústeres con formas esféricas, por lo que es menos susceptible al ruido.

El tercer problema a resolver una vez definidas las medidas de distancia entre trazas y entre clústeres es la elección del número óptimo de clústeres para cada conjuntos de datos. Puesto que no tenemos información *a priori* del modelo subyacente en el registro que queremos procesar, y debido a que es probable que este cambie debido a los eventos de *concept drift*, hemos incorporado un método de selección automática del número de clústeres empleando la medida de *Silhouette*, que analiza cuán similar es un dato respecto al resto de datos de su mismo clúster frente a lo diferente que es de los datos del resto de clústeres. El objetivo será buscar el número mínimo de clústeres donde el valor de la medida sea máximo.

3.2. Descubrimiento de modelos de procesos

Una vez realizado el agrupamiento se lleva a cabo un proceso de descubrimiento para cada clúster con el objetivo de obtener la estructura del proceso definido por las trazas correspondientes. Para realizar esta tarea existen multitud de opciones, como las presentadas en [2, 3, 13], que siguen diversas aproximaciones hasta obtener un modelo lo más simple posible y que, preferiblemente, represente la totalidad del comportamiento presente en el registro a tratar. En este caso se ha optado por usar el algoritmo *Heuristics Miner* [4], debido a que proporciona unos resultados buenos en un tiempo reducido, lo que permitiría detectar *concept drift* en tiempo real.

Cabe destacar que entre el agrupamiento de trazas y el descubrimiento de modelos se ha realizado un proceso de detección de *outliers* para evitar "contaminar" los modelos generados con ruido. En este caso la detección de *outliers* se ha limitado a realizar un filtrado de los datos que aparecen en la ventana con una frecuencia menor a un porcentaje indicado por el usuario. Es decir:

$$\frac{count(datum, window)}{size(window)} < threshold \Longrightarrow outlier$$

Los *outliers* detectados en este paso no son eliminados, sino que son excluidos de la fase de descubrimiento de modelos y añadidos a la lista de candidatos a sucesos de *concept drift*, sirviendo como base a los candidatos a considerar al tratar las trazas restantes.

3.3. Sincronización de candidatos a concept drift

Tras generar los modelos, y antes de proceder a comprobar si una traza se adapta a alguno de los modelos generados anteriormente, debemos realizar un proceso de sincronización de los candidatos a *concept drift* con el contenido de la ventana deslizante, de modo que únicamente conservaremos los candidatos que modelen alguna de las trazas en la ventana, y olvidando así los candidatos más antiguos que han quedado fuera de ésta, que consideraremos que no tienen la tasa de aparición necesaria para representar un *concept drift*, sino que constituyen simples *outliers* en el registro de eventos.

En la Figura 3 se presenta un ejemplo simplificado de este proceso. En los instantes t_n y t_{n+1} todos los candidatos forman parte del contenido de la ventana, por lo que tras la sincronización no cambian. En cambio, en el instante t_{n+2} entra un nuevo dato con valor 8 en la ventana, que pasa a formar parte del conjunto de candidatos, y tras la sincronización desaparece el 9, pues ya no está presente entre los datos observados.

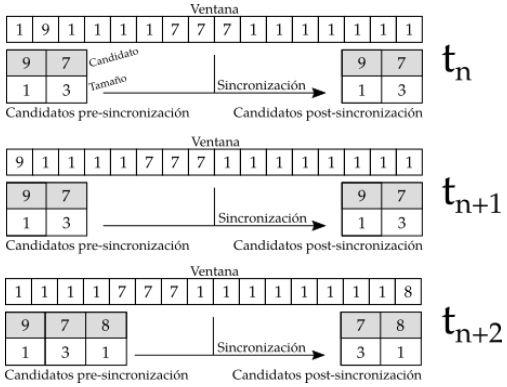


Figura 3. Ejemplo del proceso de sincronización de la ventana y los candidatos.

3.4. Comprobación de las trazas restantes

En este paso del algoritmo se comprueba si las trazas restantes del registro de eventos se adaptan o no a alguno de los modelos generados. Para ello, se toman los modelos generados en la etapa de minado y se intenta ejecutar en cada uno la traza a procesar.

Se considera que una traza se adapta a un modelo cuando todas sus tareas pueden ser ejecutadas en el orden dado y el último evento presente en ésta se corresponde con una tarea final del proceso. Si la traza es válida para alguno de los modelos generados anteriormente se considera que no es candidata a producir *concept drift*, pues es producto de una ejecución del modelo actual. Por el contrario, si la traza no representa una ejecución válida para ninguno de los modelos actuales, debe ser incluida como candidata a producir *concept drift*.

3.5. Detección de concept drift

En este paso se decide si los candidatos a *concept drift* representan un *concept drift* real o únicamente se tratan de *outliers*. Así, las trazas que no se ajustan a los modelos minados, debemos actualizar los candidatos a producir *concept drift*. Para llevar a cabo esta tarea volvemos a emplear la medida de distancia definida en (1).

En caso de que la distancia entre la traza a procesar y las trazas presentes en el candidato sea cero consideraremos la traza como una nueva ocurrencia del candidato a *concept drift* e incrementaremos el contador de ocurrencias correspondiente en uno; en caso contrario la consideraremos un nuevo candidato a *concept drift* y la añadimos a la lista de candidatos. Tras procesar la traza y actualizar la lista de candidatos a *concept drift*, comprobamos si ésta contiene algún candidato con un porcentaje de ocurrencias mayor que el umbral definido cuando se lanza el algoritmo.

$$\exists c \in driftCandidates : \frac{size(c)}{window\ size} \geq threshold$$

Si esta condición se cumple se considera que existe *concept drift*, con lo que se elimina el candidato a *concept drift* de la lista de candidatos y se vuelve a la fase de agrupamiento del algoritmo para volver a ejecutar todas las fases del algoritmo, teniendo en cuenta que los nuevos modelos generados deben incluir ahora a los candidatos que han acabado desencadenando un *concept drift* en interaciones anteriores.

4. Experimentación y resultados

La validación se realizó con un registro de eventos sintético que incluye 5 modificaciones en la estructura del proceso a lo largo del tiempo, y consta de 12.000 trazas de ejecución (2.000 para el modelo original y 2.000 para cada una de las modificaciones), 12 actividades diferentes y 87.851 eventos.

El algoritmo se ha ejecutado con varios tamaños de ventana y umbrales de detección sobre la sección del registro correspondiente a cada una de las modificaciones.

En la Tabla 1 se muestra el número de falsos positivos detectados para cada una de las configuraciones. La aparición de estos falsos positivos es debida a la fase de descubrimiento del modelo del proceso, ya que no siempre consiguen cubrir el 100 % a todas las trazas empleadas para la generación (completitud o *replay fitness* < 1). Este problema se ve agravado con el aumento de complejidad de los modelos, como se puede comprobar en el caso de la modificación 4, la más compleja de todas las versiones del proceso empleado en estas pruebas.

Además, también se ha ejecutado el algoritmo para el registro de eventos completo, empleando tamaños de ventana de 100, 200, 500, 1.000 y 2.000 trazas y umbrales para la detección de *outliers* del 5 %, 10 %, 15 %, 20 % y 25 %. Como se puede apreciar en la Tabla 2, a menor tamaño de ventana, mayor es el número de cambios detectados. Al igual que en el caso anterior, el motivo es la falta de trazas para construir un modelo con buen *fitness replay* y por ello aumenta la probabilidad de detectar cambios.

Además del número de cambios, resulta importante para valorar la calidad de la solución observar como se distribuyen estos a lo largo del registro. Los cambios reales se dan en las trazas 2.000, 4.000, 6.000, 8.000 y 10.000. En las figuras 4a y 4b se presentan los resultados de la detección para una de las mejores y la peor configuración. La Figura 4a representa el peor caso. Aunque se detectan un número de cambios muy cercano al real, estos cambios se distribuyen de manera errónea en el registro, es decir, se dan varios falsos negativos y varios falsos positivos, haciendo que la configuración no sea válida. En el caso de la mejor configuración (Figura 4b), a pesar de proporcionar un mayor número de puntos de cambio, únicamente se producen falsos negativos en el cambio entre las modificaciones 3 y 4 (traza 8.000), por lo que el resultado se aproxima más al comportamiento real del proceso.

5. Conclusiones y trabajo futuro

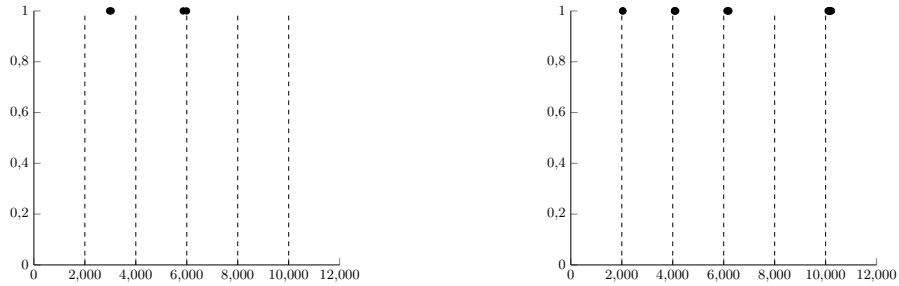
En este artículo hemos presentado una aproximación al tratamiento de *concept drift* en minería de procesos, basada en el uso de técnicas de agrupamiento de trazas, descubrimiento de modelos y análisis secuencial de los registros de

Tabla 1. Falsos positivos detectados en la ejecución de las pruebas sobre los registros parciales. w es el tamaño de ventana. t es el umbral de detección.

	Modelo original	Modificación 1	Modificación 2	Modificación 3	Modificación 4	Modificación 5
$w = 100$; $t = 5\,\%$	0	0	0	0	211	0
$w = 100$; $t = 10\,\%$	0	0	0	5	13	3
$w = 200$; $t = 5\,\%$	0	0	0	0	92	0
$w = 200$; $t = 10\,\%$	0	0	0	1	0	1
$w = 500$; $t = 5\,\%$	0	0	0	0	27	0
$w = 500$; $t = 10\,\%$	0	0	0	1	0	0

Tabla 2. Cantidad de cambios detectados en la ejecución del algoritmo sobre el registro completo. w es el tamaño de ventana. t es el umbral de detección.

	Resultados sobre el log completo				
	$t = 5\,\%$	$t = 10\,\%$	$t = 15\,\%$	$t = 20\,\%$	$t = 25\,\%$
$w = 100$	234	35	112	32	31
$w = 200$	114	17	59	7	10
$w = 500$	44	17	17	6	9
$w = 1000$	29	16	7	6	6
$w = 2000$	17	16	6	6	4



(a) $ws = 2000, t = 25\,\%$ (b) $ws = 200, t = 10\,\%$

Figura 4. Puntos de detección de *concept drift* para una de las configuraciones que mejores resultados proporciona y para una de las que peores. Las líneas verticales indican la localización exacta de los cambios. Los puntos, cambios detectados por el algoritmo.

actividad, que puede servir para mejorar los resultados de los procesos de descubrimiento, evaluación y mejora de modelos a partir de registros que presenten una evolución temporal.

No obstante, ésta se trata de una aproximación inicial cuya eficiencia depende, entre otros factores, de que los modelos descubiertan tengan una completitud cercana a 1, es decir, que describan todo el comportamiento observado en las trazas. Por tanto, como trabajo futuro se plantea la adaptación del algoritmo para que sea capaz de tratar con modelos menos genéricos en los que la completitud puede ser menor y donde la comprobación de la conformidad sea más flexible. Además, también se plantea aplicar tecnologías de análisis masivo de datos con las que mejorar el rendimiento del algoritmo.

Agradecimientos

Esta investigación está soportada por el Ministerio de Economía y Competitividad bajo los proyectos TIN2014-56633-C3-1-R y TIN2015-73566-JIN, y por el European Regional Development Fund (ERDF/FEDER) bajo el proyecto CN2012/151 de la Consellería de Educación de la Xunta de Galicia.

Referencias

1. van der Aalst, W.M.P., et al.: Process mining manifesto. In: BPM 2011 International Workshops on Business Process Management. Volume 99 of Lecture Notes in Business Information Processing., Springer (2012) 169–194
2. van der Aalst, W.M.P., de Medeiros, A.K.A., Weijters, A.J.M.M.: Genetic process mining. In: Proceedings of the 26th International Conference on Applications and Theory of Petri Nets (ICATPN 2005). Volume 3536 of Lecture Notes in Computer Science., Springer (2005) 48–69
3. Vázquez-Barreiros, B., Mucientes, M., Lama, M.: Prodigen: Mining complete, precise and minimal structure process models with a genetic algorithm. Information Sciences **294** (2015) 315–333
4. Weijters, A., van Der Aalst, W.M., De Medeiros, A.A.: Process mining with the heuristics miner-algorithm. Technische Universiteit Eindhoven, Tech. Rep. WP **166** (2006) 1–34
5. Accorsi, R., Stocker, T.: On the exploitation of process mining for security audits: the conformance checking case. In: Proceedings of the ACM Symposium on Applied Computing (SAC 2012), ACM (2012) 1709–1716
6. Munoz-Gama, J.: Conformance Checking and Diagnosis in Process Mining - Comparing Observed and Modeled Processes. Volume 270 of Lecture Notes in Business Information Processing. Springer (2016)
7. Chapela, D., Mucientes, M., Lama, M.: Discovering infrequent behavioral patterns in process models. In: Proceedings of the 15th International Conference on Business Process Management (BPM 2017). Volume 10445 of Lecture Notes in Computer Science., Springer (2017) 324–340
8. Conforti, R., Rosa, M.L., ter Hofstede, A.H.M.: Filtering out infrequent behavior from business process event logs. IEEE Transactions on Knowledge and Data Engineering **29**(2) (2017) 300–314
9. Bose, R.P.J.C., van der Aalst, W.M.P., Zliobaite, I., Pechenizkiy, M.: Dealing with concept drifts in process mining. IEEE Transactions on Neural Networks and Learning Systems **25**(1) (2014) 154–171
10. Carmona, J., Gavaldà, R.: Online techniques for dealing with concept drift in process mining. In: Proceedings of the 11th International Symposium on Advances in Intelligent Data Analysis (IDA 2012). Volume 7619 of Lecture Notes in Computer Science., Springer (2012) 90–102
11. Hompes, B., Buijs, J.C.A.M., van der Aalst, W.M.P., Dixit, P., Buurman, H.: Detecting change in processes using comparative trace clustering. In: Proceedings of the 5th International Symposium on Data-driven Process Discovery and Analysis (SIMPDA 2015). Volume 1527 of CEUR Workshop Proceedings., CEUR-WS.org (2015) 95–108
12. Rokach, L., Maimon, O.: Clustering methods. In Maimon, O., Rokach, L., eds.: The Data Mining and Knowledge Discovery Handbook. Springer (2005) 321–352
13. van der Aalst, W.M.P., Weijters, T., Maruster, L.: Workflow mining: Discovering process models from event logs. IEEE Transactions on Knowledge and Data Engineering **16**(9) (2004) 1128–1142