Universidad de Chile
Facultad de Ciencias Físicas y Matemáticas
Departamento de Ciencias de la Computación

Estimación de esfuerzo de trabajo y planificación de proyectos de desarrollo con restricciones de recursos

Tesis para optar al grado de Magíster en Ciencias, Mención Computación
Memoria para optar al título de Ingeniería Civil en Computación

Pedro Pablo Bustamante Barrera

Profesora guía:
María Cecilia Bastarrica Piñeyro

Miembros de la comisión:
Jorge Muñoz Gama
Pablo González Jure
Sergio Ochoa Delorenzi

Santiago de Chile
2021

Resumen

Una etapa fundamental en el desarrollo de software es la planificación del proyecto, que en general, implica determinar el esfuerzo de trabajo, el orden en que se realizarán las tareas, qué y cómo se distribuirán los recursos y la definición de tiempos y costos de desarrollo para lograr el éxito del proyecto. Se han desarrollado variados modelos y herramientas que buscan apoyar esta etapa.

COCOMO II es uno de las primeras propuestas. Permite, entre otras cosas, estimar el costo de software y esfuerzo de trabajo con una aproximación cada vez mayor a medida que avanza el proceso de desarrollo. Existen otros algoritmos que abordan el problema de la planificación de proyectos con recursos limitados (RCPSP). Dos de estos algoritmos propuestos recientemente serán los que se usen como base para esta tesis. Primeramente se considerará el "Time-line based model for software project scheduling with genetic algorithms" (TLB-SPS). Este utiliza un algoritmo genético, un espacio de búsqueda en tres dimensiones (tareas, empleados y períodos) y un modelo simple de aprendizaje. El otro es el "Ant Colony Optimization for Software Project Scheduling and Staffing with an Event-Based Schedule" (ACO-SPS) que utiliza un algoritmo de optimización ACO con un espacio de búsqueda de dos dimensiones, además de un planificador basado en eventos.

Estas propuestas resuelven el problema de la planificación pero presentan otras dificultades. COCOMO II requiere gran cantidad de información y experiencia en gestión de proyectos para obtener una alta precisión. TLB-SPS, al utilizar un planificador basado en períodos, podría realizar una asignación ineficiente de tareas y recursos, y debido a su espacio de búsqueda, aumentar los tiempos de ejecución. Finalmente, ACO-SPS soluciona estos problemas reduciendo su espacio de búsqueda al utilizar un planificador basado en eventos, pero carece de un modelo aprendizaje para los empleados que lo hace menos flexible.

Este trabajo propone una alternativa para la planificación de proyectos basada en las propuestas anteriores. Para ello se desarrolló el algoritmo EWEHD, que usa información histórica de procesos para estimar el esfuerzo de trabajo, las tecnología y los recursos utilizados en tareas similares. Se construyó un plugin para ProM para su libre utilización. También se extendió ACO-SPS para desarrollar un nuevo algoritmo ACO-SPS++ que incluye un modelo de experiencia y aprendizaje para los empleados que evoluciona a lo largo del proyecto.

Los resultados obtenidos muestran que haciendo uso de registros históricos, se pueden extender los enfoques tradicionales para obtener una mayor precisión, ya que usando minería de datos, se puede no solo estimar los parámetros, sino también conocerlos de manera objetiva para la organización, el tipo de proyectos desarrollados y los recursos utilizados para etapas tempranas de la planificación. Por otro lado, el algoritmo ACO-SPS++, permitió modelar la experiencia y habilidades de cada empleado, adaptándose a la naturaleza evolutiva y dinámica de los proyectos de software, logrando reducir el costo de las planificaciones hasta en un 20 % y ser una alternativa viable para tareas de planificación de proyectos.

Tabla de contenido

1. Introducción

Con el rápido desarrollo de la industria de software, las compañías se están enfrentando a un mercado cada vez más competitivo. El éxito de estas depende en gran medida de hacer la planificación de proyectos y la asignación de recursos eficientemente, para reducir el costo del desarrollo y maximizar el uso de recursos disponibles sin sacrificar la calidad de estos. Esto se dificulta y es más desafiante en proyectos de gran escala, que incluyen gran cantidad de empleados, sueldos que pagar y recursos limitados que administrar.

Para planificar un proyecto de software y realizar la asignación de recursos, el jefe de proyectos o la persona a cargo, necesita estimar la carga de trabajo y costos del proyecto. Existen técnicas y herramientas para realizar estas tareas como COCOMO II, [8] que se puede utilizar para estimar la carga de trabajo, pero debido a la intensa actividad humana y naturaleza evolutiva, podría ser inadecuada para modelar proyectos de software actuales. Otras técnicas como PERT[1] [24], CPM[2] [25] y el problema RCPSP[3] [9] solo consideran la planificación de tareas o la asignación de recursos humanos, pero no en conjunto, es decir, PERT y CPM no consideran la asignación de recursos, y los modelos basados en RCPSP, no toman en cuenta la asignación de recursos humanos y sus habilidades. La utilización de estos métodos, posiblemente deja parte del trabajo en manos del jefe de proyectos y ya que los recursos y habilidades de los empleados influyen significativamente en la eficiencia de la ejecución del proyecto, puede conducir a una ineficiente asignación de recursos o mal rendimiento de la gestión. Otros estudios relacionados con la planificación de proyectos, tales como "Time-line based model for software project scheduling with genetic algorithms" [13] y "Ant Colony Optimization for Software Project Scheduling and Staffing with an Event-Based Scheduler" [14] definen algoritmos que consideran ambos componentes de manera conjunta logrando excelentes resultados. Si bien el primero de ellos supera las desventajas de RCPSP, surgen dos problemas. Primero, debido a que solo considera la dimensión temporal para iterar en la planificación, y no las tareas, el plan producido por el modelo puede asignar dos grupos de empleados completamente diferentes a una misma tarea en diferentes períodos. Como resultado, la tarea puede ser implementada de una manera inapropiada e ineficiente. Segundo, como el modelo del algoritmo considera tres variables, tiempo, empleados y tareas, el tamaño de la representación de una planificación corresponde a una matriz de tres dimensiones, por lo que el espacio de búsqueda aumenta significativamente, aumentando los tiempos de ejecución. Por otro lado, el segundo algoritmo superó estas deficiencias mejorando los resultados obtenidos, utilizando un planificador basado en eventos y utilizando un algoritmo *Ant Colony Optimization Approach*, que dada su naturaleza heurística, si bien es posible que no se logre el óptimo absoluto,

[1] Program evaluation and review technique.
[2] Critical path method.
[3] Resource-constrained project scheduling problem.

los tiempos de búsqueda se pueden reducir enormemente, o al menos poder tener un trade-off entre tiempo de procesamiento y optimalidad de la solución. El problema es que no considera la experiencia de los empleados y un modelo de entrenamiento como lo hace el primer algoritmo.

Existen compañías en donde sus proyectos siguen procesos similares en muchos aspectos, sin embargo a veces no es posible apoyarlas desde un mismo sistema TI, debido a que cada una de ellas posee características únicas que necesitan ser mantenidas. Con el fin de proporcionar a los diseñadores de sistemas TI una metodología unificada y extensible para capturar los comportamientos de los sistemas mediante registros de eventos y flujos de eventos se difinió el estándar XES [5]. Cuando los sistemas hacen uso de este estándar en el desarrollo de un proyecto, cada acción de un proceso es registrada en un conjunto de datos contenidos en un log estandarizado. Gracias a esto, se tiene información clara y estructurada del funcionamiento de las compañías, lo que permite dar mejor soporte y mantenimiento a los procesos generando oportunidades de mejora.

Por último, resultados de un estudio [31] el cual buscaba identificar taxonomías de los principales métodos utilizados para estimar esfuerzos de desarrollo y planificación de proyectos, indicaron que gran parte de los artículos estudiados consideran la utilización de modelos paramétricos/no paramétricos y la participación de un experto en el proceso. Esto nos lleva considerar el desarrollo de nuevas herramientas que permitan ayudar al jefe de proyectos/expertos en la toma de decisiones en el proceso de estimación del esfuerzo de trabajo para proyectos de desarrollo. Por ejemplo, usando minería de procesos sobre un log estandarizado, el cual posee información de cada tarea o evento dentro de un proceso, no solo se podrían estimar parámetros sino conocerlos de manera objetiva para la organización propiamente tal, el tipo de proyectos desarrollados, los desarrolladores de esa empresa y la tecnología usada.

1.1. Solución propuesta

En el año 2013 se publicó el paper titulado “Ant Colony Optimization for Software Project Scheduling and Staffing with an Event-Based Scheduler” [14], desde ahora ACO-SPS. Este presenta un nuevo enfoque al desarrollar un modelo flexible y eficaz para la planificación de proyectos de software, utilizando un planificador basado en eventos y un algoritmo de optimización basado en el comportamiento de las colonias de hormigas, desde ahora ACO.

Un tema importante en el desarrollo fue la utilización del modelo COCOMO II para estimar el esfuerzo de trabajo de una tarea, el cual requiere gran cantidad de información para su ejecución y experiencia en gestión de proyectos para mejorar su precisión.

Por otro lado, uno de los aspectos pendientes del trabajo realizado fue considerar en el algoritmo, la experiencia del empleado y un modelo de entrenamiento, el cual si fue utilizado por K. Chang en su trabajo “Time-line based model for software

project scheduling with genetic algorithms" [13], desde ahora TLB-SPS.

Esta propuesta de tesis plantea la creación del algoritmo EWEHD (siglas en inglés de "Estimation of work effort based on historical data"), que utilizando minería de datos sobre información histórica de logs de proyectos, ser un complemento y ayuda para enfoques más tradicionales en tareas de estimación del esfuerzo de trabajo en proyectos. Además busca mejorar el trabajo realizado en el algoritmo ACO-SPS, es decir, integrar la experiencia del empleado y un modelo de entrenamiento como nuevas variables para la generación de la planificación, lo cual permitirá hacer el algoritmo más exhaustivo, flexible y más apegado a la realidad de cada proyecto de una compañía.

1.2. Objetivos

Los objetivos planteados para el desarrollo de este trabajo de tesis se presentan a continuación. Se incluye tanto el objetivo general como los objetivos específicos.

1.2.1. Objetivo general

El objetivo general consiste en desarrollar el algoritmo ACO-SPS++, el cual extiende el algoritmo ACO-SPS, incluyendo las variables de experiencia y un modelo de entrenamiento para los empleados, que permitirán hacer de este, un algoritmo más cercano a la realidad, adaptándose a la naturaleza evolutiva y dinámica de los proyectos de software actuales a través de la planificación.

Por otro lado se desarrollará el algoritmo EWEHD para determinar recursos, conocer las tareas, y estimar el esfuerzo de trabajo requerido para la ejecución de estas, utilizando registros históricos de ejecución de procesos. Se pretende que este algoritmo disminuya los tiempos de ejecución, mejore la precisión predictiva y la usabilidad, para luego usar esta información como ayuda para expertos, entregando conocimiento objetivo de los proyectos, recursos y organización, para tareas de estimación de esfuerzos de trabajo.

1.2.2. Objetivos específicos

1. Obtener y procesar logs de ejecución de procesos bajo el estándar XES [5], para obtener información referente al orden y tiempo de ejecución de tareas.

2. Diseñar e implementar el algoritmo EWEHD que utilice los datos procesados para estimar el esfuerzo de trabajo requerido para tareas que forman parte de un proyecto en unidades de persona-tiempo.

3. Desarrollar un plugin utilizando la implementación del algoritmo EWEHD.

4. Comparar la herramienta COCOMO II con el algoritmo EWEHD, considerando el tiempo y la precisión predictiva de la estimación, en relación a la cantidad de información requerida para su ejecución y la presentación de los resultados.

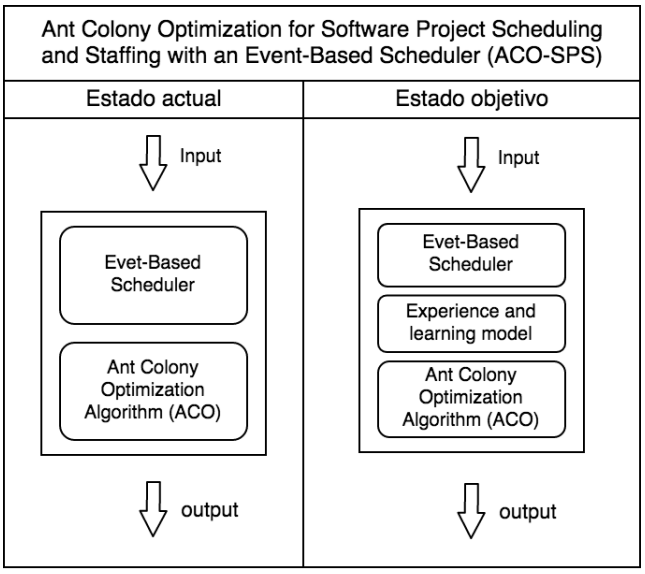


Figura 1: Diagrama que explica el estado actual de ACO-SPS y el propuesto en el objetivo.

5. Diseñar e implementar el algoritmo ACO-SPS++, que corresponde al algoritmo ACO-SPS incluyendo los parámetros de experiencia del empleado y un modelo de entrenamiento.

6. Comparar algoritmo ACO-SPS y ACO-SPS++, considerando los resultados obtenidos de los costos al evaluar la función objetivo y los tiempos de ejecución.

1.3. Metodología

1.3.1. Planteamiento y diseño

La metodología a seguir para cumplir los objetivos es la siguiente.
Para el diseño e implementación del algoritmo EWEHD y su plugin respectivo:

- Adquirir datos de logs de ejecución de procesos [21] [20] [11] desde 4TU.ResearchData [4], el cual provee datos estandarizados, seguros y bien documentados.

- Estudiar los datos (clasificadores, trazas, eventos y recursos) a utilizar para validar el algoritmo, seleccionar un subconjunto de datos apropiado y realizar el procesamiento necesario para poder ejecutar el algoritmo sobre ellos.

- Diseñar e implementar el algoritmo EWEHD que permita en base a los datos procesados, estimar el esfuerzo de trabajo requerido de las tareas de un proyecto con características o tareas similares y el éxito se medirá en base a la mejora en el desempeño, usabilidad y precisión en la estimación del trabajo requerido para las tareas de un proyecto comparado con la herramienta COCOMO II.

[4] `https://data.4tu.nl`

Además si el formato y las unidades de los valores obtenidos son compatibles con los requeridos por el algoritmo ACO-SPS++.

- Diseñar e implementar plugin para ProM [5], que utilizando el algoritmo EWEHD y logs de ejecución, permita obtener la estimación del esfuerzo de trabajo requerido por las tareas que forman parte de un proyecto y cuyo éxito se medirá en base a la correcta importación del plugin y despliegue de los resultados en el programa ProM utilizando el estándar o estructura sugerida por este.
- Diseñar y ejecutar un experimento que permita evaluar y comparar los modos de ejecución del algoritmo EWEHD en base a tiempos de estimación y ejecución, precisión predictiva y usabilidad.

Para el diseño e implementación del algoritmo ACO-SPS++, como mejora del algoritmo ACO-SPS presentado en la publicación de N. Chen y J. Zhang [14]:

- Estudiar y analizar material existente: publicación "Ant Colony Optimization for Software Project Scheduling and Staffing with an Event-Based Scheduler" [14], documentación y bibliografía relacionada.
- Implementar algoritmo ACO-SPS++ integrando las variables o parámetros de experiencia del empleado y un modelo de entrenamiento.
- Diseñar y ejecutar experimentos que permitan comparar y evaluar la ejecución del nuevo algoritmo ACO-SPS++ con el algoritmo ACO-SPS, y cuyo éxito se medirá en base a los costos obtenidos, eficiencia y tiempos de ejecución.

1.4. Contribuciones

Los aportes entregados por el trabajo una vez finalizado son:

- Algoritmo EWEHD: Validado y documentado, lo que permitirá utilizarlo sobre proyectos reales, con el objetivo de optimizar el cálculo del esfuerzo requerido para un proyecto.
- Plugin ProM: Validado y documentado, permitiendo su uso en la aplicación ProM. Posibilidad de ser agregado al repositorio oficial de plugins de ProM.
- Algoritmo de optimización ACO-SPS++: Validado y documentado, lo que permitirá utilizarlo en proyectos reales como alternativa a herramientas existentes.
- El trabajo de diseño, implementación y validación de los algoritmos es material de publicación científica debido a su potencial como avance en el estado del arte.

[5] `http://www.processmining.org/prom/start`

2. Marco teórico

Para el desarrollo del algoritmo EWEHD se utilizan logs de eventos de procesos internos de compañías, contenidos en archivos bajo el formato XES, para luego integrarlo como plugin en el *framework* ProM, haciendo uso de su arquitectura extensible. Por otro lado, el algoritmo ACO-SPS++ utiliza como base el algoritmo ACO-SPS y como guía, el modelo de entrenamiento y experiencia utilizado en TLB-SPS.

Los siguientes conceptos se describirán con más detalle en la siguiente sección:

- Estándar XES.
- COCOMO II.
- ProM.
- Modelo de aprendizaje y experiencia del algoritmo TLB-SPS.
- Algoritmo ACO-SPS.
- Modelos de aprendizaje.

2.1. XES: Extensible Event Stream

El estándar XES [5] define una gramática para un lenguaje basado en etiquetas cuyo objetivo es proporcionar a los diseñadores de sistemas de información una metodología unificada y ampliable para capturar comportamientos de sistemas mediante registros y flujos de eventos. En este estándar se incluyen un esquema XML que describe la estructura de un registro/flujo de eventos XES y otro que describe la estructura de una extensión de dicho registro/flujo. Además, en este estándar se incluye una colección básica de los llamados prototipos de extensión XES que proporcionan semántica a ciertos atributos registrados en el registro/flujo de eventos.

Una instancia de XES corresponde a un registro de eventos basado en archivos o un flujo de eventos formateado que se puede usar para transferir datos controlados por eventos de una manera unificada y extensible desde un primer actor a un segundo actor. Por lo general, el primer actor será el sitio que generará estos datos impulsados por eventos (por ejemplo, sistemas de flujo de trabajo, sistemas de manejo de casos, sistemas de adquisición, dispositivos de rayos X, y hospitales), mientras que el segundo actor será el que analice estos datos (por ejemplo, por científicos de datos y/o sistemas de software avanzados). Como resultado de este estándar, si los datos del evento se transfieren utilizando la sintaxis como se describe en este estándar, su semántica será bien entendida y clara para ambas partes.

Cabe destacar que existen aplicaciones de tipo ETL [26] que permiten extraer, transformar y cargar desde archivos en otros formatos, como Excel, o diferentes fuentes, como bases de datos, a archivos de logs en formato XES.

XES toma mucha inspiración y algunos conceptos bien probados de MXML [30], estándar previo para el registro de eventos, el cual utiliza para definir la estructura general de un registro de eventos: un log (correspondiente a un proceso) que contiene un conjunto de trazas (es decir, instancias de ejecución específicas), que a su vez contienen una secuencia de eventos y cada uno de estos tres conceptos puede contener un conjunto arbitrario de atributos. Por otro lado, XES también es radicalmente diferente en algunos aspectos como lo es en el tipo de información que puede (o qué no) contener un registro de eventos, que además de cadenas (string) los atributos también pueden contener valores de fecha, enteros o booleanos y el uso de extensiones para adjuntar semántica a los datos. El metamodelo de datos para XES, como un diagrama de clase UML 2.0, tiene este aspecto (figura 2):

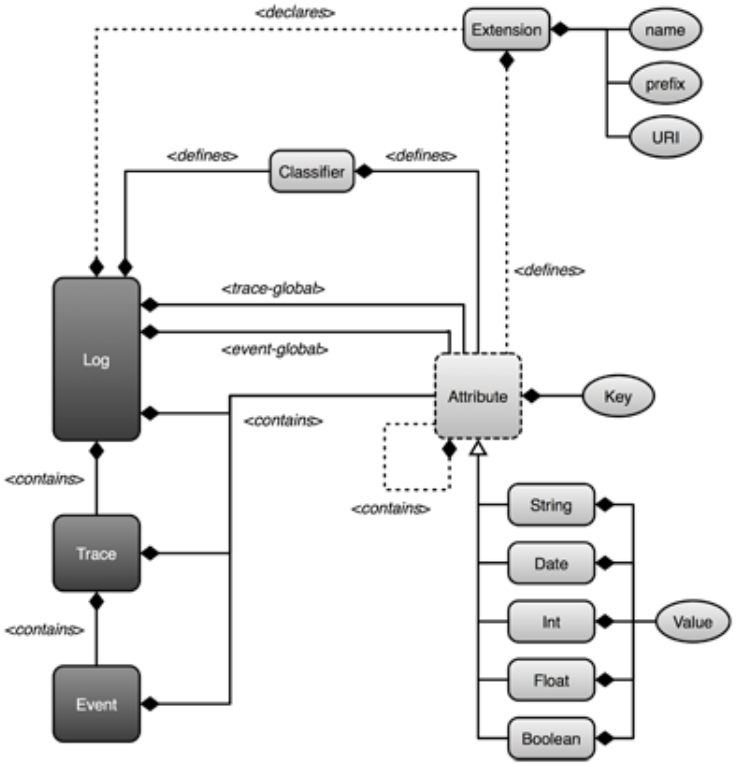


Figura 2: Diagrama del metamodelo de datos para XES [5].

Para un mejor entendimiento del diagrama, se describen los principales componentes:

1. **Log**: En el nivel superior hay un objeto log, que contiene toda la información de las trazas y eventos relacionados con un proceso específico. Ejemplos de procesos son: manejo de reclamos de seguros, usando una máquina de rayos X compleja o navegando en un sitio web. El nombre de etiqueta para el objeto log en el XML de XES es *<log>*.

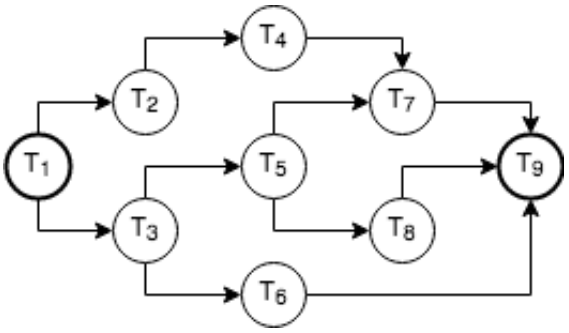


Figura 3: Ejemplo de flujo de un proceso. Sea T_1 tarea o evento inicial, T_9 tarea o evento final y $T_2, T_3, T_4, T_5, T_6, T_7, T_8$ tareas o eventos intermedias.

2. **Traza**: Un log puede contener un número arbitrario (podría ser vacío) de trazas. Cada traza corresponde a una secuencia de eventos que ocurrieron durante la ejecución de una instancia específica o caso de un proceso. Ejemplos de una traza son: un reclamo sobre un seguro, un examen en el que se emplea un máquina de rayos X o una visita a un sitio web por un usuario específico. Otro ejemplo se puede observar en las posibles trazas de un flujo de proceso (figura 3) mostradas en la figura 4. Los eventos se definen dentro de las respectivas etiquetas *<trace>*, que a su vez son elementos secundarios de la etiqueta *<log>*.

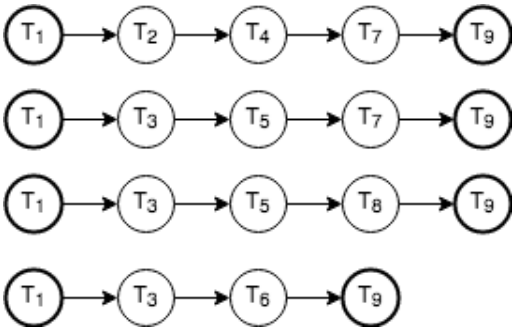


Figura 4: Ejemplo de trazas posibles de proceso descrito en figura 3.

3. **Evento**: Cada traza contiene un número arbitrario (puede ser vacío) y una secuencia particular de eventos. Cada evento representa una actividad granular

que se ha observado u ocurrido durante la ejecución de un proceso. Como tal un evento no tiene duración. Ejemplos de un evento son: se ha completado el registro de la información personal del cliente en la base de datos, la máquina de rayos X toma una fotografía o una imagen ha sido descargada por el navegador web. Los eventos se definen dentro de las respectivas etiquetas *<event>*, que son elementos secundarios de la etiqueta *<trace>*.

4. **Atributo**: Un objeto log, traza o evento no contienen información por sí mismos. Ellos solo definen la estructura del documento. Toda la información en un log de eventos es almacenada en *atributos*. Los atributos describen a su elemento padre (log, traza, etc.). Todos los atributos tienen un llave (*key*) de tipo texto (*string*) y cuyo valor (*value*) puede ser de tipo texto, fecha, entero y con punto flotante o booleano. Cada objeto puede contener una cantidad arbitraria de atributos.

   Además se usaron otros atributos del objeto log que permiten conocer la cantidad de eventos y trazas contenidas en el log.

5. **Clasificador**: El formato XES hace que la clasificación de eventos sea configurable y flexible, al introducir el concepto de clasificadores de eventos. Un clasificador de eventos asigna a cada evento una identidad, lo que lo hace comparable a otros eventos (a través de su identidad asignada). Los clasificadores se definen a través de un conjunto de atributos, de los cuales se deriva la identidad de clase de un evento. En su forma más simple, un clasificador de eventos se define por un atributo, y el valor de ese atributo generaría la identidad de clase de un evento.

   Los clasificadores de eventos se definen por el log, y puede haber un número arbitrario de clasificadores para cada documento.

   Los clasificadores de eventos se definen dentro de las respectivas etiquetas *<classifier>*, que son elementos secundarios de la etiqueta *<log>*
   en un documento.

   El clasificador utilizado en el trabajo realizado es el siguiente *<classifier name="concept:name" keys="concept:name"/>* el cual permite obtener el nombre de un evento o traza.

6. **Extensión**: El estándar XES no define un conjunto específico de atributos por log, traza o evento. Como tal, la semántica de los atributos que contienen estos elementos es ambigua, lo que dificulta la interpretación de esos datos. Esta ambigüedad se resuelve con el concepto de extensiones en XES. Una extensión define un conjunto de atributos en cualquier nivel de la jerarquía de registro XES (log, traza o evento). Al hacerlo, proporciona puntos de referencia para interpretar estos atributos (y, por lo tanto, sus elementos principales). Por lo tanto, las extensiones son principalmente un vehículo para unir la semántica

a un conjunto de atributos definidos por elemento. El nombre de las etiquetas XML de XES para las extensiones se declaran con *<extension>*, que es una etiqueta secundaria de la etiqueta *<log>*.

2.2. Minería de procesos

Los procesos son una parte integral del mundo de hoy, impulsan los servicios y las funciones internas de las empresas, los organismos gubernamentales y las organizaciones de todo el mundo. Si bien hay muchos sistemas disponibles para respaldar la ejecución de tales procesos, las prácticas actuales para monitorear y analizar esta ejecución en la realidad organizacional son limitadas y posiblemente insuficientes para hacer una mejora continua. La minería de procesos (process mining) [29] puede llenar ese vacío, proporcionando medios avanzados para el análisis y monitoreo de procesos de la vida real.

La investigación en minería de procesos se ocupa de la extracción de conocimiento sobre un proceso y de sus registros de ejecución. La minería de procesos se enfoca en obtener información sobre varias perspectivas, como la perspectiva del proceso (o flujo de control), el rendimiento, los datos y la perspectiva organizacional.

Con la misión de convertirse en la plataforma de minería de procesos estándar y proporcionar un *framework* para algoritmos de minería de procesos a usuarios y desarrolladores, fácil de usar y extender, se crea ProM, que corresponde a una aplicación extensible que admite una amplia variedad de técnicas de minería de procesos en forma de plugins. Es una plataforma independiente, implementada en Java y se puede descargar de forma gratuita [6].

2.3. COCOMO II

COCOMO II, es un modelo matemático de estimación que se encuentra en la jerarquía de modelos de estimación de software con el nombre de COCOMO, por Modelo Constructivo de Coste (Constructive Cost Model), siendo uno de los modelos de estimación de costo de software más utilizado y estudiado en la industria. Incluye tres modelos, cada uno ofrece un nivel de detalle y aproximación, cada vez mayor, a medida que avanza el proceso de desarrollo del software:

- **Composición de Aplicación**: utilizado en desarrollos de software durante la etapa temprana de prototipación, evaluación del rendimiento, y la evaluación de la madurez de la tecnología. Se usan Puntos Objeto para estimar el tamaño del software.
- **Diseño Temprano**: utilizado una vez que se han estabilizado los requisitos y arquitectura básica del software. Se evalúan las alternativas de hardware y

[6] `www.promtools.org`

software de un proyecto. Se usan Puntos Función para estimar el tamaño y el uso de un número reducido de factores de costo.

- **Post-Arquitectura**, se aplica en la etapa de desarrollo, después que se define la arquitectura del sistema y en la etapa de mantenimiento. Se usan Puntos Función y/o Líneas de Código para estimar el tamaño, con modificadores que contemplan el reuso, con y sin traducción automática y el "desperdicio"(breakage). Además hace uso de un conjunto de 17 atributos, denominados factores de costo, que permiten considerar características del proyecto referentes al personal, plataforma de desarrollo, etc., que tienen injerencia en los costos. Finalmente también utiliza 5 factores que determinan un exponente que incorpora el concepto de deseconomía y economía de escala.

Cada modelo define el esfuerzo necesario para concretar el proyecto de desarrollo de software en persona mes (PM) utilizando las siguientes fórmulas.

Composición de Aplicación usa la siguiente fórmula (1):

$$PM = NOP/PROD \tag{1}$$

siendo NOP (2), los nuevos puntos objeto y $PROD$, la productividad promedio determinada a partir del análisis de datos de proyectos.

$$NOP = OP * (100 - \%reuso)/100 \tag{2}$$

siendo OP (Puntos Objeto), tamaño del software expresado en Puntos Objeto, $\%reuso$, porcentaje de reuso que se espera lograr en el proyecto.

Diseño Temprano usa la siguiente fórmula (3):

$$PM_{estimado} = A * KSLOC^B * \prod_{i=1}^{7} EM_i \tag{3}$$

siendo A una constante que captura los efectos lineales sobre el esfuerzo de acuerdo a la variación del tamaño, $KSLOC$ el tamaño del software expresado en miles de líneas de código fuente, B factor exponencial de escala (economías y deseconomías) y EM_i a factores de costo que tienen un efecto multiplicativo sobre el esfuerzo, llamados Multiplicadores de Esfuerzo.

Post-Arquitectura usa la fórmula (4):

$$PM_{estimado} = A * KSLOC^B * \prod_{i=1}^{17} EM_i \tag{4}$$

que a diferencia del anterior, este hace uso de 17 factores de costo.

Cabe destacar que todos los parámetros, estimaciones y factores deben ser escogidos por la persona encargada bajo su experiencia y visión del proyecto a desarrollar lo que puede generar resultados distintos si el modelo es utilizado por distintos analistas.

Para más detalles de cada fórmula, los valores que puede tomar cada parámetro y los factores de costo, revisar la definición del modelo COCOMO II [7].

3. Trabajo relacionado

Se presentará una descripción general de los algoritmos TLB-SPS y ACO-SPS, además del modelo de entrenamiento y experiencia de este último. Ambos son mencionados y utilizados en la propuesta.

3.1. Time-line based model for software project scheduling with genetic algorithms [13]

Publicado el año 2008 por Carl K. Chang, Hsin-yi Jiang, Yu Di, Dan Zhu y Yujia Ge. Allí proponen un nuevo modelo de tareas respecto a su trabajo anterior [12], capaz de simular de modo más realista situaciones referentes a la administración de proyectos, utilizando un algoritmo genético para asignar tareas a empleados manteniendo una eficiente utilización de recursos.

En el modelo la representación del problema consiste en:

- Un grafo TPG[7] $= (V, E)$ donde los vértices representan las tareas y las aristas la precedencia de las tareas. Cada tarea está asociada a un esfuerzo estimado y las habilidades requeridas para realizarla.
- Una base de datos D_{emp} con la información de las habilidades, experiencias y salarios de los empleados.
- Una *fitness function* (función objetivo) requerida por el algoritmo genético y utilizada en cada unidad de tiempo.

La nueva representación corresponde a un arreglo ortogonal de 3 dimensiones, una dimensión para las tareas, otra para los empleados y finalmente el tiempo, como nueva dimensión de la representación, permitiendo mostrar el esfuerzo de cada empleado aplicado a cada tarea en cada unidad de tiempo de la planificación.

Las ventajas o nuevas posibilidades de agregar esta nueva dimensión es que permite representar la reasignación de empleados, aprendizaje, suspensión y reanudación de tareas, permitiendo además actualizar parámetros a través del tiempo. Por ejemplo, las habilidades y experiencias de los empleados se traduce en cómo estas evolucionan a través de la realización de tareas durante el desarrollo del proyecto.

Finalmente el algoritmo genético hace uso de la *fitness function*, la representación ortogonal y heurísticas para encontrar la solución cercana al óptimo, la asignación con mínimo costo.

El modelo de experiencia utilizado en el algoritmo define un nivel de experiencia inicial a nivel de tareas para cada empleado y define dos modelos de aprendizaje, uno para las habilidades y otro para la experiencia, los cuales permiten que estos

[7] Task Precedence Graph

atributos aumenten a lo largo de la planificación del proyecto y en consecuencia más aptos para realizar futuras tareas.

Se hará uso del modelo de experiencia y el enfoque de los modelos de aprendizaje para integrarlos en el nuevo algoritmo ACO-SPS++.

3.2. Ant Colony Optimization for Software Project Scheduling and Staffing with an Event-Based Scheduler [14]

Publicado el 2013 por Wei-Neng y Jun Zhang, donde proponen una nueva versión del enfoque anterior. Para evitar grandes espacios de búsqueda, presentan una planificación de tareas en base a eventos y un algoritmo de optimización basado en colonias de hormigas (ACO).

El modelo presentado considera el problema de asignación de recursos humanos y programación de tareas.

- Empleados: Vector $i = (1, 2, ..., m)$ de empleados involucrados en el proyecto. Cada empleado posee información de salario, horas de trabajo y una lista de habilidades.
- Tareas: Similar al modelo anterior, se cuenta con un grafo $TPG = (V, E)$ donde los vértices representan las tareas y las aristas la precedencia de las tareas. Cada tarea está asociada a un esfuerzo estimado, las habilidades requeridas para realizarla, máximo número de empleados, plazo y sanciones. Cabe destacar que cada tarea debe tener al menos un empleado asignado.

La principal diferencia con el modelo anterior, es la eliminación de la línea de tiempo, manteniendo así la representación del problema en dos dimensiones, tareas y empleados, y utilizando el planificador basado en eventos, es decir, solo se calcula o ajusta el modelo si se cumple alguna de estas situaciones o eventos:

- Comienzo del proyecto.
- Algún empleado se une o deja el proyecto.
- Alguna tarea finalizó y los recursos asociados están disponibles.

Esto permite por un lado mantener la información suficiente para el modelado de las tareas y reduce el espacio de búsqueda del algoritmo.

En lo que respecta al algoritmo ACO, la idea principal es simular el comportamiento de búsqueda de las hormigas, es decir, dejar químicos ($feromonas$) en el camino hacia la comida, el cual sirve como medio de comunicación para búsquedas futuras de otras hormigas. El algoritmo ACO trabaja enviando un grupo de hormigas artificiales para construir soluciones al problema iterativamente.

Este tipo de algoritmos se pueden generalizar en la repetición de estos tres procedimientos:

- Construcción de la solución: Durante cada iteración del algoritmo, un grupo de hormigas trabaja para construir una solución al problema en base a heurísticas y feromonas, siendo estas últimas registros de experiencias de búsquedas pasadas de hormigas que guían a las siguientes a tomar decisiones.
- Administración de las feromonas: Junto con la construcción de la solución, el valor de las feromonas es actualizado de acuerdo al desempeño de las soluciones construidas por las hormigas.
- Acciones demonio: Opcionales. Son operaciones centralizadas que no pueden ser realizas por una hormiga. Se utilizan para mejorar el desempeño. Un ejemplo es el *procedimiento de búsqueda local*.

Relacionando estos tres procedimientos con el propuesto en la publicación, se resume en:

- Construcción de la lista de tareas que define el orden en que las tareas serán procesadas y la matriz de asignación de empleados que define la cantidad de horas de cada empleado para cada tarea del proyecto.
- Las feromonas son valores que dependen de los salarios, horas de trabajo y tiempo de desarrollo. Los valores que se actualizan en cada iteración son las feromonas asociadas a cada componente seleccionado para cada asignación de un empleado en una tarea.
- Se incluyeron 2 procedimientos:
  - Mutación de la lista de tareas: Elegir una tarea y un destino al azar. Luego mover dicha tarea hacia el destino tanto como sea posible sin violar la precedencia de tareas definida por el grafo.
  - Mutación de matriz de asignación de empleados: Elegir una tarea y un empleado asignado a esa tarea al azar. Las horas de trabajo realizadas por ese empleado se reinician a 0. Luego se elige otro empleado al azar que no haya sido asignado para la tarea y se registran sus horas de trabajo en $25\,\%nh, 50\,\%nh, ..., maxh_u$, donde nh corresponse a las horas de trabajo en un mes y $maxh_u$ al máximo de horas de trabajo en un mes.

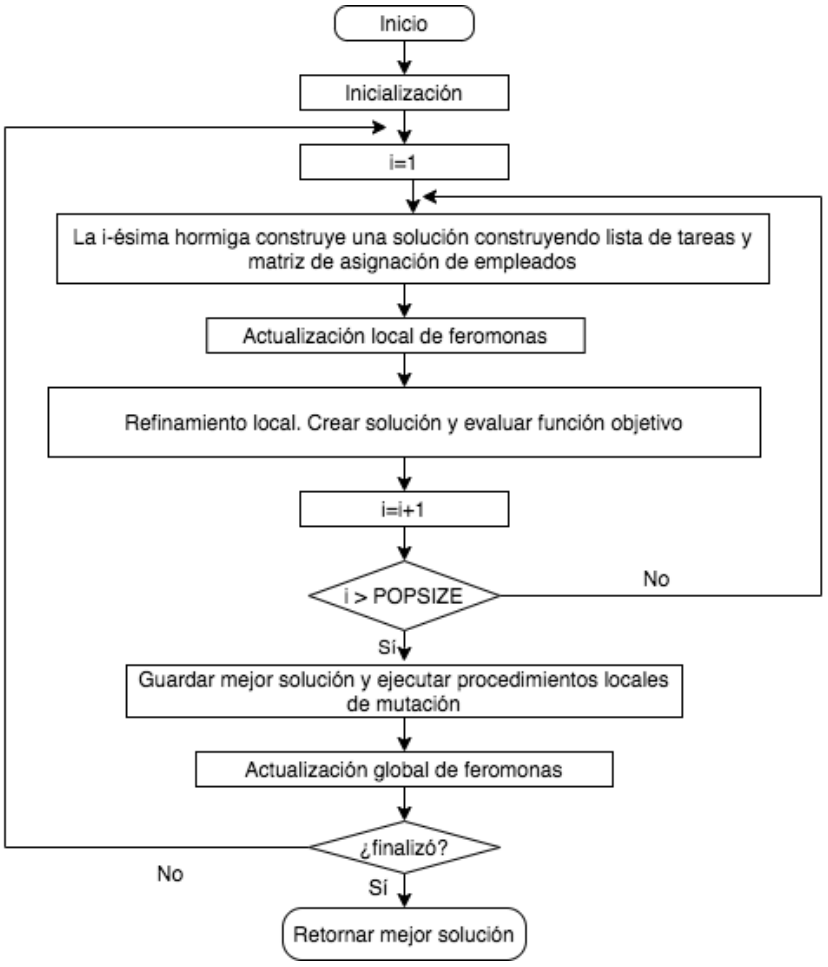


Figura 5: Flujo de algoritmo ACO-SPS. POPSIZE corresponde a la cantidad de hormigas de cada iteración. La condición de *finalización* se basa en la cantidad de soluciones encontradas (y si estas ya fueron encontradas), la cual es un parámetro del algoritmo.

Finalmente el algoritmo ACO-SPS hace uso de lo mencionado anteriomente para calcular la solución (planificación de menor coste) de entre la soluciones de cada iteración generada por cada grupo de hormigas, siendo la cantidad de iteraciones y cantidad de hormigas artificiales definida como parámetros del algoritmo.

Si bien el algoritmo de sistema de hormigas basado en grafos converge (ACS), debido al uso de un algoritmo probabilístico para resolver el problema de optimización, este no garantiza una solución óptima, sin embargo, se asegura la factibilidad de computar una solución y que sea aceptable para su aplicación.

Para el desarrollo del algoritmo ACO-SPS++ se hará uso del algoritmo ACO-SPS en su completitud, es decir, su algoritmo ACO, el planificador basado en eventos, la contrucción de la lista de tareas, la matriz de asignación de empleados y el modelamiento de tareas y empleados. Además, para el modelamiento de los empleados se integrará un modelo de experiencia a cada empleado y los modelos de entrenamiento para las habilidades y experiencias, permitiendo a estas mejorar a lo largo de la

planificación del proyecto, lo que además afectará otros componentes del algoritmo.

Para el desarrollo del algoritmo ACO-SPS++ se hará uso del algoritmo ACO-SPS en su completitud: el algoritmo ACO, el planificador basado en eventos, contrucción de la lista de tareas, matriz de asignación de empleados, el modelamiento de tareas y el modelamiento de empleados, siendo este último modificado para integrarle el modelo de experiencia a cada empleado y los modelos de entrenamiento para las habilidades y experiencias, permitiendo mejorar a lo largo de la planificación del proyecto lo que además afectará otros componentes del algoritmo.

3.3. Modelos de aprendizaje

El aprendizaje en cualquier contexto dado conduce inevitablemente a una mayor eficiencia debido a la repetición de los pasos de producción. Se han publicado investigaciones que relacionan la experiencia con la mejora de la productividad en las industrias tradicionales [32] [27] [28]. Esta mejora en la productividad a lo largo del tiempo con la experiencia se ha capturado comúnmente como una reducción en el costo unitario con el aumento del tiempo y se conoce como curva de aprendizaje. El efecto del aprendizaje en el proceso de desarrollo de software (SDP) es incluso más agudo que en las industrias tradicionales. Se podría decir que el "cambio" es una de las constantes en el desarrollo de software. Los desarrolladores tienen que aprender a usar una nueva herramienta, un nuevo lenguaje de programación, a veces un nuevo paradigma de programación, técnica de prueba, etc. Por lo tanto, el SDP podría estar constantemente involucrado en un proceso de aprendizaje a lo largo del tiempo y en múltiples versiones de un producto de software.

El fenómeno de las curvas de aprendizaje fue observado por primera vez por Wright [23]. Observó que las horas de trabajo directo/unidad disminuyen a un ritmo uniforme a medida que se duplica el número de unidades producidas. Esto llevó a la derivación del **modelo Lineal Logarítmico**. Los resultados de la aplicación de un modelo logarítmico lineal se muestran en la figura 6.

El **modelo Stanford-B** [23] se basa en la suposición de que la experiencia se traslada al siguiente proyecto/producción. En el caso del SDP, esto podría significar que la experiencia se transfiere cuando se desarrolla la próxima versión del producto de software o incluso a otro software. Para una nueva versión de la mayoría de los proyectos, las herramientas, la plataforma y el lenguaje utilizados para las pruebas permanecen sin cambios y, por lo tanto, la experiencia del evaluador ciertamente mejora la productividad en relación con un nuevo evaluador. El patrón del aprendizaje se puede observar en la figura 6.

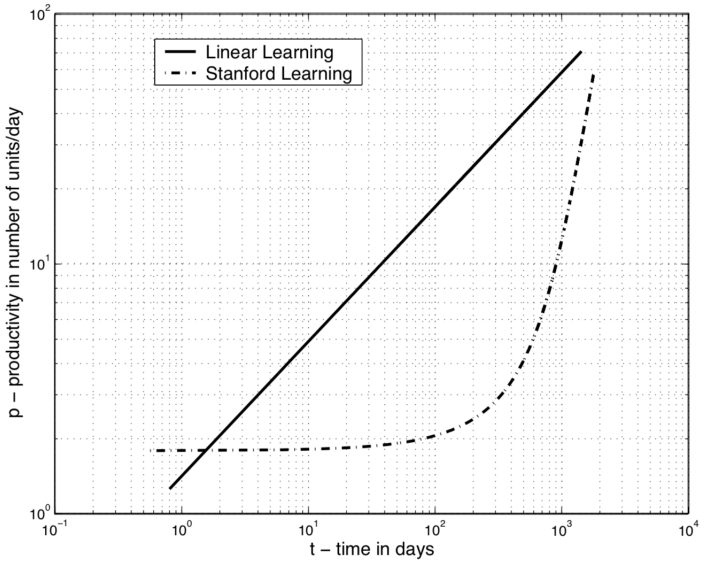


Figura 6: Modelo Lineal Logarítmico y Standord [6].

Por otro lado, el modelo **DeJong** [23] asume que una parte del proceso no se puede mejorar, es decir, hay un límite superior de productividad para los procesos de producción automatizados. Este comportamiento se observa en situaciones en las que las líneas de montaje limitan en última instancia la mejora. El patrón de aprendizaje descrito por el modelo se muestra en la figura 7.

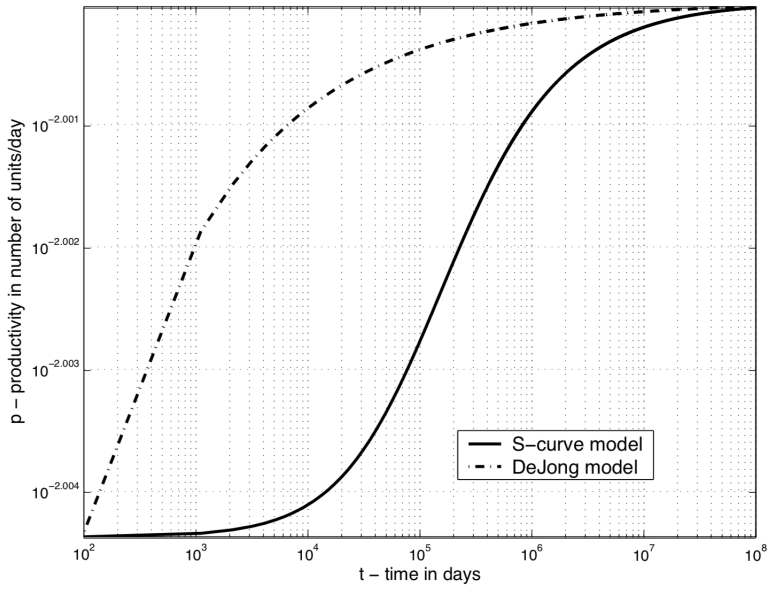


Figura 7: Modelo DeJong y Curva-S [6].

Finalmente el modelo de **S-curve** [23] presentado en la Figura 7 se basa en el supuesto derivado de la combinación del modelo Stanford-B y el modelo DeJong. Este modelo asume que la experiencia se traslada al siguiente proceso y no es posible una mejora en cada etapa del proceso. El comportamiento de este modelo es realmente exponencial como se expresa en la ecuación 5, sin embargo, aparece en forma de S cuando se grafica en una escala logarítmica.

$$y_x = y_1 + D * (x + B)^n \tag{5}$$

donde y_x es el costo de producción de la x-ésima unidad, y_1 es el costo teórico de la primera unidad, el factor de experiencia B es una constante que refleja la experiencia al comienzo del próximo lanzamiento, el factor de incompresibilidad D es una constante que refleja la naturaleza limitante del proceso, y n es una constante que refleja el cambio en la tasa de costo.

Este modelo es un buen candidato para desarrollar el patrón de aprendizaje en la industria del software debido a que el desempeño en todas las etapas del ciclo de vida del software depende de la experiencia de los ingenieros de software involucrados en el proceso de desarrollo. Un ingeniero de software experimentado se debería desempeñar mejor que un recién graduado o un ingeniero de software sin experiencia, independientemente de las etapas del ciclo de vida del software. El conocimiento del dominio, las habilidades de programación y la capacidad de prueba se transfieren al próximo proyecto o al próximo lanzamiento. Con respecto al SDP, esto refleja que el conocimiento adquirido durante un proceso de desarrollo se puede transferir a la siguiente versión o proyecto. Por otro lado, para un software dado, debido a los recursos limitados (mano de obra, adecuación de las herramientas, etc.), el rendimiento de una parte de un proceso no se puede mejorar más allá de un cierto límite. Este límite se puede alcanzar más adelante dentro de la misma versión o incluso en una versión futura del producto. Este comportamiento también fue observado por Hanakawa, Morisaki y Matsumoto [18]. El comportamiento de aprendizaje y su efecto sobre el SDP se pueden describir esencialmente como un aumento inicial gradual y luego un crecimiento brusco que conduce a cierto valor de saturación de conocimiento.

4. Implementación algoritmo EWEHD

Para cumplir con los objetivos planteados, el trabajo realizado consta de dos implementaciones, siendo la primera de ellas el algoritmo EWEHD. Para la implementación se hace uso de la estructura estandarizada y normalizada de los archivos de logs en formato XES.

A partir del log se utilizan sus siguientes componentes:

1. **Log**.
2. **Traza**.
3. **Evento**.
4. **Atributo**: Los atributos que se utilizan en el trabajo realizado son los contenidos en el componente evento y son los siguientes:

   a) **Recurso** ($org : resource$): Indica el recurso (por ejemplo: un empleado, una máquina, etc) utilizado para la realización del evento.

   b) **Fecha y hora** ($time : timestamp$): Indica un período en el tiempo en el cual ocurrió el evento.

   c) **Concepto** ($concept : name$): Indica el nombre o tipo del evento.

   Además se utilizaron otros atributos del componente log que permiten conocer la cantidad de eventos y trazas contenidas en el log.

5. **Clasificador**: El clasificador utilizado en el trabajo realizado es el siguiente: *<classifier name="concept:name" keys="concept:name"/>* el cual permite obtener el nombre de un evento o traza.

6. **Extensión**: La extensión utilizada en el trabajo realizado es la siguiente:

   <extension name="Concept" prefix="concept"
   uri="http://www.xes-standard.org/concept.xesext"/>.

   La declaración anterior indica que el registro presenta los atributos definidos por la extensión "Concept". El atributo "prefix" declara el prefijo de todos los atributos definidos por esta extensión en el registro. Esto significa que las claves de todos los atributos definidos por esta extensión estarán antepuestas por "concept" y el carácter de separación de dos puntos como se observa en el **clasificador** mencionado anteriormente. De esta manera, los atributos hacen referencia a su respectiva extensión.

4.1. EWEHD

Con los registros de procesos en archivos de logs en formato XES, el objetivo de EWEHD es hacer uso de estos para estimar la carga de trabajo asociada a cada tarea o evento pertenecientes al flujo de trabajo de sus procesos. Para esto se hará uso de los componentes del log descritos anteriormente.

Para el trabajo desarrollado se asumen los siguientes criterios:

- Dado que los eventos registrados en cada traza del log no poseen un atributo que describa la duración de estos, utilizando el atributo *time:timestamp*, se define como *timestamp* del i-ésimo evento en una traza como: $timestamp_i$ y la duración de este, como:

$$duration_i = timestamp_i - timestamp_{i-1} \tag{6}$$

- Dado lo anterior, el primer evento solo se considera como indicador del inicio de una traza y no se estimará su duración.

Además del log, la ejecución del algoritmo comprende: la inicialización de un parámetro para extraer el recurso utilizado en cada evento, un modo para la ejecución de las trazas y un modo para el cálculo de la duración de cada evento, los cuales se describen a continuación.

El parámetro para extraer el recurso (humano, máquina, etc.) utilizado en el evento o tarea es el primer parámetro solicitado por el algoritmo y corresponde a la llave del atributo perteneciente al evento que cuyo valor indica el recurso utilizado en la realización de dicho evento. Si el usuario omite el ingreso del parámetro o un evento no posee el atributo indicado por el parámetro, no se buscará el atributo o se omitirá la extracción del recurso de dicho evento, respectivamente.

Respecto a los modos de ejecución de los eventos de una traza se definen los siguientes:

- $ISOLATED$: Se asume que toda traza es independiente de las otras. Es decir, los eventos o tareas de cada traza no interfieren en el flujo de una traza a la cual no pertenecen. El orden de ejecución de cada evento se basa en el atributo $time : timestamp$.

- $CROSSED$: Se asume que existe la posibilidad de intercambio o compartición de recursos entre las trazas para la realización de las tareas o eventos. Por lo tanto se hace la unión de las trazas en una sola y cuyos eventos se ordenan de acuerdo al atributo $time : timestamp$ en forma creciente.

Finalmente los modos para el cálculo de la duración de los eventos son:

- $STRICT$: Restringe a que todo evento posea atributo *time:timestamp*. En caso contrario se genera una excepción.

- $ESTIMATED$: Permite que eventos no posean atributo *time:timestamp*, debido a una mala implementación del estándar o fecha malformada que impide su manipulación. Para el cálculo del timestamp estimado, $timestamp_{ESTIMATED}$, de un evento t_b, se utiliza el *timestamp* del evento t_a anterior más cercano y el evento t_c posterior más cercano, como vemos en la ecuación (7).

$$timestamp(t_b) = (t_c - t_a)/d \qquad con\ a < b < c \tag{7}$$

  Siendo d la cantidad de eventos o tareas desde t_a hasta t_c. Si t_a o t_c no existen se genera una excepción.

- $FLEXIBLE$: Permite que eventos no posean atributo *time:timestamp*. Estos eventos no serán considerados en la estimación final.

El algoritmo EWEHD se puede dividir en tres secciones: (1) Verificación y estimación de la fecha de registro, (2) Preparación para el modo de ejecución de las trazas y finalmente (3) Cálculo de la duración de tareas junto con desviación estándar, media y mínimos/máximos de cada tipo de evento o tarea. Cada sección se detallará a continuación.

Verificación y estimación

El algoritmo provee tres tipos de modos para la estimación y cálculo final de la duración de cada evento o tarea, $STRICT$, $ESTIMATED$ y $FLEXIBLE$. A partir de estos modos se realizan verificaciones para que cada evento perteneciente a cada traza cumpla con los requisitos del modo seleccionado.

Algoritmo 1 Verifica que cada traza cumpla con los requisitos para modo de estimación de la duración

```
1: procedure CHECKTRACE(trace, fm)
2:     first ← trace ← firstEvent()                    ▷ Buscar primer evento con atributo timestamp
3:     lastTimestamp ← first.timestamp
4:     firstIndex ← first.index
5:     traceLength ← length(trace)
6:     for i ← firstIndex + 1 to traceLength do
7:         event ← trace(i)
8:         timestamp ← event.timestamp
9:         if timestamp == NULL then
10:            if fm == FLEXIBLE then
11:                continue
12:            else if fm == STRICT then
13:                raise ERROR
14:            else if fm == ESTIMATED then
15:                nextEvent ← trace.nextEvent(i)      ▷ Buscar siguiente evento con atributo timestamp
16:                timetamp ← nextEvent.timestamp
17:                if timetamp == NULL then
18:                    raise ERROR
19:                else
20:                    event.timestamp ← (lastTimestamp + timetamp)/2   ▷ Se define como nuevo timestamp el
   punto medio con el siguiente evento
21:                    lastTimestamp ← event.timestamp
22:                end if
23:            end if
24:        else
25:            lastTimestamp ← timestamp
26:        end if
27:    end for
28:    return trace
29: end procedure
```

Para esto se desarrolló la función $CHECKTRACE$ descrito en el algoritmo 1. Como se mencionó anteriormente se asume que el primer evento de cada traza solo se utilizará como indicador de inicio de la traza (línea 2) y se considera su *timestamp* o fecha de registro como el último evento ocurrido ($lastTimestamp$) hasta el momento (línea 3) en su traza. A partir de esto, se itera sobre los eventos restantes de la traza en orden de aparición (línea 6) y se extrae su fecha de registro. Si el evento posee fecha de registro, esta se considera como último evento ocurrido (línea 25). Si la fecha de registro no existe se procede a verificar si cumple con los requisitos del modo de estimación escogido. Si es $STRICT$ (línea 12) la ejecución del algoritmo finaliza con una excepción. Si es $FLEXIBLE$ (línea 10) ese evento no es considerado y se descarta.

Finalmente si el modo seleccionado es $ESTIMATED$ (línea 14) se procede a estimar una fecha de registro para este evento. Para esto se busca el siguiente evento en la traza que posea fecha de registro (línea 15) y se calcula el punto medio entre esta y la fecha de registro del último evento occurido (línea 20). Luego esta nueva fecha de registro de considera como último evento ocurrido hasta el momento (línea 21). En el caso que no exista un evento en la traza con una fecha de registro para poder realizar la estimación, el algoritmo finaliza con una excepción (línea 17 y 18).

Preparación para el modo de ejecución

Una vez que se comprueba que el log cumple con los requisitos del algoritmo, se procede al pre-procesamiento del log en base al modo de ejecución seleccionado. Para el uso del modo de ejecución de trazas $CROSSED$, dado que existe la posibilidad de intercambio o compartición de recursos entre las trazas para la realización de las tareas o eventos, se realiza la unión de todos los eventos pertenecientes a cada traza del log en una traza única.

Algoritmo 2 Estimación de la duración de tareas de un log XES

```
1: procedure EWEHD(log, traceMode, durationMode) ▷ EWEHD sobre log, modo de trazas traceMode y modo de duración durationMode
2:     durations ← dict()                                   ▷ Inicialización de duraciones
3:     min ← dict()                                         ▷ Inicialización de mínimos
4:     max ← dict()                                         ▷ Inicialización de máximos
5:     for all e ∈ events(log) do                           ▷ Para todo tipo de evento o tarea se define min y máx por defecto
6:         durations(e.name) ← array()
7:         min(e) ← MAXINT
8:         max(e) ← MININT
9:     end for
10:    if traceMode == ISOLATED then
11:        for trace ← traces(log) do
12:            events ← CHECKTRACE(trace, durationMode)     ▷ Verificar y completar consistencia de traza
13:            i ← 1
14:            lastTimestamp ← events.get(0).timestamp
15:            length ← len(events)
16:            while i < length do
17:                currentTimestamp ← events.get(i).timestamp
18:                duration ← currentTimestamp − lastTimestamp
19:                durations(e_i) ← durations(e_i) ∪ duration
20:                min(e_i) ← MIN(durations(e_i), duration)
21:                max(e_i) ← MAX(durations(e_i), duration)
22:                i ← i + 1
23:                lastTimestamp ← currentTimestamp
24:            end while
25:        end for
26:    else if traceMode == CROSSED then
27:        events ← array()
28:        for trace ← traces(log) do
29:            currentEvents ← CHECKTRACE(trace, durationMode)   ▷ Verificar y completar consistencia de traza
30:            events = events + currentEvents
31:        end for
32:        sort(events)                                     ▷ Ordenamiento ascendente de los eventos respecto a atributo timestamp
33:        i ← 1
34:        lastTimestamp ← events.get(0).timestamp
35:        length ← len(events)
36:        while i < length do
37:            currentTimestamp ← events.get(i).timestamp
38:            duration ← currentTimestamp − lastTimestamp
39:            durations(e_i) ← durations(e_i) ∪ duration
40:            min(e_i) ← MIN(durations(e_i), duration)
41:            max(e_i) ← MAX(durations(e_i), duration)
42:            i ← i + 1
43:            lastTimestamp ← currentTimestamp
44:        end while
45:    end if
46:    sd = SD(d)
47:    med = MED(d)
48:    return min, max, sd, med
49: end procedure
```

Esto se puede observar entre las línas 27 y 31 del algoritmo 2. Luego es necesario realizar un ordenamiento ascendente de todos los eventos respecto a su atributo *timestamp* o fecha de registro, lo cual está presente en la línea 32 del mismo algoritmo. Por otro lado si el modo de ejecución escogido es $ISOLATED$, dado que se asume que cada traza es independiente de otra y la estructura inherente del log, no es necesario realizar un pre-procesamiento de estas.

Duración, desviación estándar, media, min y max

Luego de la verificación y estimación de las fechas de registro para todo evento en base al modo de estimación escogido se procede a calcular la duración de cada evento. Se realiza una iteración sobre los eventos de cada traza, local (línea 16) o global (línea 36), a partir del segundo evento presente en la traza. Finalmente la duración de un evento corresponde a la diferencia entre su fecha de registro y la fecha de registro del evento anterior. Además utilizando su atributo **Concepto** que indica el nombre o tipo de evento se verifica si su duración es un máximo (líneas 20 y 40), mínimo (líneas 21 y 41), desviación estándar (línea 46) y media (líneas 47) utilizando los eventos de su mismo tipo.

4.2. Plugin ProM

ProM es una herramienta de investigación para minería de procesos. ProM tiene un arquitectura en base a plugins la cual permite extender sus funcionalidades o herramientas.

ProM trabaja como un modelo transformador y visualizador de información, utilizando *objetos* y *visualizadores*, con un diseño de tipo $modelo - vista$ ($model - view$), siendo los objetos parte del modelo y los visualizadores parte de la vista.

Para un mejor entendimiento de los requisitos de una estructura general del plugin y para que cumpla el estándar de ProM se presenta en la figura 8 con un diagrama UML con base en Java.

Para la implementación del plugin, se hace uso del objeto XLog, el cual corresponde a una digitalización de un archivo log de formato XES, que habilita un set de instrucciones o métodos que permiten interactuar y consultar características y componentes sobre el log. Dado esto, primero se describirá la generación del objeto XLog, los objetos, la implementación del plugin EWEHD y su visualizador para ProM.

4.2.1. XLog

La generación del objeto $XLog$, desde ahora *log*, se realiza utilizando el componente **XesParser** (*parser*) incorporado dentro de un set de herramientas de ProM. Este componente permite importar un archivo de log de formato XES digitalizándolo en un objeto XLog y ser usado dentro de un plugin de ProM.

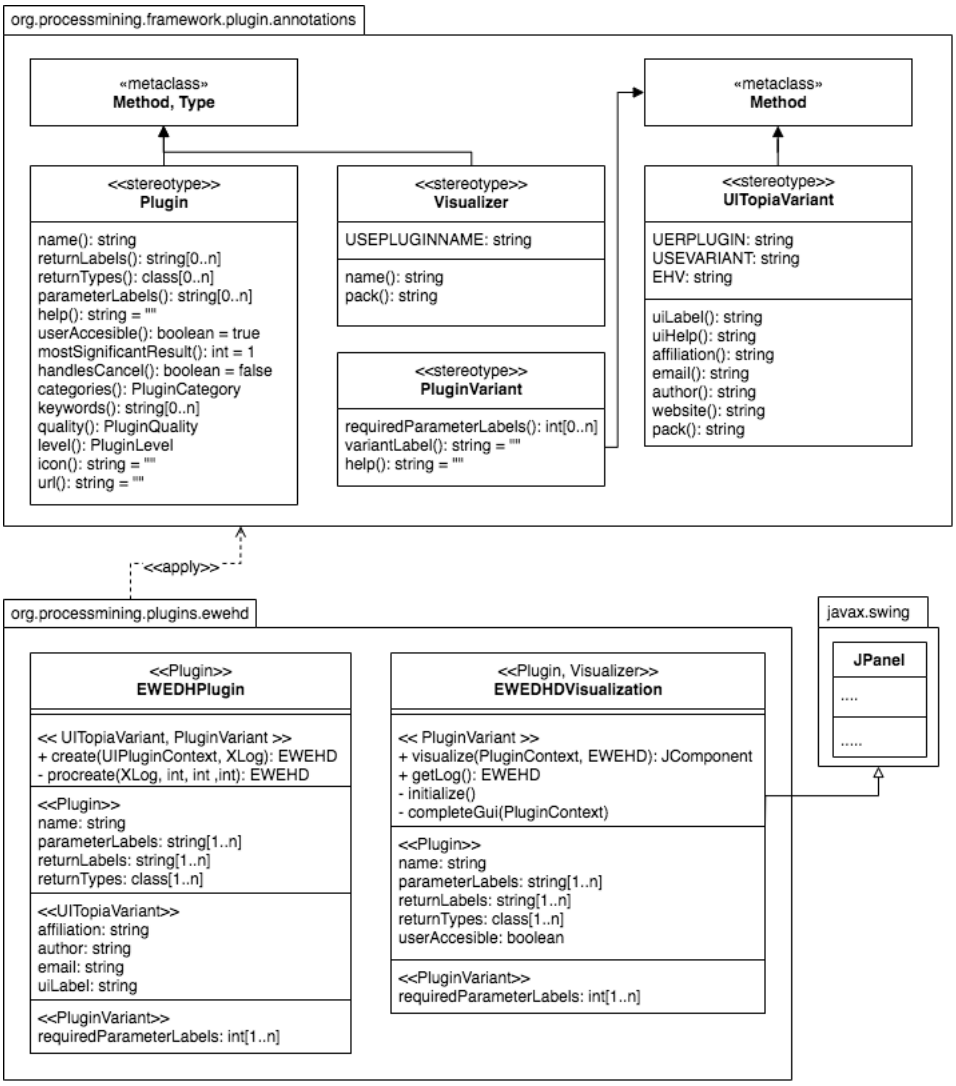


Figura 8: Diagrama UML de plugin y visualizador ProM.

4.2.2. Objetos

Para el desarrollo del componente *modelo* de la arquitectura *modelo-vista*, se utilizaron objetos regulares de Java y para el soporte de la serialización a disco se debió implementar la interface *Serializable* [4] en cada uno de ellos.

4.2.3. Plugin

La idea principal de un plugin es transformar un objeto en otro y debe implementar ciertos métodos, anotaciones y una configuración. La configuración describe exactamente cómo el plugin va a operar. Para su correcta importación, cada plugin debe ser desarrollado bajo el paquete *org.processmining.plugins.<plugin-name>*, siendo *plugin-name* el nombre del plugin. Dado esto se desarrolló la clase *EWEHD-Plugin* cuya implementación está presente en el apéndice 9.3, listado 1.

La declaración de la clase del plugin debe incluir la anotación [1] @*Plugin* que

especifica que una clase es un plugin y debe incluir los siguientes parámetros:

- **name**: De tipo *string*. Corresponde al nombre del plugin. Este se utilizará en la lista de plugins disponibles (apéndice 9.4, figura 21) dentro del programa ProM.
- **parameterLabels**: De tipo *array<string>*. Corresponde a una lista con los nombres de los posibles parámetros que el plugin requerirá para su funcionamiento. Estos también serán visibles en el listado de plugins (apéndice 9.4, figura 21) dentro de ProM.
- **returnLabels**: De tipo *array<string>*. Corresponde a una lista de los nombres de los objetos que puede retornar el plugin. Se recomienda que estos contengan el nombre de la clase de los *objetos* que retorna el plugin.
- **returnTypes**: De tipo *array<class>*. Corresponde a una lista de las *clases* de los objetos que puede retornar el plugin.

Es importante destacar que los valores de las listas $returnLabels$ y $returnTypes$ están directamente relacionados, es decir, ambas deben tener el mismo largo y para un índice i, $returnLabels[i]$ y $returnTypes[i]$ deben referirse al mismo tipo de objeto.

En conjunto, las anotaciones $PluginVariant$ y $@UITopiaVariant$ deben ser incluídas en un método de la clase e indican que este método es una variante de dicho plugin y que el plugin debe ser expuesto en la interfaz de usuario de ProM, respectivamente. Cada método que contenga estas anotaciones, en su firma debe incluir como valor de retorno una de las clases agregadas en la anotación $@Plugin$, y como primer parámetro, un objeto de tipo $UIPluginContext$ para variantes que no requieran una configuración o $PluginContext$ para las que sí lo requieran. La anotación $PluginVariant$ especifica a través del parámetro $requiredParameterLabels$, qué parámetros utiliza realmente la variante. Esta es una lista de índices pertenecientes la lista de nombres de parámetros (*parameterLabels*) de la anotación $@Plugin$. Los índices comienzan en cero. Por otro lado, $UITopiaVariant$ especifica información adicional expuesta en la interfaz de usuario de ProM, incluyendo el nombre, la afiliación y la dirección de correo electrónico del autor. También especifica la etiqueta (*uiLabel*) que se utiliza para mostrar el plugin en la interfaz de usuario. En este caso, sólo se especifica que se debe utilizar el nombre del plugin.

4.2.4. Visualizador

Como se mencionó anteriormente, el plugin transforma un objeto en otro, pero para poder visualizar este objeto y validar los resultados en ProM, es necesario el desarrollo de un visualizador. Los visualizadores son un tipo especial de plugin. Estos deben retornar un objeto de tipo $JComponent$ [2], y contener anotaciones adicionales. Dado esto se desarrolló la clase $EWEHDVisualization$ cuya implementación está presente en el apéndice 9.3, listado 2.

Al igual que un plugin, a nivel de declaración de la clase, un visualizador también debe poseer la anotación @*Plugin* (línea 215) con sus respectivos parámetros, pero en *returnTypes*, solo debe poseer un objeto de tipo *JComponent.class* y el parámetro *userAccesible* con valor *false*, para asegurarse que los usuarios no puedan ejecutar este plugin (ya que el objeto devuelto por JComponent viola varias reglas para los objetos ProM).

Además, a diferencia de un plugin, un visualizador también debe incluir la anotación @*Visualizer* (línea 216) para indicar que es un visualizador y no exponerlo en la lista de plugins de ProM (figura 21).

También a nivel de métodos se incluye la anotación @*PluginVariant* (línea 217), similar a lo hecho en el plugin, con la condición que la firma de todo método debe retornar un objeto de tipo *JComponent*, el cual será visualizado en la interfaz de ProM.

Para entender cómo interactuan todas las partes dentro de la solución desarrollada, a continuación se presenta un diagrama de flujo (figura 9) con la interacción entre ellas. Además se adjuntan en el apéndice 9.4 las imágenes de las interfaces de la implementación del plugin en ProM.

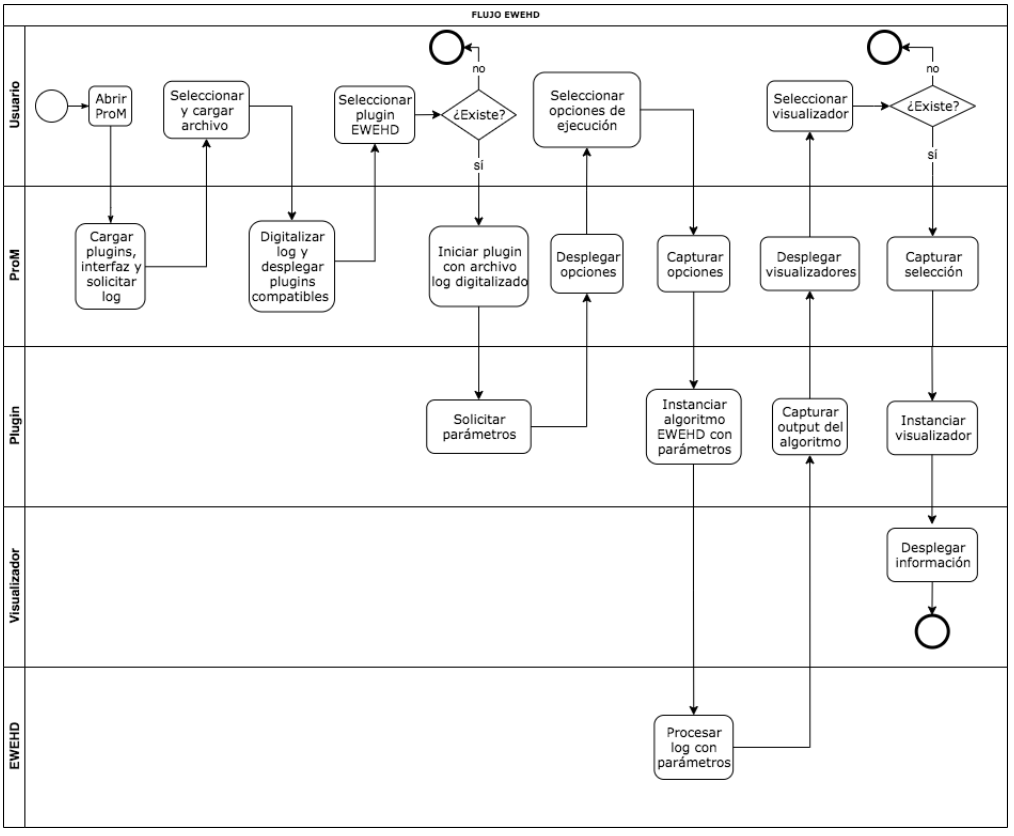


Figura 9: Diagrama de flujo plugin EWEHD

Una vez que el usuario inicia la aplicación ProM, lo primero que debe realizar es seleccionar e importar el log en formato XES que quiere procesar. Luego debe buscar

el plugin desarrollado EWEHDPlugin en la caja de búsquedas y seleccionarlo. Luego al iniciar el plugin con el archivo XES importado se procede a solicitar al usuario la configuración requerida por este para procesarlo, atributo del recurso, modo de procesamiento de trazas y modo de estimación de fecha de registro y duración. Luego que el algoritmo termina el procesamiento del archivo XES, se despliega el selector del visualizador para poder ver el resultado del algoritmo. Finalmente, una vez seleccionado el visualizador $EWEHDVisualization$ se despliega el resultado del algoritmo con información general del log, las estimaciones de duraciones y recursos de cada tipo de evento o tarea.

5. Implementación algoritmo ACO-SPS++

La segunda implementación realizada para cumplir con los objetivos planteados es el desarrollado del algoritmo ACO-SPS++.

Para la planificación de proyectos, simulando el comportamiento de las hormigas, es posible diseñar heurísticas útiles para dirigirlas a programar las tareas críticas lo antes posible y asignar las tareas del proyecto a empleados adecuados con las habilidades requeridas.

El algoritmo de colonias de hormigas ACO fue propuesto por Marco Dorigo en 1992 en su tesis de doctorado [16] y con Gambardella [15] para resolver el problema del vendedor ambulante (TSP) [17] y se ha aplicado con éxito a varios problemas de optimización y combinatoria. ACO desarrolla soluciones a medida, paso a paso y permite el uso de heurísticas basadas en problemas para guiar la dirección de búsqueda de las hormigas. Por lo tanto, dado que ACO-SPS++ (Ant Colony Optimization for Software Project Scheduling and Staffing with an Event-Based Scheduler and Training Model) también utiliza el algoritmo ACO, promete converger y funcionar bien en el problema considerado.

Con el objetivo de realizar la planificación de un proyecto de software con asignación de empleados, el algoritmo ACO-SPS++, utilizando la información de empleados (habilidades, experiencias, salarios, etc.), tareas y un algoritmo ACO, se construye una lista de tareas y una matriz de asignación de empleados.

Utilizando esta lista de tareas, la cual indica el orden en que las tareas deben ser procesadas a través de la planificación, y la matriz de asignación de empleados, que indica las horas que cada empleado dedicará a cada una de las tareas inicialmente, un planificador de tareas basado en eventos (EBS) asigna a través del avance del proyecto, la cantidad de empleados aptos a cada una de las tareas para que estas se completen.

Dado que ahora las habilidades y la experiencia de cada empleado puede variar durante la planificación, debido a que pudieron ser asignados en tareas previas, el EBS en cada iteración deberá considerar las asignaciones y actualizar las habilidades y experiencias de cada empleado involucrado y como consecuencia reconstruir la lista de tareas y matriz de asignación de empleados para las siguientes iteraciones.

Se procederá a explicar el algoritmo ACO-SPS++ en tres secciones:

- Modelos: Se describirá el modelo de empleados, tareas y aprendizaje.
- Esquema de representación de la lista de tareas, matriz de asignación de empleados y el EBS.
- Algoritmo ACO, que construye la lista de tareas y la matriz de asignación haciendo uso del nuevo modelo de entrenamiento.

5.1. Modelo de planificación de tareas de proyectos de software y asignación de recursos humanos

Se considera en este trabajo los modelos para la asignación de recursos humanos y planificación de tareas.

Modelo de empleados: En proyectos de desarrollo de software el jefe de proyectos debe ser capaz de administrar los recursos humanos, sus capacidades, salarios, restricciones de trabajo y asignarlos de manera eficiente para el éxito del desarrollo. Para esto se definen los siguientes atributos a cada i-ésimo-empleado ($i = 1, 2, 3, ..., m$):

- bs_i: Salario base por mes.
- hs_i: Salario por hora de trabajo.
- ohs_i: Salario por hora extra de trabajo.
- nh: Cantidad de horas legales de trabajo por mes.
- $maxh_i$: Cantidad máxima de horas posibles de trabajo por mes.
- $[join_i, leave_i]$: Ventana de tiempo en la cual el empleado trabajará en el proyecto.
- $\{s_i^1, s_i^2, ..., s_i^\Phi\}$: Lista de habilidades de un empleado, donde Φ es el número de la habilidad y $s_i^j \in [0, 5]$ es la competencia a la j-ésima habilidad. $s_i^j = 0$ implica que el empleado no pesee la habilidad y $s_i^j = 5$ tiene la máxima competencia en esa habilidad.
- $\{e_i^1, e_i^2, ..., e_i^\alpha\}$: Lista de experiencia del i-ésimo empleado, donde α es el número de la tarea y $e_i^n \in [0, 5]$ es la experiencia del empleado a la n-ésima tarea. $e_i^n = 0$ implica que el empleado no pesee experiencia y $s_i^n = 5$ tiene la máxima experiencia en esa tarea.

  Supongamos que el i-ésimo empleado trabaja $hours_i^t$ horas en el proyecto el t-ésimo mes ($hours_i^t < maxh_i$). Si $hours_i^t$ es mayor que la cantidad de horas legales nh, implica que el empleado trabajó horas extras ese mes. Dado esto, el salario de un empleado se define como:

$$salary_i^t = \begin{cases} bs_i + hours_i^t \cdot hs_i & hours_i^t \leq nh \\ bs_i + nh \cdot hs_i + (hours_i^t - nh) \cdot ohs_i & nh < hours_i^t \leq maxh_i \\ \infty & hours_i^t > maxh_i \end{cases} \tag{8}$$

Modelo de tareas: Una tarea es cualquier actividad involucrada en el desarrollo de software. Para la descripción y precedencia de tareas se usó un grafo TPG (del inglés, *Task Precedence Graph*) acíclico dirigido $G(T, A)$. Cada tarea está contenida en el conjunto de nodos $T = \{t_1, t_2, ..., t_n\}$, donde n es el número de tareas en el proyecto. Cabe destacar que este modelo aplica solo si se usa en un desarrollo estructurado.

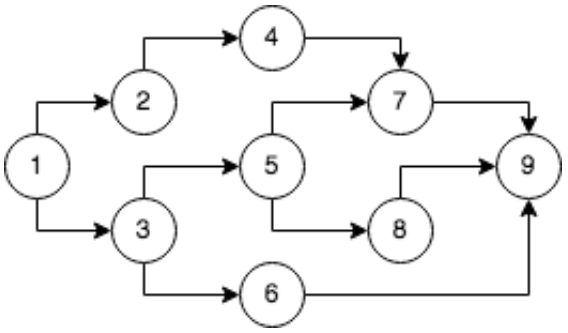


Figura 10: Ejemplo de grafo de precedencia de tareas TPG.

El conjunto de arcos A representa la precedencia de tareas. Por ejemplo, un $arc(i, j) \in A$ significa que la tarea t_i es el directo predecesor de la tarea t_j. Considerando el grafo de precedencia, solo se puede realizar una tarea si todas sus tareas predecesoras fueron concluídas. Cabe recalcar que esta solución no considera la existencia de ciclos en el grafo G, por lo que este problema no será abordado.

Para cada tarea $t_j (j = 1, 2, 3, ..., n)$ tenemos los siguientes atributos:

- pm_j: Esfuerzo estimado de trabajo en persona mes.
- SK_j: Conjunto de habilidades requeridas para la tarea.
- $maxhead_j$: Número máximo de empleados que pueden ser asignados a la tarea.
- $deadline_j$ y $penalty_j$: Fecha límite para terminar tarea y penalización por mes de atraso respectivamente.

Con estas definiciones, se define wh_{ij}^t como la cantidad de horas que trabajó el i-ésimo empleado en la j-ésima tarea el t-ésimo mes, entonces el logro, basado en [14] y [13], A_j^t obtenido por los empleados para t_j en el mes t puede ser evaluado como sigue:

1. La competencia $prof_{ij}$ del i-ésimo empleado para t_j será:

$$prof_{ij} = \prod_{id \in SK_j} \frac{s_i^{id}}{5} \tag{9}$$

   Notar que $prof_{ij} \in [0, 1]$.

2. La aptitud general del empleado $taskfit_{ij}$ del i-ésimo empleado para t_j será:

$$taskfit_{ij} = \frac{prof_{ij} + \frac{e_i^j}{5}}{2} \tag{10}$$

   Nuevamante notar que $taskfit_{ij} \in [0, 1]$.

3. La aptitud total F_j^t de los empleados asignados para t_j en el t-ésimo mes está dado por:

$$F_j^t = \frac{\sum_{i=1}^{m} taskfit_{ij} \cdot wh_{ij}^t}{\sum_{i=1}^{m} wh_{ij}^t} \tag{11}$$

4. Convertir F_j^t a un valor de costo, utilizando [13] [14], se obtiene $V = 8 - round(F_j^t \cdot 7 + 0{,}5)$, donde $V \in [1, 7]$. $V = 1$ significa que los empleados son los más adecuados para la tarea y viceversa.

5. El logro A_j^t para t_j en el t-ésimo mes será:

$$A_j^t = \frac{\sum_{i=1}^{m} wh_{ij}^t}{V} \tag{12}$$

   Dada la planificación de tareas y asignación de empleados, el resultado de la planificación debe especificar cuándo las tareas serán procesadas y cómo será distribuida la carga de trabajo de los empleados asignados a cada tarea, cumpliendo las siguientes restricciones:

   a) El orden en el que son procesadas las tareas debe cumplir con las restricciones de precedencia definidas por TPG.

   b) Las horas trabajadas por el i-ésimo empleado por mes no debe exceder el límite $maxh_i$:

$$\sum_{j=1}^{n} = wh_{ij}^t = hours_i^t \leq maxh_i \qquad t = 1, 2, 3, ... \tag{13}$$

   c) El número máximo de empleados asignados a una tarea t_j no debe superar $maxhead_j$:

$$\begin{gathered} \sum_{i=1}^{m} sign(\sum_{t=start_j}^{finish_j} wh_{ij}^t) \leq maxhead_j \\ \text{donde } sign(x) = \begin{cases} 1, & \text{si } x > 0 \\ 0, & \text{si } x = 0 \end{cases} \end{gathered} \tag{14}$$

   d) Todas las tareas deben ser completadas. Para una tarea t_j la suma de sus logros durante la ventana de tiempo $[start_j, finish_j]$ debe cumplir:

$$\sum_{t=start_j}^{finishj} A_j^t \geq pm_j \tag{15}$$

Finalmente el objetivo del algoritmo es la minimización de la siguiente función:

$$f = \sum_{t=1}^{end} salary_i^t + \sum_{j=1}^{n} penalty_j \tag{16}$$

siendo el primer componente el salario de los empleados y el segundo, las penalizaciones por cada tarea.

5.2. Modelo de aprendizaje

La investigación sobre la mejora en el aprendizaje con el tiempo para los sistemas de software y su efecto en SDP durante el ciclo de vida del software ha llevado al desarrollo de modelos de aprendizaje. Como se mencionó inicialmente, existen varios modelos de aprendizaje: lineales, Stanford-b, DeJong, modelos tipo S-curve y como el propuesto por Ghaffari Abu y Joao W. Cangussu [6]. Con este último, se utiliza el siguiente modelo de aprendizaje para el trabajo realizado:

$$\dot{l} = Kl(1 - \frac{l}{s}) \tag{17}$$

Siendo $\dot{l}$ la tasa de cambio en el conocimiento igual al producto del conocimiento actual l y el conocimiento restante para ser adquirido $(1 - l)$. s es el nivel de saturación, es decir, el nivel máximo alcanzable que para este trabajo es 5. Por otro lado, la constante K representa la tasa de aprendizaje.

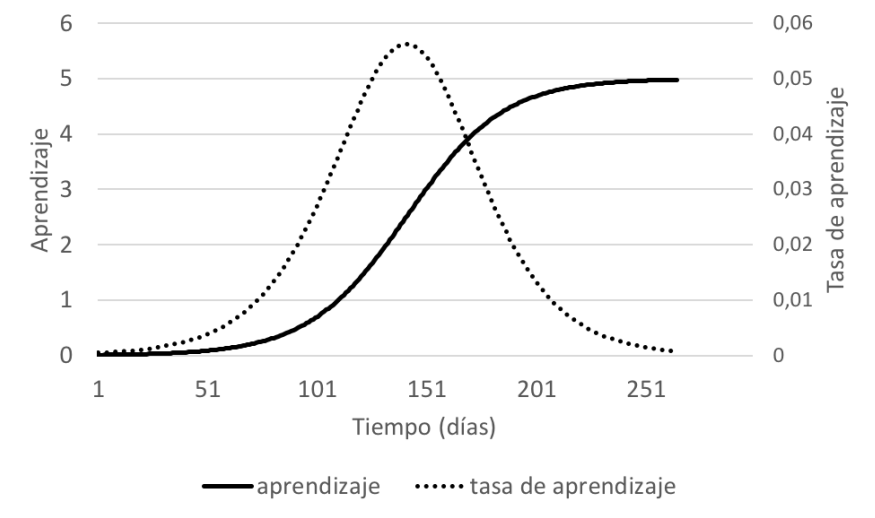


Figura 11: Curva de aprendizaje y tasa de de cambio del conocimiento. Adaptado desde [6].

La figura 11 muestra las características de aprendizaje con respecto al tiempo dedicado al proyecto derivado de la ecuación (17) . El comportamiento en forma de S de la curva de aprendizaje se rige por el conocimiento existente, el valor de K y el conocimiento restante que se adquirirá para lograr la saturación s. La figura 11

también muestra la tasa de cambio en el aprendizaje, que tiene un máximo en el punto de inflexión de la curva de aprendizaje.

Se supone que el aprendizaje se satura al valor máximo de 5 al final del ciclo de vida del software. Sin embargo, la tasa de cambio en el aprendizaje comienza a disminuir en el punto de inflexión como lo muestra el máximo de la curva de tasa de aprendizaje.

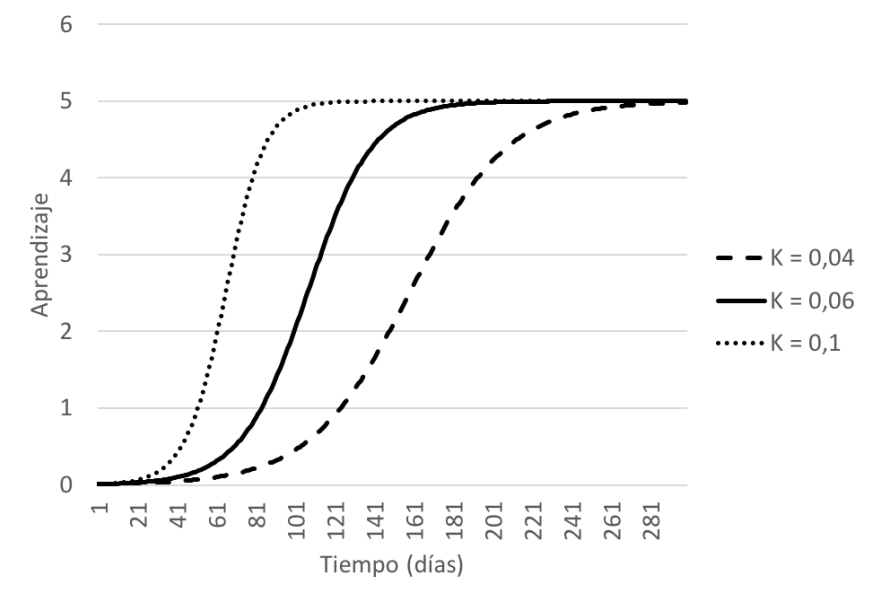


Figura 12: Los diferentes valores de K proporcionan a los jefes de proyectos la capacidad de fijar la tasa de aprendizaje individual. Adaptado desde [6].

La figura 12 muestra que cuanto mayor es el valor de K, el cual varía entre 0 y 1, mayor es el coeficiente de la pendiente y, por lo tanto, más rápido es el aprendizaje. Los valores más altos de K significan un mayor aprendizaje y eso significa un logro más rápido del nivel de saturación. K proporciona a los gerentes o jefes de proyecto establecer la tasa de aprendizaje para cualquier empleado o para todo el grupo. Por otro lado, un mejor conocimiento sobre el proyecto, las herramientas, la plataforma utilizada para desarrollar el software, el conocimiento teórico y práctico al comienzo del proyecto también conduce a un aprendizaje más rápido y, por lo tanto, aumenta la curva de aprendizaje como se muestra en la figura 13 .

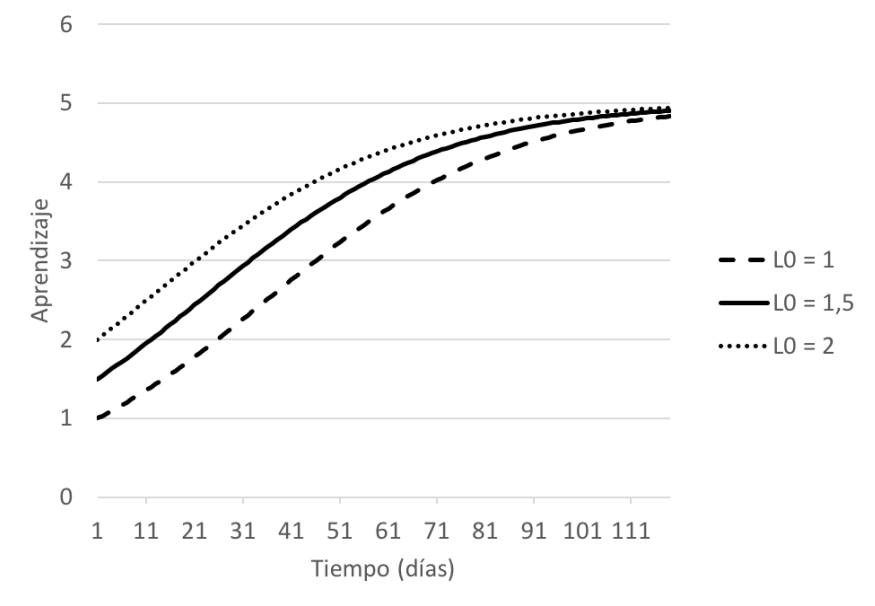


Figura 13: Efecto del aprendizaje inicial en el aprendizaje. Adaptado desde [6].

5.3. Esquema de representación y planificador basado en eventos (EBS)

El modelo clásico del problema de planificación de proyectos con recursos limitados (RCPSP) para determinar el orden en el cual se procesarán las tares utiliza una lista de tareas (18) donde $(p_1, p_2, ..., p_n,)$ es una permutación de las tareas $(1, 2, ..., n)$. Para resolver conflictos de recursos considera las prioridades de cada tarea, siendo la que con mayor prioridad la tarea que se procesará primero [10]. Dado que RCPSP solo está enfocado en el problema de la planificación y no considera el problema de la asignación de empleados, es inadecuado para modelar completamente el problema de la planificación de proyectos de software.

$$(t_{p_1}, t_{p_2}, ..., t_{p_n}) \tag{18}$$

Para poder abarcar el problema de la planificación de trabajo con múltiples habilidades y experiencias, junto con la asignación de los recursos humanos, se utilizó la lista de tareas (18) y una matriz de asignación (19). Para aumentar la flexiblidad, reducir restricciones y preferencias de tareas a un empleado, cada valor representa la carga de trabajo asignada inicialmente de un empleado a una tarea y para ajustar

estos valores a través del curso del proyecto se diseñó el EBS.

$$\begin{bmatrix} \text{Lista de tareas} & (t_{p_1}, t_{p_2}, ..., t_{p_n}) \\ \text{Matriz de asignacion de empleados} & \begin{Bmatrix} pwh_{11} & pwh_{12} & ... & pwh_{1n} \\ pwh_{21} & pwh_{22} & ... & pwh_{21} \\ ... & ... & ... & ... \\ pwh_{m1} & pwh_{m2} & ... & pwh_{mn} \end{Bmatrix} \end{bmatrix} \tag{19}$$

Dado que la construcción de la lista de tareas y la matriz de asignación de empleados se explicará en la siguiente sección, se procederá a detallar el funcionamiento del EBS.

El EBS es el encargado de asignar iterativamente, utilizando una unidad de tiempo, empleados a cada una de las tareas a través de la planificación. En este algoritmo se utiliza de unidad de tiempo t en meses y las asignaciones realizadas por el EBS solo ocurren cuando sucede un evento. Se considera un evento el mes t si es que satisface alguna de las siguientes condiciones:

1. $t = 1$ es el comienzo del proyecto.
2. Un empleado se une o deja el proyecto en t.
3. Si finalizó una tarea en $t - 1$ y los empleados asignados a esa tarea fueron liberados.

Luego el ajuste de la carga de trabajo de acuerdo a (19) se basa en dos reglas. Primero, si ocurre un conflicto de recursos entre dos tareas, la tarea que aparezca primero en la lista de tareas (19) tendrá mayor prioridad para utilizar los recursos en conflicto. Segundo, solo si un evento ocurre, ocurrirán nuevas asignaciones de trabajo, es decir, si no ocurre un evento las cargas de trabajo se mantienen iguales al mes anterior.

Algoritmo 3 Pseudocódigo de EBS

```
1: procedure EBSSCHEDULER(                                                    ▷ )EBS
2:     empleados                                          ▷ Inicializar empleados para el proyecto
3:     eventos          ▷ Agregar como eventos cuando empleados se unen o dejan el proyecto e inicio de este
4:     t ← 1
5:     while proyecto no ha finalizado do
6:         if t es un evento then
7:             tareas = array()          ▷ Tareas que pueden ser implementadas en t de acuerdo a la lista de tareas
8:             while tareas no esté vacío do
9:                 tarea_j                          ▷ Seleccionar y extraer primera tarea desde tareas
10:                for i ← m do                                          ▷ Iterar sobre empleados
11:                    if pwh_ij ≤ horas restantes del i-ésimo empleado en t then
12:                        wh^t_ij ← pwh_ij
13:                    else
14:                        wh^t_ij ← horas restantes del i-ésimo empleado en t
15:                    end if
16:                end for
17:            end while
18:            Refinamiento local: permitir a empleados usar todas sus horas de trabajo en el proyecto
19:        end if
20:        Evaluar estado de tareas
21:        if Si alguna tarea finalizó then
22:            eventos ← eventos + (t + 1)
23:            Refinamiento local: liberar cargas de trabajo redundantes.
24:        end if
25:        t = t + 1
26:        Actualizar habilidades de empleados
27:        Actualizar experiencia de empleados
28:        Actualizar lista de tareas y matriz de asignación
29:    end while
30: end procedure
```

Considerando el pseudocódigo del EBS (3) se detallará el modelo de entrenamiento (línea 26) y experiencia (línea 27) agregados al EBS original y cómo varían a través del curso de la planificación del proyecto.

Modelo de entrenamiento de empleados

A diferencia del algoritmo original (ACO-SPS) que consideraba que el nivel de habilidades de cada empleado se mantenían constantes a través del desarrollo o planificación del proyecto, el nuevo algoritmo ACO-SPS++ utilizando el modelo de entrenamiento descrito anteriormente, si el i-ésimo empleado tiene una competencia inicial s_i^{j0} para una habilidad específica j, su competencia después de T días de entrenamiento será:

$$s_i^j(T) = \begin{cases} s_i^{j0} & \text{si } T = 0 \\ s_i^j(T-1) + K \cdot s_i^j(T-1) \cdot (1 - \frac{s_i^j(T-1)}{5}) & \text{si } T > 0 \end{cases} \tag{20}$$

donde K es la velocidad de aprendizaje de un empleado.

Modelo de experiencia de empleados

Además del modelo de entrenamiento a nivel de habilidades y a diferencia del algoritmo original, también se incorporó un modelo de entrenamiento para cada

empleado. Un empleado puede tener baja competencia en una tarea, pero si trabaja en esta tarea el tiempo suficiente, se volverá más competente como resultado de la experiencia. Esto a menudo se conoce como "capacitación en el trabajo" (*On-the-job training*) [19]. Se supone que cada empleado tiene experiencia inicial (e_i^{n0}) para cada tarea.

Si el nivel de experiencia de un i-ésimo empleado inicialmente es e_i^{n0} para una tarea n específica al comienzo del proyecto, si trabaja T meses en esta tarea, entonces su experiencia al final de T será:

$$e_i^n(T) = \begin{cases} e_i^{n0} & \text{si } T = 0 \\ e_i^n(T-1) + K \cdot e_i^n(T-1) \cdot (1 - \frac{e_i^n(T-1)}{5}) & \text{si } T > 0 \end{cases} \tag{21}$$

Para evitar confusiones, debe tenerse en cuenta que este modelo utiliza el término "experiencia" de manera diferente a otros modelos existentes. Para otros modelos, las categorías de experiencia son muy amplias, fijas y se miden en años de experiencia laboral en cierta escala (por ejemplo, 1-7). Este criterio se usa para predecir tiempos para actividades grandes a largo plazo, por lo que este rango más amplio es apropiado. En cambio, acá las categorías de experiencia son específicas de las tareas y se utilizan para predecir los tiempos de finalización de actividades pequeñas y específicas. Por lo tanto, en este trabajo, es apropiado que permitamos que el nivel de experiencia aumente durante el curso del proyecto.

Dado que se modifican las habilidades y experiencias de cada empleado en cada iteración del EBS, se debe actualizar la lista de tareas y la matriz de asignación con los nuevos valores de habilidades y experiencias de los empleados utilizando los métodos de construcción descritos en la siguiente sección.

5.4. Algoritmo ACO propuesto

Un algoritmo ACO trabaja utilizando un grupo artificial de hormigas para construir soluciones al problema iterativamente, el cual puede ser descrito en tres procedimientos:

1. **Construcción de la solución**: Por cada iteración del algoritmo un grupo de hormigas artificiales construyen una solución al problema. Cada hormiga, de manera constructiva va seleccionando componentes para formar una solución final. Esta selección es guiada por las feromonas y heurísticas. Cada feromona es un registro de la experiencia pasada de otra hormiga que guía a las siguientes para tomar decisiones. Entonces mientras más feromonas pertenezcan a las mejores soluciones atraerán a las próximas hormigas en iteraciones posteriores. Por otro lado la heurístaca ayuda a las hormigas a tener mayor probabilidad de seleccionar mejores componentes para la construcción de la solución.

2. **Administración de feromonas**: A través de las iteraciones para la construcción de soluciones, las feromonas se van actualizando de acuerdo a la calidad

de cada solución. Las hormigas depositan más feromonas a los componentes con mayor calidad o mejor solución. El algoritmo posee feromonas locales y globales. La actualización de la feromona local ocurre luego que una hormiga hace una selección de un componente reduciendo su valor, permitiendo así que las siguientes hormigas tengan una mayor oportunidad de elegir otro posible componente para la solución. Por otro lado, la feromona global aumenta el valor de las feromonas asociadas a los componentes de la mejor solución encontrada en cada iteración. En otras palabras, en cada iteración, para el grupo de hormigas será más atractivo la selección de componentes de las mejores soluciones encontradas.

3. **Acciones *"demonio"*** (Deamon actions): Se pueden utilizar para implementar acciones centralizadas que no pueden ser realizadas por hormigas individuales. Algunos ejemplos son la activación de un procedimiento de optimización local o la recopilación de información global que puede usarse para decidir si es útil o no depositar una feromona adicional para sesgar el proceso de búsqueda desde una perspectiva no local. Como ejemplo práctico, el demonio puede observar el camino encontrado por cada hormiga en la colonia y elegir depositar una feromona adicional en los componentes utilizados por la hormiga que construye la mejor solución. Para el trabajo realizado se utiliza una mutación de la matriz de asignación de empleados. A partir de la mejor solución encontrada en cada iteración se selecciona una tarea t_j al azar. Luego un empleado i asignado a esa tarea se selecciona al azar y su carga de trabajo para esa tarea se resetea a $pwh_{ij} = 0$. Luego otro empleado u que no haya sido asignado a t_j se selecciona al azar y su carga de trabajo pwh_{uj} es asignada aleatoreamente entre los valores $\{25\,\%nh, 50\,\%nh, ..., maxh_u\}$. Si la nueva solución es mejor, reemplaza a la actual mejor solución de la iteración. La cantidad de mutaciones $mutatenum$ por iteración es definido como parámetro del algoritmo.

Para resolver el problema propuesto se utilizó la variante de un algoritmo ACO llamado ACS [15], el cual se destaca principalmente en dos aspectos. Primero, en el procedimiento de construcción de la solución, el ACS aplica una regla de selección proporcional pseudoaleatoria que sesga agresivamente la selección de los componentes con la feromona máxima (con mayor valor) y los valores heurísticos. De esta manera, el ACS explota fuertemente la experiencia de búsqueda pasada de hormigas y tiene una velocidad de convergencia rápida. Segundo, en el procedimiento de gestión de feromonas, ACS tiene dos reglas de actualización de feromonas, es decir, la actualización global y la actualización local. La regla de actualización global hace que los componentes correspondientes a la mejor solución hasta ahora sean más atractivos. La regla de actualización local reduce la feromona en los componentes recién seleccionados por las hormigas para aumentar la diversidad de búsqueda del algoritmo. Con estas características y basándose en el enfoque ACS se presenta el flujo del nuevo algoritmo ACO-SPS++ en la figura 14 donde se agregan las

interacciones del modelo de entrenamiento y experiencia en negrita:

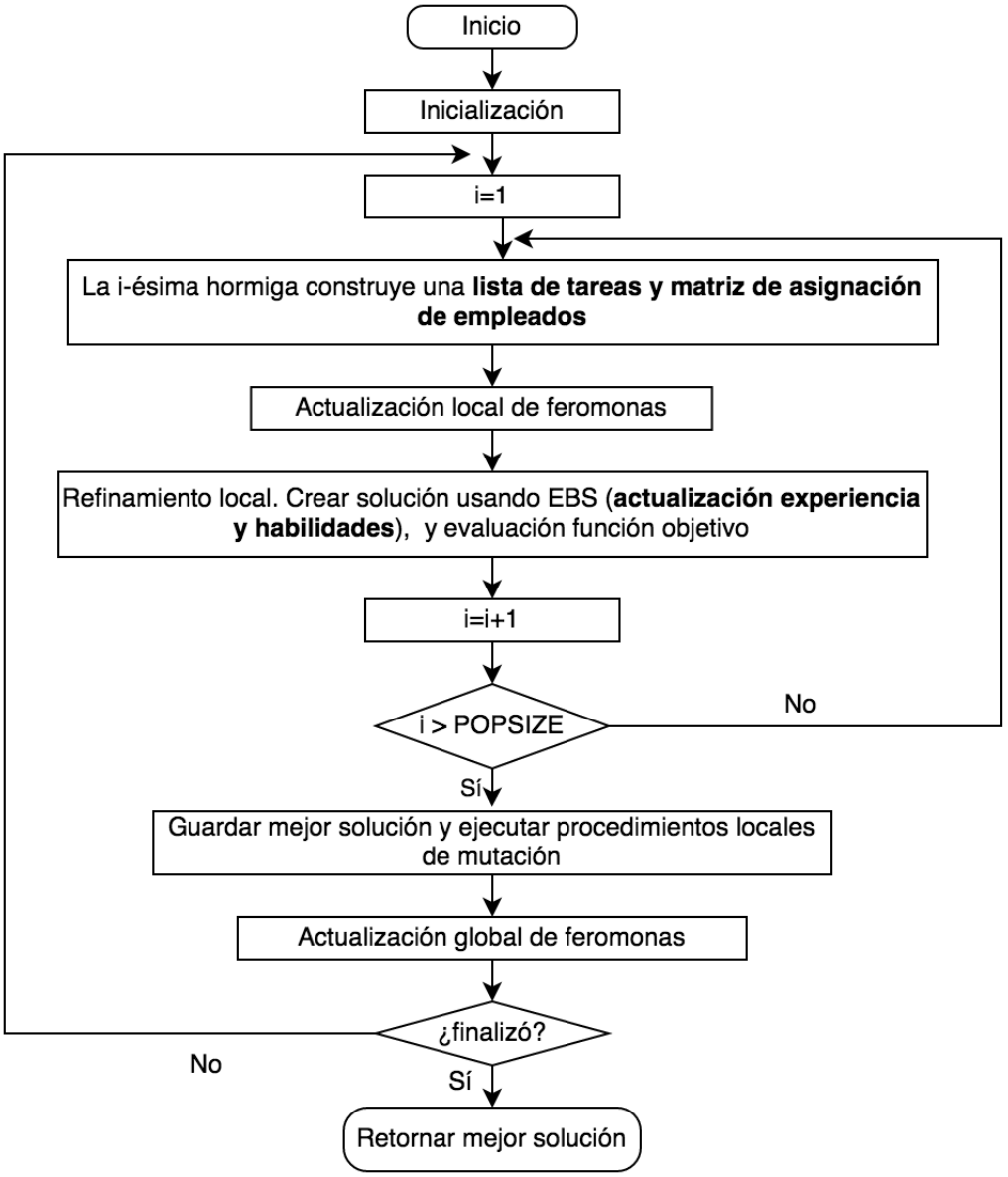


Figura 14: Flujo de algoritmo ACO-SPS++

La inclusión del nuevo modelo de entrenamiento y experiencia alteró el flujo del algoritmo original en tres puntos: (1) La construcción de la lista de tareas, (2) la construcción de la matriz de asignación de empleados y (3) la actualización de las habilidades y experiencia de los empleados en cada iteración del EBS luego de las asignaciones. Los demás componentes del flujo están detallados en la publicación original [14] y solo se detallará los tres mencionados anteriormente.

Construcción de la lista de tareas

La lista de tareas es una lista ordenada $(t_{p_1}, t_{p_2}, ..., t_{p_n})$ que satisface las restricciones de precedencia definadas por el grafo TPG. Para la construcción de la lista de tareas el algoritmo hace uso de una feromona y una heurístisca. Tanto la feromona como el procedimiento en la contrucción de la lista se mantuvieron iguales al algoritmo ACO-SPS original [14], por lo tanto se detallará solamente la heurística utilizada en la construcción.

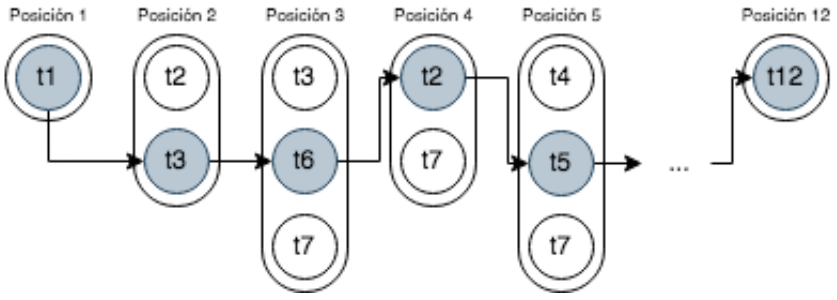


Figura 15: Ejemplo construcción de lista de tareas

La heurística utilizada para la contrucción de la lista de tareas es $MINSLK$ (*minimun slack*) [22]. Una tarea con un valor pequeño $MINSLK$ implica que la tarea es más urgente. El $MINSLK$ para una tarea t_j, el cual es denotado como $MINSLK_j$ puede ser estimado como se describe a continuación:

1. Estimar la menor duración de cada tarea. Se define duración como la diferencia entre la fecha de finalización e inicio de la tarea. Para estimar la menor duración de una tarea t_j primero se seleccionan $maxhead_j$ empleados más capacitados para t_j y se asume que estos empleados dedican todas sus horas de trabajo en t_j. $maxhead_j$ corresponde a la cantidad máxima de empleados para la tarea t_j y las capacidades de los empleados para t_j pueden ser evaluadas por (10). Dado esto, la menor duración posible de t_j puede ser estimada con (11) y (12).

2. Basados en las menores duraciones de cada tarea, se puede evaluar la fecha inicial y final de cada una de ellas, siendo el $MINSLK$ la diferencia entre ambas fechas. Entonces la heurística para la tarea $\eta_t(j)$, denotado como $n_t(j)$,

será:

$$\eta_t(j) = 1/MINSLK_j \tag{22}$$

Dado que las ecuaciones 11 y 12 dependen de las habilidades y experiencias de cada empleado, un cambio en estas, afectará la heurística $MINSLK$ y por lo tanto la construcción de la lista de tareas.

Construcción de la matriz de asignación de empleados

La matriz de asignación de empleados (19) especifica originalmente las horas de trabajo asignadas a los empleados para cada tarea. En general, las horas de trabajo asignadas pwh_{ij} al i-ésimo empleado para una tarea t_j pertenece al rango $[0, maxh_i]$. Para la construcción de esta matriz, al igual que para la lista de tareas, se utilizan dos feromonas y una heurística, siendo esta última modificada para integrarle el modelo de experiencia. Dado que las feromonas y el procedimiento de construcción no fueron alterados desde el algoritmo original, solo se detallará la heurística.

La heurística de elegir el i-ésimo empleado para trabajar en la tarea t_j se denota como $\eta_e(i, j)$ y se define como:

$$\eta_t(j) = taskfit_{ij}/hs_i \tag{23}$$

El significado de esta heurística es la tasa entre la competencia $taskfit_{ij}$ del i-ésimo empleado para la tarea t_j y el salario por hora del empleado. Un empleado con bajo salario y alta afinidad (habilidades y experiencia) para t_j es más probable que sea elegido para trabajar en la tarea t_j. Dado que $taskfit_{ij}$ depende de las habilidades y experiencias de cada empleado, un cambio en estas, afectará la heurística $MINSLK$ y por lo tanto la construcción de la matriz de asignación de empleados.

6. Validación

Para la validación de los algoritmos desarrollados, en los experimentos del algoritmo EWEHD se utilizaron 3 registros reales y 5 registros creados artificialmente. Para el algoritmo ACO-SPS++ se utilizaron 60 instancias sintéticas.

6.1. Algoritmo EWEHD

Respecto a los registros creados artificialmente, se definieron eventos, recursos y duraciones aleatoreamente. A partir de estos, se construyeron registros de ejecución de procesos, para finalmente utilizar estas instancias en la ejecución del algoritmo EWEHD y evaluar el correcto funcionamiento del algoritmo.

Estas instancias se generaron utilizando la siguiente estrategia:

- Definir un conjunto de eventos y recursos.
- Definir la cantidad de trazas y aleatoreamente la cantidad de eventos en cada una de ellas, asegurando que contengan al menos un evento.
- Definir aleatoreamente a partir del conjunto de eventos y recursos, la cantidad que cada evento aparecerá en cada traza y los recursos que utilizará, asegurando que al menos exista un recurso por evento.
- A partir de la cantidad de registros de cada tipo de evento, generar aleatoreamente un conjunto de duraciones para cada uno de ellos y calcular la desviación estándar, mediana, mínimos, máximos y los recursos utilizados.
- Conociendo las duraciones de cada evento, construir las trazas tal que el atributo *timestamp* de cada evento permita obtener estas duraciones.
- Utilizar instancia en el algoritmo EWEHD.

Basado en la estrategia mencionada, se generaron 5 instancias aleatoreas. La información básica de cada instancia está detallada en el cuadro 1.

Nombre	Eventos	Recursos	Trazas	Rango eventos por traza
a10-10-10	10	10	10	[1, 5]
a15-15-20	15	15	20	[1, 10]
a20-20-30	20	20	30	[1, 20]
a25-25-40	25	25	40	[1, 30]
a30-30-50	30	30	50	[1, 40]

Cuadro 1: Información de instancias artificiales.

Para la realización de los experimentos con registros reales se utilizaron archivos XES obtenidos desde la base de datos disponible en data.4tu.nl, los cuales pertenecen

Nombre	Nº de trazas	Nº de eventos	Nº promedio eventos por traza
Hospital Billing - Event Log	100000	451359	4.51
WABO, CoSeLoG project	1434	8577	5.98
Sepsis Cases - Event Log	1050	15214	14.49

Cuadro 2: Información de instancias reales.

a registros de procesos reales de empresas. La información básica de estas instancias se detallan en el cuadro 2.

Para obtener más información de cada uno de los archivos utilizados, se pueden revisar sus extractos en el anexo 9.1.

6.1.1. Experimentos

Utilizando las instancias detalladas anteriormente se desarrollaron los siguientes experimentos:

- Sobre las instancias artificiales, evaluar y comparar los resultados obtenidos a través de la ejecución del algoritmo EWEHD con los definidos en la generación de las instancias.
- Sobre las instancias reales, evaluar tiempos de ejecución y estimaciones de duración al utilizar el algoritmo en modo de procesamiento ISOLATED y CROSSED y si estos pueden ser de ayuda para la estimación cargas de trabajo en proyectos similares.
- Comparar algoritmo EWEHD con COCOMO II respecto a usabilidad, requisitos para su ejecución y fiabilidad de resultados.

6.1.2. Resultados

Los resultados obtenidos al ejecutar el algoritmo EWEHD sobre las instancias artificales en comparación a los establecidos inicialmente en su generación fueron los siguientes:

Nombre	% de similitud de duraciones	% de similitud de tareas
a10-10-10	100	100
a15-15-20	100	100
a20-20-30	100	100
a25-25-40	100	100
a30-30-50	100	100

Cuadro 3: Resultados al ejecutar el algoritmo EWEHD en las instancias artificiales en modo de proceso ISOLATED y modo de estimación STRICT.

Como se observa en el cuadro 3 la ejecución del algoritmo EWEHD logró igualar y obtener los mismos valores que fueron definidos al momento de la creación de las instancias artificiales, tanto recursos como estimación de las duraciones.

Por otro lado, los resultados obtenidos al ejecutar el algoritmo EWEHD en cada instancia real, utilizando el modo de procesamiento de trazas $ISOLATED$ y $CROSSED$, describiendo estimación mínima, máxima, media y desviación estándar son presentados a continuación. Destacar que todos los experimentos fueron realizados utilizando el modo STRICT para la estimación de la duración en **segundos**.

Para la instancia Hospital Billing - Event Log ejecutando el algoritmo EWEHD en modo ISOLATED y CROSSED, los resultados fueron los siguientes:

Archivo XES	**Modo de proceso**	**Modo estimación**	**Duración (seg)**	
Hospital Billing	ISOLATED	STRICT	30	
Eventos				
Nombre/Tipo	**Mínimo**	**Máximo**	**Mediana**	**Desviación Estándar**
CODE OK	0	5.79E7	348155.5	3935277.15
NEW	1	11	1	0.36
STORNO	0	5.88E7	1821769	1.11E7
REOPEN	0	7.30E7	206325	3518478.83
REJECT	0	3.39E7	21	3483627.65
CHANGE END	9	8697045	94145	2227916.06
FIN	0	4.14E7	7569647.5	4147736.66
RELEASE	0	5.19E7	22669	1332760.66
BILLED	0	6.19E7	695506	4524985.53
DELETE	0	4.94E7	50005.5	3600143.19
ZDBC-BEHAN	174655	174655	174655	0
JOIN-PAT	0	6.22E7	1	9337801.24
CODE NOK	1	4.93E7	433694.5	5473739.37
CODE ERROR	2	990812	259327	223410.36
MANUAL	4	5.72E7	1426391	1E7
SET STATUS	0	7.16E7	14266.5	8337143.90
EMPTY	2	1.13E7	18627	680868.44
CHANGE DIAGN	4	3.71E7	627724.5	2464266.04

Cuadro 4: Resultados al ejecutar el algoritmo EWEHD en el archivo XEX Hospital Billing - Event Log en modo de proceso *ISOLATED* y modo de estimación *STRICT*.

Archivo XES	**Modo de proceso**	**Modo estimación**	**Duración (seg)**	
Hospital Billing	CROSSED	STRICT	19	
Eventos				
Nombre/Tipo	**Mínimo**	**Máximo**	**Mediana**	**Desviación Estándar**
CODE OK	0	45578	53	418.48
NEW	0	24002	1	833.11
STORNO	0	18034	219	773.96
REOPEN	0	23008	173	695.53
REJECT	0	6141	17	384.72
CHANGE END	2	1329	186	319.70
FIN	0	45096	164	844.43
RELEASE	0	33977	18	472.36
BILLED	0	20658	110	498.06
DELETE	0	58943	219	1280.87
ZDBC-BEHAN	6297	6297	6297	0
JOIN-PAT	0	5984	1	472.78
CODE NOK	0	7293	42	248.13
CODE ERROR	1	352	41	89.55
MANUAL	1	34133	168.5	1825.57
SET STATUS	0	13196	110	891.86
EMPTY	1	7977	331	1216
CHANGE DIAGN	0	75162	295	1143.88

Cuadro 5: Resultados al ejecutar el algoritmo EWEHD en el archivo XEX Hospital Billing - Event Log en modo de proceso *CROSSED* y modo de estimación *STRICT*.

Como se observa arriba en los cuadros 4, 5 y el cuadro 13 presente en el apéndice 9.5, se logró obtener las duraciones y los recursos de todas las tareas contenidas en el archivo XES, lo que permite estimar la carga de trabajo requerida para cada una de ellas.

Para la instancia Sepsis Cases - Event Log ejecutando el algoritmo EWEHD en modo ISOLATED y CROSSED, los resultados fueron los siguientes:

Archivo XES	**Modo de proceso**	**Modo estimación**	**Duración (seg)**	
Sepsis Cases	ISOLATED	STRICT	3	
Eventos				
Nombre/Tipo	**Mínimo**	**Máximo**	**Mediana**	**Desviación Estándar**
CRP	0	874800	0	89157.99
Release B	610	562500	73792	130681.24
Release A	960	1651500	166749	176362.36
Release D	7200	959400	228570	265189.61
Release C	9000	1650600	366600	416711.45
Release E	10800	619200	75150	212903.05
Admission IC	0	503868	6484	49820.51
Return ER	25191	3.60E7	4083842	7731161.74
ER Triage	10	113577	378	10124.57
IV Antibiotics	0	31054	40	4447.64
Leucocytes	0	1306800	491	97435.76
IV Liquid	0	83473	953	13062.44
ER Registration	64	46948	14327	12537.99
Admission NC	5	626400	9118.5	48141.23
LacticAcid	0	165494	0	22747.31
ER Sepsis Triage	1	40399	28	2184.90

Cuadro 6: Resultados al ejecutar el algoritmo EWEHD en el archivo XES Sepsis Case - Event Log en modo de proceso *ISOLATED* y modo de estimación *STRICT*.

Archivo XES	**Modo de proceso**	**Modo estimación**	**Duración (seg)**	
Sepsis Cases	CROSSED	STRICT	7	
Eventos				
Nombre/Tipo	**Mínimo**	**Máximo**	**Mediana**	**Desviación Estándar**
CRP	0	172800	0	7665.46
Release B	35	72300	5548.5	12645.70
Release A	0	97200	3600	7210.86
Release D	339	9900	4110	2883.46
Release C	0	19200	1920	4354.86
Release E	1800	10920	5752	3605.79
Admission IC	0	24480	2570	5200.23
Return ER	51	1771988	5253	135747.39
IV Antibiotics	0	21901	36	2990.91
ER Triage	10	116946	3349	10181.44
Leucocytes	0	172800	0	9580.22
IV Liquid	0	52123	565	4401.76
ER Registration	1	22969	3176	5687.95
Admission NC	0	43895	1928.5	4662.61
LacticAcid	0	68640	0	7360.62
ER Sepsis Triage	1	16777	28	1245.86

Cuadro 7: Resultados al ejecutar el algoritmo EWEHD en el archivo XES Sepsis Case - Event Log en modo de proceso *CROSSED* y modo de estimación *STRICT*.

Como se observa arriba en los cuadros 6, 7 y el cuadro 14 presente en el apéndice 9.5, se logró obtener las duraciones y los recursos de todas las tareas contenidas en el archivo XES, lo que permite estimar la carga de trabajo requerida para cada una de ellas.

Para la instancia WABO, CoSeLoG project ejecutando el algoritmo EWEHD en modo ISOLATED y CROSSED, los resultados fueron los siguientes:

Archivo XES	**Modo de proceso**	**Modo estimación**	**Duración (seg)**	
WABO, CoSeLoG project	ISOLATED	Strict	3	
Eventos				
Nombre/Tipo	**Mínimo**	**Máximo**	**Mediana**	**Desviación Estándar**
T09-1	16.27	466.94	56.36	152.01
T20	10.84	116.58	19.11	23.39
T04	0.63	2950473.63	31.74	180137.40
T07-4	15.44	598822.84	43.33	218915.65
T19	13.35	152.46	37.7	33.22
T09-2	591.95	591.95	591.95	0
T07-2	0.69	1821034.99	99.16	501916.26
T06	0.52	2.32E7	41.87	1051195.76
T18	11.88	22.14	17.70	3.63
T13	10.17	1478578.57	739294.37	739284.19
T09-3	13.96	3113940.41	37.93	1029809.37
T03	0.70	5967762.15	500.60	1152780.53
T09-4	15.95	2260.93	22.82	958.57
T10	11.67	3618586.27	31.91	249880.73
T14	12.27	593297.75	32.99	135381.26
T07-3	16.79	630312.69	546.87	267287.54
T11	0.48	7764094.62	1289.24	1213456.75
T16	16.53	1309130.36	180.61	287441.50
T02	0.58	9588298.16	633.49	824394.06
T05	8.64	1906331.95	36.35	183986.11
T07-5	18.89	1740933.93	82.63	420361.44
T17	14.23	784.59	37.99	177.07
T12	10.30	599894.15	24.9	93292.53
T15	9.95	1376242.22	9857.01	388113.73
T08	28.13	71991.39	72.53	20446.69
T07-1	10.54	2941755.27	60.18	588461.82

Cuadro 8: Resultados al ejecutar el algoritmo EWEHD en el archivo XES WABO, CoSeLoG project en modo de proceso *ISOLATED* y modo de estimación *STRICT*.

Archivo XES	**Modo de proceso**	**Modo estimación**	**Duración (seg)**	
WABO, CoSeLoG project	CROSSED	STRICT	4	
Eventos				
Nombre/Tipo	**Mínimo**	**Máximo**	**Mediana**	**Desviación Estándar**
T09-1	16.27	466.94	56.36	152.01
T20	10.84	116.58	19.11	23.39
T04	0.63	413572.02	29.60	20316.79
T07-4	15.44	1998.37	33.91	731.66
T19	13.35	152.46	34.47	31.65
T09-2	189.88	189.88	189.88	0
T07-2	0.69	89962.44	99.16	15588.53
T06	0.06	934102.21	67.98	38551.58
T18	11.88	22.14	17.70	3.63
T13	10.17	2769.88	1390.02	1379.85
T09-3	13.96	1012.88	37.93	318.95
T03	0.70	231419.92	267.54	32364.43
T09-4	15.95	2260.93	22.82	891.98
T10	5.27	395538.12	30.84	13638.95
T07-3	16.79	4121.98	132.60	1707.44
T14	12.27	17261.73	32.99	3021.64
T11	0.48	70370.64	847.75	16253.89
T16	16.53	55759.82	154.97	12235.32
T02	0.27	518399.58	1327.68	48009.67
T05	0	327717.42	33	18432.08
T07-5	8.30	69090.84	66.58	17311.28
T17	14.23	784.59	37.99	167.37
T12	5.16	234298.78	24.9	36068.91
T15	6.88	251546.07	218.25	43485.72
T08	28.13	60692.33	72.53	13806.95
T07-1	10.54	251920.03	56.65	51733.61

Cuadro 9: Resultados al ejecutar el algoritmo EWEHD en el archivo XES WABO, CoSeLoG project en modo de proceso *CROSSED* y modo de estimación *STRICT*.

Como se observa arriba en los cuadros 8, 9 y el cuadro 15 presente en el apéndice 9.5, se logró obtener las duraciones y los recursos de todas las tareas contenidas en el archivo XES, lo que permite obtener la carga de trabajo requerida para cada una de ellas.

Cabe destacar que el uso del modo de proceso $CROSSED$ reduce los valores de las duraciones de las tareas en todas las instancias. En especial al considerar el valor de la desviación estándar, que refleja una menor dispersión en los valores obtenidos y como consecuencia podría implicar una mejor toma de decisiones.

6.2. Algoritmo ACO-SPS++

Los instancias artificiales se derivan desde PSPLIB [3], siendo estos ampliamente utilizados en estudios de gestión de proyectos en diversos campos (incluida la gestión de proyectos de software). Por lo tanto, se considera que estas redes de proyectos de referencia son útiles para capturar las relaciones de precedencia de tareas de los proyectos de software. Para garantizar que las instancias generadas aleatoreamente capturen correctamente las características de los proyectos del mundo real, se generaron las instancias utilizando las siguientes estrategias:

- Definir 4 tipos de empleados: personal experto regular, personal normal regular, personal experto temporal y personal normal temporal. El personal experto es bueno en una o más habilidades y su salario es alto. Personal normal son buenos en una o dos habilidades y ellos son la mayoría dentro de un proyecto. El personal temporal se contrata temporalmente al proyecto y solo se paga por las horas invertidas en este. Bajo estos criterios las propiedades de cada empleado son generadas aleatoriamente de acuerdo al tipo de personal. Finalmente, se creó un componente que se encarga de verificar que todas las habilidades requeridas para el proyecto han sido cubiertas.

- Definir la carga de trabajo (persona mes) para cada tarea aleatoriamente y la cantidad máxima de personal para cada tarea usando el modelo COCOMO. Para esto último, se utilizó el modelo COCOMO, siendo posible usar otros modelos o técnicas, pero siempre manteniéndolo para en todo el proceso. De acuerdo al modelo de COCOMO [7], la fórmula para el tiempo de desarrollo de una tarea es:

$$TDEV = 3 * PM^{(0{,}33+0{,}2\cdot(B-1{,}01))} \tag{24}$$

  Donde PM es el esfuerzo estimado en persona-mes. Si fijamos $B = 1$, asumiendo que las economías y deseconomías de escala están en equilibrio, obtenemos $TDEV = 3 * PM^{0{,}328}$. Luego la cantidad promedio de empleados para una tarea será $\frac{PM}{TDEV} \approx \frac{1}{3}PM^{0{,}672}$. Para este estudio se decidió doblar el tamaño (o 1 si PM es muy pequeño), es decir, $maxhead_j = max\left\{1, round(\frac{2}{3}PM^{0{,}672})\right\}$

Basado en las estrategias mencionadas se generaron 60 instancias aleatoreas usando 60 diferentes proyectos desde PSPLIB. La información básica de cada instancia esta detallada en el cuadro 10.

Nombre	Tareas	Habilidades	Empleados	Soluciones
j301_1 – j301_10	32	5	10	100
j302_1 – j302_10	32	5	15	100
j303_1 – j303_10	32	10	20	100
j601_1 – j601_10	62	5	10	200
j602_1 – j602_10	62	5	15	200
j603_1 – j603_10	62	10	20	200

Cuadro 10: Información de instancias artificiales. La columna **soluciones** indica la cantidad de veces que se ejecutó el algoritmo sobre cada instancia, para luego elegir la mejor.

6.2.1. Experimentos

Utilizando las instancias detalladas anteriormente se desarrollaron los siguientes experimentos:

- La función objetivo de este algoritmo considera minimizar el costo del proyecto de software. Por lo tanto, primero se comparan los costos de proyectos de las soluciones obtenidas por el algoritmo ACO-SPS y ACO-SPS++ y la duración para el cálculo de cada solución.

- Comparar la estructura de las soluciones encontradas tanto por el algoritmo ACO-SPS y ACO-SPS++, respecto a los diagramas de Gantt de cada plan.

Los parámetros iniciales propuestos para los experimentos fueron los siguientes: el número de hormigas es *POPSIZE=10*, la tasa de actualización de feromonas ρ es 0,1, el parámetro β en el procedimiento que genera la matriz de asignación de empleados es 2. Estos parámetros pueden influenciar significativamente en el desempeño del algoritmo. En el trabajo original [14] se probaron varias configuraciones y se encontró que la configuración original sugerida por Dorigo y Gambardella [15] todavía contribuye a un buen desempeño del algoritmo. Entonces para ser consistentes con la comparación se mantuvo la misma configuración.

Respecto a los tres parámetros q_t, q_{e1} y q_{e2} utilizados en las reglas proporciales pseudoaleatorias desarrolladas en el ACS [15] en el procedimiento de construcción de la solución, fueron asignados q_t y q_{e2} como 0,5, utilizados para la construcción de la lista de tarea y la selección de la carga de trabajo de un empleado para una tarea, respectivamente y parámetro q_{e1} usado para la selección de empleados se asignó como 0,9. Nuevamente estos valores fueron definidos en base al trabajo original, ya que guían al algoritmo a un mejor desempeño.

6.2.2. Resultados

Los resultados obtenidos, tanto el mejor como el promedio, utilizando la comparación de costos obtenidos por cada algoritmo son presentados en la siguiente tabla:

	mejor	promedio	mejor	promedio	mejor	promedio	mejor	promedio
instancia	j301_1		j301_2		j301_3		j301_4	
ACO-SPS	**1105190**	**1128640**	1339190	1382930	1292190	1320270	1232190	1252200
ACO-SPS++	1164190	1226620	**1302190**	**1359030**	**1133190**	**1187890**	**1053190**	**1138420**
instancia	j301_5		j301_6		j301_7		j301_8	
ACO-SPS	1479190	1508010	852021	1240588	1412190	1430990	1397190	1431770
ACO-SPS++	**1262190**	**1374540**	**789060**	**984885**	**1263190**	**1318940**	**1223190**	**1278160**
instancia	j301_9		j301_10		j302_1		j302_2	
ACO-SPS	1885190	1916160	1384190	1410520	1266190	1303370	1568190	1614930
ACO-SPS++	**1552190**	**1651830**	**1074190**	**1118910**	**969190**	**1035450**	**1315190**	**1481510**
instancia	j302_3		j302_4		j302_5		j302_6	
ACO-SPS	1576190	1620210	1854190	1885660	1307190	1338130	1870190	1898770
ACO-SPS++	**1233190**	**1357910**	**1654190**	**1739960**	**1206190**	**1257020**	**1756190**	**1843120**
instancia	j302_7		j302_8		j302_9		j302_10	
ACO-SPS	1437190	1471210	1642190	1677530	1622190	1659700	**1322190**	**1363860**
ACO-SPS++	**1178190**	**1261290**	**1546190**	**1607010**	**1372190**	**1442360**	1429190	1510420
instancia	j303_1		j303_2		j303_3		j303_4	
ACO-SPS	1772171	1790031	1790171	1821690	1780190	1815890	1817190	1855950
ACO-SPS++	**1521171**	**1599111**	**1457169**	**1549180.23**	**1321190**	**1417530**	**1544190**	**1633700**
instancia	j303_5		j303_6		j303_7		j303_8	
ACO-SPS	1431190	1457580	1495190	1516570	1791190	1813250	1659190	1714590
ACO-SPS++	**1198190**	**1262430**	**1286190**	**1348960**	**1527190**	**1610920**	**1353190**	**1448590**
instancia	j303_9		j303_10		j601_1		j601_2	
ACO-SPS	1593190	1623240	1708190	1720720	2874225	2910780	2782225	2828865
ACO-SPS++	**1359190**	**1429420**	**1275190**	**1379170**	**2321225**	**2408635**	**2436225**	**2580530**
instancia	j601_3		j601_4		j601_5		j601_6	
ACO-SPS	3032225	3075295	2365225	2435800	3190225	3253735	3051041	3151156.3
ACO-SPS++	**2578225**	**2713600**	**2093225**	**2172075**	**3001225**	**3124410**	**2786152**	**2895463.695**
instancia	j601_7		j601_8		j601_9		j601_10	
ACO-SPS	2545225	2596615	3093176	3126475.785	2865225	2902530	3130225	3171915
ACO-SPS++	**2270225**	**2355205**	**2472176**	**2589785**	**2382225**	**2548525**	**2825225**	**2926235**
instancia	j602_1		j602_2		j602_3		j602_4	
ACO-SPS	2804176	2908454.975	3021225	3080450	2857225	2909765	2408176	2474510
ACO-SPS++	**2418176**	**2514638.07**	**2729225**	**2830725**	**2455225**	**2583510**	**2231209**	**2330669**
instancia	j602_5		j602_6		j602_7		j602_8	
ACO-SPS	3144176	3186874	3048225	**3108040**	3205225	3243735	2971225	3025905
ACO-SPS++	**2534199**	**2694605**	**2988225**	3151690	**2732225**	**2823050**	**2535225**	**2635045**
instancia	j602_9		j602_10		j603_1		j603_2	
ACO-SPS	2729225	2779445	2944225	3012165	3432225	3485470	3563171	3612864
ACO-SPS++	**2640225**	**2767160**	**2349225**	**2440820**	**2860225**	**3108380**	**3245170**	**3364478**
instancia	j603_3		j603_4		j603_5		j603_6	
ACO-SPS	2908741	3269983	3337225	3383670	3917225	3940835	3046176	3133279
ACO-SPS++	**2245686**	**2654253**	**2574225**	**2690775**	**3133225**	**3254000**	**2631203**	**2788182**
instancia	j603_7		j603_8		j603_9		j603_10	
ACO-SPS	3400225	3447815	3221161	3264650	3668225	3714375	2998225	3061760
ACO-SPS++	**2644225**	**2740290**	**2654171**	**2812909**	**3149225**	**3284235**	**2366225**	**2519220**

Cuadro 11: Comparación de resultados de costos de las 60 instancias.

Para una mejor representación de los resultados se presenta la comparación de los mejores costos de cada algoritmo en las siguientes figuras 16 y 17:

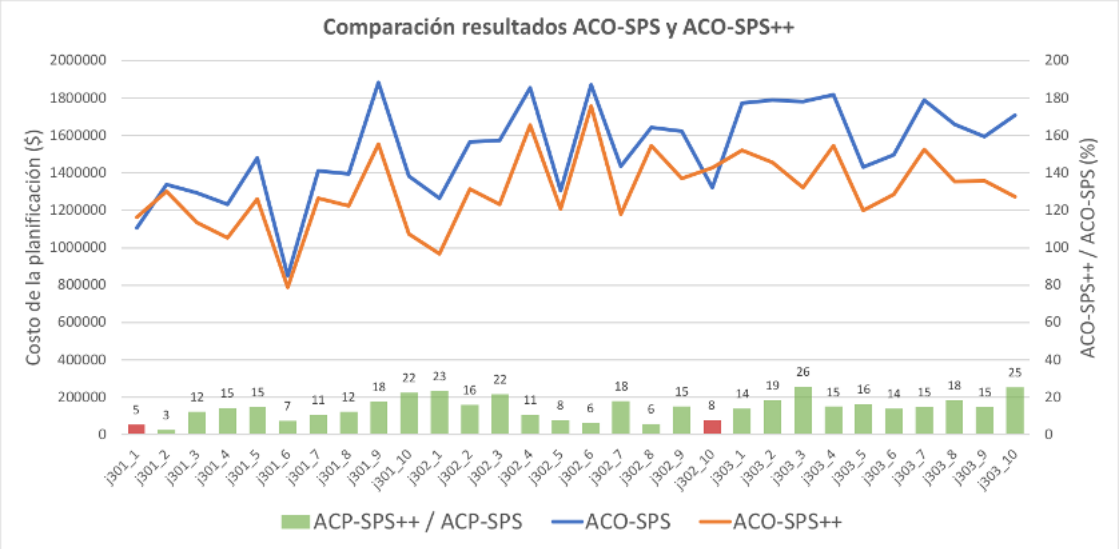


Figura 16: Comparación de costos entre los algoritmos ACO-SPS y ACO-SPS++ para las instancias j301_X, j302_X y j303_X.

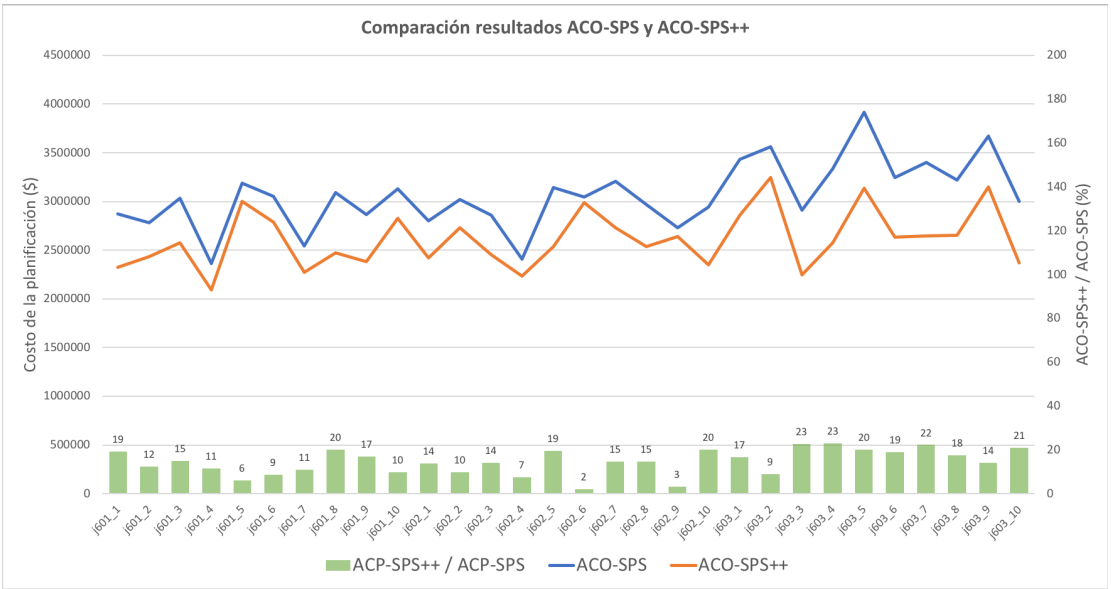


Figura 17: Comparación de costos entre los algoritmos ACO-SPS y ACO-SPS++ para las instancias j601_X, j602_X y j603_X.

Como se puede observar en el cuadro 11 y figuras 16 y 17, el nuevo algoritmo ACO-SPS++ obtuvo mejores resultados en casi todas las instancias, es decir, generó planificaciones de proyectos con menor costo en comparación al algoritmo ACO-SPS original, logrando reducirlos de entre un 3 % hasta un 26 %. Por otro lado, el costo

de planificación de las instancias j301_1 y j302_10 fue mayor, 5 % y 8 % respectivamante. Si bien no se hizo un estudio sobre las causas que guiaron al algoritmo a llegar a esa solución, se puede deber a la estructura del grafo de precedencia de tareas o las características intrínsecas de los empleados y tareas que fueron seleccionadas artificialmente.

Los resultados obtenidos, tanto el mejor como el promedio, utilizando para la comparación los tiempos (segundos) de ejecución obtenidos por cada algoritmo son presentados en el siguiente cuadro 12:

	mejor	promedio	mejor	promedio	mejor	promedio	mejor	promedio
instancia	j301_1		j301_2		j301_3		j301_4	
ACO-SPS	**0.122**	**0.168**	**0.126**	**0.270**	**0.125**	**0.130**	**0.126**	**0.130**
ACO-SPS++	2.696	2.734	2.659	2.708	2.725	2.747	2.767	2.784
instancia	j301_5		j301_6		j301_7		j301_8	
ACO-SPS	**0.123**	**0.128**	**0.132**	**0.137**	**0.127**	**0.131**	**0.127**	**0.135**
ACO-SPS++	2.727	2.743	2.913	2.939	2.752	2.780	2.538	2.582
instancia	j301_9		j301_10		j302_1		j302_2	
ACO-SPS	**0.126**	**0.131**	**0.125**	**0.139**	**0.170**	**0.176**	**0.168**	**0.173**
ACO-SPS++	2.757	2.784	2.566	2.650	3.787	3.810	3.694	3.736
instancia	j302_3		j302_4		j302_5		j302_6	
ACO-SPS	**0.179**	**0.184**	**0.174**	**0.182**	**0.180**	**0.185**	**0.180**	**0.185**
ACO-SPS++	3.892	3.954	3.846	3.884	3.878	3.909	3.936	3.962
instancia	j302_7		j302_8		j302_9		j302_10	
ACO-SPS	**0.177**	**0.182**	**0.173**	**0.179**	**0.171**	**0.176**	**0.176**	**0.181**
ACO-SPS++	3.866	3.888	3.839	3.869	3.773	3.804	3.905	3.935
instancia	j303_1		j303_2		j303_3		j303_4	
ACO-SPS	**0.286**	**0.294**	**0.274**	**0.281**	**0.287**	**0.294**	**0.292**	**0.299**
ACO-SPS++	6.374	6.420	6.138	6.174	6.358	6.401	6.448	6.498
instancia	j303_5		j303_6		j303_7		j303_8	
ACO-SPS	**0.282**	**0.293**	**0.283**	**0.291**	**0.281**	**0.289**	**0.282**	**0.289**
ACO-SPS++	6.305	6.365	6.321	6.356	6.235	6.269	6.304	6.350
instancia	j303_9		j303_10		j601_1		j601_2	
ACO-SPS	**0.277**	**0.284**	**0.283**	**0.370**	**0.501**	**0.514**	**0.479**	**0.486**
ACO-SPS++	6.171	6.204	6.123	6.201	**18.552**	**18.937**	19.079	19.150
instancia	j601_3		j601_4		j601_5		j601_6	
ACO-SPS	**0.502**	**0.509**	**0.483**	**0.487**	**0.479**	**0.487**	**0.473**	**0.480**
ACO-SPS++	19.765	19.973	19.042	19.397	18.666	18.953	18.287	18.575
instancia	j601_7		j601_8		j601_9		j601_10	
ACO-SPS	**0.539**	**0.549**	**0.568**	**0.576**	**0.548**	**0.555**	**0.491**	**0.498**
ACO-SPS++	19.316	19.521	19.937	20.032	19.557	19.742	19.239	19.399
instancia	j602_1		j602_2		j602_3		j602_4	
ACO-SPS	**0.740**	**0.748**	**0.704**	**0.718**	**0.726**	**0.738**	**0.710**	**0.730**
ACO-SPS++	28.293	28.503	28.328	29	27.715	28.348	27.337	27.947
instancia	j602_5		j602_6		j602_7		j602_8	
ACO-SPS	**0.682**	**0.696**	**0.737**	**0.752**	**0.688**	**0.697**	**0.721**	**0.730**
ACO-SPS++	27.493	27.842	27.585	27.786	27.915	28.139	33.430	33.721
instancia	j602_9		j602_10		j603_1		j603_2	
ACO-SPS	**0.682**	**0.711**	**0.724**	**0.770**	**1.068**	**1.081**	**1.164**	**1.189**
ACO-SPS++	28.822	29.207	27.124	27.256	42.266	42.795	43.951	45.110
instancia	j603_3		j603_4		j603_5		j603_6	
ACO-SPS	**1.113**	**1.144**	**1.072**	**1.157**	**1.103**	**1.113**	**1.094**	**1.110**
ACO-SPS++	42.468	43.237	42.440	43.122	44.529	44.857	42.909	43.728
instancia	j603_7		j603_8		j603_9		j603_10	
ACO-SPS	**1.117**	**1.127**	**1.096**	**1.118**	**1.041**	**1.058**	**1.081**	**1.092**
ACO-SPS++	45.559	45.870	45.636	46.702	44.239	44.782	44.047	44.154

Cuadro 12: Comparación de resultados de tiempos (segundos) de ejecución de las 60 instancias.

Al contrario de los resultados anteriores, basados en el costo de la planificación, si ahora se considera el tiempo de ejecución de ambos algoritmos ACO-SPS++ y ACO-SPS, este último logró considerablemente mejores resultados. Esto era esperable debido a la constante generación de una nueva lista de tareas y matriz de asignación de empleados luego de cada asignación de empleados en cada unidad de tiempo de la planificación.

7. Conclusiones

Dada la naturaleza del trabajo realizado las conclusiones se dividirán en dos secciones, primero las del algoritmo EWEHD y luego las del algoritmo ACO-SPS++, ambas detalladas a continuación.

7.1. Algoritmo EWEHD

El objetivo principal del algoritmo EWEHD es poder estimar la carga de trabajo utilizando registros históricos para proyecto similares. Dado los resultados obtenidos luego de la ejecución del algoritmo sobre las instancias artificiales, para verificar la correctitud y sobre las instancias reales, a través de las duraciones mínimas, máximas, mediana, desviación estándar y los recursos utilizados en cada tarea es posible realizar esta estimación. Utilizando la cantidad de personas/recursos utilizados y el tiempo de realización se puede estimar el esfuerzo necesario como:

$$E(t) = P(t) * T(t) \tag{25}$$

Siendo para cada tarea t, el esfuerzo necesario de trabajo E como el producto de P, recursos utilizados y T tiempo de desarrollo.

En comparación al modelo constructivo de costos COCOMO y sus tres sub modelos: composición de aplicación, diseño temprano y post-arquitectura con sus ecuaciones para estimar la carga de trabajo de un proyecto (1), (3) y (4) respectivamente, que propone dejar en manos del encargado definir el conjunto de parámetros bajo su experiencia y visión del proyecto, no considerar el uso de librerías, frameworks o paradigmas de programación y medir el coste del producto de acuerdo a su tamaño y otras características. Lo anterior puede generar resultados poco fiables, debido en general, a que cada estimación o elección de parámetros pueden ser seleccionados de manera distinta por diferentes analistas o encargados, para un mismo proyecto.

EWEHD, en cambio, necesita registros históricos de un proyecto y que obviamente, estos registros pertenezcan a un proyecto de similares características. Si bien este hecho puede llevar a que la estimación no sea del todo acertada por las posibles diferencias, igualmente EWEHD puede ser una alternativa para estimaciones en etapas tempranas del desarrollo como lo es COCOMO con su modelo Composición de la Aplicación, considerando cada una de las tareas por separado y cuya etapa no requiere de una estimación tan precisa. Además, junto a su utilización a través de ProM en forma de plugin, lo hace más usable debido a su flujo guiado, desde la selección del algoritmo, ingreso de parámetros y posterior selección del visualizador hasta el despliegue de los resultados, permitiendo a personas sin conocimientos de programación hacer uso de este algoritmo. También debido a que el desarrollo se realizó en el lenguaje de programación Java y como plugin de ProM, hospedado en un repositorio público en GitHub [8], ambos gratuitos, permite que el algoritmo sea

[8]Repositorio plugin: `https://github.com/ppbustamante/acoplus`

utilizado por cualquier persona y desde cualquier dispositivo con soporte de Java.

Es aquí donde EWEHD puede considerarse como una opción, ya que al hacer uso de registros históricos cuyo origen corresponden a ejecuciones de procesos reales de una empresa, hace de la estimación más fiable y cercana a la realidad y pueden extender los enfoques tradicionales para obtener una mayor precisión, ya que usando minería de datos, se puede no solo estimar los parámetros, sino también conocerlos de manera objetiva para la organización, el tipo de proyectos desarrollados y los recursos utilizados para etapas tempranas de la planificación.

Por último, mencionar la compatibilidad de los resultados obtenidos de la estimación, ya que al poseer unidades en recurso-tiempo, los hacen útiles y compatibles como parámetros para el algoritmo ACO-SPS++ para definir la carga de trabajo necesaria para las tareas a planificar.

Un tema importante como trabajo futuro para mejorar la calidad de la estimación es considerar como parámetro del algoritmo los resultados de otras ejecuciones del algoritmo en otros registros históricos y compararlos con otros métodos o modelos de estimación.

7.2. Algoritmo ACO-SPS++

La función objetivo del algoritmo ACO considerada en este trabajo, es la de minimizar el costo del desarrollo de software de la planificación, el cual está regido por los salarios de los empleados y las penalizaciones por atrasos. Estos atrasos se calculan luego que todas las tareas hayan sido completadas en la planificación, y corresponden a aquellas que superaron el tiempo máximo para ser completadas (16). A partir de esto, primero se compara el desempeño del algoritmo propuesto ACO-SPS++ con el algoritmo original ACO-SPS.

Si bien los tiempos de ejecución del algoritmo ACO-SPS++ son mayores que el algoritmo original para las instancias generadas comparando el mejor y el resultado promedio, en relación a los costos de planificación (cuadro 11) se observa que ACO-SPS++ produce los mejores resultados y el mejor promedio de resultados en casi todos los casos, es decir, logró obtener costos de planificación menores que el algoritmo original. En algunas instancias, ACO-SPS++ logró reducir los costos entre 10 % y 20 %. Esto se debe a la inclusión del nuevo modelo de entrenamiento y experiencia y cómo estos afectan en la construcción de la lista de tareas y la matriz de asignación de empleados (19) en cada iteración del planificador basado en eventos. Primero, la heurística utilizada para la construcción de la lista de tareas (22) hace uso de las ecuaciones (11) y (12) utilizando los $maxhead_j$ empleados más competentes para cada tarea. Dado que en cada iteración los empleados asignados a alguna tarea por el planificador de eventos mejoran sus habilidades, su aptitud general (10) también aumenta, permitiendo que el término de una tarea se complete antes (15) y como consecuencia posicionando estas tareas en un comienzo de la lista de tareas en cada iteración, mientras se cumplan las restricciones del TPG. Segundo, la contrucción

de la matriz de asignación de empleados hace uso de la heurística (23) de elegir un i-ésimo empleado para trabajar en una j-ésima tarea, la cual se define como la competencia de un empleado con dicha tarea sobre su salario por hora. Dado que su competencia aumenta en cada iteración y su sueldo es fijo, aumenta la probabilidad de elegir a estos empleados para trabajar en las tareas que requieran habilidades similares.

Analizando las estructuras de las soluciones encontradas por ambos algoritmos, se traza una sección de los diagramas de Gantt de los planes encontrados (figura 18) donde se puede observar la posible influencia del aprendizaje obtenido por el empleado 2# durante los meses de Octubre y Noviembre, mejorando sus habilidades y por lo tanto sus competencias para realizar la tarea #8 y elegido por sobre el empleado 1#.

ACO-SPS	task #1 1#200	task #4 1#100 3#100 10#120	task #9 4#200 9#100	task #5 7#100	task #3 7#100	task #8 1#100 3#100
ACO-SPS++	task #1 7#100	task #4 2#150 4#200 7#100	task #9 6#200 9#100	task #5 2#150	task #3 2#150	task #8 2#150 3#100
	07/2020	08/2020	09/2020	10/2020	11/2020	12/2020

Figura 18: Parte de la planificación construida por ACO-SPS y ACO-SPS++.

Como se ha mencionado, las técnicas, modelos o algoritmos de gestión y planificación de proyectos tradicionales y ampliamente aceptados, no logran hacer frente de manera efectiva a la naturaleza evolutiva y dinámica de los proyectos de software modernos, por lo que este nuevo algoritmo ACP-SPS++, que incluye un modelo de entrenamiento y experiencia, puede ser una alternativa viable para ayudar a guiar a los jefes de proyecto en sus rutinas de planificación en la actualidad.

Los temas potenciales para el trabajo futuro incluyen las siguientes mejoras al algoritmo y modelo:

- Mejor representación de las habilidades, empleados y tareas. Agrupar las habilidades en clases de niveles similares en dificultad y hacer que la tasa de aprendizaje en función de esta. Dividir a los empleados en categorías con habilidades preferidas y grupos predefinidos para la capacitación de estos y por último, definir categorías para las tareas y asignar niveles de criticidad de habilidades requeridas, creando así de manera efectiva una lista predeterminada de habilidades y niveles mínimos de competencia requeridos.
- Considerar el desarrollo de una interfaz de usuario para permitir de manera simple el requerimiento de los parámetros para la ejecución y un mejor despliegue de los resultados para el gerente o jefe de proyectos, como por ejemplo, una carta Gantt.
- Utilizar otros modelos de aprendizaje y comparar los resultados utilizando cada uno de ellos.

8. Bibliografía

[1] Java annotation. `https://docs.oracle.com/javase/8/docs/api/java/lang/annotation/Annotation.html`. Último acceso: 2021-04-13.

[2] Jcomponent. `https://docs.oracle.com/javase/7/docs/api/javax/swing/JComponent.html`. Último acceso: 2019-11-10.

[3] Project scheduling problem library - psplib. `https://www.om-db.wi.tum.de/psplib/`. Último acceso: 2021-04-13.

[4] Serializable. `https://docs.oracle.com/javase/7/docs/api/java/io/Serializable.html`. Último acceso: 2020-01-10.

[5] Ieee standard for extensible event stream (xes) for achieving interoperability in event logs and event streams. *IEEE Std 1849-2016*, pages 1–50, 2016.

[6] Ghaffari Abu and João Cangussu. A learning model for software development processes. pages 474–479, 01 2005.

[7] Barry Boehm, Chris Abts, B Clark, and S Devnani-Chulani. Cocomo ii model definition manual. *The University of Southern California*, 102, 1997.

[8] Barry W. Boehm, Clark, Horowitz, Brown, Reifer, Chulani, Ray Madachy, and Bert Steece. *Software Cost Estimation with Cocomo II with Cdrom*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition, 2000.

[9] Peter Brucker, Andreas Drexl, Rolf Möhring, Klaus Neumann, and Erwin Pesch. Resource-constrained project scheduling: Notation, classification, models, and methods. *European Journal of Operational Research*, 112(1):3 – 41, 1999.

[10] Peter Brucker, Andreas Drexl, Rolf Möhring, Klaus Neumann, and Erwin Pesch. Resource-constrained project scheduling: Notation, classification, models, and methods. *European Journal of Operational Research*, 112:3–41, 02 1999.

[11] J.C.A.M. Buijs. Environmental permit application process ('WABO'), CoSeLoG project – Municipality 5. 6 2014.

[12] Carl K. Chang, Mark J. Christensen, and Tao Zhang. Genetic algorithms for project management. *Annals of Software Engineering*, 11(1):107–139, 2001.

[13] Carl K. Chang, Hsin yi Jiang, Yu Di, Dan Zhu, and Yujia Ge. Time-line based model for software project scheduling with genetic algorithms. *Information and Software Technology*, 50(11):1142 – 1154, 2008.

[14] W. N. Chen and J. Zhang. Ant colony optimization for software project scheduling and staffing with an event-based scheduler. *IEEE Transactions on Software Engineering*, 39(1):1–17, Jan 2013.

[15] M. Dorigo and L. M. Gambardella. Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53–66, 1997.

[16] M. Dorigo, V. Maniezzo, and A. Colorni. Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 26(1):29–41, 1996.

[17] Merrill M Flood. The traveling-salesman problem. *Operations research*, 4(1):61–75, 1956.

[18] N. Hanakawa, S. Morisaki, and K. Matsumoto. A learning curve based simulation model for software development. In *Proceedings of the 20th International Conference on Software Engineering*, pages 350–359, 1998.

[19] Harold Kerzner. Project management : A systems approach to planning, scheduling, and controlling / h. kerzner. 01 1979.

[20] Felix Mannhardt. Sepsis Cases - Event Log. 12 2016.

[21] Felix Mannhardt. Hospital Billing - Event Log. 8 2017.

[22] L. Ozdamar. A genetic algorithm approach to a general category project scheduling problem. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 29(1):44–59, 1999.

[23] L. B.S. Raccoon. A learning curve primer for software engineers. *SIGSOFT Softw. Eng. Notes*, 21(1):77–86, January 1996.

[24] Daniel D. Roman. The pert system: An appraisal of program evaluation review technique. *Academy of Management Journal*, 5(1):57–65, 1962.

[25] Avraham Shtub and Shlomo Globerson. *Project management: Processes, methodologies, and economics.* , 2006.

[26] Jorge Toro Valdivia. Herramienta ETL para logs de procesos. *Santiago, Chile: Universidad de Chile - Facultad de Ciencias Físicas y Matemáticas*, 2016.

[27] D. R. Towill and J. E. Cherrington. Learning curve models for predicting the performance of amt. *The International Journal of Advanced Manufacturing Technology*, 9(3):195–203, May 1994.

[28] Denis R. Towill. Forecasting learning curves. *International Journal of Forecasting*, 6(1):25 – 38, 1990.

[29] Wil M. P. van der Aalst. *Process Mining - Data Science in Action, Second Edition.* Springer, 2016.

[30] Boudewijn F van Dongen and Wil MP Van der Aalst. A meta model for process mining data. *EMOI-INTEROP*, 160:30, 2005.

[31] Tomás Vera, S. Ochoa, and D. Perovich. Survey of software development effort estimation taxonomies. 2018.

[32] Louis E. Yelle. The learning curve: Historical review and comprehensive survey. *Decision Sciences*, 10(2):302–328, 1979.

9. Apéndice

9.1. Extracto de instancias de archivos XES

Se agrega metadata de archivos utilizados en experimentos del algoritmo EWEHD.

9.1.1. Hospital Billing - Event Log

```
<metadata>
   <doi>doi:10.4121/uuid:76c46b83-c930-4798-a1c9-4be94dfeb741</doi>
   <name>Hospital Billing - Event Log.xes.gz</name>
   <description>The 'Hospital Billing' event log was obtained from the financial
       modules of the ERP system of a regional hospital. The event log contains
       events that are related to the billing of medical services that have been
       provided by the hospital. Each trace of the event log records the activities
        executed to bill a package of medical services that were bundled together.
       The event log does not contain information about the actual medical services
        provided by the hospital.  The 100,000 traces in the event log are a random
        sample of process instances that were recorded over three years. Several
       attributes such as the 'state' of the process, the 'caseType', the
       underlying 'diagnosis' etc. are included in the event log. Events and
       attribute values have been anonymized. The time stamps of events have been
       randomized for this purpose, but the time between events within a trace has
       not been altered.  More information about the event log can be found in the
       related publications.</description>
   <language>Englisch</language>
   <log_type>Real-life</log_type>
   <process_type>Implicitly structured</process_type>
   <creation>
   <insitute>Eindhoven University of Technology</insitute>
   <person>F. Mannhardt</person>
   <place>Eindhoven</place>
   <time>Mon Jul 31 00:00:00 CEST 2017</time>
   </creation>
   <source>
   <insitute>Eindhoven University of Technology, Department of Mathematics and
       Computer Science</insitute>
   <institute_type>Hospital</institute_type>
   <program>ERP System</program>
   </source>
   <rights type="Public">
   </rights>
   <number_of_traces>100000</number_of_traces>
   <number_of_events>451359</number_of_events>
   <events_per_trace>4.514</events_per_trace>
   <min_events_per_trace>1</min_events_per_trace>
   <max_events_per_trace>217</max_events_per_trace>
   <global_attributes>
      <trace_level>
         <attribute key="concept:name" default="DEFAULT"/>
      </trace_level>
      <event_level>
         <attribute key="concept:name" default="DEFAULT"/>
         <attribute key="lifecycle:transition" default="DEFAULT"/>
         <attribute key="time:timestamp" default="1970-01-01T01:00:00+01:00"/>
      </event_level>
   </global_attributes>
   <meta_extensions>
      <MetaData_Concept prefix="meta_concept" uri="http://www.xes-standard.org/
          meta_concept.xesext">
         <different_names_average>4.303</different_names_average>
```

```
39          <different_names_max>12</different_names_max>
40          <different_names_min>1</different_names_min>
41          <different_names_standard_deviation>2.165</
                different_names_standard_deviation>
42          <different_names_total>18</different_names_total>
43          <named_events_average>4.514</named_events_average>
44          <named_events_max>217</named_events_max>
45          <named_events_min>1</named_events_min>
46          <named_events_standard_deviation>2.782</named_events_standard_deviation>
47          <named_events_total>451359</named_events_total>
48       </MetaData_Concept>
49       <MetaData_General prefix="meta_general" uri="http://www.xes-standard.org/
             meta_general.xesext">
50          <classifiers>1</classifiers>
51          <events_average>4.514</events_average>
52          <events_max>217</events_max>
53          <events_min>1</events_min>
54          <events_standard_deviation>2.782</events_standard_deviation>
55          <events_total>451359</events_total>
56          <traces_total>100000</traces_total>
57       </MetaData_General>
58       <MetaData_LifeCycle prefix="meta_life" uri="http://www.xes-standard.org/
             meta_life.xesext">
59          <different_transitions_average>1.0</different_transitions_average>
60          <different_transitions_max>1</different_transitions_max>
61          <different_transitions_min>1</different_transitions_min>
62          <different_transitions_standard_deviation>0.0</
                different_transitions_standard_deviation>
63          <different_transitions_total>1</different_transitions_total>
64          <transition_events_average>4.514</transition_events_average>
65          <transition_events_max>217</transition_events_max>
66          <transition_events_min>1</transition_events_min>
67          <transition_events_standard_deviation>2.782</
                transition_events_standard_deviation>
68          <transition_events_total>451359</transition_events_total>
69       </MetaData_LifeCycle>
70       <MetaData_Time prefix="meta_time" uri="http://www.xes-standard.org/meta_time.
             xesext">
71          <duration_average>1.1004136257E7</duration_average>
72          <duration_max>8.9460392E7</duration_max>
73          <duration_min>0.0</duration_min>
74          <duration_standard_deviation>1.1307906308E7</duration_standard_deviation>
75          <duration_total>1.100413625651E12</duration_total>
76          <log_duration>9.7800338E7</log_duration>
77          <log_end_time>2016-01-19T08:58:56+01:00</log_end_time>
78          <log_start_time>2012-12-13T10:13:18+01:00</log_start_time>
79       </MetaData_Time>
80    </meta_extensions>
81    <extensions>
82       <Concept prefix="concept" uri="http://www.xes-standard.org/concept.xesext">
83          <name>Hospital Billing - Event Log</name>
84       </Concept>
85       <Lifecycle prefix="lifecycle" uri="http://www.xes-standard.org/lifecycle.
             xesext">
86       </Lifecycle>
87       <Time prefix="time" uri="http://www.xes-standard.org/time.xesext">
88       </Time>
89    </extensions>
90 </metadata>
```

9.1.2. WABO, CoSeLoG project

```
<metadata>
   <doi>doi:10.4121/uuid:c399c768-d995-4086-adda-c0bc72ad02bc</doi>
   <name>CoSeLoG WABO 5.xes.gz</name>
   <description>This data originates from the CoSeLoG project executed under NWO
       project number 638.001.211. Within the CoSeLoG project the (dis)similarities
        between several processes of different municipalities in the Netherlands
       has been investigated. This data is part of a collection of 5 event logs
       that record the execution of a building permit application process in five
       different anonymous municipalities. The recording of these processes is
       comparable which means that activity labels in the different event logs
       refer to the same activities performed in the five municipalities. The event
        log contains additional data on the case/trace and event level. In total
       there are 1! ! 6 trace level attributes which include the parts the permit
       consists of, the planned end date and the responsible actor. The event
       attributes contain the activity label, the timestamp of execution and the
       human resource involved. Furthermore, information regarding the planned date
        of execution and the due date is included. Additionally events may contain
       additional process data. Some attributes contain Dutch terms and phrases.</
       description>
   <language>Dutch</language>
   <log_type>Real-life</log_type>
   <process_type>Explicitly structured</process_type>
   <creation>
   <insitute>Eindhoven University of Technology</insitute>
   <person>Buijs, J.C.A.M.</person>
   <place>Eindhoven</place>
   <time>Fri May 23 00:00:00 CEST 2014</time>
   </creation>
   <source>
   <insitute>Buijs, J.C.A.M. (PhD candidate)</insitute>
   <institute_type>Municipality</institute_type>
   <model>WABO reference model</model>
   <program>Case handling system</program>
   </source>
   <rights type="Public">
   </rights>
   <lifecycle_model>standard</lifecycle_model>
   <number_of_traces>892</number_of_traces>
   <number_of_events>43896</number_of_events>
   <events_per_trace>49.211</events_per_trace>
   <min_events_per_trace>5</min_events_per_trace>
   <max_events_per_trace>100</max_events_per_trace>
   <global_attributes>
      <trace_level>
         <attribute key="IDofConceptCase" default="UNKNOWN"/>
         <attribute key="Includes_subCases" default="UNKNOWN"/>
         <attribute key="Responsible_actor" default="UNKNOWN"/>
         <attribute key="SUMleges" default="-1.0"/>
         <attribute key="caseProcedure" default="UNKNOWN"/>
         <attribute key="caseStatus" default="UNKNOWN"/>
         <attribute key="case_type" default="UNKNOWN"/>
         <attribute key="concept:name" default="UNKNOWN"/>
         <attribute key="endDate" default="1970-01-01T00:00:00+01:00"/>
         <attribute key="endDatePlanned" default="1970-01-01T00:00:00+01:00"/>
         <attribute key="landRegisterID" default="UNKNOWN"/>
         <attribute key="last_phase" default="UNKNOWN"/>
         <attribute key="parts" default="UNKNOWN"/>
         <attribute key="requestComplete" default="UNKNOWN"/>
         <attribute key="startDate" default="1970-01-01T00:00:00+01:00"/>
         <attribute key="termName" default="UNKNOWN"/>
      </trace_level>
      <event_level>
```

```
48          <attribute key="action_code" default="UNKNOWN"/>
49          <attribute key="concept:name" default="UNKNOWN"/>
50          <attribute key="dateFinished" default="UNKNOWN"/>
51          <attribute key="dateStop" default="UNKNOWN"/>
52          <attribute key="dueDate" default="1970-01-01T00:00:00+01:00"/>
53          <attribute key="lifecycle:transition" default="UNKNOWN"/>
54          <attribute key="monitoringResource" default="UNKNOWN"/>
55          <attribute key="org:resource" default="UNKNOWN"/>
56          <attribute key="planned" default="1970-01-01T00:00:00+01:00"/>
57          <attribute key="question" default="UNKNOWN"/>
58          <attribute key="taskForMainCase" default="true"/>
59          <attribute key="taskName" default="UNKNOWN"/>
60          <attribute key="time:timestamp" default="1970-01-01T00:00:00+01:00"/>
61       </event_level>
62    </global_attributes>
63    <meta_extensions>
64       <MetaData_Concept prefix="meta_concept" uri="http://www.xes-standard.org/
             meta_concept.xesext">
65          <different_names_average>47.967</different_names_average>
66          <different_names_max>99</different_names_max>
67          <different_names_min>1</different_names_min>
68          <different_names_standard_deviation>14.495</
                different_names_standard_deviation>
69          <different_names_total>350</different_names_total>
70          <named_events_average>49.211</named_events_average>
71          <named_events_max>100</named_events_max>
72          <named_events_min>5</named_events_min>
73          <named_events_standard_deviation>14.955</named_events_standard_deviation>
74          <named_events_total>43896</named_events_total>
75       </MetaData_Concept>
76       <MetaData_General prefix="meta_general" uri="http://www.xes-standard.org/
             meta_general.xesext">
77          <classifiers>2</classifiers>
78          <events_average>49.211</events_average>
79          <events_max>100</events_max>
80          <events_min>5</events_min>
81          <events_standard_deviation>14.955</events_standard_deviation>
82          <events_total>43896</events_total>
83          <traces_total>892</traces_total>
84       </MetaData_General>
85       <MetaData_LifeCycle prefix="meta_life" uri="http://www.xes-standard.org/
             meta_life.xesext">
86          <different_transitions_average>1.0</different_transitions_average>
87          <different_transitions_max>1</different_transitions_max>
88          <different_transitions_min>1</different_transitions_min>
89          <different_transitions_standard_deviation>0.0</
                different_transitions_standard_deviation>
90          <different_transitions_total>1</different_transitions_total>
91          <transition_events_average>49.211</transition_events_average>
92          <transition_events_max>100</transition_events_max>
93          <transition_events_min>5</transition_events_min>
94          <transition_events_standard_deviation>14.955</
                transition_events_standard_deviation>
95          <transition_events_total>43896</transition_events_total>
96       </MetaData_LifeCycle>
97       <MetaData_Organization prefix="meta_org" uri="http://www.xes-standard.org/
             meta_org.xesext">
98          <different_groups_average>1.0</different_groups_average>
99          <different_groups_max>1</different_groups_max>
100         <different_groups_min>1</different_groups_min>
101         <different_groups_standard_deviation>0.0</
                different_groups_standard_deviation>
102         <different_groups_total>1</different_groups_total>
103         <different_resources_average>2.811</different_resources_average>
```

```
104          <different_resources_max>6</different_resources_max>
105          <different_resources_min>1</different_resources_min>
106          <different_resources_standard_deviation>0.769</
                 different_resources_standard_deviation>
107          <different_resources_total>15</different_resources_total>
108          <different_roles_average>1.0</different_roles_average>
109          <different_roles_max>1</different_roles_max>
110          <different_roles_min>1</different_roles_min>
111          <different_roles_standard_deviation>0.0</
                 different_roles_standard_deviation>
112          <different_roles_total>1</different_roles_total>
113          <group_events_average>49.211</group_events_average>
114          <group_events_max>100</group_events_max>
115          <group_events_min>5</group_events_min>
116          <group_events_standard_deviation>14.955</group_events_standard_deviation>
117          <group_events_total>43896</group_events_total>
118          <resource_events_average>49.211</resource_events_average>
119          <resource_events_max>100</resource_events_max>
120          <resource_events_min>5</resource_events_min>
121          <resource_events_standard_deviation>14.955</
                 resource_events_standard_deviation>
122          <resource_events_total>43896</resource_events_total>
123          <role_events_average>49.211</role_events_average>
124          <role_events_max>100</role_events_max>
125          <role_events_min>5</role_events_min>
126          <role_events_standard_deviation>14.955</role_events_standard_deviation>
127          <role_events_total>43896</role_events_total>
128       </MetaData_Organization>
129       <MetaData_Time prefix="meta_time" uri="http://www.xes-standard.org/meta_time.
              xesext">
130          <duration_average>7769189.772</duration_average>
131          <duration_max>6.7357006E7</duration_max>
132          <duration_min>40370.0</duration_min>
133          <duration_standard_deviation>6626247.226</duration_standard_deviation>
134          <duration_total>6.930117277E9</duration_total>
135          <log_duration>1.30032E8</log_duration>
136          <log_end_time>2014-01-06T00:00:00+01:00</log_end_time>
137          <log_start_time>2009-11-23T00:00:00+01:00</log_start_time>
138       </MetaData_Time>
139    </meta_extensions>
140    <extensions>
141       <Concept prefix="concept" uri="http://www.xes-standard.org/concept.xesext">
142          <name>CoSeLoG WABO 5</name>
143       </Concept>
144       <Lifecycle prefix="lifecycle" uri="http://www.xes-standard.org/lifecycle.
              xesext">
145          <model>standard</model>
146       </Lifecycle>
147       <Organizational prefix="org" uri="http://www.xes-standard.org/org.xesext">
148       </Organizational>
149       <Semantic prefix="semantic" uri="http://www.xes-standard.org/semantic.xesext"
              >
150       </Semantic>
151       <Time prefix="time" uri="http://www.xes-standard.org/time.xesext">
152       </Time>
153    </extensions>
154 </metadata>
```

9.1.3. Sepsis Cases - Event Log

```
<metadata>
   <doi>doi:10.4121/uuid:915d2bfb-7e84-49ad-a286-dc35f063a460</doi>
   <name>Sepsis Cases - Event Log.xes.gz</name>
   <description>This real-life event log contains events of sepsis cases from a
       hospital. Sepsis is a life threatening condition typically caused by an
       infection. One case represents the pathway through the hospital. The events
        were recorded by the ERP system of the hospital. There are about 1000 cases
        with in total 15,000 events that were recorded for 16 different activities.
        Moreover, 39 data attributes are recorded, e.g., the group responsible for
       the activity, the results of tests and information from checklists. Events
       and attribute values have been anonymized. The time stamps of events have
       been randomized, but the time between events within a trace has not been
       altered.</description>
   <language>English</language>
   <log_type>Real-life</log_type>
   <process_type>Unknown</process_type>
   <creation>
   <insitute>Eindhoven University of Technology</insitute>
   <person>Mannhardt, Felix</person>
   <place>Eindhoven</place>
   <time>Thu Dec 01 14:58:44 CET 2016</time>
   </creation>
   <source>
   <insitute>Eindhoven University of Technology</insitute>
   <institute_type>Hospital</institute_type>
   <program>ERP System</program>
   </source>
   <rights type="Public">
   </rights>
   <number_of_traces>1050</number_of_traces>
   <number_of_events>15214</number_of_events>
   <events_per_trace>14.49</events_per_trace>
   <min_events_per_trace>3</min_events_per_trace>
   <max_events_per_trace>185</max_events_per_trace>
   <global_attributes>
      <trace_level>
         <attribute key="concept:name" default="DEFAULT"/>
      </trace_level>
      <event_level>
         <attribute key="concept:name" default="DEFAULT"/>
         <attribute key="lifecycle:transition" default="DEFAULT"/>
         <attribute key="org:group" default="DEFAULT"/>
         <attribute key="time:timestamp" default="1970-01-01T01:00:00+01:00"/>
      </event_level>
   </global_attributes>
   <meta_extensions>
      <MetaData_Concept prefix="meta_concept" uri="http://www.xes-standard.org/
          meta_concept.xesext">
         <different_names_average>9.133</different_names_average>
         <different_names_max>12</different_names_max>
         <different_names_min>3</different_names_min>
         <different_names_standard_deviation>2.106</
             different_names_standard_deviation>
         <different_names_total>16</different_names_total>
         <named_events_average>14.49</named_events_average>
         <named_events_max>185</named_events_max>
         <named_events_min>3</named_events_min>
         <named_events_standard_deviation>11.476</named_events_standard_deviation>
         <named_events_total>15214</named_events_total>
      </MetaData_Concept>
      <MetaData_General prefix="meta_general" uri="http://www.xes-standard.org/
          meta_general.xesext">
```

```
51          <classifiers>2</classifiers>
52          <events_average>14.49</events_average>
53          <events_max>185</events_max>
54          <events_min>3</events_min>
55          <events_standard_deviation>11.476</events_standard_deviation>
56          <events_total>15214</events_total>
57          <traces_total>1050</traces_total>
58       </MetaData_General>
59       <MetaData_LifeCycle prefix="meta_life" uri="http://www.xes-standard.org/
             meta_life.xesext">
60          <different_transitions_average>1.0</different_transitions_average>
61          <different_transitions_max>1</different_transitions_max>
62          <different_transitions_min>1</different_transitions_min>
63          <different_transitions_standard_deviation>0.0</
                different_transitions_standard_deviation>
64          <different_transitions_total>1</different_transitions_total>
65          <transition_events_average>14.49</transition_events_average>
66          <transition_events_max>185</transition_events_max>
67          <transition_events_min>3</transition_events_min>
68          <transition_events_standard_deviation>11.476</
                transition_events_standard_deviation>
69          <transition_events_total>15214</transition_events_total>
70       </MetaData_LifeCycle>
71       <MetaData_Organization prefix="meta_org" uri="http://www.xes-standard.org/
             meta_org.xesext">
72          <different_groups_average>5.169</different_groups_average>
73          <different_groups_max>9</different_groups_max>
74          <different_groups_min>2</different_groups_min>
75          <different_groups_standard_deviation>1.5</
                different_groups_standard_deviation>
76          <different_groups_total>26</different_groups_total>
77          <different_resources_average>1.0</different_resources_average>
78          <different_resources_max>1</different_resources_max>
79          <different_resources_min>1</different_resources_min>
80          <different_resources_standard_deviation>0.0</
                different_resources_standard_deviation>
81          <different_resources_total>1</different_resources_total>
82          <different_roles_average>1.0</different_roles_average>
83          <different_roles_max>1</different_roles_max>
84          <different_roles_min>1</different_roles_min>
85          <different_roles_standard_deviation>0.0</
                different_roles_standard_deviation>
86          <different_roles_total>1</different_roles_total>
87          <group_events_average>14.49</group_events_average>
88          <group_events_max>185</group_events_max>
89          <group_events_min>3</group_events_min>
90          <group_events_standard_deviation>11.476</group_events_standard_deviation>
91          <group_events_total>15214</group_events_total>
92          <resource_events_average>14.49</resource_events_average>
93          <resource_events_max>185</resource_events_max>
94          <resource_events_min>3</resource_events_min>
95          <resource_events_standard_deviation>11.476</
                resource_events_standard_deviation>
96          <resource_events_total>15214</resource_events_total>
97          <role_events_average>14.49</role_events_average>
98          <role_events_max>185</role_events_max>
99          <role_events_min>3</role_events_min>
100         <role_events_standard_deviation>11.476</role_events_standard_deviation>
101         <role_events_total>15214</role_events_total>
102      </MetaData_Organization>
103      <MetaData_Time prefix="meta_time" uri="http://www.xes-standard.org/meta_time.
             xesext">
104         <duration_average>2459733.94</duration_average>
105         <duration_max>3.6488789E7</duration_max>
```

```
106           <duration_min>122.0</duration_min>
107           <duration_standard_deviation>5230403.112</duration_standard_deviation>
108           <duration_total>2.582720637E9</duration_total>
109           <log_duration>4.9691202E7</log_duration>
110           <log_end_time>2015-06-05T12:25:11+02:00</log_end_time>
111           <log_start_time>2013-11-07T08:18:29+01:00</log_start_time>
112        </MetaData_Time>
113     </meta_extensions>
114     <extensions>
115        <Concept prefix="concept" uri="http://www.xes-standard.org/concept.xesext">
116           <name>Sepsis Cases - Event Log</name>
117        </Concept>
118        <Lifecycle prefix="lifecycle" uri="http://www.xes-standard.org/lifecycle.
               xesext">
119        </Lifecycle>
120        <Organizational prefix="org" uri="http://www.xes-standard.org/org.xesext">
121        </Organizational>
122        <Time prefix="time" uri="http://www.xes-standard.org/time.xesext">
123        </Time>
124     </extensions>
125  </metadata>
```

9.2. Implementación algoritmo EWEHD

Se agregan implementaciones de algoritmos internos utilizados en el algoritmo EWEHD.

Algoritmo 4 Desviación estándar

```
1: procedure SD(d)                                  ▷ Calcular la desviación estandar
2:     r ← dict()
3:     for e ← d do
4:         sum ← 0
5:         sd ← 0
6:         times ← d(e)
7:         for time ← times do
8:             sum ← sum + time
9:         end for
10:        mean ← sum/len(times)
11:        for time ← times do
12:            sd ← sd + POW(time − mean, 2)
13:        end for
14:        r(e) ← SQRT(sd/len(times))
15:    end for
16:    return r
17: end procedure
```

Algoritmo 5 Mediana

```
1: procedure MED(d)                                 ▷ Calcular la mediana
2:     r ← dict()
3:     for e ← d do
4:         times ← d(e)
5:         size ← len(times)
6:         if MOD(size, 2) == 0 then
7:             r(e) ← (times(size/2) + times(size/2 − 1))/2
8:         else
9:             r(e) ← times(size/2)
10:        end if
11:    end for
12:    return r
13: end procedure
```

Algoritmo 6 Promedio

```
1: procedure AVG(d)                                   ▷ Calcular promedio
2:     sum ← 0
3:     for i ← d do
4:         sum ← sum + i
5:     end for
6:     return sum/size(d)
7: end procedure
```

9.3. Implementación Plugin EWEHD para ProM

```
126 package org.processmining.plugins.ewehd;
127
128 import org.deckfour.xes.model.*;
129 import org.processmining.contexts.uitopia.UIPluginContext;
130 import org.processmining.contexts.uitopia.annotations.UITopiaVariant;
131 import org.processmining.framework.plugin.annotations.Plugin;
132 import org.processmining.framework.plugin.annotations.PluginVariant;
133 import org.processmining.models.ewehd.Configuration;
134 import org.processmining.models.ewehd.EWEHD;
135 import org.processmining.models.ewehd.tools.EmptyIcon;
136
137 import javax.swing.*;
138
139 import static javax.swing.JOptionPane.*;
140
141 @Plugin(name = "EWEHDPlugin", parameterLabels = {"Log"}, returnLabels = {"EWEHD
        Object"}, returnTypes = {EWEHD.class})
142 public class EWEHDPlugin {
143
144     @UITopiaVariant(affiliation = "University of Chile", author = "Pedro
            Bustamante", email = "ppbb15@gmail.com", uiLabel = UITopiaVariant.
            USEPLUGIN)
145     @PluginVariant(requiredParameterLabels = {0})
146     public EWEHD create(UIPluginContext context, XLog xlog) {
147
148         int groupMode = JOptionPane.showOptionDialog(
149             null,
150             "Select group mode for the EWEHD algorithm",
151             "Group mode",
152             DEFAULT_OPTION,
153             QUESTION_MESSAGE,
154             new EmptyIcon(),
155             new Object[]{"GENERAL", "RESOURCE"},
156             "STRICT"
157         );
158
159         if (groupMode == CLOSED_OPTION) {
160             groupMode = 0;
161         }
162
163         int processMode = JOptionPane.showOptionDialog(
164                 null,
165                 "Select process mode for the EWEHD algorithm",
166                 "Process mode",
167                 DEFAULT_OPTION,
168                 QUESTION_MESSAGE,
169                 new EmptyIcon(),
170                 new Object[]{"LINEAL", "PARALLEL"},
```

```
171                     "LINEAL"
172             );
173
174             if (processMode == CLOSED_OPTION) {
175                 processMode = 0;
176             }
177
178             int durationMode = JOptionPane.showOptionDialog(
179                     null,
180                     "Select duration mode for the EWEHD algorithm",
181                     "Duration mode",
182                     DEFAULT_OPTION,
183                     QUESTION_MESSAGE,
184                     new EmptyIcon(),
185                     new Object[]{"STRICT", "ESTIMATED", "FLEXIBLE"},
186                     "STRICT"
187             );
188
189             if (durationMode == CLOSED_OPTION) {
190                 durationMode = 0;
191             }
192
193             return procreate(xlog, groupMode, processMode,  durationMode);
194         }
195
196         private EWEHD procreate(XLog xlog, int groupMode, int processMode, int
                durationMode) {
197             EWEHD ewehd = new EWEHD(xlog, new Configuration(groupMode, processMode,
                     durationMode));
198             return ewehd.process();
199         }
200     }
```

Listado 1: Implementación plugin EWEHD

```
201 package org.processmining.plugins.ewehd;
202
203 import com.fluxicon.slickerbox.components.IconVerticalTabbedPane;
204 import org.processmining.contexts.uitopia.annotations.Visualizer;
205 import org.processmining.framework.plugin.PluginContext;
206 import org.processmining.framework.plugin.annotations.Plugin;
207 import org.processmining.framework.plugin.annotations.PluginVariant;
208 import org.processmining.models.ewehd.EWEHD;
209 import org.processmining.models.ewehd.tools.EWEHDUI;
210 import org.processmining.plugins.interactivevisualization.
        InteractivityContext;
211
212 import javax.swing.*;
213 import java.awt.*;
214
215 @Plugin(name = "EWEHDVisualization", parameterLabels = {"EWEHD"},
        returnLabels = {"EWEHD Viewer"}, returnTypes = {JComponent.class},
        userAccessible = true)
216 @Visualizer
217 public class EWEHDVisualization extends JPanel {
218     private static final long serialVersionUID = 4258730373866637365L;
219
220     private final static String SUMMARYICON = "summary48-2.png";
221     private Image summaryIcon = new ImageIcon(EWEHDVisualization.class.
            getResource(SUMMARYICON), SUMMARYICON).getImage();
222     private EWEHD log;
223     private InteractivityContext interactivityContext = new
            InteractivityContext();
224
225     @PluginVariant(requiredParameterLabels = {0})
226     public JComponent visualize(PluginContext context, EWEHD log) {
227         this.log = log;
228         initialize();
229         completeGui(context);
230         return this;
231     }
232
233     private void initialize() {
234         setBackground(new Color(40, 40, 40));
235         setBorder(BorderFactory.createEmptyBorder());
236         setLayout(new BorderLayout());
237     }
238
239     private void completeGui(PluginContext context) {
240         EWEHDUI ewehdui = new EWEHDUI(this, context);
241         IconVerticalTabbedPane iconTabs = new IconVerticalTabbedPane(new
                Color(230, 230, 230, 210), new Color(20, 20,
242                     20, 160));
243         iconTabs.addTab("Summary", summaryIcon, ewehdui);
244         add(iconTabs, BorderLayout.CENTER);
245         revalidate();
246         repaint();
247     }
248
249     public EWEHD getLog() {
250         return log;
251     }
252 }
```

Listado 2: Implementación visualizador EWEHD

9.4. Flujo Plugin EWEHD para ProM

Para comprender la utilización del plugin desarrollado para ProM, y hacer uso del algoritmo EWEHD, a continuación de presentará el flujo del proceso.

Una vez que el programa ProM es abierto y haya finalizado su inicialización, se debe seleccionar el archivo XES a procesar utilizando el gestor de archivos del sistema, como se observa en la figura 19.

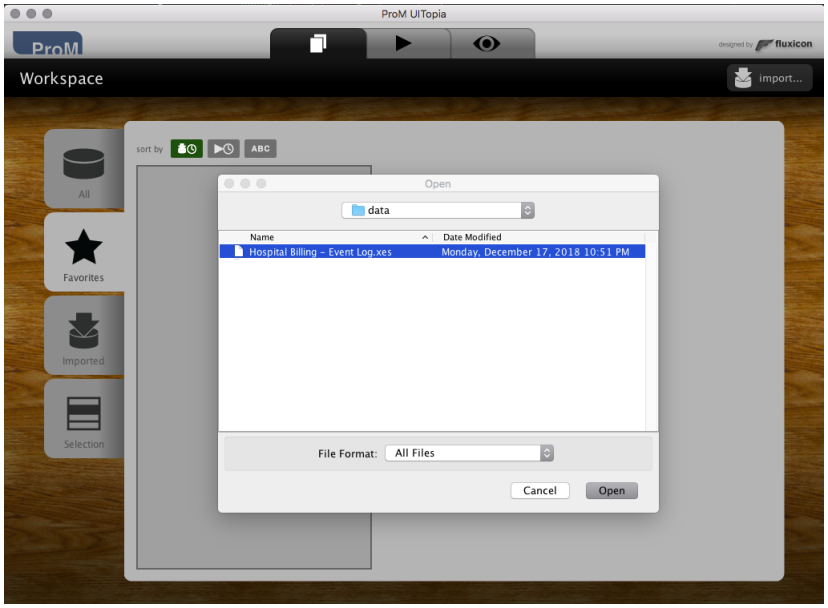


Figura 19: Búsqueda y selección del archivo XES.

Luego que el archivo XES es seleccionado, ProM iniciará el proceso de importación (ver figura 20), mostrando su avance a través de una pantalla de carga con una barra de avance.

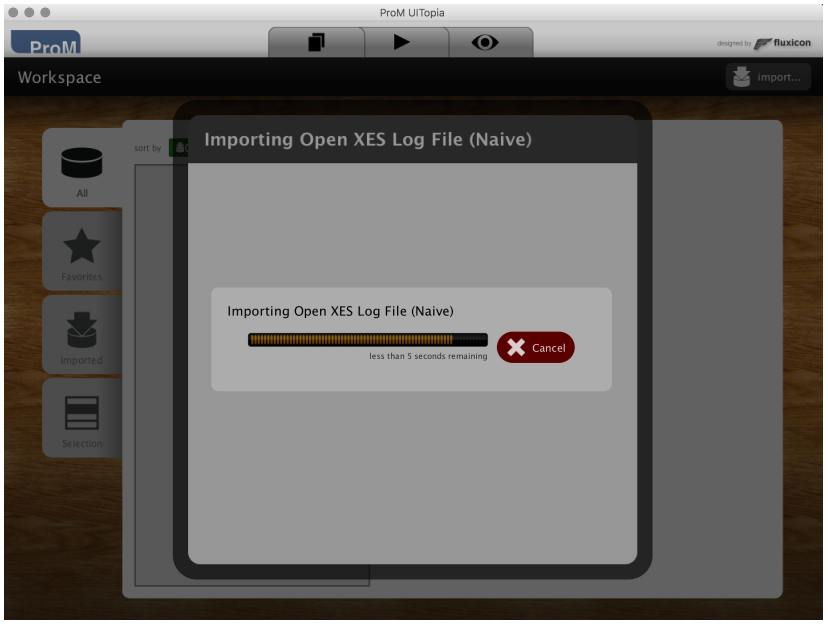


Figura 20: Importación del archivo XES.

Una vez finalizada la importación, la siguiente etapa será la selección del plugin a aplicar sobre el archivo XES. ProM automáticamente mostrará un listado de plugins que sean compatibles con el procesamiento de archivos XES, y para hacer la búsqueda más rápida, provee una barra de búsqueda, en la cual, al ingresar la palabra EWEHD, desplegará el plugin desarrollado (ver figura 21).

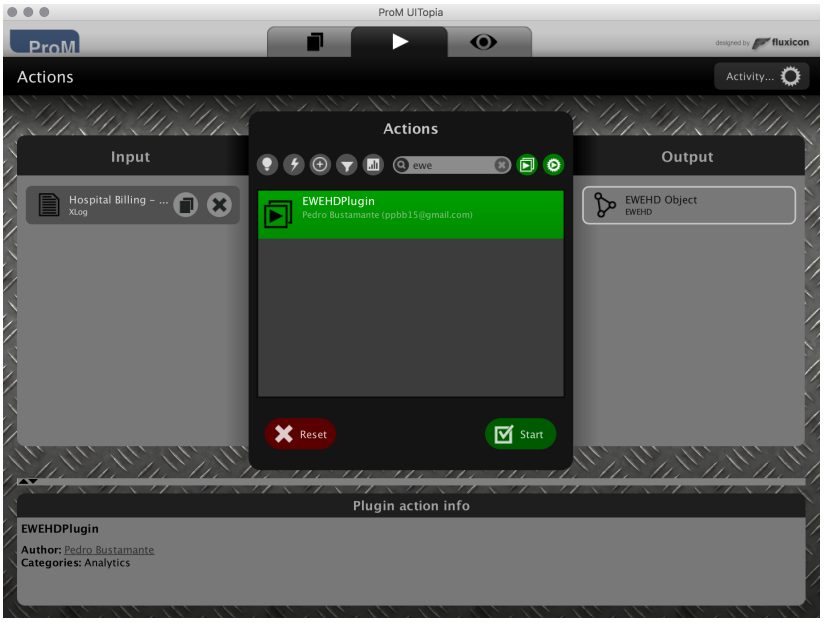


Figura 21: Selección del plugin EWEHD.

Luego que el plugin inicialice, se procederá a pedir los parámetros para su funcionamiento. Primero el nombre del atributo para extraer los recursos (ver figura

22), luego seleccionar el modo de ejecución de las tareas dentro del los procesos (ver figura 23) y finalmente, el modo en el que se calculacurán las estimaciones de la duración de las tareas (ver figura 24).

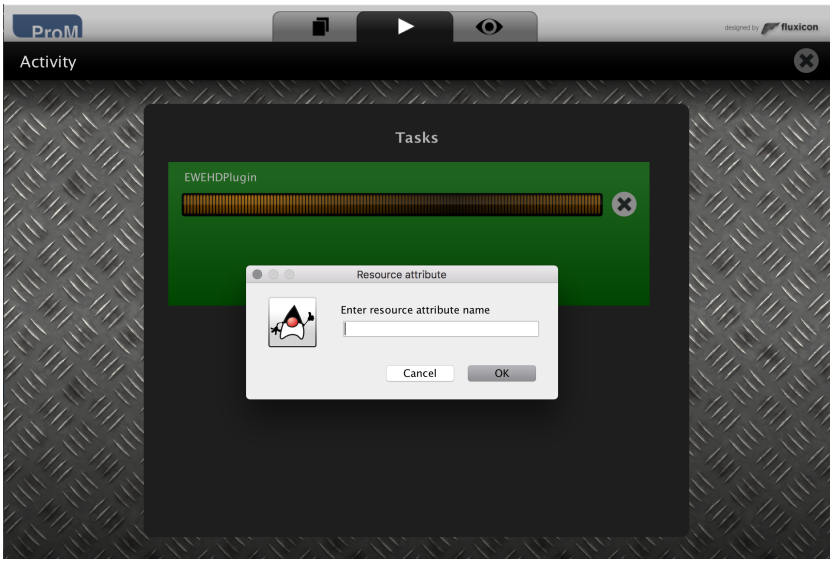


Figura 22: Ingreso nombre de atributo de recursos

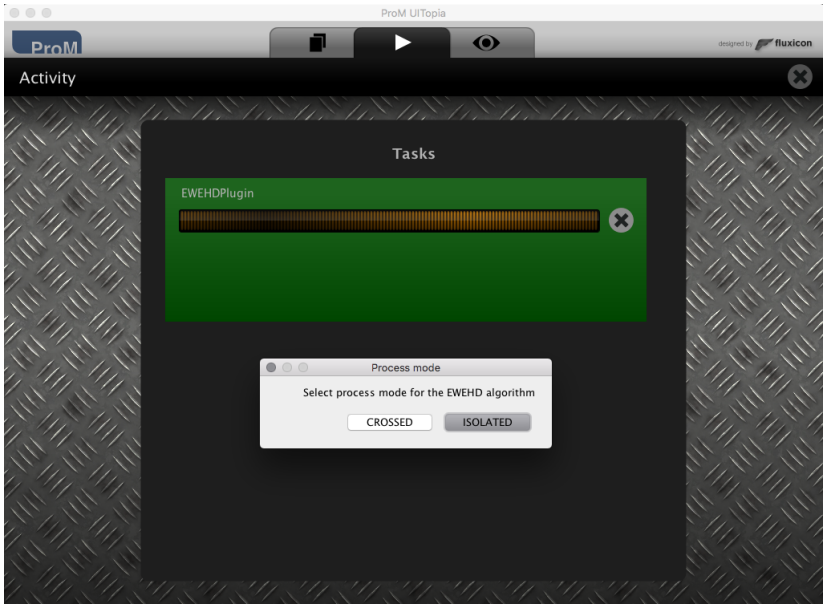


Figura 23: Selección del modo de ejecución de procesos.

Al seleccionar todos los parámetros, el algoritmo iniciará su ejecución, mostrando su progreso a través de una barra de carga (ver figura 24). Finalizando, ProM desplegará una pantalla se selección de visualizadores, donde el usuario tendrá que elegir el visualizador con el cual quiere procesar el resultado del algoritmo, para luego desplegarlo en pantalla (ver figura 25).

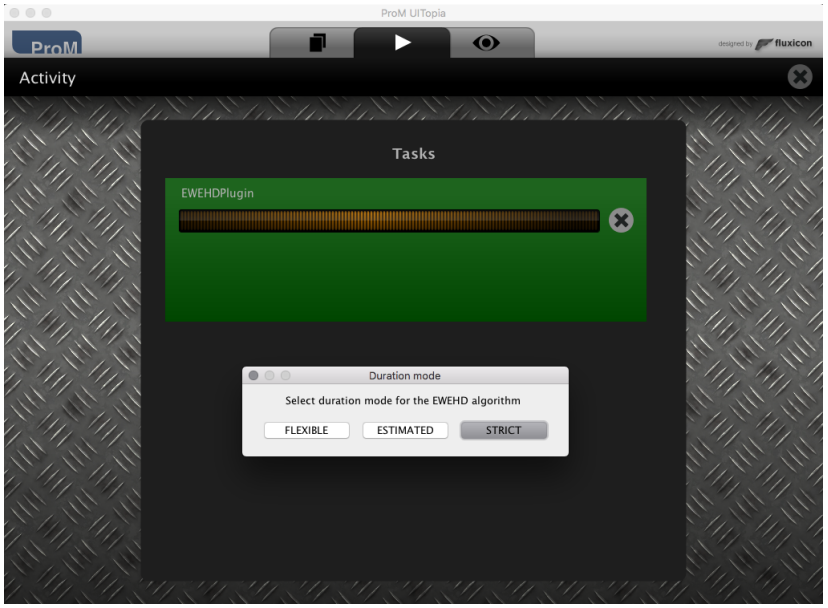


Figura 24: Selección del modo para el cálculo de duraciones

Figura 25: Seleccionar visualizador

Una vez seleccionado el visualizador *EWEHDVisualization*, ProM mostrará los resultados de la ejecucíón del algoritmo; resumen general del archivo XES (ver figura 26 y 27), estimaciones de las duraciones de cada tarea (ver figura 28) y finalmente, los recursos utilizados en cada tarea (29).

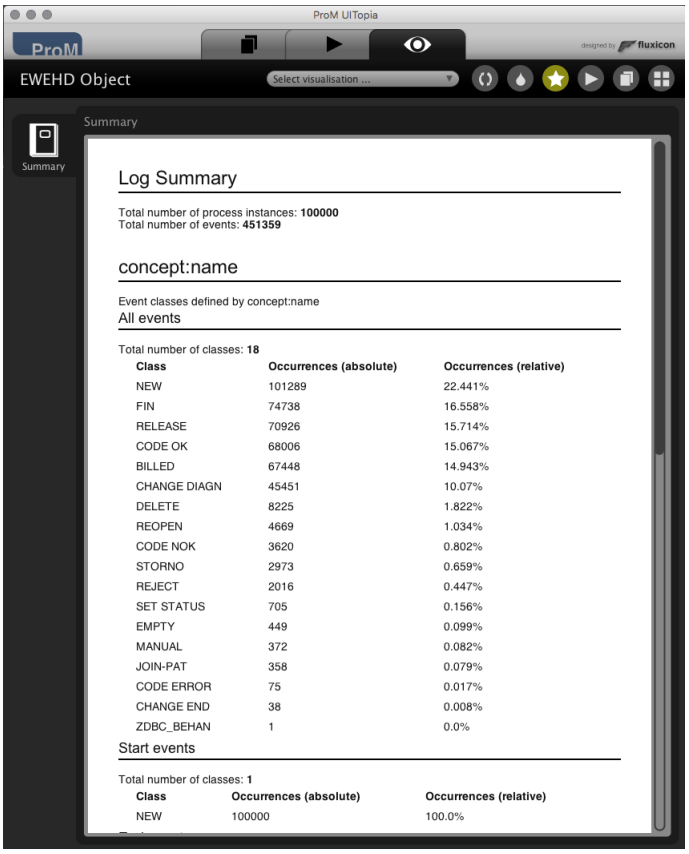


Figura 26: Resultados - Parte 1

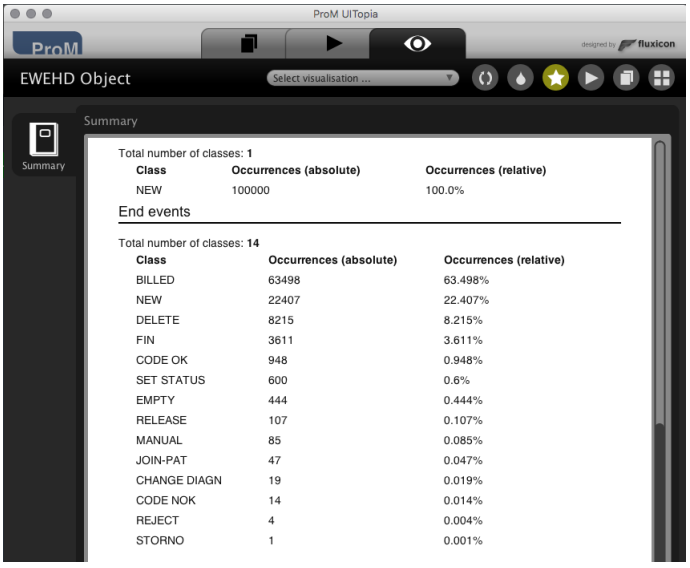
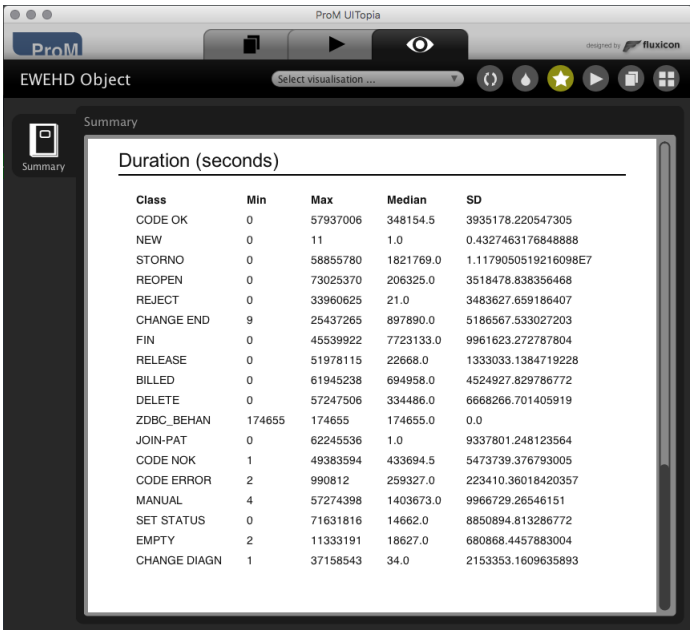
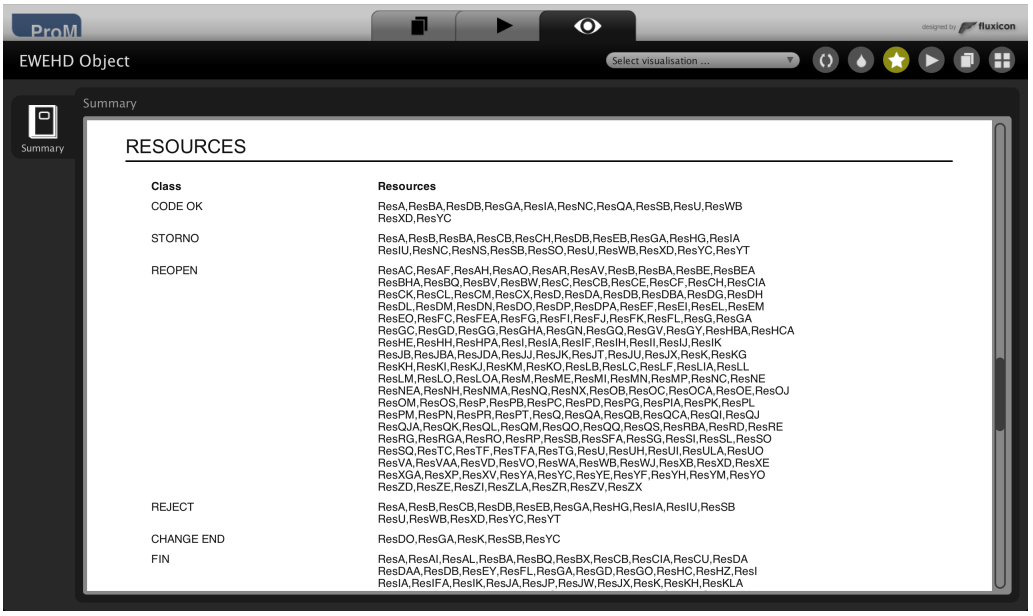


Figura 27: Resultados - Parte 2



Class	Min	Max	Median	SD
CODE OK	0	57937006	348154.5	3935178.220547305
NEW	0	11	1.0	0.4327463176848888
STORNO	0	58855780	1821769.0	1.1179050519216098E7
REOPEN	0	73025370	206325.0	3518478.838356468
REJECT	0	33960625	21.0	3483627.659186407
CHANGE END	9	25437265	897890.0	5186567.533027203
FIN	0	45539922	7723133.0	9961623.272787804
RELEASE	0	51978115	22668.0	1333033.1384719228
BILLED	0	61945238	694958.0	4524927.829786772
DELETE	0	57247506	334486.0	6668266.701405919
ZDBC_BEHAN	174655	174655	174655.0	0.0
JOIN-PAT	0	62245536	1.0	9337801.248123564
CODE NOK	1	49383594	433694.5	5473739.376793005
CODE ERROR	2	990812	259327.0	223410.36018420357
MANUAL	4	57274398	1403673.0	9966729.26546151
SET STATUS	0	71631816	14662.0	8850894.813286772
EMPTY	2	11333191	18627.0	680868.4457883004
CHANGE DIAGN	1	37158543	34.0	2153353.1609635893

Figura 28: Resultados - Parte 3



Class	Resources
CODE OK	ResA,ResBA,ResDB,ResGA,ResIA,ResNC,ResQA,ResSB,ResU,ResWB ResXD,ResYC
STORNO	ResA,ResB,ResBA,ResCB,ResCH,ResDB,ResEB,ResGA,ResHG,ResIA ResIU,ResNC,ResNS,ResSB,ResSO,ResU,ResWB,ResXD,ResYC,ResYT
REOPEN	ResAC,ResAF,ResAH,ResAO,ResAR,ResAV,ResB,ResBA,ResBE,ResBEA ResBHA,ResBQ,ResBV,ResBW,ResC,ResCB,ResCE,ResCF,ResCH,ResCIA ResCK,ResCL,ResCM,ResCX,ResD,ResDA,ResDB,ResDBA,ResDG,ResDH ResDL,ResDM,ResDN,ResDO,ResDP,ResDPA,ResEF,ResEI,ResEL,ResEM ResEO,ResFC,ResFEA,ResFG,ResFI,ResFJ,ResFK,ResFL,ResG,ResGA ResGC,ResGD,ResGG,ResGHA,ResGN,ResGQ,ResGV,ResGY,ResHBA,ResHCA ResHE,ResHH,ResHPA,ResI,ResIA,ResIF,ResIH,ResII,ResIJ,ResIK ResJB,ResJBA,ResJDA,ResJJ,ResJK,ResJT,ResJU,ResJX,ResK,ResKG ResKH,ResKI,ResKJ,ResKM,ResKO,ResLB,ResLC,ResLF,ResLIA,ResLL ResLM,ResLO,ResLOA,ResM,ResME,ResMI,ResMN,ResMP,ResNC,ResNE ResNEA,ResNH,ResNMA,ResNQ,ResNX,ResOB,ResOC,ResOCA,ResOE,ResOJ ResOM,ResOS,ResP,ResPB,ResPC,ResPD,ResPG,ResPIA,ResPK,ResPL ResPM,ResPN,ResPR,ResPT,ResQ,ResQA,ResQB,ResQCA,ResQI,ResQJ ResQJA,ResQK,ResQL,ResQM,ResQO,ResQQ,ResQS,ResRBA,ResRD,ResRE ResRG,ResRGA,ResRO,ResRP,ResSB,ResSFA,ResSG,ResSI,ResSL,ResSO ResSQ,ResTC,ResTF,ResTFA,ResTG,ResU,ResUH,ResUI,ResULA,ResUO ResVA,ResVAA,ResVD,ResVO,ResWA,ResWB,ResWJ,ResXB,ResXD,ResXE ResXGA,ResXP,ResXV,ResYA,ResYC,ResYE,ResYF,ResYH,ResYM,ResYO ResZD,ResZE,ResZI,ResZLA,ResZR,ResZV,ResZX
REJECT	ResA,ResB,ResCB,ResDB,ResEB,ResGA,ResHG,ResIA,ResIU,ResSB ResU,ResWB,ResXD,ResYC,ResYT
CHANGE END	ResDO,ResGA,ResK,ResSB,ResYC
FIN	ResA,ResAI,ResAL,ResBA,ResBQ,ResBX,ResCB,ResCIA,ResCU,ResDA ResDAA,ResDB,ResEY,ResFL,ResGA,ResGD,ResGO,ResHC,ResHZ,ResI ResIA,ResIFA,ResIK,ResJA,ResJP,ResJW,ResJX,ResK,ResKH,ResKLA

Figura 29: Resultados - Parte 4

9.5. Recursos de instancias XES

Archivo XES	Modo de proceso	Modo estimación	Duración (seg)
Hospital Billing	ISOLATED	STRICT	30

Eventos	
Nombre/Tipo	**Recursos**
CODE OK	[ResA, ResBA, ResDB, ResGA, ResIA, ResNC, ResQA, ResSB, ResU, ResWB, ResXD, ResYC]
STORNO	[ResA, ResB, ResBA, ResCB, ResCH, ResDB, ResEB, ResGA, ResHG, ResIA, ResIU, ResNC, ResNS, ResSB, ResSO, ResU, ResWB, ResXD, ResYC, ResYT]
REOPEN	[ResAC, ResAF, ResAH, ResAO, ResAR, ResAV, ResB, ResBA, ResBE, ResBEA, ResBHA, ResBQ, ResBV, ResBW, ResC, ResCB, ResCE, ResCF, ResCH, ResCIA, ResCK, ResCL, ResCM, ResCX, ResD, ResDA, ResDB, ResDBA, ResDG, ResDH, ResDL, ResDM, ResDN, ResDO, ResDP, ResDPA, ResEF, ResEI, ResEL, ResEM, ResEO, ResFC, ResFEA, ResFG, ResFI, ResFJ, ResFK, ResFL, ResG, ResGA, ResGC, ResGD, ResGG, ResGHA, ResGN, ResGQ, ResGV, ResGY, ResHBA, ResHCA, ResHE, ResHH, ResHPA, ResI, ResIA, ResIF, ResIH, ResII, ResIJ, ResIK, ResJB, ResJBA, ResJDA, ResJJ, ResJK, ResJT, ResJU, ResJX, ResK, ResKG, ResKH, ResKI, ResKJ, ResKM, ResKO, ResLB, ResLC, ResLF, ResLIA, ResLL, ResLM, ResLO, ResLOA, ResM, ResME, ResMI, ResMN, ResMP, ResNC, ResNE, ResNEA, ResNH, ResNMA, ResNQ, ResNX, ResOB, ResOC, ResOCA, ResOE, ResOJ, ResOM, ResOS, ResP, ResPB, ResPC, ResPD, ResPG, ResPIA, ResPK, ResPL, ResPM, ResPN, ResPR, ResPT, ResQ, ResQA, ResQB, ResQCA, ResQI, ResQJ, ResQJA, ResQK, ResQL, ResQM, ResQO, ResQQ, ResQS, ResRBA, ResRD, ResRE, ResRG, ResRGA, ResRO, ResRP, ResSB, ResSFA, ResSG, ResSI, ResSL, ResSO, ResSQ, ResTC, ResTF, ResTFA, ResTG, ResU, ResUH, ResUI, ResULA, ResUO, ResVA, ResVAA, ResVD, ResVO, ResWA, ResWB, ResWJ, ResXB, ResXD, ResXE, ResXGA, ResXP, ResXV, ResYA, ResYC, ResYE, ResYF, ResYH, ResYM, ResYO, ResZD, ResZE, ResZI, ResZLA, ResZR, ResZV, ResZX]
REJECT	[ResA, ResB, ResCB, ResDB, ResEB, ResGA, ResHG, ResIA, ResIU, ResSB, ResU, ResWB, ResXD, ResYC, ResYT]
CHANGE END	[ResDO, ResGA, ResK, ResSB, ResYC]
FIN	[ResA, ResAI, ResAL, ResBA, ResBQ, ResBX, ResCB, ResCIA, ResCU, ResDA, ResDAA, ResDB, ResEY, ResFL, ResGA, ResGD, ResGO, ResHC, ResHZ, ResI, ResIA, ResIFA, ResIK, ResJA, ResJP, ResJW, ResJX, ResK, ResKH, ResKLA, ResL, ResLB, ResME, ResMI, ResNC, ResNE, ResNJA, ResOB, ResOY, ResPIA, ResPN, ResPP, ResQ, ResQA, ResQJ, ResQR, ResSB, ResTF, ResVA, ResVG, ResWA, ResWB, ResWQA, ResXD, ResXF, ResYC, ResYW, ResZD]
RELEASE	[ResA, ResAM, ResBA, ResDB, ResGA, ResIA, ResNC, ResQA, ResSB, ResU, ResUK, ResVD, ResWA, ResWB, ResYC]
BILLED	[ResA, ResB, ResCB, ResDB, ResEB, ResGA, ResHG, ResIA, ResIU, ResNC, ResU, ResWB, ResXD, ResYC]
DELETE	[ResA, ResAA, ResAC, ResAD, ResADA, ResAEA, ResAF, ResAGA, ResAH, ResAJA, ResAK, ResAM, ResANA, ResAO, ResAQ, ResAQA, ResAV, ResAW, ResBA, ResBB, ResBE, ResBEA, ResBK, ResBL, ResBQ, ResBV, ResC, ResCA, ResCB, ResCD, ResCDA, ResCGA, ResCH, ResCIA, ResCJ, ResCM, ResCN, ResCOA, ResCW, ResCX, ResD, ResDA, ResDB, ResDDA, ResDG, ResDH, ResDIA, ResDJ, ResDMA, ResDN, ResDO, ResE, ResEB, ResEC, ResEF, ResEL, ResEM, ResEO, ResES, ResFB, ResFJA, ResFL, ResFT, ResFU, ResGA, ResGCA, ResGD, ResGE, ResGG, ResGI, ResGL, ResGN, ResGY, ResHCA, ResHH, ResHHA, ResHJ, ResHQA, ResI, ResIA, ResIB, ResIC, ResID, ResIF, ResIG, ResIH, ResIJ, ResIK, ResJDA, ResJG, ResJH, ResJJ, ResJK, ResJM, ResJU, ResJV, ResK, ResKJ, ResKM, ResKO, ResKS, ResKX, ResLC, ResLM, ResLO, ResLOA, ResLPA, ResLT, ResM, ResMD, ResME, ResMHA, ResMI, ResMOA, ResMV, ResN, ResNA, ResNC, ResNE, ResNH, ResNN, ResNQ, ResOC, ResOD, ResOE, ResOG, ResOIA, ResOJ, ResOK, ResOL, ResOM, ResOP, ResOPA, ResOS, ResOZ, ResP, ResPA, ResPC, ResPD, ResPG, ResPIA, ResPJ, ResPK, ResPN, ResPP, ResPR, ResPT, ResQ, ResQA, ResQB, ResQCA, ResQE, ResQH, ResQJ, ResQK, ResQL, ResQM, ResQO, ResQX, ResR, ResRCA, ResRE, ResRG, ResRGA, ResRJA, ResRM, ResROA, ResRP, ResRQ, ResSB, ResSD, ResSF, ResSG, ResSH, ResSI, ResSJ, ResSL, ResSN, ResSP, ResSQ, ResT, ResTE, ResTF, ResTG, ResU, ResUH, ResUK, ResUO, ResUQ, ResUR, ResVAA, ResVB, ResVD, ResVJ, ResVL, ResVW, ResWA, ResWAA, ResWB, ResWC, ResWJ, ResWP, ResWQ, ResXB, ResXD, ResXG, ResXGA, ResXH, ResXI, ResXR, ResXT, ResYA, ResYC, ResYD, ResYE, ResYF, ResYL, ResYMA, ResYQ, ResYV, ResZA, ResZD, ResZE, ResZG, ResZI, ResZL, ResZO, ResZR, ResZV, ResZX]
JOIN-PAT	[ResAM, ResB, ResBZ, ResHG, ResIBA, ResNA, ResNC, ResPY, ResYT]

CODE NOK	[ResA, ResBA, ResDB, ResGA, ResIA, ResNC, ResQA, ResRGA, ResSB, ResWB, ResYC]
CODE ERROR	[ResDB, ResGA, ResYC]
MANUAL	[ResA, ResBA, ResIA, ResNC, ResU]
SET STATUS	[ResBA, ResCH, ResDB, ResGA, ResIA, ResPC, ResQA, ResSB, ResWB, ResXD, ResYC]
EMPTY	[ResBA, ResCH, ResDB, ResGA, ResSB, ResWB, ResXD, ResYC]
CHANGE DIAGN	[ResAA, ResAB, ResAC, ResAD, ResAEA, ResAF, ResAGA, ResAH, ResAHA, ResAI, ResAK, ResANA, ResAO, ResAQ, ResAR, ResAT, ResAW, ResBA, ResBB, ResBF, ResBG, ResBH, ResBHA, ResBK, ResBO, ResBOA, ResBQA, ResBS, ResBV, ResC, ResCA, ResCB, ResCC, ResCD, ResCE, ResCEA, ResCF, ResCG, ResCI, ResCIA, ResCJ, ResCL, ResCN, ResCO, ResCP, ResCQA, ResCX, ResCZ, ResD, ResDA, ResDC, ResDF, ResDFA, ResDG, ResDH, ResDHA, ResDIA, ResDJ, ResDK, ResDL, ResDM, ResDMA, ResDN, ResDO, ResDP, ResEA, ResEC, ResEE, ResEEA, ResEF, ResEFA, ResEGA, ResEI, ResEJA, ResEO, ResEOA, ResER, ResEU, ResF, ResFAA, ResFB, ResFDA, ResFE, ResFF, ResFJA, ResFM, ResFOA, ResFP, ResG, ResGAA, ResGBA, ResGC, ResGD, ResGE, ResGG, ResGI, ResGJA, ResGL, ResGO, ResGPA, ResGQ, ResGR, ResGY, ResGZ, ResH, ResHA, ResHB, ResHC, ResHD, ResHE, ResHH, ResHI, ResHJ, ResHLA, ResHN, ResHV, ResHX, ResI, ResID, ResIH, ResIJA, ResIM, ResINA, ResIQ, ResIR, ResIS, ResIW, ResIY, ResJB, ResJBA, ResJE, ResJF, ResJG, ResJJ, ResJK, ResJNA, ResJV, ResJZ, ResK, ResKE, ResKF, ResKH, ResKJ, ResKL, ResKLA, ResKM, ResKT, ResKW, ResLAA, ResLB, ResLC, ResLD, ResLF, ResLH, ResLIA, ResLJA, ResLK, ResLM, ResLMA, ResLNA, ResLP, ResLQ, ResLW, ResM, ResMA, ResMD, ResME, ResMG, ResMJA, ResNA, ResNB, ResND, ResNE, ResNF, ResNG, ResNL, ResNN, ResNU, ResNY, ResOA, ResOAA, ResOB, ResOC, ResOD, ResOE, ResOEA, ResOG, ResOH, ResOJ, ResOK, ResOL, ResOM, ResP, ResPA, ResPB, ResPD, ResPF, ResPG, ResPJ, ResPK, ResPL, ResPM, ResPN, ResPQ, ResPR, ResPS, ResPT, ResQ, ResQA, ResQAA, ResQB, ResQCA, ResQE, ResQG, ResQH, ResQJ, ResQL, ResQM, ResQP, ResR, ResRB, ResRD, ResRDA, ResRF, ResRFA, ResRG, ResRH, ResRJ, ResRN, ResRNA, ResRP, ResRQ, ResRR, ResRV, ResRW, ResRY, ResS, ResSA, ResSB, ResSC, ResSEA, ResSF, ResSG, ResSH, ResSI, ResSJ, ResSK, ResSLA, ResSP, ResSQ, ResSW, ResT, ResTA, ResTB, ResTF, ResTFA, ResTH, ResTJ, ResTM, ResTP, ResTPA, ResTS, ResTU, ResTX, ResTY, ResUA, ResUB, ResUDA, ResUF, ResUFA, ResUI, ResUJ, ResUK, ResUKA, ResUN, ResUOA, ResUU, ResUZ, ResVA, ResVD, ResVDA, ResVE, ResVF, ResVI, ResVIA, ResVJ, ResVLA, ResVN, ResVO, ResVP, ResVS, ResW, ResWA, ResWB, ResWC, ResWD, ResWG, ResWJ, ResWK, ResWKA, ResWM, ResWMA, ResWO, ResWP, ResWQ, ResWT, ResWX, ResX, ResXA, ResXAA, ResXB, ResXD, ResXDA, ResXE, ResXF, ResXH, ResXI, ResXJ, ResXM, ResXQ, ResXR, ResXV, ResY, ResYAA, ResYF, ResYGA, ResYHA, ResYI, ResYK, ResYKA, ResYL, ResYMA, ResYO, ResYOA, ResYPA, ResYY, ResZ, ResZA, ResZB, ResZC, ResZE, ResZG, ResZH, ResZI, ResZL, ResZM, ResZN, ResZR, ResZV]

Cuadro 13: Recursos extraídos al ejecutar el algoritmo EWEHD en el archivo XEX Hospital Billing - Event Log en modo de proceso *ISOLATED* y modo de estimación *STRICT*.

Archivo XES	**Modo de proceso**	**Modo estimación**	**Duración (seg)**
Sepsis Case	ISOLATED	STRICT	3

Eventos	
Nombre/Tipo	**Recursos**
CRP	[B]
Release B	[E]
Release A	[E]
Release D	[E]
Release C	[E]
Release E	[E]
Admission IC	[J, K, P, W]
Return ER	[?]
ER Triage	[C]
IV Antibiotics	[A, L]
Leucocytes	[B]
IV Liquid	[A, L]
ER Registration	[A, L]
Admission NC	[D, F, G, H, I, J, K, M, N, O, P, Q, R, S, T, U, V, W, X, Y]
LacticAcid	[B]
ER Sepsis Triage	[A, L]

Cuadro 14: Recursos extraídos al ejecutar el algoritmo EWEHD en el archivo XES Sepsis Case - Event Log en modo de proceso *ISOLATED* y modo de estimación *STRICT*.

Archivo XES	Modo de proceso	Modo estimación	Duración (seg)
WABO, CoSeLoG project	ISOLATED	STRICT	3

Eventos	
Nombre/Tipo	**Recursos**
T09-1	[Resource09, Resource12, Resource15, Resource22, test]
T20	[Resource01, Resource03, Resource05, Resource07, Resource08, Resource09, Resource11, Resource12, Resource13]
T04	[Resource01, Resource02, Resource03, Resource04, Resource05, Resource06, Resource07, Resource08, Resource09, Resource10, Resource11, Resource12, Resource13, Resource14, Resource15, Resource16, Resource17, Resource18, Resource19, Resource20, Resource21, Resource22, Resource23, Resource24, Resource25, Resource26, Resource28, Resource29, Resource31, Resource33, Resource34, Resource35, Resource36, Resource38, admin1, admin2, admin3]
T07-4	[Resource02, Resource03, Resource11, Resource26, Resource32]
T19	[Resource01, Resource03, Resource05, Resource07, Resource08, Resource09, Resource11, Resource12, Resource13]
T09-2	[Resource26]
T07-2	[Resource02, Resource09, Resource12, Resource15, Resource21, Resource23, Resource25, Resource28, Resource29, Resource32, Resource33, Resource40, Resource41, admin2, test]
T06	[Resource01, Resource02, Resource03, Resource04, Resource05, Resource06, Resource07, Resource08, Resource09, Resource11, Resource12, Resource13, Resource14, Resource15, Resource16, Resource17, Resource18, Resource20, Resource21, Resource22, Resource23, Resource25, Resource26, Resource28, Resource29, Resource31, Resource33, Resource34, Resource35, Resource36, Resource37, admin2, test]
T18	[Resource05, Resource11]
T13	[Resource17, admin2]
T09-3	[Resource25, Resource28, test]
T03	[Resource01, Resource02, Resource05, Resource06, Resource07, Resource08, Resource09, Resource11, Resource12, Resource13, Resource17, Resource18, Resource21, Resource26, Resource28, Resource30, admin2]
T09-4	[Resource05, Resource09, Resource12, Resource26]
T10	[Resource01, Resource02, Resource03, Resource04, Resource05, Resource06, Resource07, Resource08, Resource09, Resource11, Resource12, Resource13, Resource14, Resource15, Resource16, Resource17, Resource18, Resource20, Resource21, Resource22, Resource23, Resource25, Resource26, Resource28, Resource29, Resource30, Resource31, Resource33, Resource34, Resource35, Resource36, Resource37, admin1, admin2]
T14	[Resource02, Resource03, Resource04, Resource05, Resource06, Resource07, Resource08, Resource10, Resource15, Resource17, Resource19, Resource21, admin1]
T07-3	[Resource11, Resource12, Resource15, Resource26, Resource35]
T11	[Resource01, Resource02, Resource03, Resource04, Resource05, Resource06, Resource07, Resource08, Resource09, Resource13, Resource15, Resource17, Resource21, admin1, admin2]
T16	[Resource01, Resource03, Resource05, Resource07, Resource08, Resource09, Resource11, Resource12, Resource13]
T02	[Resource01, Resource02, Resource03, Resource04, Resource05, Resource06, Resource07, Resource08, Resource09, Resource10, Resource11, Resource12, Resource13, Resource14, Resource15, Resource16, Resource17, Resource18, Resource19, Resource21, Resource22, Resource24, Resource25, Resource26, Resource28, Resource29, Resource30, Resource32, Resource34, Resource38, Resource39, admin2]
T05	[Resource01, Resource02, Resource03, Resource04, Resource05, Resource06, Resource07, Resource08, Resource09, Resource10, Resource11, Resource12, Resource13, Resource14, Resource15, Resource16, Resource17, Resource18, Resource20, Resource21, Resource22, Resource23, Resource25, Resource26, Resource27, Resource28, Resource29, Resource31, Resource34, Resource36, TEST, admin1, admin2]
T07-5	[Resource05, Resource09, Resource12, Resource13, Resource23, Resource25, Resource26, Resource33, Resource37, admin2]
T17	[Resource01, Resource03, Resource05, Resource07, Resource08, Resource09, Resource11, Resource12, Resource13]
T12	[Resource01, Resource02, Resource03, Resource04, Resource05, Resource06, Resource07, Resource13, Resource15, Resource17, Resource19, Resource21, Resource24, admin1]
T15	[Resource01, Resource02, Resource03, Resource04, Resource05, Resource07, Resource11, Resource17, Resource21, Resource26, admin1]
T08	[Resource05, Resource09, Resource12, Resource15, Resource22, Resource25, Resource28, Resource33, test]
T07-1	[Resource02, Resource05, Resource06, Resource09, Resource12, Resource21, Resource22, Resource23, Resource24, Resource26, admin2]

Cuadro 15: Recursos extraídos al ejecutar el algoritmo EWEHD en el archivo XES WABO, CoSe-LoG project en modo de proceso *ISOLATED* y modo de estimación *STRICT*.