Ciencias de la Información Vol. 46, No. 1, enero-abril, pp. 73 - 80, 2015

Descubrimiento de Modelos Procesos aplicando los algoritmos Escalador de Colinas y Estrategia Evolutiva

Joan Jaime Puldón
Hassan Cedeño Rondón
Randy Camacho Eng
David Paredes Miranda

En el presente artículo se analiza y modela el problema del descubrimiento de modelos de procesos, con el fin de aplicarle los algoritmos metaheurísticos basados en un punto y en poblaciones. Existen dos razones principales para el creciente interés en mejorar la eficiencia de los algoritmos de descubrimiento en la minería de procesos. Primeramente los sistemas de información que soportan los procesos de negocio en las organizaciones, registran la información detallada de las ejecuciones. Estas contienen un conocimiento potencial para la toma de decisiones. Además de la creciente necesidad de apoyar los procesos de negocios en ambientes competitivos y que cambian rápidamente. Para alcanzar el objetivo planteado se seleccionan los algoritmos metaheurísticos que más se ajusten al problema planteado: el Escalador de Colinas y Estrategia Evolutiva. El aporte principal consiste en la modelación de una primera aproximación al problema de descubrimiento de modelos utilizando las estrategias metaheurísticas.

Palabras clave: escalador de colina; estrategia evolutiva; minería de procesos; metaheurísticas, descubrimiento de modelos

RESUMEN

ABSTRACT

In this paper is analyzed and modeled the problem of discovery process models, in order to apply metaheuristic algorithms based on a point and populations. There are two main reasons for the growing interest to improving the efficiency of the discovery algorithms in process mining. First or all, the information systems, that support business processes in organizations, record the details of the executions. These contain a potential knowledge for decision making. In addition to the growing need to support business processes in competitive and rapidly changing environments. In order to meet the objective of this work that is selected best suited metaheuristics algorithms to the problem: The Hill Climbing and Evolutionary Strategy. The main contribution consists in modeling a first approach to the problem of finding models using metaheuristic strategies.

Keywords: hill climbing; evolutionary strategy; process mining; metaheuristic; descovery models

Introducción

Las empresas en la actualidad cuentan con herramientas de software que dan soporte a sus operaciones. La maduración de tecnologías como sistemas de base de datos y redes de computadoras es capaz de habilitar el soporte automatizado para la ejecución de pasos dentro de un proceso de negocio y su administración como un todo(Rozinat, 2005).La minería de procesos es el campo que trabaja con los procesos de negocio; descubre, monitorea y mejora los procesos reales extrayendo conocimiento de registros de eventos generados por las aplicaciones que lo soportan. La misma se divide en 3

tipos: descubrimiento, conformidad y mejoramiento(Aalst, 2011).

El descubrimiento cuenta con técnicas que proveen diferentes métodos que permiten descubrir modelos de procesos sobre la base de sus trazas de ejecución. Un modelo de proceso puede ser derivado de lastrazas de ejecución, reflejando el comportamiento observado y por consiguiente facilitando el entendimiento de lo que está pasando realmente (Rozinat, 2005).Una técnica de descubrimiento toma un registro de eventos y produce un modelo basado solamente en los datos que existen en el registro. Para muchas organizaciones es sorprendente ver que las técnicas existentes son capaces de descubrir los procesos reales meramente basado en las muestras de ejecución que se almacenan en los registros de eventos (Medeiros, 2011).

Existen herramientas en la actualidad creadas para aplicar minería de procesos a registros de eventos reales. Una de las más utilizadas es el ProM (Günther, 2011), a pesar de que los algoritmos que emplea presentan deficiencias: el *algoritmo-á* presenta problemas con el ruido (que es un comportamiento poco frecuente en las trazas que no sigue el ciclo de vida de un proceso estándar), el funcionamiento incompleto (el registro tiene trazas con una cantidad de eventos insuficiente) y la creación de caminos complejos. Los algoritmos heurístico y de minería difusa se basan en la idea de no incorporar caminos poco frecuentes (Stahl, 2011) (Hee, 2002) (Ribeiro, 2010) (Aalst, 2007) (Adriansyah, 2009). Esto trae como consecuencia que se excluyan del análisis las situaciones excepcionales que ante condiciones extremas muestran su comportamiento . El algoritmo genético, por otro lado, no es eficiente ante registros y eventos muy largos (Aalst, 2011)(Puldón, 2012) (Dogen, 2011) (Medeiros, 2006). Todo lo anterior constituye la principal motivación para buscar nuevas alternativas en el descubrimiento de los modelos de procesos a partir de registros de eventos.

Los algoritmos encargados de descubrir los modelos de procesos tienen en cuenta dos características esenciales para medir la calidad de sus resultados: el ajuste (fitness) y la eficiencia. Muchos de los algoritmos existentes poseen estas propiedades, pero pueden ser mejorados. Por un lado los algoritmos que consiguen encontrar un modelo que se ajusta de forma óptima al registro de eventos, resulta que no es lo suficientemente rápido, asi mismo los que son eficientes, descubren en ocasiones modelos que se desvían totalmente del contenido del registro de eventos (Dogen, 2011) (Günther, 2007). Resulta muy amplio el espacio de búsqueda del problema de descubrimiento de modelos de procesos, pues existen múltiples combinaciones de tareas para modelar el comportamiento del proceso a partir de las trazas. Los algoritmos metaheurísticos son utilizados para mejorar y resolver problemas de este tipo. Es por eso que se decide aplicar los algoritmos escalador de colinas y estrategia evolutiva al problema del descubrimiento de modelos de procesos a partir de registros de eventos, usando diferentes operadores de mutación (Konar, 2000) (López, 2004).

Materiales y métodos

Modelar un problema de optimización combinatoria se compone de datos básicos, codificación, función objetivo y operadores en correspondencia con los algoritmos empleados para encontrar la solución. Comúnmente los datos básicos aportan información necesaria para realizar la evaluación de una solución mediante la función objetivo, o la generación de una solución al aplicar un operador, durante el proceso de búsqueda llevado a cabo por las metaheurísticas. Experimentando con las diferentes variantes del problema y las distintas configuraciones del algoritmo metaheurístico, se explora su conducta para identificar en qué condiciones se obtienen los mejores resultados.

La construcción de los operadores y la codificación se basaron en la experiencia de los autores en la construcción de modelos de procesos y la aplicación de metaheurísticas a problemas de optimización combinatoria, de juegos, formalizados y reales.Para la experimentación se utilizaron dos algoritmos metaheurísticos el EC con los operadores OBMO, SLTAO y SAOO y el EE con mutación en un punto y en dos puntos respectivamente. En cada caso se realizaron 10 000 iteraciones con 20 repeticiones para cada uno de las 5 configuraciones en cada registro de eventos. La construcción de la solución inicial a partir del Modelo Causal utiliza un umbral de dependencia aleatorio con un valor de precisión (k) en la función objetivo de 0,2.

Determinar que metaheurística es mejor emplear en un problema en particular es un tema de investigación abierto, por lo tanto al no contar con referencias que hayan hecho algo similar anteriormente se decidió por dos algoritmos simples, pero cada uno de filosofías diferentes (basado en un punto y en poblaciones de puntos).Una vez realizada todas las pruebas de los algoritmos en cada una de las instancias estudiadas del problema se analiza el comportamiento promedio de ejecución cada metaheurística en cada registro de eventos. El objetivo es determinar cuál metaheurística tiene un mejor comportamiento en cada una de las 5 instancias y sus configuraciones para poder evaluar la influencia de cada elemento en el resultado obtenido.

Resultados y discusión

El principal aporte del presente trabajo se centra en modelar como un problema de optimización combinatoria, descubrir modelos de procesos. Para ello se parte de la conformación del Modelo Causal a partir de un registro de eventos como aparece y luego se modelan la codificación, la función objetivo y los operadores. Se observa finalmente el comportamiento de las metaheurísticas Escalador de Colinas y Estrategia Evolutiva para los elementos modelados. A continuación se brindan los principales detalles de los elementos anteriores.

De la Matriz de Dependencia al Modelo Causal

Los algoritmos metaheurísticos requieren de un estado inicial para comenzar a ejecutarse. Por tanto es necesario construir una solución preliminar a partir de las relaciones de causalidad que se encuentran implícitas en el registro de eventos, este se denota en lo adelante con *L*. Seguidamente se describen los pasos para lograr este objetivo (Medeiros, 2006).

En la etapa de pre-procesado se agrupan las trazas iguales para determinar las trazas tipo y se cuentan la cantidad de instancias de estas que existen en el registro. La igualdad entre las trazas se mide a partir de que la secuencia de eventos sean las mismas. En la Fig. 1 se representa un ejemplo de registro de eventos $L = [\langle A \rangle^x, \ldots, \langle B \rangle^y]$ donde A y B son dos cadenas de actividades

que corresponden a instancias de ejecución del proceso y *x*, *y* son la frecuencia de aparición de las instancias en el registro de eventos (x, y ∈ N).

$L=$
$[\langle a, e\rangle^{5}, \langle a, b,c,e\rangle^{10}, \langle a, b,c,e\rangle^{10}, \langle a, b, e\rangle^{1},$
$\langle a, c,e\rangle^{1}, \langle a, d,e\rangle^{10}, \langle a, d,d,e\rangle^{2},$
$\langle a, d,d,d,e\rangle^{1}]$

Figura 1: Registro de eventos L.

Al registro de eventos se le aplica la técnica de minería heurística descrita en (Aalst, 2011). La Tabla 1 muestra el número de veces que una actividad está directamente seguida por otra, la entrada $d >_L d = 4$ significa que en el registro la actividad *d* es seguida cuatro veces por otra *d* (dos veces en $\langle a, d, d, e\rangle^{2}$ y dos veces en $\langle a, d, d, d, e\rangle^{1}$). Utilizando la Tabla 1 se puede calcular el valor de la relación de dependencia entre cualquier par de actividades.

Tabla 1. Relaciones de dependencia entre las tareas del registro *L*.

$\|>L=\|$	*a*	*b*	*c*	*d*	*e*
a	0	11	11	13	5
b	0	0	10	0	11
c	0	10	0	0	11
d	0	0	0	4	13
e	0	0	0	0	0

Luego se utiliza la siguiente definición (Aalst, 2011):

Definición 2.1: (Medida de Dependencia) Teniendo el registro de eventos *L* donde a y b pertenecen al grupo de eventos del registro. $|a >_L b|$ es el número de veces que *a* es seguido directamente por *b* en L.

$a =>_L b|$ es el valor de la relación de dependencia entre a y b:

$$|a \Rightarrow_L b| = \begin{cases} \frac{|a>_L b| - |b>_L a|}{|a>_L b| + |b>_L a| + 1} & \text{if } a \neq b \\ \frac{|a>_L a|}{|a>_L a| + 1} & \text{if } a = b \end{cases} \quad (1)$$

Aplicandola definición anterior al registro de eventos *L* se obtiene la Tabla 2.

A partir de la Tabla 2 se puede obtener un grafo de dependencia basado en las relaciones de dependencia entre los nodos,

Tabla 2. Medidas de dependencia entre las tareas del registro *L*

$\|\Rightarrow_L\|$	*a*	*b*	*c*	*d*	*e*
a	$\frac{0}{0+1}=0$	$\frac{11-0}{11+0+1}=0.92$	$\frac{11-0}{11+0+1}=0.92$	$\frac{13-0}{13+0+1}=0.93$	$\frac{5-0}{5+0+1}=0.83$
b	$\frac{0-11}{0+11+1}=-0.92$	$\frac{0}{0+1}=0$	$\frac{10-10}{10+10+1}=0$	$\frac{0-0}{0+0+1}=0$	$\frac{11-0}{11+0+1}=0.92$
c	$\frac{0-11}{0+11+1}=-0.92$	$\frac{10-10}{10+10+1}=0$	$\frac{0}{0+1}=0$	$\frac{0-0}{0+0+1}=0$	$\frac{11-0}{11+0+1}=0.92$
d	$\frac{0-13}{0+13+1}=-0.93$	$\frac{0-0}{0+0+1}=0$	$\frac{0-0}{0+0+1}=0$	$\frac{4}{4+1}=0.80$	$\frac{13-0}{13+0+1}=0.93$
e	$\frac{0-5}{0+5+1}=-0.83$	$\frac{0-11}{0+11+1}=-0.92$	$\frac{0-11}{0+11+1}=-0.92$	$\frac{0-13}{0+13+1}=-0.93$	$\frac{0}{0+1}=0$

donde la etiqueta de los arcos representa el valor de la relación (Véase la Fig. 2).

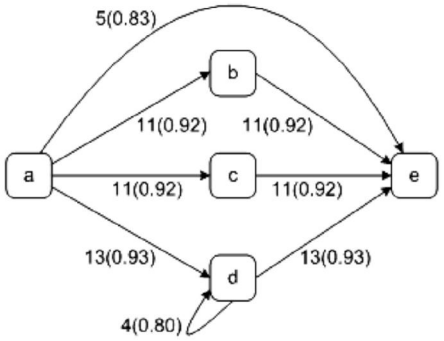

Figura 2: Relaciones de dependencia entre las tareas del registro

Los arcos del grafo de dependencia representados en la Fig. 2 están etiquetados con los valores de frecuencia y la medida de dependencia de la relación entre las actividades. En la construcción de la solución inicial no es necesario obtener el grafo, solo se requiere deducir la Matriz Causal (CM por sus siglas en inglés) a partir de las medidas de dependencia calculadas en el paso anterior. La CM es el resultado de aplicar un umbral o filtro a los valores de la Tabla 2, seleccionando solo aquellos que sean superiores. Por ejemplo si se utiliza el umbral 0.9 se obtiene la CM que aparece en la Tabla 3.

El modelo causal proviene de la representación que se utiliza para las redes causales. La siguiente definición muestra cómo se obtiene(Aalst, 2011):

Definición 2.2:Una red o modelo causal es una *tupla* C = (A, a_i, a_0, D, I, O) donde:

A : es un grupo finito de eventos,
a_i:es el evento inicial,
a_0:es el evento final,
D :son las relaciones de dependencia,
I :son los nodos entrantes por evento,
O :son los nodos salientes por evento.

El objetivo es extraer una Red Causal del registro de eventos *L*. Los nodos de la matriz causal son los elementos del conjunto de eventos *A* que cumplen la condición de tener un valor de dependencia superior al umbral. Las relaciones de la matriz causal constituyen el conjunto de dependencia *D*. En la red causal existe un evento inicial a_i y un evento final a_0 con respecto a la *CM*, de no existir se crearían artificialmente. Para los nodos entrantes y salientes se necesita saber si existen uniones y divisiones (AND y OR respectivamente). Por tanto se toma en consideración que: si un nodo *a* es seguido directamente por los nodos *b* y *c* la misma cantidad de veces, y además *b* sigue directamente *c* la misma cantidad de veces que *c* sigue directamente *b*, entonces se

Tabla 3. Relaciones de entrada y salida utilizando AND y OR.

ACTIVITY	I(ACTIVITY)	O(ACTIVITY)
a	{}	{{b,c},{d},{e}}
b	{a}	{e}
c	{a}	{e}
d	{a}	{{d},{e}}
e	{{b,c},{d},{a}}	{}

tiene un AND de lo contrario es un OR. Partiendo de la definición anterior se obtiene la Tabla 4 donde las llaves {} agrupa los subconjuntos de eventos en unión si se aplican en el primer nivel de precedencia y los que se encuentran en división cuando aumenta la precedencia. Véase el modelo de la Fig.3 que corresponde a la Tabla 3 para un mejor entendimiento de la representación.

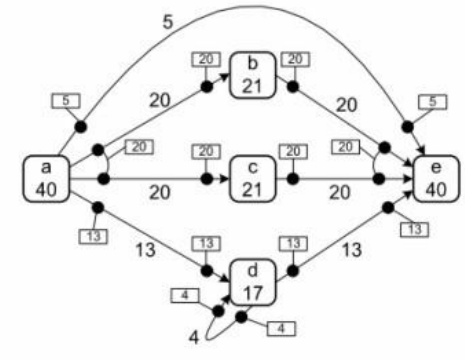

Figura 3. Red Causal correspondiente a la Matriz Causal.

Codificación

Codificar el problema de descubrimiento de modelos de procesos tiene como objetivo representar las relaciones entre los eventos que conforman las trazas del registro. Además permite representar la forma en que se encuentran vinculadas las relaciones (unión o división). La unión sucede cuando dos eventos se ejecutan a la misma vez (el llamado AND) y la división es cuando no dependen uno del otro para ejecutarse (el llamado OR). Se usa una lista de enteros para etiquetar los tipos de eventos y así disminuir el costo computacional de la representación del estado. A continuación se observa una representación basada en las relaciones resultantes de la Tabla 3: (a, b, -2) (a, c, -1) (a, d, -1) (a, e, -1) (b, e, -1) (c, e, -1) (d, d, -1) (d, e, -1).

Como se puede observar en la Tabla 3 hay un total de 5 eventos: a, b, c, d, e. Para la codificación de las relaciones entre las actividades se utiliza una terna. La primera posición refiere la actividad de origen, la segunda a la actividad destino y la tercera establece la manera en que está vinculada con la siguiente relación. Por ejemplo, las relaciones (a, b) y (a, c) están en unión, consecuentemente en la representación se refleja el valor -2 para identificarla. Para establecer cuándo termina la relación de unión se coloca el valor -1. En la Fig. 4 se muestra un ejemplo de codificación que

0	1	-2	0	2	-1	0	3	-1	0	4	-1	1	4	-1	2	4	-1	3	3	-1	3	4	-1

Figura 4. Codificación del Problema.

es generada a partir de la lista de relaciones de la Tabla 3.

Nótese que las posiciones dedicadas a las actividades tienen valores enteros, los que corresponden al índice de la actividad en la lista cargada en memoria como se muestra en la Tabla 4.

Tabla 4. Representación de las tareas cargadas en memoria.

Tareas	A	B	C	D	E
Índices	0	1	2	3	4

Por ejemplo la primera terna representa la relación de dependencia de *a* con *b* R1, la que se encuentra en unión con la relación de dependencia *a* con *c* R2, luego en la tercera posición de R1 está el valor -2. La relación R3 se encuentra en división con el resto de las relaciones que tienen la actividad *a* como origen, consecuentemente en la tercera posición de la terna se coloca el valor -1. Esta codificación tiene como ventaja que a partir de las relaciones se obtiene la representación del modelo. Además los operadores pueden generar nuevas soluciones al cambiar los elementos que interviene en las diferentes posiciones de la terna según su dominio.

Evaluación

El proceso de evaluación es muy importante, porque guía a la metaheurística en qué decisión tomar y hacia dónde dirigirse para lograr la mejor solución. En este caso el proceso se basa en calcular el ajuste (fitness en inglés), a partir de una cadena que representa la codificación. La función de ajuste utilizada está basada en la que fue definida en (Medeiros, 2006), aquella se concibió para utilizarse en modelos de Redes de Petri (Darondeau, 1998) y los elementos que la conforman se calculan al *parsear* cada una de las trazas del registro en el modelo. Como se ha mencionado anteriormente en esta solución se emplearán las redes causales. Por tanto se cambiará la semántica de la fórmula original de la siguiente manera:

El método *allMissingTokens (L, CM)* se reemplazó por el método *MissingEvents (L, CM),* en este último se cuenta la cantidad de veces que no se encuentra habilitada la tarea correspondiente en el modelo según la secuencia de la traza que se está parsendo. Consecuentemente se cambió la función *numTracesMissingTokens (L, CM)* por *numTracesMissingEvents (L, CM)* bajo el mismo análisis. Por otro lado, los métodos *numTracesExtraTokensLeftBehind (L, CM)* y *allExtraTokensLeftBehind (L, CM) no* se consideran en la expresión, pues el modelo de representación seleccionado no tiene en cuenta las señales que no fueron consumidas durante el *parseo* de la traza.

La función de ajuste queda definida de la siguiente forma:

Existen dos funciones parciales de *fitness*: completamiento y precisión. La notación usada en la función parcial de completamiento es la siguiente. *Parsed Events (L, CM)* brinda el número total de eventos en el registro *L* que pueden ser parseados por el modelo causal *CM*. *EventsLog (L)* brinda el número de eventos en el registro. *MissingEvents (L, CM)* indica el número de eventos perdidos en las trazas. *TracesLog (L)* indica el número de trazas en *L*. La función *numTracesMissing Events (L, CM)* indica las trazas donde hay eventos perdidos durante el parseo.

Ajuste Parcial de Completamiento (*complete)*. Teniendo que *L* es un registro de eventos lleno (con al menos una traza) y que *CM* es un modelo causal. Entonces el ajuste parcial *complete* es una función delimitada en el dominio [-»,1] donde 1 es el resultado deseado, y se define como:

$$complete(L, CM) = \frac{ParsedEvents(L, CM) - punishment}{EventsLog(L)} \quad (2)$$

$$punishment = \frac{MissingEvents(L, CM)}{TracesLog(L) - numTracesMissingEvents(L, CM) + 1} \quad (3)$$

La notación usada en la función de ajuste parcial de precisión es la siguiente. *Enabled Events (L, CM)* indica el número de eventos que fueron habilitados durante el parseo del registro *L* por el modelo causal *CM*. *EnabledEvents (L, CM [])* aplica lo dicho en la función *EnabledEvents (L, CM)* a todos los elementos en la población de modelos causalesCM []. La función *max (EnabledEvents (L, CM []))* retorna el valor máximo de eventos habilitados en un modelo de CM [].

Ajuste Parcial de Precisión (*precision*). Teniendo que *L* es un registro de eventos lleno, que *CM* es un modelo causal y que *CM []* es una población modelos causales que contiene a CM. El ajuste parcial de *precision* es una función delimitada en el dominio [0,1] donde 0 es el resultado deseado, y se define como:

$$precision(L, CM, CM[\square]) = \frac{EnabledEvents(L, CM)}{\max(EnabledEvents(L, CM[\square]))} \quad (4)$$

Para definir la medida de *fitness*, se decide que el completamiento es más relevante que la precisión, aunque las dos medidas se tienen en cuenta en su diseño. Fitness (*fitnessCalculate*) considera que *L* es un registro de eventos lleno, *CM* es un modelo causal y *CM []* es una población de modelos causales que contiene a *CM*. Teniendo las funciones parciales de *fitness: complete* y *precision* dadas en las definiciones 2 y 4. Teniendo k que es un número entre 0 y 1. Entonces el *fitnessCalculate* es una función delimitada en el dominio [-»,1] donde 1 es el resultado deseado, y se define como:

$$fitnessCalculate(L, CM, CM[\square]) = complete(L, CM) - k * precision(L, CM, CM[\square]) \quad (5)$$

Operadores

La biblioteca de clases BiCIAM (Calderín, 2013) (Miranda, 2008) contiene algunos operadores que han sido creados para resolver determinadas situaciones y problemas. Dichos operadores se utilizan con el algoritmo poblacional Estrategia Evolutiva (EE) (Calderín, 2013) (Rosete, 2000)(Konar, 2000)(Limonta, 2012). Por otro lado el algoritmo basado en un punto Escalador de Colina (EC) utiliza operadores implementados en esta solución. Se han definido tres operadores, OBMO(Miranda, 2008), SLTAO y SAOO, los dos últimos se implementaron específicamente para el problema en cuestión. Seguidamente se explica la forma en que genera una nueva solución. Observar que la primera fila de las tablas representa los índices de la lista de enteros que constituye el estado y la segunda y tercera fila son los valores del estado en esa posición antes y después respectivamente.

Tabla 5. Operador OBMO.

Índices	0	1	2	3	4	5	6	7	8
antes	0	1	-2	0	**2**	-1	**0**	3	-1
después	0	1	-2	0	**0**	-1	**2**	3	-1

Este operador de Mutación Basado en Orden (OBMO), intercambia dos posiciones escogidas aleatoriamente. En este caso se garantiza que seleccione una posición donde se encuentre un evento válido y no un tipo de relación. De manera que cambia o crea nuevas relaciones entre eventos. En la Tabla 5 se intercambian las posiciones 4 y 6 de la lista, este cambio genera un lazo en la tarea 0 y una relación entre las tareas 2 y 3. Además se observa que las columnas 2, 5 y 8 son las que no pueden ser intercambiadas.

SwapLinkToAndOperator (SLTAO)
Los valores de las posiciones de la lista que son múltiplos de 3 indican la forma en que se vinculan las relaciones entre ellas. Estos valores pueden ser -2 o -1, indicando que las relaciones se vinculan mediante una unión o división respectivamente. El operador selecciona una posición aleatoria múltiplo de 3 y cambia su valor por -2. La Tabla 6 muestra su funcionamiento.

ORxANDSwapAndOrOperator (SAOO)
Este operador funciona combinando el método anterior con una técnica similar. Consiste en escoger una posición aleatoria que sea múltiplo de 3 y cambiar su valor por -2, pero en caso de que el valor de la posición sea -2 entonces se modificaría a -1. Al igual que el anterior, el cambio realizado genera una interpretación diferente del modelo resultante. La Tabla 7 muestra su funcionamiento.

Para el caso del EE se decide utilizar el método de selección por truncamiento. La mutación en uno y dos puntos respectivamente, y como forma de reemplazo el estado-estable.

Experimentos

En la ejecución de los experimentos se utilizan 3 registros de eventos tomados del proceso de declaración de mercancías de una agencia de envío de cargas aéreas. La información recopilada se obtuvo de una única fuente, que de manera centralizada es empleada por dicha agencia para registrar todos los datos relacionados con el despacho de mercancías.

A continuación se da una breve descripción de cada uno de estos registros de eventos:

DeclaraciónDeMercancías3.xes:
el registro de eventos contiene 31417

Tabla 6. Operador de Mutación SLTAO

Índices	0	1	2	3	4	5	6	7	8
antes	0	1	-2	0	2	**-1**	0	3	-1
después	0	1	-2	0	2	**-2**	0	3	-1

Tabla 7. Operador de Mutación SLTAO

.

Índices	0	1	2	3	4	5	6	7	8
antes	0	1	**-2**	0	2	-1	0	3	-1
después	0	1	**-1**	0	2	-1	0	3	-1

instancias o casos con 126918 eventos. También existen 12 tipos de eventos, 131 tipos de trazas y 371 usuarios vinculados. Cada caso posee un mínimo de 1, un promedio de 5 y un máximo de 28 de eventos. En las instancias se encuentra 1 tipo de evento como mínimo, 3 como promedio y 8 como máximo. Las instancias poseen tipos de evento repetidos, pues existen como promedio 5 eventos en cada instancia pero solo 3 son distintos.

DeclaraciónDeMercancías4.xes:
el registro de eventos contiene 33643 instancias o casos con 129801 eventos. Además tiene 130 tipos de trazas y 12 tipos de eventos. Las otras características son iguales al registro anterior.

DeclaraciónDeMercancías2.xes:
el registro de eventos contiene 27925 instancias o casos con 112742 eventos. Además tiene 124 tipos de trazas y 12 tipos de eventos. Las otras características son iguales a la de los dos registros anteriores.

A continuación en las gráficas que se presentan se describe el comportamiento del algoritmo EC con cada operador en los diferentes registros. De manera que el eje x describe las iteraciones por las que pasa el algoritmo, el eje y se corresponde con el valor normalizado de la función objetivo. La Fig. 5(a) muestra los resultados de EC_SLTAO para los tres registros.

Como se puede apreciar en los tres registros DeclaraciónDeMercancías3.xes (DM3), DeclaraciónDeMercanías4.xes (DM4) y DeclaraciónDeMercancías2.xes (DM2) EC_SLTAO tiene un comportamiento similar, encontrando un óptimo local casi al inicio, con resultados alejados de una buena solución, teniendo en cuenta que mientras más se acerquen las soluciones a 1 es mejor. En el caso de DM2 el algoritmo obtiene soluciones más pobres que en DM3 y DM4. Esto demuestra que el algoritmo mantiene un mejor comportamiento en registros más grandes.

En la Fig. 5 (b) se muestran los resultados de EC_SAOO para los diferentes registros de eventos. Como se puede apreciar este algoritmo tiene un comportamiento similar en cada registro. Su desempeño es pobre, debido que las soluciones encontradas están lejos de 1, aunque están cercanas entre sí. El algoritmo asciende rápidamente y luego queda atrapado en óptimos locales en todos los registros desde el inicio de cada corrida. Además se aprecia que en DM2 tiene un mejor resultado. Esto demuestra que el algoritmo mantiene un mejor comportamiento en registros de eventos pequeños. En la Fig. 6(a) se observan los resultados de EC_OBMO para diferentes registros.

El algoritmo se incrementa rápidamente generando soluciones mejores que los algoritmos anteriores. Con los registros DM3 y DM4 se obtienen mejores resultados porque el tamaño de estos aumenta con respecto a DM2 y se puede desempeñar mejor. Se puede observar que al aumentar la cantidad de trazas y a la vez las trazas tipo, se obtienen mejores resultados. Esto se debe a que se puede observar un mayor número de comportamientos diferentes en los registros. En la Fig. 6(b) se muestran los resultados de EE_1P para todos los registros.

Como se puede apreciar este algoritmo tiene

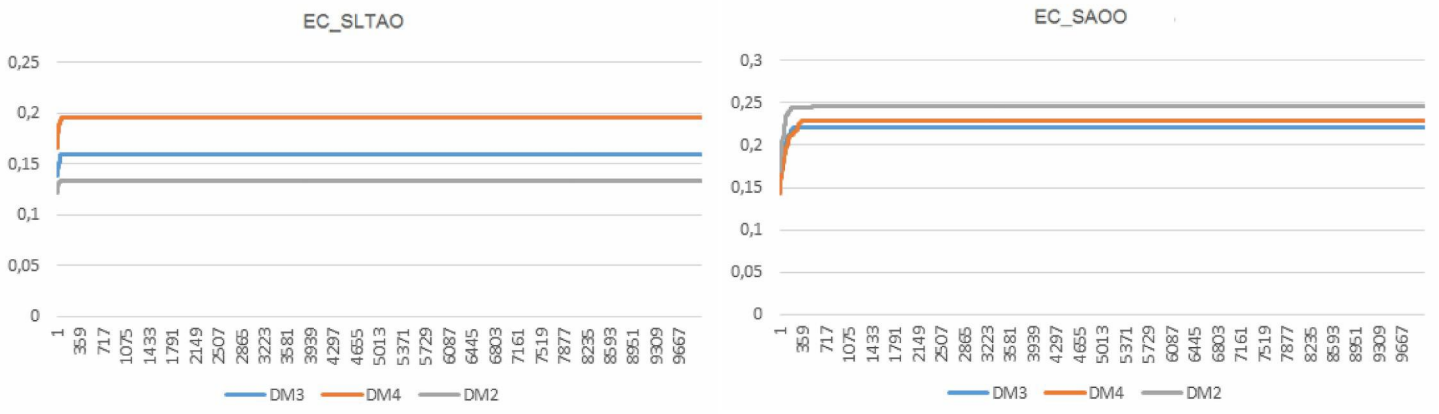

Figura 5. Algoritmo EC_SLTAO (a) Algoritmo EC_SAOO (b).

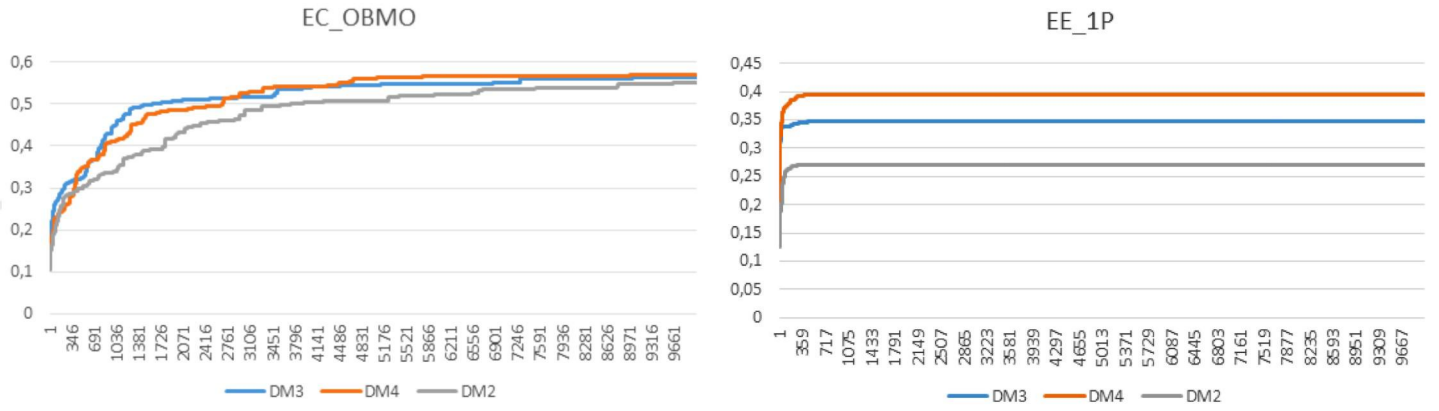

Figura 6. Algoritmo EC_OBMO (a) Algoritmo EE_1P (b).

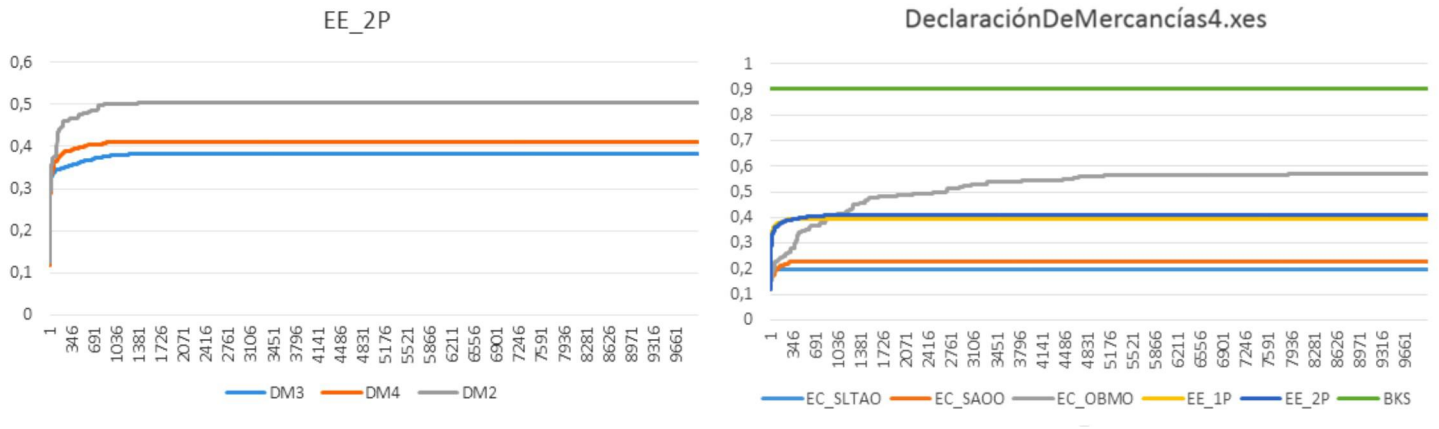

Figura 7. Algoritmo EE_2P (a) Algoritmos EC y EE en DM4 (b).

un comportamiento similar en cada registro, describiéndose con un aumento rápido. Su desempeño es pobre, a causa de que las soluciones encontradas están lejos de 1, aunque se mantienen cercanas entre sí. El algoritmo queda atrapado en óptimos locales en la mayoría de los problemas. Además se aprecia que en DM4 tiene un mejor resultado. En la Fig. 7 (a) se observan los resultados de EE_2P para todos los registros.

El algoritmo aumenta rápidamente y se queda atrapado en un óptimo local en todos los registros. En el registro DM2 se comporta mejor que en DM3 y DM4, esto se debe al pequeño tamaño de este registro. En DM3 y DM4 por el contrario, obtiene los peores resultados debido a que también aumenta el tamaño del problema y no puede desempeñarse mejor.

A continuación se observa la relación del comportamiento de aquellos algoritmos metaheurísticos que fueron utilizados en los experimentos, con las características de los registros utilizados en las pruebas.

En la Fig. 7 (b) se puede observar el comportamiento de los algoritmos EC y EE en el fichero DM4. Se puede concluir que los algoritmos EC_SLTAO y EC_SAOO, son los de más pobre desempeño, alcanzando soluciones lejanas de la BKS y quedando atrapados rápidamente en óptimos locales. Por otra parte con un comportamiento discreto están EE_1P y EE_2P que en este caso encuentran soluciones cercanas entre ellas, pero relativamente lejanas de BKS, EE_2P fue ligeramente mejor que EE_1P. El EC_OBMO obtiene los resultados más cercanos a BKS. De manera general éste último encuentra soluciones similares en registros con características semejantes, mientras que los demás obtienen resultados más discretos.

Conclusiones

Luego de realizar los experimentos, las siguientes conclusiones obtenidas se ajustan a las condiciones y características de los registros de eventos analizados:

• El algoritmo EC_SLTAO se mostró como el de más pobre desempeño, brindando soluciones muy lejanas de la BKS.

• EC_SAOO en la mayoría de los casos tiene el mismo comportamiento, aunque ligeramente mejor que el EC_SLTAO.

• El EE_1P no es aplicable en registros como DM2, sin embargo en registros más grandes obtiene soluciones aceptables.

• Los algoritmos EC_OBMO y EE_2P de manera general encuentran buenas soluciones, aunque EC_OBMO es el que provee un mejor desempeño en cuanto a resultados se refiere.

Bibliografía

Aalst, W.v.d. and K.v. Hee. (2002). Workflow Management Models, Methods, and Systems, T.M.P. Cambridge, Editor.: Massachusetts London, England.

Aalst, W.M.P.v.d. and C.W. Günther. (2007). Finding Structure in Unstructured Processes: The Case for Process Mining, in ACSD '07 Proceedings of the Seventh International Conference on Application of Concurrency to System Design.. p. 7-10.

Aalst, W.M.P.v.d., B.F.v. Dongen, and C. Günther. (2010). ProM: The Process Mining Toolkit. Department of Mathematics and Computer Science Department of Industrial Engineering and Innovation Sciences: Eindhoven University of Technology, Eindhoven, The Netherlands.

Aalst, W.M.P. (2011). Process Mining. Discovery, Conformance and Enhancement of Business Processes. Berlin: Springer-Verlag.

Aalst, W.V.D., A. Adriansyah, and A.K.A.D. Medeiros. (2011). Manifiesto sobre minería de procesos. Springer-Verlag.

Aalst, W.M.P.v.d., V. Rubin, and B.F.v. Dongen. (2011). Process Mining: A Two-Step Approach using Transition Systems and Regions. Eindhoven University of Technology: Eindhoven, The Netherlands.

Aalst, W.M.P.v.d. and C. Stahl. (2011). Modeling Business Processes: A Petri Net Oriented Approach. MIT Press: Cambridge, Massachusetts

Badouel, E. and P. Darondeau. (1998). Theory of regions. Lectures on Petri Nets I: Basic Models. Lecture Notes in Computer Science. W. y. R. Reisig, Springer-Verlag: Berlin. p. 529–586.

Buijs, i.J.C.A.M. (2010) Mapping Data Sources to XES in a Generic Way, in Department of

los procesos de negocio de Mathematics and Computer Science Architecture of Information Systems Research Group. University of Technology Eindhoven: Eindhoven.

Bongen, B.F.v. and A. Adriansyah. (2010). Process Mining: Fuzzy Clustering and Performance Visualization, in Business Process Management Workshops (2009). Berlin Heidelberg, Lecture Notes in Business Information Processing, Springer-Verlag. p. 158-169.

Günther, C.W. and E. Verbeek. (2012). XES Standard Definition. University of Technology Eindhoven: Eindhoven, The Netherlands.

Hernández, R.D. and J.F. Calderín. (2013). Especificaciones Funcionales BiCIAM.: Complejo de InvestigacionesTecnológicasIntegradas (CITI), La Habana, Cuba.

Konar, A. (2000). Artificial Intelligence and Soft Computing, in Behavioral and Cognitive Modeling of the Human Brain. CRC Press: Calcutta, India.

Limonta, L.M. (2012). Comparación de algoritmos metaheurísticos en el Problema de la Planificación de Tareas., in Facultad de Ingeniería Informática. Instituto Superior Politécnico «José Antonio Echeverría»: La Habana.

Medeiros, A.K.A.d. (2006). Genetic Process Mining. TechnischeUniversiteit Eindhoven: Eindhoven, The Netherlands.

Miranda, D.P. and J.F. Caderín. (2008). BIBLIOTECA DE CLASES PARA LAUNIFICACION DE ALGORITMOS METAHEURISTICOS BASADOS EN UN PUNTO, in Centro de Estudios de Ingeniería y Sistemas (CEIS). Instituto Superior Politécnico «José Antonio Echeverría»: Ciudad de La Habana, Cuba.

Puldón, J.J. (2012). Minería de Procesos aplicada al proceso de Declaración de Mercancías en la AGR. ISPJAE: La Habana.

Rosete, A. (2000). Una solución flexible y eficiente para el trazado de grafos basada en el Escalador de Colinas Estocástico, in CEIS., ISPJAE: La Habana.

Rozinat, A.. (2005). Conformance Testing: Measuring the Alignment Between Event Logs and Process Models. TechnischeUniversiteit Eindhoven.

Santana, J.B., C.C. Rodríguez, and F.C.G. López. (2004). Metaheurísticas: una revisión actualizada. Universidad de La Laguna.

Unther, C.W.G. and W.M.P.v.d. Aalst. (2007). Fuzzy Mining – Adaptive Process Simplification Based on Multi-Perspective Metrics, in International Conference on Business Process Management (BPM 2007). Lecture Notes in Computer Science, Springer-Verlag: Berlin. p. 328–343.

Weijters, A.J.M.M. and J.T.S. Ribeiro, (2010). Flexible Heuristics Miner (FHM). Eindhoven University of Technology: Eindhoven, The Netherlands

Recibido: 21 de noviembre de 2014.
Aprobado en su forma definitiva:
17 de febrero de 2015

Joan Jaime Puldón
Facultad de Ingeniería Informática
Instituto Superior Politécnico José Antonio Echeverría, (CUJAE), La Habana, Cuba.
Correo e.: yjaime@ceis.cujae.edu.cu

Hassan Cedeño Rondón
Instituto Superior Politécnico José Antonio Echeverría, (CUJAE), La Habana, Cuba.

Randy Camacho Eng
Instituto Superior Politécnico José Antonio Echeverría, (CUJAE), La Habana, Cuba.

David Paredes Miranda
Instituto Superior Politécnico José Antonio Echeverría, (CUJAE), La Habana, Cuba.
Correo e.: dparedes@ceis.cujae.edu.cu