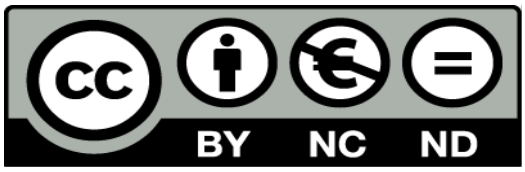
UNIVERSIDAD DE LA RIOJA

TESIS DOCTORAL

Título
Planificación operativa del mantenimiento de material rodante ferroviario de alta velocidad
Autor/es
Juan Suardíaz Gargallo
Director/es
Eliseo Pablo Vergara González
Facultad
Escuela Técnica Superior de Ingeniería Industrial
Titulación
Departamento
Ingeniería Mecánica
Curso Académico



publicaciones.unirioja.es
E-mail: publicaciones@unirioja.es

UNIVERSIDAD DE LA RIOJA

DEPARTAMENTO DE INGENIERÍA MECÁNICA
PROGRAMA DE DOCTORADO DE INNOVACIÓN
EN INGENIERÍA DE PRODUCTO Y PROCESOS I

TESIS DOCTORAL

PLANIFICACIÓN OPERATIVA DEL MANTENIMIENTO DE MATERIAL RODANTE FERROVIARIO DE ALTA VELOCIDAD

DIRECTOR:
Dr. Eliseo Vergara González

AUTOR:
D. Juan Suardíaz Gargallo

JULIO 2019

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

"Sea grato para ti organizar de forma mesurada las acciones de modo que se llenen los graneros de fruta madura".

Hesiodo, Trabajos y Días, 305.

La disponibilidad del material rodante y sus costes de mantenimiento son cruciales para la rentabilidad de sociedades propietarias u operadoras de flotas ferroviarias. Un aumento de la disponibilidad del material rodante permite realizar el mismo volumen de servicios comerciales con una flota menor o ampliar el aprovechamiento de la flota disponible efectuando más servicios comerciales. Por este motivo los propietarios y operadores están interesados en métodos que aumenten la disponibilidad de sus flotas.

Se entiende como disponibilidad del material rodante al tiempo durante el cual se le puede emplear para el servicio comercial. La disponibilidad de la flota viene limitada por las paradas operacionales y por las paradas de mantenimiento. Las paradas operacionales se deben entre otros factores a las interrupciones nocturnas, a las fluctuaciones estacionales de la demanda de

transporte o a las restricciones marcadas por la infraestructura para su mantenimiento. Por ejemplo, típicamente no hay servicios comerciales nocturnos y los operadores estacionan el material rodante en vías de aparcamiento durante la noche. También es habitual que la cantidad de servicios comerciales varíe durante la semana, con una mayor afluencia de viajeros los viernes por la tarde y los lunes por la mañana, o con fluctuaciones estacionales debidas a los períodos vacacionales.

Las paradas de mantenimiento se emplean para realizar las tareas técnicas necesarias que garantizan la seguridad y la funcionalidad de los vehículos en el marco de las obligaciones legales y contractuales del mantenedor. Estas tareas técnicas se realizan en diferentes equipamientos por un equipo de técnicos y con el vehículo en un estado de seguridad definido para cada tarea. Debido a las distancias entre los equipamientos, a las necesarias cualificaciones de los trabajadores y a los tiempos necesarios para cambiar el estado del vehículo, la secuenciación de las tareas de mantenimiento en un flujograma desempeña un papel determinante en la duración de la parada de mantenimiento.

Esta secuenciación de tareas hoy en día es una tarea larga y tediosa que realiza de manera manual un planificador de mantenimiento. Ello limita su aplicación a las paradas de mantenimiento más habituales, impidiendo o dificultando enormemente una secuenciación de todas y cada una de las posibles combinaciones de tareas o el análisis de múltiples configuraciones de equipos de técnicos para determinar su composición óptima.

Así pues, un método informatizado y fiable para secuenciar las tareas de mantenimiento en un flujograma optimizado podría reducir las paradas de mantenimiento de los vehículos incrementando su disponibilidad. Adicionalmente, este método informatizado de secuenciación de flujogramas serviría para realizar simulaciones y análisis del aprovechamiento de los recursos, contribuyendo a una reducción de los costes.

1.1 MANTENIMIENTO DEL MATERIAL RODANTE DE ALTA VELOCIDAD

La estrategia clásica del mantenimiento de material rodante en general y de trenes de alta velocidad en particular consiste en la ejecución de una serie de tareas proactivas ("tareas realizadas en un plazo determinado entes de que ocurra un fallo, con el propósito de prevenir que el equipo caiga en estado de fallo", (Moubray, 1997)) y en la ejecución de tareas de mantenimiento correctivo para devolver el equipo a su estado operativo después de un fallo, tal como se define en CEN EN 13306:2.

Las tareas proactivas están especificadas en un plan de mantenimiento inicialmente fijado por el fabricante. El plan de mantenimiento está estructurado en intervenciones que consisten en grupos de tareas proactivas que deben realizarse antes de que se alcance un determinado intervalo temporal o de kilometraje. Los intervalos de las intervenciones superiores son habitualmente múltiplos de las intervenciones inferiores (p. ej. 25.000 – 100.000 – 200.000 km, etc.). Las intervenciones superiores suelen incluir las tareas de las intervenciones de menor intervalo.

Modificaciones del plan de mantenimiento están sujetas a exhaustivos análisis de riesgos según la Directiva Europea 2004/49/EC y a la aprobación de la Agencia Estatal de Seguridad Ferroviaria en el caso de que puedan afectar a la seguridad de la flota.

Las intervenciones de los intervalos inferiores en material rodante de alta velocidad consisten básicamente en inspecciones visuales del sistema de rodadura y se realizan durante las paradas operacionales nocturnas de manera que apenas disminuyen la disponibilidad de la flota al realizarse durante un turno de noche. Las intervenciones con intervalos superiores suponen una estadía superior a un turno de noche y sí tienen un impacto en la disponibilidad.

Sin embargo, no es obligatorio que todas las tareas pertenecientes a la intervención se realicen durante la misma estadía de mantenimiento siempre y cuando que ninguna sobrepase el intervalo marcado. Esto permite plantear una estrategia de mantenimiento distribuido, que consiste en adelantar las tareas de una intervención y agruparlas en módulos con las tareas de las intervenciones inferiores (véase Figura 1) de manera que se obtengan varias ventajas:

- Nivelación de la carga de trabajo del personal y del aprovechamiento de la infraestructura del taller.
- Reducción de estadías largas que inmovilicen el tren durante varios días al realizarlas en bloques menores durante los turnos de noche.

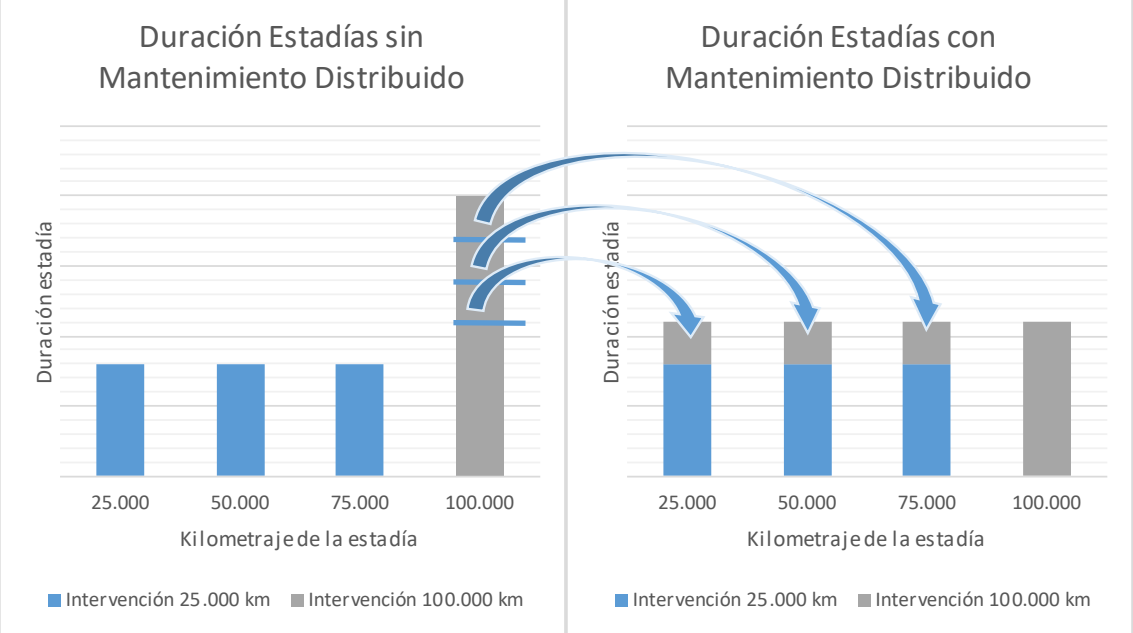


Figura 1 Efecto del mantenimiento distribuido en la duración de estadías

La aplicación del mantenimiento distribuido implica un arduo trabajo del planificador de mantenimiento para determinar el contenido de los módulos de tareas y cómo integrarlas en las intervenciones inferiores de manera que el tiempo de ejecución sea el mínimo posible. Ya no se trata sólo de optimizar la duración de un conjunto predeterminado de tareas, sino de repartir las tareas de distintas maneras entre varias estadías para conseguir un óptimo global.

1.1.1 RESTRICCIONES POR EL ESTADO DE SEGURIDAD DEL VEHÍCULO

No es posible realizar la mayoría de las tareas de mantenimiento con el equipamiento activo debido a su peligrosidad. En algunos casos habrá riesgo de electrocución y en otros, riesgo de traumatismos severos por la acción de piezas móviles. Por ello se hace necesario desactivar y asegurar el equipamiento antes de proceder a su mantenimiento.

En algunos casos la desactivación y aseguramiento tiene lugar a nivel local y no influye a los trabajos desarrollados en el resto del tren. En estos casos, el tiempo de llevar el equipamiento a un estado seguro suele ser corto y está considerado dentro del tiempo previsto para la tarea. Sin embargo, hay estados de seguridad que afectan a todo el tren y que son comunes a muchos equipamientos. Por ejemplo, los trabajos en el techo o bajo bastidor han de realizarse con la catenaria desconectada (hilo de alimentación eléctrica, en alta velocidad a 25 kV AC, 50 Hz) y puesta a tierra, y con el sistema de alta tensión del tren desconectado y cortocircuitado.

Los cambios del estado de seguridad pueden tomar bastante tiempo, especialmente la reconexión de sistemas eléctricos de alimentación como la catenaria a 25 kV. En tal caso hay que:

- Informar a todo el personal que se encuentre trabajando en el vehículo (en algunos casos con longitudes de 400 m).
- Retirar las puestas a tierra y sistemas de cortocircuito de los convertidores auxiliares y de tracción (hasta un total de 12 convertidores en algunas flotas).
- Retirar las puestas a tierra de los transformadores y del sistema de alta tensión del tren.
- Cerrar todas las tapas de los cofres eléctricos.

- Retirar la puesta a tierra de la catenaria.
- Reconectar la catenaria.
- Subir el pantógrafo.
- Cerrar el disyuntor principal del vehículo.

Es obvio que este tipo de operaciones debe reducirse a un mínimo imprescindible. Máxime cuando durante este tiempo no suele ser posible realizar actividad alguna de mantenimiento debido a la incertidumbre del estado real del vehículo (no está ni verdaderamente desconectado ni del todo activado).

Ejemplos de estados de seguridad del vehículo en material rodante de alta velocidad son:

- Catenaria conectada / desconectada y puesta a tierra.
- Sistema de alta tensión y convertidores estáticos conectados / desconectados y puestos a tierra.
- Batería (110 V DC) conectada / desconectada y puesta a tierra.
- Alimentación externa (440 V 3AC, 50 Hz) conectada / desconectada y puesta a tierra.
- Tubería depósito principal (10 bar, sistema de alimentación de los sistemas neumáticos) presurizada / despresurizada y compresor condenado.

Una tarea de mantenimiento puede requerir que un estado de seguridad esté desactivado (p. ej. la inspección del pantógrafo se hace con alta tensión puesta a tierra) o que esté activado (p. ej. la prueba del sistema de freno requiere de los compresores y por lo tanto de alta tensión) o puede ser indiferente (p. ej. la inspección visual de los asientos se puede hacer con o sin alta tensión).

El planificador de mantenimiento debe agrupar tareas de mantenimiento que requieren estados de seguridad iguales o compatibles y minimizar los cambios de estados de seguridad para conseguir un tiempo total de estadía lo más corto posible.

1.1.2 RESTRICCIONES POR LA LOCALIZACIÓN DE LOS EQUIPAMIENTOS

El material rodante de alta velocidad tiene longitudes típicas que van desde los 200 m hasta los 400 m. Los equipamientos pueden encontrarse sobre el techo (por ejemplo, pantógrafo, descargador de sobretensión, seccionadores, etc.), en el interior (aseos, restaurante, asientos, etc.), en el exterior (ventanas, puertas, etc.) y en la zona bajo bastidor (bogies, convertidores estáticos, compresores, etc.). La Figura 2 ilustra las posibles localizaciones de equipamientos.

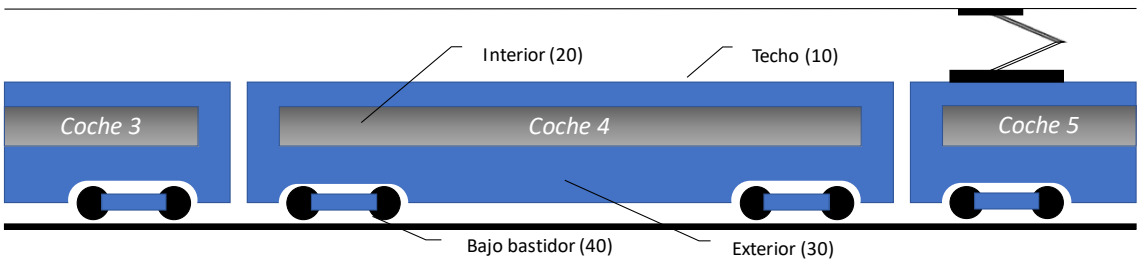


Figura 2 Localizaciones de equipamientos

El tiempo requerido por un técnico para desplazarse de un equipamiento a otro no es despreciable. El tiempo necesario para recorrer 200 m con herramientas es de unos 4 minutos. El peor caso posible es acceder al techo, puesto que típicamente sólo hay un punto de acceso en el extremo de las toperas del taller (donde terminan las vías de mantenimiento). Si el técnico

se encuentra en la zona bajo bastidor del otro coche extremo del tren y necesita subir a su techo puede emplear fácilmente 9 minutos en recorrer 200 m en una dirección, acceder a la plataforma elevada, y recorrer los 200 m en la dirección opuesta.

En un tren con ocho coches y cuatro localizaciones por coche (techo, interior, exterior y bajo bastidor) hay un total de 496 posibles trayectos, asumiendo que el tiempo en recorrer un trayecto es independiente de su sentido. Esta información es suministrada a al algoritmo secuenciador en una tabla como la de la Figura 3.

Area	Coche	Localización	110	120	130	140	210	220	230	240	310	320	330	340	410	420	430	440	510	520	530	540	610	620	630	640	710	720	730	740	810	820	830	840
bajo bast. [040]	8	840	6	6	4	4	6	6	4	4	7	7	3	3	7	7	3	3	8	8	2	2	8	8	2	2	9	9	1	1	9	9	1	0
exterior [030]	8	830	6	6	4	4	6	6	4	4	7	7	3	3	7	7	3	3	8	8	2	2	8	8	2	2	9	9	1	1	9	9	0	
interior [020]	8	820	6	4	6	6	6	4	6	6	7	3	7	7	7	3	7	7	8	2	8	8	8	2	8	8	9	1	9	9	9	0		
techo [010]	8	810	4	6	6	6	4	6	6	6	3	7	7	7	3	7	7	7	2	8	8	8	2	8	8	8	1	9	9	9	0			
bajo bast. [040]	7	740	6	6	4	4	6	6	3	3	7	7	3	3	7	7	2	2	8	8	2	2	8	8	1	1	9	9	1	0				
exterior [030]	7	730	6	6	4	4	6	6	3	3	7	7	3	3	7	7	2	2	8	8	2	2	8	8	1	1	9	9	0					
interior [020]	7	720	6	4	6	6	6	3	6	6	7	3	7	7	7	2	7	7	8	2	8	8	8	1	8	8	9	0						
techo [010]	7	710	4	6	6	6	3	6	6	6	3	7	7	7	2	7	7	7	2	8	8	8	1	8	8	8	0							
bajo bast. [040]	6	640	5	5	3	3	5	5	3	3	6	6	2	2	6	6	2	2	7	7	1	1	7	7	1	0								
exterior [030]	6	630	5	5	3	3	5	5	3	3	6	6	2	2	6	6	2	2	7	7	1	1	7	7	0									
interior [020]	6	620	5	3	5	5	5	3	5	5	6	2	6	6	6	2	6	6	7	1	7	7	7	0										
techo [010]	6	610	3	5	5	5	3	5	5	5	2	6	6	6	2	6	6	6	1	7	7	7	0											
bajo bast. [040]	5	540	5	5	3	3	5	5	2	2	6	6	2	2	6	6	1	1	7	7	1	0												
exterior [030]	5	530	5	5	3	3	5	5	2	2	6	6	2	2	6	6	1	1	7	7	0													
interior [020]	5	520	5	3	5	5	5	2	5	5	6	2	6	6	6	1	6	6	7	0														
techo [010]	5	510	3	5	5	5	2	5	5	5	2	6	6	6	1	6	6	6	0															
bajo bast. [040]	4	440	4	4	2	2	4	4	2	2	5	5	1	1	5	5	1	0																
exterior [030]	4	430	4	4	2	2	4	4	2	2	5	5	1	1	5	5	0																	
interior [020]	4	420	4	2	4	4	4	2	4	4	5	1	5	5	5	0																		
techo [010]	4	410	2	4	4	4	2	4	4	4	1	5	5	5	0																			
bajo bast. [040]	3	340	4	4	2	2	4	4	1	1	5	5	1	0																				
exterior [030]	3	330	4	4	2	2	4	4	1	1	5	5	0																					
interior [020]	3	320	4	2	4	4	4	1	4	4	5	0																						
techo [010]	3	310	2	4	4	4	1	4	4	4	0																							
bajo bast. [040]	2	240	3	3	1	1	3	3	1	0																								
exterior [030]	2	230	3	3	1	1	3	3	0																									
interior [020]	2	220	3	1	3	3	3	0																										
techo [010]	2	210	1	3	3	3	0																											
bajo bast. [040]	1	140	3	3	1	0																												
exterior [030]	1	130	3	3	0																													
interior [020]	1	120	3	0																														
techo [010]	1	110	0																															

Figura 3 Tiempos de desplazamiento entre localizaciones

El planificador deberá favorecer que la secuencia de tareas en el flujograma sea tal que los tiempos de desplazamiento entre localizaciones sean mínimos, al menos a lo largo del camino crítico. Tomando como ejemplo la inspección visual de los asientos de viajeros, esta tarea se realiza por un técnico en cada uno de los coches. La secuencia normal de ejecución debería ser desde el coche 1 hasta el 8 consecutivamente y vuelta al punto de partida. Otra secuencia equivalente es comenzando desde el coche 8 y de vuelta al 1. Una tercera posibilidad es inspeccionar los coches impares a la ida y los pares a la vuelta.

Un punto interesante para considerar es que el tiempo de desplazamiento depende por un lado de la configuración del vehículo, pero también del taller de mantenimiento. Si en el taller sólo hay un acceso en cada plataforma elevada los tiempos de desplazamiento al techo serán más desfavorables que si hay dos accesos a cada plataforma, uno a cada extremo del tren.

1.1.3 RESTRICCIONES POR LA DISPONIBILIDAD DE PERSONAL DE MANTENIMIENTO

Cada tarea de mantenimiento exige una determinada cantidad de personal con unas cualificaciones determinadas. La gestión de la cualificación del personal adquiere una relevancia especial con la Regulación de la Comisión Europea nº 445/2011, que en su Anexo II, apartado I 6.1.c obliga a que “La organización deberá instaurar un sistema de gestión de la competencia que prevea la asignación de personal con la competencia adecuada a las tareas pertinentes”. Si

bien el ámbito de aplicación de la regulación 445/2011 se circunscribe al mantenimiento de vagones de mercancías, se ha constituido como el estado del arte para todo el mantenimiento de material rodante y será ampliado oficialmente en breve a vehículos de transporte de viajeros.

Por ello, y para garantizar una asignación adecuada de personal según sus cualificaciones, el planificador de mantenimiento debe trabajar con una lista de recursos (personal) y sus cualificaciones. Debe comprobar si los recursos disponibles son suficientes para la realización de los flujogramas propuestos dentro del algoritmo. Si los flujogramas requieren una cantidad superior a la cantidad disponible de recursos con una cualificación dada, son rechazados. El planificador también asigna tareas específicas a cada recurso, de manera que cada uno recibe un cronograma-itinerario con la lista de sus tareas asignadas, sus localizaciones, duraciones y la hora a la que comienzan.

Esta asignación de tareas permite analizar la productividad del equipo de mantenimiento, sus tiempos de espera, e identificar si algún recurso está infrautilizado o es superfluo.

1.1.4 RESTRICCIONES PREDECESOR – SUCESOR

La inmensa mayoría de las tareas de mantenimiento de una intervención son independientes entre sí y no presentan relaciones predecesor-sucesor. No es relevante si primero se realiza la inspección de techo y después la comprobación funcional de puertas o viceversa. Tampoco es relevante en qué orden se comprueba el espesor de las guarniciones de freno en los bogies.

Sin embargo, hay casos en los que existen parejas de tareas que se ejecutan bajo diferentes estados de seguridad, con una relación predecesor-sucesor, y que deben realizarse durante la misma estadía. Tal es el caso de una inspección con el equipamiento desactivado y en la que se desmonta algún componente cuyo correcto funcionamiento debe comprobarse a posteriori con el equipamiento de nuevo en servicio. Un ejemplo sería la sustitución del filtro de aceite de un compresor neumático, que se realiza sin alta tensión y sin batería, y la posterior prueba de estanqueidad del sistema neumático, que se realiza con alimentación de alta tensión.

Estas relaciones predecesor-sucesor deben estar identificadas en los datos de entrada del planificador de mantenimiento, que se asegura de que ambas operaciones se realicen durante la misma estadía y en el orden correcto.

1.1.5 PRIORIZACIÓN DE TAREAS DE MANTENIMIENTO

En algunos casos, el número de tareas para secuenciar es fijo. Sin embargo, en muchas aplicaciones, como distribuir las tareas de una intervención a lo largo de varias estadías de intervenciones inferiores (véase la Figura 1), la cantidad de tareas es variable y depende del tiempo disponible. Por esta razón es necesario definir un método para priorizar tareas.

Las tareas de una intervención no necesitan ser realizadas juntas durante la misma estadía del vehículo, pero cada tarea debe ser ejecutada como muy tarde justo antes de alcanzar el intervalo de la intervención. De acuerdo con esta restricción, el planificador calcula lo cerca que se encuentra dicha tarea de alcanzar su límite desde su última ejecución. Si la tarea en cuestión está mas cerca de un determinado umbral de “prioridad alta” será realizada en esta estadía. Lopes et al. (2017) describen un método más complejo de asignación de prioridad basado en un enfoque orientado al riesgo. Consideran el porcentaje de retraso respecto al intervalo de la intervención, la tasa de impacto a la operación comercial y la criticalidad de la tarea. Se ha descartado su uso debido a la complejidad del cálculo de riesgos.

Todas las tareas de prioridad alta deben ejecutarse durante la estadía. Si el tiempo necesario es mayor que el tiempo de parada disponible, el tren deberá permanecer en el taller.

1.1.6 APLICACIONES DE UN MÉTODO INFORMATIZADO PARA LA SECUENCIACIÓN DE TAREAS DE MANTENIMIENTO

La secuenciación de tareas de mantenimiento en un flujograma con duración mínima es un trabajo prolijo y tedioso realizado manualmente por el planificador de mantenimiento. Las herramientas informáticas disponibles actualmente en los sistemas de gestión del mantenimiento (*Computerized Maintenance Management System*, CMMS) no resuelven esta tarea de forma automática tomando en consideración las restricciones y condicionantes mencionados, sino que a lo sumo proveen un entorno gráfico en el que se representan las tareas, la disponibilidad de recursos y así ayudan al planificador a realizar su secuenciación.

Una aplicación informática (algoritmo de secuenciación) que secuencie las tareas cumpliendo las restricciones indicadas anteriormente puede aplicarse a la resolución de varios problemas como los que se exponen a continuación.

PLANIFICACIÓN DE ESTADÍAS

La funcionalidad general de la secuenciación de tareas de mantenimiento es la búsqueda de un flujograma de tareas cuya duración sea lo más corta posible y que tenga en cuenta los tiempos de desplazamiento entre localizaciones de equipamientos, los tiempos para cambiar los estados de seguridad del tren, la disponibilidad y cualificación de recursos, y las relaciones predecesor-sucesor de algunas parejas de tareas.

El empleo extensivo de la secuenciación a todas las intervenciones de mantenimiento ayudaría a mejorar la productividad de los recursos disponibles.

ANÁLISIS DE LA DEMANDA DE PERSONAL

Los costes de personal suponen la posición más relevante de los costes de mantenimiento. Es por ello esencial estimar con la mayor exactitud la cantidad y cualificación del personal de mantenimiento.

Tradicionalmente se realizan los cálculos de personal mediante una técnica *bottom-up* (ascendente). Esto consiste en listar todas las tareas de mantenimiento, realizar una estimación de su esfuerzo en horas-hombre, multiplicar el esfuerzo por su número de veces que se ejecutarán las tareas en la flota durante el contrato de mantenimiento y así se llega a una suma total neta de horas-hombre. Dividiendo por los años de mantenimiento y por las horas que trabaja cada recurso al año se consigue una cantidad neta de personal. Finalmente, se aplica un factor de productividad basado en el mantenimiento de otras flotas semejantes y se obtiene la cantidad total necesaria de personal.

Este método es capaz de dar una estimación aproximada y un orden de magnitud aceptable. Sin embargo, no tiene en cuenta que durante las épocas de ejecución de las grandes revisiones la demanda de personal sube, ni tiene en cuenta optimizaciones del flujograma de cada intervalo.

La secuenciación optimizada de cada uno de las intervenciones de mantenimiento permite identificar el nivel de productividad de cada uno de los recursos empleados. También facilita analizar los impactos en la duración de la intervención ante una reducción de personal. Así se

hace posible identificar la composición mínima del equipo técnico capaz de realizar la intervención en la estadía prevista.

ANÁLISIS DE RETORNO DE INVERSIONES EN INFRAESTRUCTURA

Los tiempos de desplazamiento del personal de mantenimiento y los tiempos para cambiar el estado de seguridad del vehículo dependen en gran medida de la infraestructura del taller. Por ello, mejoras del taller permiten aumentar la productividad del personal de mantenimiento. Sin embargo, la cuantificación de estos aumentos de productividad no es sencilla.

El algoritmo de secuenciación es capaz de estimar variaciones de la productividad calculando los tiempos necesarios para la ejecución del mantenimiento teniendo en cuenta las reducciones en los tiempos de desplazamiento y de cambio del estado de seguridad antes y después de modificar la infraestructura. De esta forma es posible estimar el tiempo de reembolso de la inversión de manera altamente cualificada.

ESTRATEGIA DE MANTENIMIENTO DISTRIBUIDO

El mantenimiento distribuido consiste en dividir una intervención en módulos que puedan realizarse durante una estadía corta, preferiblemente durante un turno de noche. De este modo la carga de trabajo del equipo de mantenimiento es más regular y se aumenta la disponibilidad del vehículo (véase Figura 1, página 4).

La implementación del mantenimiento distribuido implica un arduo trabajo del planificador de mantenimiento para determinar el contenido de los módulos de tareas y cómo integrarlas en las intervenciones inferiores de manera que el tiempo de ejecución sea el más corto posible.

ESTRATEGIA DE MANTENIMIENTO OPORTUNISTA

El mantenimiento oportunista consiste en aprovechar estadías operacionales no planificadas de los trenes para realizar parte de las tareas de mantenimiento. Tal sería el caso de vehículos aparcados durante la noche en un taller. Si el taller tiene vías libres de mantenimiento con foso y acceso al techo, y si hay personal cualificado disponible, será posible realizar parte del mantenimiento pendiente.

Esta estrategia es difícilmente realizable con los medios actuales. Para ponerla en práctica se necesitaría que un planificador de mantenimiento analizara las tareas pendientes o a punto de vencer, las priorizara y secuenciara un flujograma optimizado. La dificultad es que el planificador dispone de muy poco tiempo desde la detección de la oportunidad y su puesta en práctica.

La implementación del mantenimiento oportunista pasa por la automatización. El CMMS (sistema de gestión de mantenimiento) gestiona la lista de tareas de mantenimiento y sus vencimientos. El algoritmo de secuenciación puede aplicar un criterio de priorización y planificar la mayor cantidad posible de tareas en el tiempo disponible.

De este modo es posible aplicar una estrategia de mantenimiento oportunista y aumentar la disponibilidad de la flota.

2. ESTADO DEL ARTE

"Tal es la situación de los peloponesios, según a mí me parece, o similar; la nuestra, en cambio, carece de las deficiencias que en ellos he criticado, y cuenta además con otras ventajas mayores que ellos no comparten".

Tucídides, Historia de la Guerra del Peloponeso, Libro I, 143 (primer discurso de Pericles).

La gestión de activos es una creciente ocupación de los operadores y de los propietarios de flotas de trenes de alta velocidad. Prueba de ello es la propagación de la serie de normas ISO 55000, que desde su publicación en 2014 encuentra cada vez mayor resonancia a nivel mundial. Parte relevante de la gestión de activos es la importancia que atribuye a la disponibilidad y a los costes de mantenimiento en los costes de ciclo de vida (*life cycle costs*) de activos (material rodante, navíos, refinerías, etc.).

En 1994, Dekker & Smeitink (1994) tratan de la importancia de la disponibilidad de activos para maximizar su rendimiento y de que una posibilidad consiste en aprovechar paradas imprevistas por motivos operacionales para realizar tareas de mantenimiento.

Tan & Kramer (1997) disertan acerca de la importancia de la fiabilidad y disponibilidad de bienes de equipo en factorías químicas y de su mejora mediante mejores estrategias de mantenimiento.

Carrese & Ottone (2006) efectúan un decisivo paso en la gestión de activos al plantear un modelo global de costes de ciclo de vida de una flota de tranvías.

Keizer, Teunter & Veldman (2015) afirman que los costes de mantenimiento pueden constituir hasta el 60% de los costes de producción de empresas manufactureras.

Busstra (2015) y Apallius de Vos & van Dongen (2015) subrayan la importancia de la disponibilidad del material rodante para los Ferrocarriles Holandeses. Ambos artículos proponen una modularización y agrupación de grandes revisiones (grandes intervenciones en intervalos de entre 4 y 12 años) para ejecutar los módulos individualmente en las épocas menos ocupadas del año o durante las horas valle. La agrupación de tareas de mantenimiento en subgrupos de las intervenciones en el marco de una estrategia de mantenimiento distribuido es una de las aplicaciones de un algoritmo de secuenciación.

Jonsson (2015) apunta a la necesidad de reducir las paradas de mantenimiento de centrales de generación eléctrica para afrontar el aumento de demanda de potencia eléctrica.

Huan (2017) propone un sistema de programación inteligente de tareas de mantenimiento de campos de paneles fotovoltaicos con el objetivo de alcanzar el máximo grado de desempeño del activo.

Raknes et al. (2017) presentan un modelo de asignación de tareas de mantenimiento en un parque marítimo de aerogeneradores para reducir sus costes de mantenimiento y aumentar su profitabilidad.

Valdivieso-Sarabia et al. (2017) busca un método de asignación de tareas de mantenimiento en una "*smart city*" para optimizar el aprovechamiento de los activos de la ciudad reduciendo los tiempos de reparación.

Men et al. (2018) abordan la mejora del aprovechamiento de la infraestructura ferroviaria buscando una armonización del mantenimiento de la infraestructura con los horarios del servicio comercial para optimizar la gestión de activos global.

De las anteriores publicaciones se infiere el alto grado de concienciación de la comunidad científica y empresarial respecto a la gestión de activos y la relevancia de su disponibilidad productiva.

Respecto al método aplicado en la resolución del problema, varios autores han publicado soluciones para problemas similares, pero ninguno de ellos cumple con todos los requisitos y restricciones, lo que ha llevado al desarrollo de un método propio.

Como se verá más adelante, el algoritmo propuesto en esta tesis presenta como elemento más novedoso el uso intensivo de matrices de conectividad de grafos. Tampoco se ha encontrado ningún método comparable.

En los siguientes párrafos se expondrán diferentes enfoques, sus logros y sus carencias respecto al problema objeto de esta tesis.

Budai et al. (2004) afirma que el aumento de la demanda de infraestructura de transporte público incrementa el deterioro de activos (vías, electrificación, material rodante) y reduce el tiempo disponible para mantenimiento. Su artículo expone un método para reducir el tiempo de ocupación de la vía para su mantenimiento con el objetivo de usar mayoritariamente períodos sin servicio comercial. Las tareas de mantenimiento se pueden realizar bajo tres

estados de seguridad diferentes: primero durante los tiempos entre trenes (aproximadamente 30 minutos), segundo durante fines de semana y bloqueando la vía, tercero bloqueando la vía durante varios días hasta que se completa el trabajo. Las tareas de mantenimiento son ejecutadas por diversos equipos que pueden estar trabajando en paralelo en diferentes vías. Estas restricciones marco son similares a aquellas de material rodante de alta velocidad con estados de seguridad, diferentes equipos y el tiempo de traslado de una localización a la siguiente. El artículo no aporta ninguna descripción de las tres heurísticas aplicadas. El problema de asignación del personal de mantenimiento no está incluido y el tiempo requerido para cambiar el estado de seguridad no está considerado. Los algoritmos se aplican a un problema con sólo 16 tareas de mantenimiento y 5 enlaces ferroviarios.

El problema de planificación de proyecto con restricción de recursos (*Resource-Constrained Project Scheduling Problem*, RCPSP) consiste en encontrar el flujograma de un proyecto con el tiempo de ejecución más corto y que pueda ser ejecutado con los limitados recursos disponibles. Montoya-Torres et al. (2010) explican que la planificación con PERT (*Project Evaluation and Review Technique*) y CPM (*Critical Path Method*) no considera restricciones de recursos y no pueden resolver el RCPSP. El RCPSP es NP-hard y sólo puede ser resuelto con algoritmos heurísticos. El método propuesto por Montoya-Torres et al. (2019) es un algoritmo genético (*Genetic Algorithm*, GA) con listas lineales de tareas y no usa matrices de conectividad para representar los grafos de los flujogramas. Este GA no considera posibles restricciones adicionales debidas a los estados de seguridad del tren requeridos para cada tarea de mantenimiento. Tampoco considera el tiempo de desplazamiento de una tarea a la siguiente. El GA ha sido aplicado en ejemplos de 30, 60 y 90 tareas.

Ait-Kadi et al. (2011) sugieren un modelo mejorado para la asignación de recursos (solución del RCPSP). Han aplicado LINGO, un software de programación lineal. De nuevo, no se usa la matriz de conectividad de grafo y las restricciones dadas por los estados de seguridad del material rodante durante cada tarea de mantenimiento no son consideradas. El tiempo de desplazamiento de una tarea a la siguiente tampoco se considera. El ejemplo resuelto tiene 4 recursos de tres tipos diferentes, 4 herramientas de apoyo (también recursos) y un total de 23 tareas en 11 máquinas.

Peng et al. (2011) presentan un modelo de flujograma en el espacio-tiempo para resolver el problema de planificación de mantenimiento de vías. El objetivo es minimizar los costes totales de desplazamiento del equipo de mantenimiento, así como el impacto de proyectos de mantenimiento en la operación ferroviaria. Peng et al. (2011) aplican un enfoque heurístico iterativo por medio de una matriz tridimensional que representa los movimientos a lo largo de la red ferroviaria y del tiempo, pero usándola para formular los costes de transporte en un modelo de programación lineal. Esto es de nuevo una propuesta para solucionar el RCPSP y considera los tiempos de desplazamiento. Sin embargo, las restricciones debidas a los estados de seguridad y los tiempos para cambiarlos no se tienen en cuenta.

Sarker et al. (2013) desarrollaron un algoritmo evolucionario híbrido para solventar el problema de planificación de tareas (*Job Scheduling Problem*, JSP). JSP es similar al problema planteado en esta tesis. Consiste en asignar tareas a máquinas, que es lo mismo que asignar recursos a tareas de mantenimiento. Sin embargo, el algoritmo propuesto no considera los tiempos de preparación de las máquinas (equivalente a los tiempos de cambio de los estados de seguridad) y no usa grafos como modelo del problema, sino listas de tareas para cada máquina.

Dao et al. (2016) proponen un algoritmo genético (GA) para solucionar la planificación de la producción (qué hacer, cuánto hacer, dónde hacerlo y en qué orden hacerlo). Es un problema NP-hard. Es el mismo problema planteado, pero con otros términos:

- "tipo y número de productos para producir" = "tareas de mantenimiento para ser ejecutadas"
- "asignación de productos a líneas de producción" = "asignación de tareas de mantenimiento a recursos"
- "secuencia de producción de los productos seleccionados" = "determinación del grafo del flujograma"

Sin embargo, el algoritmo desarrollado por Dao et al. no toma en consideración los estados de seguridad de las máquinas ni el tiempo requerido para cambiar su configuración cuando se cambia el producto. El algoritmo no usa ningún grafo como modelo subyacente, sino secuencias de fabricación para las líneas de producción.

Aguirre & Papageorgiu (2018) se ocupan del mismo problema que Dao et al. (2016). Aguirre & Papageorgiu aplican programación lineal en enteros mixta (*Mixed-Integer Linear Programming*, MILP). Sin embargo, presentan las mismas limitaciones que Dao et al.

Rashidnejad et al. (2018) abordan un algoritmo genético con clasificación no-dominada II (*Non-dominated Sorting Genetic Algorithm II*, NSG-II) para reducir los costes y la probabilidad de fallo en el mantenimiento de instalaciones geográficamente dispersas. Busca una secuencia para cada uno de los equipos de mantenimiento considerando la probabilidad de fallo de las distintas instalaciones según su vida útil restante. Este problema también es similar al problema planteado en esta tesis. Considera los costes de transporte de los diferentes equipos de mantenimiento que trabajan en paralelo. Pero el modelo no recoge las influencias de los estados de seguridad de las instalaciones y está más enfocado a los riesgos de no ejecutar una tarea de mantenimiento preventivo.

Desgraciadamente, ninguna de las metodologías identificadas en este análisis cumple los requisitos planteados por el problema que se desea resolver, por lo que se hace necesario desarrollar un método nuevo.

3. MATERIALES Y MÉTODOS

"Así pues, Jenias se personó en Sardes con unos cuatro mil hoplitas procedentes de las ciudades; Próxeno se presentó con cerca de mil quinientos hoplitas y quinientos gimnetas; Sofeneto de Estínfalo con mil hoplitas; Sócrates de Acaya con alrededor de quinientos; mientras que Pasión de Mégara lo hizo con trescientos hoplitas y trescientos peltastas".

Jenofonte, Anábasis, Libro I, 3.

En el desarrollo de esta tesis se han aplicado diferentes materiales y métodos, enumerándose los más relevantes a continuación:

- Algoritmos genéticos.
- Grafos.
- Función de distribución estadística de Weibull.
- Lenguaje de programación Java™.
- Datos reales de mantenimiento de material rodante ferroviario de alta velocidad.

Tal como se ha discutido en el apartado "Estado del Arte", el problema de secuenciación de tareas de mantenimiento teniendo en cuenta las restricciones expuestas en "Mantenimiento del Material Rodante de Alta Velocidad" es un problema NP-hard. Esto implica que el tiempo para encontrar la solución óptima crece de manera superior a la polinomial respecto al número de tareas. En este tipo de casos se opta por aplicar heurísticas como los algoritmos genéticos.

El algoritmo de secuenciación necesita una estructura de datos para representar y manipular los flujogramas. Dado que los flujogramas son un caso particular de grafo, resulta inevitable que se aplique teoría de grafos a la resolución de este problema.

En determinadas aplicaciones del algoritmo de secuenciación como en la aplicación descrita en el apartado “Estrategia de Mantenimiento Oportunista”, página 9, se realizan simulaciones del servicio comercial de un tren de alta velocidad. En esta simulación, el kilometraje recorrido entre estadías del tren en el taller y el tiempo disponible para realizar las tareas de mantenimiento son variables aleatorias modeladas con funciones de distribución de Weibull.

La aplicación informática ha sido desarrollada en el lenguaje de programación Java con el entorno de desarrollo integrado NetBeans. El ordenador empleado ha sido un Asus K551L con sistema operativo Windows 10.

Las tareas de las intervenciones son tareas reales previstas en planes de mantenimiento de flotas actualmente en servicio.

3.1 ALGORITMOS GENÉTICOS

Kramer (2017) define los algoritmos genéticos como un enfoque heurístico para realizar búsquedas aplicables a un amplio rango de problemas de optimización. La evolución es la base de los algoritmos genéticos. En los siguientes párrafos de este apartado se provee una breve visión global de los algoritmos genéticos basada en el libro *Genetic Algorithm Essentials* de Kramer (2017).

El pseudo-código del algoritmo genético clásico es el siguiente:

```
Inicializar población
REPEAT {
     REPEAT {
          Cruce / mestizaje
          Mutación
          Mapeado del genotipo al fenotipo
          Cálculo función de aptitud
     } WHILE (NOT(población completa))
     Selección de población progenitora
} WHILE (NOT(condición de terminación))
```

En primer lugar, se inicializa un conjunto de soluciones, denominado población. Se recomienda que esta inicialización cubra aleatoriamente la totalidad del espacio de soluciones o que modele e incorpore un conocimiento basado en la experiencia.

El cruce es un operador que permite la combinación del material genético de dos o más soluciones. La mayor parte de las especies en la Naturaleza tienen dos progenitores. En algoritmos genéticos es posible definir operadores de cruce con más de dos progenitores. En el cruce se incluyen dos pasos intermedios: la búsqueda de pareja(s) y el apareamiento. La búsqueda de parejas es un procedimiento en el cual se definen los progenitores. El apareamiento en algoritmos genéticos es el método de combinación del material genético.

La mutación es un operador que introduce una perturbación del material genético, es decir que modifica la solución, tras el apareamiento. Las mutaciones tienen una naturaleza esencialmente aleatoria.

El fenotipo es la solución perseguida por el algoritmo genético, mientras que genotipo es la representación de la solución con la que trabaja el algoritmo genético. El mapeado es el procedimiento que traduce el genotipo (representación interna de la solución) al fenotipo (solución).

La función de aptitud toma como entrada el fenotipo y evalúa la capacidad del fenotipo de solucionar el problema de optimización planteado.

La población progenitora es el subconjunto de soluciones que pasa a la siguiente generación y que será la base para la siguiente fase de cruces. Su selección se basa al menos parcialmente en los resultados de la función de aptitud. El criterio más empleado es el llamado elitista, que consiste en seleccionar como progenitores de la siguiente generación a las soluciones que hayan conseguido las mejores puntuaciones de la función de aptitud.

3.2 TEORÍA DE GRAFOS

Biggs (1996) define los grafos de la siguiente manera:

"Formalmente, un *grafo general* Γ consiste en tres cosas: un conjunto $V\Gamma$, un conjunto $E\Gamma$, y una relación de incidencia, es decir, un subconjunto de $V\Gamma \times E\Gamma$. A un elemento de $V\Gamma$ se le denomina *vértice*, a un elemento de $E\Gamma$ se le denomina *borde* (*edge* en inglés), y se requiere que la relación de incidencia sea tal que un borde sea incidente con un vértice (siendo entonces un bucle) o con dos vértices. Si cada borde es incidente con dos vértices y no hay dos bordes que sean incidentes con el mismo par de vértices, entonces decimos que Γ es un *grafo estricto*, o más brevemente, un *grafo*."

Un flujograma es susceptible de ser definido como un grafo en el que las tareas son los vértices y los borde son las relaciones de predecesor / sucesor entre las tareas. Sin embargo, la definición de Biggs (1996) aplica a grafos en los que los pares de vértices que definen los bordes no están ordenados y entonces las relaciones de predecesor / sucesor no están definidas.

Gould (2012) define el caso particular del *grafo direccional* Γ como un conjunto formado por un conjunto de n vértices $V = \{v_1, v_2, \dots, v_n\}$ más un conjunto de bordes definidos como m parejas de vértices ordenados $E = \{(v_i, v_j)_1, (v_k, v_l)_2, \dots, (v_y, v_z)_m\}$. Es decir:

$$\Gamma = \left\{V = \{v_1, v_2, \dots, v_n\}, E = \{(v_i, v_j)_1, (v_k, v_l)_2, \dots, (v_y, v_z)_m\}\right\}$$

El grafo direccional (en lo sucesivo, grafo) es la herramienta matemática perfecta para la representación de flujogramas y se usará profusamente en el desarrollo de esta tesis.

3.2.1 GRAFO Y MATRIZ DE CONECTIVIDAD

Cada grafo, una vez numerados sus vértices, puede ser representado por una matriz de conectividad C. Para un grafo constituido por n vértices (tareas), la matriz de conectividad equivalente tiene una dimensión de $n \times n$ (nota: en este documento los vértices estarán numerados de 0 a n-1). Sus elementos c_{ij} se definen como:

$$c_{ij} = \begin{cases} 1 \; si \; el \; vértice \; j \; es \; predecesor \; del \; vértice \; i \\ 0 \; en \; caso \; contrario \end{cases}$$

La relación biunívoca se ilustra con el siguiente ejemplo:

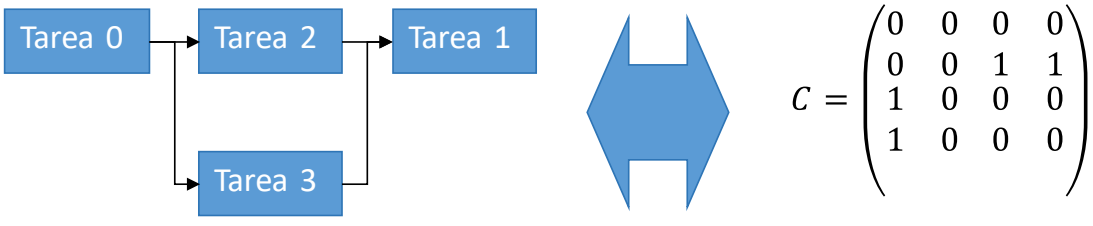


Figura 4 Relación biunívoca entre grafo y matriz de conectividad

Multiplicando la matriz de conectividad por sí misma se obtiene una matriz cuyos elementos son:

$$c_{ij}^2 = \sum_{k=0}^{k=n-1} c_{ik}c_{kj}$$

$$c_{ij}^2 = \textit{número de caminos que parten del vértice } j \textit{ y terminan en el vértice } i \textit{ a través de un vértice intermedio}$$

Esto se debe a que $c_{ik}c_{kj} \neq 0$ si y sólo si el vértice j es predecesor del vértice k y el vértice k es predecesor del vértice i. La suma de los términos $c_{ik}c_{kj}$ da el número total de caminos que parten del vértice j y terminan en el vértice i.

Si se multiplica sucesivas veces la matriz de conectividad por su lado izquierdo, se obtendrán matrices de conectividad de orden superior tales que:

$$c_{ij}^m = \begin{cases} \textit{número de caminos que parten del vértice } j \textit{ y terminan} \\ \textit{en el vértice } i \textit{ a través de } m-1 \textit{ vértices intermedios} \end{cases}$$

Véase el siguiente ejemplo:

$$C^1 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \end{pmatrix}; C^2 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$C^3 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}; C^4 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

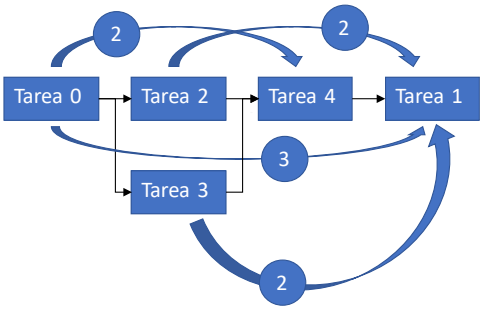


Figura 5 Sucesivas multiplicaciones de la matriz de conectividad por sí misma

La matriz C^2 indica que hay dos caminos que conducen del vértice 0 al vértice 4 a través de un vértice intermedio (precisamente a través de los vértices 2 y 3), que los vértices 2 y 1 están unidos a través de un vértice intermedio, el mismo caso se aplica a los vértices 3 y 1, mientras que la matriz C^3 indica que los vértices 0 y 1 están unidos a través de dos caminos de dos vértices cada uno.

3.2.2 MATRIZ DE CONECTIVIDAD ACUMULATIVA

Se define la matriz de conectividad acumulativa de orden m como:

$$K^m = \sum_{i=1}^{m} C^i$$

$$k_{ij}^m = \begin{cases} \textit{número de caminos partiendo del vértice } j \textit{ y terminando} \\ \textit{en el vértice } i \textit{ a través de } 0, 1, 2, \dots\ m-1 \textit{ vértices intermedios} \end{cases}$$

Esta característica de k_{ij}^m es interesante porque $k_{ii}^m \neq 0$ significaría que el vértice i es predecesor de sí mismo y por lo tanto existe al menos un bucle cerrado en el grafo.

$$C^1 = K^1 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{pmatrix} ; K^2 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & \boxed{1} & 1 & 1 \\ 1 & 0 & 1 & \boxed{1} & 1 \\ 2 & 0 & 1 & 1 & \boxed{2} \end{pmatrix}$$

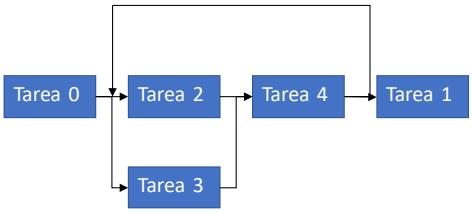


Figura 6 Ejemplo de identificación de un bucle cerrado

El número máximo de vértices enlazados consecutivamente en un grafo sin que se repita ningún vértice en el lazo es n. Como consecuencia, la matriz de conectividad acumulativa de orden $n-1$ es suficiente para identificar cualquier bucle del grafo.

3.2.3 CARACTERÍSTICAS DE MATRICES DE CONECTIVIDAD DE FLUJOGRAMAS

Los flujogramas sólo tienen un vértice inicial y un vértice final, y ningún vértice es predecesor de sí mismo directamente. Si se numera el vértice inicial como 0 y el vértice final como 1, la matriz de conectividad de un flujograma cumplirá que:

$$c_{0i} = 0 \; \forall \; i \mid 0 \leq i < n$$
$$c_{i1} = 0 \; \forall \; i \mid 0 \leq i < n$$
$$c_{ii} = 0 \; \forall \; i \mid 0 \leq i < n$$

Esto quiere decir que los elementos de la primera fila superior, los elementos de la segunda columna de la izquierda y que los elementos de la diagonal de la matriz de conectividad siempre serán iguales a 0.

Adicionalmente, los flujogramas no tienen ningún bucle cerrado. Esto significa que los elementos diagonales de las matrices de conectividad acumulativas de cualquier orden son siempre nulos.

3.2.4 ISOMORFISMO DE GRAFOS Y MATRIZ DE TRANSPOSICIÓN DE VÉRTICES

La matriz de intercambio de vértices es un método ya conocido en teoría de grafos. Pese a no ser una novedad, una breve explicación en este apartado es necesaria puesto que se emplea dentro del algoritmo de secuenciación para variar el orden de los vértices en la matriz de conectividad.

Intuitivamente, dos grafos son isomorfos si tienen las mismas conexiones entre sus vértices. Esta sencilla definición es problemática porque dos grafos isomorfos pueden estar representados de maneras muy diferentes tanto gráficamente como por su matriz de conectividad. Así pues, el isomorfismo de dos grafos a menudo no es reconocible directamente ni a partir de su representación gráfica ni basándose en sus matrices de conectividad.

Por ejemplo, si se numera un grafo de diferentes maneras se obtendrán diferentes matrices de conectividad, aunque se trate del mismo grafo.

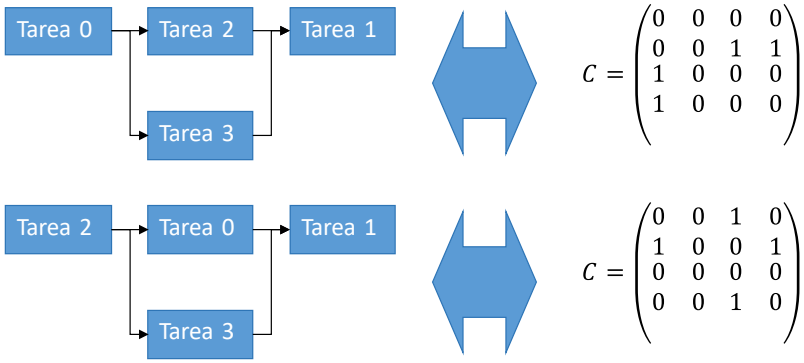


Figura 7 Ejemplo de cambio de la matriz de conectividad cuando se renumeran sus vértices

Análogamente, el grafo definido por una matriz de conectividad se puede representar visualmente de diversas maneras dando la impresión de que se trata de grafos distintos.

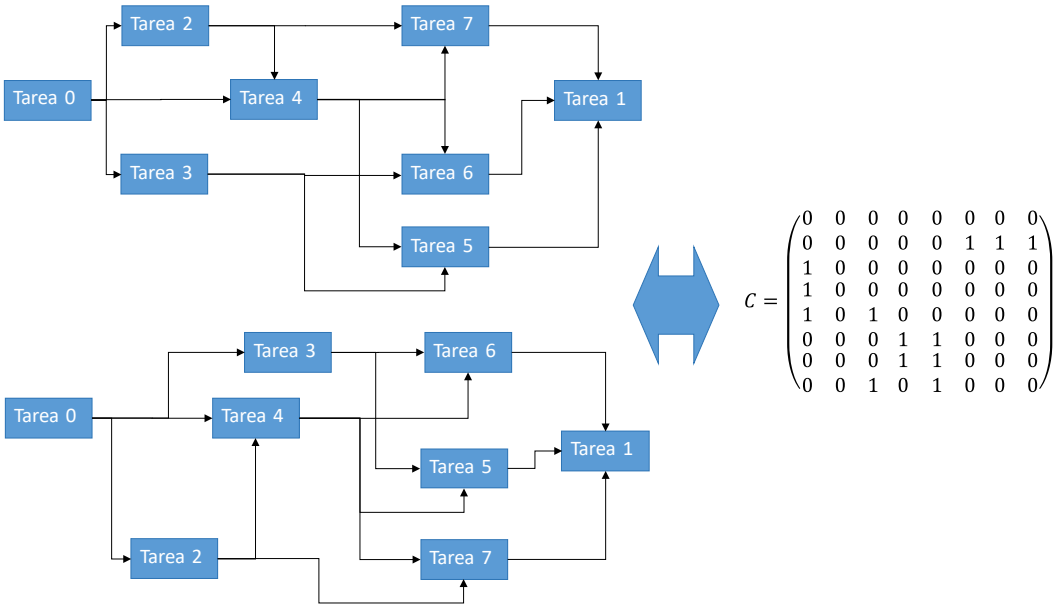


Figura 8 Ejemplo de dos representaciones visuales del mismo grafo con la misma matriz de conectividad

Para establecer un método que permita reconocer si dos grafos son isomorfos se debe en primer lugar entender cómo se transforma la matriz de conectividad entre grafos isomórficos.

El caso más sencillo es si se tiene un grafo como el de la Figura 9 y se desean intercambiar los vértices 2 y 5.

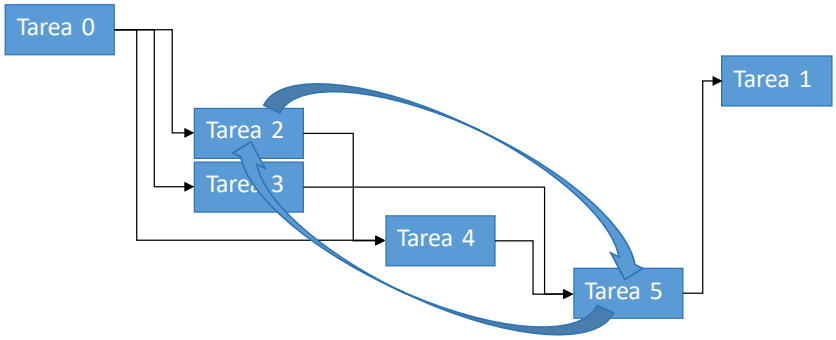


Figura 9 Intercambio de la numeración de dos vértices en un grafo

Como se puede observar en las siguientes matrices, los cambios en la matriz de conectividad consisten en primer lugar en un intercambio de las filas 2 y 5, y, en segundo lugar, en un intercambio de las columnas 2 y 5. Esto es equivalente a decir que el vértice 2 de la nueva matriz de conectividad pasa a tener los antecesores y sucesores del vértice 5 de la matriz original de conectividad, y viceversa.

$$C = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \end{pmatrix}; C^{intermedia} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix};$$

$$C' = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

En general, si se intercambian los vértices p y q de una matriz de conectividad C en un grafo de n vértices, aplican las siguientes fórmulas:

$$c'_{ip} = c_{iq} \; \forall \, i \mid (0 \leq i < n) \wedge (i \neq p, q)$$

$$c'_{iq} = c_{ip} \; \forall \, i \mid (0 \leq i < n) \wedge (i \neq p, q)$$

$$c'_{pi} = c_{qi} \; \forall \, i \mid (0 \leq i < n) \wedge (i \neq p, q)$$

$$c'_{qi} = c_{pi} \; \forall \, i \mid (0 \leq i < n) \wedge (i \neq p, q)$$

$$c'_{pp} = c_{qq}$$

$$c'_{pq} = c_{qp}$$

$$c'_{qq} = c_{pp}$$

$$c'_{qp} = c_{pq}$$

Es posible definir una matriz de transposición para ejecutar esta operación con dos multiplicaciones matriciales. En el ejemplo de la Figura 9, si se denomina W a la matriz de transposición:

$$W = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}; C^{intermedia} = W \times C = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Como se puede ver, multiplicando la matriz de transposición por el lado izquierdo de la matriz de conectividad se han intercambiado las filas 2 y 5 en $C^{intermedia}$. Si se multiplica $C^{intermedia}$ por el lado derecho por la transpuesta de W:

$$W^T = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}; C' = W \times C \times W^T = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Se obtendrá la nueva matriz de conectividad para los vértices intercambiados. Se puede definir la matriz de transposición de dos vértices p y q como:

$$w_{ij} = \begin{cases} 1 \; si \; i = j \; \forall \; (i, j \neq p) \wedge (i, j \neq q) \\ 1 \; si \; (i = p) \wedge (j = q) \\ 1 \; si \; (i = q) \wedge (j = p) \\ 0 \; en \; otro \; caso \end{cases}$$

La demostración es trivial. La multiplicación de $C^{intermedia} = W \times C$ se expresa como:

$$c_{ij}^{intermedia} = \sum_{k=0}^{n-1} w_{ik} \times c_{kj} \; \Rightarrow$$

$$\Rightarrow c_{ij}^{intermedia} = \begin{cases} c_{ij} \; si \; (i \neq p, q) \\ c_{qj} \; si \; (i = p) \\ c_{pj} \; si \; (i = q) \end{cases}$$

Análogamente, la multiplicación de $C' = (W \times C) \times W^T$ se expresa como:

$$c'_{ij} = \sum_{k=0}^{n-1} c_{ik}^{intermedia} \times w_{kj}^T = \sum_{k=0}^{n-1} c_{ik}^{intermedia} \times w_{jk} \; \Rightarrow$$

$$\Rightarrow c'_{ij} = \begin{cases} c_{ij} \; si \; (i \neq p, q) \wedge (j \neq p, q) \\ c_{qj} \; si \; (i = p) \wedge (j \neq p, q) \\ c_{pj} \; si \; (i = q) \wedge (j \neq p, q) \\ c_{iq} \; si \; (j = p) \wedge (i \neq p, q) \\ c_{ip} \; si \; (j = q) \wedge (i \neq p, q) \\ c_{qq} \; si \; (i = p) \wedge (j = p) \\ c_{pp} \; si \; (i = q) \wedge (j = q) \\ c_{pq} \; si \; (i = q) \wedge (j = p) \\ c_{qp} \; si \; (i = p) \wedge (j = q) \end{cases}$$

Existe una definición sencilla de la matriz de transposición para varios vértices simultáneos. La forma general de la matriz de transposición en tal caso es:

$$w_{ij} = \begin{cases} 1\ si\ intercambiamos\ el\ vértice\ j\ a\ la\ posición\ del\ vértice\ i \\ 0\ en\ otro\ caso \end{cases}$$

Esta expresión se deriva fácilmente si se concatenan varios intercambios sucesivos de parejas de vértices.

Finalmente es posible dar una definición más formal al isomorfismo de grafos. Dos grafos son isomorfos si y sólo si existe una matriz de transposición W tal que las matrices de conectividad C y C' de los grafos están relacionados por la fórmula $C' = W \times C \times W^T$.

3.3 FUNCIÓN DE DISTRIBUCIÓN ESTADÍSTICA DE WEIBULL

La función de distribución estadística descrita por Weibull (1951) es una de las más empleadas para modelar intervalos entre dos sucesos, por ejemplo, la vida útil de un equipo (Moubray, 1997).

La función de densidad de una variable aleatoria con la distribución de Weibull es:

$$f(x;k,\lambda) = \begin{cases} \frac{k}{\lambda}\left(\frac{x}{\lambda}\right)^{k-1} e^{-(x/\lambda)^k} si\ x \geq 0 \\ 0\ si\ x < 0 \end{cases}$$

$k > 0$ es el parámetro de forma y $\lambda > 0$ es el parámetro de escala.

La función de distribución de una variable aleatoria con la distribución de Weibull es:

$$F(x;k,\lambda) = \begin{cases} 1 - e^{-(x/\lambda)^k} si\ x \geq 0 \\ 0\ si\ x < 0 \end{cases}$$

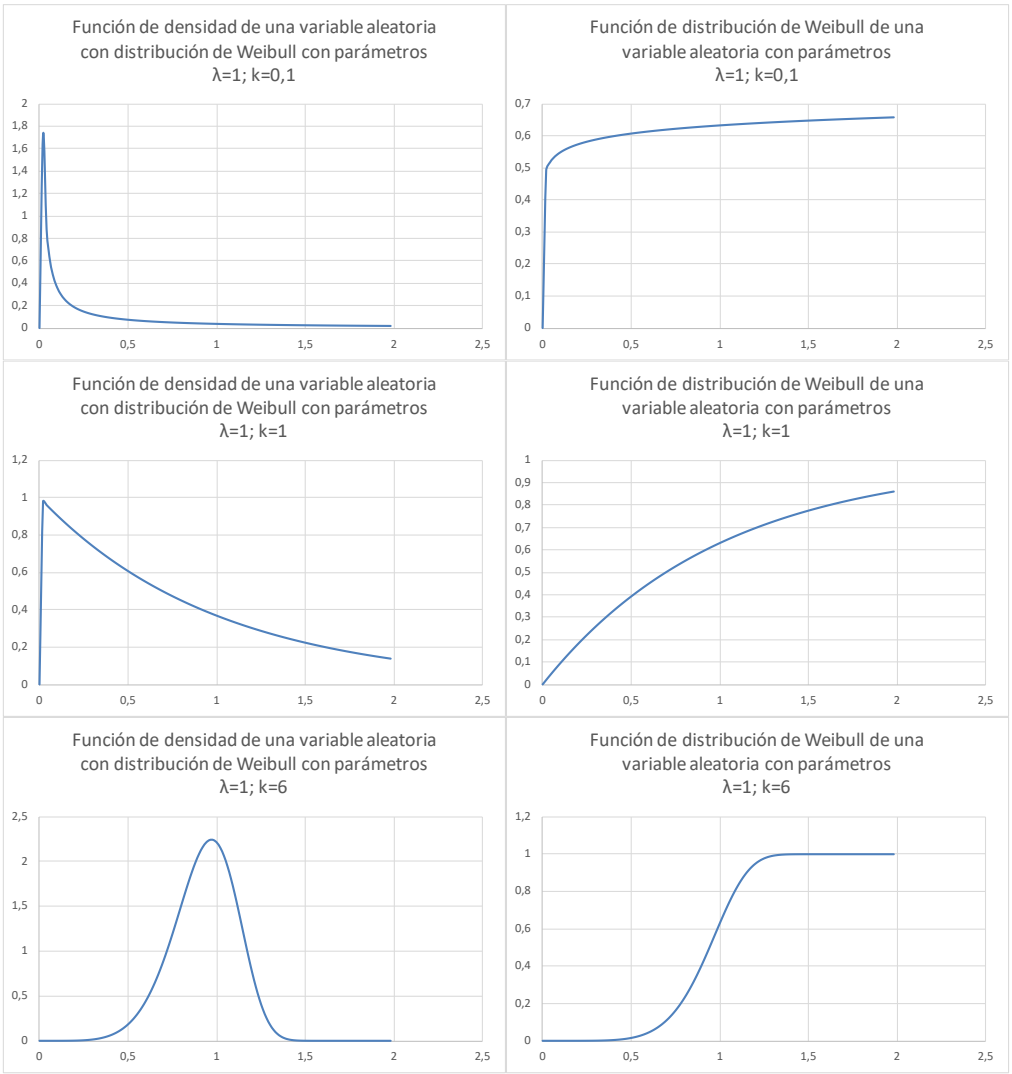


Figura 10 Funciones de densidad y de distribución de Weibull de una variable aleatoria para distintos valores del factor de forma k

3.4 LENGUAJE DE PROGRAMACIÓN DE LA APLICACIÓN INFORMÁTICA

El lenguaje de programación que se ha empleado en el desarrollo de la aplicación informática es *Java™ Platform Standard Edition 8*, de Oracle. Java es tanto un lenguaje de programación de alto nivel como una plataforma informática. Sus principales características son:

- Simplicidad
- Orientado al objeto
- Distribuido
- Dinámico
- Arquitectura neutral
- Portable
- Altas prestaciones
- Robusto y seguro

En el lenguaje de programación Java el código fuente se escribe en ficheros de texto con la extensión `.java`. Los ficheros fuente se compilan en ficheros `.class` por el compilador `javac`. Un fichero `.class` no contiene código ejecutable en ningún procesador. En su lugar contiene *bytecodes*, el código máquina de la máquina virtual Java (*Java Virtual Machine*, Java VM). La herramienta Java *launcher* corre la aplicación con una instancia de la Java VM. Se trata por lo tanto de un lenguaje de programación híbrido entre compilador e intérprete.

Se ha elegido Java para programar la aplicación informática básicamente debido a dos factores:

1. La abundancia de material de formación disponible en internet.
2. Se trata de un software de libre disposición que no precisa la adquisición de licencias.

La desventaja de Java es que no es un lenguaje de programación totalmente compilado, por lo que el código máquina generado no corre directamente sobre el ordenador y precisa de la intervención del intérprete Java *launcher* sobre la Java VM. Esto ralentiza sus tiempos de ejecución y reduce sus prestaciones.

Para el desarrollo y depurado de la aplicación informática se ha recurrido a la plataforma NetBeans IDE 8.0.2. NetBeans es una potente herramienta que permite el desarrollo de los diferentes ficheros fuente. Proporciona una revisión automática de la sintaxis de Java y herramientas avanzadas de depuración a nivel profesional.

3.5 PLAN DE MANTENIMIENTO EMPLEADO EN LOS CASOS DE ESTUDIO

El plan de mantenimiento y las estimaciones de duración de tareas empleadas para los casos estudiados en el capítulo “Aplicaciones”, página 101 y siguientes, son datos reales del mantenimiento de un tren de alta velocidad.

4. RESULTADOS Y DISCUSIÓN

"Me ha sucedido lo que a los glotones, que se arrojan sobre todas las viandas que se les presentan y no saborean ninguna. Antes de haber resuelto perfectamente la primera cuestión que se ha propuesto sobre la naturaleza de la justicia, he procurado indagar detenidamente si era vicio o virtud, habilidad o ignorancia. Otra cuestión nos ha salido al encuentro, a saber: si la injusticia es más ventajosa que la justicia, y no he podido menos de abandonar la primera y pasar a la segunda".

Platón, La República o el Estado, conclusión del Libro I.

Ninguna de las metodologías identificadas en el análisis del estado del arte cumple con los requisitos planteados por el problema que se desea resolver. Este hecho ha llevado al desarrollo de un novedoso algoritmo de secuenciación operativa del mantenimiento (*Operational Maintenance Sequencing Algorithm*, OMSA).

La solución propuesta en OMSA es un algoritmo heurístico auxiliado por operaciones algebraicas de la matriz de conectividad del grafo del flujograma.

El algoritmo heurístico se basa en el procedimiento adoptado por un planificador de mantenimiento. El planificador parte de las tareas inicial y final de la lista de tareas pendientes e inserta una a una el resto de las tareas en diferentes posiciones del flujograma. Para cada una de las posiciones de la nueva tarea, comprueba si el flujograma cumple todas las restricciones y, si es así, calcula la duración total de la estadía. El flujograma con la duración más corta se toma como base para insertar la siguiente tarea. Este procedimiento se repite sucesivamente con el resto de las tareas hasta que se completa la lista.

En realidad, un planificador experimentado no comprueba todas las posiciones posibles porque le es posible reconocer y descartar directamente posiciones claramente desfavorables. También es capaz de identificar las zonas en las que la nueva tarea tiene el menor impacto en la duración del flujograma. Esta visión global no es directamente implementable en una aplicación informática.

Como ya se ha visto en la revisión del estado del arte, esta heurística no ha sido empleada en el pasado.

El flujograma es un grafo en el que las tareas son los vértices y puede ser representado por su matriz de conectividad. La inserción de un nuevo vértice (o tarea) en el grafo (o flujograma) se representa por la inserción de una nueva columna y una nueva fila en la matriz de conectividad. Diferentes operaciones de multiplicación de la matriz de conectividad hacen posible que OMSA recorra el flujograma contando los tiempos de mantenimiento, identificando cambios de los estados de seguridad y bloqueando el tiempo necesario para sus cambios, asignando recursos a las tareas, identificando relaciones predecesor-sucesor, y contando los tiempos de desplazamiento de cada recurso individual.

Existen las siguientes relaciones de equivalencia entre flujograma, grafo y matriz de conectividad:

Flujograma ≡ Grafo ≡ Matriz de conectividad

Tarea i ≡ Vértice i ≡ Columna i + fila i

Relación predecesor-sucesor ij ≡ Borde ij ≡ Elemento c_{ij} de la matriz de conectividad

Estos usos de la matriz de conectividad para manipular el flujograma son novedosos y han sido desarrollados expresamente para implementar la heurística que soluciona este problema. La matriz de conectividad ha demostrado ser una herramienta versátil y efectiva en el análisis informatizado de grafos / flujogramas.

4.1 ALGORITMO CUASI-GENÉTICO

La experiencia en el mantenimiento de material rodante de alta velocidad indica que los planificadores emplean un método sistemático para secuenciar los flujogramas de las intervenciones. El primer problema es aprehender esta heurística. Tratando el tema con un planificador de mantenimiento se hizo evidente que la heurística empleada era modelable como un algoritmo genético sin cruces ("*crossovers*"), como se verá más adelante.

La implementación de esta heurística en OMSA consiste en iniciar con una tarea inicial (que se denominará tarea 0) y una tarea final (que se denominará tarea 1). OMSA inserta sucesivamente el resto de las tareas una a una. La introducción de una tarea inicia el equivalente de una generación (inicializar una población) en un algoritmo genético propiamente dicho (véase Figura 11).

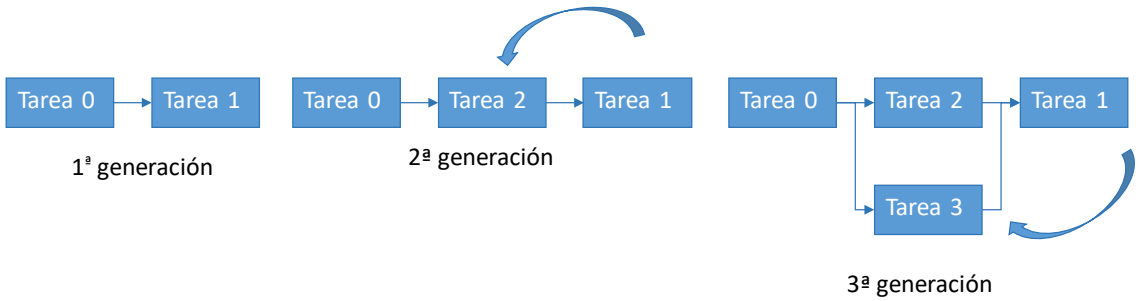


Figura 11 Sucesivas generaciones en OMSA

Cada nueva tarea se inserta en varias posiciones del grafo / flujograma, véase Figura 12. Cada inserción equivale a un genotipo (una mutación) en un algoritmo genético, aunque en este caso se prescinde del elemento aleatorio típico. OMSA calcula la duración del flujograma en curso (equivalente a la función de aptitud o "*fitness function*") considerando los tiempos de desplazamiento a las localizaciones, el tiempo para cambiar los estados de seguridad, y si los recursos disponibles son suficientes. Si el flujograma requiere demasiados recursos o si hay dos tareas en paralelo con estados de seguridad incompatibles, o si el recurso asignado no tiene suficiente tiempo para llegar a la siguiente localización, o si no se cumple alguna relación predecesor-sucesor, se descarta el flujograma. La mutación viable con la duración más corta será usada como base progenitora de la siguiente generación. Este proceso se repite hasta que se hayan procesado todas las tareas pendientes o hasta que se haya alcanzado el máximo tiempo disponible.

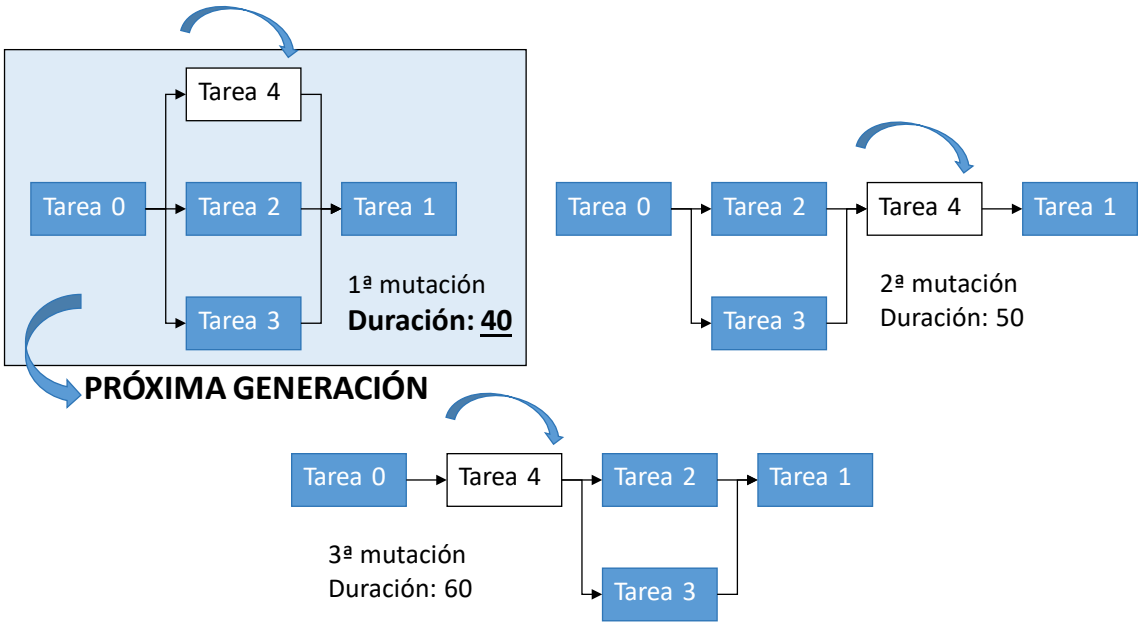


Figura 12 Sucesivas mutaciones en OMSA

Esta heurística favorece flujogramas con pocos cambios de estados de seguridad y de localización y con tantas tareas en paralelo como sea posible dados los recursos disponibles. Una mutación con frecuentes conexiones y desconexiones de la catenaria (25 kV AC 50 Hz) se verá fuertemente penalizada en la duración total y las nuevas tareas insertadas terminarán ancladas en áreas con estados de seguridad iguales o compatibles y con localizaciones cercanas. El resultado final será un agrupamiento de tareas semejante a la heurística del planificador.

OMSA no es un algoritmo genético desde un estricto punto de vista. Incluye generaciones, mutaciones y una función de aptitud, pero no tiene entrecruzamientos entre fenotipos como debería tener un algoritmo genético clásico de acuerdo con Kramer (2017) y las mutaciones no son aleatorias sino sistemáticas. Sin embargo, las analogías con un algoritmo genético facilitan su explicación.

4.2 TÉCNICAS DE ANÁLISIS Y MANIPULACIÓN DE GRAFOS

Existe una relación biunívoca entre un grafo y su matriz de conectividad. Esta relación biunívoca posibilita que cualquier operación imaginable del grafo se refleje en una operación equivalente de la matriz de conectividad. Los grafos son representaciones gráficas de estructuras y conceptos que pueden ser fácilmente comprendidos por el cerebro humano. Las matrices son representaciones numéricas de las mismas estructuras y conceptos pero que pueden ser manipuladas fácilmente por una aplicación informática.

El objetivo de este capítulo es presentar una serie de técnicas algebraicas que traducen operaciones como "encuentra todos los bucles cerrados en este grafo" o "inserta un nuevo vértice i paralelo al vértice j" en operaciones algebraicas de la matriz de conectividad.

Las técnicas aquí presentadas han sido implementadas con éxito para desarrollar OMSA. La técnica más compleja desarrollada en esta tesis ha sido el método descrito en el apartado "Algoritmo de Reordenación del Flujograma".

4.2.1 INSERCIÓN DE NUEVOS VÉRTICES EN EL GRAFO

Esta sección presenta las técnicas para insertar vértices en un grafo. La matriz de conectividad permite que el software inserte nuevos vértices (tareas) como sea requerido para construir el flujograma y así secuenciar las tareas de mantenimiento.

Un nuevo vértice insertado p se representa por una columna adicional en la posición p más una fila adicional en la posición p de la matriz de conectividad.

INSERCIÓN DE VÉRTICE PARALELO

El método para insertar un nuevo vértice p paralelo a un vértice q en el grafo puede realizarse de la siguiente manera:

$$c_{ip} = c_{iq} \; \forall \, i \mid 0 \leq i < p$$

$$c_{pi} = c_{qi} \; \forall \, i \mid 0 \leq i < p$$

$$c_{pp} = 0$$

Estas fórmulas describen el hecho de que los vértices paralelos tienen los mismos predecesores y sucesores.

$$C_3 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}; C_4 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

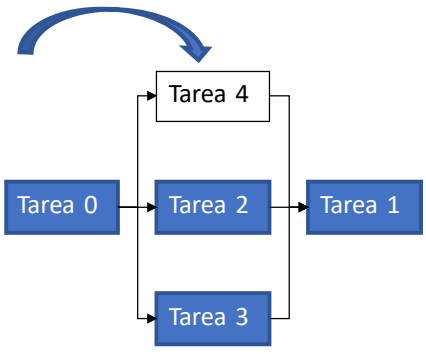


Figura 13 Inserción de un vértice paralelo (tarea 4 paralela a las tareas 2 y 3)

INSERCIÓN DE VÉRTICE PREDECESOR

El método para insertar un nuevo vértice p como predecesor directo de un vértice q en el grafo puede realizarse de la siguiente manera:

$$c_{pi} = c_{qi} \; \forall \, i \mid 0 \leq i < p$$

$$c_{ip} = \begin{cases} 1 \; si \; i = q \\ 0 \; en \; otro \; caso \end{cases}$$

$$c_{qi} = \begin{cases} 1 \; si \; i = p \\ 0 \; en \; otro \; caso \end{cases}$$

$$c_{pp} = 0$$

Estas fórmulas describen el hecho de que un vértice predecesor directo p recibe los enlaces desde los vértices que estaban enlazados al vértice q y que el vértice q sólo tiene al vértice p como predecesor.

$$C_3 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}; C_4 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \end{pmatrix}$$

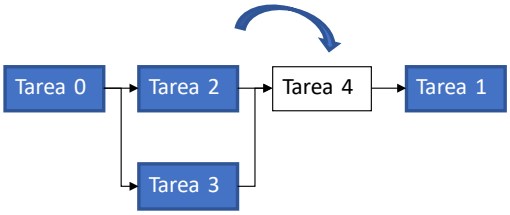


Figura 14 Inserción de un vértice predecesor (tarea 4 predecesora de tarea 1)

INSERCIÓN DE VÉRTICE SUCESOR

El método para insertar un nuevo vértice p como sucesor directo de un vértice q en el grafo puede realizarse de la siguiente manera:

$$c_{ip} = c_{iq} \; \forall \, i \mid 0 \leq i < p$$

$$c_{pi} = \begin{cases} 1 \; si \; i = q \\ 0 \; en \; otro \; caso \end{cases}$$

$$c_{iq} = \begin{cases} 1 \; si \; i = p \\ 0 \; en \; otro \; caso \end{cases}$$

$$c_{pp} = 0$$

Estas fórmulas describen el hecho de que un vértice sucesor p recibe los vértices sucesores que estaban enlazados al vértice q y que el vértice q tiene sólo al vértice p como sucesor.

$$C_3 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}; C_4 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

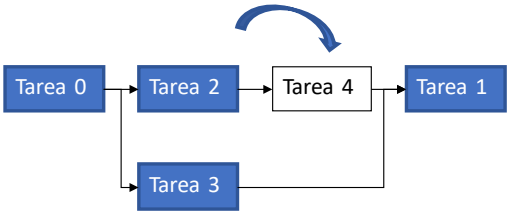


Figura 15 Inserción de un vértice sucesor (tarea 4 sucesora de tarea 2)

INSERCIÓN DE VÉRTICE PARALELO AL CAMINO ENTRE DOS VÉRTICES

El método para insertar un nuevo vértice p paralelo al camino comenzando con el vértice q y terminando con el vértice r en el grafo puede realizarse de la siguiente manera:

$$c_{pi} = c_{qi} \; \forall \; i \mid 0 \leq i < p$$

$$c_{ip} = c_{ir} \; \forall \; i \mid 0 \leq i < p$$

$$c_{pp} = 0$$

Estas fórmulas describen el hecho de que el vértice p recibe los vértices predecesores del vértice q y los vértices sucesores del vértice r.

$$C_4 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \end{pmatrix}; C_5 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

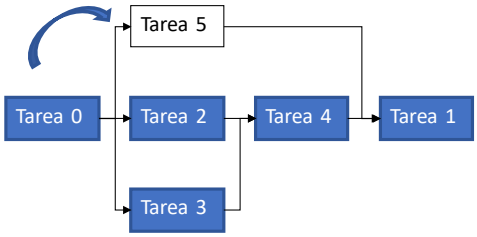


Figura 16 Inserción de un vértice paralelo al camino entre otros dos vértices (tarea 5 paralela al camino desde la tarea 2 a la tarea 4)

4.2.2 MATRICES DE AVANCE Y DE RETROCESO

Las matrices de avance y de retroceso son nuevos conceptos desarrollados en esta tesis para su aplicación en OMSA. En algunos casos, p. ej. para calcular la duración total de un flujograma, es necesario recorrer sistemáticamente todos los vértices y caminos de un grafo. Con este propósito se comienza definiendo la matriz nodo N^0 del vértice 0 como sigue:

$$n_{ij}^0 = \begin{cases} 1 \; si \; i = j = 0 \\ 0 \; en \; otro \; caso \end{cases}$$

Si se multiplicase la matriz nodo del vértice 0 por la matriz de conectividad por el lado izquierdo se obtendría una matriz cuya primera columna izquierda contendría los sucesores del vértice 0 y por ello representaría el primer paso del grafo (o flujograma). Ésta sería la primera matriz de avance o matriz de avance de orden uno y puede representarse como sigue:

$$S^1 = C \times N^0$$

Multiplicando la matriz de avance de orden uno por la matriz de conectividad por el lado izquierdo se obtendrá una matriz cuya primera columna izquierda contiene los sucesores de los sucesores del vértice 0, es decir, los vértices pertenecientes al segundo paso del grafo (o flujograma). Si un vértice se alcanza a través de diversos caminos de igual longitud, la matriz de avance en la que aparece este vértice mostrará el número de caminos que llegan a él. Si un vértice es alcanzado por diferentes caminos de diferente longitud el vértice aparecerá en las matrices de avance de igual orden a las longitudes de los caminos.

$$N^0 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}; C = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}; S^1 = C \times N^0 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

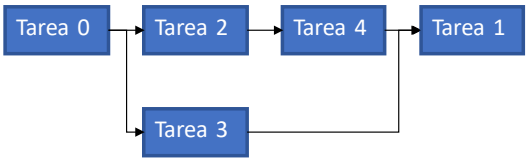


Figura 17 Ejemplo de matriz de avance de primer orden

Se representa la matriz de avance de orden m de la siguiente manera:

$$S^m = C^m \times N^0$$

El método de las matrices de avance permite que OMSA discurra por todos los vértices y caminos del grafo. Cuando todos los elementos de la matriz de avance sean nulos no habrá más vértices sucesores y se habrá llegado al final del grafo.

Si un grafo de n vértices no tiene ningún bucle cerrado el camino más largo posible será una serie de n vértices y la matriz de avance de orden n siempre será nula.

Las matrices de retroceso también son un nuevo concepto. Si se multiplica la matriz de nodo del vértice 1 (último vértice del flujograma) por la matriz de conectividad por el lado derecho se obtendrá una matriz cuya segunda fila contando desde arriba contendrá los predecesores del vértice 1 y por lo tanto representará el paso previo del grafo / flujograma. La matriz de retroceso de orden uno se representa de la siguiente manera:

$$R^1 = N^1 \times C$$

Aplicando un razonamiento análogo, la matriz de retroceso de orden m se representa de la siguiente manera:

$$R^m = N^1 \times C^m$$

El método de las matrices de retroceso permite al algoritmo recorrer en sentido inverso todos los vértices y caminos del grafo. Cuando todos los elementos de la matriz de retroceso son nulos el algoritmo ha llegado al inicio del grafo.

Si un grafo de n vértices no tiene ningún bucle cerrado el camino más largo posible será una serie de n vértices y la matriz de retroceso de orden n siempre será nula.

4.2.3 RECORRIDO SISTEMÁTICO DE GRAFOS

El siguiente sencillo ejemplo tiene como objetivo ilustrar el uso de las matrices de avance en una nueva técnica sin recursividad para recorrer todos los caminos de un grafo y calcular la duración

del flujograma correspondiente. Se pueden definir las siguientes variables para el cálculo de la duración del flujograma:

$$d_m = duración\ de\ la\ tarea\ 'm'$$
$$t_{im} = tiempo\ de\ inicio\ de\ la\ tarea\ 'm'$$
$$t_{fm} = tiempo\ final\ de\ la\ tarea\ 'm'$$

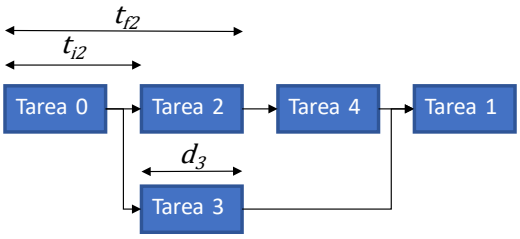


Figura 18 Grafo de flujograma, ilustración de variables para el cálculo de su duración

Primer paso, matriz de avance de orden uno:

$$S^1 = C \times N^0 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix} \times \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

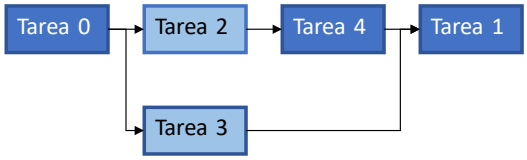


Figura 19 La matriz de avance de orden uno muestra que los vértices sucesores del vértice 0 son el 2 y el 3

Los tiempos calculados son:

$$t_{i0} = 0$$
$$t_{f0} = d_0$$
$$t_{i2} = t_{f0}$$
$$t_{f2} = t_{i2} + d_2$$
$$t_{i3} = t_{f0}$$
$$t_{f3} = t_{i3} + d_3$$

Segundo paso, matriz de avance de orden dos:

$$S^2 = C \times S^1 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix} \times \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

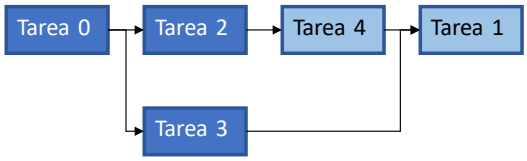


Figura 20 La matriz de avance de orden dos muestra que los vértices sucesores después del 2 y el 3 son el 1 y el 4

Los tiempos calculados son:

$$t_{i4} = t_{f2}$$
$$t_{f4} = t_{i4} + d_4$$
$$t_{i1} = t_{f3}$$
$$t_{f1} = t_{i1} + d_1$$

Tercer paso, matriz de avance de orden tres:

$$S^3 = C \times S^2 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix} \times \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

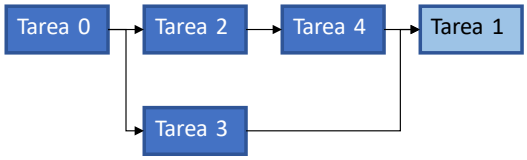


Figura 21 La matriz de avance de orden tres muestra que el vértice sucesor después de los vértices 1 y 4 es sólo el vértice 1

Los tiempos calculados son:

$$t_{i1} = máximo(t_{i1}\ previo\ , t_{f4})$$
$$t_{f1} = t_{i1} + d_1$$

La duración total del flujograma es el tiempo final del vértice 1: t_{f1}.

4.2.4 PRUEBA DE SUCESIÓN / PRECEDENCIA

Las matrices de avance y retroceso poseen características que pueden aprovecharse para identificar si un vértice p es predecesor de otro vértice q. Se parte de la matriz de nodo N^p del vértice p definida como sigue:

$$n_{ij}^p = \begin{cases} 1\ si\ i = j = p \\ 0\ en\ otro\ caso \end{cases}$$

Si se multiplica la matriz de nodo del vértice p por la matriz de conectividad por el lado izquierdo se obtendrá la primera matriz de avance que parte del vértice p y cuya columna p contiene el primer grupo de vértices sucesores del vértice p.

$$S^{p1} = C \times N^p$$

$s_{qp}^{p1} \neq 0$ significa que p es un predecesor directo de q. Aplicando la misma lógica, si se llegase a un $s_{qp}^{pm} \neq 0$ significaría que p es un predecesor de q tras pasar por $m - 1$ vértices.

Para averiguar si un vértice p es sucesor de otro vértice q se usarían las matrices de retroceso en vez de las matrices de avance. También es posible emplear igualmente las matrices de avance comenzando desde la matriz nodo N^q.

4.2.5 PROXIMIDAD DE UN VÉRTICE o A LOS VÉRTICES INICIAL Y AL FINAL DEL GRAFO

En algunos casos se hace necesario medir la proximidad de un determinado vértice o al vértice inicial o al vértice final del grafo teniendo en cuenta no sólo el número máximo o mínimo de vértices intermedios que hay que recorrer hasta llegar a los extremos del grafo sino también considerando la complejidad de las ramificaciones.

A tal efecto, se define como avance cuadrático de grado m del vértice o a:

$$aq_o^m = \sum_{i=0}^{n-1} (k_{io}^m)^2$$

Como se ha visto previamente, k_{io}^m es el número de caminos partiendo del vértice o y terminando en el vértice i a través de 0,1,2…, m-1 vértices intermedios. Elevando este término al cuadrado dentro del sumatorio se consiguen valores superiores de aq_o^m para vértices o que, manteniendo la misma cantidad de vértices sucesores, tienen más caminos que convergen a los vértices i. El máximo valor posible de m es n-1. El valor de aq_o^m se mantiene constante a partir del momento en que todos los caminos que parten de o han alcanzado el vértice final. Los valores máximos de aq_o^m los alcanza el vértice inicial del grafo, mientras que los valores mínimos proceden del vértice final. La serie de avances cuadráticos del vértice o es:

$$AQ_o = \{aq_o^0, aq_o^1, \dots, aq_o^{n-1}\}$$

La serie de avances cuadráticos AQ_o de los vértices puede usarse para ordenar los vértices según su proximidad a los extremos del grafo.

Análogamente, se define como retroceso cuadrático de grado m del vértice o a:

$$rq_o^m = \sum_{i=0}^{n-1} (k_{oi}^m)^2$$

Como se ha visto previamente, k_{oi}^m es el número de caminos partiendo del vértice i y terminando en el vértice o a través de 0,1,2…, m-1 vértices intermedios. Elevando este término al cuadrado dentro del sumatorio se consiguen valores superiores de rq_o^m para vértices o que, manteniendo la misma cantidad de vértices predecesores, tienen más caminos que convergen a los vértices i. El máximo valor posible de m es n-1. El valor de rq_o^m se mantiene constante a partir del momento en que todos los caminos que llegan a o parten desde el vértice inicial. Los valores máximos de rq_o^m los alcanza el vértice final, mientras que los valores mínimos proceden del vértice inicial. La serie de retrocesos cuadráticos del vértice o es:

$$RQ_o = \{rq_o^0, rq_o^1, \dots, rq_o^{n-1}\}$$

La serie de retrocesos cuadráticos RQ_o de los vértices puede usarse para ordenar vértices cuyas series de avances cuadráticos AQ_o sean iguales.

Como ejemplo se calcularán las series de avances cuadráticos y de retrocesos cuadráticos de los vértices del siguiente grafo:

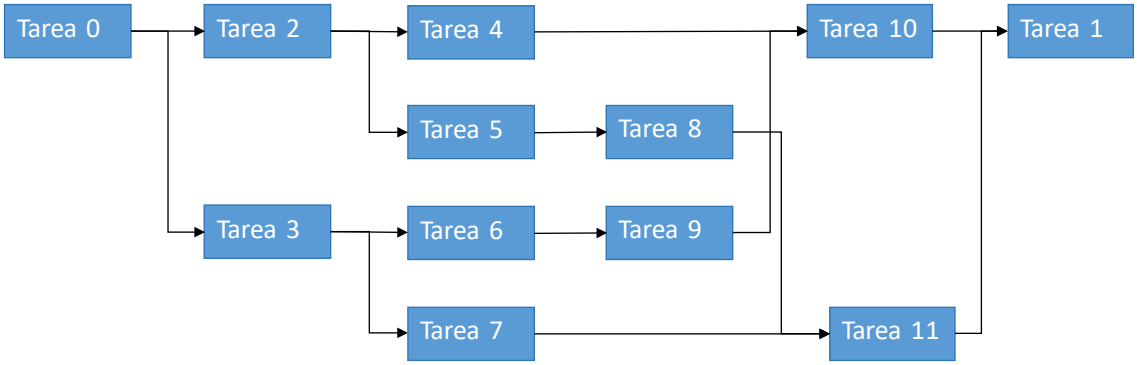


Figura 22 Grafo para ejemplo de series cuadráticas

La matriz de conectividad (que es igual a la matriz de conectividad acumulativa de primer orden) es:

$$C = K^1 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \end{pmatrix}; \begin{matrix} rq_0^1 = 0 \\ rq_1^1 = 2 \\ rq_2^1 = 1 \\ rq_3^1 = 1 \\ rq_4^1 = 1 \\ rq_5^1 = 1 \\ rq_6^1 = 1 \\ rq_7^1 = 1 \\ rq_8^1 = 1 \\ rq_9^1 = 1 \\ rq_{10}^1 = 2 \\ rq_{11}^1 = 2 \end{matrix}$$

La matriz de conectividad acumulativa de segundo orden es:

$$K^2 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \end{pmatrix}; \begin{matrix} rq_0^2 = 0 \\ rq_1^2 = 6 \\ rq_2^2 = 1 \\ rq_3^2 = 1 \\ rq_4^2 = 2 \\ rq_5^2 = 2 \\ rq_6^2 = 2 \\ rq_7^2 = 2 \\ rq_8^2 = 2 \\ rq_9^2 = 2 \\ rq_{10}^2 = 4 \\ rq_{11}^2 = 4 \end{matrix}$$

La matriz de conectividad acumulativa de tercer orden es:

$$K^3 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \end{pmatrix}; \begin{matrix} rq_0^3 = 0 \\ rq_1^3 = 10 \\ rq_2^3 = 1 \\ rq_3^3 = 1 \\ rq_4^3 = 2 \\ rq_5^3 = 2 \\ rq_6^3 = 2 \\ rq_7^3 = 2 \\ rq_8^3 = 3 \\ rq_9^3 = 3 \\ rq_{10}^3 = 6 \\ rq_{11}^3 = 6 \end{matrix}$$

La matriz de conectividad acumulativa de cuarto orden es:

$$K^4 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 2 & 2 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 2 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \end{pmatrix}; \begin{matrix} rq_0^4 = 0 \\ rq_1^4 = 20 \\ rq_2^4 = 1 \\ rq_3^4 = 1 \\ rq_4^4 = 2 \\ rq_5^4 = 2 \\ rq_6^4 = 2 \\ rq_7^4 = 2 \\ rq_8^4 = 3 \\ rq_9^4 = 3 \\ rq_{10}^4 = 9 \\ rq_{11}^4 = 9 \end{matrix}$$

La matriz de conectividad acumulativa de quinto orden es:

$$K^5 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 0 & 2 & 2 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 2 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \end{pmatrix}; \begin{matrix} rq_0^5 = 0 \\ rq_1^5 = 32 \\ rq_2^5 = 1 \\ rq_3^5 = 1 \\ rq_4^5 = 2 \\ rq_5^5 = 2 \\ rq_6^5 = 2 \\ rq_7^5 = 2 \\ rq_8^5 = 3 \\ rq_9^5 = 3 \\ rq_{10}^5 = 9 \\ rq_{11}^5 = 9 \end{matrix} \quad \cdots \quad \begin{matrix} rq_0^{11} = 0 \\ rq_1^{11} = 32 \\ rq_2^{11} = 1 \\ rq_3^{11} = 1 \\ rq_4^{11} = 2 \\ rq_5^{11} = 2 \\ rq_6^{11} = 2 \\ rq_7^{11} = 2 \\ rq_8^{11} = 3 \\ rq_9^{11} = 3 \\ rq_{10}^{11} = 9 \\ rq_{11}^{11} = 9 \end{matrix}$$

Las siguientes matrices de conectividad acumulativa de orden superior son iguales a K^5 puesto que del vértice inicial al vértice final sólo hay cinco pasos.

Así pues, las series de retrocesos cuadráticos RQ_i de los vértices son:

$$RQ_0 = \{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\}$$

$$RQ_1 = \{2, 6, 10, 20, 32, 32, 32, 32, 32, 32, 32, 32\}$$

$$RQ_2 = \{1, 1, 1, 1, 1, 1, 1\ 1, 1, 1, 1, 1\}$$

$$RQ_3 = \{1, 1, 1, 1, 1, 1, 1\ 1, 1, 1, 1, 1\}$$

$$RQ_4 = \{1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2\}$$

$$RQ_5 = \{1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2\}$$

$$RQ_6 = \{1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2\}$$

$$RQ_7 = \{1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2\}$$

$$RQ_8 = \{1, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3\}$$

$$RQ_9 = \{1, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3\}$$

$$RQ_{10} = \{2, 4, 6, 9, 9, 9, 9, 9, 9, 9, 9, 9\}$$

$$RQ_{11} = \{2, 4, 6, 9, 9, 9, 9, 9, 9, 9, 9, 9\}$$

Si no hubiera más información que las series de retrocesos cuadráticos se sabría inmediatamente:

- que el vértice inicial es el cero porque no tiene predecesores
- que el vértice final es el uno porque es el que tiene más predecesores
- que los vértices 10 y 11 son los más cercanos al vértice final
- que los vértices 2 y 3 son los más cercanos al vértice inicial
- que los vértices 4, 5, 6 y 7 forman un grupo que se encuentra más cercano del vértice inicial que el grupo formado por los vértices 8 y 9

Si se calculan las series de avances cuadráticos se obtendrán lo siguientes valores:

$$AQ_0 = \{2, 6, 10, 20, 32, 32, 32, 32, 32, 32, 32, 32\}$$

$$AQ_1 = \{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\}$$

$$AQ_2 = \{2, 4, 6, 9, 9, 9, 9, 9, 9, 9, 9, 9\}$$

$$AQ_3 = \{2, 4, 6, 9, 9, 9, 9, 9, 9, 9, 9, 9\}$$

$$AQ_4 = \{1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2\}$$

$$AQ_5 = \{1, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3\}$$

$$AQ_6 = \{1, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3\}$$

$$AQ_7 = \{1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2\}$$

$$AQ_8 = \{1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2\}$$

$$AQ_9 = \{1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2\}$$

$$AQ_{10} = \{1, 1, 1, 1, 1, 1, 1\ 1, 1, 1, 1, 1\}$$

$$AQ_{11} = \{1, 1, 1, 1, 1, 1, 1\ 1, 1, 1, 1, 1\}$$

Las series de avances cuadráticos permiten ampliar las afirmaciones dadas por las series de retrocesos cuadráticos:

- Si bien los vértices 4, 5, 6 y 7 están a la misma distancia del vértice inicial, los vértices 4 y 7 están más cercanos del vértice final que los vértices 5 y 6

Este método ha demostrado su utilidad en el desarrollo de OMSA para ordenar los vértices previamente al algoritmo de reordenación del flujograma (véase a partir de la página 56).

4.2.6 ALGORITMO DE REORDENACIÓN DEL FLUJOGRAMA

El grafo del flujograma calculado por OMSA no está ordenado de manera que pueda ser fácilmente comprendido por una persona. Los vértices (tareas) se van numerando conforme aparecen en la lista de tareas pendientes y mantienen su numeración durante el procesamiento del algoritmo. Debido a ello, el flujograma resultante no presenta las tareas por orden cronológico ni predecesor-sucesor y se hace necesario reordenar los vértices adecuadamente para que el flujograma resulte más inteligible a una persona.

En el flujograma ordenado todas las tareas predecesoras de una tarea dada se encuentran por encima y las tareas inmediatamente predecesoras se encuentran lo más próximas posible.

En la Figura 23 se presenta un ejemplo de un grafo desordenado, tal como es calculado por OMSA con su matriz de conectividad, y del grafo isomórfico ordenado con su nueva matriz de conectividad. Obsérvese que la matriz de conectividad varia notablemente.

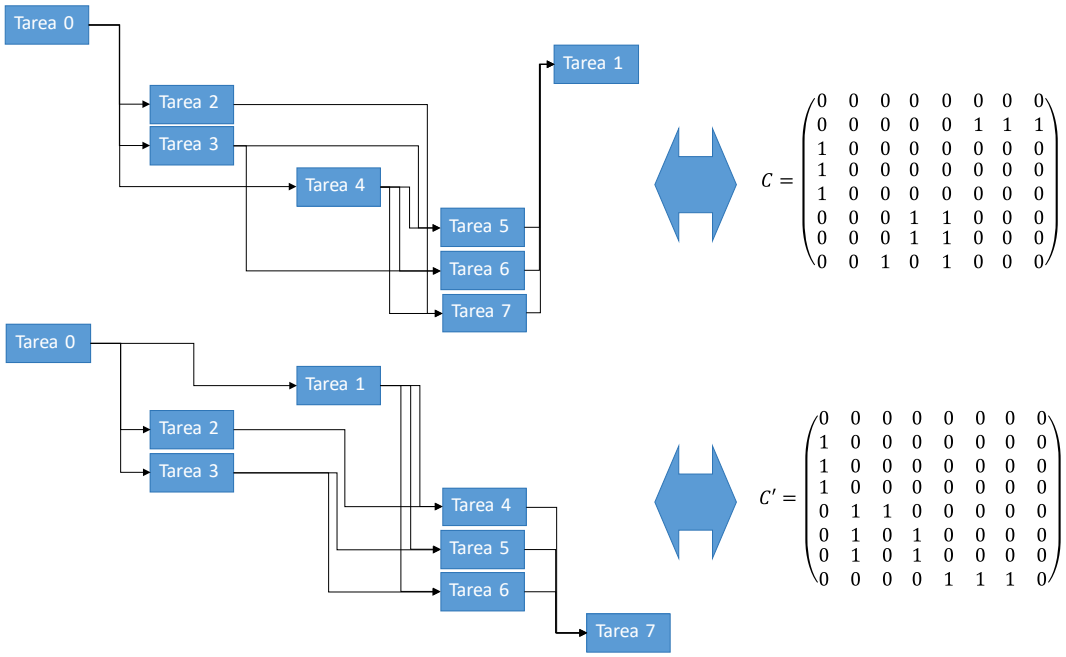


Figura 23 Ejemplo de un grafo desordenado y de su grafo isomórfico ordenado

Se puede comprobar fácilmente que la matriz de intercambio entre las dos matrices de conectividad del ejemplo de la Figura 23 es:

$$W = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

A continuación, se presentará el algoritmo de reordenación del flujograma en los siguientes pasos:

- Pseudo-código del algoritmo de reordenación del flujograma
- Aplicación del algoritmo de reordenación del flujograma a un grafo sencillo
- Implementación del algoritmo aplicando matrices de avance y matrices de transposición de vértices
- Primera mejora del algoritmo aplicando la prueba de sucesión / precedencia
- Segunda mejora del algoritmo aplicando las series de avances y retrocesos cuadráticos de los vértices para realizar una reordenación previa

PSEUDO-CÓDIGO DEL ALGORITMO DE REORDENACIÓN DEL FLUJOGRAMA

El algoritmo de reordenación del flujograma presenta un pseudo-código relativamente simple (consúltese la Figura 27 para mejor comprensión de los términos):

```
REPEAT {
     identificar vérticePrevio del grafo;
     identificar siguienteVértice del grafo;
     identificar últimoVértice directamente posterior a vérticePrevio y ya
     reordenado;
     IF (siguienteVértice no está a continuación de últimoVértice) {
          mover siguienteVértice a continuación de últimoVertice
     }
}
WHILE (vértice final alcanzado por todos los caminos posibles)
```

APLICACIÓN DEL ALGORITMO DE REORDENACIÓN DEL FLUJOGRAMA A UN CASO SENCILLO

Se aplica el algoritmo de la mano de un sencillo ejemplo consistente en un grafo de siete vértices numerados del 0 al 6:

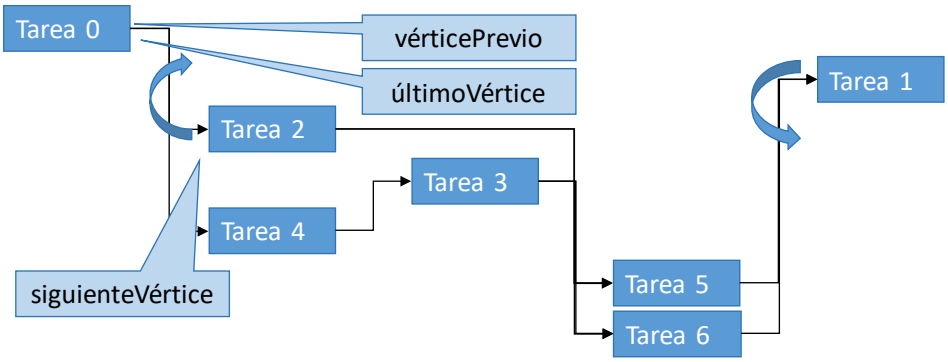


Figura 24 Grafo inicial, primera reordenación es subir la tarea 2 a continuación de la tarea 0

En la Figura 24 se indica que el vérticePrevio es el 0 puesto que el algoritmo se encuentra en su primer paso. El siguienteVértice que se va a procesar por ser sucesor directo del vérticePrevio 0 es el 2. Como el siguiente vértice de la lista después del vérticePrevio 0 es el 1 y no es un sucesor directo, el últimoVértice es también el vérticePrevio 0.

El algoritmo indica que se debe subir el siguienteVértice 2 directamente a continuación del últimoVértice 0 y rodar el resto de vértices intermedios una posición hacia abajo.

Nótese que la numeración de los vértices varía en cada paso del algoritmo.

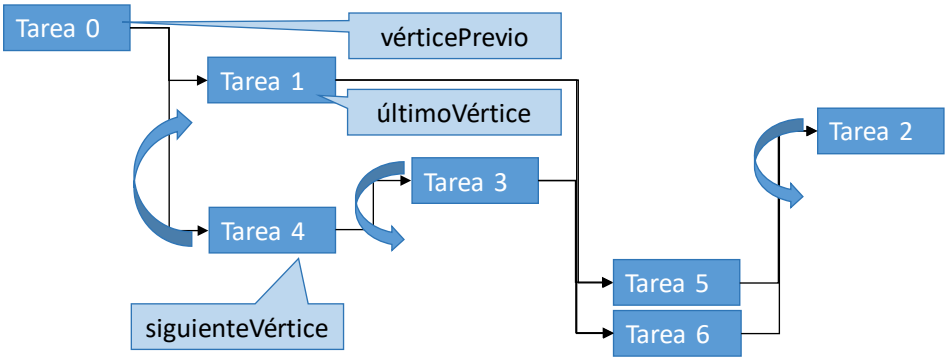


Figura 25 Segundo paso del algoritmo de reordenación

En la Figura 25 se mantiene el vérticePrevio 0 puesto que todavía no se han reordenado todos sus vértices sucesores. El últimoVértice pasa a ser 1 y siguienteVértice es 4.

El algoritmo indica que se debe subir el últimoVértice 4 directamente a continuación del últimoVértice 1 y bajar una posición los vértices 2 y 3.

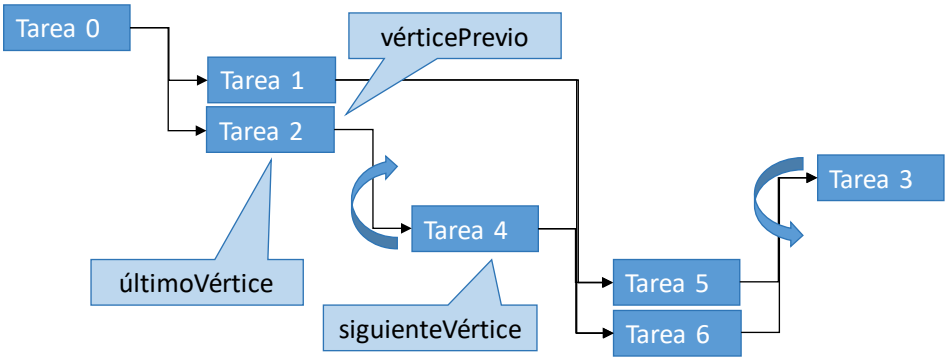


Figura 26 Tercer paso del algoritmo de reordenación

En la Figura 26 el nuevo vérticePrevio es la tarea 2 puesto que todos los vértices sucesores de 0 ya han sido reordenados y la tarea 2 es la siguiente que aparece usando el método de matrices de avance (véase "Implementación del Algoritmo Aplicando Matrices de Avance y Matrices de Transposición de Vértices", página 48 y siguientes). El siguienteVértice es 4 y el últimoVértice es 2.

El algoritmo indica que se debe subir el siguienteVértice 4 directamente a continuación del últimoVértice 2 y bajar una posición el vértice 3.

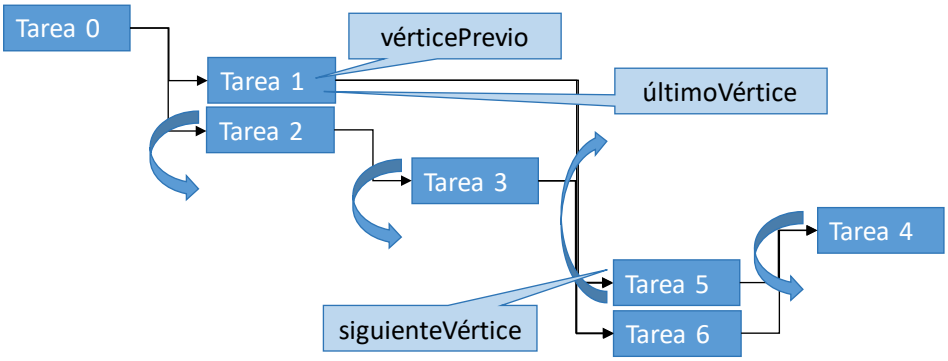


Figura 27 Cuarto paso del algoritmo de reordenación

En la Figura 27 el nuevo vérticePrevio es la tarea 1. El siguienteVértice es 5 y el últimoVértice es 1. El algoritmo indica que se debe subir el siguienteVértice 5 a continuación del últimoVértice 1 y bajar una posición los vértices 2, 3 y 4.

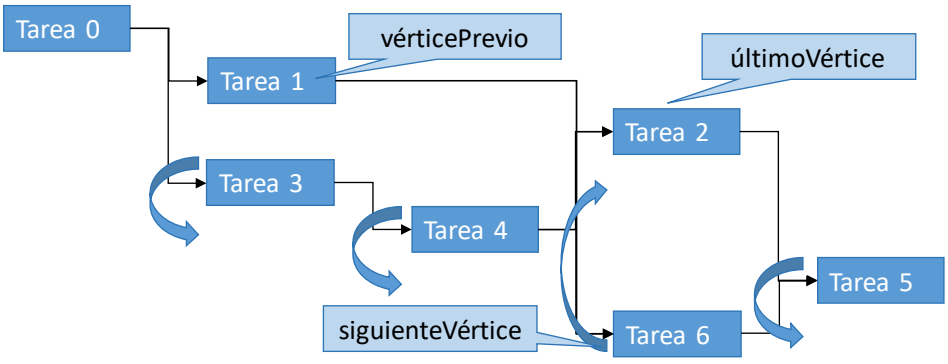


Figura 28 Quinto paso del algoritmo de reordenación

En la Figura 28 el vérticePrevio sigue siendo la tarea 1. El siguienteVértice ha pasado a ser el 6 y el últimoVértice es la tarea 2, recién reordenada en el paso anterior. El algoritmo indica que se debe subir el siguienteVértice 6 a continuación del últimoVértice 2 y bajar una posición los vértices 3, 4 y 5.

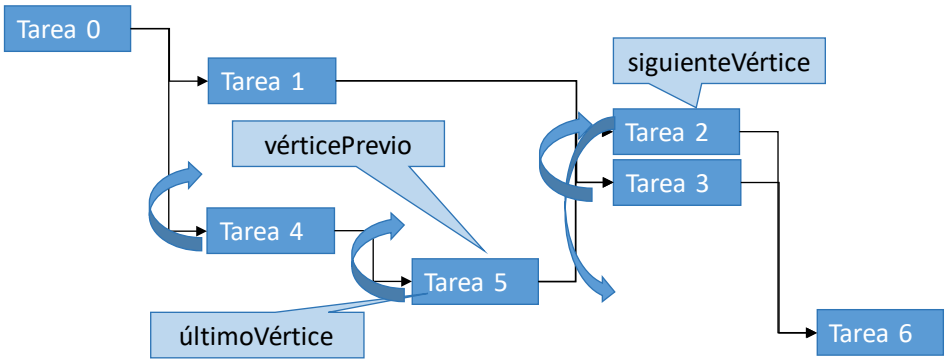


Figura 29 Sexto paso del algoritmo de reordenación

En la Figura 29 el vérticePrevio pasa a ser la tarea 5. El siguienteVértice es el 2 y el últimoVértice es el 5. El algoritmo indica que se debe bajar el siguienteVértice 2 a continuación del últimoVértice 5 y subir una posición los vértices 3, 4 y 5.

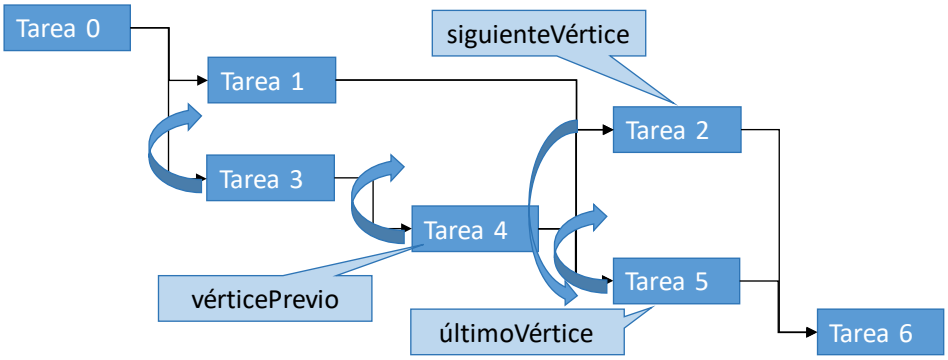


Figura 30 Séptimo paso del algoritmo de reordenación

En la Figura 30 el vérticePrevio es la tarea 4. El siguienteVértice es el 2 y el últimoVértice es el 5. El algoritmo indica que se debe bajar el siguienteVértice 2 a continuación del últimoVértice 5 y subir una posición los vértices 3, 4 y 5.

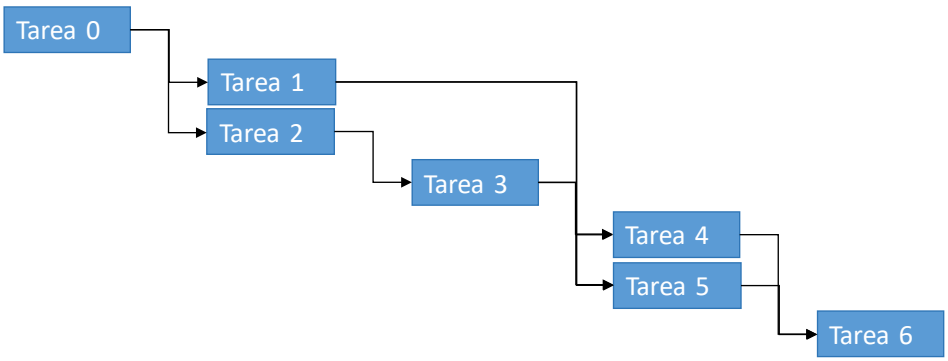


Figura 31 Último paso del algoritmo de reordenación

En la Figura 31 se ha llegado a la forma final del grafo reordenado. Al aplicar el algoritmo con últimoVértice igual a 4 y a 6 el resultado será que no es necesaria ninguna reordenación.

IMPLEMENTACIÓN DEL ALGORITMO APLICANDO MATRICES DE AVANCE Y MATRICES DE TRANSPOSICIÓN DE VÉRTICES

A continuación, se presenta la implementación del algoritmo aplicando la matriz de conectividad, las sucesivas matrices de avance y las matrices de transposición empleadas para reordenar los vértices del grafo.

Para recorrer el grafo se empleará el método descrito en el apartado "Si un grafo de n vértices no tiene ningún bucle cerrado el camino más largo posible será una serie de n vértices y la matriz de retroceso de orden n siempre será nula.

Recorrido Sistemático de Grafos". Se parte de la matriz nodo del vértice 0 y se la multiplica por la matriz de conectividad por el lado izquierdo. De este modo se calculan los elementos de la siguiente matriz de avance, avanzando sistemáticamente por el grafo e identificando sucesivos vértices previos y siguientes vértices tal como se describe en el pseudo-código del apartado "Pseudo-código del Algoritmo de Reordenación del Flujograma". Dado que sólo la columna del vértice inicial de las matrices de avance contiene elementos distintos de cero, las matrices de avance se representarán con una única columna.

Cuando un elemento de la siguiente matriz de avance es distinto de cero, significa que el elemento de la matriz de conectividad que se acaba de usar para el cálculo de la multiplicación es distinto de cero. Los índices de dicho elemento de la matriz de conectividad identifican cuáles son el vértice previo y el siguiente vértice.

La identificación del último vértice directamente posterior al vértice previo viene dada por la matriz de avance que se está calculando. Si el elemento de la nueva matriz de avance inmediatamente posterior al elemento correspondiente al vértice previo es cero, significa que el siguiente vértice puede colocarse inmediatamente a continuación. Si dicho elemento es distinto de cero, significa que el algoritmo deberá seguir buscando hasta encontrar un elemento distinto de cero.

La reordenación se realiza con una matriz de transposición calculada según el método del apartado "Isomorfismo de Grafos y Matriz de Transposición de Vértices".

A continuación, se aplica el algoritmo al grafo presentado en la Figura 24. En primer lugar se calcula la primera matriz de avance partiendo de la matriz de nodo del vértice 0:

$$C \times N^0 = S^1$$

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \end{pmatrix} \times \begin{pmatrix} 1 & \dots & 0 \\ 0 & & 0 \\ 0 & & 0 \\ 0 & \ddots & 0 \\ 0 & & 0 \\ 0 & & 0 \\ 0 & \dots & 0 \end{pmatrix} = \begin{pmatrix} 0 & \dots & \vdots \\ 0 & & \vdots \\ 1 & & \vdots \\ \vdots & \ddots & \vdots \\ \vdots & & \vdots \\ \vdots & & \vdots \\ \vdots & \dots & \vdots \end{pmatrix}$$

Esto se corresponde con el estado representado en la Figura 24.

El primer elemento de S^1 distinto de cero es $s_{20}^1 = 1$ y aparece al multiplicar n_{00}^0 por c_{20}. Los subíndices indican que el algoritmo ha pasado del vértice previo 0 al siguiente vértice 2 en su

recorrido sistemático del grafo. El hecho de que el elemento de la matriz de avance S^1 inmediatamente inferior al vértice previo 0 es $s_{10}^1 = 0$ indica que el vértice 1 no es un sucesor inmediato del vértice 0 y que por lo tanto se puede subir el vértice 2 a la posición del vértice 1.

La matriz de transposición para conmutar los vértices 1➔2 y 2➔1 es:

$$W_1 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

El algoritmo aplica la matriz de transposición W_1 para conmutar los vértices 1 y 2 de la matriz de conectividad C, de la matriz nodo del vértice 0 N^0, y de la primera matriz de avance S^1. Continuando con la multiplicación matricial:

$$C \times N^0 = S^1$$

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} \times \begin{pmatrix} 1 & \cdots & 0 \\ 0 & & 0 \\ 0 & & 0 \\ 0 & \ddots & 0 \\ 0 & & 0 \\ 0 & & 0 \\ 0 & \cdots & 0 \end{pmatrix} = \begin{pmatrix} 0 & \cdots & \vdots \\ 1 & & \vdots \\ 0 & & \vdots \\ 0 & \ddots & \vdots \\ 1 & & \vdots \\ \vdots & & \vdots \\ \vdots & \cdots & \vdots \end{pmatrix}$$

Esto se corresponde con el estado representado en la Figura 25.

El siguiente elemento de S^1 distinto de cero es $s_{40}^1 = 1$ y aparece al multiplicar n_{00}^0 por c_{40}. Los subíndices indican que el algoritmo ha pasado del vértice previo 0 al siguiente vértice 4 en su recorrido sistemático del grafo. El hecho de que el elemento de la matriz de avance S^1 inmediatamente inferior al vértice previo 0 es $s_{10}^1 = 1$ indica que el vértice 1 también es un sucesor inmediato del vértice 0 y que por lo tanto se debe subir el vértice 4 a la posición del vértice 2.

La matriz de transposición para conmutar los vértices 4➔2, 2➔3 y 3➔4 (véase "Isomorfismo de Grafos y Matriz de Transposición de Vértices") es:

$$W_2 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

El algoritmo conmuta los vértices y continúa con la multiplicación matricial:

$$C \times N^0 = S^1$$

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \times \begin{pmatrix} 1 & \dots & 0 \\ 0 & & 0 \\ 0 & & 0 \\ 0 & \ddots & 0 \\ 0 & & 0 \\ 0 & & 0 \\ 0 & \dots & 0 \end{pmatrix} = \begin{pmatrix} 0 & \dots & 0 \\ 1 & & 0 \\ 1 & & 0 \\ 0 & \ddots & 0 \\ 0 & & 0 \\ 0 & & 0 \\ 0 & \dots & 0 \end{pmatrix}$$

La matriz de avance de primer orden ha sido completada sin que aparezcan más elementos distintos de cero con lo que el algoritmo pasa a calcular los vértices de la matriz de avance de segundo orden:

$$C \times S^1 = S^2$$

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \times \begin{pmatrix} 0 & \dots & 0 \\ 1 & & 0 \\ 1 & & 0 \\ 0 & \ddots & 0 \\ 0 & & 0 \\ 0 & & 0 \\ 0 & \dots & 0 \end{pmatrix} = \begin{pmatrix} 0 & \dots & \vdots \\ 0 & & \vdots \\ 0 & & \vdots \\ 0 & \ddots & \vdots \\ 1 & & \vdots \\ \vdots & & \vdots \\ \vdots & \dots & \vdots \end{pmatrix}$$

Esto se corresponde con el estado representado en la Figura 26.

El primer elemento de S^2 distinto de cero es $s^2_{40} = 1$ y aparece al multiplicar s^1_{20} por c_{42}. Los subíndices indican que el algoritmo ha pasado del vértice previo 2 al siguiente vértice 4 en su recorrido sistemático del grafo. El hecho de que el elemento de la matriz de avance S^2 inmediatamente inferior al vértice previo 2 es $s^2_{30} = 0$ indica que el vértice 3 no es un sucesor inmediato del vértice 2 y que por lo tanto se debe subir el vértice 4 a la posición del vértice 3.

La matriz de transposición para conmutar los vértices 3➔4 y 4➔3 es:

$$W_3 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

El algoritmo conmuta los vértices y continúa con la multiplicación matricial:

$$C \times S^1 = S^2$$

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} \times \begin{pmatrix} 0 & \dots & 0 \\ 1 & & 0 \\ 1 & & 0 \\ 0 & \ddots & 0 \\ 0 & & 0 \\ 0 & & 0 \\ 0 & \dots & 0 \end{pmatrix} = \begin{pmatrix} 0 & \dots & \vdots \\ 0 & & \vdots \\ 0 & & \vdots \\ 1 & \ddots & \vdots \\ 0 & & \vdots \\ 1 & & \vdots \\ \vdots & \dots & \vdots \end{pmatrix}$$

Esto se corresponde con el estado representado en la Figura 27.

El siguiente elemento de S^2 distinto de cero es $s_{50}^2 = 1$ y aparece al multiplicar s_{10}^1 por c_{51}. Los subíndices indican que el algoritmo ha pasado del vértice previo 1 al siguiente vértice 5 en su recorrido sistemático del grafo. El hecho de que el elemento de la matriz de avance S^2 inmediatamente inferior al vértice previo 1 es $s_{20}^2 = 0$ indica que el vértice 2 no es un sucesor inmediato del vértice 1 y que por lo tanto se debe subir el vértice 5 a la posición del vértice 2.

La matriz de transposición para conmutar los vértices 5➔2, 2➔3, 3➔4 y 4➔5 es:

$$W_4 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

El algoritmo conmuta los vértices y continúa con la multiplicación matricial:

$$C \times S^1 = S^2$$

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \times \begin{pmatrix} 0 & \cdots & 0 \\ 1 & & 0 \\ 0 & & 0 \\ 1 & \ddots & 0 \\ 0 & & 0 \\ 0 & & 0 \\ 0 & \cdots & 0 \end{pmatrix} = \begin{pmatrix} 0 & \cdots & \vdots \\ 0 & & \vdots \\ 1 & & \vdots \\ 0 & \ddots & \vdots \\ 1 & & \vdots \\ 0 & & \vdots \\ 1 & \cdots & \vdots \end{pmatrix}$$

Esto se corresponde con el estado representado en la Figura 28.

El siguiente elemento de S^2 distinto de cero es $s_{60}^2 = 1$ y aparece al multiplicar s_{10}^1 por c_{61}. Los subíndices indican que el algoritmo ha pasado del vértice previo 1 al siguiente vértice 6 en su recorrido sistemático del grafo. El hecho de que el elemento de la matriz de avance S^2 inmediatamente inferior al vértice previo 1 es $s_{20}^2 = 1$ indica que el vértice 2 también es un sucesor inmediato del vértice 1 y que por lo tanto se debe subir el vértice 6 a la posición del vértice 3.

La matriz de transposición para conmutar los vértices 6➔3, 3➔4, 4➔5 y 5➔6 es:

$$W_5 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

El algoritmo conmuta los vértices y continúa con la multiplicación matricial:

$$C \times S^1 = S^2$$

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \end{pmatrix} \times \begin{pmatrix} 0 & \cdots & 0 \\ 1 & & 0 \\ 0 & & 0 \\ 0 & \ddots & 0 \\ 1 & & 0 \\ 0 & & 0 \\ 0 & \cdots & 0 \end{pmatrix} = \begin{pmatrix} 0 & \cdots & 0 \\ 0 & & 0 \\ 1 & & 0 \\ 1 & \ddots & 0 \\ 0 & & 0 \\ 1 & & 0 \\ 0 & \cdots & 0 \end{pmatrix}$$

La matriz de avance de segundo orden ha sido completada sin que aparezcan más elementos distintos de cero con lo que el algoritmo pasa a calcular los vértices de la matriz de avance de tercer orden:

$$C \times S^2 = S^3$$

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \end{pmatrix} \times \begin{pmatrix} 0 & \dots & 0 \\ 0 & & 0 \\ 1 & & 0 \\ 1 & \ddots & 0 \\ 0 & & 0 \\ 1 & & 0 \\ 0 & \dots & 0 \end{pmatrix} = \begin{pmatrix} 0 & \dots & \vdots \\ 0 & & \vdots \\ 1 & & \vdots \\ \vdots & \ddots & \vdots \\ \vdots & & \vdots \\ \vdots & & \vdots \\ \vdots & \dots & \vdots \end{pmatrix}$$

Esto se corresponde con el estado representado en la Figura 29.

El siguiente elemento de S^3 distinto de cero es $s_{20}^3 = 1$ y aparece al multiplicar s_{50}^2 por c_{25}. Los subíndices indican que el algoritmo ha pasado del vértice previo 5 al siguiente vértice 2 en su recorrido sistemático del grafo. En este caso el elemento de la matriz de avance S^3 inmediatamente inferior al vértice previo 5 todavía está sin calcular y por lo tanto se debe bajar el vértice 2 a la posición del vértice 5.

La matriz de transposición para conmutar los vértices 2➔5, 5➔4, 4➔3 y 3➔2 es:

$$W_6 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

El algoritmo conmuta los vértices y continúa con la multiplicación matricial:

$$C \times S^2 = S^3$$

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \end{pmatrix} \times \begin{pmatrix} 0 & \dots & 0 \\ 0 & & 0 \\ 1 & & 0 \\ 0 & \ddots & 0 \\ 1 & & 0 \\ 1 & & 0 \\ 0 & \dots & 0 \end{pmatrix} = \begin{pmatrix} 0 & \dots & \vdots \\ 0 & & \vdots \\ 1 & & \vdots \\ \vdots & \ddots & \vdots \\ \vdots & & \vdots \\ \vdots & & \vdots \\ \vdots & \dots & \vdots \end{pmatrix}$$

Esto se corresponde con el estado representado en la Figura 30.

El siguiente elemento de S^3 distinto de cero es $s_{20}^3 = 1$ y aparece al multiplicar s_{40}^2 por c_{24}. Los subíndices indican que el algoritmo ha pasado del vértice previo 4 al siguiente vértice 2 en su recorrido sistemático del grafo. En este caso el elemento de la matriz de avance S^3 inmediatamente inferior al vértice previo 4 todavía está sin calcular y por lo tanto se debe bajar el vértice 2 a la posición del vértice 4.

La matriz de transposición para conmutar los vértices 2➔4, 4➔3 y 3➔2 es:

$$W_7 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

El algoritmo conmuta los vértices y continúa con la multiplicación matricial:

$$C \times S^2 = S^3$$

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \end{pmatrix} \times \begin{pmatrix} 0 & \cdots & 0 \\ 0 & & 0 \\ 0 & & 0 \\ 1 & \ddots & 0 \\ 1 & & 0 \\ 1 & & 0 \\ 0 & \cdots & 0 \end{pmatrix} = \begin{pmatrix} 0 & \cdots & \vdots \\ 0 & & \vdots \\ 0 & & \vdots \\ 0 & \ddots & \vdots \\ 1 & & \vdots \\ \vdots & & \vdots \\ \vdots & \cdots & \vdots \end{pmatrix}$$

El siguiente elemento de S^3 distinto de cero es $s_{40}^3 = 1$ y aparece al multiplicar s_{30}^2 por c_{43}. Los subíndices indican que el algoritmo ha pasado del vértice previo 3 al siguiente vértice 4 en su recorrido sistemático del grafo. En este caso no es necesaria ninguna conmutación de vértices puesto que el siguiente vértice 4 ya está a continuación del último vértice tras el 3, que es el vértice 4.

El algoritmo continúa con la multiplicación matricial:

$$C \times S^2 = S^3$$

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \end{pmatrix} \times \begin{pmatrix} 0 & \cdots & 0 \\ 0 & & 0 \\ 0 & & 0 \\ 1 & \ddots & 0 \\ 1 & & 0 \\ 1 & & 0 \\ 0 & \cdots & 0 \end{pmatrix} = \begin{pmatrix} 0 & \cdots & \vdots \\ 0 & & \vdots \\ 0 & & \vdots \\ 0 & \ddots & \vdots \\ 1 & & \vdots \\ 1 & & \vdots \\ \vdots & \cdots & \vdots \end{pmatrix}$$

El siguiente elemento de S^3 distinto de cero es $s_{50}^3 = 1$ y aparece al multiplicar s_{30}^2 por c_{53}. Los subíndices indican que el algoritmo ha pasado del vértice previo 3 al siguiente vértice 5 en su recorrido sistemático del grafo. En este caso no es necesaria ninguna conmutación de vértices puesto que el siguiente vértice 5 ya está a continuación del vértice previo 3.

El algoritmo continúa con la multiplicación matricial:

$$C \times S^2 = S^3$$

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \end{pmatrix} \times \begin{pmatrix} 0 & \cdots & 0 \\ 0 & & 0 \\ 0 & & 0 \\ 1 & \ddots & 0 \\ 1 & & 0 \\ 1 & & 0 \\ 0 & \cdots & 0 \end{pmatrix} = \begin{pmatrix} 0 & \cdots & \vdots \\ 0 & & \vdots \\ 0 & & \vdots \\ 0 & \ddots & \vdots \\ 1 & & \vdots \\ 1 & & \vdots \\ 1 & \cdots & \vdots \end{pmatrix}$$

Esto se corresponde con el estado de la Figura 31.

El siguiente elemento de S^3 distinto de cero es $s_{60}^3 = 1$ y aparece al multiplicar s_{50}^2 por c_{65}. Los subíndices indican que el algoritmo ha pasado del vértice previo 5 al siguiente vértice 6 en su recorrido sistemático del grafo. En este caso no es necesaria ninguna conmutación de vértices puesto que el siguiente vértice 6 ya está a continuación del vértice previo 5.

El algoritmo continúa con la multiplicación matricial sin que aparezcan nuevos elementos de S^3 distintos de cero. El siguiente paso consiste en calcular la matriz de avance de cuarto orden S^4. El resultado de esta operación es cero en todos los elementos excepto $s_{65}^4 = 1$ lo que indica que el vértice previo es 5 y el siguiente vértice es 6. Puesto que ambos vértices están en el orden adecuado no es necesaria ninguna conmutación adicional de vértices. La matriz de avance de quinto orden no tiene elementos distintos de cero, lo cual indica que el algoritmo ha completado el recorrido sistemático del grafo y que ha llegado a su conclusión.

PRIMERA MEJORA DEL ALGORITMO APLICANDO LA PRUEBA DE SUCESIÓN / PRECEDENCIA

Durante la depuración y validación del algoritmo de reordenación del flujograma tal como está descrito en el apartado anterior apareció una dificultad con el grafo representado en la Figura 32:

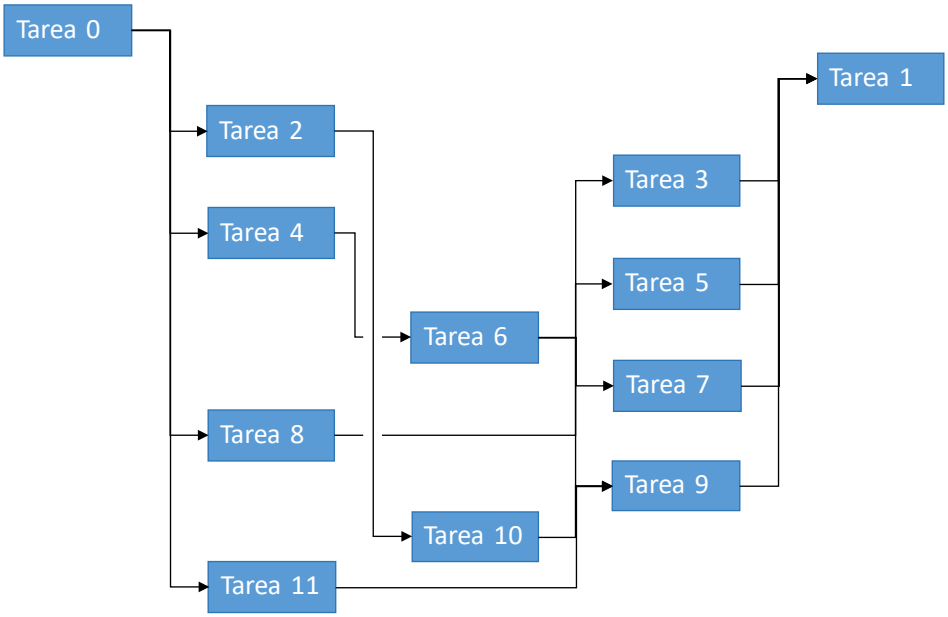


Figura 32 Grafo antes de aplicar el algoritmo de reordenación del flujograma

Uno de los grafos intermedios durante la ejecución del algoritmo es el siguiente:

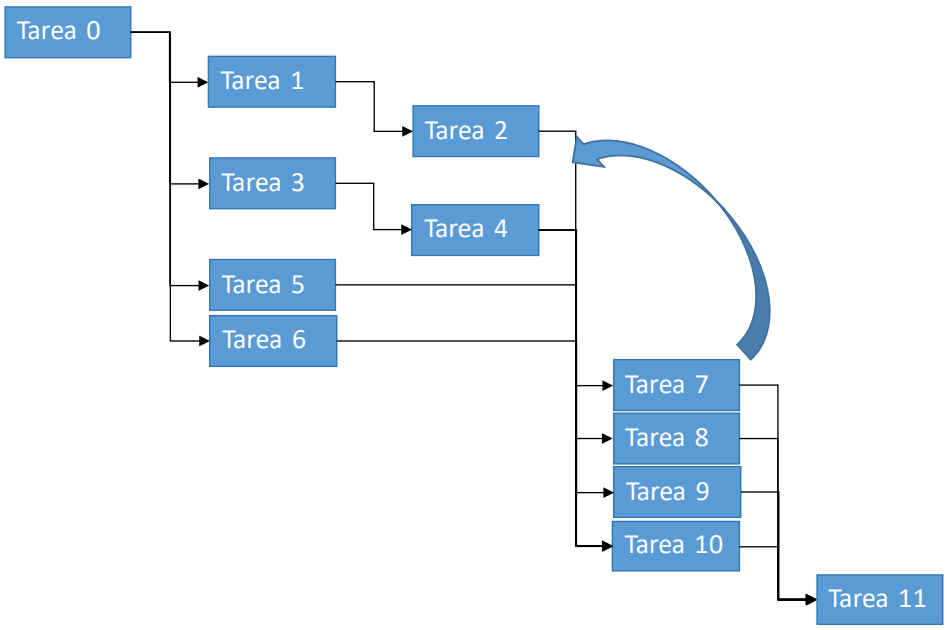


Figura 33 Grafo intermedio en la reordenación del grafo de la Figura 32

En este estado intermedio del algoritmo de reordenación, el vértice previo es el 2 y el siguiente vértice es el 7. Como el vértice 3 no es antecesor directo del 7, el algoritmo desplazaría el vértice 7 directamente a continuación del 2 y los vértices 8, 9 y 10 le seguirían a continuación. En pasos subsiguientes los vértices 7, 8, 9 y 10 finalizarían entre el vértice 4 y el 5. El grafo final es:

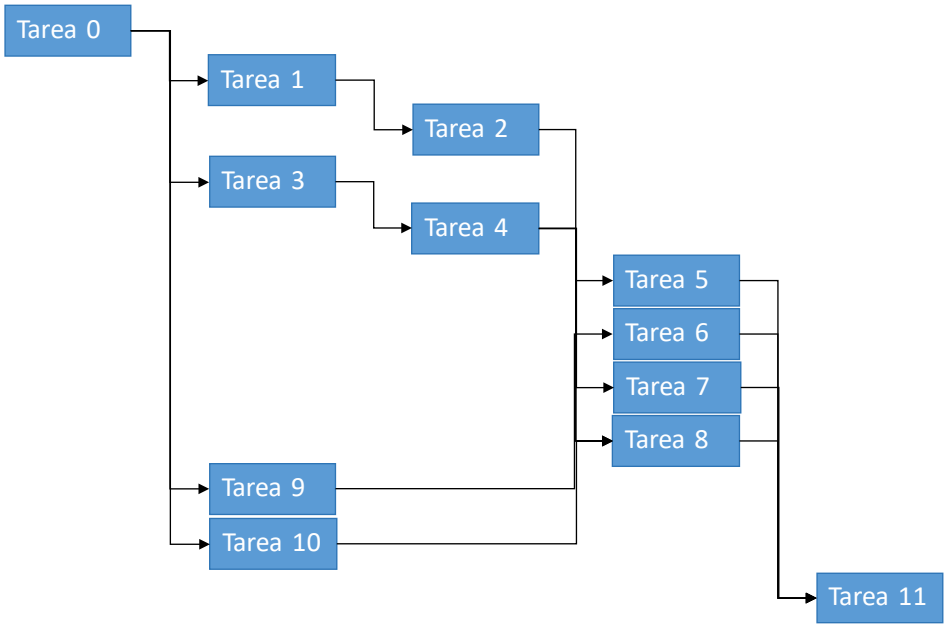


Figura 34 Grafo resultante tras aplicar el algoritmo hasta el final

La versión inicial del algoritmo de reordenación del flujograma coloca el siguiente vértice en la primera posición nula de la matriz de avance tras el vértice previo, es decir, sólo se consideran los vértices inmediatamente previos (vértices 2 y 4 en el ejemplo presentado).

La versión mejorada del algoritmo de reordenación del flujograma coloca el siguiente vértice en la primera posición que no sea un vértice predecesor. Para ello se aplica el algoritmo presentado en el apartado "Prueba de Sucesión / Precedencia". Esta mejora hace que el vértice 7 de la Figura 33 no se coloque en la posición del vértice 3.

SEGUNDA MEJORA DEL ALGORITMO APLICANDO LAS SERIES DE AVANCES Y RETROCESOS CUADRÁTICOS DE LOS VÉRTICES

La siguiente dificultad apareció con un grafo cuya apariencia final tras aplicarle el algoritmo de reordenación del flujograma fue:

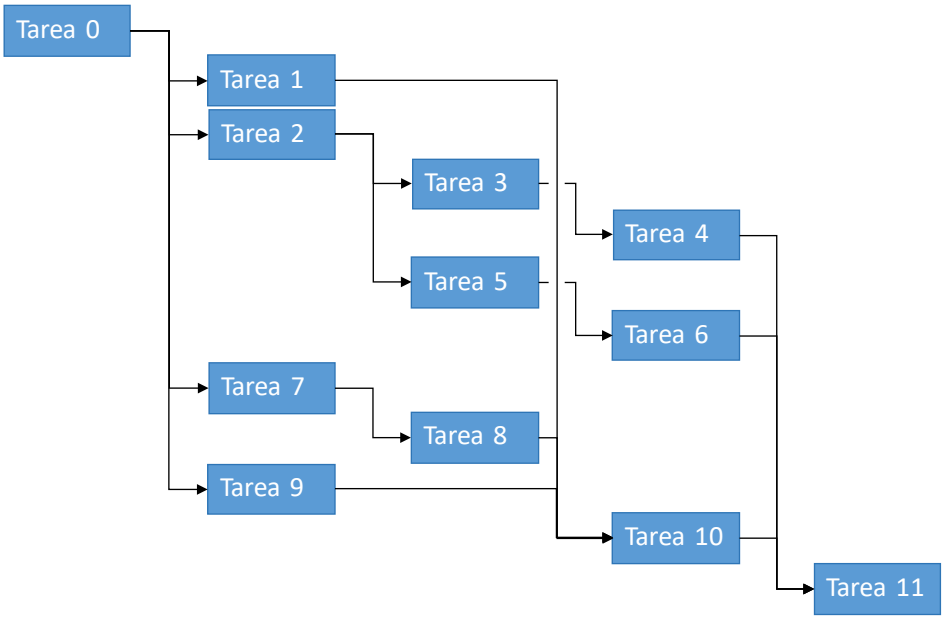


Figura 35 Grafo ilustración para la segunda mejora del algoritmo de reordenación de flujogramas

Los vértices 1 y 9 tienen los mismos predecesores y sucesores y se diferencian claramente del resto de vértices por su posición en el grafo. Sus series de avances cuadráticos (véase apartado "Proximidad de un Vértice *o* a los Vértices Inicial y al Final del Grafo") son idénticas. El algoritmo de reordenación de flujogramas hubiera debido agrupar los vértices 1 y 9 pero debido a su posición separada en el grafo original han quedado separados. La segunda mejora del algoritmo original de reordenación de flujogramas consiste en calcular las series de avances cuadráticos de todos los vértices y en ordenarlos según su cercanía al vértice 0. De este modo los vértices 1 y 9 entran consecutivamente en el algoritmo de reordenación de flujogramas y así se mantienen hasta el resultado final de la Figura 36.

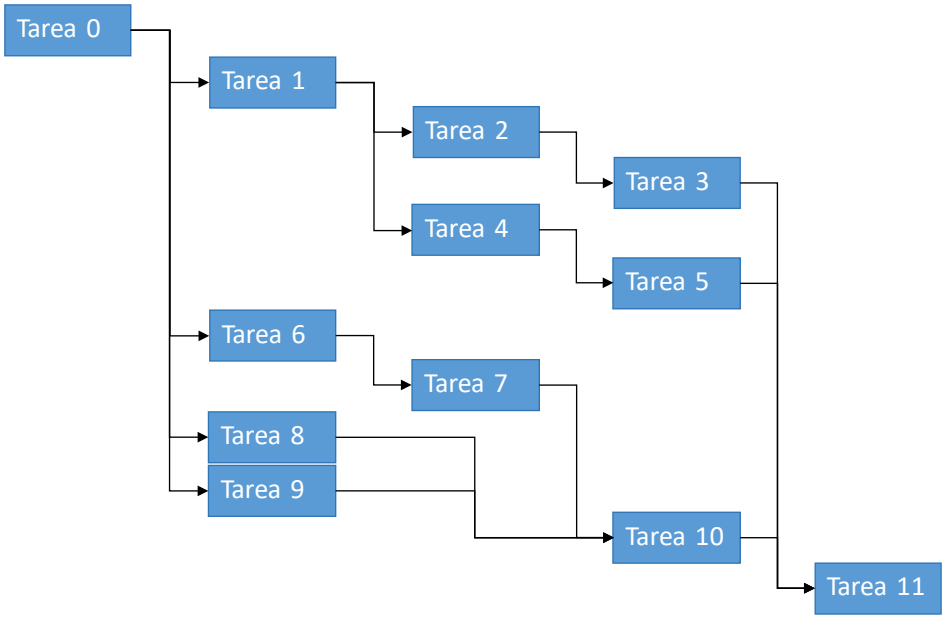


Figura 36 Grafo de la Figura 35 tras aplicar el algoritmo de reordenación de flujogramas mejorado

4.3 APLICACIÓN INFORMÁTICA

En este capítulo se presenta la aplicación informática que implementa el algoritmo de secuenciación de mantenimiento operacional OMSA. Se subdivide en los siguientes apartados:

- Entrada de datos. Explicación del formato de los ficheros de entrada y de sus contenidos.
- Clases. Explicación de las variables y los métodos de las clases principales.
- Salida de datos. Semejante a la entrada de datos.

Las clases de la aplicación informática están estructuradas tal como se representa en la Figura 37.

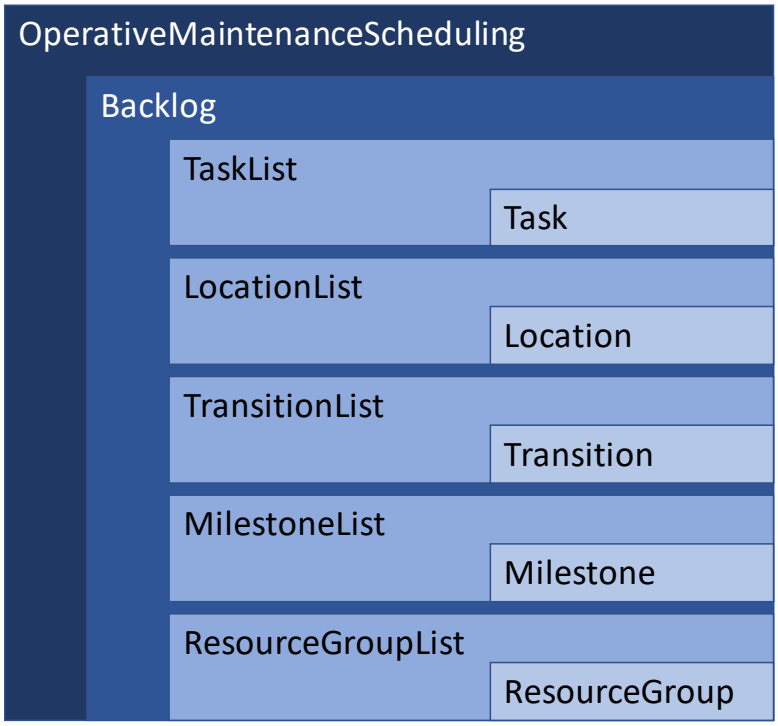


Figura 37 Estructura de clases de la aplicación informática

Los principales métodos de las clases están representados en la Figura 38.

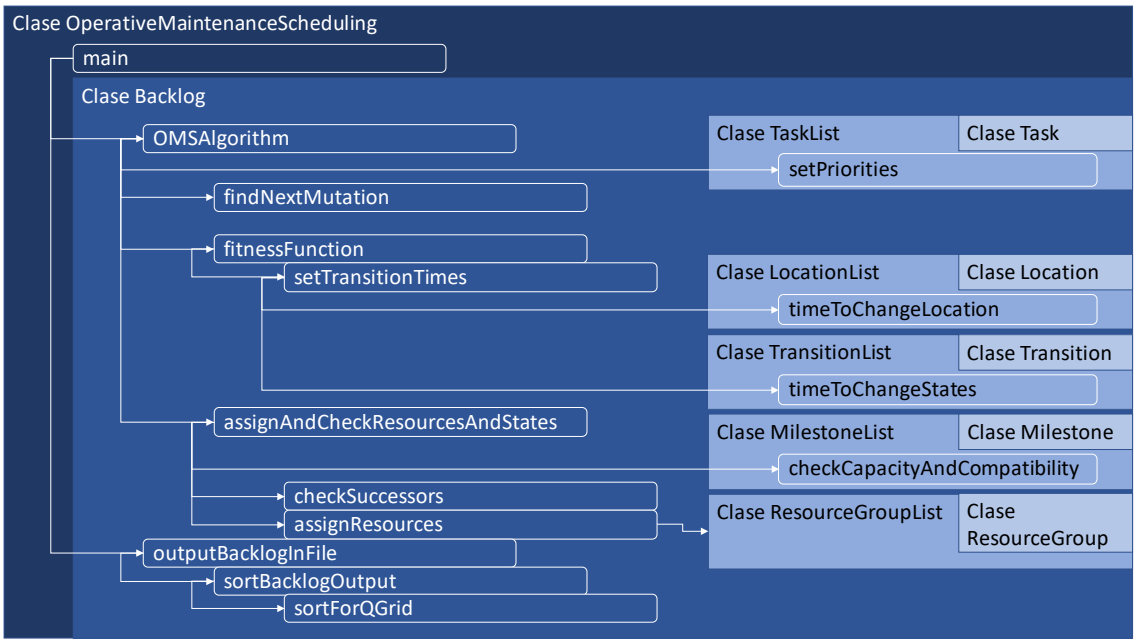


Figura 38 Principales métodos de la aplicación informática

4.3.1 ENTRADA DE DATOS

La aplicación informática lee los datos de cuatro ficheros distintos en formato CSV (*comma separated values*, valores separados por comas):

- Fichero de datos de tareas de mantenimiento.
- Fichero de datos con las distancias entre las localizaciones del tren.
- Fichero de datos con los tiempos de cambio del estado de seguridad del tren.
- Fichero de datos de recursos y cualificaciones.

Los ficheros CSV emplean un formato sencillo para almacenar tablas. Los datos de cada campo en un registro están representados en formato de texto y los distintos campos están separados por una coma o por un punto y coma. Cada registro finaliza con un carácter de cambio de línea.

Dentro de los diferentes formatos CSV existentes se ha elegido el de MS-DOS por ser el más sencillo de implementar con el lenguaje de programación Java.

Todos los ficheros de datos de entrada han sido originados en la aplicación Excel de Microsoft Office 365 en el formato estándar XLSX. Excel ofrece la posibilidad de guardar los archivos con el formato CSV de MS-DOS.

DATOS DE TAREAS DE MANTENIMIENTO

Los campos del registro de cada tarea de mantenimiento están representados en la siguiente tabla.

Campo	Descripción
Código	Código de la tarea de mantenimiento. Debe ser único para cada tarea.
Nombre	Nombre de la tarea de mantenimiento.

Campo	Descripción
	Hay dos nombres reservados de tarea: • INITIAL. Denominación de la primera tarea que se debe ejecutar en el flujograma de la estadía. En las estadías de mantenimiento de trenes de alta velocidad, la primera tarea consiste en la lectura del libro de maquinista, en el que se anotan las deficiencias observadas durante la explotación comercial, y en la lectura de la diagnosis del sistema de control del tren. • FINAL. Ésta es la última tarea de mantenimiento antes de dar útil el tren para el servicio comercial. En esta tarea se realiza una última inspección para asegurar que todos los sistemas están operativos, las tapas cerradas, etc. Las tareas INITIAL y FINAL deben estar siempre presentes en la lista de tareas.
Duración	Duración de la tarea en minutos.
Estado de seguridad	El estado de seguridad viene dado por los estados de diferentes equipamientos del tren. En casi todas las simulaciones realizadas en esta tesis el estado de seguridad del tren se compone de los estados de los dos siguientes equipamientos: • Batería del tren, 110V DC, conectada. • Catenaria del taller, 25kV AC 50Hz, conectada. El estado de seguridad requerido para la ejecución de una tarea de mantenimiento está codificado mediante una A, B o C para cada estado de seguridad contemplado. Los significados son: • A: estado tal como se describe textualmente. • B: estado opuesto al descrito. • C: estado indiferente para la tarea. Por ejemplo, si para la ejecución de una tarea es necesario que la batería del tren esté conectada y la catenaria esté conectada, el estado de seguridad sería AA. Si por ejemplo la batería debe estar conectada y el estado de la catenaria es indiferente, el estado de seguridad sería AC.
Localización	El código de localización en las simulaciones realizadas en esta tesis se compone de tres dígitos. El primer dígito representa el coche del tren y va de 1 a 8 en un tren de 8 coches. Los siguientes dos dígitos representan la localización en el coche: • 10: techo • 20: interior • 30: exterior • 40: bajo bastidor Estas localizaciones están representadas en la Figura 2 de la página 5.
Intervalo	Intervalo en kilómetros de la intervención a la que pertenece la tarea de mantenimiento.
Umbral prioridad alta	Si la distancia hasta el vencimiento del intervalo de la tarea es menor que este umbral, la tarea recibe la prioridad más alta (1) y es ejecutada obligatoriamente durante la siguiente estadía.
Umbral prioridad media ejecución	Si la tarea no ha sido ejecutada ninguna vez durante la simulación en curso o la distancia hasta el vencimiento del

Campo	Descripción
	intervalo es menor que este umbral, la tarea recibe la prioridad media (2) y es ejecutada durante la estadía en curso si el tiempo disponible lo permite. Si en una simulación sólo se emplease el umbral de prioridad alta del campo anterior el efecto sería que las tareas de intervenciones superiores no se ejecutarían hasta que se sobrepasara el umbral de prioridad alta. En tal caso, todas las tareas de la intervención deberían ejecutarse en la siguiente estadía y se sobrepasaría con seguridad el tiempo disponible asignado.
Técnicos de bogies (sistema de rodadura)	Número de técnicos especialistas en bogies o sistema de rodadura necesarios para esta tarea.
Técnicos no especialistas	Número de técnicos sin cualificación especializada necesarios para esta tarea.
Técnicos electricistas	Número de técnicos electricistas necesarios para esta tarea.
Técnicos mecánicos	Número de técnicos mecánicos necesarios para esta tarea.
Tarea sucesora	Código de la tarea sucesora que ha de ejecutarse siempre en combinación con la actual. En el caso de que no haya ninguna tarea sucesora, este campo tiene el valor “na” (no aplicable).

Tabla I Datos de tareas de mantenimiento

DATOS DE DISTANCIAS ENTRE LOCALIZACIONES

Las distancias entre localizaciones se miden con el tiempo en minutos necesario para transitar desde una localización a la siguiente. Los campos del fichero con esta información están representados en la siguiente tabla.

Campo	Descripción
Localización de salida	Código de la localización de salida en las simulaciones realizadas en esta tesis se compone de tres dígitos. El primer dígito representa el coche del tren y va de 1 a 8 en un tren de 8 coches. Los siguientes dos dígitos representan la localización dentro del coche: • 10: techo • 20: interior • 30: exterior • 40: bajo bastidor Estas localizaciones están representadas en la Figura 2 de la página 5.
Localización de llegada	Código de la localización de llegada en las simulaciones realizadas en esta tesis se compone de tres dígitos. El primer dígito representa el coche del tren y va de 1 a 8 en un tren de 8 coches. Los siguientes dos dígitos representan la localización dentro del coche: • 10: techo • 20: interior • 30: exterior • 40: bajo bastidor

Campo	Descripción
	Estas localizaciones están representadas en la Figura 2 de la página 5.
Tiempo de desplazamiento	Tiempo de desplazamiento en minutos desde la localización de salida a la de llegada.

Tabla II Datos de distancias entre localizaciones

Los datos de distancias empleados por defecto en las simulaciones de esta tesis se encuentran en la Figura 3 de la página 6.

DATOS DE TIEMPOS DE CAMBIO DEL ESTADO DE SEGURIDAD DEL TREN

Los tiempos de cambio de estado están almacenados en una tabla con un único campo. Sus contenidos son:

- 30;"\n";45;"\n";15; "\n";15; "\n"

Que se corresponden con las siguientes acciones:

Acción	Tiempo (minutos)
Desconectar catenaria, 25kV AC 50Hz	30
Conectar catenaria, 25kV AC 50Hz	45
Desconectar batería, 110V DC.	15
Conectar batería, 110V DC.	15

Tabla III Datos de tiempos de cambio de estado de seguridad del tren

DATOS DE RECURSOS Y DE SUS CUALIFICACIONES

Los datos de los recursos y de su cualificación se encuentran almacenados de la siguiente manera para cada cualificación:

1. Primer registro: número de recursos.
2. Segundo registro: cualificación.
3. Siguientes registros: nombres de los recursos.

En las simulaciones de esta tesis se han usado 4 diferentes cualificaciones:

1. Técnico de sistema de rodadura.
2. Técnico no cualificado.
3. Técnico electricista.
4. Técnico mecánico.

4.3.2 CLASE *OPERATIVEMAINTENANCESCHEDULING*

La clase *OperativeMaintenanceScheduling* contiene la función principal *main* de la aplicación. La función *main* inicializa los datos. La simulación del mantenimiento operativo se realiza con un bucle. En cada ejecución del bucle se simula una estadía del tren en el taller. Para ello se calcula el tiempo disponible de la estadía, se incrementa el contador de kilometraje y se invoca al algoritmo de secuenciación operacional de mantenimiento. El algoritmo calcula las prioridades de las tareas de mantenimiento y un flujograma con asignación de recursos y cumpliendo las restricciones especificadas.

VARIABLES

Esta clase no tiene variables relevantes

MÉTODO *MAIN*

El pseudo-código del método es:

```
Método main:

Inicializa datos de la clase Backlog
Abre fichero de salida
FOR (número de repeticiones deseadas) {
     Calcula kilometraje añadiendo valor
     Calcula tiempo de taller disponible
     backlog.OMSAlgorithm
     backlog.outputBacklogInFile     // Guarda datos en fichero
}
Cierra fichero de salida
```

4.3.3 CLASE *BACKLOG*

La clase *Backlog* contiene las funcionalidades principales de la aplicación informática, que están implementadas en los métodos *OMSAlgorithm* y *outputBacklogInFile*. El resto de las clases de la aplicación gestionan los datos relativos a tareas, localizaciones, tiempos de transición de estado de seguridad, hitos y recursos.

VARIABLES

En este apartado se presentan únicamente las variables que aparecen en los pseudo-códigos de los métodos más aquellas de especial relevancia.

Variable	Descripción
connectivity	Integer[][] *connectivity* Matriz de conectividad del grafo del flujograma. Sus dimensiones son *numberOfTasks x numberOfTasks*. La gestión de la matriz de conectividad se encuentra en el centro de los algoritmos que permiten implementar OMSA y a través de la matriz *connectivity* se ponen en práctica las técnicas presentadas en el capítulo "Técnicas de Análisis y Manipulación de Grafos", página 33.
generation	int *generation* Contador de las generaciones tal como se describen en el capítulo "Algoritmos Genéticos", página 17. Su primer valor es 1.
locationList	LocationList *locationList* Clase con información de las localizaciones de las tareas de mantenimiento (véase el apartado "Clase *Location*", página 75).
milestoneList	MilestoneList *milestoneList* Clase con información de los hitos empleados para el control de recursos y estados de seguridad (véase el apartado "Clase *MilestoneList*", página 79).
mutationIsPlausible	boolean *mutationIsPlausible* Esta variable es *false* si se cumple al menos una de las siguientes condiciones:

Variable	Descripción
	• Hay alguna incompatibilidad al menos entre el estado de seguridad de un equipamiento durante la tarea y tras la tarea (véase apartado "Método *timeToChangeStates*", página 76). • La siguiente mutación no es factible (véase apartado "Método *findNextMutation*", página 66). • No hay recursos suficientes para ejecutar las tareas del flujograma o hay tareas en paralelo con estados de seguridad incompatibles (véase apartado "Método *checkCapacityAndCompatibility*", página 79). • Hay al menos una tarea sucesora posicionada antes de la tarea predecesora (véase apartado "Método *checkSuccessors*", página 70). • Al menos en una ocasión no hay tiempo necesario para que un recurso se desplace de la localización de una tarea a la siguiente (véase apartado "Método *assignResources*", página 71).
nodeFinish	Integer[] *nodeFinish* Matriz de *numberOfTask* números enteros. Esta variable guarda el tiempo de finalización de cada tarea dentro del flujograma (véase apartado "Método *setTransitionTimes*", página 68).
nodeStart	Integer[] *nodeStart* Matriz de *numberOfTask* números enteros. Esta variable guarda el tiempo de inicio de cada tarea dentro del flujograma (véase apartado "Método *setTransitionTimes*", página 68).
nodeStatesAfterTask	char[][] *nodeStatesAfterTask* Matriz de caracteres con dimensiones *numberOfTasks x numberOfStates*. Guarda, para cada tarea, el código de cada uno de los estados de seguridad tras su finalización. Sus valores se asignan en el método *setTransitionTimes* pero se calculan en el método *transitionList.timeToChangeStates*. (véanse los apartados "Método *setTransitionTimes*", página 68, y "Método *timeToChangeStates*", página 76).
nodeStatesAtTask	char[][] *nodeStatesAtTask* Matriz de caracteres con dimensiones *numberOfTasks x numberOfStates*. Guarda, para cada tarea, el código de cada uno de los estados de seguridad durante su ejecución. Sus valores se asignan en el método *setTransitionTimes* pero se calculan en el método *transitionList.timeToChangeStates*. (véanse los apartados "Método *setTransitionTimes*", página 68, y "Método *timeToChangeStates*", página 76).
numberOfResourceGroups	int *numberOfResourceGroups* Número de distintos grupos de recursos por cualificación. En esta tesis hemos empleado 4 tipos de recursos (véase el apartado "Datos de Recursos y de sus Cualificaciones", página 62).
numberOfStates	int *numberOfStates* Número de diferentes estados de seguridad de los equipamientos. En esta tesis hemos empleado 2 tipos de estados (véase el apartado "Datos de Tiempos de Cambio del Estado de Seguridad del Tren", página 62).
numberOfTasks	int *numberOfTasks*

Variable	Descripción
	Numero total de tareas de mantenimiento.
resourceGroupList	ResourceGroupList *resourceGroupList* Clase con información de los recursos y de sus cualificaciones (véase el apartado "Clase *ResourceGroupList*", página 82).
shortestDuration	Integer *shortestDuration* Duración del flujograma más corto de la generación actual (véase "Método *OMSAlgorithm*", página 65).
stateChangeFinish	Integer[] *stateChangeFinish* Matríz de *numberOfTasks* elementos. Almacena el tiempo de finalización del cambio de estados de seguridad detrás de cada tarea (véase "Método *setTransitionTimes*", página 68).
stateChangeStart	Integer[] *stateChangeStart* Matríz de *numberOfTasks* elementos. Almacena el tiempo de inicio del cambio de estados de seguridad detrás de cada tarea (véase "Método *setTransitionTimes*", página 68).
taskList	TaskList *taskList* Clase con información de las tareas de mantenimiento (véase el apartado "Clase *Task*", página 73).
transitionList	TransitionList *transitionList* Clase con información de los estados de seguridad (véase el apartado "Clase *TransitionList*", página 76).

Tabla IV Variables de la clase Backlog

MÉTODO *OMSALGORITHM*

El método *OMSAlgorithm* es el núcleo de la aplicación informática y la implementación del algoritmo presentado en el apartado "Algoritmo Cuasi-Genético", página 31.

```
Método OMSAlgorithm (duraciónDisponible, kilometraje):

Inicializar datos de la clase Backlog

generation = 1
mutationIsPlausible = false
pararBucleGeneraciones = false
pararBucleMutaciones = false

taskList.setPriorities
Ordenar taskList por mayor kilometraje desde última ejecución
Ordenar taskList por prioridad

WHILE (NOT pararBucleGeneraciones) {
     shortestDuration = 0
     primeraMutacion = true
     pararBucleMutaciones = (prioridadTareaActual == 3)

     WHILE (NOT pararBucleMutaciones) {
          Guardar mutación progenitora de la generación
          pararBucleMutaciones = findNextMutation

          IF (mutationIsPlausible) {
               duraciónMutación = fitnessFunction

               IF (mutationIsPlausible) {
                    assignAndCheckResourcesAndStates
```

```
                    IF (mutationIsPlausible AND ((duraciónMutación <
                    shortestDuration) OR (primeraMutación))) {
                         shortestDuration = duraciónMutación
                         Guardar mutación actual como óptima
                         primeraMutación = false
                    }
               }
          }
          Recuperar mutación progenitora de la generación
     }


     IF (prioridadTareaActual == 3) {
          pararBucleGeneraciones = true
          generation = generation – 1
          shortestDuration = duraciónÓptimaPrevia
     }
     ELSE {
          IF ((shortestDuration > duraciónDisponible) AND
          (prioridadTareaActual != 1) AND (tarea actual no tiene sucesor)) {
               pararBucleGeneraciones = true
               generation = generation – 1
               shortestDuration = duraciónÓptimaPrevia
               Restaurar generación previa
          }

          IF (NOT pararBucleGeneraciones) {
               Restaurar mutación óptima
               generation = generation + 1

               IF (generation >= numberOfTasks) {
                    pararBucleGeneraciones = true
                    generation = generation - 1
               }
               ELSE {
                    Marcar tareas de la generación como ejecutadas
                    duraciónÓptimaPrevia = shortestDuration
                    Guardar generación actual
               }
          }
     }
}
Actualizar el kilometraje de las tareas ejecutadas
```

MÉTODO *FINDNEXTMUTATION*

Este método calcula la siguiente mutación dentro de la generación actual. Tal como se explica en el capítulo “Algoritmo Cuasi-Genético”, página 31, cada mutación consiste en la inserción de la nueva tarea en diferentes posiciones del flujograma o, empleando el vocabulario de la teoría de grafos, consiste en la inserción de un nuevo vértice en diferentes posiciones del grafo.

El método *findNextMutation* devuelve el valor *true* si la mutación calculada es la última de la generación actual.

Por cada vez que se invoca el método *findNextMutation* se van generando las mutaciones según el siguiente esquema:

1. Inserción de la nueva tarea i como predecesor de cada tarea j del grafo (excepto de la tarea inicial, para la cual se asigna *mutationIsPlausible* = false) según el método descrito en el apartado “Inserción de Vértice Predecesor”, página 34.

2. Inserción de la nueva tarea i como sucesor de cada tarea j del grafo (excepto de la tarea final, para la cual se asigna *mutationIsPlausible* = false) según el método descrito en el apartado "Inserción de Vértice Sucesor", página 34.
3. Inserción de la nueva tarea i en paralelo a cada tarea j del grafo (excepto de las tareas inicial y final, para las cuales se asigna *mutationIsPlausible* = false) según el método descrito en el apartado "Inserción de Vértice Paralelo", página 33.
4. Inserción de la nueva tarea i con las mismas tareas sucesoras de cada tarea j del grafo y con las tareas predecesoras de cada tarea predecesora de j (excepto para las tareas que no tengan al menos dos predecesoras, para las cuales se asigna *mutationIsPlausible* = false) según el método descrito en el apartado "Inserción de Vértice Paralelo al Camino entre dos Vértices", página 35.

MÉTODO *FITNESSFUNCTION*

El método *fitnessFunction* recorre sistemáticamente el flujograma empleando el método descrito en el apartado "Recorrido Sistemático de Grafos", página 36. Este método toma la matriz nodo del vértice 0 N^0 y la multiplica por la izquierda por la matriz de conectividad del grafo para ir calculando las matrices de avance.

Los tiempos de inicio y finalización de cada tarea se van calculando en el método *setTransitionTimes*, y se almacenan en las matrices *nodeStart* y *nodeFinish*. Durante los cálculos de *OMSAlgorithm* la tarea 0 es la inicial y la tarea 1 es la final. Por ello la duración total del flujograma será el valor almacenado en *nodeFinish[1]*, ya que es el tiempo de finalización de la tarea final 1.

```
Método fitnessFunction:

pararBucle = false
auxInt = 0;
Inicializar matriz nodo del vértice 0 multMatriz
Inicializar matriz auxiliar auxMatriz a 0

WHILE (NOT pararBucle) {
     FOR (i=0 ; (i<=generation) AND (NOT pararBucle) ; i++) {
          FOR (j=0 ; (j<=generation) AND (NOT pararBucle) ; j++) {

               auxInt = connectivity[i][j]*multMatriz[j]
               auxMatriz[i][j] = auxMatriz[i][j] + auxInt

               IF (auxInt ¡= 0) {
                    pararBucle = setTransitionTimes(i)
               }
          }
     }

     IF (NOT pararBucle) {
          multMatriz = auxMatriz
          auxMatriz = 0

          IF (multMatriz = 0) {
               pararBucle = true
          }
     }
}
RETURN (nodeFinish[1])
```

MÉTODO *SETTRANSITIONTIMES*

El método *setTransitionTimes* calcula los tiempos de transición entre un conjunto de tareas predecesoras y sus tareas sucesoras. Tanto las tareas predecesoras como las sucesoras quedan definidas por una de las tareas sucesoras, que es comunicada al método como el parámetro *i*.

Esta función también actualiza los estados de seguridad del tren durante la transición y los estados de seguridad de las tareas sucesoras.

Devuelve *true* en el caso de detectar una incompatibilidad de estados de seguridad. Actualiza la variable *mutationIsPlausible*.

```
Método setTransitionTimes(i):

malaTransición = false
últimoTiempoFinalVértice = 0

// Cálculo del máximo tiempo final de los vértices predecesores

FOR (j=0 ; j<=generation ; j++) {
     IF ((connectivity[i][j] ¡= 0)AND(nodeFinish[j]>últimoTiempoFinalVértice {
          últimoTimpoFinalVértice = nodeFinish[j]
     }
}

// Cálculo del tiempo máximo de transición entre cada predecesor de i
// y todos sus sucesores

tiempoMáximoTransición = 0

FOR (j=0 ; j<=generation ; j++) {
     IF (connectivity[i][j] ¡= 0) {
          FOR (k=0 ; k<= generation ; k++) {
               IF (connectivity[k][j] ¡= 0) {
                    tiempoTransición =
                    transitionList.timeToChangeStates(nodeStatesAtTask[k] ,
                    nodeStatesAfterTask[j])

                    IF (tiempoTransición < 0) {
                         malaTransición = true
                         mutationIsPlausible = false
                    }
                    ELSE {
                         nodeStatesAtTask[k] = transitionList.stateAtTask
                         nodeStatesAfterTask[k] = transitionList.stateAtTask
                         nodeStatesAfterTask[j] =
                         transitionList.statePreviousToTask

                         IF (tiempoTransición > tiempoMáximoTransición) {
                              tiempoMáximoTransición = tiempoTransición
                         }
                    }
               }
          }
     }
}

// Identifica cada predecesor de i (se le llamará 'j') y a continuación
// cada sucesor de j (se le llamará 'k').
// Para cada pareja de j y k calcula los tiempos de transición (cambio de
// localización más cambio de estado de seguridad) y el estado de seguridad
// del tren tras j y k.

FOR (j=0 ; ((j<=generation)AND(mutationIsPlausible)) ; j++) {
     IF (connectivity[i][j] != 0) {
```

```
          FOR (k=0 ; ((k<=generation)AND(mutationIsPlausible)) ; k++) {
               IF (connectivity[k][j] ¡= 0) {

                    // Tiempo máximo de tránsito a k
                    // desde todos sus predecesores

                    máximoTiempoTránsito = 0;

                    FOR (l=0 ; l<=generation ; l++) {
                         IF (connectivity[k][l] ¡= 0) {
                              tiempoAux =
                              locationList.timeToChangeLocation(k,l)
                              IF (tiempoAux>máximoTiempoTránsito) {
                                   máximoTiempoTránsito = tiempoAux
                              }
                         }
                    }

                    // Calcula tiempo de inicio y finalización
                    // de las tareas sucesoras

                    tiempoAux = últimoTiempoFinalVértice +
                    máximoTiempoTránsito

                    IF (tiempoAux > nodeStart[k]) {
                         nodeStart[k] = tiempoAux
                         nodeFinish[k] = nodeStart[k] + duraciónTarea(k)
                    }

                    // Calcula tiempos de inicio y finalización del cambio
                    // de estado de seguridad

                    stateChangeStart[j] = últimoTiempoFinalVértice
                    stateChangeFinish[j] = últimoTiempoFinalVértice +
                    máximoTiempoTránsito
               }
          }
     }
}

RETURN (malaTransición)
```

MÉTODO *ASSIGNANDCHECKRESOURCESANDSTATES*

Desde este método se preparan los datos y se invocan los métodos que comprueban si el flujograma en curso cumple con las restricciones de recursos y compatibilidad de estados de seguridad.

Si el flujograma cumple las restricciones, se asigna el valor *true* a la variable *mutationIsPlausible*.

El control del cumplimiento de las restricciones y compatibilidades se realiza mediante hitos (*milestones*) de las clases *MilestoneList* y *Milestone*. A cada tarea del flujograma se le asignan cuatro hitos:

- Hito inicial de la tarea.
- Hito final de la tarea.
- Hito inicial de la transición de estado de seguridad tras el final de la tarea.
- Hito final de la transición de estado de seguridad tras el final de la tarea.

Cada hito contiene las siguientes informaciones:

- Tiempo en minutos del hito desde el inicio del flujograma.

- Si el hito es el inicio de una tarea o del cambio de estado.
- El número de tarea al que está vinculado el hito.
- Si es un hito de tarea:
  - Los recursos necesarios para su ejecución.
  - Los estados de seguridad requeridos para su ejecución.
- Si es un hito de procedimiento de cambio de estado de seguridad tras el final de una tarea:
  - Los recursos son todos cero.
  - Los estados de seguridad tras la finalización de la tarea.

La Figura 39 ilustra el concepto de hitos.

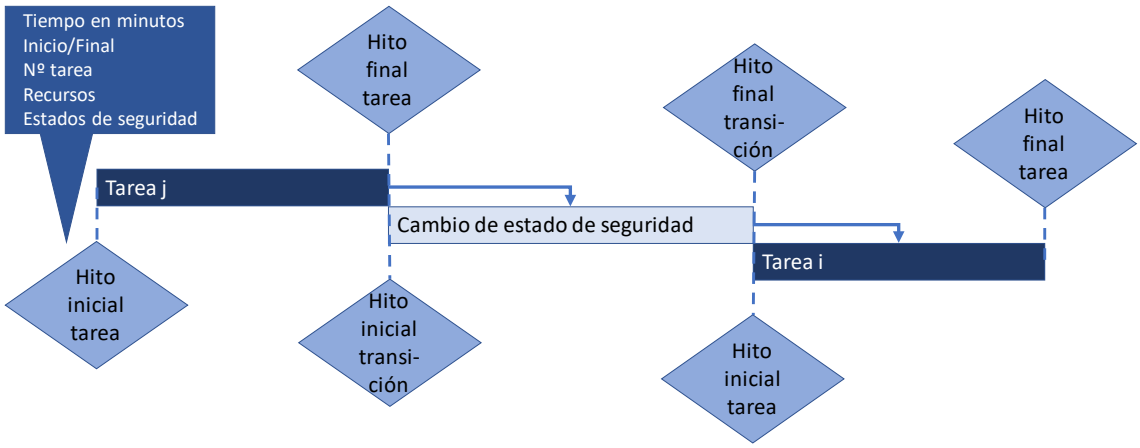


Figura 39 Ilustración del concepto de hitos

El método *assignAndCheckResourcesAndStates* prepara los hitos necesarios e inicializa sus datos para después invocar los métodos que realizarán las comprobaciones (*milestoneList.checkCapacityAndCompatibility*, *checkSuccessors*, y *assignResources*).

El pseudo-código del método *assignAndCheckResourcesAndStates* es como sigue:

```
Método assignAndCheckResourcesAndStates:

Inicializa los hitos de las (generation+1) tareas que componen el flujograma

mutationIsPlausible = milestoneList.CheckCapacityAndCompatibility(generation)

IF (mutationIsPlausible) {
     mutationIsPlausible = checkSuccessors()
}
IF mutationIsPlausible) {
     mutationIsPlausible = assignResources()
}
```

MÉTODO *CHECKSUCCESSORS*

Este método comprueba que todas las tareas marcadas como sucesoras están efectivamente situadas como tales en el flujograma. Devuelve el valor *true* si las tareas sucesoras comienzan después de finalizar las tareas predecesoras.

```
Método checkSuccessors:

resultado = true

FOR (i=0 ; ((i<=generation)AND(resultado)) ; i++) {
     sucesor = leeSucesorTarea(i)
```

```
     IF (sucesor!=”na”) {
          FOR (j=0 ; ((j<=generation)AND(resultado)) ; j++) {
               IF (códigoTarea(j)=sucesor) {
                    IF (nodeFinish[i]>nodeStart[j]) {
                         resultado=false
                    }
               }
          }
     }
}
RETURN (resultado)
```

MÉTODO *ASSIGNRESOURCES*

Este método asigna recursos de manera individual y específica a cada tarea y comprueba si los recursos tienen tiempo suficiente para desplazarse desde la localización de su tarea anterior a la localización de su tarea actual.

El método devuelve *true* si la asignación de recursos ha sido completada exitosamente y si los recursos disponen de tiempo suficiente para sus desplazamientos.

El pseudo-código de este método es:

```
Método assignResources(númeroHitos):

asignaciónOK = true

FOR (i=0 ; ((i<númeroHitos)AND(asignaciónOK)) ; i++) {

     tareaDelHito = milestoneList.copiaTareaDelHito(i)
     esHitoDePrincipioDeTarea = milestoneList.copiaBooleanPrincipioDeHito(i)

     IF (esHitoDePrincipioDeTarea) {

          FOR (j=0 ; ((j<=numberOfResourceGroups)AND(asignaciónOK)) ; j++) {

               recursosNecesarios = milestoneList.copiaRecursos(i,j)
               recursosDisponibles =
               resourceGroupList.copiaRecursosDisponibles(j)
               recursosEncontrados = 0
               continúaBucle = NOT (recursosNecesarios = 0)

               FOR (k=0 ; ((k<recursosDisponibles)AND(continúaBucle)) ; k++){

                    IF (recursosEncontrados<recursosNecesarios) {

                         recursoActualEstáDisponible =
                         resourceGroupList.copiaDisponibilidadRecurso(j,k)
                         recursoTieneTiempoParaDesplazamiento =
                         resourceGroupList.
                         compruebaTiempoDesplazamiento(tareaDelHito,j,k)

                         IF (recursoTieneTiempoParaDesplazamiento AND
                         recursoActualEstáDisponible) {
                              resouceGroupList.
                              reservaRecurso(tareaDelHito,j,k)
                              recursosEncontrados = recursosEncontrados + 1
                         }
                    }
                    ELSE {
                         continúaBucle = false
                    }
               }

               IF (recursosEncontrados<recursosNecesarios) {
```

```
                    asignaciónOK = false
                }
            }
        }
        ELSE {
            FOR (j=0 ; j<numberOfResourceGroups ; j++) {

                recursosDisponibles =
                resourceGroupList.copiaRecursosDisponibles(j)

                FOR (k=0 ; k<recursosDisponibles ; k++) {

                    tareaDelRecurso = resourceGroupList.copiaTareaActual(j,k)

                    IF (tareaDelRecurso = tareaDelHito) {

                        resourceGroupList.eliminarReserva(j,k)
                    }
                }
            }
        }
}
RETURN (asignaciónOK)
```

MÉTODO *OUTPUTBACKLOGINFILE*

El método *outputBacklogInFile* guarda los resultados de la simulación realizada por *OSMAlgorithm* en un fichero en formato CSV con los campos y la estructura definida en el apartado "Salida de Datos" de la página 84.

El fichero de salida se abre desde el método *main* de la clase *OperativeMaintenanceScheduling* y se pasa como parámetro al método outputBacklogInFile en formato *DataOutputStream* (clase estándar de Java). El método *outputBacklogInFile* añade los resultados de la simulación al final del fichero, permitiendo así que se guarden varias simulaciones consecutivas.

Tal como se explicó en el apartado "Algoritmo de Reordenación del Flujograma", página 43, el flujograma resultado de la aplicación de OMSA debe ser reordenado para su mejor comprensión. Esto se realiza con los métodos *sortBacklogOutput* y *sortForQGrid*.

MÉTODO *SORTBACKLOGOUTPUT*

El método *sortBacklogOutput* implementa el algoritmo descrito en el apartado "Algoritmo de Reordenación del Flujograma", página 43. El grafo del flujograma se almacena en la variable *connectivity* de la clase *Backlog*.

MÉTODO *SORTFORQGRID*

El método *sortForQGrid* implementa el algoritmo descrito en el apartado "Proximidad de un Vértice *o* a los Vértices Inicial y al Final del Grafo", página 39, y emplea las series de avances cuadráticos para agrupar vértices con un entorno topológico semejante.

4.3.4 CLASE *TASKLIST*

La clase *TaskList* gestiona los datos relativos a las tareas de mantenimiento a nivel de conjunto. Los datos de cada tarea en particular están almacenados en objetos de la clase *Task*.

La mayor parte de los métodos de la clase *TaskList* se limitan a pasar información entre la clase *Backlog* y los objetos de la clase *Task*.

VARIABLES

En este apartado se presentan las variables más relevantes de la clase.

Variable	Descripción
listLength	int *listLength* Número de tareas en la matriz *taskList*.
resourcesNumber	int *resourcesNumber* Número de tipos distintos de recursos que pueden ser necesarios para la ejecución de una tarea. Esta información se le pasa como parámetro al método constructor de la clase.
taskList	Task[] *taskList* Matriz de objetos de la clase *Task*, esta matriz tiene *listLength* elementos (véase apartado "Clase *Task*", página 73.

Tabla V Variables de la clase TaskList

MÉTODO *SETPRIORITIES*

El método *setPriorities* calcula la prioridad de cada tarea de mantenimiento según el kilometraje recorrido desde la última ejecución de cada tarea. Para ello invoca el método *setPriority* de cada objeto de la clase *Task* contenido en la matriz *taskList*.

```
Método setPriorities(kilometraje):

FOR (i=0 ; i<listLength ; i++) {
     taskList[i].setPriority(kilometraje)
}
```

4.3.5 CLASE *TASK*

Cada objeto de la clase *Task* contiene información acerca de una de las tareas de mantenimiento. Esta información consiste esencialmente en los campos especificados en el apartado "Datos de Tareas de Mantenimiento", página 59.

VARIABLES

En este apartado se presentan las variables de la clase empleadas en el método *setPriority*. El resto de las variables relevantes está descrito en el apartado "Datos de Tareas de Mantenimiento", página 59.

Variable	Descripción
firstExecution	boolean *firstExecution* Esta variable es *true* si la tarea todavía no se ha ejecutado.
interval	Integer *interval* Intervalo en kilómetros de la intervención a la que pertenece la tarea de mantenimiento.
mileageLastExecution	Integer *mileageLastExecution* Kilometraje durante la pasada ejecución de la tarea de mantenimiento.
priority	Integer *priority* Prioridad de ejecución de la tarea: Prioridad 1: Ejecutar inmediatamente. Prioridad 2: Ejecutar si hay tiempo disponible. Prioridad 3: No ejecutar.
thresholdP1	Integer *thresholdP1* Si el kilometraje recorrido desde la última ejecución de la tarea es menor que *thresholdP1*, la prioridad de la tarea es 1.
thresholdP2	Integer *thresholdP2* Si el kilometraje recorrido desde la última ejecución de la tarea es menor que *thresholdP2* y mayor que *thresholdP1*, la prioridad de la tarea es 2. Si el kilometraje recorrido desde la última ejecución de la tarea es mayor que *thresholdP2*, la prioridad de la tarea es 3.

Tabla VI Variables de la clase Task

MÉTODO *SETPRIORITY*

El método *setPriority* calcula la prioridad de una tarea de mantenimiento según el kilometraje recorrido desde su última ejecución.

```
Método setPriority(kilometraje):

delta = interval – (kilometraje – mileageLastExecution)

IF (delta<=thresholdP1) {
     priority = 1
}
ELSE {
     IF ((firstExecution)OR(delta<=thresholdP2)) {
          priority = 2
     }
     ELSE {
          priority = 3
     }
}
```

4.3.6 CLASE *LOCATIONLIST*

La clase *LocationList* gestiona los datos relativos a las localizaciones de las tareas de mantenimiento a nivel de conjunto. Los datos de cada desplazamiento en particular están almacenados en objetos de la clase *Location*.

VARIABLES

En este apartado se presentan las variables más relevantes de la clase.

Variable	Descripción
locationLength	int *locationLength* Número de elementos en la matriz *locationList*.
locationList	Location[] *locationList* Matriz de objetos de la clase *Location*, esta matriz tiene *locationLength* elementos (véase apartado "Clase *Location*", página 75.

Tabla VII Variables de la clase TaskList

MÉTODO *TIMETOCHANGELOCATION*

El método *timeToChangeLocation* calcula y devuelve el tiempo necesario para desplazarse desde la localización de una tarea de mantenimiento hasta la siguiente.

```
Método timeToChangeLocation(localizaciónFinal , localizaciónInicial):

bucleContinúa = true
resultado = 0

FOR (i=0 ; ((i<locationLength)AND(bucleContinúa) ; i++) {
     IF ((locationList[i].startLocation = localizaciónInicial) AND
     (locationList[i].finalLocation = localizaciónFinal)) {
          resultado = locationList[i].duration
          bucleContinúa = false
     }
}
RETURN (resultado)
```

4.3.7 CLASE *LOCATION*

Cada objeto de la clase *Location* contiene información acerca el tiempo de tránsito entre dos localizaciones de mantenimiento. Esta información consiste en los campos especificados en el apartado "Datos de Distancias entre Localizaciones", página 61.

VARIABLES

En este apartado se presentan las variables más relevantes de la clase.

Variable	Descripción
duration	Integer *duration* Duración del tránsito en minutos.
finalLocation	String *finalLocation* Código de la localización de llegada.
startLocation	String *startLocation* Código de la localización de salida.

Tabla VIII Variables de la clase Task

MÉTODOS

Esta clase no tiene ningún método relevante.

4.3.8 CLASE *TRANSITIONLIST*

La clase *TransitionList* gestiona las transiciones entre estados de seguridad de los equipamientos del tren. Dado el estado de seguridad requerido para la ejecución de una tarea y el estado durante la transición previa a dicha tarea, comprueba y ajusta ambos para que sean compatibles.

VARIABLES

En este apartado se presentan las variables más relevantes de la clase.

Variable	Descripción
numberOfStates	int *numberOfStates* Número de diferentes estados de seguridad.
stateAtTask	char[] *stateAtTask* Matriz de *numberOfStates* estados de seguridad requeridos por la tarea en curso.
statePreviousToTask	char[] *statePreviousToTask* Matriz de *numberOfStates* estados de seguridad durante el periodo de transición previo a la tarea en curso
transitionList	Transition[] *transitionList* Matriz de *numberOfStates* objetos de la clase *Transition*. En esta matriz se encuentra la información con los tiempos de transición de estados de seguridad.

Tabla IX Variables de la clase transitionList

MÉTODO *TIMETOCHANGESTATES*

El estado de seguridad de cada equipamiento se codifica como se indica en la Tabla X.

Código	Estado de seguridad
A	El estado de seguridad está activo. Ejemplo: La batería (110 V, DC) está conectada
B	El estado de seguridad está desactivado. Ejemplo: La batería (110 V, DC) está desconectada.
C	El estado de seguridad no influye en la ejecución de la tarea. Ambos estados son compatibles.
D	Sólo durante transiciones: El equipamiento se encuentra en el proceso de pasar de activo a desactivado.
E	Sólo durante transiciones: El equipamiento se encuentra en el proceso de pasar de desactivado a activo.

Tabla X Codificación de posibles estados de seguridad de equipamientos

El método *timeToChangeStates* recibe como parámetros los estados de seguridad requeridos por la tarea en curso más los estados de seguridad previos a la tarea en curso, los actualiza, y calcula y devuelve el tiempo de transición de estados de seguridad. Las diferentes

combinaciones de estados previos y actuales, y sus consecuencias están representadas en la Tabla XI:

statePreviousToTask	próximoEstado	Nuevos estados	Tiempo de transición
Igual a próximoEstado	Distinto de D y E	stateAtTask = próximoEstado statePreviousToTask sin cambios	0
D	B o C	stateAtTask = B statePreviousToTask sin cambios	A➔B
E	A o C	stateAtTask = A statePreviousToTask sin cambios	B➔A
A o B	C	stateAtTask = statePreviousToTask statePreviousToTask sin cambios	0
C	A o B	stateAtTask = próximoEstado statePreviousToTask sin cambios	0
A	B	stateAtTask = B statePreviousToTask = D	A➔B
B	A	stateAtTask = A statePreviousToTask = E	B➔A
Otros valores ➔ **ERROR**			-1

Tabla XI Diferentes combinaciones de estados previos y actuales y de los ajustes que de ellos se siguen

A continuación, se incluye el pseudo-código de este método:

```
Método timeToChangeStates(próximoEstado , previoEstado):

resultado = 0
cambioDeEstadoCompatible = true

statePreviousToTask = previoEstado

FOR (j=0 ; ((j<=numberOfStates)AND(cambioDeEstadoCompatible)) ; j++) {

     IF ((statePreviousToTask[j]=próximoEstado[j]) AND
     (próximoEstado[j]!='D') AND
     (próximoEstado[j]!='E')) {
          resultado = resultado + 0
          stateAtTask[j] = próximoEstado[j]
     }
     ELSE {
          IF ((statePreviousToTask[j]='D') AND
          ((próximoEstado[j]='B') OR
          (próximoEstado[j]='C'))) {
               resultado = resultado + transitionList[j].duraciónAB
               stateAtTask[j] = 'B'
          }
          ELSE {
```

```
                    IF ((statePreviousToTask[j]='E') AND
                    ((próximoEstado[j]='A') OR
                    (próximoEstado[j]='C'))) {
                         resultado = resultado + transitionList[j].duraciónBA
                         stateAtTask[j] = 'A'
                    }
                    ELSE {
                         IF (((statePreviousToTask[j]='A') OR
                         (statePreviousToTask[j]='B')) AND
                         (próximoEstado[j]='C')) {
                              resultado = resultado + 0
                              stateAtTask[j] = statePreviousToTask[j]
                         }
                         ELSE {
                              IF (((próximoEstado[j]='A') OR
                              (próximoEstado[j]='B')) AND
                              (statePreviousToTask[j]='C')) {
                                   resultado = resultado + 0
                                   stateAtTask[j] = próximoEstado[j]
                              }
                              ELSE {
                                   IF ((statePreviousToTask[j]='A') AND
                                   (próximoEstado[j]='B')) {
                                        resultado = resultado +
                                        transitionList[j].durationAB
                                        stateAtTask[j] = próximoEstado[j]
                                        statePreviousToTask[j] = 'D'
                                   }
                                   ELSE {
                                        IF ((statePreviousToTask[j]='B') AND
                                        (próximoEstado[j]='A')) {
                                             resultado = resultado +
                                             transitionList[j].durationBA
                                             stateAtTask[j] = próximoEstado[j]
                                             statePreviousToTask[j] = 'E'
                                        }
                                        ELSE {
                                             cambioDeEstadoCompatible = false
                                             resultado = -1
                                        }
                                   }
                              }
                         }
                    }
               }
          }
     }
}

RETURN (resultado)
```

4.3.9 CLASE *TRANSITION*

Cada objeto de la clase *Transition* contiene el tiempo de transición del estado activo al inactivo y viceversa de un estado de seguridad. Esta información proviene del fichero tal como se especifica en el apartado "Datos de Tiempos de Cambio del Estado de Seguridad del Tren", página 62.

VARIABLES

En este apartado se presentan las variables más relevantes de la clase.

Variable	Descripción
durationAB	Integer *durationAB* Tiempo de transición del estado A al B.
durationBA	Integer *durationBA* Tiempo de transición del estado B al A.

Tabla XII Variables más relevantes de la clase Transition

MÉTODOS

Esta clase no tiene ningún método relevante.

4.3.10 CLASE *MILESTONELIST*

La clase *MilestoneList* gestiona los hitos que se emplean para el control del uso de recursos y de la compatibilidad de estados de seguridad durante la ejecución de las tareas y durante el período de transición tras las mismas.

Los hitos se generan en el método *assignAndCheckResourcesAndStates* de la clase *Backlog* (véase apartado "Método *assignAndCheckResourcesAndStates*", página 69) y se analizan en el método *checkCapacityAndCompatibility* de la clase *MilestoneList*.

VARIABLES

En este apartado se presentan las variables más relevantes de la clase.

Variable	Descripción
maximumAvailableResources	Integer[] *maximumAvailableResources* Matriz de *numberOfResourceTypes* elementos. Cada elemento es la cantidad de recursos disponibles de cada tipo (p.ej. 1 técnico de refrigeración, 2 técnicos de bogies, etc.).
milestoneList	Milestone[] *milestoneList* Matriz de 4 veces la cantidad de tareas de mantenimiento. En cada elemento de la matriz se guardan los datos de un hito.
numberOfResourceTypes	int *numberOfResourceTypes* Número de diferentes tipos de cualificaciones de recursos.
numberOfStateTypes	int *numberOfStateTypes* Número de diferentes estados de seguridad.

Tabla XIII Variables más relevantes de la clase MilestoneList

MÉTODO *CHECKCAPACITYANDCOMPATIBILITY*

El método *checkCapacityAndCompatibility* es invocado desde el método *AssignAndCheckResourcesAndStates* de la clase *Backlog*, en el que se guardan los datos de las tareas de mantenimiento y de los cambios de estado posteriores. Hay un total de cuatro hitos por tarea de mantenimiento:

- Hito inicial de la tarea de mantenimiento.
- Hito final de la tarea.
- Hito inicial del cambio de estado tras finalizar la tarea de mantenimiento.
- Hito final del cambio de estado.

Los hitos son ordenados cronológicamente. En el caso de que dos hitos tengan el mismo tiempo se colocan en primer lugar los hitos de finalización, sean de una tarea propiamente dicha o de un cambio de estado.

El análisis posterior se ilustra con la Figura 40. Los hitos contienen las cantidades necesarias de cada recurso para cada tarea y sus correspondientes estados. Analizando los hitos por orden cronológico es posible determinar las necesidades de cada recurso y si en algún momento sobrepasan el límite de recursos disponibles. En la ilustración se aprecia que inicialmente sólo está activa la tarea l, que requiere de 2 técnicos. Siendo 3 el número de técnicos disponibles no hay ningún problema en la ejecución. Poco después comienza la tarea j, que requiere el concurso de un técnico adicional. El número total de técnicos requeridos asciende a 3, todavía dentro de las posibilidades. Durante el cambio de estado no se requiere ningún recurso y la cantidad de técnicos se reduce a 0. El problema aparece cuando concurren las tareas i y k, precisándose un total de 4 técnicos, uno más de los disponibles.

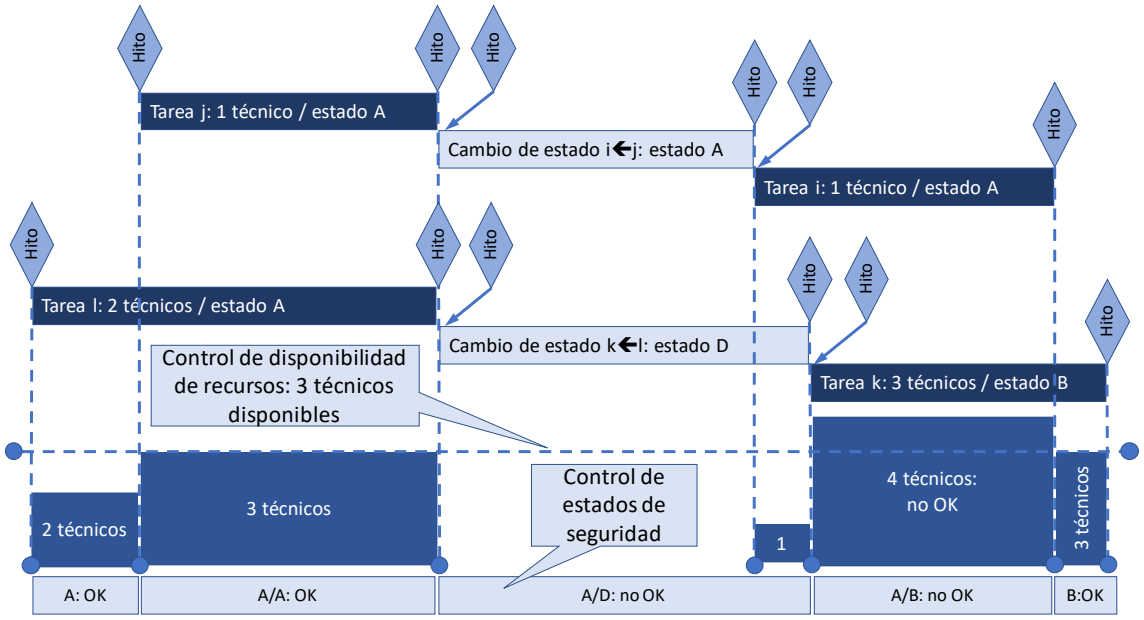


Figura 40 Ilustración del método de control de disponibilidad de recursos y de compatibilidad de estados de seguridad

El procedimiento con los estados es semejante. Por cada tipo de estado de seguridad (batería conectada/desconectada, catenaria activa/desactivada, etc.) existen cinco contadores, uno por cada código (A, B, C, D y E). En la ilustración de la Figura 40 se considera sólo un tipo de estado de seguridad para simplificar el razonamiento. Si comienza una tarea con código A se incrementa el contador de código A en una unidad. Al comenzar una segunda tarea con el código A, su contador pasa a ser 2. Cuando ambas tareas finalizan el contador disminuye en dos unidades. Si en un momento dado los contadores de dos códigos incompatibles (p.ej. A y D, o A y B) son distintos de 0, tendremos una incompatibilidad de estados de seguridad. Es lo que ocurre en la ilustración durante los cambios de estado y durante el transcurso en paralelo de las tareas i y k. En resumen, los códigos A, B, D y E son incompatibles entre sí y sólo uno de sus contadores puede ser distinto de cero. El código C es compatible con todos.

El método devuelve *true* si no se superan los recursos disponibles y los estados de seguridad son compatibles.

Pseudo-código del método *checkCapacityAndCompatibility*:

```
Método checkCapacityAndCompatibility(númeroDeHitos):

capazYCompatible = true

ordenaHitosCronológicamente

FOR (i=0 ; ((i<númeroDeHitos) AND (capazYCompatible)) ; i++) {

     FOR (j=0 ; ((j<numberOfResourceTypes) AND (capazYCompatible)) ; j++) {

          IF (milestoneList[i].startTime) {
               recursosAcumulados[j] = recursosAcumulados[j] +
               milestoneList[i].recursos[j]
          }
          ELSE {
               recursosAcumulados[j] = recursosAcumulados[j] -
               milestoneList[i].recursos[j]
          }

          IF (recursosAcumulados[j] > maximumAvailableResources[j]) {
               capazYCompatible = false
          }
     }

     FOR (j=0 ; ((j<numberOfStateTypes) AND (capazYCompatible)) ; j++) {

          estadoAux = milestoneList[i].states[j]

          IF (milestoneList[i].startTime) {
               IF (estadoAux = 'A') {
                    estadosAcumulados[j][0]++
               }
               IF (estadoAux = 'B') {
                    estadosAcumulados[j][1]++
               }
               IF (estadoAux = 'C') {
                    estadosAcumulados[j][2]++
               }
               IF (estadoAux = 'D') {
                    estadosAcumulados[j][3]++
               }
               IF (estadoAux = 'E') {
                    estadosAcumulados[j][4]++
               }
          }
          ELSE {
               IF (estadoAux = 'A') {
                    estadosAcumulados[j][0]--
               }
               IF (estadoAux = 'B') {
                    estadosAcumulados[j][1]--
               }
               IF (estadoAux = 'C') {
                    estadosAcumulados[j][2]--
               }
               IF (estadoAux = 'D') {
                    estadosAcumulados[j][3]--
               }
               IF (estadoAux = 'E') {
                    estadosAcumulados[j][4]--
               }
          }

          capazYCompatible = ((estadosAcumulados[j][0]=0) AND
               (estadosAcumulados[j][1]=0) AND
               (estadosAcumulados[j][3]=0)) OR
```

```
                ((estadosAcumulados[j][0]=0) AND
                (estadosAcumulados[j][1]=0) AND
                (estadosAcumulados[j][4]=0)) OR
                ((estadosAcumulados[j][0]=0) AND
                (estadosAcumulados[j][3]=0) AND
                (estadosAcumulados[j][4]=0)) OR
                ((estadosAcumulados[j][1]=0) AND
                (estadosAcumulados[j][3]=0) AND
                (estadosAcumulados[j][4]=0))
    }
}

RETURN (capazYCompatible)
```

4.3.11 CLASE *MILESTONE*

Los objetos de la clase *Milestone* contienen las informaciones de los hitos.

VARIABLES

En este apartado se presentan las variables más relevantes de la clase.

Variable	Descripción
startTime	boolean *startTime* Esta variable es *true* si el hito pertenece al comienzo de una tarea o de un período de transición.
states	char[] *states* Estados de seguridad almacenados en el hito.
resources	Integer[] *resources* Recursos necesarios para la ejecución de la tarea de mantenimiento
time	int *time* Tiempo en minutos asociado al hito.

Tabla XIV Variables más relevantes de la clase Milestone

MÉTODOS

La clase *Milestone* no contiene ningún método relevante para la descripción de la aplicación informática.

4.3.12 CLASE *RESOURCEGROUPLIST*

La clase *ResourceGroupList* gestiona globalmente los datos de los recursos y sus cualificaciones. La Figura 41 ilustra un ejemplo de la estructura de datos.

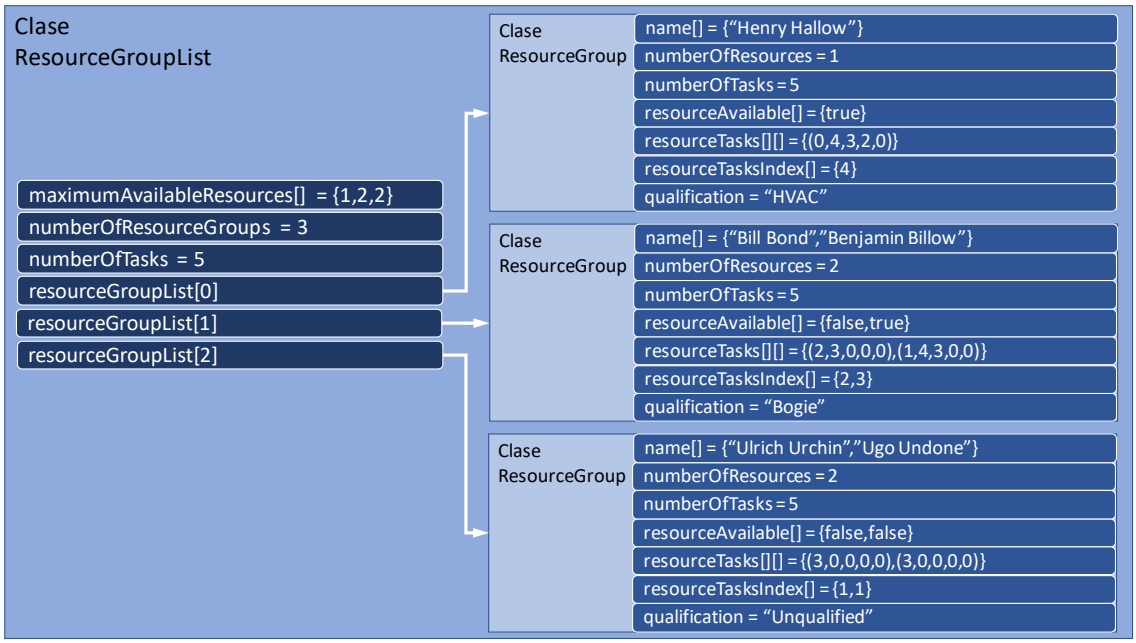


Figura 41 Ejemplo ilustrativo de la estructura de datos de las clases ResourceGroupList y GroupList

VARIABLES

En este apartado se presentan las variables más relevantes de la clase.

Variable	Descripción
maximumAvailableResources	int[] *maximumAvailableResources* Matriz de *numberOfResourceGroups* elementos. Cada elemento representa el número total de recursos disponibles de un grupo.
numberOfResourceGroups	int *numberOfResourceGroups* Número de grupos de recursos por cualificación.
numberOfTasks	int *numberOfTasks* Número máximo de tareas de mantenimiento. Esta información proviene de un objeto de la clase *TaskList* a través del objeto de la clase *Backlog*.
resourceGroupList	ResourceGroup[] *resourceGroupList* Matriz de *numberOfResourceGroups* elementos. Cada elemento contiene la cualificación de los recursos del grupo, los nombres de los recursos, si están actualmente disponibles y en qué tareas va a participar cada recurso.

Tabla XV Variables más relevantes de la clase ResourceGroupList

MÉTODOS

La clase *ResourceGroupList* no contiene ningún método relevante para la descripción de la aplicación informática.

4.3.13 CLASE *RESOURCEGROUP*

La clase *ResourceGroup* gestiona datos de los grupos de recursos. La Figura 41 ilustra un ejemplo de la estructura de datos.

VARIABLES

En este apartado se presentan las variables más relevantes de la clase.

Variable	Descripción
name	String[] *name* Matriz de *numberOfResources* elementos. Cada elemento contiene el nombre de un recurso con la cualificación de este grupo de recursos.
numberOfResources	int *numberOfResources* Número de recursos con la cualificación de este grupo de recursos.
numberOfTasks	int *numberOfTasks* Número máximo de tareas de mantenimiento que pueden ser asignadas a cada recurso.
resourceAvailable	boolean[] *resourceAvailable* Matriz de *numberOfResources* elementos. Cada elemento indica si el recurso está disponible (*true*) o no (*false*).
resourceTasks	int[][] *resourceTasks* Matriz de *numberOfResources x numberOfTasks* elementos. Cada elemento de subíndices (i,j) contiene el número identificativo de la tarea que realizará el recurso 'i' en el ordinal 'j'.
resourceTasksIndex	int[] *resourceTasksIndex* Matriz de *numberOfResources* elementos. Para cada recurso del objeto ResourceGroup, el elemento correspondiente de esta matriz es el índice de la siguiente posición libre de la matriz *resourceTasks*.
qualification	char[] *qualification* Qualification of the resources of this class.

Tabla XVI Variables más relevantes de la clase ResourceGroup

MÉTODOS

La clase *ResourceGroup* no contiene ningún método relevante para la descripción de la aplicación informática.

4.3.14 SALIDA DE DATOS

La aplicación informática entrega los resultados de OMSA en un fichero en formato CSV (*comma separated values*, valores separados por comas) idéntico al de los ficheros de entrada de datos.

Para cada ejecución del método *outputBackLogInfile* (véase página 72) se guarda en el mismo fichero en primer lugar información acerca del flujograma calculado y en segundo lugar información personalizada para cada recurso con sus tareas.

Los campos guardados relativos al flujograma por cada tarea realizada son los siguientes:

Campo	Descripción
Mileage Kilometraje	Kilometraje durante la estadía simulada.
Code Código	Código identificativo de la tarea de mantenimiento.
Description Descripción	Descripción de la tarea de mantenimiento.
Start Inicio	Tiempo inicial en minutos de la tarea de mantenimiento.
Finish Final	Tiempo final en minutos de la tarea de mantenimiento.
Task duration Duración de la tarea	Duración en minutos de la tarea de mantenimiento.
Location Localizacion	Código de localización de la tarea de mantenimiento (véase "Datos de Distancias entre Localizaciones", página 61).
Counter Contador	Contador de las tareas realizadas durante la estadía de mantenimiento.
Predecessors Predecesores	Tareas predecesoras de la actual, se usa como referencia el campo *Counter* arriba mencionado.
Resources Recursos	Recursos necesarios para la ejecución de la tarea de mantenimiento. La cantidad de personal de cada tipo de cualificación está separada por una coma.
States Estados de seguridad	Codificación del estado de seguridad requerido para la ejecución de la tarea de mantenimiento.
Interval Intervalo	Intervalo en kilómetros de la intervención a la que pertenece la tarea de mantenimiento.
Mileage since last maintenance Kilometraje desde el último mantenimiento	Kilometraje recorrido desde la última estadía en la que se ha ejecutado esta tarea de mantenimiento.
Mileage at last maintenance Kilometraje en el mantenimiento previo	Kilometraje en el que se ha realizado el mantenimiento previo.
Priority Prioridad	Prioridad de la tarea de mantenimiento.
Successor Sucesor	Código de la tarea de mantenimiento que se debe realizar obligatoriamente con la tarea actual dentro de la misma estadía.
Connectivity matrix Matriz de conectividad	Datos de la matriz de conectividad del flujograma

Tabla XVII Salida de datos: informaciones del flujograma

Adicionalmente se guarda en otra fila el kilometraje, el tiempo total de ejecución del flujograma y el tiempo máximo disponible asignado a la estadía.

Los campos guardados relativos a las tareas de cada recurso son los siguientes:

Campo	Descripción
Resource Recurso	Nombre del recurso
Code Código	Código identificativo de la tarea de mantenimiento.
Description Descripción	Descripción de la tarea de mantenimiento.
Start Inicio	Tiempo inicial en minutos de la tarea de mantenimiento.
Finish Final	Tiempo final en minutos de la tarea de mantenimiento.
Task duration Duración de la tarea	Duración en minutos de la tarea de mantenimiento.
Location Localizacion	Código de localización de la tarea de mantenimiento (véase "Datos de Distancias entre Localizaciones", página 61).

Tabla XVIII Salida de datos: información acerca de las tareas de cada recurso

4.4 VALIDACIÓN DEL MÉTODO

4.4.1 COMPARACIÓN CON UN CASO REAL

La validación del método se ha realizado comparando el flujograma de una estadía real realizado por un planificador experimentado de mantenimiento y el resultado arrojado por la aplicación informática.

El caso real consistió en la realización de parte de 62 tareas pertenecientes a la intervención de 100.000 km (intervención M1) de un tren de alta velocidad de ocho coches y 200 m de longitud. A las 62 tareas se les han añadido una tarea inicial más una tarea final de 5 minutos cada una que sirven de inicio y fin del grafo del flujograma.

El taller donde se realizó la intervención no cumplía con los requisitos habituales para realizar el mantenimiento de trenes de alta velocidad. En vez de proporcionar una vía con foso y acceso al techo a lo largo de toda la longitud del tren, tan solo se disponía de una vía con un foso con capacidad para un coche y de una segunda vía adicional con una plataforma de acceso al techo para un solo coche.

Esta severa limitación de la infraestructura disponible obligaba a desplazar el tren con una locomotora de maniobras siempre que el equipo de mantenimiento cambiaba de coche o accedía al techo. La mayor parte del tiempo de ejecución de una maniobra de desplazamiento se emplea en la preparación del tren, en el acoplamiento de la locomotora de maniobras y en las tareas de aseguramiento del tren una vez en su posición de destino. El tiempo de desplazamiento varía marginalmente entre mover el tren una longitud de uno o dos coches.

El modelado de estas restricciones se hizo definiendo un estado de seguridad que indica si el tren se encuentra en la vía con acceso al techo (valor A del estado) o no (valor B), más un estado de seguridad por cada coche que indica si el coche en cuestión está situado en la zona de acceso y se puede trabajar en él (valor A) o no (valor B).

Como se puede observar en la Figura 42, si un tren se encuentra en la vía con foso y el coche 8 se encuentra sobre la zona de foso su estado sería BBBBBBBBA, significando la primera B que el tren no está en la vía de acceso al techo, las siguientes 7 Bs significan que no se puede trabajar en los coches 1 a 7, y la A final significa que el coche 8 es en el que se puede trabajar y se encuentra sobre el foso.

Análogamente, el tren representado en la Figura 42 sobre la vía con plataforma de acceso al techo tendrá el estado ABBBBBABB.

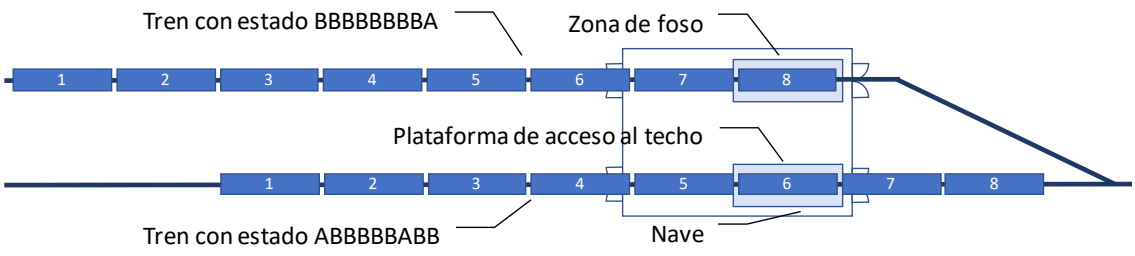


Figura 42 Ilustración estados de seguridad de vía con plataforma y de acceso a coches

En el modelo se han empleado un total de once estados de seguridad:

Estado de seguridad	Tiempo A→B	Tiempo B→A
Batería de 110 V DC conectada	10 min	10 min
Alimentación neumática exterior conectada	5 min	5 min
Tren en vía con plataforma de acceso al techo	1 min	1 min
Coche 1 accesible para realizar tareas de mantenimiento	1 min	1 min
Coche 2 accesible para realizar tareas de mantenimiento	1 min	1 min
Coche 3 accesible para realizar tareas de mantenimiento	1 min	1 min
Coche 4 accesible para realizar tareas de mantenimiento	1 min	1 min
Coche 5 accesible para realizar tareas de mantenimiento	1 min	1 min
Coche 6 accesible para realizar tareas de mantenimiento	1 min	1 min
Coche 7 accesible para realizar tareas de mantenimiento	1 min	1 min
Coche 8 accesible para realizar tareas de mantenimiento	1 min	1 min

Tabla XIX Estados de seguridad del caso real para validación de la aplicación

Los tiempos de cambio de estado de seguridad de la vía con acceso al techo y de acceso a cada coche figuran con tiempos de transición de un minuto en vez de valores cercanos a la hora, tal como se apuntaba al presentar este caso. Ello se debe a que la aplicación informática desarrollada obliga a emplear un tiempo de transición distinto de cero y se ha tomado el mínimo valor entero.

Los estados de seguridad se emplean en este modelo para asegurar que sólo se trabaja en el coche que esté situado sobre el foso o junto a la plataforma de acceso al techo.

Como consecuencia de que los tiempos de cambio de estado han de ser distintos de cero, los tiempos de desplazamiento desde un puesto de trabajo al siguiente han sido ajustados. Suponiendo que el equipo de mantenimiento ha finalizado sus tareas bajo bastidor del coche 3 con alimentación de batería y sin alimentación neumática (estado ABBBBABBBBB) y pasa a trabajar en el techo del coche 4 con batería y sin alimentación neumática (estado ABABBBABBBB), tendríamos un total de tres cambios de estado (acceso a techo de B a A, coche 3 de A a B, y coche 4 de B a A) con lo que el algoritmo computaría un tiempo de transición de 3 min. Si el tiempo total real es de 60 minutos, entonces la tabla de desplazamiento deberá devolver 57 minutos para el desplazamiento de la localización 340 (bajo bastidor del coche 3) a la localización 410 (techo del coche 4).

La Figura 43 muestra los tiempos de desplazamiento entre localizaciones empleados para el modelado del caso de validación de manera que compensen los tiempos de cambio de estado.

Area	Coche	Localización	110	120	130	140	210	220	230	240	310	320	330	340	410	420	430	440	510	520	530	540	610	620	630	640	710	720	730	740	810	820	830	840
bajo bast. [040]	8	840	63	64	64	64	62	63	63	63	61	62	62	62	60	61	61	61	59	60	60	60	58	59	59	59	57	58	58	58	59	0	0	0
exterior [030]	8	830	63	64	64	64	62	63	63	63	61	62	62	62	60	61	61	61	59	60	60	60	58	59	59	59	57	58	58	58	59	0	0	
interior [020]	8	820	63	64	64	64	62	63	63	63	61	62	62	62	60	61	61	61	59	60	60	60	58	59	59	59	57	58	58	58	59	0		
techo [010]	8	810	64	63	63	63	63	62	62	62	62	61	61	61	61	60	60	60	60	59	59	59	59	58	58	58	58	57	57	57	0			
bajo bast. [040]	7	740	62	63	63	63	61	62	62	62	60	61	61	61	59	60	60	60	58	59	59	59	57	58	58	58	59	0	0	0				
exterior [030]	7	730	62	63	63	63	61	62	62	62	60	61	61	61	59	60	60	60	58	59	59	59	57	58	58	58	59	0	0					
interior [020]	7	720	62	63	63	63	61	62	62	62	60	61	61	61	59	60	60	60	58	59	59	59	57	58	58	58	59	0						
techo [010]	7	710	63	62	62	62	62	61	61	61	61	60	60	60	60	59	59	59	59	58	58	58	58	57	57	57	0							
bajo bast. [040]	6	630	61	62	62	62	60	61	61	61	59	60	60	60	58	59	59	59	57	58	58	58	59	0	0	0								
exterior [030]	6	620	61	62	62	62	60	61	61	61	59	60	60	60	58	59	59	59	57	58	58	58	59	0	0									
interior [020]	6	620	61	62	62	62	60	61	61	61	59	60	60	60	58	59	59	59	57	58	58	58	59	0										
techo [010]	6	610	62	61	61	61	61	60	60	60	60	59	59	59	59	58	58	58	58	57	57	57	0											
bajo bast. [040]	5	540	60	61	61	61	59	60	60	60	58	59	59	59	57	58	58	58	59	0	0	0												
exterior [030]	5	530	60	61	61	61	59	60	60	60	58	59	59	59	57	58	58	58	59	0	0													
interior [020]	5	520	60	61	61	61	59	60	60	60	58	59	59	59	57	58	58	58	59	0														
techo [010]	5	510	61	60	60	60	60	59	59	59	59	58	58	58	58	57	57	57	0															
bajo bast. [040]	4	440	59	60	60	60	58	59	59	59	57	58	58	58	59	0	0	0																
exterior [030]	4	430	59	60	60	60	58	59	59	59	57	58	58	58	59	0	0																	
interior [020]	4	420	59	60	60	60	58	59	59	59	57	58	58	58	59	0																		
techo [010]	4	410	60	59	59	59	59	58	58	58	58	57	57	57	0																			
bajo bast. [040]	3	340	58	59	59	59	57	58	58	58	59	0	0	0																				
exterior [030]	3	330	58	59	59	59	57	58	58	58	59	0	0																					
interior [020]	3	320	58	59	59	59	57	58	58	58	59	0																						
techo [010]	3	310	59	58	58	58	58	57	57	57	0																							
bajo bast. [040]	2	240	57	58	58	58	59	0	0	0																								
exterior [030]	2	230	57	58	58	58	59	0	0																									
interior [020]	2	220	57	58	58	58	59	0																										
techo [010]	2	210	58	57	57	57	0																											
bajo bast. [040]	1	140	59	0	0	0																												
exterior [030]	1	130	59	0	0																													
interior [020]	1	120	59	0																														
techo [010]	1	110	0																															

Figura 43 Tiempos de desplazamiento entre localizaciones en el caso real de la validación

Los recursos previstos en el caso real eran dos especialistas en bogies y frenos más un técnico generalista de mantenimiento. En los datos de entrada los especialistas en bogies y frenos se les denominó “Benito” y “Bernardo”, y al técnico generalista “Gerardo”.

La lista de tareas es la siguiente:

Código	Descripción	Duración	Estados	Loc.	Interv.	Prio.1	Prio.2	Bog.	Gen.	Suce.
0	INITIAL	5	ABBABBBBBBB	120	10	1	2	0	1	na
9999	FINAL	5	ABBBBBBBBBA	820	10	1	2	0	1	na
1	Control de dotacion del tren	20	ACBABBBBBBB	120	10	1	2	0	1	na
2	Control de dotacion del tren	20	ACBBABBBBBB	220	10	1	2	0	1	na
3	Control de dotacion del tren	20	ACBBBABBBBB	320	10	1	2	0	1	na
4	Control de dotacion del tren	20	ACBBBBABBBB	420	10	1	2	0	1	na
5	Control de dotacion del tren	20	ACBBBBBABBB	520	10	1	2	0	1	na
6	Control de dotacion del tren	20	ACBBBBBBABB	620	10	1	2	0	1	na
7	Control de dotacion del tren	20	ACBBBBBBBAB	720	10	1	2	0	1	na
8	Control de dotacion del tren	20	ACBBBBBBBBA	820	10	1	2	0	1	na
9	Inspeccion de faldones laterales y de tapas bajo bastidor	30	CCBABBBBBBB	130	10	1	2	0	1	na
10	Inspeccion de faldones laterales y de tapas bajo bastidor	30	CCBBABBBBBB	230	10	1	2	0	1	na
11	Inspeccion de faldones laterales y de tapas bajo bastidor	30	CCBBBABBBBB	330	10	1	2	0	1	na
12	Inspeccion de faldones laterales y de tapas bajo bastidor	30	CCBBBBABBBB	430	10	1	2	0	1	na
13	Inspeccion de faldones laterales y de tapas bajo bastidor	30	CCBBBBBABBB	530	10	1	2	0	1	na
14	Inspeccion de faldones laterales y de tapas bajo bastidor	30	CCBBBBBBABB	630	10	1	2	0	1	na
15	Inspeccion de faldones laterales y de tapas bajo bastidor	30	CCBBBBBBBAB	730	10	1	2	0	1	na
16	Inspeccion de faldones laterales y de tapas bajo bastidor	30	CCBBBBBBBBA	830	10	1	2	0	1	na

Código	Descripción	Duración	Estados	Loc.	Interv.	Prio.1	Prio.2	Bog.	Gen.	Suce.
17	Inspeccion visual de ventanas	30	CCBABBBBBBB	120	10	1	2	0	1	na
18	Inspeccion visual de ventanas	30	CCBBABBBBBB	220	10	1	2	0	1	na
19	Inspeccion visual de ventanas	30	CCBBBABBBBB	320	10	1	2	0	1	na
20	Inspeccion visual de ventanas	30	CCBBBBABBBB	420	10	1	2	0	1	na
21	Inspeccion visual de ventanas	30	CCBBBBBABBB	520	10	1	2	0	1	na
22	Inspeccion visual de ventanas	30	CCBBBBBBABB	620	10	1	2	0	1	na
23	Inspeccion visual de ventanas	30	CCBBBBBBBAB	720	10	1	2	0	1	na
24	Inspeccion visual de ventanas	30	CCBBBBBBBBA	820	10	1	2	0	1	na
25	Inspeccion de bogies	90	CBBABBBBBBB	140	10	1	2	2	0	na
26	Inspeccion de bogies	90	CBBBABBBBBB	240	10	1	2	2	0	na
27	Inspeccion de bogies	90	CBBBBABBBBB	340	10	1	2	2	0	na
28	Inspeccion de bogies	90	CBBBBBABBBB	440	10	1	2	2	0	na
29	Inspeccion de bogies	90	CBBBBBBABBB	540	10	1	2	2	0	na
30	Inspeccion de bogies	90	CBBBBBBBABB	640	10	1	2	2	0	na
31	Inspeccion de bogies	90	CBBBBBBBBAB	740	10	1	2	2	0	na
32	Inspeccion de bogies	90	CBBBBBBBBBA	840	10	1	2	2	0	na
33	Inspeccion de pantografo 25 kV AC	40	BBABBBABBBB	410	10	1	2	0	1	na
34	Mantenimiento de seccionador de pantografo AC	40	BBABBBABBBB	410	10	1	2	0	1	na
35	Inspeccion de pantografo 25 kV AC	40	BBABBBBABBB	510	10	1	2	0	1	na
36	Mantenimiento de seccionador de pantografo AC	40	BBABBBBABBB	510	10	1	2	0	1	na
37	Inspeccion y mantenimiento de disyuntor principal 25 kV AC	40	BBABBBABBBB	410	10	1	2	0	1	na
38	Inspeccion y mantenimiento de disyuntor principal 25 kV AC	40	BBABBBBABBB	510	10	1	2	0	1	na
39	Mantenimiento de seccionador de linea de techo de 25 kV AC	40	BBABBBABBBB	410	10	1	2	0	1	na
40	Mantenimiento de seccionador de linea de techo de 25 kV AC	40	BBABBBBABBB	510	10	1	2	0	1	na
41	Inspeccion de transductor de alta tension	40	BBABBBABBBB	410	10	1	2	0	1	na
42	Inspeccion de transductor de alta tension	40	BBABBBBABBB	510	10	1	2	0	1	na
43	Inspeccion de transductor de corriente de red	40	BBABBBABBBB	410	10	1	2	0	1	na
44	Inspeccion de transductor de corriente de red	40	BBABBBBABBB	510	10	1	2	0	1	na
45	Inspeccion y mantenimiento de los acoplamientos de transmision	20	CCBABBBBBBB	140	10	1	2	2	0	na
46	Inspeccion y mantenimiento de los acoplamientos de transmision	20	CCBBBABBBBB	340	10	1	2	2	0	na
47	Inspeccion y mantenimiento de los acoplamientos de transmision	20	CCBBBBBBABB	640	10	1	2	2	0	na
48	Inspeccion y mantenimiento de los acoplamientos de transmision	20	CCBBBBBBBBA	840	10	1	2	2	0	na
49	Inspeccion y mantenimiento de las cajas reductoras	20	CCBABBBBBBB	140	10	1	2	2	0	na
50	Inspeccion y mantenimiento de las cajas reductoras	20	CCBBBABBBBB	340	10	1	2	2	0	na
51	Inspeccion y mantenimiento de las cajas reductoras	20	CCBBBBBBABB	640	10	1	2	2	0	na
52	Inspeccion y mantenimiento de las cajas reductoras	20	CCBBBBBBBBA	840	10	1	2	2	0	na
53	Inspeccion del sistema de freno	140	ACBABBBBBBB	140	10	1	2	2	0	na
54	Inspeccion del sistema de freno	140	ACBBABBBBBB	240	10	1	2	2	0	na

Código	Descripción	Duración	Estados	Loc.	Interv.	Prio.1	Prio.2	Bog.	Gen.	Suce.
55	Inspeccion del sistema de freno	140	ACBBBABBBBB	340	10	1	2	2	0	na
56	Inspeccion del sistema de freno	140	ACBBBBABBBB	440	10	1	2	2	0	na
57	Inspeccion del sistema de freno	140	ACBBBBBABBB	540	10	1	2	2	0	na
58	Inspeccion del sistema de freno	140	ACBBBBBBABB	640	10	1	2	2	0	na
59	Inspeccion del sistema de freno	140	ACBBBBBBBAB	740	10	1	2	2	0	na
60	Inspeccion del sistema de freno	140	ACBBBBBBBBA	840	10	1	2	2	0	na
61	Inspeccion visual de equipamiento de cabina	20	ACBABBBBBBB	120	10	1	2	0	1	na
62	Inspeccion visual de equipamiento de cabina	20	ACBBBBBBBBA	820	10	1	2	0	1	na

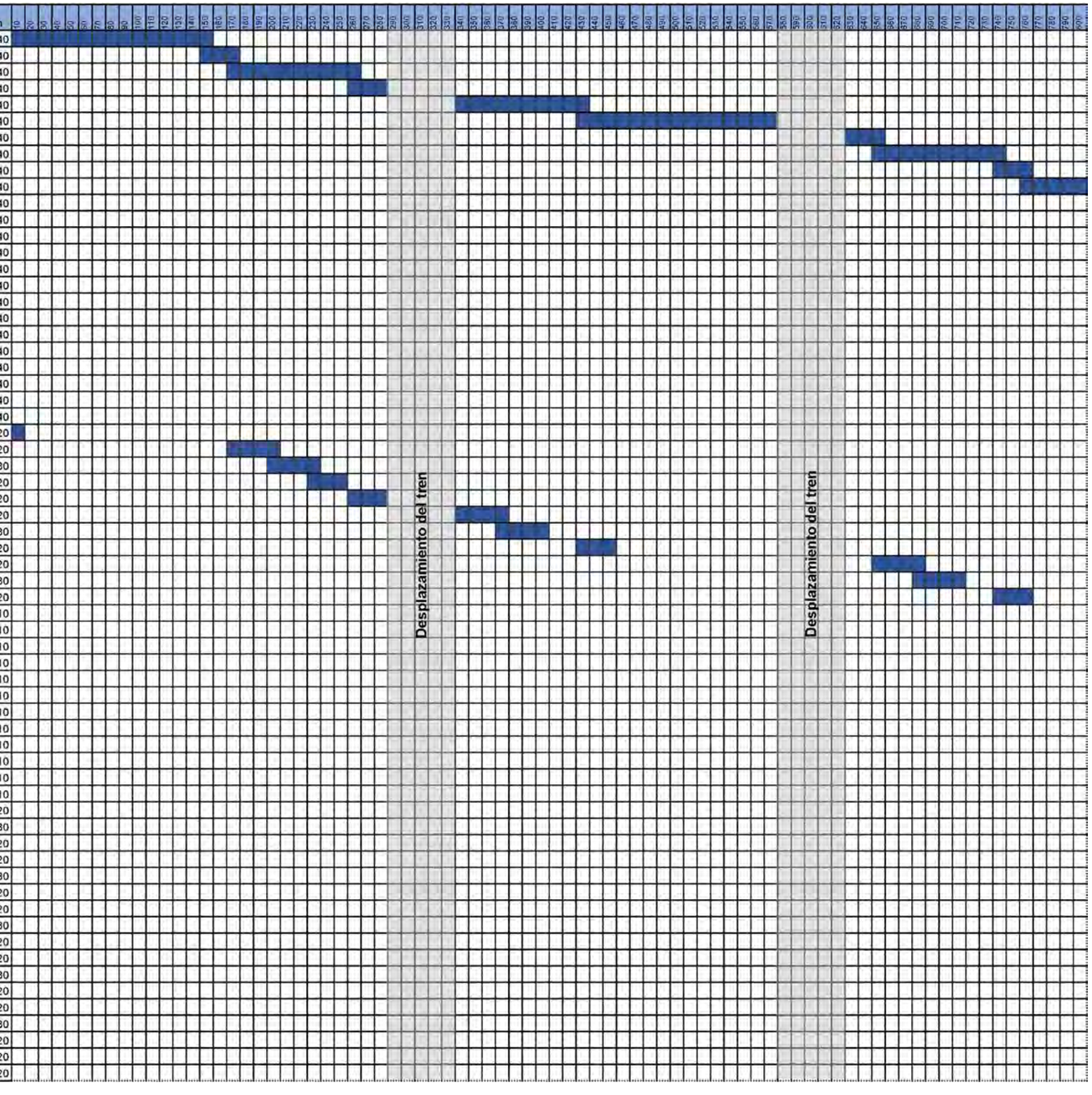
Tabla XX Tareas del caso real para validación de la aplicación

La aplicación informática se ejecutó en 27 segundos y la duración total del flujograma calculado fue de 3.050 minutos. El planificador de mantenimiento había calculado 3.020 minutos, pero su flujograma no incluía las tareas “INITIAL” ni “FINAL”, de 5 minutos cada una, ni los tiempos necesarios para la conexión y desconexión de la batería, de 10 minutos cada operación, quedando así explicada la diferencia de 30 minutos entre ambos cálculos.

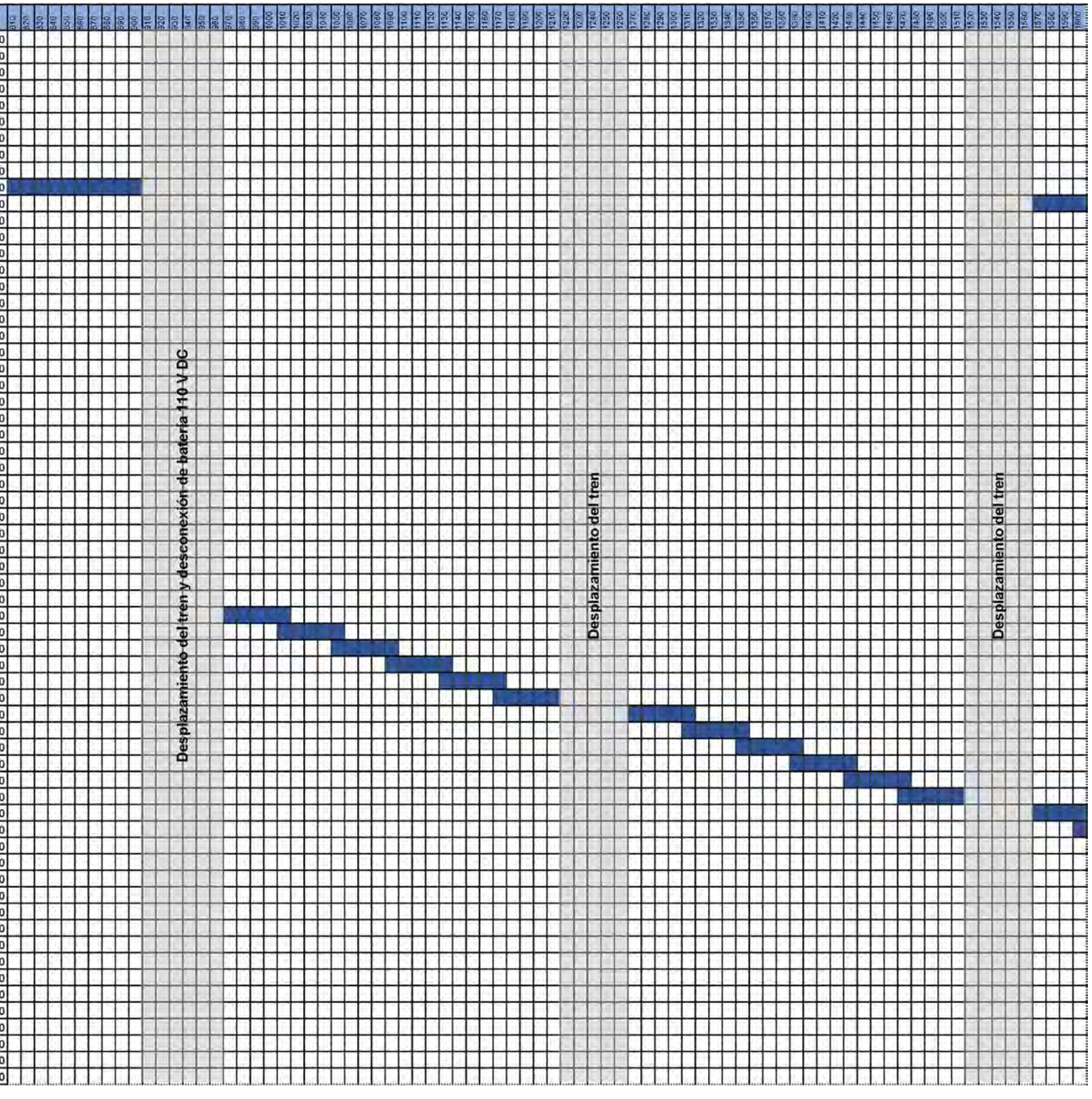
En las siguientes páginas se muestra las tareas asignadas a uno de los especialistas de bogies (ambos tienen las mismas tareas asignadas) y al técnico generalista de mantenimiento. El algoritmo generó un flujograma muy semejante al del planificador de mantenimiento:

- Coche 1: Trabajos con batería dentro del coche y bajo bastidor, trabajando en paralelo los dos especialistas de bogies con el técnico generalista.
- Coche 2: Como coche 1, teniendo en cuenta que este coche no tiene ni acoplamientos, ni caja reductoras ni cabina.
- Coche 3: Como coche 1, sin cabina.
- Coche 4: Desconexión de batería y traslado a la plataforma de acceso al techo, trabajos en el techo.
- Coche 5: Plataforma de acceso al techo, trabajos en el techo.
- Coche 4: Traslado vía de foso, trabajos bajo bastidor, reconexión de batería, trabajos en el interior. Coche 4 no tiene ni acoplamientos ni cajas reductoras.
- Coche 5: Trabajos bajo bastidor y en el interior, este coche tampoco tiene acoplamientos ni cajas reductoras.
- Coche 6: Trabajos bajo bastidor y en el interior, este coche tiene acoplamientos y cajas reductoras.
- Coche 7: Trabajos bajo bastidor y en el interior, este coche tampoco tiene acoplamientos ni cajas reductoras.
- Coche 8: Semejante al coche 1, final de los trabajos de mantenimiento.

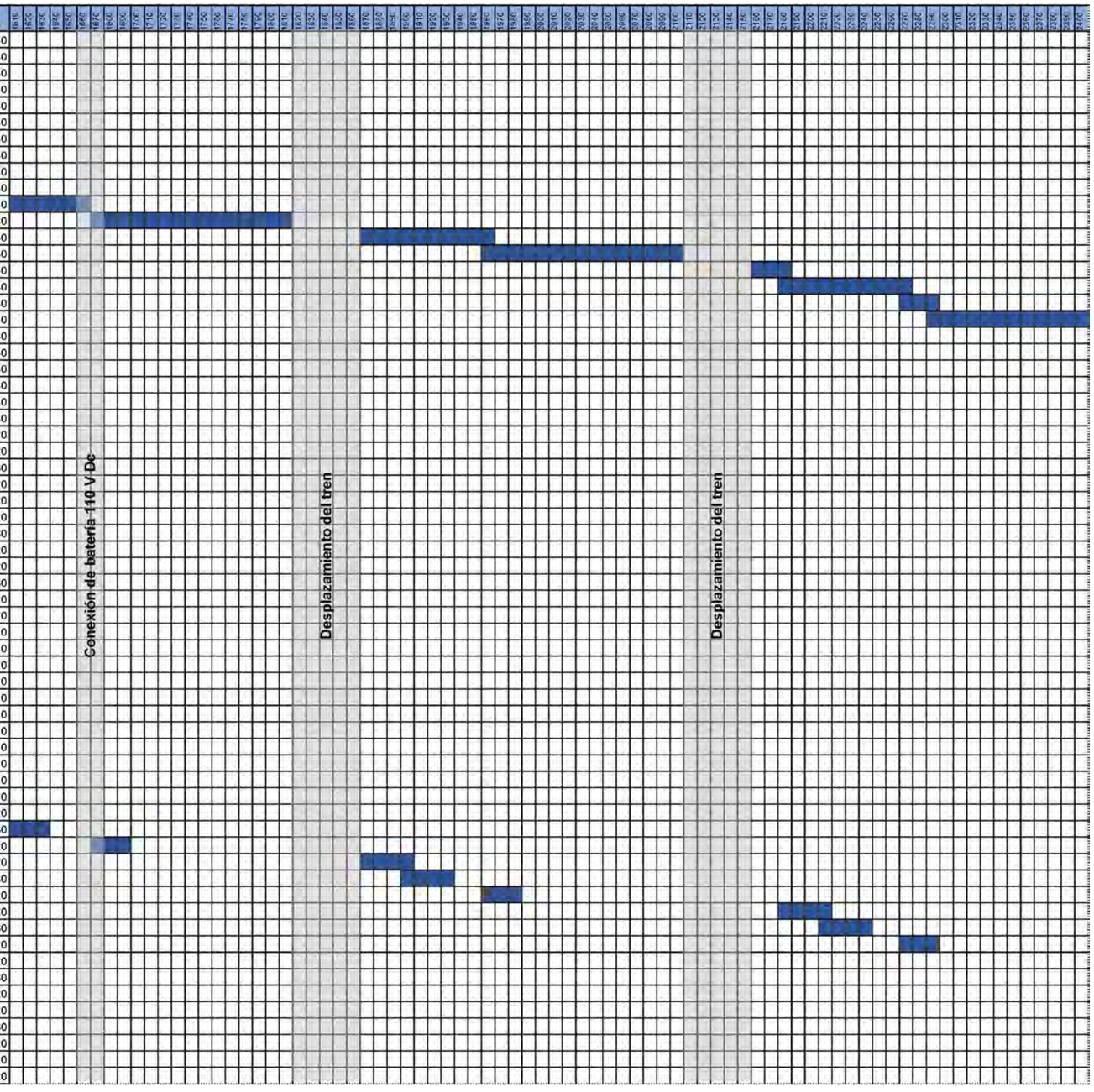
Codigo	Recurso/Descripcion	Inicio	Fin	Duracion	Localizacion
53	Benito/Inspeccion del sistema de freno	5	145	140 mins	140
49	Benito/Inspeccion y mantenimiento de las cajas reductoras	145	165	20 mins	140
25	Benito/Inspeccion de bogies	165	255	90 mins	140
45	Benito/Inspeccion y mantenimiento de los acoplamientos de transmision	255	275	20 mins	140
26	Benito/Inspeccion de bogies	335	425	90 mins	240
54	Benito/Inspeccion del sistema de freno	425	565	140 mins	240
50	Benito/Inspeccion y mantenimiento de las cajas reductoras	625	645	20 mins	340
27	Benito/Inspeccion de bogies	645	735	90 mins	340
46	Benito/Inspeccion y mantenimiento de los acoplamientos de transmision	735	755	20 mins	340
55	Benito/Inspeccion del sistema de freno	755	895	140 mins	340
28	Benito/Inspeccion de bogies	1565	1655	90 mins	440
56	Benito/Inspeccion del sistema de freno	1665	1805	140 mins	440
29	Benito/Inspeccion de bogies	1865	1955	90 mins	540
57	Benito/Inspeccion del sistema de freno	1955	2095	140 mins	540
51	Benito/Inspeccion y mantenimiento de las cajas reductoras	2155	2175	20 mins	640
30	Benito/Inspeccion de bogies	2175	2265	90 mins	640
47	Benito/Inspeccion y mantenimiento de los acoplamientos de transmision	2265	2285	20 mins	640
58	Benito/Inspeccion del sistema de freno	2285	2425	140 mins	640
31	Benito/Inspeccion de bogies	2485	2575	90 mins	740
59	Benito/Inspeccion del sistema de freno	2575	2715	140 mins	740
52	Benito/Inspeccion y mantenimiento de las cajas reductoras	2775	2795	20 mins	840
32	Benito/Inspeccion de bogies	2795	2885	90 mins	840
48	Benito/Inspeccion y mantenimiento de los acoplamientos de transmision	2885	2905	20 mins	840
60	Benito/Inspeccion del sistema de freno	2905	3045	140 mins	840
0	Gerardo/INITIAL	0	5	5 mins	120
17	Gerardo/Inspeccion visual de ventanas	165	195	30 mins	120
9	Gerardo/Inspeccion de faldones laterales y de tapas bajo bastidor	195	225	30 mins	130
61	Gerardo/Inspeccion visual de equipamiento de cabina	225	245	20 mins	120
1	Gerardo/Control de dotacion del tren	255	275	20 mins	120
18	Gerardo/Inspeccion visual de ventanas	335	365	30 mins	220
10	Gerardo/Inspeccion de faldones laterales y de tapas bajo bastidor	365	395	30 mins	230
2	Gerardo/Control de dotacion del tren	425	445	20 mins	220
19	Gerardo/Inspeccion visual de ventanas	645	675	30 mins	320
11	Gerardo/Inspeccion de faldones laterales y de tapas bajo bastidor	675	705	30 mins	330
3	Gerardo/Control de dotacion del tren	735	755	20 mins	320
43	Gerardo/Inspeccion de transductor de corriente de red	965	1005	40 mins	410
41	Gerardo/Inspeccion de transductor de alta tension	1005	1045	40 mins	410
39	Gerardo/Mantenimiento de seccionador de linea de techo de 25 kV AC	1045	1085	40 mins	410
37	Gerardo/Inspeccion y mantenimiento de disyuntor principal 25 kV AC	1085	1125	40 mins	410
34	Gerardo/Mantenimiento de seccionador de pantografo AC	1125	1165	40 mins	410
33	Gerardo/Inspeccion de pantografo 25 kV AC	1165	1205	40 mins	410
44	Gerardo/Inspeccion de transductor de corriente de red	1265	1305	40 mins	510
42	Gerardo/Inspeccion de transductor de alta tension	1305	1345	40 mins	510
40	Gerardo/Mantenimiento de seccionador de linea de techo de 25 kV AC	1345	1385	40 mins	510
38	Gerardo/Inspeccion y mantenimiento de disyuntor principal 25 kV AC	1385	1425	40 mins	510
36	Gerardo/Mantenimiento de seccionador de pantografo AC	1425	1465	40 mins	510
35	Gerardo/Inspeccion de pantografo 25 kV AC	1465	1505	40 mins	510
20	Gerardo/Inspeccion visual de ventanas	1565	1595	30 mins	420
12	Gerardo/Inspeccion de faldones laterales y de tapas bajo bastidor	1595	1625	30 mins	430
4	Gerardo/Control de dotacion del tren	1665	1685	20 mins	420
21	Gerardo/Inspeccion visual de ventanas	1865	1895	30 mins	520
13	Gerardo/Inspeccion de faldones laterales y de tapas bajo bastidor	1895	1925	30 mins	530
5	Gerardo/Control de dotacion del tren	1955	1975	20 mins	520
22	Gerardo/Inspeccion visual de ventanas	2175	2205	30 mins	620
14	Gerardo/Inspeccion de faldones laterales y de tapas bajo bastidor	2205	2235	30 mins	630
6	Gerardo/Control de dotacion del tren	2265	2285	20 mins	620
23	Gerardo/Inspeccion visual de ventanas	2485	2515	30 mins	720
15	Gerardo/Inspeccion de faldones laterales y de tapas bajo bastidor	2515	2545	30 mins	730
7	Gerardo/Control de dotacion del tren	2575	2595	20 mins	720
24	Gerardo/Inspeccion visual de ventanas	2795	2825	30 mins	820
16	Gerardo/Inspeccion de faldones laterales y de tapas bajo bastidor	2825	2855	30 mins	830
62	Gerardo/Inspeccion visual de equipamiento de cabina	2855	2875	20 mins	820
8	Gerardo/Control de dotacion del tren	2885	2905	20 mins	820
9999	Gerardo/FINAL	3045	3050	5 mins	820



Codigo	Recurso/Descripcion	Inicio	Fin	Duracion	Localizacion
53	Benito/Inspeccion del sistema de freno	5	145	140 mins	140
49	Benito/Inspeccion y mantenimiento de las cajas reductoras	145	165	20 mins	140
25	Benito/Inspeccion de bogies	165	255	90 mins	140
45	Benito/Inspeccion y mantenimiento de los acoplamientos de transmision	255	275	20 mins	140
26	Benito/Inspeccion de bogies	335	425	90 mins	240
54	Benito/Inspeccion del sistema de freno	425	565	140 mins	240
50	Benito/Inspeccion y mantenimiento de las cajas reductoras	625	645	20 mins	340
27	Benito/Inspeccion de bogies	645	735	90 mins	340
46	Benito/Inspeccion y mantenimiento de los acoplamientos de transmision	735	755	20 mins	340
55	Benito/Inspeccion del sistema de freno	755	895	140 mins	340
28	Benito/Inspeccion de bogies	1565	1655	90 mins	440
56	Benito/Inspeccion del sistema de freno	1665	1805	140 mins	440
29	Benito/Inspeccion de bogies	1865	1955	90 mins	540
57	Benito/Inspeccion del sistema de freno	1955	2095	140 mins	540
51	Benito/Inspeccion y mantenimiento de las cajas reductoras	2155	2175	20 mins	640
30	Benito/Inspeccion de bogies	2175	2265	90 mins	640
47	Benito/Inspeccion y mantenimiento de los acoplamientos de transmision	2265	2285	20 mins	640
58	Benito/Inspeccion del sistema de freno	2285	2425	140 mins	640
31	Benito/Inspeccion de bogies	2485	2575	90 mins	740
59	Benito/Inspeccion del sistema de freno	2575	2715	140 mins	740
52	Benito/Inspeccion y mantenimiento de las cajas reductoras	2775	2795	20 mins	840
32	Benito/Inspeccion de bogies	2795	2885	90 mins	840
48	Benito/Inspeccion y mantenimiento de los acoplamientos de transmision	2885	2905	20 mins	840
60	Benito/Inspeccion del sistema de freno	2905	3045	140 mins	840
0	Gerardo/INITIAL	0	5	5 mins	120
17	Gerardo/Inspeccion visual de ventanas	165	195	30 mins	120
9	Gerardo/Inspeccion de faldones laterales y de tapas bajo bastidor	195	225	30 mins	130
61	Gerardo/Inspeccion visual de equipamiento de cabina	225	245	20 mins	120
1	Gerardo/Control de dotacion del tren	255	275	20 mins	120
18	Gerardo/Inspeccion visual de ventanas	335	365	30 mins	220
10	Gerardo/Inspeccion de faldones laterales y de tapas bajo bastidor	365	395	30 mins	230
2	Gerardo/Control de dotacion del tren	425	445	20 mins	220
19	Gerardo/Inspeccion visual de ventanas	645	675	30 mins	320
11	Gerardo/Inspeccion de faldones laterales y de tapas bajo bastidor	675	705	30 mins	330
3	Gerardo/Control de dotacion del tren	735	755	20 mins	320
43	Gerardo/Inspeccion de transductor de corriente de red	965	1005	40 mins	410
41	Gerardo/Inspeccion de transductor de alta tension	1005	1045	40 mins	410
39	Gerardo/Mantenimiento de seccionador de linea de techo de 25 kV AC	1045	1085	40 mins	410
37	Gerardo/Inspeccion y mantenimiento de disyuntor principal 25 kV AC	1085	1125	40 mins	410
34	Gerardo/Mantenimiento de seccionador de pantografo AC	1125	1165	40 mins	410
33	Gerardo/Inspeccion de pantografo 25 kV AC	1165	1205	40 mins	410
44	Gerardo/Inspeccion de transductor de corriente de red	1265	1305	40 mins	510
42	Gerardo/Inspeccion de transductor de alta tension	1305	1345	40 mins	510
40	Gerardo/Mantenimiento de seccionador de linea de techo de 25 kV AC	1345	1385	40 mins	510
38	Gerardo/Inspeccion y mantenimiento de disyuntor principal 25 kV AC	1385	1425	40 mins	510
36	Gerardo/Mantenimiento de seccionador de pantografo AC	1425	1465	40 mins	510
35	Gerardo/Inspeccion de pantografo 25 kV AC	1465	1505	40 mins	510
20	Gerardo/Inspeccion visual de ventanas	1565	1595	30 mins	420
12	Gerardo/Inspeccion de faldones laterales y de tapas bajo bastidor	1595	1625	30 mins	430
4	Gerardo/Control de dotacion del tren	1665	1685	20 mins	420
21	Gerardo/Inspeccion visual de ventanas	1865	1895	30 mins	520
13	Gerardo/Inspeccion de faldones laterales y de tapas bajo bastidor	1895	1925	30 mins	530
5	Gerardo/Control de dotacion del tren	1955	1975	20 mins	520
22	Gerardo/Inspeccion visual de ventanas	2175	2205	30 mins	620
14	Gerardo/Inspeccion de faldones laterales y de tapas bajo bastidor	2205	2235	30 mins	630
6	Gerardo/Control de dotacion del tren	2265	2285	20 mins	620
23	Gerardo/Inspeccion visual de ventanas	2485	2515	30 mins	720
15	Gerardo/Inspeccion de faldones laterales y de tapas bajo bastidor	2515	2545	30 mins	730
7	Gerardo/Control de dotacion del tren	2575	2595	20 mins	720
24	Gerardo/Inspeccion visual de ventanas	2795	2825	30 mins	820
16	Gerardo/Inspeccion de faldones laterales y de tapas bajo bastidor	2825	2855	30 mins	830
62	Gerardo/Inspeccion visual de equipamiento de cabina	2855	2875	20 mins	820
8	Gerardo/Control de dotacion del tren	2885	2905	20 mins	620
9999	Gerardo/FINAL	3045	3050	5 mins	820



Codigo	Recurso/Descripcion	Inicio	Fin	Duracion	Localizacion
53	Benito/Inspeccion del sistema de freno	5	145	140 mins	140
49	Benito/Inspeccion y mantenimiento de las cajas reductoras	145	165	20 mins	140
25	Benito/Inspeccion de bogies	165	255	90 mins	140
45	Benito/Inspeccion y mantenimiento de los acoplamientos de transmision	255	275	20 mins	140
26	Benito/Inspeccion de bogies	335	425	90 mins	240
54	Benito/Inspeccion del sistema de freno	425	565	140 mins	240
50	Benito/Inspeccion y mantenimiento de las cajas reductoras	625	645	20 mins	340
27	Benito/Inspeccion de bogies	645	735	90 mins	340
46	Benito/Inspeccion y mantenimiento de los acoplamientos de transmision	735	755	20 mins	340
55	Benito/Inspeccion del sistema de freno	755	895	140 mins	340
28	Benito/Inspeccion de bogies	1565	1655	90 mins	440
56	Benito/Inspeccion del sistema de freno	1665	1805	140 mins	440
29	Benito/Inspeccion de bogies	1865	1955	90 mins	540
57	Benito/Inspeccion del sistema de freno	1955	2095	140 mins	540
51	Benito/Inspeccion y mantenimiento de las cajas reductoras	2155	2175	20 mins	640
30	Benito/Inspeccion de bogies	2175	2265	90 mins	640
47	Benito/Inspeccion y mantenimiento de los acoplamientos de transmision	2265	2285	20 mins	640
58	Benito/Inspeccion del sistema de freno	2285	2425	140 mins	640
31	Benito/Inspeccion de bogies	2485	2575	90 mins	740
59	Benito/Inspeccion del sistema de freno	2575	2715	140 mins	740
52	Benito/Inspeccion y mantenimiento de las cajas reductoras	2775	2795	20 mins	840
32	Benito/Inspeccion de bogies	2795	2885	90 mins	840
48	Benito/Inspeccion y mantenimiento de los acoplamientos de transmision	2885	2905	20 mins	840
60	Benito/Inspeccion del sistema de freno	2905	3045	140 mins	840
0	Gerardo/INITIAL	0	5	5 mins	120
17	Gerardo/Inspeccion visual de ventanas	165	195	30 mins	120
9	Gerardo/Inspeccion de faldones laterales y de tapas bajo bastidor	195	225	30 mins	130
61	Gerardo/Inspeccion visual de equipamiento de cabina	225	245	20 mins	120
1	Gerardo/Control de dotacion del tren	255	275	20 mins	120
18	Gerardo/Inspeccion visual de ventanas	335	365	30 mins	220
10	Gerardo/Inspeccion de faldones laterales y de tapas bajo bastidor	365	395	30 mins	230
2	Gerardo/Control de dotacion del tren	425	445	20 mins	220
19	Gerardo/Inspeccion visual de ventanas	645	675	30 mins	320
11	Gerardo/Inspeccion de faldones laterales y de tapas bajo bastidor	675	705	30 mins	330
3	Gerardo/Control de dotacion del tren	735	755	20 mins	320
43	Gerardo/Inspeccion de transductor de corriente de red	965	1005	40 mins	410
41	Gerardo/Inspeccion de transductor de alta tension	1005	1045	40 mins	410
39	Gerardo/Mantenimiento de seccionador de linea de techo de 25 kV AC	1045	1085	40 mins	410
37	Gerardo/Inspeccion y mantenimiento de disyuntor principal 25 kV AC	1085	1125	40 mins	410
34	Gerardo/Mantenimiento de seccionador de pantografo AC	1125	1165	40 mins	410
33	Gerardo/Inspeccion de pantografo 25 kV AC	1165	1205	40 mins	410
44	Gerardo/Inspeccion de transductor de corriente de red	1265	1305	40 mins	510
42	Gerardo/Inspeccion de transductor de alta tension	1305	1345	40 mins	510
40	Gerardo/Mantenimiento de seccionador de linea de techo de 25 kV AC	1345	1385	40 mins	510
38	Gerardo/Inspeccion y mantenimiento de disyuntor principal 25 kV AC	1385	1425	40 mins	510
36	Gerardo/Mantenimiento de seccionador de pantografo AC	1425	1465	40 mins	510
35	Gerardo/Inspeccion de pantografo 25 kV AC	1465	1505	40 mins	510
20	Gerardo/Inspeccion visual de ventanas	1565	1595	30 mins	420
12	Gerardo/Inspeccion de faldones laterales y de tapas bajo bastidor	1595	1625	30 mins	430
4	Gerardo/Control de dotacion del tren	1665	1685	20 mins	420
21	Gerardo/Inspeccion visual de ventanas	1865	1895	30 mins	520
13	Gerardo/Inspeccion de faldones laterales y de tapas bajo bastidor	1895	1925	30 mins	530
5	Gerardo/Control de dotacion del tren	1955	1975	20 mins	520
22	Gerardo/Inspeccion visual de ventanas	2175	2205	30 mins	620
14	Gerardo/Inspeccion de faldones laterales y de tapas bajo bastidor	2205	2235	30 mins	630
6	Gerardo/Control de dotacion del tren	2265	2285	20 mins	620
23	Gerardo/Inspeccion visual de ventanas	2485	2515	30 mins	720
15	Gerardo/Inspeccion de faldones laterales y de tapas bajo bastidor	2515	2545	30 mins	730
7	Gerardo/Control de dotacion del tren	2575	2595	20 mins	720
24	Gerardo/Inspeccion visual de ventanas	2795	2825	30 mins	820
16	Gerardo/Inspeccion de faldones laterales y de tapas bajo bastidor	2825	2855	30 mins	830
62	Gerardo/Inspeccion visual de equipamiento de cabina	2855	2875	20 mins	820
8	Gerardo/Control de dotacion del tren	2885	2905	20 mins	820
9999	Gerardo/FINAL	3045	3050	5 mins	820



Codigo	Recurso/Descripcion	Inicio	Fin	Duracion	Localizacion
53	Benito/Inspeccion del sistema de freno	5	145	140 mins	140
49	Benito/Inspeccion y mantenimiento de las cajas reductoras	145	165	20 mins	140
25	Benito/Inspeccion de bogies	165	255	90 mins	140
45	Benito/Inspeccion y mantenimiento de los acoplamientos de transmision	255	275	20 mins	140
26	Benito/Inspeccion de bogies	335	425	90 mins	240
54	Benito/Inspeccion del sistema de freno	425	565	140 mins	240
50	Benito/Inspeccion y mantenimiento de las cajas reductoras	625	645	20 mins	340
27	Benito/Inspeccion de bogies	645	735	90 mins	340
46	Benito/Inspeccion y mantenimiento de los acoplamientos de transmision	735	755	20 mins	340
55	Benito/Inspeccion del sistema de freno	755	895	140 mins	340
28	Benito/Inspeccion de bogies	1565	1655	90 mins	440
56	Benito/Inspeccion del sistema de freno	1665	1805	140 mins	440
29	Benito/Inspeccion de bogies	1865	1955	90 mins	540
57	Benito/Inspeccion del sistema de freno	1955	2095	140 mins	540
51	Benito/Inspeccion y mantenimiento de las cajas reductoras	2155	2175	20 mins	640
30	Benito/Inspeccion de bogies	2175	2265	90 mins	640
47	Benito/Inspeccion y mantenimiento de los acoplamientos de transmision	2265	2285	20 mins	640
58	Benito/Inspeccion del sistema de freno	2285	2425	140 mins	640
31	Benito/Inspeccion de bogies	2485	2575	90 mins	740
59	Benito/Inspeccion del sistema de freno	2575	2715	140 mins	740
52	Benito/Inspeccion y mantenimiento de las cajas reductoras	2775	2795	20 mins	840
32	Benito/Inspeccion de bogies	2795	2885	90 mins	840
48	Benito/Inspeccion y mantenimiento de los acoplamientos de transmision	2885	2905	20 mins	840
60	Benito/Inspeccion del sistema de freno	2905	3045	140 mins	840
0	Gerardo/INITIAL	0	5	5 mins	120
17	Gerardo/Inspeccion visual de ventanas	165	195	30 mins	120
9	Gerardo/Inspeccion de faldones laterales y de tapas bajo bastidor	195	225	30 mins	130
61	Gerardo/Inspeccion visual de equipamiento de cabina	225	245	20 mins	120
1	Gerardo/Control de dotacion del tren	255	275	20 mins	120
18	Gerardo/Inspeccion visual de ventanas	335	365	30 mins	220
10	Gerardo/Inspeccion de faldones laterales y de tapas bajo bastidor	365	395	30 mins	230
2	Gerardo/Control de dotacion del tren	425	445	20 mins	220
19	Gerardo/Inspeccion visual de ventanas	645	675	30 mins	320
11	Gerardo/Inspeccion de faldones laterales y de tapas bajo bastidor	675	705	30 mins	330
3	Gerardo/Control de dotacion del tren	735	755	20 mins	320
43	Gerardo/Inspeccion de transductor de corriente de red	965	1005	40 mins	410
41	Gerardo/Inspeccion de transductor de alta tension	1005	1045	40 mins	410
39	Gerardo/Mantenimiento de seccionador de linea de techo de 25 kV AC	1045	1085	40 mins	410
37	Gerardo/Inspeccion y mantenimiento de disyuntor principal 25 kV AC	1085	1125	40 mins	410
34	Gerardo/Mantenimiento de seccionador de pantografo AC	1125	1165	40 mins	410
33	Gerardo/Inspeccion de pantografo 25 kV AC	1165	1205	40 mins	410
44	Gerardo/Inspeccion de transductor de corriente de red	1265	1305	40 mins	510
42	Gerardo/Inspeccion de transductor de alta tension	1305	1345	40 mins	510
40	Gerardo/Mantenimiento de seccionador de linea de techo de 25 kV AC	1345	1385	40 mins	510
38	Gerardo/Inspeccion y mantenimiento de disyuntor principal 25 kV AC	1385	1425	40 mins	510
36	Gerardo/Mantenimiento de seccionador de pantografo AC	1425	1465	40 mins	510
35	Gerardo/Inspeccion de pantografo 25 kV AC	1465	1505	40 mins	510
20	Gerardo/Inspeccion visual de ventanas	1565	1595	30 mins	420
12	Gerardo/Inspeccion de faldones laterales y de tapas bajo bastidor	1595	1625	30 mins	430
4	Gerardo/Control de dotacion del tren	1665	1685	20 mins	420
21	Gerardo/Inspeccion visual de ventanas	1865	1895	30 mins	520
13	Gerardo/Inspeccion de faldones laterales y de tapas bajo bastidor	1895	1925	30 mins	530
5	Gerardo/Control de dotacion del tren	1955	1975	20 mins	520
22	Gerardo/Inspeccion visual de ventanas	2175	2205	30 mins	620
14	Gerardo/Inspeccion de faldones laterales y de tapas bajo bastidor	2205	2235	30 mins	630
6	Gerardo/Control de dotacion del tren	2265	2285	20 mins	620
23	Gerardo/Inspeccion visual de ventanas	2485	2515	30 mins	720
15	Gerardo/Inspeccion de faldones laterales y de tapas bajo bastidor	2515	2545	30 mins	730
7	Gerardo/Control de dotacion del tren	2575	2595	20 mins	720
24	Gerardo/Inspeccion visual de ventanas	2795	2825	30 mins	820
16	Gerardo/Inspeccion de faldones laterales y de tapas bajo bastidor	2825	2855	30 mins	830
62	Gerardo/Inspeccion visual de equipamiento de cabina	2855	2875	20 mins	820
8	Gerardo/Control de dotacion del tren	2885	2905	20 mins	820
9999	Gerardo/FINAL	3045	3050	5 mins	820

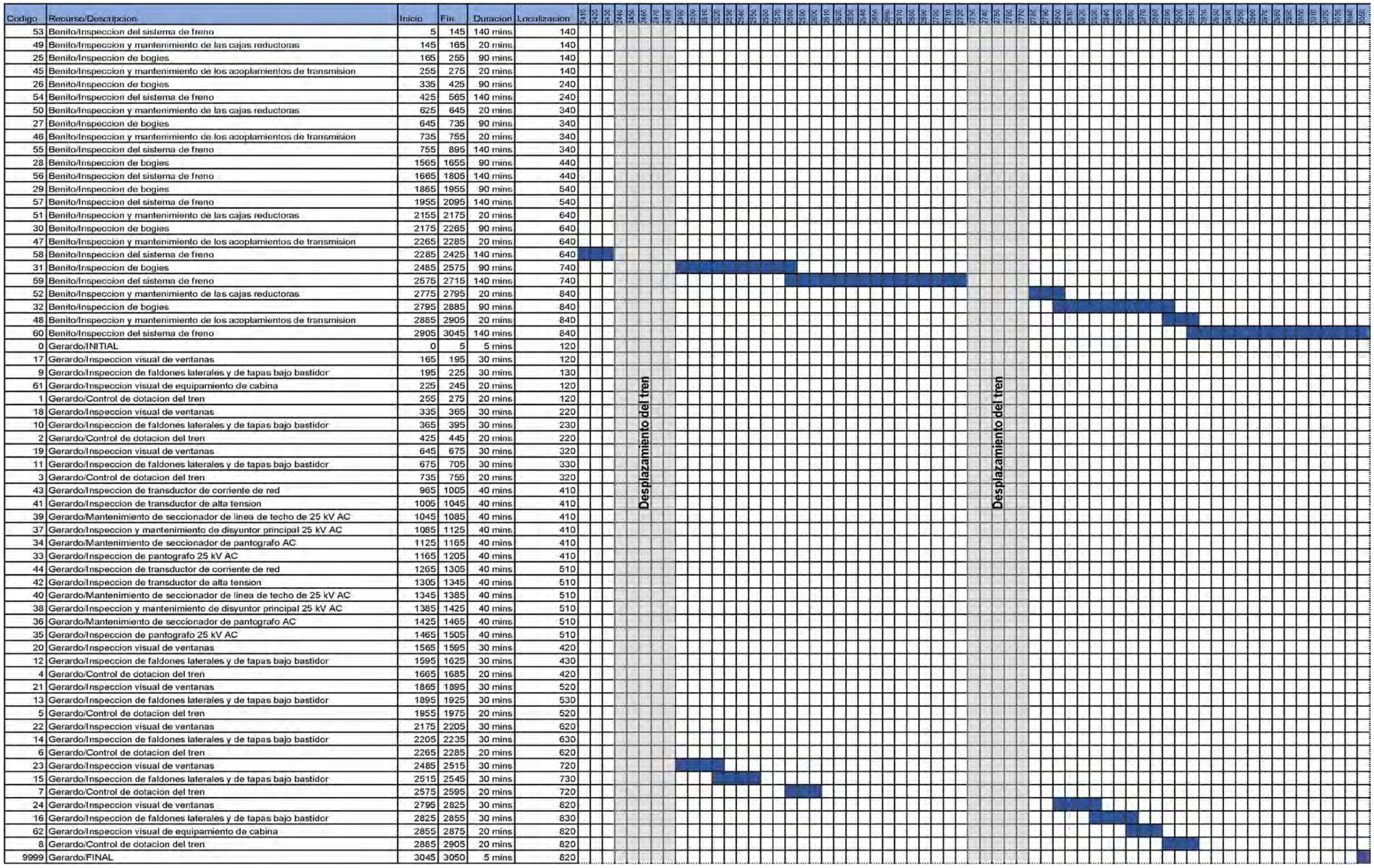


Figura 44 Flujograma creado por OMSA

Los datos de entrada, la tabla con el resultado de OMSA y el cronograma se encuentran en el Fichero Excel “OMS_validación_064_190212_01_00h_00m_27s” (referencia por trazabilidad).

4.4.2 ROBUSTEZ DEL ALGORITMO

Durante el desarrollo de la aplicación informática se observó que el flujograma de las tareas de la estadía calculado por OMSA depende del orden de las tareas de mantenimiento en la lista de entrada (véase apartado "Datos de Tareas de Mantenimiento", página 59). Este hecho plantea la pregunta de hasta qué punto el resultado de OMSA se acerca al flujograma óptimo.

Para contestar a esta pregunta se ha tomado una lista de tareas de mantenimiento y se la ha reordenado aleatoriamente un total de mil veces antes de aplicarle OMSA a cada una de las secuencias aleatorias.

En estas simulaciones se ha considerado un taller con una vía de foso y acceso al techo a lo largo de los ocho coches y 200 m de longitud del tren. Los tiempos de desplazamiento entre localizaciones de mantenimiento son los presentados en la Figura 3 de la pagina 6. Los estados de seguridad son sólo dos:

Estado de seguridad	Tiempo A→B	Tiempo B→A
Catenaria de 25 kV AC 50 Hz conectada	30 min	45 min
Batería de 110 V DC conectada	15 min	15 min

Tabla XXI Estados de seguridad de las simulaciones para evaluar la robustez de la aplicación

Los recursos previstos para la realización de las tareas son los siguientes:

- Especialistas de bogie: 2
- Técnicos generalistas: 2
- Electricistas: 2
- Mecánicos: 1
- Especialistas en señalización: 1

La lista de tareas es la siguiente:

Código	Descripción	Dura-ción	Esta-dos	Loc.	Interv.	Prio.1	Prio.2	Bog.	Gen	Elec	Mec	Señ	Suc.
0	INITIAL	10	AA	120	1000	1000	1000	0	0	1	1	0	na
99999	FINAL	10	AA	820	1000	1000	1000	0	0	1	1	0	na
7	Inspeccion montaje antena I2	20	BC	140	1000	1000	1000	0	0	0	0	1	9
8	Inspeccion montaje antena I2	20	BC	840	1000	1000	1000	0	0	0	0	1	10
9	Inspeccion montaje antena - M1 detalle	30	BC	140	1000	1000	1000	0	0	0	0	1	na
10	Inspeccion montaje antena - M1 detalle	30	BC	840	1000	1000	1000	0	0	0	0	1	na
13	Enganche automatico	20	BB	130	1000	1000	1000	0	1	0	1	0	na
14	Enganche automatico	20	BB	830	1000	1000	1000	0	1	0	1	0	na
23	Inspeccion visual de freno bajo bastidor	40	BC	140	1000	1000	1000	1	1	0	0	0	244
24	Inspeccion visual de freno bajo bastidor	40	BC	240	1000	1000	1000	1	1	0	0	0	na
25	Inspeccion visual de freno bajo bastidor	40	BC	340	1000	1000	1000	1	1	0	0	0	na
26	Inspeccion visual de freno bajo bastidor	40	BC	440	1000	1000	1000	1	1	0	0	0	na
27	Inspeccion visual de freno bajo bastidor	40	BC	540	1000	1000	1000	1	1	0	0	0	na

Código	Descripción	Dura-ción	Esta-dos	Loc.	Interv.	Prio.1	Prio.2	Bog.	Gen	Elec	Mec	Señ	Suc.
28	Inspeccion visual de freno bajo bastidor	40	BC	640	1000	1000	1000	1	1	0	0	0	na
29	Inspeccion visual de freno bajo bastidor	40	BC	740	1000	1000	1000	1	1	0	0	0	na
30	Inspeccion visual de freno bajo bastidor	40	BC	840	1000	1000	1000	1	1	0	0	0	245
41	Inspeccion equipamiento electrico en techo	20	BB	210	1000	1000	1000	0	0	1	0	0	na
42	Inspeccion equipamiento electrico en techo	20	BB	710	1000	1000	1000	0	0	1	0	0	na
112	Inspeccion de interiorismo	20	AA	120	1000	1000	1000	0	1	0	0	0	na
113	Inspeccion de interiorismo	20	AA	220	1000	1000	1000	0	1	0	0	0	na
114	Inspeccion de interiorismo	20	AA	320	1000	1000	1000	0	1	0	0	0	na
115	Inspeccion de interiorismo	20	AA	420	1000	1000	1000	0	1	0	0	0	na
116	Inspeccion de interiorismo	20	AA	520	1000	1000	1000	0	1	0	0	0	na
117	Inspeccion de interiorismo	20	AA	620	1000	1000	1000	0	1	0	0	0	na
118	Inspeccion de interiorismo	20	AA	720	1000	1000	1000	0	1	0	0	0	na
119	Inspeccion de interiorismo	20	AA	820	1000	1000	1000	0	1	0	0	0	na
156	Pantografo	15	BB	210	1000	1000	1000	0	0	1	0	0	na
157	Pantografo	15	BB	710	1000	1000	1000	0	0	1	0	0	na
206	Inspeccion de motor de traccion	20	BC	340	1000	1000	1000	1	0	0	0	0	na
207	Inspeccion de motor de traccion	20	BC	640	1000	1000	1000	1	0	0	0	0	na
244	Ensayo funcional de freno	10	AA	140	1000	1000	1000	1	1	0	0	0	na
245	Ensayo funcional de freno	10	AA	840	1000	1000	1000	1	1	0	0	0	na

Tabla XXII Tareas de las simulaciones para evaluar la robustez de la aplicación

La Figura 45 muestra la distribución estadística de los resultados de OMSA. El 90% de las duraciones de los flujogramas calculados por OMSA se encontraron entre 427 y 473 minutos, es decir, dentro del 10,77% por encima de la duración más corta encontrada.

El tiempo de ejecución de las mil repeticiones fue de 2 horas, 42 minutos y 5 segundos, es decir 9,725 segundos en media por ejecución de OMSA.

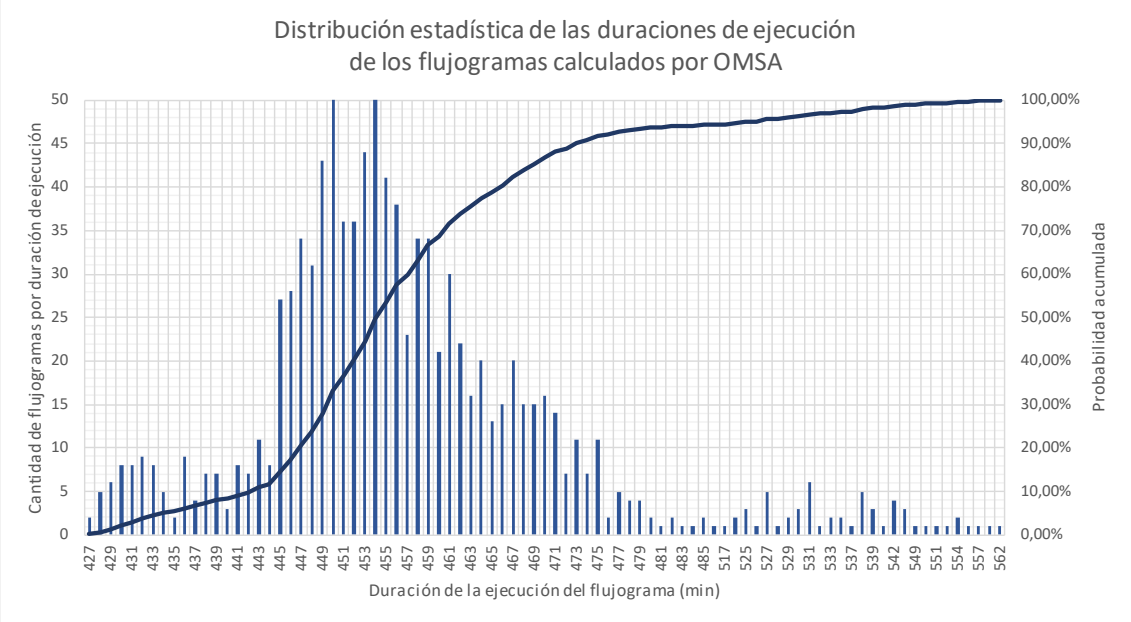


Figura 45 Distribución estadística de los resultados del algoritmo OMSA para 1.000 entradas aleatorias

Los datos de entrada, la tabla con el resultado de OMSA y el cronograma se encuentran en el Fichero Excel "OMS_validación 1000_032_190203_01_02h_42m_05s" (referencia por trazabilidad).

La dispersión de los resultados depende de las restricciones dadas por los estados de seguridad, de la homogeneidad de los tiempos de desplazamiento y de la cantidad de recursos. Si aplicamos OMSA a una simulación con pocos estados de seguridad, tiempos de desplazamientos con poca dispersión entre ellos y con recursos redundantes, los flujogramas pueden ser más variados. Al haber pocos estados de seguridad es posible paralelizar las tareas de más formas distintas. Si los tiempos de desplazamiento son muy semejantes, el orden de realización de las tareas, y con ello los tiempos de desplazamiento, no tendrá una gran influencia en la duración total. Finalmente, si hay abundancia de recursos será posible plantear flujogramas con más tareas en paralelo, dando mayor variedad a cómo se pueden desarrollar las tareas.

Se ha tomado el caso real presentado en el apartado "Comparación con un Caso Real", página 87, y se han reordenado aleatoriamente sus tareas de mantenimiento antes de aplicar OMSA a cada una de las secuencias aleatorias.

La Figura 46 muestra de nuevo la distribución estadística de los resultados de OMSA. Ahora el 90% de las duraciones de los flujogramas calculados por OMSA se encontraron entre 3050 y 3073 minutos, es decir, dentro del 0,75% por encima de la duración más corta encontrada.

El tiempo de ejecución de las mil repeticiones fue de 6 horas, 7 minutos y 23 segundos, es decir, 22,04 segundos en media por ejecución de OMSA.

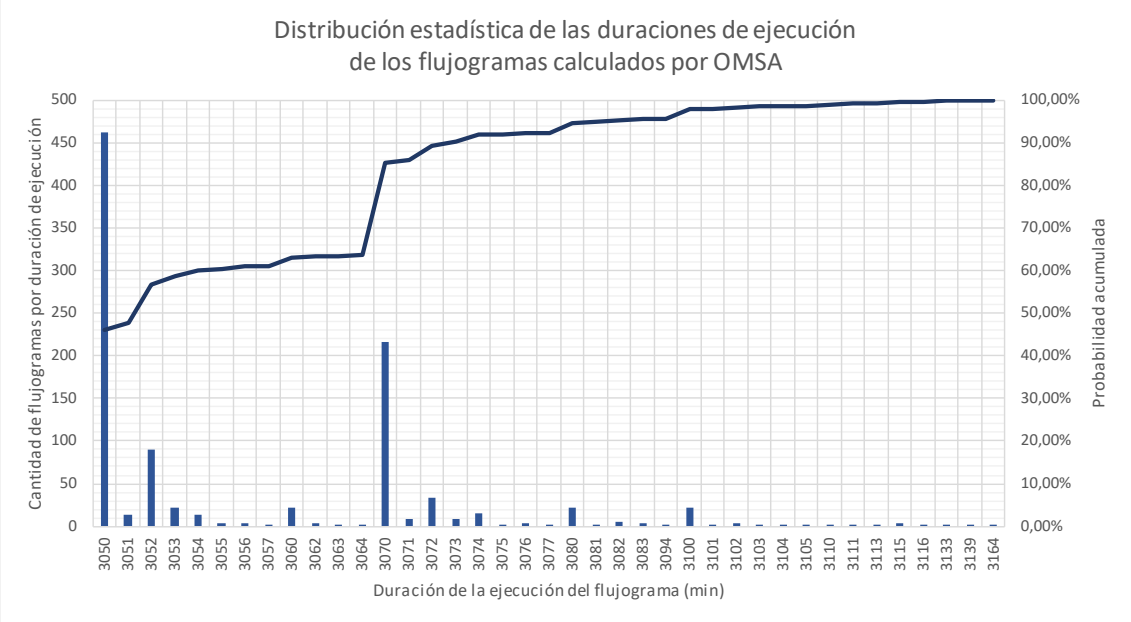


Figura 46 Distribución estadística de los resultados del algoritmo OMSA para 1.000 entradas aleatorias de una simulación restrictiva

La mayor parte de los flujogramas tienen una duración de 3.050 minutos (caso óptimo) o de 3.070 minutos. Los flujogramas con duración de 3.070 min se caracterizan porque no se pasa de los trabajos en el techo del coche 4 a los trabajos en el techo del coche 5, ambos sin tensión de batería 110 V DC, sino que del techo del coche 4 se pasa a la zona bajo bastidor del coche 4 a realizar trabajos con batería, y después se continúa en el techo del coche 5, de nuevo con batería. Esto implica que se realizan una conexión y una desconexión adicional de la batería de 110 V DC con un incremento total de 20 min.

Los datos de entrada, la tabla con el resultado de OMSA y el cronograma se encuentran en el Fichero Excel "OMS_validación 1000_064_190203_01_06h_07m_23s" (referencia por trazabilidad).

Un posible método de minimizar variaciones en el resultado del algoritmo dependiendo del orden de las tareas en la tabla de datos de entrada sería la realización de un algoritmo genético superpuesto a OMSA con las siguientes características:

- La población inicial consistiría en diferentes versiones aleatorias de la lista de tareas ordenadas diferentemente, incluyendo tal vez versiones ordenadas por localización, por estados de seguridad, o por los recursos empleados.
- El cruce o mestizaje se realizaría mezclando las listas para generar nuevas secuencias.
- El cálculo de la función de aptitud es OMSA.
- Se seleccionaría la población progenitora de la siguiente generación a partir de las listas de entrada que arrojasen las duraciones de ejecución menores.

No se ha seguido esta línea de mejora debido a que requiere una potencia de cálculo superior y el empleo de un lenguaje de programación compilado (p.ej. C++), lo cual sobrepasa el presupuesto disponible.

4.5 APLICACIONES

4.5.1 PLANIFICACIÓN DE ESTADÍAS

La aplicación directa e inmediata de OMSA es la planificación de una estadía de mantenimiento. Como se mencionó en el apartado “Planificación de Estadías”, página 8, la creación de un flujograma para los trabajos de una estadía es un proceso largo y tedioso realizado manualmente por el planificador de mantenimiento. Por ello en muchos casos no se realizan flujogramas para las estadías.

Como datos de entrada se han tomado las 287 tareas de mantenimiento reales de un tren de alta velocidad de ocho coches y 200 m de longitud durante la intervención de 100.000 km (intervención M1). A las 287 tareas se les han añadido una tarea inicial más una tarea final de 5 minutos cada una que marcan el inicio y el final de los trabajos.

El taller donde se realiza la intervención cumple con los requisitos habituales para realizar el mantenimiento de trenes de alta velocidad, con un foso de 200 m y excelente acceso al techo. Los tiempos de desplazamiento entre localizaciones de mantenimiento están representados en la Figura 3, página 6.

Los estados de seguridad están representados en la Tabla XXI, página 97.

Los recursos previstos para la realización de las tareas son los siguientes:

- Especialistas de bogie: 2
- Técnicos generalistas: 4
- Electricistas: 2
- Mecánicos: 2

Debido a su tamaño, no se incluye aquí la lista de tareas como en ejemplos anteriores. La lista de tareas se puede consultar en el Fichero Excel “OMS_planificación estadía M1_289_190218_02_12h_41m_06s” (referencia por trazabilidad).

El tiempo de ejecución del algoritmo fue de 12 horas, 41 minutos y 6 segundos, excesivamente largo para una implementación comercial que requerirá menores tiempos de ejecución. Las posibles medidas para reducir el tiempo de cálculo son el uso de un lenguaje de programación compilado en vez de Java, que es un intérprete, y que la aplicación informática corra sobre una máquina más potente.

Los datos de entrada, la tabla con el resultado de OMSA y el cronograma se encuentran en el Fichero Excel “OMS_planificación estadía M1_289_190218_02_12h_41m_06s” (referencia por trazabilidad). Desgraciadamente, el cronograma es demasiado extenso para ser presentado en el formato de este documento.

Acerca del resultado de OMSA:

- El flujograma incluye una desconexión y una conexión de la batería más de las necesarias, pero esto sólo representa 20 minutos del total de 2.665 minutos (0,75% de 44 horas, 25 minutos, o aproximadamente 8 turnos y medio).
- El resultado cumple estrictamente las restricciones de personal, tiempos de traslado, estados de seguridad, etc.

- Los recursos de tipo “general” están atareados durante todo el flujograma y forman un cuello de botella.
- Los recursos de tipo “electricista” también están ocupados de manera permanente y evitan una reducción del tiempo de estadía.
- Los recursos de tipo “bogie” y “mecánico” no están ocupados durante todo el tiempo.
- Los desplazamientos de los recursos se limitan en general a coches y localizaciones contiguas, siempre y cuando sus tareas se encuentren en el camino crítico.
- OMSA no está diseñado para entregar resultados regulares, es decir, no organiza las tareas de manera idéntica en coches idénticos.

4.5.2 AUMENTO DE LA PRODUCTIVIDAD DEL PERSONAL DE MANTENIMIENTO

OMSA permite realizar diversas simulaciones de una estadía con distintas composiciones del equipo de mantenimiento. De esta forma se pueden comparar los tiempos de estadía en función de la cantidad y cualificación del personal disponible y buscar la composición del equipo más económica que sea capaz de ejecutar las tareas de la estadía en el tiempo disponible.

Desde el punto de vista de una estadía aislada, el menor equipo capaz de realizar las tareas asignadas en el tiempo de estancia del tren en el taller es el equipo más económico. Sin embargo, puede que este equipo no sea la opción más económica si contemplamos el total de las estadías de vehículos planificadas. Debido a que muchas tareas deben realizarse por un equipo cuyos miembros tienen diversas cualificaciones, puede ocurrir que la falta de recursos con una cualificación determinada genere un cuello de botella que impida el aprovechamiento pleno de los recursos de la otra cualificación. Tomemos como ejemplo la tarea “Comprobacion y mantenimiento de morro y cinematica” en los coches extremo 1 y 8, y la cual es realizada por un equipo formado por un mecánico y un asistente no cualificado. Si el equipo de mantenimiento tiene dos mecánicos y sólo un asistente no cualificado no será posible realizar la tarea en ambos coches en paralelo y causará un tiempo de espera para uno de los mecánicos. De este modo, disminuirá la productividad medida como el cociente entre el tiempo mínimo neto de ejecución de las tareas por el tiempo total en el que los recursos están vinculados a la estadía.

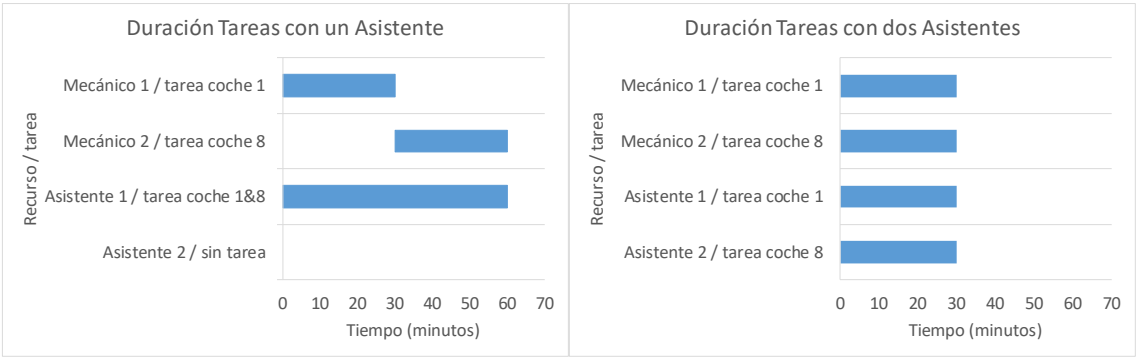


Figura 47 Comparación de la duración de tareas con un asistente o con dos

En el caso de planificar dos mecánicos y un asistente, la duración total de los trabajos es de 60 minutos (véase gráfico izquierdo de la Figura 47). El asistente está ocupado durante los 60 minutos que está vinculado (reservado) a los trabajos, pero cada mecánico sólo trabaja 30

minutos de los 60 que está vinculado a la estadía. Dado que el total de trabajo neto necesario es de 120 minutos (30 + 30 + 60), la productividad es:

$$Productividad = \frac{\sum Tiempos\ necesarios\ para\ realizar\ las\ tareas}{\sum Tiempos\ que\ los\ recursos\ están\ vinculados} = \\ = \frac{30 + 30 + 30 + 30}{60 + 60 + 60} = 2/3$$

En el caso de planificar dos mecánicos y dos asistentes, la duración total de los trabajos es de 30 minutos (véase gráfico derecho de la Figura 47). Cada asistente es asignado a un mecánico para realizar la tarea en uno de los coches. El tiempo de vinculación es igual al tiempo de ejecución de las tareas y la productividad es del 100%.

Si calculamos los costes como el tiempo de vinculación de los recursos a las tareas multiplicado por el coste de una hora de trabajo veremos que los menores costes no se dan para el equipo más reducido sino para el de mayor productividad:

$$Costes = Coste\ horario \times \sum Tiempos\ que\ los\ recursos\ están\ vinculados = \\ = Coste\ horario \times \frac{\sum Tiempos\ necesarios\ para\ realizar\ las\ tares}{Productividad} = \\ = \frac{Constante}{Productividad}$$

La consecuencia de este planteamiento puede ser que se planifiquen más recursos de los estrictamente necesarios para finalizar las tareas de la estadía en el tiempo de disponibilidad del tren en el taller. Esto es correcto siempre y cuando haya más trenes con tareas pendientes que puedan realizar los recursos una vez liberados.

En este apartado, OMSA calculará la secuenciación de tareas de una estancia dada con diferentes composiciones del equipo de mantenimiento. El objetivo es identificar el equipo de mantenimiento que permita alcanzar la mayor productividad. Los tiempos de vinculación de cada recurso se contarán desde el inicio de su primera tarea asignada hasta la finalización de su última tarea. De este modo se penalizarán tiempos intermedios de espera.

Como datos de entrada se han tomado las 287 tareas de mantenimiento empleadas en el apartado “Planificación de Estadías”, página 101. Debido al largo tiempo de simulación con esta cantidad de tareas, se las han agrupado reduciéndolas a un grupo equivalente de 96 tareas más las tareas inicial y final de 5 minutos cada una que marcan el inicio y el final de los trabajos. Se han agrupado tareas con idéntica localización, cualificación de operarios y estados de seguridad idénticos.

La suma de tiempos estrictamente necesarios para realizar las tareas es de 9.280 minutos y se usará como referencia para medir la productividad.

El taller donde se realiza la intervención cumple con los requisitos habituales para realizar el mantenimiento de trenes de alta velocidad, con un foso de 200 m y excelente acceso al techo. Los tiempos de desplazamiento entre localizaciones de mantenimiento están representados en la Figura 3, página 6.

Los estados de seguridad están representados en la Tabla XXI, página 97.

Debido a su tamaño, no se incluye aquí la lista de tareas como en ejemplos anteriores. La lista de tareas se puede consultar en el Fichero Excel “OMS_optimizacion

personal_098_190209_01_00h_06m_12s" (referencia por trazabilidad). La lista de tareas es la misma en todas las simulaciones de este apartado.

PRIMERA SIMULACIÓN

Los recursos previstos para la realización de las tareas son los siguientes:

- Especialistas de bogie: 2
- Técnicos generalistas: 4
- Mecánicos: 2
- Electricistas: 2

Tiempo de ejecución del algoritmo: 6 minutos y 12 segundos

Los datos de entrada, la tabla con el resultado de OMSA y el cronograma se encuentran en el Fichero Excel "OMS_optimizacion personal_098_190209_01_00h_06m_12s" (referencia por trazabilidad de los cálculos).

Tiempo calculado para la estadía: 1.963 minutos = 32 horas y 43 minutos

Los tiempos de vinculación de cada recurso son:

Recurso	Inicio tareas (minutos)	Final tareas (minutos)	Tiempo de vinculación (minutos)
Bogie 1	537	1716	1179
Bogie 2	543	1265	722
Generalista 1	8	1953	1945
Generalista 2	8	1954	1946
Generalista 3	8	1716	1708
Generalista 4	10	1537	1527
Mecánico 1	0	1963	1963
Mecánico 2	10	1954	1944
Electricista 1	0	1963	1963
Electricista 2	11	1805	1794

Tabla XXIII Tiempos de vinculación de cada recurso a la estadía (simulación 1)

Suma de los tiempos de vinculación de los recursos: 16.691 minutos

Productividad: **55,60%**

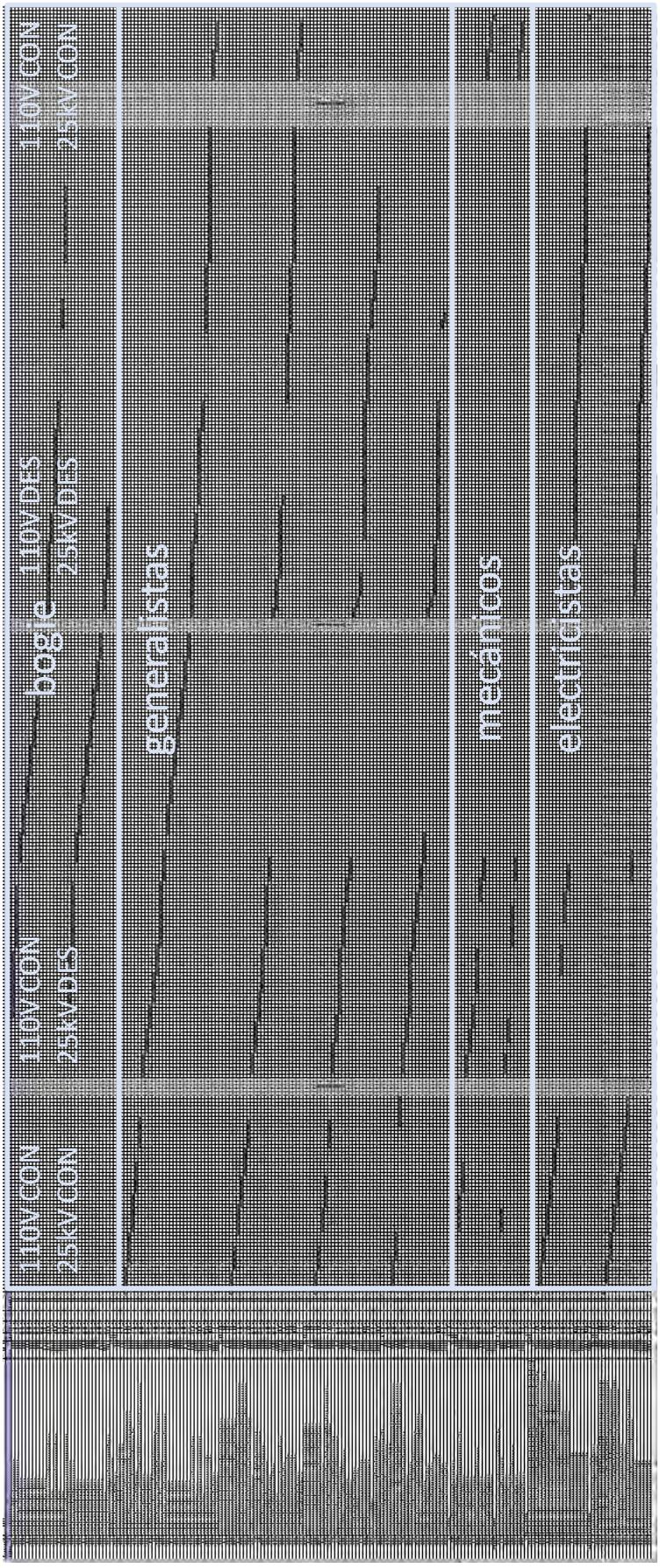


Figura 48 Cronograma de la primera simulación

Acerca del cronograma y del resultado de OMSA:

- Se realiza el número mínimo posible de cambios de estado:
  - Tareas con batería y 25kV hasta el minuto 421 ➔ desconexión 25kV finalizada en el minuto 451
  - Tareas con batería hasta el minuto 1.085 ➔ desconexión de batería finalizada en el minuto 1.100
  - Tareas sin batería ni 25kV hasta el minuto 1.805 ➔ Reconexión de batería y 25kV finalizada en el minuto 1.865
  - Últimas tareas con batería y 25kV hasta el minuto 1963
- El resultado cumple estrictamente las restricciones de personal, tiempos de traslado, estados de seguridad, etc.
- Los especialistas de bogie suponen un cuello de botella en la fase 110V CON / 25kV DES, en la que se ejecutan las tareas de "Comprobacion de sistema de freno", que requieren dos especialistas de bogies y un técnico general.
- Tres de los cuatro técnicos generalistas están desaprovechados durante la fase 110V CON / 25kV debido a la falta de especialistas de bogie.
- Los dos mecánicos tienen largos tiempos sin ocupación.
- Los dos electricistas definen el camino crítico durante la fase 110V CON / 25kV CON, y durante la realización de las tareas "Comprobacion equipo alta tension en techo" en la fase 110V DES / 25kV DES.

SEGUNDA SIMULACIÓN

En la asignación de tareas de la simulación anterior se observa que los mecánicos y tres de los técnicos generalistas están claramente desaprovechados. Por ello se decide reducir ambos tipos de recursos a 1 y 3, respectivamente.

Los recursos previstos para la realización de las tareas son los siguientes:

- Especialistas de bogie: 2
- Técnicos generalistas: 3
- Mecánicos: 1
- Electricistas: 2

Tiempo de ejecución del algoritmo: 3 minutos y 44 segundos

Los datos de entrada, la tabla con el resultado de OMSA y el cronograma se encuentran en el Fichero Excel "OMS_optimizacion personal_098_190209_02_00h_03m_44s" (referencia por trazabilidad).

Tiempo calculado para la estadía: 2.288 minutos = 38 horas y 8 minutos

Los tiempos de vinculación de cada recurso son:

Recurso	Inicio tareas (minutos)	Final tareas (minutos)	Tiempo de vinculación (minutos)
Bogie 1	640	1724	1084
Bogie 2	640	1680	1040
Generalista 1	7	2219	2212

Recurso	Inicio tareas (minutos)	Final tareas (minutos)	Tiempo de vinculación (minutos)
Generalista 2	7	2219	2212
Generalista 3	8	1728	1720
Mecánico 1	0	2288	2288
Electricista 1	0	2288	2288
Electricista 2	9	2219	2210

Tabla XXIV Tiempos de vinculación de cada recurso a la estadía (simulación 2)

Suma de los tiempos de vinculación de los recursos: 15.054 minutos

Productividad: **61,64%**

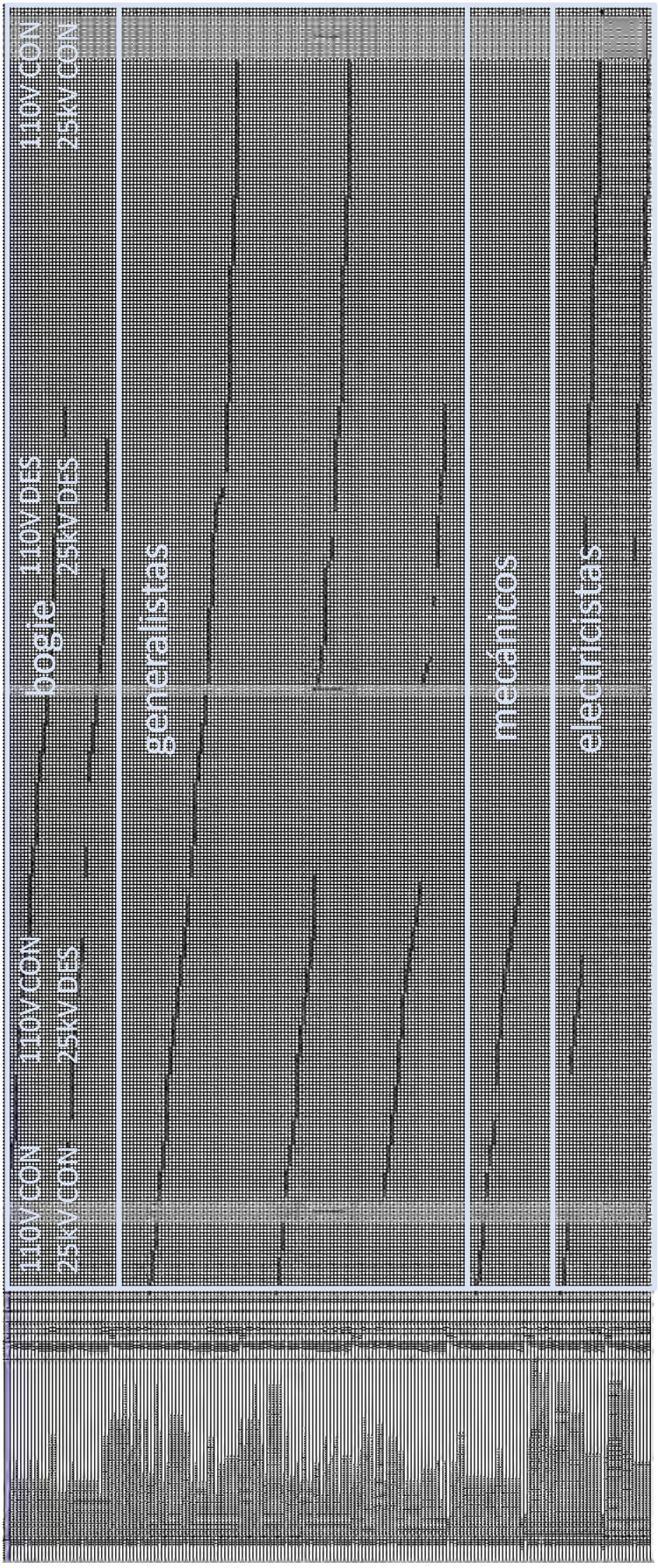


Figura 49 Cronograma de la segunda simulación

Acerca del cronograma y del resultado de OMSA:

- Se realiza el número mínimo posible de cambios de estado:
  - Tareas con batería y 25kV hasta el minuto 565 ➔ desconexión 25kV finalizada en el minuto 595
  - Tareas con batería hasta el minuto 1.313 ➔ desconexión de batería finalizada en el minuto 1.328
  - Tareas sin batería ni 25kV hasta el minuto 2.219 ➔ Reconexión de batería y 25kV finalizada en el minuto 2.279
  - Tarea final con batería y 25kV hasta el minuto 2.288
- El resultado cumple estrictamente las restricciones de personal, tiempos de traslado, estados de seguridad, etc.
- Los dos especialistas de bogie suponen un cuello de botella en la fase 110V CON / 25kV DES durante la ejecución de las tareas de "Comprobacion de sistema de freno", para las que se requieren dos especialistas de bogies y un técnico generalista.
- Dos de los técnicos generalistas están ocupados casi continuamente. La ocupación del tercero no es tan baja que se pueda renunciar a él.
- El mecánico es suficiente para realizar las tareas mecánicas.
- Los dos electricistas definen el camino crítico durante la fase 110V DES / 25kV DES para la realización de las tareas "Comprobacion equipo alta tension en techo". Estas tareas son especialmente largas.

TERCERA SIMULACIÓN

En la asignación de tareas de la simulación anterior se observa que los especialistas de bogie y los electricistas generan un cuello de botella por lo que se decide aumentar ambos tipos de recursos a 3 y 3.

Los recursos previstos para la realización de las tareas son los siguientes:

- Especialistas de bogie: 3
- Técnicos generalistas: 3
- Mecánicos: 1
- Electricistas: 3

Tiempo de ejecución del algoritmo: 3 minutos y 21 segundos

Los datos de entrada, la tabla con el resultado de OMSA y el cronograma se encuentran en el Fichero Excel "OMS_optimizacion personal_098_190209_03_00h_03m_21s" (referencia por trazabilidad).

Tiempo calculado para la estadía: 2.026 minutos = 33 horas y 46 minutos

Los tiempos de vinculación de cada recurso son:

Recurso	Inicio tareas (minutos)	Final tareas (minutos)	Tiempo de vinculación (minutos)
Bogie 1	621	1434	813
Bogie 2	621	1368	747
Bogie 3	631	1339	708

Recurso	Inicio tareas (minutos)	Final tareas (minutos)	Tiempo de vinculación (minutos)
Generalista 1	5	1957	1952
Generalista 2	5	1957	1952
Generalista 3	8	1663	1655
Mecánico 1	0	2026	2026
Electricista 1	0	2026	2026
Electricista 2	8	1957	1949
Electricista 3	10	1553	1543

Tabla XXV Tiempos de vinculación de cada recurso a la estadía (simulación 3)

Suma de los tiempos de vinculación de los recursos: 15.371 minutos

Productividad: **60,37%**

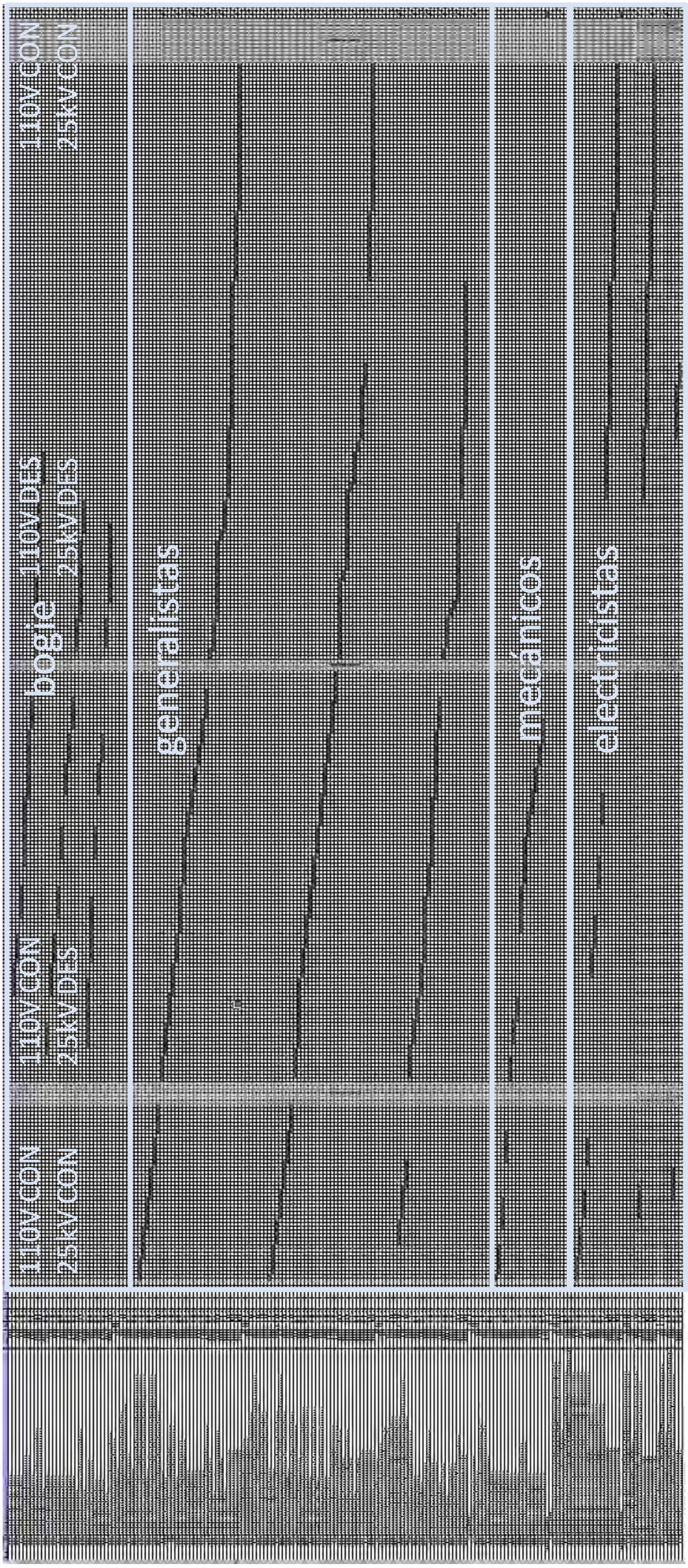


Figura 50 Cronograma de la tercera simulación

Acerca del cronograma y del resultado de OMSA:

- Se realiza el número mínimo posible de cambios de estado:
  - Tareas con batería y 25kV hasta el minuto 554 ➔ desconexión 25kV finalizada en el minuto 584
  - Tareas con batería hasta el minuto 1.139 ➔ desconexión de batería finalizada en el minuto 1.154
  - Tareas sin batería ni 25kV hasta el minuto 1.957 ➔ Reconexión de batería y 25kV finalizada en el minuto 2.019
  - Tarea final con batería y 25kV hasta el minuto 2.026
- El resultado cumple estrictamente las restricciones de personal, tiempos de traslado, estados de seguridad, etc.
- Los tres especialistas de bogie no son un cuello de botella en la fase 110V CON / 25kV DES durante la ejecución de las tareas de "Comprobacion de sistema de freno", para las que se requieren dos especialistas de bogies y un técnico generalista.
- Los tres técnicos generalistas están ocupados casi continuamente.
- El mecánicoes suficiente para realizar las tareas mecánicas.
- Dos de los electricistas definen el camino crítico durante la fase 110V DES / 25kV DES para la realización de las tareas "Comprobacion equipo alta tension en techo". Estas tareas son especialmente largas.

CUARTA SIMULACIÓN

En la asignación de tareas de la simulación se aprecia que los tres especialistas de bogies y el mecánico tienen tiempos de espera debido a que no hay suficientes técnicos generalistas, que son necesarios para asistir al mecánico y a los especialistas de bogies en sus tareas. Por ello se asigna un técnico generalista adicional. Puesto que las tareas de bogies se realizan en su mayoría por dos especialistas de bogies se añade un especialista de bogies adicional para tener un número par.

Los recursos previstos para la realización de las tareas son los siguientes:

- Especialistas de bogie: 4
- Técnicos generalistas: 4
- Mecánicos: 1
- Electricistas: 3

Tiempo de ejecución del algoritmo: 5 minutos y 10 segundos

Los datos de entrada, la tabla con el resultado de OMSA y el cronograma se encuentran en el Fichero Excel "OMS_optimizacion personal_098_190216_04_00h_05m_10s" (referencia por trazabilidad).

Tiempo calculado para la estadía: 1.897 minutos = 31 horas y 37 minutos

Los tiempos de vinculación de cada recurso son:

Recurso	Inicio tareas (minutos)	Final tareas (minutos)	Tiempo de vinculación (minutos)
Bogie 1	216	907	691

Recurso	Inicio tareas (minutos)	Final tareas (minutos)	Tiempo de vinculación (minutos)
Bogie 2	216	783	567
Bogie 3	256	795	539
Bogie 4	256	866	610
Generalista 1	8	1888	1880
Generalista 2	216	1732	1516
Generalista 3	216	1723	1507
Generalista 4	217	1731	1514
Mecánico 1	0	1897	1897
Electricista 1	0	1897	1897
Electricista 2	8	1732	1724
Electricista 3	9	173	164

Tabla XXVI Tiempos de vinculación de cada recurso a la estadía (simulación 4)

Suma de los tiempos de vinculación de los recursos: 14.506 minutos

Productividad: **63,97%**

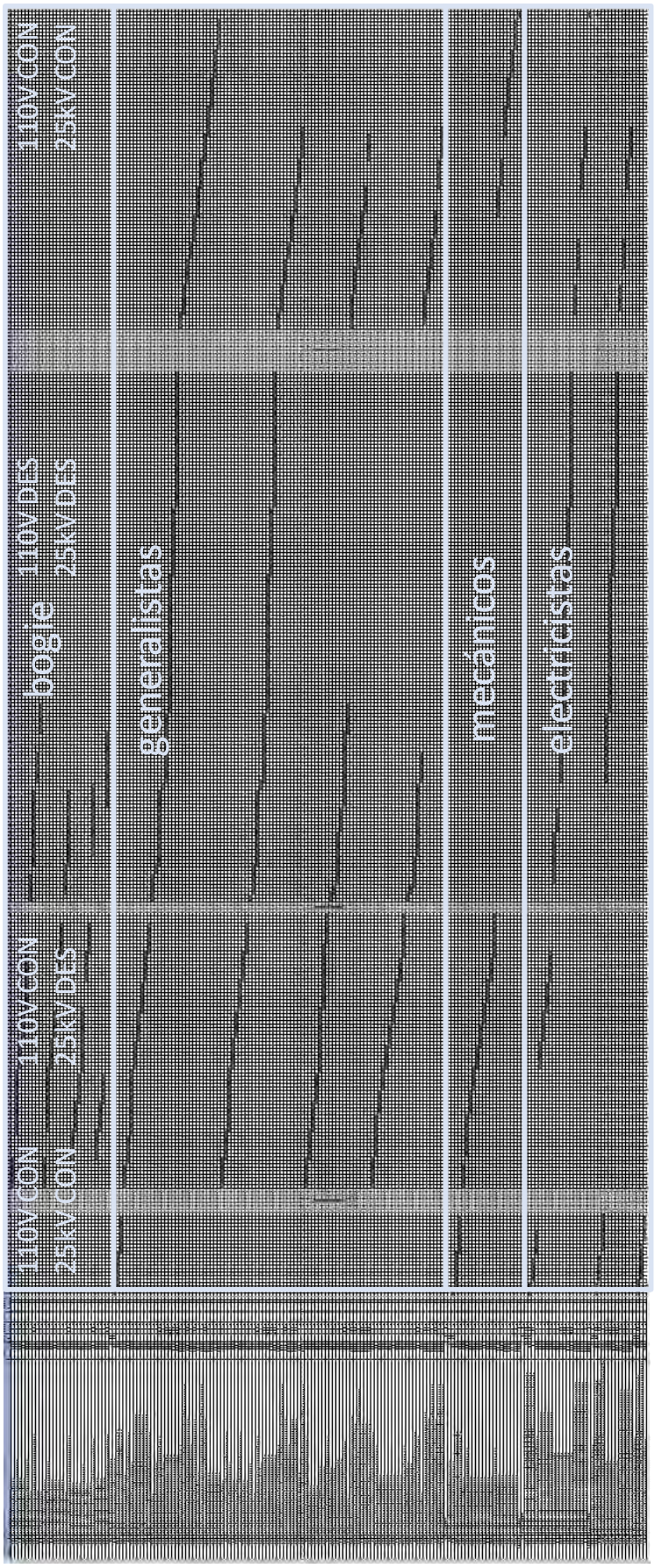


Figura 51 Cronograma de la cuarta simulación

Acerca del cronograma y del resultado de OMSA:

- Se realiza el número mínimo posible de cambios de estado:
  - Tareas con batería y 25kV hasta el minuto 179 ➔ desconexión 25kV finalizada en el minuto 209
  - Tareas con batería hasta el minuto 609 ➔ desconexión de batería finalizada en el minuto 624
  - Tareas sin batería ni 25kV hasta el minuto 1.385 ➔ Reconexión de batería y 25kV finalizada en el minuto 1.445
  - Nuevamente tareas con batería y 25kV hasta el minuto 1.897
- El resultado cumple estrictamente las restricciones de personal, tiempos de traslado, estados de seguridad, etc.
- Los cuatro especialistas de bogie no son un cuello de botella en la fase 110V CON / 25kV DES durante la ejecución de las tareas de "Comprobacion de sistema de freno", para las que se requieren dos especialistas de bogies y un técnico generalista.
- Dos de los técnicos generalistas están ocupados casi continuamente. La ocupación del tercero y del cuarto durante la fase 110V DES / 25kV DES se debe a que sólo hay dos electricistas y no es posible formar un equipo adicional de dos electricistas más dos técnicos generalistas para realizar las tareas "Comprobación equipo alta tension en techo"..
- El mecánico es suficiente para realizar las tareas mecánicas.
- Dos de los electricistas definen el camino crítico durante la fase 110V DES / 25kV DES para la realización de las tareas "Comprobacion equipo alta tension en techo". Estas tareas son especialmente largas.

QUINTA SIMULACIÓN

En la asignación de tareas de la simulación anterior se observa que las tareas eléctricas en el techo forman un cuello de botella porque para su realización precisan de dos electricistas más dos técnicos generalistas. Dado que hay tres electricistas disponibles, el aprovechamiento óptimo del tercer electricista requiere que haya un segundo electricista y los técnicos generalistas suficientes. Por ello se decide aumentar ambos tipos de recursos a 4 y 5, respectivamente.

Los recursos previstos para la realización de las tareas son los siguientes:

- Especialistas de bogie: 4
- Técnicos generalistas: 5
- Mecánicos: 1
- Electricistas: 4

Tiempo de ejecución del algoritmo: 43 minutos y 25 segundos

Los datos de entrada, la tabla con el resultado de OMSA y el cronograma se encuentran en el Fichero Excel "OMS_optimizacion personal_098_190224_05_00h_43m_25s" (referencia por trazabilidad).

Tiempo calculado para la estadía: 1.462 minutos = 24 horas y 22 minutos

Los tiempos de vinculación de cada recurso son:

Recurso	Inicio tareas (minutos)	Final tareas (minutos)	Tiempo de vinculación (minutos)
Bogie 1	488	1135	647
Bogie 2	488	1044	556
Bogie 3	527	1093	566
Bogie 4	527	1013	486
Generalista 1	7	1339	1332
Generalista 2	7	1391	1384
Generalista 3	9	1339	1330
Generalista 4	9	1391	1382
Generalista 5	10	1294	1284
Mecánico 1	0	1462	1462
Electricista 1	0	1462	1462
Electricista 2	7	1339	1332
Electricista 3	7	1339	1332
Electricista 4	8	1391	1383

Tabla XXVII Tiempos de vinculación de cada recurso a la estadía (simulación 5)

Suma de los tiempos de vinculación de los recursos: 15.938 minutos

Productividad: **58,23%**

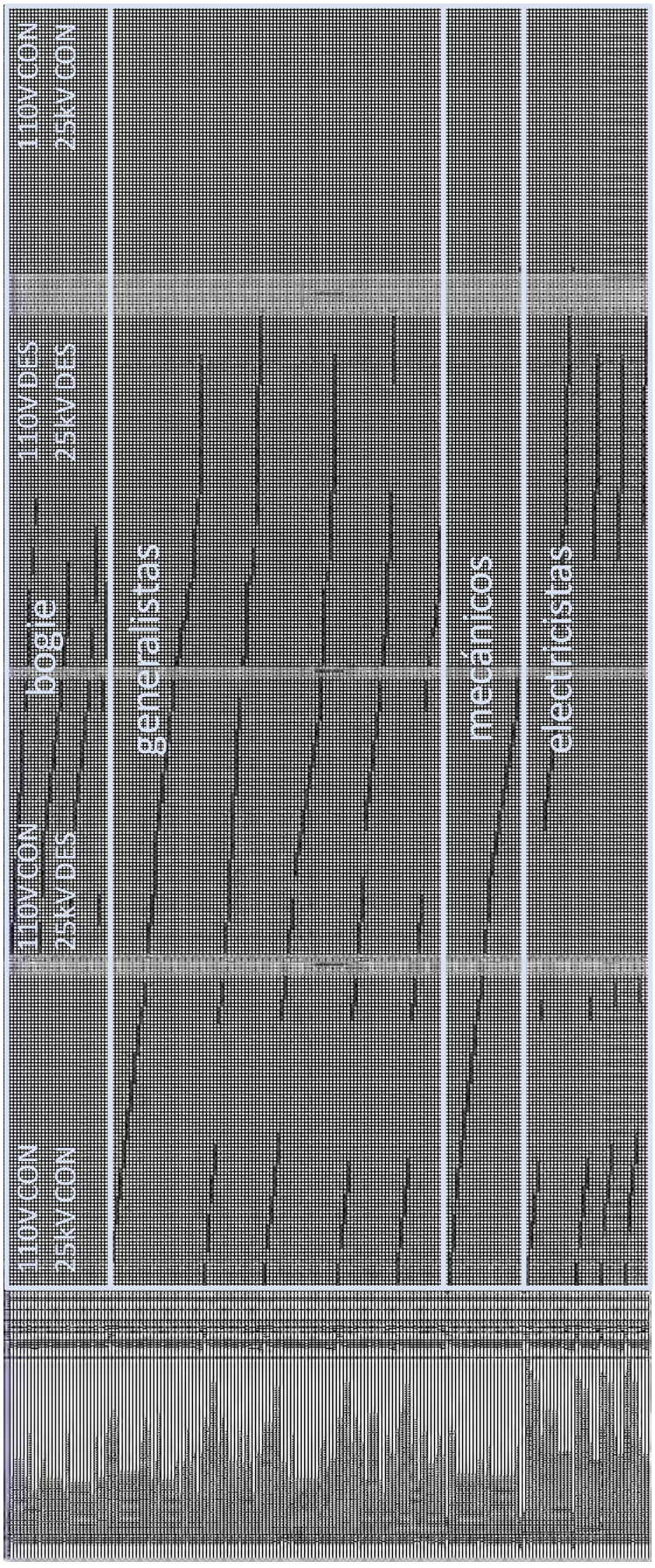


Figura 52 Cronograma de la quinta simulación

Acerca del cronograma y del resultado de OMSA:

- Se realiza el número mínimo posible de cambios de estado:
  - Tareas con batería y 25kV hasta el minuto 451 ➔ desconexión 25kV finalizada en el minuto 481
  - Tareas con batería hasta el minuto 879 ➔ desconexión de batería finalizada en el minuto 894
  - Tareas sin batería ni 25kV hasta el minuto 1.391 ➔ Reconexión de batería y 25kV finalizada en el minuto 1.451
  - Tarea final con batería y 25kV hasta el minuto 1.462
- El resultado cumple estrictamente las restricciones de personal, tiempos de traslado, estados de seguridad, etc.
- Los cuatro especialistas de bogie no son un cuello de botella en la fase 110V CON / 25kV DES durante la ejecución de las tareas de "Comprobacion de sistema de freno", para las que se requieren dos especialistas de bogies y un técnico general.
- Cuatro de los técnicos generalistas están ocupados casi continuamente. La ocupación del quinto no es tan baja que se pueda renunciar a él.
- El recurso de tipo "mecánico" es un cuello de botella en la fase 110V CON / 25kV CON.
- Dos de los recursos de tipo "eléctrico" definen el camino crítico durante la fase 110V DES / 25kV DES para la realización de las tareas "Comprobacion equipo alta tension en techo". Estas tareas son especialmente largas.

SEXTA SIMULACIÓN

En la asignación de tareas de la simulación anterior se observa que el mecánico genera un cuello de botella en la fase 110V CON / 25kV DES. Por ello se decide aumentarlos a 2.

Los recursos previstos para la realización de las tareas son los siguientes:

- Especialistas de bogie: 4
- Técnicos generalistas: 5
- Mecánicos: 2
- Electricistas: 4

Tiempo de ejecución del algoritmo: 17 minutos y 0 segundos

Los datos de entrada, la tabla con el resultado de OMSA y el cronograma se encuentran en el Fichero Excel "OMS_optimizacion personal_098_190310_06_00h_17m_00s" (referencia por trazabilidad).

Tiempo calculado para la estadía: 1.312 minutos = 21 horas y 52 minutos

Los tiempos de vinculación de cada recurso son:

Recurso	Inicio tareas (minutos)	Final tareas (minutos)	Tiempo de vinculación (minutos)
Bogie 1	345	1039	694
Bogie 2	345	1092	747
Bogie 3	374	753	379
Bogie 4	374	765	391

Recurso	Inicio tareas (minutos)	Final tareas (minutos)	Tiempo de vinculación (minutos)
Generalista 1	6	1241	1235
Generalista 2	6	1152	1146
Generalista 3	8	1152	1144
Generalista 4	8	1092	1084
Generalista 5	43	1241	1198
Mecánico 1	0	1312	1312
Mecánico 2	8	590	582
Electricista 1	0	1312	1312
Electricista 2	8	1152	1144
Electricista 3	9	1152	1143
Electricista 4	11	1241	1230

Tabla XXVIII Tiempos de vinculación de cada recurso a la estadía (simulación 6)

Suma de los tiempos de vinculación de los recursos: 14.741 minutos

Productividad: **62,95%**

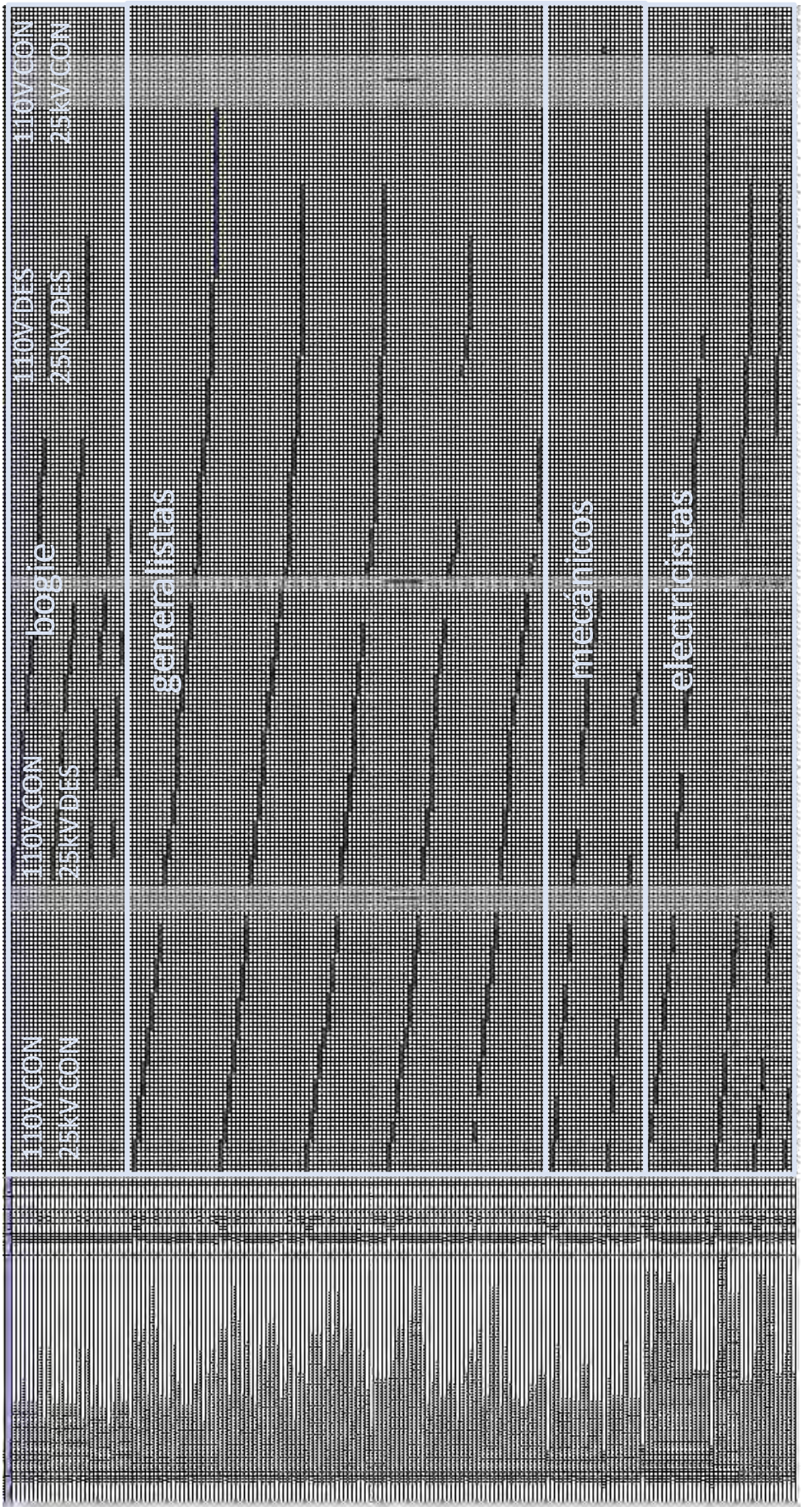


Figura 53 Cronograma de la sexta simulación

Acerca del cronograma y del resultado de OMSA:

- Se realiza el número mínimo posible de cambios de estado:
  - Tareas con batería y 25kV hasta el minuto 307 ➔ desconexión 25kV finalizada en el minuto 337
  - Tareas con batería hasta el minuto 681 ➔ desconexión de batería finalizada en el minuto 696
  - Tareas sin batería ni 25kV hasta el minuto 1.241 ➔ Reconexión de batería y 25kV finalizada en el minuto 1.301
  - Tarea final con batería y 25kV hasta el minuto 1.312
- El resultado cumple estrictamente las restricciones de personal, tiempos de traslado, estados de seguridad, etc.
- Los cuatro especialistas de bogie no son un cuello de botella en la fase 110V CON / 25kV DES durante la ejecución de las tareas de "Comprobacion de sistema de freno", para las que se requieren dos especialistas de bogies y un técnico generalista.
- Los cinco técnicos generalistas están ocupados casi continuamente.
- Los dos mecánicos están bien ocupados.
- Los cuatro electricistas están bien ocupados.

CONCLUSIONES

La Tabla XXIX ilustra los resultados de las simulaciones con diferentes composiciones del equipo de mantenimiento:

Simulación	**1**	**2**	**3**	**4**	**5**	**6**
Especialistas de bogie	2	2	3	4	4	4
Técnicos generalistas	4	3	3	4	5	5
Mecánicos	2	1	1	1	1	2
Electricistas	2	2	3	3	4	4
Tiempo de ejecución de OMSA	00:06:12	00:03:44	00:03:21	00:05:10	00:43:25	00:17:00
Tiempo de estadía (minutos)	1.963	2.288	2.026	1.897	1.462	1.312
Tiempo de estadía	32:43:00	38:08:00	33:46:00	31:37:00	24:22:00	21:52:00
Productividad	55,60%	61,64%	60,37%	63,97%	58,23%	62,95%

Tabla XXIX Tabla resumen de los resultados de las simulaciones

La simulación 1 arroja un tiempo de estadía de 32:43:00 y una productividad del 55,60% de los recursos. Al analizar el cronograma y las tareas asignadas a cada recurso se observa que un técnico generalista y un mecánico tienen baja ocupación y tiempos de espera largos, y se decide eliminarlos en la siguiente simulación.

La simulación 2 ocupa por tanto a dos recursos menos. El tiempo de estadía se prolonga a 38:08:00, siendo el mayor de todas las simulaciones. A cambio, la productividad de los recursos aumenta a un 61,64% dado que los recursos tienen menores tiempos de espera. En el

cronograma y en la asignación de tareas a los recursos se aprecia que el camino crítico pasa por los especialistas de bogies y por los electricistas, de manera que se decide complementar el equipo con un especialista de bogie y un electricista adicionales para reducir el tiempo de estadía.

La simulación 3 vuelve a ocupar 10 recursos, como la simulación 1. Sin embargo, la ocupación de los recursos es más elevada y permite un aumento notable de la productividad a un 60,37%. El tiempo de estadía total es de un 33:46:00. El cronograma y las asignaciones de tareas a los recursos apuntan a que los especialistas de bogie siguen estando en el camino crítico. En particular, tiene sentido planificar un número par de especialistas de bogie porque en varias tareas deben trabajar simultáneamente dos especialistas en el mismo bogie. Los técnicos generalistas también están en el camino crítico y se decide asignar uno más hasta un total de 4.

La simulación 4 ocupa a 12 recursos y alcanza con un 63,97% la productividad más elevada de todas las simulaciones. El tiempo de estadía asciende a 31:37:00. Del cronograma y de las asignaciones de tareas a los recursos se desprende que un electricista más un técnico generalista adicionales permitirían la ejecución en paralelo de más tareas de inspección y mantenimiento del equipamiento eléctrico del techo, tareas que han pasado a encontrarse en el camino crítico.

En la simulación 5 se ocupa a un total de 14 recursos y se consigue una notable reducción del tiempo de estadía hasta 24:22:00. Sin embargo, la productividad desciende al 58,23%. De nuevo el cronograma y las asignaciones de tareas a los recursos hacen patente que existe un recurso bloqueando el camino crítico, en este caso se trata del mecánico. Asignando un segundo mecánico al equipo se consigue que varias tareas, por ejemplo, la inspección y mantenimiento del morro más la cinemática, se realicen en paralelo.

En la simulación 6 y última se reduce el tiempo de estadía a su mínimo con 21:52:00 mientras que se vuelve a aumentar la productividad al 62,95%, el segundo valor más alto tras el de la simulación 4.

Como se puede apreciar, OMSA permite realizar análisis del impacto de diferentes composiciones del equipo de mantenimiento, tanto para averiguar los tiempos de estadía posibles como para determinar el equipo de mantenimiento de mayor productividad.

4.5.3 ANÁLISIS DE RETORNO DE INVERSIONES EN INFRAESTRUCTURA

En el apartado “Comparación con un Caso Real”, página 87, se estudió la realización de diversas tareas de mantenimiento de la intervención M1 (100.000 km) de un tren de alta velocidad. La particularidad más llamativa del caso era que el taller no cumplía con los requisitos habituales de acceso al techo del vehículo y bajo bastidor. Debido a estas limitaciones los tiempos necesarios para acceder al techo y a la zona bajo bastidor del tren eran muy dilatados por las maniobras del tren a los puestos con foso y a la plataforma de acceso al techo.

Es posible realizar una simulación del tiempo de estadía resultante si la infraestructura cumpliera los requisitos habituales. Para ello se toma la lista de tareas del caso analizado y que se encuentra en la Tabla XX, página 91, y se sustituyen los siguientes datos de entrada:

- tabla de cambios de estado usada en el caso real (véase la Tabla XIX, página 88) por la tabla de cambios de estado usada en el resto de simulaciones (véase la Tabla XXI, página 97)
- tiempos de desplazamiento entre localizaciones usada en el caso real (véase la Figura 43, página 89) por los tiempos de desplazamiento usados en el resto de simulaciones (véase la Figura 2, página 5).

Se mantienen los mismos recursos:

- Especialistas en bogies: 2
- Técnico generalista: 1

El resultado es una reducción del tiempo de estadía de 3.050 minutos a 2.416 minutos. Los beneficios de un cambio de infraestructura son los siguientes:

- Reducción de los costes de personal directo de mantenimiento en un 20,79%.
- Eliminación del coste del maquinista de maniobras.
- Eliminación de los costes de mantenimiento y amortización de la locomotora de maniobras.
- Aumento de la disponibilidad del tren al completarse antes el mantenimiento.
- Eliminación de los riesgos laborales asociados a las maniobras.

El cálculo de la amortización de la inversión depende de la valoración global de estos beneficios y del coste de construcción de la vía con foso y con plataforma corrida de acceso al techo.

OMSA aporta un dato fundamental en el análisis de retorno de la inversión al poder cuantificar objetivamente la reducción de costes de personal directo de mantenimiento y el aumento de la disponibilidad.

4.5.4 DESARROLLO DE UNA ESTRATEGIA MANTENIMIENTO DISTRIBUIDO

En el apartado “Aumento de la Productividad del Personal”, página 102 y siguientes, se aplica OMSA para analizar la productividad del equipo de mantenimiento y la duración de la estadía dependiendo de la composición del equipo de mantenimiento.

La sexta simulación arroja una estadía de 1.312 minutos, es decir, de 21 horas y 52 minutos, lo que se corresponde con casi tres turnos laborales. El intervalo de la intervención analizada es de 100.000 km y se corresponde con una estadía cada tres meses si para cada tren asumimos un kilometraje medio anual de 400.000 km. Por lo tanto, el equipo que realiza esta intervención estará ocupado tres días de cada tres meses. En una flota grande se puede garantizar un nivel de ocupación regular y constante, pero en flotas pequeñas la fluctuación del aprovechamiento de recursos es grande.

La disponibilidad de la flota se ve afectada por las estadías de manera semejante. La parada de un vehículo durante tres días en una flota grande no es un problema, pero sí que supone un considerable impacto en una flota pequeña. Desde el punto de vista del operador ferroviario es esencial mantener una disponibilidad constante de la flota.

La solución para minimizar las fluctuaciones del aprovechamiento de recursos y de la disponibilidad de la flota es la distribución de las tareas de la intervención en estadías más cortas. Esta estrategia se denomina mantenimiento distribuido.

El mantenimiento distribuido consiste en dividir la intervención en módulos que puedan realizarse durante una estadía corta, preferiblemente durante un turno de noche. De este modo la carga de trabajo del equipo de mantenimiento es más regular y se aumenta la disponibilidad del vehículo. La Figura 54 ilustra este concepto.

OMSA es capaz de distribuir las tareas de una intervención. Para ello se limita el tiempo máximo de cada estadía y se invoca OMSA con el intervalo en el que se desea realizar los módulos de mantenimiento distribuido. Al ejecutar OMSA por primera vez, adjudica una prioridad 2 a las tareas puesto que todavía no se ha ejecutado ninguna (véase el apartado “Clase *Task*”, página

73 y siguientes, para información detallada del método de priorización) y así se incluyen en el módulo tantas tareas como sea posible hasta alcanzar el tiempo límite. Las tareas se van agrupando en los módulos y se puede analizar si es posible distribuir la intervención según los parámetros deseados.

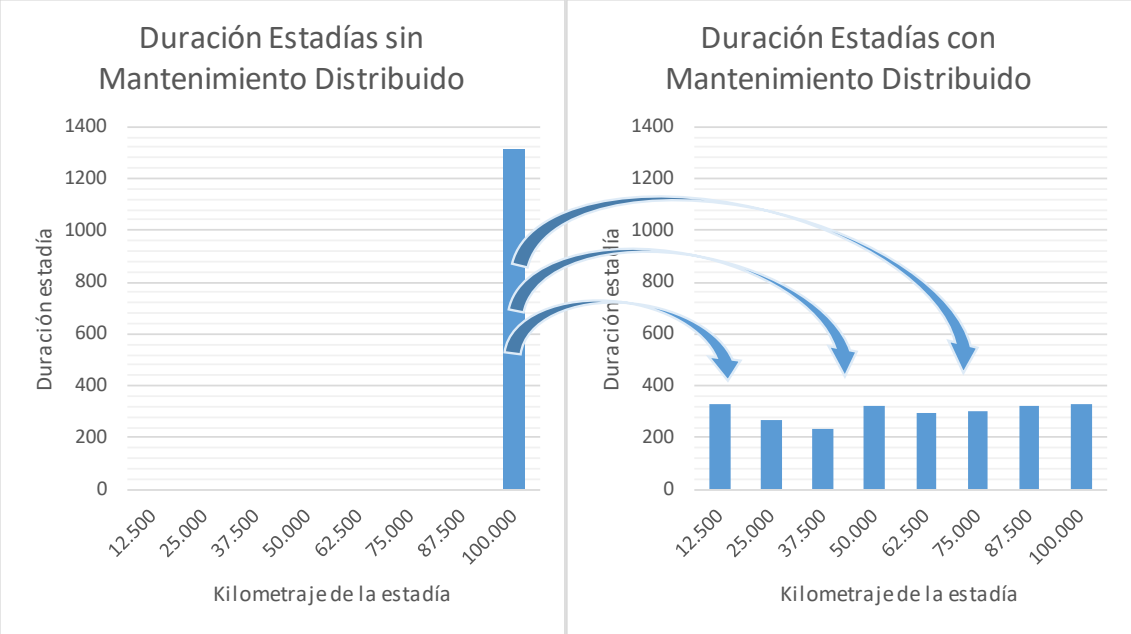


Figura 54 Concepto de mantenimiento distribuido

En este estudio se distribuirá la intervención analizada en el apartado “Sexta Simulación”, página 118 y siguientes, con la misma configuración del equipo de mantenimiento. Se desea distribuir la intervención en:

- Módulos con intervalos de 12.500 km
- Estadías de máximo 5 horas y 30 minutos (330 minutos)

Tiempo de ejecución del algoritmo: 1 minuto y 23 segundos

Los datos de entrada, la tabla con el resultado de OMSA y los cronogramas de los módulos se encuentran en el Fichero Excel “OMS_Mantenimiento distribuido_098_190324_11_00h_01m_23s” (referencia por trazabilidad).

OMSA es capaz de separar las tareas de la intervención en 8 módulos diferentes que se repiten cada 100.000 km. Cada módulo contiene siempre las mismas tareas, pero en cada repetición el orden de las tareas en la lista de entrada es ligeramente diferente, por lo que el flujograma resultado de OMSA y la duración total de la estadía cambia ligeramente. De esta manera es posible seleccionar el flujograma más corto de los calculados para el módulo.

Figura 55 Mantenimiento distribuido: duración de las estadías de los módulos

La Figura 55 representa las duraciones de las estadías. El módulo realizado a los 12.500 km se repite a los 112.500 km, 212.500 km, 312.500 km, etc.. El módulo realizado a los 25.000 km se repite a los 125.000 km, 225.000 km, 325.000 km, etc.. Es posible seleccionar el flujograma de duración más corta para cada módulo:

Módulo	Duración mínima	Estadía (km)
Módulo I (12.500 km)	328	12.500
Módulo II (25.000 km)	267	25.000
Módulo III (37.500 km)	232	37.500
Módulo IV (50.000 km)	320	50.000
Módulo V (62.500 km)	292	62.500
Módulo VI (75.000 km)	304	575.000
Módulo VII (87.500 km)	319	487.500
Módulo VIII (100.000 km)	326	600.000

Tabla XXX Estadías mínimas de cada módulo de mantenimiento distribuido

Aplicando los flujogramas calculados para las estadías indicadas en la Tabla XXX se consigue un mantenimiento distribuido de la intervención de 100.000 km.

La Figura 56 muestra los porcentajes de aprovechamiento del intervalo de 100.000 km en cada estadía. El porcentaje de aprovechamiento es el cociente entre el kilometraje recorrido desde la última ejecución de una tarea de mantenimiento y el intervalo de la intervención a la que pertenece. Se aprecia como durante las primeras estadías el porcentaje de aprovechamiento es bajo, puesto que se realizan tareas por primera vez, hasta que se alcanza un aprovechamiento del 100% en todas las estadías.

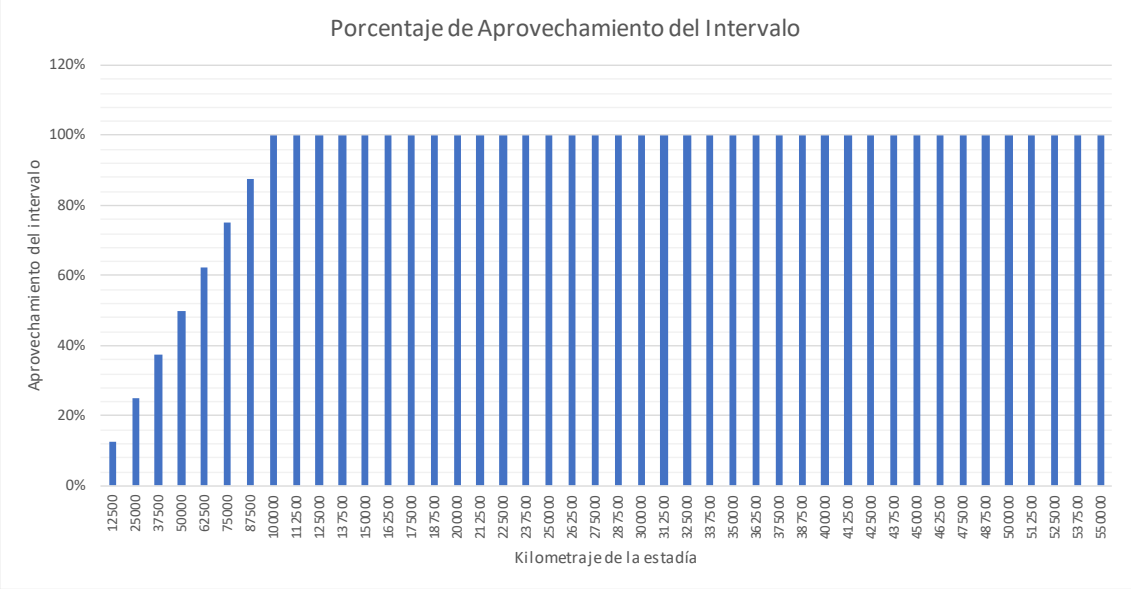


Figura 56 Mantenimiento distribuido: porcentajes de aprovechamiento del intervalo

4.5.5 DESARROLLO DE UNA ESTRATEGIA DE MANTENIMIENTO OPORTUNISTA

El mantenimiento oportunista consiste en aprovechar paradas operacionales para realizar tareas de mantenimiento. En el mantenimiento distribuido asumimos unos intervalos fijos (12.500 km en el ejemplo del apartado anterior) y un tiempo de estadía fijo (330 minutos). En el mantenimiento oportunista ni el intervalo ni el tiempo de estadía son regulares y se comportan de acuerdo con una función de distribución estadística.

La aletoriedad en el mantenimiento oportunista lleva a que no se distribuyan las tareas de mantenimiento en módulos fijos. El tiempo disponible en cada estadía es variable y en él se planifican y ejecutan todas las tareas de prioridad uno más las de prioridad dos en el tiempo sobrante, si lo hay.

OMSA se ejecuta igual que en el estudio del apartado anterior, excepto por los intervalos recorridos y el tiempo disponible de estadía, que se modelan con sendas variables aleatorias con funciones de distribución de Weibull. La Tabla XXXI muestra los parámetros k de forma y λ de escala.

Módulo	Parámetro de forma k	Parámetro de escala λ
Variable aleatoria del intervalo entre estadías (km)	5,155	12.500
Variable aleatoria del tiempo disponible de estadía (min)	20	340

Tabla XXXI Parámetros de forma y de escala de las variables aleatorias empleadas

Se han empleado funciones de distribución de Weibull debido a su facilidad de programación, a su versatilidad y a que habitualmente se emplean para modelar intervalos entre dos sucesos (véase el capítulo “Función de Distribución Estadística de Weibull”, página 25). El parámetro de forma de la función estadística del intervalo entre estadías se basan en datos reales de explotación comercial de una flota de trenes de alta velocidad. Su parámetro de escala es 12.500 km por paralelismo con el caso de estudio de mantenimiento distribuido. Para el parámetro de

forma de la función de distribución estadística del tiempo disponible de estadía se ha tomado un valor alto para limitar la varianza. Su parámetro de escala es 340 minutos, para conseguir un valor medio de 330 minutos, también por paralelismo con el caso de estudio de mantenimiento distribuido.

Tiempo de ejecución del algoritmo: 2 minutos y 18 segundos

Los datos de entrada, la tabla con el resultado de OMSA y los cronogramas se encuentran en el Fichero Excel "OMS_Mantenimiento oportunista_098_190324_03_00h_02m_18s" (referencia por trazabilidad).

Los tiempos de cada estadía están representados en la Figura 57.

Figura 57 Mantenimiento oportunista: duración de las estadías con intervalos y tiempos de estadía variables

Se observa que en general se consigue realizar las tareas en el tiempo disponible. La excepción más llamativa ocurre en la estadía de 328.729 km. En este caso, la estadía anterior había tenido lugar a los 314.021 km y muchas tareas habían pasado a tener prioridad uno durante los 14.708 km transcurridos.

La estadía de 397.885 km representa el caso contrario. No se realiza ninguna tarea porque la estadía anterior había tenido lugar a 390.265 km, sólo a 7.620 km, y no había ninguna tarea pendiente ni de prioridad uno ni de prioridad dos.

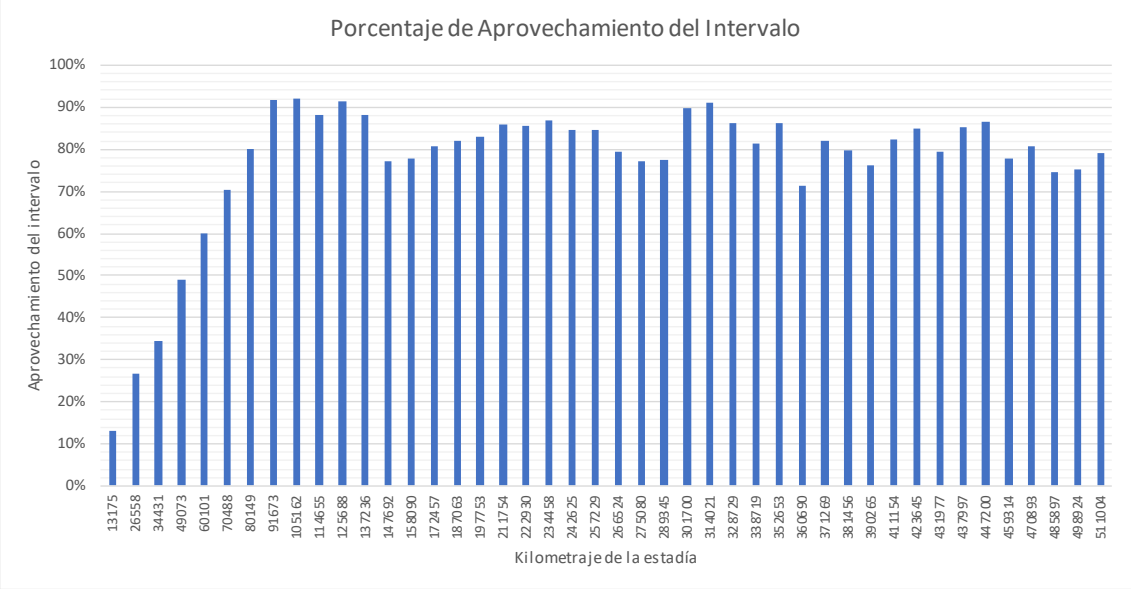


Figura 58 Mantenimiento oportunista: porcentajes de aprovechamiento del intervalo

Al contrario que en el caso del mantenimiento distribuido en el que se asumían unos intervalos y unos tiempos disponibles fijos, en el mantenimiento oportunista los porcentajes de aprovechamiento del intervalo son variables. Es posible aumentar el porcentaje de aprovechamiento si reducimos la variable *thresholdP1* (véase apartado "Clase *Task*", página 73 y siguientes). Cuanto menor sea su valor, tanto más esperará OMSA para ejecutar la tarea. El efecto negativo de un valor pequeño de *thresholdP1* es que se sobrepasará con más probabilidad el intervalo de la intervención, lo cual no es admisible para tareas relevantes para la seguridad.

5. CONCLUSIONES Y LÍNEAS DE INVESTIGACIÓN FUTURAS

"Compañeros, ya estoy harto de empaquetar el bagaje, de marchar, de correr, de llevar las armas, de ir en formación, de montar guardia, de combatir... Ahora que hemos llegado al mar, mi deseo es descansar ya de una vez de tantas penurias y navegar lo que queda de viaje hasta Grecia tirado sobre la cubierta como Odiseo".

Jenofonte, Anábasis, Libro V, 2 (Leonte de Turios al llegar al mar en Trapezunte, Cólquide).

En esta tesis se ha presentado el problema de secuenciación de tareas de mantenimiento en trenes de alta velocidad. Esta secuenciación de tareas está sujeta a una serie de restricciones para las que no se ha encontrado ninguna solución en el estado del arte.

Debido a ello, se ha desarrollado un novedoso método que mediante un algoritmo cuasi-genético analiza sucesivos flujogramas buscando la mejor solución. Los flujogramas se representan por medio de la matriz de conectividad de su grafo y en esta tesis se han desarrollado diversas técnicas para su manipulación y análisis.

El algoritmo desarrollado (*Operational Maintenance Sequencing Algorithm*, OMSA) se ha aplicado con éxito a diversos casos de estudio:

- Planificación de estadías
- Aumento de la productividad del personal de mantenimiento
- Análisis del retorno de inversiones en infraestructura
- Desarrollo de una estrategia de mantenimiento distribuido
- Desarrollo de una estrategia de mantenimiento oportunista

Estas técnicas permiten realizar mejoras claves en la gestión de activos, reduciendo costes de mantenimiento, aumentando la disponibilidad de la flota y creando una clara ventaja competitiva para los operadores y mantenedores ferroviarios.

Pese a los avances alcanzados, OMSA es susceptible de mejora:

- Disminución de la dependencia del resultado según el orden de las tareas de mantenimiento en la lista de entrada. Esto se puede conseguir mediante la superposición de un algoritmo genético tal como se propone al final del apartado “Robustez del Algoritmo”, a partir de la página 97.
- Aumento de la rapidez de ejecución del algoritmo mediante la implementación en un lenguaje de programación compilado tal como se propone al final del apartado “Planificación de Estadías”, a partir de la página 101.
- Programación de un interfaz gráfico. En su estado actual, la entrada y salida de datos se realiza mediante tablas Excel y no es fácilmente manejable.

Pese a que todo el proyecto de investigación ha sido enfocado al mantenimiento de material rodante ferroviario de alta velocidad, este método puede aplicarse igualmente a activos cuyo mantenimiento esté sujeto a restricciones semejantes.

6. BIBLIOGRAFÍA Y REFERENCIAS

Aguirre, A. M., & Papageorgiu, L. G. (2018). Medium-term optimization-based approach for the integration of production planning, scheduling and maintenance. Computers and Chemical Engineering. doi: 10.1016/j.compchemeng.2018.04.030

Ait-Kadi, D., Menje, J.-B., & Kane, H. (2011). Resources assignment model in maintenance activities scheduling. International Journal of Production Research, vol.49 no. 22, 6677-6689.

Apallius de Vos, J. I., & Van Dongen, L. (2015). Performance Centered Maintenance as a core policy in strategic maintenance control. Procedia CIRP, 255-258. doi: 10.1016/j.procir.2015.08.016

Biggs, N. (1996). Algebraic Graph Theory. Cambridge University Press.

Budai, G., Huisman, D., & Dekker, R. (2004). Scheduling preventive railway maintenance activities. IEEE International Conference on Systems, Man and Cybernetics.

Busstra, M., & Van Dongen, L. (2015). Creating value by integrating logistic trains services and maintenance activities. Procedia CIRP, 250-254. doi: 10.1016/j.procir.2015.07.051

Carrese, S., & Ottone, G. (2006). A model for the management of a tram fleet. European Journal of Operational Research, Vol. 175, 1628-1651.

Dao, S. D., Abhary, K., & Marian, R. (2016). An improved genetic algorithm for multidimensional optimization. Journal of Industrial Engineering International. doi: 10.1007/s40092-016-0181-7

Dekker, R., & Smeitink, E. (1994). Preventive Maintenance at Opportunities of Restricted Duration. Naval Reserach Logistics, Vol. 41, 335-353.

Gould, R. (2012). Graph Theory. New York: Dover Publications, Inc.

Huan, C. (2017). Intelligent maintenance scheduling system for maximum performance of solar-energy generating system. Sensors and Materials, 1579-1588. doi: 10.18494/sam.2017.1706

Jonsson, M.T. (2015). Power Plant Maintenance Scheduling using Dependency Structure Matrix and Evolutionary Optimization. Proceedings of the World Congress on Engineering and Computer Science 2015, Vol. II.

Keizer, M.C.A.O., Teunter, R. H., & Veldman, J. (2016). Clustering condition-based maintenance for systems with redundancy and economic dependencies. European Journal of Operational Research doi:10.1016/j.ejor.2015.11.008

Kramer, O. (2017). Genetic Algorithm Essentials. Cham, Switzerland: Springer-Verlag. doi:10.1007/978-3-319-52156-5

Lopes, I. S., Senra, P., Neto, B., Costa, R., Sousa, M., & Cabo, T. (2017). Multi-Criteria Classification for Prioritization of Preventive Maintenance Tasks to Support Maintenance Scheduling. Proceedings of the 2017 IEEE IEEM, 978-1-5386-0948-4/17

Men, L., Mu, C., Hong, X., Chen, R., Luan, X., & Ma, T. (2018). Integrated Optimization Model on Maintenance Time Window and Train Timetabling. Proceedings of the 3rd International Conference on Electrical and Information Technologies for Rail Transportation (EITRT) 2017. Transportation. doi: 10.1007/978-981-10-7989-4_87

Montoya-Torres, J., Gutierrez-Franco, E, & Pirachican-Mayorga, C. (2010). Project scheduling with limited resources using a genetic algorithm. International Journal of Project Management, 619-628. doi:10.1016/j.ijproman.2009.10.003

Moubray, J. (1997). Reliability-centred Maintenance. Oxford: Butterworth-Heinemann.

Peng, F., Kang, S., Li, X., Ouyang, Y., Somani, K., & Acharya, D. (2011). A Heuristic approach to the railroad track maintenance scheduling problem. Computer-Aided Civil and Infrastructure Engineering, vol. 26, 129-145.

Raknes, N. T., Ødeskaug, K. Stålhane, & M. Hvattum, L. M. (2017). Scheduling of Maintenance Tasks and Routing of a Joint Vessel Fleet for Multiple Offshore Wind Farms. Journal of Marine Science and Engineering, 11-20. doi: 10.3390/jmse5010011

Rashidnejad, M., Ebrahimnejad, S., & Safari, J. (2018). A bi-objective model of preventive maintenance planning in distributed systems considering vehicle routing problem. Computers & Industrial Engineering. doi:10.1016/j.cie.2018.05.001

Sarker, R., Omar, M., Hasan, S. M. K., & Essam, D. (2013). Hybrid Evolutionary Algorithm for job scheduling under machine maintenance. Applied Soft Computing, vol. 13, 1440-1447.

Tan, J., & Kramer, M. (1997). A general framework for preventive maintenance optimization in chemical process operations. Computers chem. Engng., Vol. 21, 1451-1469.

Valdivieso-Sarabia, R.J., Marín-Alonso, O., Guerrero-Gómez, F. G., Ferrández-Pastor, F. J., Mora-Pascual, J., & García-Chamizo, J. M. (2017). Scheduler for automatic management of maintenance jobs in large-size systems: a case study applied to smart city. Ubiquitous Computing and Ambient Intelligence. 11th International Conference, UCAmI 2017. doi: 10.1007/978-3-319-67585-5_6

Weibull, W. (1951). A statistical distribution function of wide applicability. Journal of applied Mechanics – Transactions of the ASME, vol. 18, 293-297.