UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA INDUSTRIAL

ALGORITMO PARA LA SINCRONIZACIÓN DEL TRANSPORTE PRINCIPAL EN UNA OPERACIÓN MINERA DE COBRE

TESIS PARA OPTAR AL GRADO DE
MAGÍSTER EN GESTIÓN DE OPERACIONES
MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL INDUSTRIAL

MAURICIO IGNACIO ZAVALLA GIMÉNEZ

PROFESOR GUÍA:
RAFAEL EPSTEIN NUMHAUSER
PROFESOR CO-GUÍA:
RODOLFO URRUTIA URIBE

MIEMBROS DE LA COMISIÓN:
JOSÉ CORREA HAEUSSLER

SANTIAGO DE CHILE
2019

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE MAGÍSTER EN GESTIÓN DE OPERACIONES
POR: MAURICIO IGNACIO ZAVALLA GIMÉNEZ
FECHA: 2019
PROF. GUÍA: RAFAEL EPSTEIN NUMHAUSER PROF. CO-GUÍA: RODOLFO URRUTIA URIBE

ALGORITMO PARA LA SINCRONIZACIÓN DEL TRANSPORTE PRINCIPAL EN UNA OPERACIÓN MINERA DE COBRE

Resumen

- Esta tesis describe un algoritmo desarrollado para sincronizar el desplazamiento de los trenes de la etapa de transporte principal en una operación minera de cobre, los que llevan mineral desde el interior de la mina hasta zonas de descarga sobre la superficie.
- La compañía es líder en el mercado del cobre, con ingresos anuales de 11.500 millones de dólares y además cuenta con la mina subterránea más grande del mundo, donde se producen 500 mil toneladas de cobre fino al año. En un turno normal un tren realiza aprox. 4 viajes de 1.500 toneladas. En total, por turno se mueven unas 50 mil toneladas, lo que equivale a 450 toneladas de cobre fino y a un ingreso de 3 millones de dólares.
- El principal desafío operacional de la mina se debe a que la red férrea posee un número bajo de rutas entre el interior de la mina y la zona de descarga, así que con frecuencia se obstruyen los 8 trenes que normalmente operan, haciendo compleja la tarea de coordinar su recorrido. El segmento de vía que más dificulta la operación es un túnel de 7 km. de vía única, tramo que toma 10 minutos atravesar y donde solo pueden circular trenes en un mismo sentido, de modo que si dos trenes lo quieren cruzar en dirección opuesta a la vez, uno de ellos debe esperar al menos 10 minutos, lo que genera congestión.
- El problema se divide en dos: establecer a cuál pique tiene que dirigirse cada tren en cada vuelta del turno y determinar cuál es el recorrido óptimo de los trenes en respuesta a esa asignación. Esta tesis se hace cargo solo del segundo, pues es el más complejo de resolver y el que más valor entrega debido a la serie de potenciales interferencias que se generan entre los trenes, donde se debe tomar una decisión en cada una de ellas.
- El algoritmo se basa en el método de árboles de decisión apoyado por una poda de ramas inspirada en Branch & Bound. Además, para aumentar el rendimiento se aplica una paralelización del problema en una cantidad n de threads que se ejecutan a la vez.
- El algoritmo analiza cerca de 2 millones de combinaciones por hora para sincronizar las rutas de los trenes, de manera que se obtengan soluciones donde se complete un mayor número de viajes dentro de cada turno en relación a lo que se logra en la realidad. Las combinaciones son construidas mediante un árbol binario, el que es recorrido de forma inteligente gracias a una poda que ayuda a identificar tempranamente las ramas que generan soluciones de alta y de baja calidad, eliminando tempranamente estas últimas.
- Se aplicó el algoritmo en dos instancias para evaluarlo. Primero se ejecutó usando los datos de un turno pasado, donde se logró una sincronización en la que se realizó dos vueltas más que en la realidad y luego se construyó un caso para comprobar si es factible completar el máximo teórico de 40 vueltas en un turno de 8 horas, lo que tuvo resultados positivos. En ambos casos se llegó en pocos minutos a muy buenas soluciones, cumpliendo con los objetivos por los cuáles el algoritmo fue desarrollado.

Agradecimientos

Quiero agradecer a todas aquellas personas que me brindaron su apoyo a lo largo de este período de formación como ingeniero.

En primer lugar, agradezco a mis padres Norka y Luis, a mis hermanos Andrés y Alexandra y a mis abuelos Elena y Jaime, quienes han sido mi inspiración durante toda mi vida y me han dado la fuerza para que pueda cumplir mis metas. También agradezco a mis tíos Erika y Wladimir, a mis primos Jasper y Thomas y a mis abuelos Julia y Carlos, por haber estado siempre preocupados por mi desarrollo personal.

Agradezco a mis amigos Hugo, Fernanda, Juan Pablo, Nacho y Yonathan, con quienes he compartido grandes e innumerables experiencias y constantemente me dieron ánimo para que siguiera trabajando en mis objetivos.

También agradezco a Vilma, Analía y Vicente quienes han sido mi segunda familia desde que llegué a Santiago y que me han acogido en todo instante con mucho cariño.

Además, quiero agradecer a todos mis profesores y compañeros de universidad que contribuyeron significativamente en mis estudios, donde destaco el valioso aporte otorgado por Rodolfo y Francisco en la elaboración de esta tesis.

Finalmente, agradezco a Catalina que ha sido una persona muy importante desde el inicio de mi carrera, la que me ha acompañado, ha confiado siempre en mi y ha estado apoyándome incondicionalmente en los buenos y malos momentos.

Tabla de Contenido

Índice de Ilustraciones

Capítulo 1

Introducción

1.1. Antecedentes de la Empresa

La empresa chilena a la cuál pertenece la división minera estudiada en esta tesis, centra sus labores en la producción de cátodos y concentrado de cobre, obteniendo de manera secundaria otros minerales como molibdeno, oro, plata y ácido sulfúrico, aunque en volúmenes bastante menores. En el año 2016 contaba con 18.605 trabajadores propios y recibió ingresos por 11.500 millones de dólares, siendo una de las compañías líderes en el mercado del cobre a nivel mundial.

La mina en la que se desarrolló este trabajo corresponde a uno de los yacimientos de cobre subterráneo más grandes del planeta y cuenta con 3.000 kilómetros de galerías subterráneas en las que trabajan 4.500 personas. Durante el 2016 se produjo casi medio millón de toneladas de cobre fino.

1.2. Descripción del Proceso de Transporte de Mineral

La mina se sitúa al interior de un cerro en el que se traza una división horizontal que actualmente consta de 5 niveles, desde el nivel superior 4 hasta el nivel inferior 8. Entre los niveles 4 y 7 se distribuyen los diferentes sectores mineros y puntos de extracción de rocas, mientras que en el nivel 8 opera el transporte principal, cuya función es llevar el mineral mediante trenes desde el interior de la mina hacia la zona de descarga, ubicada sobre la superficie a unos 10 kilómetros al costado del cerro, donde además se encuentran las plantas de procesamiento.

El 88 % del mineral extraído en la mina es acarreado al sector de descarga a través del transporte principal, mientras que el 12 % restante se traslada directamente a otra zona que está situada en la parte alta del cerro.

Para entender los procesos por los cuáles se mueve el mineral, se explicarán las opera-

ciones efectuadas en el interior de la mina, en el transporte principal y en las plantas de procesamiento. Cabe destacar que este trabajo se enfoca en el transporte principal, por lo tanto, dicha etapa será abordada en mayor profundidad que las otras y no se tomará en consideración ese 12 % de mineral que es llevado hasta la zona ubicada en la parte superior del cerro.

1.2.1. Interior Mina

La mina se divide en grandes sectores productivos, cada uno con características particulares, los que se distribuyen entre los niveles 4 y 7 del cerro.

El movimiento de mineral comienza cuando las palas LHD (por las siglas en inglés para "carga, transporte y descarga") cargan rocas desde los puntos de extracción que tienen asignados dentro del sector minero. Cada sector minero posee su propia flota de estas máquinas, las que pueden tener una capacidad de 7, 13 o 17 toneladas.

Las palas se mueven desde los puntos de extracción a través de calles hasta descargar el contenido en piques de traspaso, que son estructuras cilíndricas orientadas verticalmente cuya función es dejar fluir el mineral hacia abajo para hacerlo llegar a las siguientes etapas de transporte.

El mineral descargado en los piques de traspaso es recibido por la etapa de "transporte intermedio", la que puede ser efectuada por camiones, correas transportadoras o trenes, dependiendo del sector. El transporte intermedio mueve el mineral desde los piques de traspaso hasta los piques principales.

Al igual que los piques de traspaso, los piques principales son estructuras cilíndricas verticales, aunque de mucho mayor tamaño. En la actualidad existen alrededor de 20 piques principales y tienen como función hacer fluir el mineral desde niveles superiores hasta el nivel 8, estableciendo el punto de conexión entre los sectores productivos y el transporte principal. Todo sector minero tiene entre uno y tres piques asociados.

La mayoría de los piques principales reciben el mineral en las mismas condiciones en las que fue extraído por las palas LHD, sin embargo, hay piques que trabajan con mineral previamente chancado. Lo anterior se debe a que algunos sectores productivos poseen chancadores primarios al interior de la mina, cuya labor es reducir el tamaño de las rocas, de modo que entregan a sus piques rocas de menor dimensión que los otros sectores. El mineral que es chancado se conoce como mineral fino, mientras que el no chancado se denomina mineral grueso.

La parte inferior del pique cuenta con un "buzón", una especie de tapón para evitar que el mineral se caiga y que solo es abierto cuando llega un tren para recibir una entrega. En el momento en que un pique principal posee las suficientes toneladas para cargar un tren (alrededor de 1.500 toneladas), se dice que el pique tiene una "entrega" disponible para ser acarreada.

Lo máximo de mineral que puede contener un pique a la vez varía entre 1 y 4 entregas. Los piques con mayor capacidad cumplen un importante rol cuando los trenes no están cargando entregas debido a largas detenciones en las operaciones del transporte principal o de las plantas de procesamiento, dado que pueden seguir recibiendo mineral por algunas horas antes de llenarse, permitiendo que los sectores productivos continúen trabajando con normalidad durante ese período.

En la Figura 1.1 se muestra un esquema con los procesos por los que pasa el mineral en el interior de la mina hasta llegar al transporte principal. Notar que el transporte intermedio varía dependiendo del sector productivo, pudiendo efectuarse por correas, camiones o trenes. Además, la imagen ilustra el caso de un pique principal que recibe mineral fino desde un chancador primario (sector A en la figura).

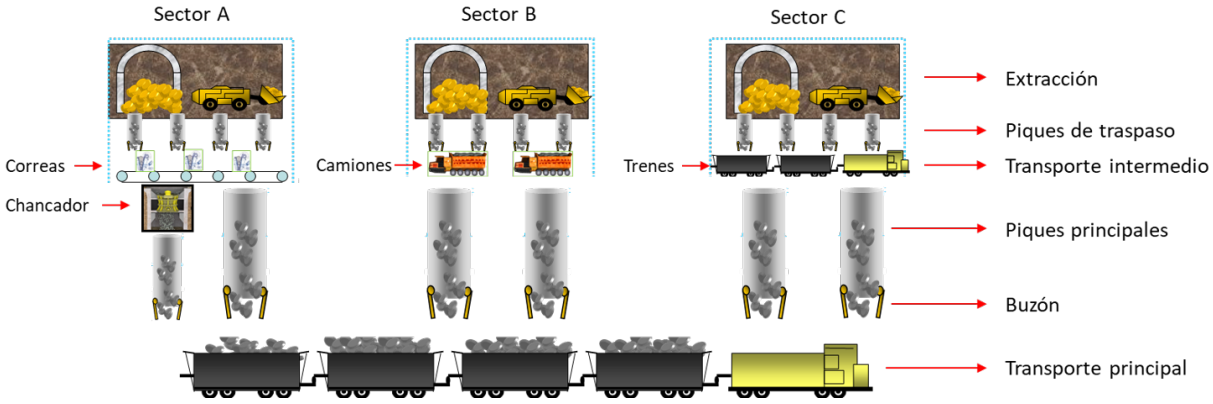

Figura 1.1: Etapas por las que pasa el mineral en el interior de la mina.

1.2.2. Transporte Principal

El nivel 8 de la mina, correspondiente al transporte principal, consta de una serie de líneas férreas donde operan regularmente 8 trenes, pudiendo ser menos en situaciones especiales como en caso de mantenimientos. Toda la red es de una sola vía, es decir, los trenes transitan en ambos sentidos por las mismas líneas ferroviarias, a diferencia de lo que ocurre en redes como la del Metro donde todas las vías son dobles y pueden circular simultáneamente dos trenes por el mismo punto y en sentido contrario.

Conceptualmente la red férrea puede ser dividida en tres zonas: interior mina, túnel principal y zona de descarga (ubicada sobre la superficie), tal como se observa en la Figura 1.2.

Las líneas férreas situadas al interior de la mina se ubican en el nivel 8 del cerro, por lo tanto, es en esta zona donde se encuentra la parte inferior de todos los piques principales. Los segmentos de línea de este sector son conocidos como "cruzados" y permiten a los trenes acceder a los puntos donde se distribuyen los piques. Cuando un tren ingresa a la zona de carga de un pique principal que cuenta con una o más entregas disponibles, se abre el buzón del pique y fluye el mineral sobre los carros del tren hasta llenarlos, proceso que toma 18

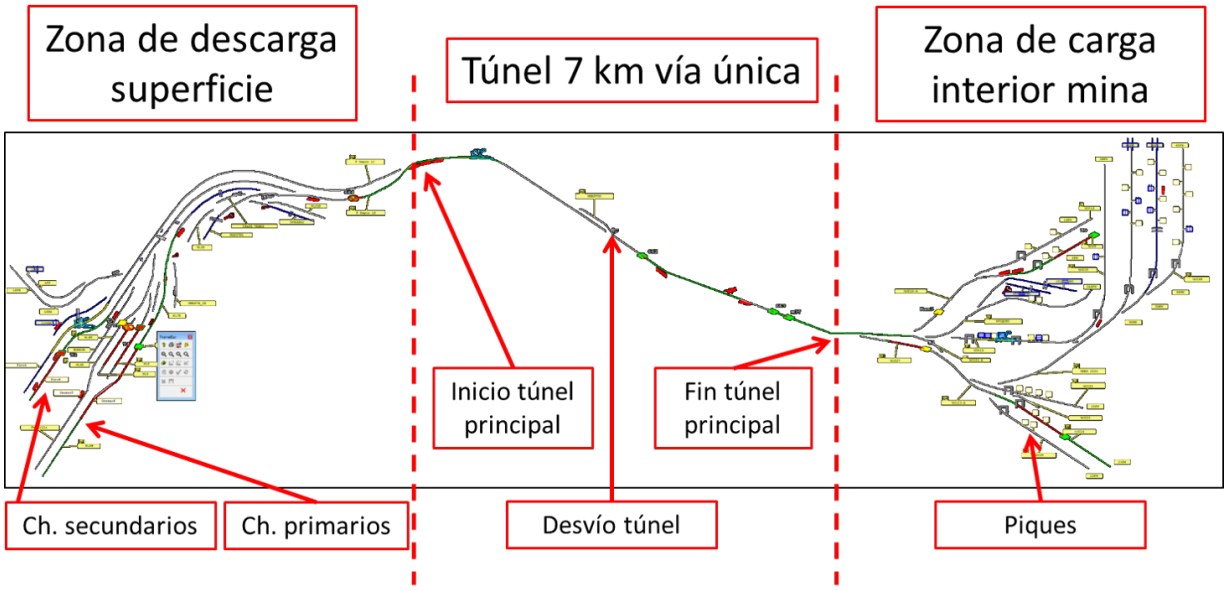

Figura 1.2: Layout de la red ferroviaria.

minutos en promedio. Como algunos piques entregan mineral grueso y otros mineral fino, se dispone de trenes especiales para cada caso. De los 8 trenes, 6 transportan mineral grueso, mientras que los 2 restantes son cargados con mineral fino.

Terminado el proceso de carga, los trenes se dirigen hacia su zona de descarga ubicada fuera del cerro. El recorrido se caracteriza por un túnel de 7 kilómetros de longitud que realiza la transición entre el interior del cerro y la superficie. El punto extremo del túnel orientado hacia la zona de descarga se considera el inicio del túnel, mientras que el punto extremo que da hacia la mina es el final del túnel principal.

Como la red es de vía única, el túnel impone una gran dificultad operacional, pues si un tren está utilizando el túnel para dirigirse hacia el interior de la mina, todos los trenes que desean cruzar el túnel en dirección contraria deben esperar hasta que el tren recorra los 7 kilómetros y lo desocupe. Lo mismo ocurre cuando un tren utiliza el túnel dirigiéndose hacia la zona de descarga y otros intentan moverse hacia la mina a través de ese tramo. La única alternativa disponible para que dos trenes utilicen el túnel en sentido contrario al mismo tiempo es un desvío en la mitad del túnel, el que tiene capacidad solamente para un tren.

Cuando los trenes cargados de mineral atraviesan por completo el túnel principal, llegan al sector de descarga. En dicha zona existen cuatro líneas de descarga, dos para trenes de mineral fino y dos para trenes de mineral grueso. En esas líneas los trenes se detienen a vaciar la carga sobre buzones[1], que son unos contenedores usados para acumular momentáneamente las rocas. El proceso de descarga toma en promedio 12 minutos para los trenes de mineral fino y 15 minutos para los de grueso.

El mineral depositado en los buzones pasa directamente a un chancador, cuya función es reducir la dimensión de las rocas. Cada línea de descarga tiene un chancador. Las líneas

[1]No confundir con el buzón que poseen los piques principales, el que consiste en una especie de tapón cuyo objetivo es controlar la salida del mineral desde los piques.

llamadas L0 y L1 poseen "chancadores secundarios y terciarios [2]", lo que significa que chancan el mineral por segunda y tercera vez. En dichas líneas ingresan exclusivamente los trenes de mineral fino, ya que ese mineral fue chancado anteriormente dentro de la mina. Por otra parte, la línea L5 y L6 cuentan con "chancadores primarios", así que ahí descargan los trenes que transportan mineral grueso para que pueda ser chancado por primera vez.

Las vías férreas poseen algún grado de pendiente en ciertos tramos. Ese hecho sumado a que el peso de un tren cargado es 1.500 toneladas superior que el de un tren descargado, hace que la velocidad de traslado de los trenes por los tramos sea diferente para cada sentido. Por lo tanto, el tiempo para llegar desde una línea de descarga a un pique, no es el mismo que al hacer el recorrido inverso por la misma ruta.

Es importante enfatizar que existe una clara independencia entre las misiones que cumplen los trenes de mineral fino y los de mineral grueso. Los 2 trenes de mineral fino pueden realizar el proceso de carga solo en alguno de los 3 piques principales que reciben mineral previamente chancado para luego dirigirse a descargar únicamente en las líneas L0 o L1, las que cuentan con chancadores secundarios. Respecto a los 6 trenes de mineral grueso, estos cargan en cualquiera de los aproximadamente 17 piques que trabajan con mineral sin chancar y después descargan en las líneas L5 o L6, correspondientes a las líneas con chancadores primarios.

La nomenclatura que se utiliza para nombrar a los trenes es según una numeración. Los trenes de mineral fino se denotan por T2 y T3, mientras que los trenes de mineral grueso se denominan T4, T5, T6, T7, T8 y T9. En algunos casos particulares se utiliza un tercer tren de mineral fino llamado T1.

1.2.3. Plantas de Procesamiento

Las plantas de procesamiento se ubican al costado del cerro, sobre la superficie. Esta fase comienza una vez que el mineral se chanca, ya sea por un chancador primario o secundario. Posteriormente, el mineral se lleva hasta unas moliendas para reducir aún más las dimensiones de las rocas. La etapa de molienda se divide en dos grandes líneas: la línea de las moliendas SAG y la línea de la molienda convencional.

A las moliendas SAG, conocidas como SAG 1 y SAG 2, llega directamente la mayoría del mineral que se procesa en los chancadores primarios del sector de descarga. Las rocas son trasladadas mediante correas transportadoras desde los chancadores hasta las pilas SAG, que son básicamente cerros de mineral acumulado que se encuentra a la espera de pasar por la molienda SAG respectiva. El mineral llega desde los chancadores primarios con una dimensión de entre 7 y 10 pulgadas aproximadamente y sale de las moliendas SAG con una medida de 150 micrónes, donde un micrón corresponde a 10^{-6} metros. La SAG 1 tiene una capacidad de trabajo de 1.440 toneladas por hora y la SAG 2 una capacidad de 2.450 toneladas por hora.

Por otra parte, a la molienda convencional llega directamente todo el mineral proveniente de los chancadores secundarios, a través de correas transportadoras. Además, recibe un porcentaje variable de mineral desde los chancadores primarios gracias a una correa denominada

[2]Desde aquí en adelante se les llamará simplemente chancadores secundarios.

T3. Esta molienda necesita que el mineral haya pasado por el chancado secundario, pues solo puede procesar rocas de menor tamaño, así que se utiliza un chancador secundario de la línea llamada L2 para chancar todo lo que proviene desde la correa T3. La molienda convencional tiene una capacidad de 3.000 toneladas por hora.

Finalmente, el mineral atraviesa distintos procesos físicos y químicos, tales como flotación y filtración, para obtener el concentrado de cobre que contiene alrededor de un 30 % del metal. Estas fases no serán detalladas debido a que escapan del ámbito interés de este trabajo.

En la Figura 1.3 se muestra un esquema que resume las etapas por las que pasa el mineral, desde la extracción hasta la molienda.

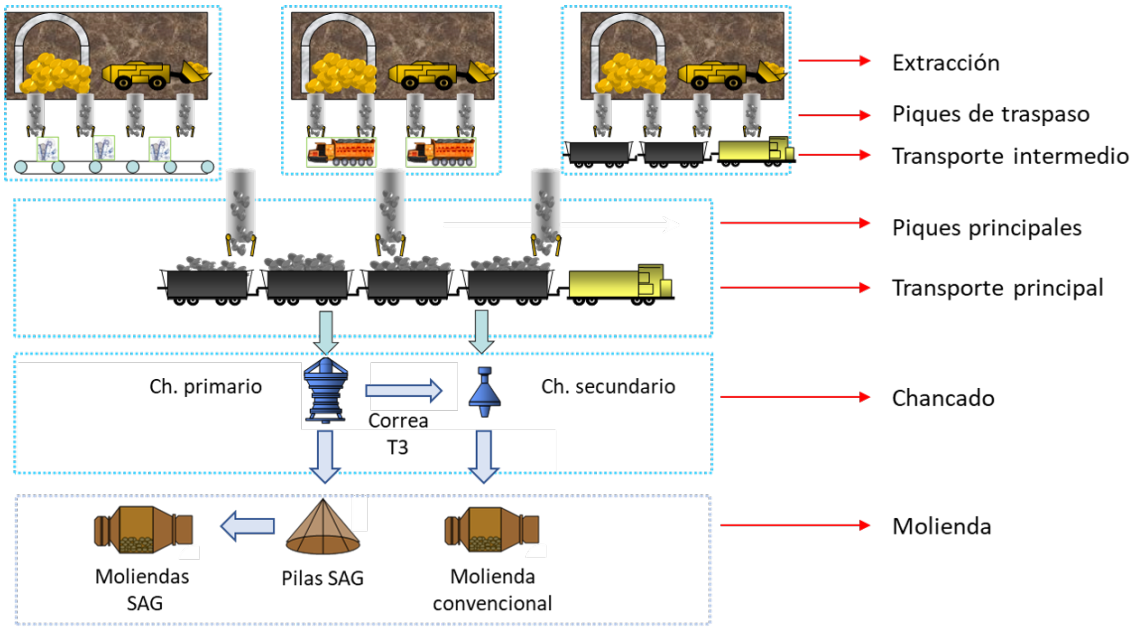

Figura 1.3: Etapas por las que pasa el mineral, desde la extracción hasta la molienda.

1.3. Motivación

Diariamente en la mina se procesan alrededor de 145.000 toneladas de mineral, donde 125.000 llegan a la zona de descarga principal ubicada sobre la superficie y las 20.000 restantes tienen como destino un sector productivo más antiguo que se sitúa sobre el cerro. Dada la magnitud de la mina y el volumen de mineral con el que se trabaja, las operaciones tienen una alta complejidad, lo que hace necesario buscar continuamente mejoras en el desempeño de los procesos para hacerlos más eficientes y así aumentar la productividad año tras año. Uno de los grandes desafíos actuales se encuentra en la etapa de transporte principal, pues la red férrea es de una sola vía y los trenes se interfieren frecuentemente al usarla.

La restricción que impone la vía única conlleva un gran problema a la hora de coordinar el movimiento de los 8 trenes que generalmente están operando de manera simultánea, haciendo que esa labor resulte muy compleja. Particularmente las dificultades se concentran en el túnel

de 7 kilómetros, dado que son alrededor de 10 minutos los que pasan desde que un tren ingresa al túnel hasta que lo desocupa, lo que con mucha frecuencia tiene como consecuencia 10 minutos de espera de otro tren que intenta utilizar el túnel en sentido contrario. También se generan bastantes obstrucciones entre los trenes que entran y salen de las zonas de carga de los piques principales, pues la red férrea al interior de la mina es muy limitada.

En la mina se trabaja las 24 horas del día divididas en 3 turnos. Como los cambios de turno implican también un recambio del personal que opera los trenes, es necesario que todos los trenes finalicen el turno en el sector de descarga, lugar donde se realiza en relevo de operarios. Eso se consigue gracias a una normativa que impide que los trenes ingresen al túnel en dirección hacia la mina desde una hora antes del termino del turno, para asegurar que todos los trenes estén en la zona de descarga cuando se cumpla el horario. De esa forma, debido a los cambios de turno, la sincronización de los trenes se considera como un problema discreto entre turnos.

En cada turno la coordinación de los trenes es misión de un operario denominado "despachador", quien de forma permanente ordena el desplazamiento de los trenes, planifica hacia cuál pique principal se debe dirigir cada tren y determina por cuál ruta se tiene que mover para llegar a destino. Además, el despachador tiene la misión de decidir cuál será el tren que debe avanzar y cuál será el que debe esperar cada vez que se producen interferencias entre trenes que intentan avanzar en dirección opuesta por un mismo tramo de la vía ferroviaria.

Si bien los despachadores cuentan con mucha experiencia en la coordinación de los trenes, carecen del apoyo de herramientas que les ayuden a evaluar diferentes decisiones, en un problema que tiene millones de combinaciones posibles solo en un turno laboral. Cada vez que un despachador se enfrenta a una obstrucción de trenes, él determina a cuál tren darle prioridad para avanzar, lo que impacta en el resultado final del turno completo. Sin embargo, no tienen manera de evaluar el posible resultado en caso de haber tomado la decisión contraria, entonces realizan su trabajo sin saber con certeza si están priorizando los trenes de forma correcta o errónea.

En ese contexto, este proyecto nace con la intención de crear un algoritmo que permita encontrar en poco tiempo soluciones de alto nivel para la sincronización de los trenes en su circulación a través de la red férrea. La idea es que el algoritmo se conecte a los sistemas de información de la mina y reciba como input ciertas características del turno a evaluar, para que luego pruebe rápidamente distintas combinaciones de la coordinación en el desplazamiento de los trenes, siempre respetando las restricciones operacionales de la mina. Después de algunos minutos en ejecución, el algoritmo debería entregar una sincronización que permita realizar dentro del turno una cantidad de vueltas mayor de lo que se podría lograr coordinando los trenes de forma manual, o que permita hacer el mismo número de vueltas pero completadas en un mejor tiempo, de modo que la mayoría de los trenes alcancen a descargar durante el turno en curso para que así no comiencen el siguiente turno cargados en la zona de descarga. Justamente ese es el valor de la herramienta, pues evaluará solo en pocos minutos un problema con millones de soluciones, entregando un gran apoyo en la toma de decisiones al personal de la mina.

Hay que tener en consideración que en algunos turnos la mina tiene una tasa de producción muy lenta, por problemas técnicos u otras razones, transformándose en el cuello de botella

de todo el proceso. En aquellos casos existe una baja disponibilidad de entregas en piques principales, así que el problema de coordinación de trenes resulta más fácil de resolver para el despachador de turno y aunque el algoritmo determine una muy buena sincronización, no podrá realizar mayor número de vueltas porque no habrá más entregas para acarrear. Por otra parte, en los turnos donde existe una alta tasa de producción en la mina, el transporte principal pasa a ser el cuello de botella y la coordinación de los trenes se vuelve un problema complejo de resolver, justificando el uso del algoritmo.

El algoritmo tiene el potencial para ser utilizado de tres formas diferentes: aprendizaje de turnos pasados, coordinación del turno actual y pronóstico de turnos futuros. En primer lugar, la herramienta puede ser usada para el aprendizaje de los despachadores, lo que se consigue realizando la sincronización de trenes en turnos pasados de los cuáles se tiene información completa, para luego comparar las decisiones y resultados reales con lo entregado por el algoritmo. Esto permite que los despachadores identifiquen errores, buenas y malas prácticas, oportunidades de mejora y patrones en la coordinación de los trenes que tengan como resultado una mayor cantidad de vueltas por turno. Ese aprendizaje ayuda a que en el futuro los despachadores tomen mejores decisiones.

El segundo uso potencial es en todo turno que esté pronto a comenzar, pues el algoritmo puede realizar una sugerencia a los despachadores sobre la sincronización ideal que deben seguir los trenes. En este caso se tiene información parcial, dado que se sabe las entregas disponibles en piques y condiciones de la red al inicio del turno, pero para el resto del turno se debe estimar la tasa de producción de la mina y el desempeño de las plantas de chancado de la zona de descarga.

Finalmente, el algoritmo puede ser utilizado para turnos hipotéticos o turnos futuros, con una estimación completa de las entregas y condiciones de las plantas de chancado, donde no se tenga certeza absoluta de dichos parámetros a través de todo el turno. Este tipo de casos le otorgan al personal de la mina una solución que pronostica cuántas vueltas se podrían realizar en un turno con iguales o similares características que las estimadas o simuladas, lo que puede ser aplicado para evaluar cambios de políticas operacionales, como por ejemplo, modificaciones en los horarios de inicio y final de cada turno o en las planificaciones de los mantenimientos.

Lo que se espera es que el algoritmo provoque un aumento en el flujo de mineral desde la mina hacia las plantas chancadoras y, por ende, un incremento en la tasa de producción de la división minera en todos los turnos donde el transporte principal resulta ser cuello de botella. La ganancia en ingresos que se obtiene al realizar una vuelta adicional en un turno, bordea los US$70.000, una cifra muy relevante para la empresa.

A pesar de que el algoritmo puede probar rápidamente muchas más combinaciones que una persona, a priori no necesariamente los resultados serán mejores que lo conseguido por los despachadores, pues podría ser que la coordinación que ellos hacen no es superable por un algoritmo de estas características, lo que haría innecesaria la herramienta. Sin embargo, se identificaron oportunidades de mejora en algunas decisiones que son usadas por los despachadores. Estas decisiones tienen relación con reglas simples y rígidas que se aplican en base a la experiencia, pero que a veces no son óptimas, dado que no tienen el apoyo de una herramienta que permita advertirles la existencia de mejores alternativas.

1.3.1. Ejemplo Oportunidades de Mejora

Para clarificar esas oportunidades de mejora que existen al coordinar la ruta de los trenes, se mostrará un caso donde se entorpecen dos trenes en el túnel y se debe tomar la decisión de cuál tren dejar avanzar y cuál hacer esperar.

En la operación diaria, frecuentemente se le da prioridad al tren que se dirige desde la mina hacia la zona de descarga, en desmedro del que se desplaza hacia la mina. En algunos de esos casos la decisión opuesta puede tener como consecuencia la realización de al menos una vuelta más durante el turno, como en el ejemplo descrito a continuación.

Por simplicidad, para este ejemplo se trabajará en una red con menos tramos que en la realidad (Figura 1.4). La red comienza con dos puntos de descarga, el primero asociado a un chancador primario y el otro asociado a un chancador secundario. En cada caso existe una vía que conecta la zona de descarga con el inicio del túnel principal. El otro extremo del túnel principal se enlaza con tres vías, las que se dirigen a tres piques diferentes.

Se establece que el tiempo de descarga de los trenes es de 12 minutos y que el tiempo de carga en los piques es de 13 minutos. En las vías se indica el tiempo que se demora un tren en desplazarse por cada segmento, el cual se asumirá que es igual para ambos sentidos de movimiento, a diferencia de lo que ocurre en la red real donde el tiempo varía con la dirección de desplazamiento.

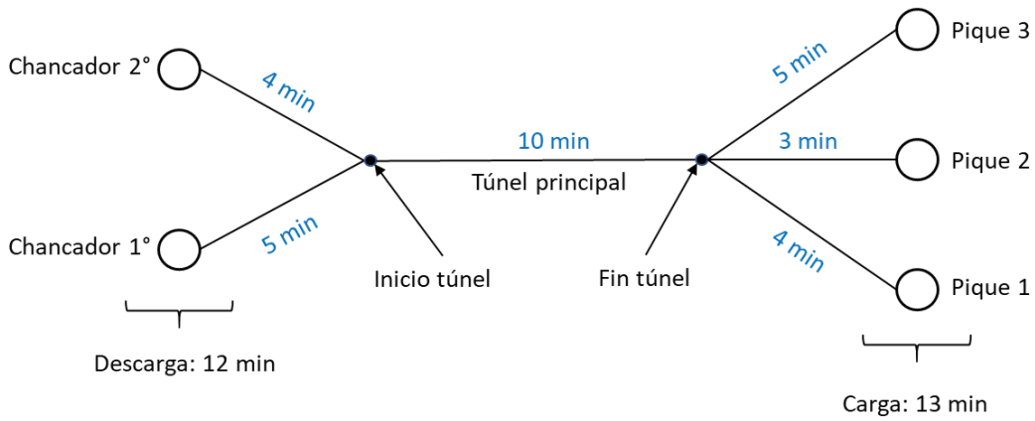

Figura 1.4: Red ferroviaria ficticia para ejemplificar oportunidades de mejora en la sincronización de trenes.

Una de las principales restricciones que deben respetar los trenes es la hora límite que tienen para entrar al túnel principal en dirección hacia la mina, producto de que en el cambio de turno el relevo de los operarios de trenes se hace en el sector de descarga, de manera que ningún tren puede quedar al interior del túnel principal o de la mina al finalizar el turno. En el ejemplo se mostrará un turno B (comienza a las 16:00 y termina a las 00:00) que tiene como hora límite las 23:00, es decir, después de esa hora ningún tren puede cruzar por el inicio del túnel principal en dirección hacia la mina y solo pueden circular por ahí los trenes que se dirigen desde la mina hacia la zona de descarga. La restricción se mantiene hasta la hora en que comience el siguiente turno.

La situación evaluada comienza a las 21:50. Inicialmente un tren de mineral fino, que para este ejemplo se le llamará T1, va saliendo de la línea de descarga y se dirige hasta el pique 1, mientras que un tren de mineral grueso, denominado T2 en esta explicación, está saliendo del pique 3 y tiene como destino el chancador primario. Si ambos trenes siguen sus rutas naturales, chocarían de frente al interior del túnel a las 22:00 aproximadamente (Figura 1.5).

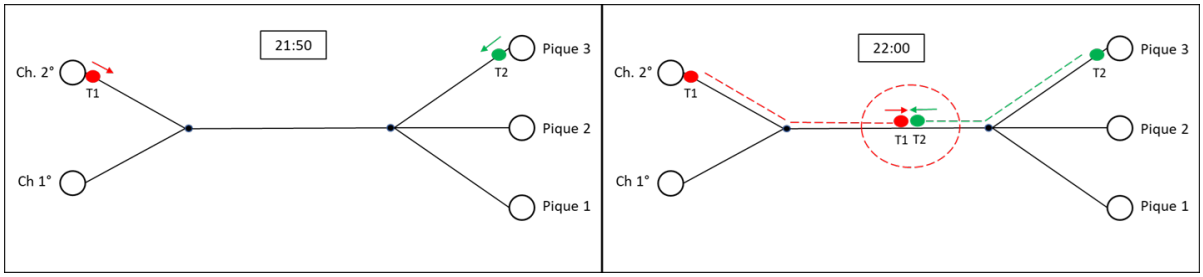

Figura 1.5: Caso que ejemplifica oportunidades de mejora en la sincronización de trenes. Se muestra la situación inicial de los trenes y el choque que se ocasionaría si ambos avanzan por el túnel.

Para que los trenes no choquen solo uno de ellos puede ingresar al túnel principal, mientras que el segundo tiene que esperar justo afuera hasta el instante en que sea desocupado. Existen dos opciones: dejar avanzar al T1 o al T2, donde ambas alternativas serán analizadas para comparar sus resultados.

Opción 1: se da preferencia al tren que va hacia la zona de descarga. Al dejar avanzar al T2 por el túnel y hacer esperar 11 minutos al T1 junto al inicio del túnel principal, se tendrán los siguientes hitos en el desplazamiento de los trenes por las vías (Figura 1.6):

- **22:05:** el T2 sale del túnel hacia el sector de descarga, así que el T1 puede hacer ingreso inmediato al túnel para desplazarse hasta la mina. El T2 arriba a la línea de descarga del chancador primario a las 22:10.
- **22:19**: el T1 llega al pique 1 y comienza el proceso de carga, el que finaliza a las 22:32. En cuanto al T2, culmina la descarga de mineral a las 22:22 y luego se dirige a buscar una entrega en el pique 2.

  El problema es que ambos trenes se van a interferir dentro del túnel principal, pues el T1 haría su entrada a las 22:36 en dirección hacia la zona de descarga, mientras que el T2 saldría del túnel un minuto más tarde en dirección mina. Con el objetivo de que el T1 alcance a regresar a la mina antes de la hora límite, se le dará prioridad para que tenga la posibilidad de realizar otra vuelta, asignándole una espera de 19 minutos al T2 afuera del inicio del túnel principal.
- **22:46**: en este instante el T1 sale del túnel hacia el sector de descarga, mientras que el T2 completa los 19 minutos de espera y hace ingreso al túnel principal.
- **22:50**: el T1 llega a la línea de descarga de mineral fino y el T2 transita a través del túnel para llegar al pique 2. A las 23:02 finaliza la descarga del T1.
- **23:06**: el T1 llega al inicio del túnel, pero se ha superado la hora límite de subida, por lo tanto, no puede entrar al túnel para buscar otra entrega.

Notar que en el segundo "choque" se decidió dejar avanzar al T1 para que tuviera la oportunidad de completar otra vuelta, hecho que no sucedió. De haber escogido dar prioridad

al T2, el T1 habría esperado 1 minuto, llegando al inicio del túnel en dirección mina a las 23:07 sin lograr dar otra vuelta, al igual que en la opción elegida en el ejemplo. La gran diferencia es que el T2 no tendría los 19 minutos de espera. Pero, como el caso representa el fin de un turno, esos minutos de ventaja del T2 no significan una ganancia en número de vueltas, pues al retornar de la mina el tren debe permanecer en la zona de descarga hasta que se haga el cambio de los operadores. De esa forma, ambas alternativas de decisión en el segundo "choque" tienen el mismo resultado en la cantidad de viajes realizados.

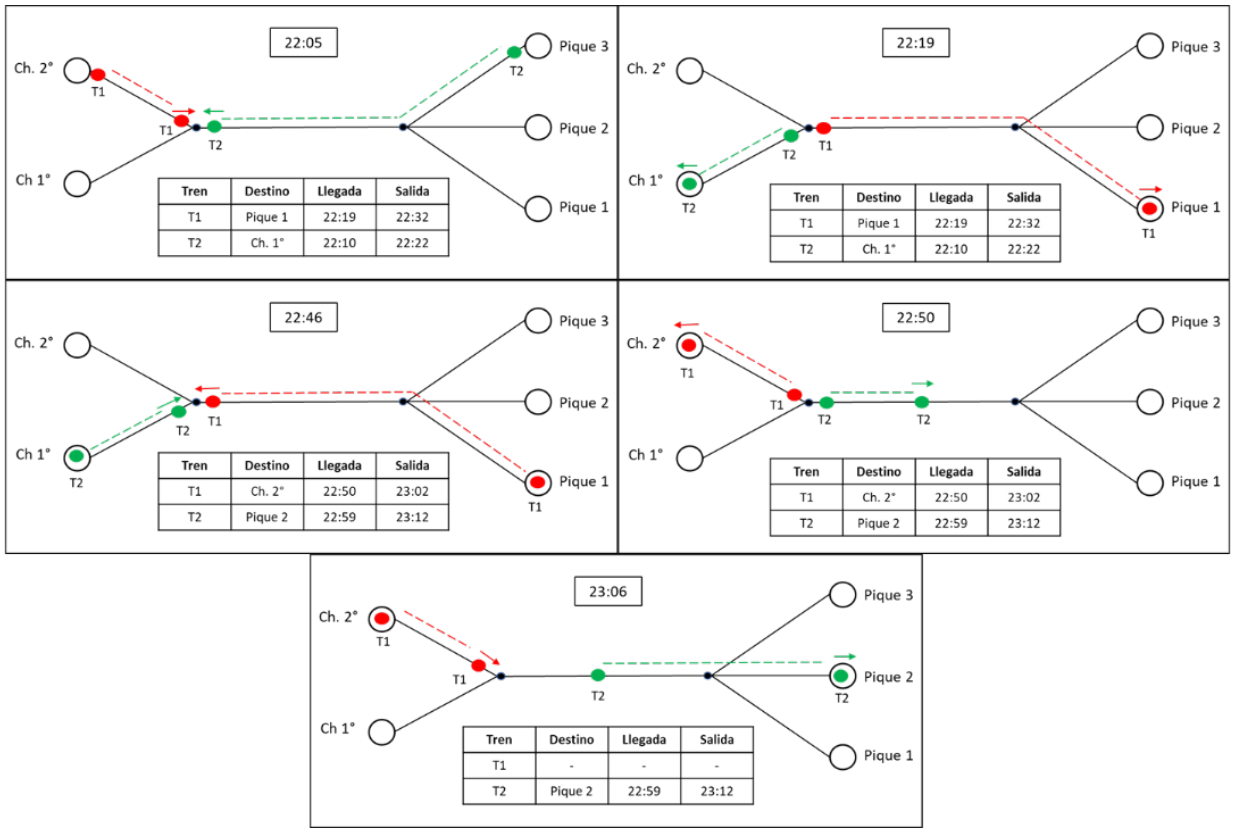

Figura 1.6: Caso que ejemplifica oportunidades de mejora en la sincronización de trenes. La secuencia muestra lo que ocurre si se deja avanzar al T2 en la interferencia que se produce alrededor de las 22:00 con el T1 en el túnel principal.

Opción 2: se da preferencia al tren que va en dirección mina. De esa forma, el tren descargado T1 ingresa al túnel principal, mientras que el tren cargado T2 espera por 9 minutos afuera del túnel. En esa situación se producen los siguientes eventos importantes (Figura 1.7):

- **22:04:** el T1 llega a la salida del túnel, momento en que el T2 hace ingreso para trasladarse hasta el sector de descarga. A las 22:08 el T1 llega al pique 1 y comienza el proceso de carga.
- **22:19:** el T2 llega al chancador primario para descargar, lo que finaliza a las 22:31. Por otra parte, el T1 termina de cargar a las 22:21 y pone en marcha su trayecto hacia la zona de descarga.
- **22:36:** los trenes no se interfieren, ya que el T1 desocupa el túnel a las 22:35 y el T2 entra un minuto más tarde, de modo que ninguno de los dos trenes debe esperar.

- **22:39:** en ese instante el T1 llega a descargar al chancador secundario, donde se mantiene hasta las 22:51. En cuanto al T2, se encuentra en la mitad del túnel en dirección hacia la mina.
- **22:55:** el T1 llega hasta el inicio del túnel con intenciones de dirigirse hacia la mina para buscar una entrega en el pique 1. Como la hora límite es hasta las 23:00, el tren puede ingresar sin problemas al túnel y logra realizar la vuelta.

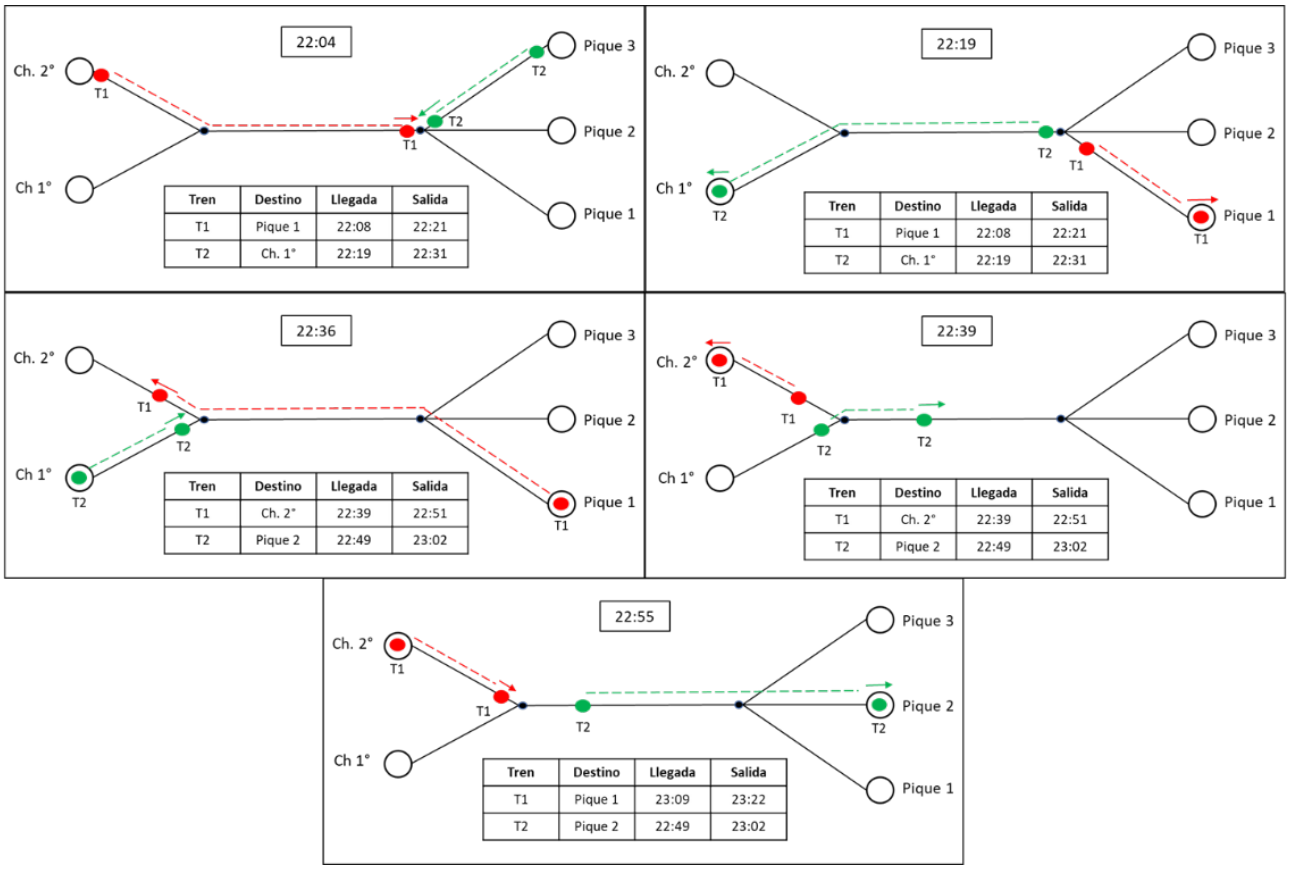

Figura 1.7: Caso que ejemplifica oportunidades de mejora en la sincronización de trenes. La secuencia muestra lo que ocurre si se deja avanzar al T1 en la interferencia que se produce alrededor de las 22:00 con el T2 en el túnel principal.

En el caso descrito se obtuvo resultados distintos al aplicar las dos decisiones opuestas. La alternativa de darle prioridad al T2 que iba hacia la zona de descarga, tal como hace con frecuencia el despachador de trenes, atrasó al T1 e impidió que pudiera realizar otra vuelta, mientras que la opción de dejar avanzar al T1 permitió que el tren lograra dar esa vuelta sin inconvenientes y sin perjudicar las vueltas del T2.

Esto demuestra el tipo de oportunidades de mejora que se presentan a la hora de coordinar los trenes y que deben ser aprovechadas por el algoritmo, donde solo una vuelta más en un turno se traduce en US$70.000 de ingresos adicionales.

1.3.2. Sincronización de Trenes y Asignación de Trenes a Piques

El trabajo que realiza el despachador de trenes se puede separar en dos importantes tareas. En primer lugar, el despachador determina la asignación de los trenes a las entregas disponibles en los piques principales y en segundo lugar, coordina el movimiento de los trenes para cumplir con esa asignación. Si bien puede parecer que la primera tarea es la más compleja, en realidad ocurre todo lo contrario.

Para dejar en claro lo anterior, es preciso analizar de manera independiente los trenes de mineral fino y los de grueso. Recordar que en general se utilizan 2 trenes de mineral fino, los que van a buscar entregas solamente a 3 piques principales. De 10 entregas de mineral fino que se producen, 8 provienen de 2 piques principales que se encuentran en el mismo cruzado de la mina. En consecuencia, asignar los trenes de mineral fino a las entregas resulta bastante sencillo, pues no existen muchas alternativas. Se envían los dos trenes a los dos piques antes mencionados y cuando se genera una entrega de mineral fino en otro sector productivo, se envía uno de los trenes a cargar esa entrega de mineral.

De esa manera, el problema de asignación queda reducido únicamente a los 6 trenes de mineral grueso. En este caso, al haber más trenes y más piques principales, existe una mayor cantidad de combinaciones posibles y el problema no es trivial. Sin embargo, la mayoría de las entregas se generan en los piques de solo dos sectores mineros, así que las alternativas existentes para los trenes no son tantas, lo que significa que el problema posee una complejidad menor.

Por lo tanto, la mayor dificultad a la que se enfrenta el despachador del turno tiene relación con la coordinación de los trenes para responder a la asignación de trenes a piques escogida. Para realizar ese trabajo se deben sincronizar simultáneamente los 8 trenes que se encuentran circulando por la misma red. Inicialmente el despachador tiene que determinar la ruta por la que circulará cada tren para llegar a un pique o a un chancador. Luego, durante todo el recorrido de los trenes, debe prever la gran cantidad de interferencias que se producen debido a las limitaciones de la red ferroviaria, especialmente en el sector interior mina y en el túnel principal. En cada una de las interferencias hay que analizar la posibilidad de generar cambios en las rutas de los trenes de modo que se eviten choques de trenes. Si no hay otras rutas disponibles, es necesario decidir cuál tren dejar pasar y cuál hacer esperar, asignándole un tiempo de espera al tren sin preferencia.

Tomar esa cantidad de decisiones sobre los 8 trenes durante todo el turno sin duda representa una tarea muy difícil, pues según cifras empíricas, en promedio hay del orden de 10^{12} combinaciones diferentes para coordinar los trenes solo en un turno. Por esa razón, el algoritmo no abordará el problema de asignación y solo se hará cargo de la sincronización de los trenes, que es el problema de mayor complejidad para los despachadores.

1.4. Objetivos

1.4.1. Objetivo General

Desarrollar un algoritmo para sincronizar los trenes durante un turno dado en una operación minera de cobre, que le sirva a los despachadores de trenes como herramienta de apoyo para lograr aumentar el flujo de mineral entre la mina y los chancadores ubicados en el sector de descarga.

1.4.2. Objetivos Específicos

1. Crear un algoritmo que permita probar en poco tiempo una gran cantidad de combinaciones para coordinar los trenes dentro de un turno.
2. Definir indicadores para determinar la calidad de la sincronización propuesta por el algoritmo cada vez que se ejecute.
3. Construir un método para recorrer soluciones de manera eficiente, priorizando ciertos grupos de soluciones de alta calidad por sobre otros.
4. Realizar un análisis para evaluar la rapidez y eficiencia del algoritmo, así como la calidad de las soluciones que encuentra.

1.5. Alcances y Limitaciones

Como se explicó previamente, esta tesis tiene como objetivo desarrollar un algoritmo para sincronizar los trenes que transportan mineral en una operación minera de cobre, de modo que aumente el flujo de toneladas acarreadas entre el interior de la mina y la zona de descarga en los turnos en que sea aplicado. La sincronización debe respetar las condiciones y restricciones del turno en análisis, tales como los mantenimientos en las líneas férreas y piques, la cantidad de trenes y chancadores en funcionamiento, las entregas disponibles en los piques y la hora límite de entrada al túnel producto del relevo de operarios en cada cambio de turno.

A continuación, se detallan las limitaciones y simplificaciones que presenta el trabajo:

- El algoritmo será analizado bajo condiciones determinísticas, así que se probará utilizando turnos pasados donde se conocen con exactitud todas las restricciones y entregas de viajes disponibles en los piques principales. En este trabajo no se incorporará la aleatoriedad en las condiciones de la mina.
- Las decisiones que hay que tomar para cada tren a lo largo del turno se pueden dividir en dos:
    1. Decidir a cuál pique principal se va a dirigir cada tren en cada vuelta, considerando que el pique al que el tren es asignado necesita tener al menos una entrega disponible.

2. Determinar la ruta que seguirán los trenes por la línea férrea para llegar a los piques y chancadores. Una vez escogido el recorrido, durante todo el turno se deben evaluar cambios de ruta en caso de haber interferencias entre trenes, para así evitar choques. En caso de no existir caminos alternativos disponibles, hay que decidir a cuál tren otorgar prioridad para utilizar cierto tramo de la red ferroviaria y a cuál asignar minutos de espera.

Dado que el segundo tipo de decisiones tienen una mayor complejidad, en esta tesis se asumirá como fija la asignación de los trenes a las entregas disponibles en los piques principales, mientras que el algoritmo se encargará de coordinar las rutas y el movimiento de los trenes para responder a esa asignación. A pesar de eso, la rapidez del algoritmo dará la posibilidad de correr muchas veces el mismo turno probando distintas asignaciones de trenes a piques, de modo que también se puedan identificar algunas mejoras en ese aspecto.

- La red ferroviaria por la que circulan los trenes será simplificada, lo que tendrá como resultado una red con menor cantidad de rutas distintas. Esto será hecho principalmente en el sector de descarga donde existen varios caminos alternativos para desplazarse entre el túnel y los chancadores (o viceversa) de similar tiempo de traslado, así que con poca frecuencia se producen problemas en la coordinación de los trenes, de manera que modelar en exactitud esa parte de la red no genera un gran aporte para los despachadores. Esta simplificación ayuda a reducir el número de rutas posibles a las que se enfrentan los trenes, disminuyendo levemente la complejidad del problema.
- Se pretende llegar a soluciones de alta calidad (medidas en cantidad de vueltas dadas y en tiempo que les toma a los trenes completarlas) en poco tiempo de ejecución, por lo tanto, el algoritmo será diseñado para buscar rápidamente soluciones cercanas al óptimo, sin asegurar la optimalidad global del problema.

1.6. Metodología

La metodología aplicada para el desarrollo de esta tesis consta de los siguientes pasos:

- **Revisión de antecedentes operacionales de la mina:** leer información acerca de los principales procesos e indicadores operacionales de la mina, con especial énfasis en las etapas de transporte de mineral.
- **Comprensión del problema:** tomar conocimiento del problema que afecta a la etapa de transporte principal de la operación minera, identificando las posibles causas, consecuencias y barajando alternativas para resolverlo.
- **Levantamiento de información relevante:** procesar y analizar las bases de datos que almacenen información útil para el planteamiento de posibles soluciones del problema.
- **Revisión bibliográfica:** indagar en la literatura acerca de trabajos, conceptos y herramientas que puedan guiar e inspirar el desarrollo de la propuesta de solución.
- **Modelamiento de la solución:** basándose en lo aprendido desde la literatura, modelar el problema mediante un algoritmo de optimización que logre encontrar soluciones

de alta calidad para resolver la coordinación de los trenes de la etapa de transporte principal de la mina.

- **Ajuste y validación del algoritmo:** realizar simulaciones de casos reales para ajustar los parámetros del modelo. Una vez calibrado el modelo, hacer una validación de la calidad de las soluciones entregadas por el algoritmo utilizando otros casos reales.
- **Análisis de resultados:** analizar exhaustivamente los resultados obtenidos al resolver el problema con el modelo de optimización desarrollado, identificando el valor generado por la herramienta para la empresa y proponiendo mejoras para perfeccionar su funcionamiento en el futuro.

1.7. Estructura de la Tesis

La estructura usada en esta tesis para describir el trabajo realizado es la siguiente:

- **Capítulo 2. Marco Teórico:** Se realiza una revisión bibliográfica de antecedentes, conceptos y herramientas con aplicaciones de interés para el problema que se resuelve en esta tesis, tales como grafos, el método de árboles de decisión, el algoritmo de Branch & Bound y la técnica de resolución en paralelo.
- **Capítulo 3. Descripción del Algoritmo:** Se explica el algoritmo basado en la estructura de un árbol de decisión binario, así como los métodos ideados para hacer que la construcción de las ramas sea lo más eficiente posible, incluyendo un procedimiento de poda de ramas que es apoyado por cotas elaboradas sobre los nodos.
- **Capítulo 4. Resolución en Paralelo:** Se describe el método de paralización del algoritmo, el que aprovecha la capacidad computacional permitiendo construir una mayor cantidad de nodos por minutos y realizar un recorrido más homogéneo por el árbol de decisión.
- **Capítulo 5. Resultados:** Corresponde a la presentación de los resultados obtenidos al aplicar el algoritmo en dos instancias, luego de haber calibrado y validado el modelo. En el primer caso se pretende lograr un mayor número de vueltas que en el turno original y en el segundo caso se intenta sincronizar un turno donde se complete el número máximo de viajes.
- **Capítulo 6. Conclusiones:** Se mencionan las principales conclusiones originadas a partir del trabajo desarrollado en esta tesis y de sus resultados, identificando las limitaciones que posee el algoritmo y proponiendo posibles trabajos futuros para resolverlas.

Capítulo 2

Marco Teórico

El problema del que esta tesis se hace cargo se caracteriza por involucrar una serie de decisiones respecto a la ruta que debe seguir cada uno de los trenes a través de la línea férrea y a las esperas que se les puede asignar dentro del turno producto de la interferencia con otros trenes. Con el objetivo de comenzar el trabajo con una sólida base teórica, en este capítulo se explican conceptos y herramientas con aplicaciones muy útiles en problemas de esa naturaleza, como el método de árboles de decisión, por ejemplo.

2.1. Grafos

Un grafo $G(N, A)$ consiste en un conjunto finito de nodos N y un conjunto finito de arcos A [2]. Cada arco conecta exactamente a dos nodos, a excepción de aquellos arcos que conectan a un nodo consigo mismo. Un nodo puede estar asociado a un número entero de arcos, donde se dice que esos arcos son incidentes al nodo, y en casos particulares hay grafos con nodos sin conexiones. Gráficamente los nodos se representan por círculos y los arcos por líneas que los enlazan, como se muestra en la Figura 2.1.

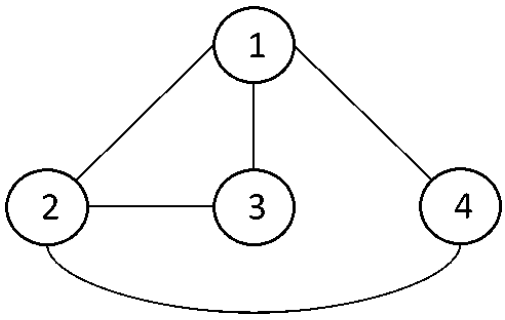

Figura 2.1: Grafo de 4 nodos y 5 arcos.

Si el conjunto N posee m elementos, entonces los nodos son denotados por $n_1, \ldots, n_m$. Por otra parte, los arcos se denotan según el subconjunto de nodos conectan, es decir, si un arco conecta al nodo n_{i} con el nodo n_j, será representado como $\{n_{\mathrm{i}}, n_j\}$.

En la Figura 2.1 se tiene un grafo cuyo conjunto de nodos es $N = \{n_1, n_2, n_3, n_4\}$ y el conjunto de arcos es $A = \{\{n_1, n_2\}, \{n_1, n_3\}, \{n_1, n_4\}, \{n_2, n_3\}, \{n_2, n_4\}\}$.

2.1.1. Grafo Dirigido

Un grafo se dice que es dirigido si los arcos poseen un sentido de desplazamiento, lo que significa que uno de los nodos conectados corresponde al nodo inicial, mientras que el otro es el nodo de destino del arco. En este tipo de grafos los arcos no se denotan como conjuntos, sino que por pares ordenados (n_{i}, n_j), donde n_{i} es el nodo de inicio y n_j el nodo de destino. Eso implica que $(n_{\mathrm{i}}, n_j) \neq (n_j, n_{\mathrm{i}})$ si $\mathrm{i} \neq j$.

El grafo dirigido de la Figura 2.2 tiene igual conjunto de nodos que el árbol de la Figura 2.1, pero el conjunto de arcos cambia a $A = \{(n_1, n_2), (n_1, n_3), (n_2, n_3), (n_2, n_4), (n_4, n_1)\}$. Notar que en este caso los arcos son ilustrados con flechas que indican su dirección.

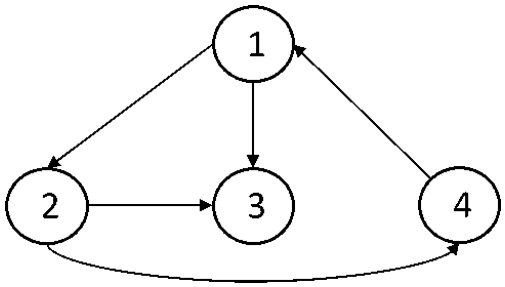

Figura 2.2: Grafo dirigido de 4 nodos y 5 arcos.

2.1.2. Ruta

Una ruta se define en un grafo dirigido como una secuencia de nodos que están conectados entre sí por arcos dirigidos que van en un mismo sentido. Se les dice nodo inicial y nodo final al primer y último nodo de la ruta, respectivamente. La ruta es "hacia adelante" si toda la secuencia de nodos sigue la dirección de los arcos dirigidos, mientras que la ruta es "hacia atrás" si sigue la dirección contraria. Una ruta es "simple" cuando no tiene nodos repetidos.

En la Figura 2.3 se muestran dos posibles rutas para llegar desde el nodo 1 (inicial) hasta el nodo 3 (final), a partir del grafo de la Figura 2.2. Ambas rutas son simples y hacia adelante.

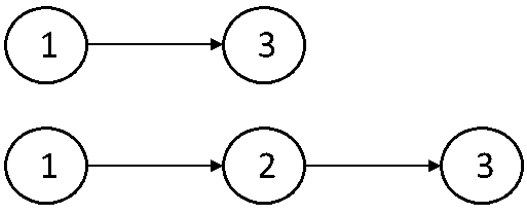

Figura 2.3: Dos rutas simples para llegar desde el nodo 1 al nodo 3.

2.1.3. Ciclo

Se le denomina ciclo a una ruta en la que el nodo inicial y final coinciden. La ruta de la Figura 2.4, construida según el grafo de la Figura 2.2, comienza y termina en el nodo 1, entonces corresponde a un ciclo.

Un grafo se dice acíclico si no contiene ciclos entre sus nodos.

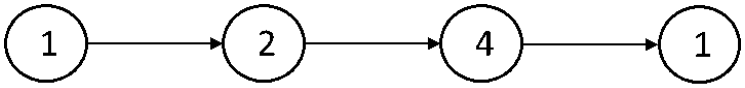

Figura 2.4: Ciclo desde el nodo 1 hasta el nodo 1.

2.1.4. Grafo Conexo

Un grafo es conexo si para todo par de nodos n_{i}, n_j existe al menos una ruta que comienza en n_{i} y termina en n_j.

2.2. Árboles

Un árbol se define como un grafo acíclico y conexo, es decir, no existen ciclos y cada par de nodos tiene una ruta que los conecta. Si en el grafo $G(N, A)$ existe un árbol que contiene todos los nodos, se dice que es un árbol de expansión de $G(N, A)$.

Algunas de las propiedades que cumplen los árboles son:

- Un árbol de n nodos tiene exactamente $(n-1)$ arcos.
- Un árbol tiene al menos dos nodos con grado igual a 1, donde el grado se define como el número de arcos que inciden al nodo.
- Cada par de nodos de un árbol está conectado por una única ruta simple.
- Un subgrafo sobre un grafo $G(N, A)$ de n nodos es un árbol de expansión si y solo si es conexo y tiene $(n-1)$ arcos.

En la Figura 2.5 se muestra un árbol de expansión sobre un grafo de 6 nodos. Si el grafo tuviera un séptimo nodo que no formara parte del árbol, entonces el árbol perdería la condición de ser árbol de expansión sobre el grafo.

2.2.1. Árbol Dirigido

Un árbol es dirigido si cumple con todas las propiedades de un árbol y además sus arcos son dirigidos.

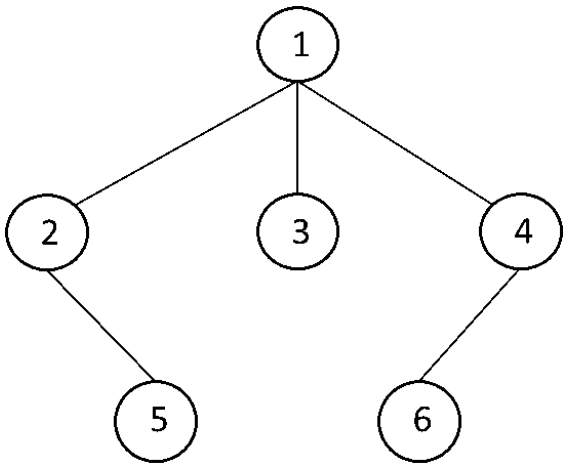

Figura 2.5: Árbol de expansión sobre un grafo de 6 nodos.

Este tipo de estructuras son muy utilizadas para el ordenamiento de datos y el desarrollo de algoritmos, de forma que poseen una notación particular para facilitar su uso, como se define en [4]:

- **Nodo padre:** el nodo n_{i} es padre del nodo n_j si están conectados por el arco dirigido (n_{i}, n_j).
- **Nodo hijo:** el nodo n_j es hijo del nodo n_{i} si ambos están conectados por el arco dirigido (n_{i}, n_j).
- **Raíz:** el nodo n_1 se denomina raíz del árbol si es padre de uno o más nodos y además no tiene padre. Es el nodo inicial del árbol y cada árbol tiene una única raíz.
- **Hoja:** una hoja es un nodo que no tiene hijos. También son denominados nodos terminales.
- **Subárbol:** es un subconjunto de nodos del árbol tal que juntos pueden formar un árbol de menor tamaño. La raíz de un subárbol puede tener padre dentro del árbol, pero no dentro del subárbol.
- **Ancestro:** el nodo n_l es ancestro del nodo n_m si existe una ruta hacia adelante que conecta a n_l con n_m. En todo árbol se cumple que existe siempre una única ruta que conecta a la raíz con cualquier nodo, por lo tanto, la raíz es ancestro de todos los nodos del árbol.
- **Descendiente:** el nodo n_m es descendiente de n_l si existe una ruta hacia adelante que conecta a n_l con n_m. Todos los nodos del árbol son descendientes de la raíz.
- **Profundidad:** la profundidad del nodo n_k es igual al largo de la ruta entre la raíz y n_k.
- **Altura:** la altura del nodo n_k es el máximo largo de ruta desde el nodo hasta alguna hoja. La altura del árbol es igual a la altura de la raíz y siempre se cumple que la altura del árbol tiene el mismo valor que la profundidad de la hoja más profunda.

En la Figura 2.6 se observa un árbol dirigido de 15 nodos. A partir de ese árbol, se ejemplifican los términos descritos anteriormente:

- n_1 es la raíz del árbol.

- $\{n_4, n_7, n_8, n_{12}, n_{13}\}$ es un subárbol de raíz n_4.
- n_2 es padre de n_6.
- n_7 es hijo de n_4.
- n_6 es ancestro de n_{14}.
- n_{12} es descendiente de n_4.
- $n_3, n_5, n_7, n_9, n_{11}, n_{12}, n_{13}, n_{14}, n_{15}$ son las hojas del árbol.
- La profundidad de n_{12} es 3. La mayor profundidad la tienen las hojas n_{14} y n_{15} cuyo valor es 4.
- La altura de n_4 es 2, mientras que la altura de n_3 es 0. La altura del árbol es 4.

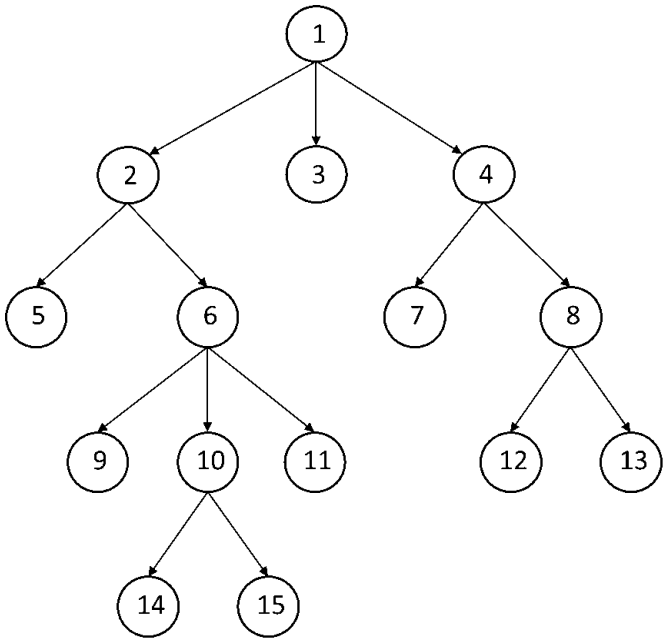

Figura 2.6: Árbol dirigido de 15 nodos.

2.2.2. Árbol Binario

Un árbol binario es aquel donde cada nodo es padre de exactamente dos nodos, a excepción de las hojas. Uno de los hijos es conocido como hijo o rama de la izquierda mientras que el otro es el hijo o rama de la derecha.

Si al árbol de la Figura 2.6 se le quitan los nodos n_3 y n_{11}, se transforma en un árbol binario (Figura 2.7). Desde el punto de vista de la raíz del árbol, el nodo n_2 es su rama de la izquierda, mientras que el nodo n_4 es la rama de la derecha.

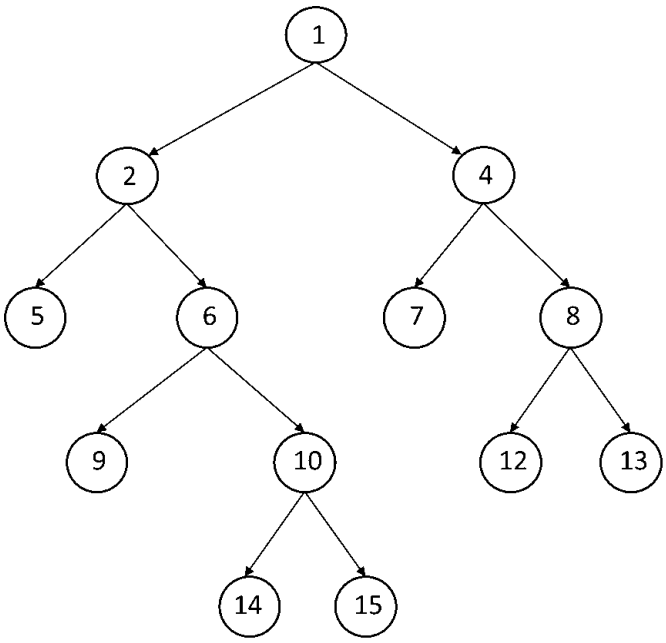

Figura 2.7: Árbol binario de 13 nodos.

2.3. Árboles de Decisión

Los árboles de decisión son un método de trabajo ordenado para resolver problemas de toma de decisiones. Entre mayor es la complejidad y tamaño del problema, más útil se hace el uso de la metodología, siendo muy provechosa cuando se aplica a problemas en que se deben tomar decisiones secuenciales en distintos instantes de tiempo y donde se cumple que entre una decisión y otra se agrega nueva información o cambian los valores de algunas variables [5].

Un árbol de decisión puede ser caracterizado mediante la estructura de un árbol dirigido. Los nodos representan un evento aleatorio o determinístico que al producirse genera una nueva combinación de valores para las variables del problema, mientras que los arcos indican las decisiones que pueden ser tomadas a partir de ese instante.

Si el problema tiene un número finito de decisiones, entonces cualquier camino que comience desde la raíz y avance a través de los nodos tendrá un final, llegando a una hoja donde ya no quedan decisiones por tomar. Estos nodos terminales u hojas se caracterizan por tener en su memoria la secuencia de decisiones tomadas para llegar a ese punto, además del beneficio (respecto a la función objetivo del problema) que implica haber escogido esas decisiones.

La forma de resolver un árbol de decisión comienza por construir completamente el árbol con todas las posibles decisiones y eventos, para luego hacer un barrido desde las ramas terminales hasta la raíz, determinando en cada nodo la decisión que reporte mayor utilidad, junto con el beneficio asociado.

Este método será simplemente ocupado como referencia para resolver el problema en el

que se enfoca esta tesis, pues es usado preferentemente con problemas que poseen eventos aleatorios, mientras que en este trabajo solo se consideran eventos determinísticos. Dado lo anterior, se ejemplificará el método con un sencillo problema que cuenta únicamente con eventos determinísticos.

2.3.1. Ejemplo de Árbol de Decisión Determinístico

El ejemplo de elaboración propia, consiste en que una persona desea viajar a dos ciudades de Europa y tiene la posibilidad de comprar solamente un pasaje aéreo, donde las alternativas disponibles son Madrid, Roma y Londres. Como cuenta con poco presupuesto, la segunda ciudad escogida debe estar ubicada en el mismo país al que llega en avión para poder trasladarse de forma terrestre. Si llega a Madrid, puede visitar Barcelona o Valencia, si llega a Roma sus opciones son Milán o Florencia y si llega a Londres podría ir a Liverpool o Manchester. Cada una de esas ciudades le reportan ciertas unidades de beneficio a la persona, dependiendo de sus gustos y preferencias. En la parte izquierda de la Figura 2.8 se muestra la tabla de beneficios de cada destino.

Lo descrito anteriormente implica que la persona debe tomar dos decisiones secuenciales, así que puede ser representado mediante un árbol de decisión, tal como se observa en la parte derecha de la Figura 2.8. En primera instancia cuenta con tres opciones y cada una de ellas abre un abanico de dos opciones diferentes.

Ciudad	Beneficio
Madrid	20
Roma	15
Londres	25
Barcelona	20
Valencia	10
Milán	15
Florencia	10
Liverpool	5
Manchester	10

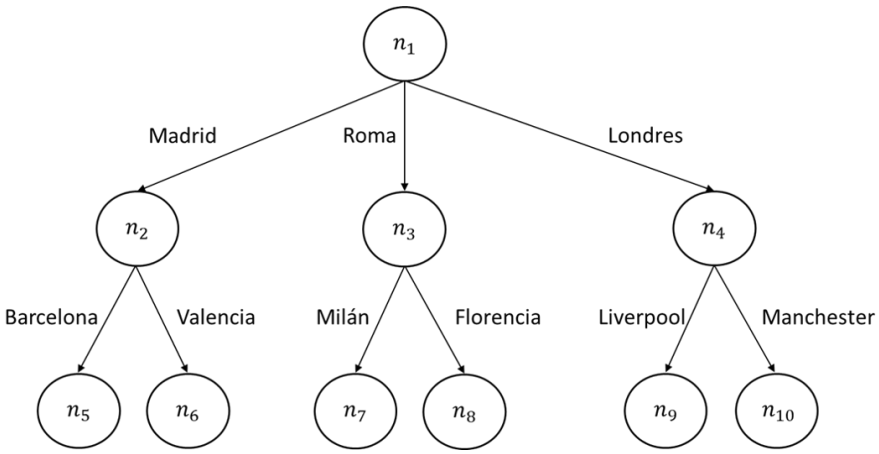

Figura 2.8: Tabla de beneficio de visitar cada ciudad y árbol de decisión del ejemplo planteado.

La forma de resolver el árbol es recorriéndolo desde las hojas hasta la raíz. Inicialmente se calcula el beneficio total de la secuencia de decisiones para llegar hasta las hojas. Por ejemplo, la hoja n_7 (según la notación de la Figura 2.8) se origina a partir de la secuencia Roma-Milán, de manera que tiene un beneficio de $15 + 15 = 30$.

Una vez que se obtienen las utilidades de todos los nodos terminales (Figura 2.9), se procede a decidir la segunda ciudad a visitar, dado que se comienza desde el final. Eso se consigue simplemente inclinándose por el arco que deriva en la hoja de mayor beneficio en cada caso. Desde el nodo n_2 se decide Barcelona, desde n_3 se escoge Milán y desde n_4 se prefiere Manchester. Esas elecciones quedan detalladas en el árbol al destacar los arcos

asociados a ellas, tal como se muestra en la Figura 2.9. Además, cuando se toma la decisión en el nodo previo se escribe el beneficio que genera la elección, para que se pueda transmitir esa información a los niveles superiores del árbol. En el caso de n_4 la decisión fue Manchester que tiene utilidad de 35 unidades, por lo tanto, en n_4 se escribe ese valor.

Finalmente, en el momento en que se tienen listos los beneficios de los nodos n_2, n_3 y n_4, se procede a decidir la primera ciudad en visitar. En este caso la persona decidirá Madrid que reporta un beneficio de 40. Por lo tanto, la secuencia de elecciones óptima es viajar a Madrid y luego trasladarse a Barcelona.

La raíz debe contener el beneficio de la serie de decisiones óptima del árbol, la que posteriormente se puede determinar con facilidad recorriendo los arcos destacados. La misma lógica de resolver el árbol se aplica para problemas más grandes y problemas que incluyen eventos aleatorios.

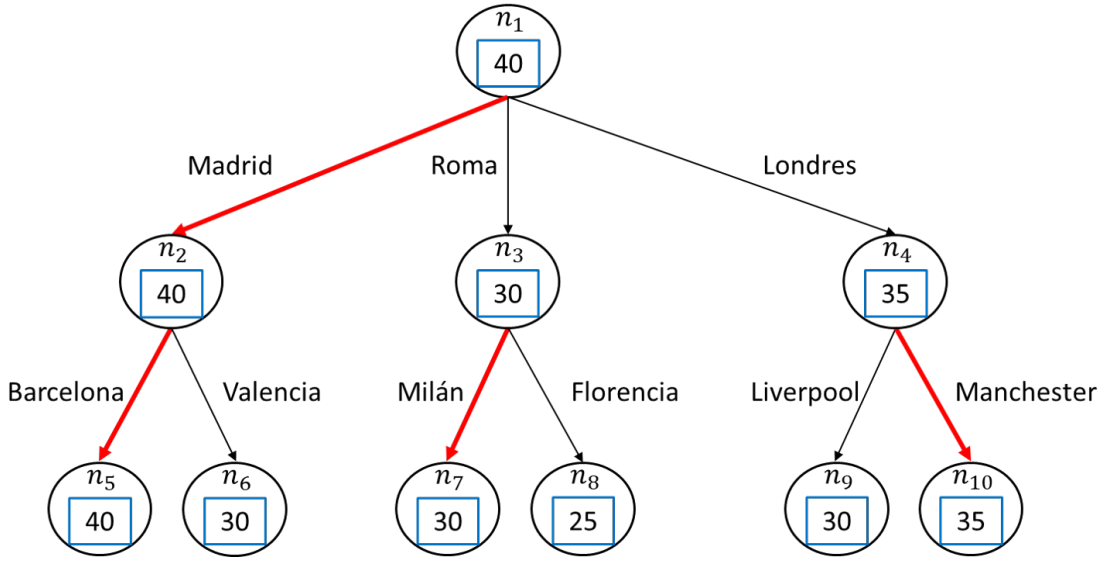

Figura 2.9: Resolución del árbol de decisión del ejemplo.

2.4. Branch & Bound

Branch & Bound es un algoritmo basado en una estructura de árbol, que es utilizado para resolver problemas de programación entera. La particularidad de este método es que no necesita recorrer completamente el árbol para llegar al óptimo, sino que rápidamente va descartando ramas con soluciones de baja calidad sin que estas hayan llegado a los nodos terminales.

2.4.1. Problemas de Programación Lineal y Entera

Un problema de optimización se caracteriza por tener variables de decisión, restricciones sobre esas variables y una función objetivo. La programación lineal es el área de la optimización que se encarga de aquellos problemas cuyas variables de decisión pertenecen al conjunto de los números reales, pudiendo ser enteras o no. Por otra parte, en la programación entera las variables están restringidas a tomar solo valores del conjunto de los números enteros.

En ambos casos para que el problema pueda ser resuelto con los métodos clásicos, tanto la función objetivo como las restricciones deben ser funciones y ecuaciones lineales en las variables, respectivamente. Eso significa que ni en la función objetivo ni en las restricciones puede haber variables elevadas a una potencia distinta de uno o dentro de funciones no lineales como la función parte entera, por ejemplo. Sin embargo, las constantes que acompañan a esas variables si tienen permitido estar en funciones con esas características.

La siguiente maximización es un problema de programación lineal, ya que las variables deben ser mayores que cero, pero no necesariamente enteras.

$$\begin{gathered} \text{máx } 3x_1 - x_2 \\ x_1 + x_2 \leq 4 \\ x_1 - x_2 \leq 3 \\ x_1, x_2 \geq 0 \end{gathered}$$

La solución es $x_1 = 3,5$ y $x_2 = 0,5$, obteniéndose una función objetivo de 10.

Si al mismo problema se le agrega la condición de que las variables deben ser enteras, la solución cambia a $x_1 = 3$ y $x_2 = 0$, con una función objetivo igual a 9, peor que el valor anterior.

Cuando a un problema de programación entera se le quita esa restricción que lo caracteriza, se obtiene una programación lineal llamada "relajación lineal" del problema entero. Siempre se cumple que el valor óptimo de la relajación lineal es mejor o igual que el óptimo del problema entero, dado que tiene menos restricciones.

2.4.2. Algoritmo Branch & Bound

El algoritmo de Branch & Bound permite resolver problemas de programación entera basándose en la ramificación de la relajación lineal del problema, con el objetivo de encontrar cotas para el valor óptimo [1].

Como la relajación lineal tiene un valor óptimo mejor o igual que el problema entero, su solución establece una cota superior para el óptimo de este último. Por otra parte, cualquier solución factible del problema entero puede ser consideraba como una cota inferior del valor óptimo.

Antes de describir el algoritmo es necesario aclarar dos conceptos que son frecuentemente utilizados. El primero es "incumbente", que se denota por Z y se define como la mejor solución factible del problema entero encontrada hasta el momento. Mientras que el segundo es "problema activo", en referencia a cualquiera de los subproblemas creados que no haya sido ramificado ni descartado previamente. Un problema activo se representa como P_{i}, con i un número asignado generalmente por orden de creación.

Para que la explicación sea más sencilla de interpretar, cada paso del algoritmo será ejemplificado con la primera iteración del siguiente problema entero de minimización:

$$\begin{aligned} \text{mín } & -2x_1 - 3x_2 \\ & 8x_1 + 9x_2 \leq 36 \\ & 3x_1 + 7x_2 \leq 22 \\ & x_1, x_2 \in Z_+ \end{aligned}$$

En la inicialización del algoritmo se establece como primer problema activo P_0 a la relajación lineal del problema entero. Además, se define el incumbente $Z = +\infty$ (cuando es una maximización se debe definir $Z = -\infty$). Luego, los pasos de Branch & Bound son:

1. Elegir un problema activo P_{i} que se encuentre aún sin resolver. Si no hay, se termina el algoritmo y el valor óptimo es el incumbente Z.
   - El único problema activo en la primera iteración es la relajación lineal P_0, así que es el problema escogido.
2. Resolver el problema activo elegido. Si P_{i} es infactible regresar al paso 1 y si es factible seguir al paso 3.
   - P_0 resulta ser factible, donde el valor óptimo de la función objetivo es $z^* = -10,76$, con $x_1^* = 1,86$ y $x_2^* = 2,34$.
3. Si z^* es peor o igual que el incumbente, entonces regresar a 1. Si en cambio es mejor que el incumbente, avanzar al paso 4. Notar que en una minimización z^* es mejor cuando tiene un valor menor que Z, mientras que en una maximización pasa lo contrario.
   - $z^* = -10,76$ es menor que $Z = +\infty$, por lo tanto, se avanza al siguiente paso.
4. Si $x^* \in Z_+$, es decir, los valores óptimos de todas las variables de P_{i} son números enteros, actualizar el incumbente como $Z = z^*$ y regresar al paso 1. Si alguna de las variables es no entera, ir al paso 5.
   - Ninguna de las dos variables tiene un valor entero, de modo que no se actualiza el incumbente y se continúa con el siguiente paso.
5. Escoger solo una de las variables x_{i}^* que no son enteras y crear dos nuevos problemas activos iguales al actual P_{i}, pero agregando una restricción a cada uno. Al primero se le debe incorporar la restricción $x_{\text{i}} \leq \lfloor x_{\text{i}}^* \rfloor$ (función parte entera inferior) y al segundo se le adiciona $x_{\text{i}} \geq \lceil x_{\text{i}}^* \rceil$ (función parte entera superior).
   - En este caso se escoge la variable $x_1^* = 1,86$ con parte entera inferior igual a 1 y parte entera superior de valor 2. De esa forma se crea un nuevo problema activo P_1 al que se le agrega la restricción $x_1 \leq 1$ y otro problema P_2 al que se le adiciona $x_1 \geq 2$.

Notar que en los pasos 2, 3 y 4 existe una opción que obliga a volver al paso 1, impidiendo que el problema activo llegue al último paso donde el nodo se ramifica en dos. Esas tres condiciones son los llamados "criterios de poda": si el problema es infactible, si se llega a una solución solo con variables enteras y si el óptimo es peor que el incumbente. La primera condición de poda se basa en que ninguna rama originada a partir de ese problema infactible podría ser factible, mientras que los otros dos criterios se fundamentan en que las ramas originadas desde ese problema activo no van a generar soluciones mejores que el incumbente actual.

En la Figura 2.10 se muestra el desarrollo completo del árbol del ejemplo. El valor óptimo se consigue en el P_9 con $z^* = -10$, $x_1^* = 2$ y $x_2^* = 2$. Las ramas podadas del árbol son P_8 y P_6 por ser infactibles, P_3 y P_9 por ser soluciones enteras y P_7 con P_{10} por no ser mejores que el incumbente, que en esa iteración vale $Z = -10$.

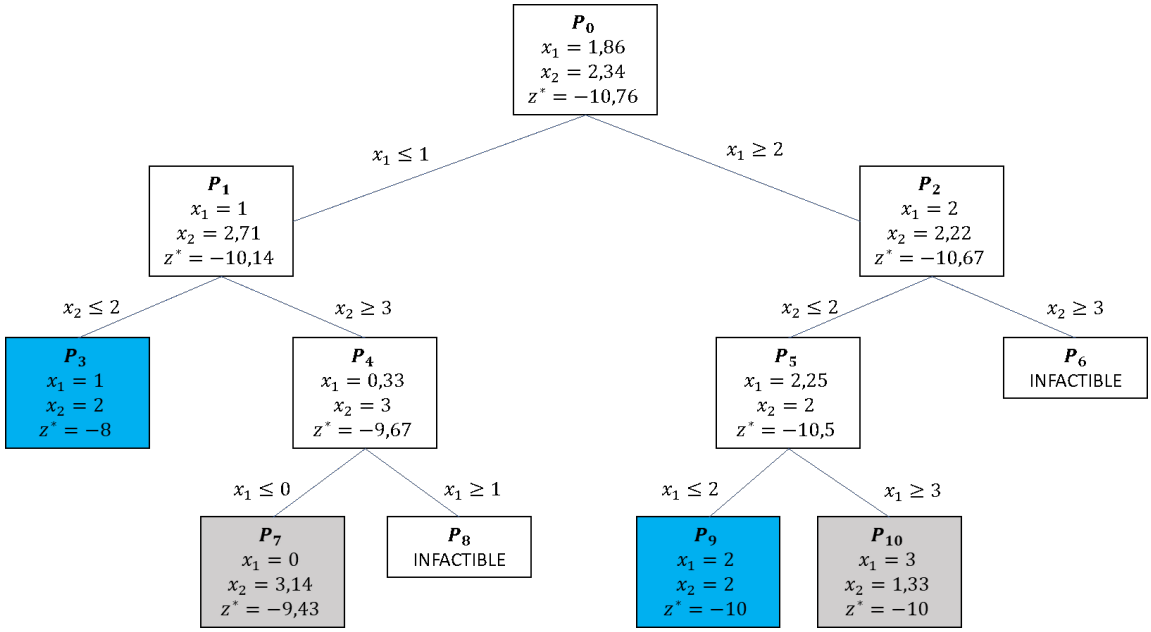

Figura 2.10: Árbol del algoritmo Branch & Bound para un problema de minimización.

En el paso 2 de Branch & Bound se escoge uno de los problemas activos para avanzar con el algoritmo, pero no se detallan criterios de elección. De todas formas, existen dos criterios típicos para hacer esa elección y recorrer el árbol:

- **Recorrido en anchura:** se escogen los problemas activos de manera que se recorra el árbol horizontalmente. En el ejemplo se aplica este criterio, pues una vez que se crean los problemas P_3 y P_4, estos se dejan en pausa y se vuelve a P_2 para completar ese nivel horizontal. Así mismo, antes de llegar a P_7, se itera en P_5 y P_6.
- **Recorrido en profundidad:** la segunda forma es recorrerlo en profundidad, es decir, priorizar una dirección del árbol e iterar hasta que ya no sea posible seguir ramificando. Si en el ejemplo se hubiese priorizado la rama de la izquierda, entonces P_3, P_4, P_7 y P_8 habrían sido recorridos antes que P_2.

No es posible asegurar que como norma general uno de los tipos de recorrido sea mejor, pero en algunos casos uno de los métodos permite encontrar el óptimo en menos iteraciones

que el otro, dependiendo de las características del problema.

2.5. Computación Paralela

Todos los computadores cuentan con un **sistema operativo**, por ejemplo Windows o Linux, que cumple con un rol fundamental en el funcionamiento del equipo. A su vez, un sistema operativo tiene como unidad principal al **kernel**, el que se encarga de gestionar la relación entre cada software y hardware asociado al PC. Un **software** corresponde a un programa o aplicación que puede ser ejecutado para cumplir cierta función o tarea, mientras que un **hardware** es un componente físico del computador, como un procesador o una memoria, el que contiene algún recurso útil para que los software puedan ejecutarse. Específicamente, el kernel administra los limitados recursos físicos que constantemente son solicitados por los software activos, así que en todo instante debe decidir a cuál programa otorgar un procesador y un espacio de memoria RAM, y durante cuánto tiempo, según una agenda de prioridad que es determinada por el mismo kernel. Cuando se cumple ese tiempo, el espacio facilitado al software se libera y puede ser reasignado a otro programa que necesite aquellos recursos.

Para comprender más en detalle la asignación de recursos durante la ejecución de un software, hay que saber que los programas están construidos en base a una serie de códigos escritos en cierto lenguaje de programación, los que entregan una precisa lista de instrucciones acerca de lo que debe realizar el software. Como se mencionó en el párrafo anterior, cuando se ejecuta un programa el kernel abre un proceso dentro de la CPU y le asigna al software un porcentaje de la memoria, además de un procesador completo o uno de sus subprocesos, entendiendo como subproceso a una de las unidades que conforman el procesador. Luego, el kernel hará que todas las tareas codificadas en el software sean procesadas de manera secuencial, es decir, solo una vez que se termina de completar una instrucción recién se pasa a la siguiente, respetando el orden con el que fueron programadas. Sin embargo, en muchos casos existen tareas que pueden ser procesadas de manera independiente sin la necesidad de que culmine una de ellas para trabajar en la otra. Dado eso, un software que tenga esa característica puede ser programado según los denominados threads (hilos en español), donde cada thread corresponde a una porción independiente del código que el kernel puede ejecutar de forma paralela a otros threads del mismo programa, limitado ciertamente por la cantidad de subprocesos que contenga el procesador del PC utilizado.

Ejecutar más de una tarea en paralelo otorga una gran ventaja, pues aumenta la velocidad a la que se procesan las instrucciones, lo que se significa en que un mismo problema puede ser resuelto en menor tiempo. El incremento en la velocidad también se puede traducir en que durante el mismo tiempo es posible resolver un problema de mayor magnitud.

Según la naturaleza del código bajo el cuál está programado el software, la paralelización se puede realizar en distintos niveles, siendo los más comunes el nivel de procesos y el nivel de información. La paralelización de procesos se produce cuando se ejecuta de manera simultánea más de un thread del programa en diferentes subprocesos del procesador, donde los threads pueden eventualmente acceder a las mismas funciones, pero siempre se hacen cargo de tareas independientes. Por otra parte, la paralelización de información se aplica si los datos son

divisibles en partes autónomas, de modo que cada thread lee y modifica solo su respectiva porción de la información.

A pesar de que los threads se ejecutan de forma independiente, es sumamente necesario que haya una buena coordinación entre ellos, lo que se logra con una constante comunicación tanto en el proceso como en la información. Eso quiere decir que los threads deben coordinarse en todo momento para turnarse el uso de las funciones del código que más de un thread necesita leer y además deben compartir con una alta frecuencia los datos que cada uno de ellos va generando, para así llegar a una respuesta conjunta a los requerimientos del usuario del software.

2.5.1. Teoría de Paralelización

Tal como se explicó anteriormente, la utilización de varios threads corriendo en paralelo permite aumentar la velocidad del programa y disminuir el tiempo necesario para finalizar la ejecución, respecto a lo que se consigue aplicando el clásico método secuencial. La ganancia en tiempo de ejecución ($G(n)$) obtenida gracias al hecho de paralelizar en n threads se puede definir por la fórmula descrita a continuación, donde $T(1)$ es el tiempo requerido para ejecutar el programa con solo 1 thread (lo que equivale a correrlo totalmente en forma secuencial) y $T(n)$ es el tiempo en que termina la ejecución del programa al usar n threads [3]:

$$G(n) = \frac{T(1)}{T(n)}$$

Además de la ganancia en tiempo, la eficiencia $E(n)$ es otra medida importante para medir la calidad de cada thread y de la paralelización en general, la que se define como el promedio por thread de la ganancia en tiempo:

$$E(n) = \frac{G(n)}{n} = \frac{T(1)}{n \cdot T(n)}$$

En la mayoría de los casos, un software no puede paralelizarse por completo debido a que algunas partes del código necesitan ejecutarse obligatoriamente de manera secuencial y a que ciertas funciones son requeridas por todos los threads para cumplir con sus respectivas tareas. La fracción no paralelizable del programa genera un cuello de botella en la ganancia de tiempo lograda al ejecutar varios threads, convirtiéndose en un factor limitante para el aumento de la eficiencia. Por ejemplo, si la parte no paralelizable demora h horas en ejecutarse, entonces aunque sean muy eficientes los threads, el programa jamás podrá demorarse menos de h horas.

En ese sentido, la **Ley de Amdahl** entrega una fórmula para establecer la ganancia en tiempo máxima esperada ($GM(n, p, s)$) que se puede conseguir al ejecutar en paralelo n threads del mismo software. Este indicador está en función del porcentaje del programa que es posible paralelizar (p) y el porcentaje no paralelizable ($s = 1 - p$):

$$GM(n, p, s) = \frac{1}{\frac{p}{n} + s}$$

La Figura 2.11 muestra como se comporta la ley de Amdahl en términos de la ganancia de tiempo máxima que se puede conseguir paralelizando en n threads, para distintos porcentajes de la parte paralelizable de cierto programa. Se observa que a medida que aumenta el número de threads también aumenta la ganancia máxima, aunque desde cierto punto la tasa de incremento se reduce, llegando hasta una cantidad de threads donde ya no es posible elevar la ganancia máxima. Además, se visualiza que entre mayor es la porción paralelizable del problema, es más la ganancia de tiempo máxima que se puede alcanzar.

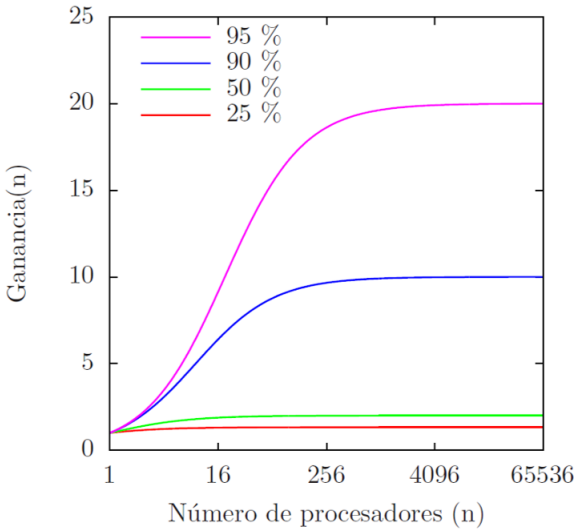

Figura 2.11: Gráfico que representa la le de Amdahl en términos de la ganancia de tiempo máxima que se puede alcanzar para distinta cantidad de threads. Cada línea muestra la curva para diferentes porcentajes de partes paralelizables de algún programa.

Por otra parte, existe la **Ley de Gustafson**, la que indica que la ganancia en tiempo ($G(n)$) que se alcanza al ejecutar n threads está dada por la siguiente ecuación, donde s es la porción no paralelizable del programa:

$$G(n) = n - s \cdot (n - 1)$$

En la Figura 2.12 está representado mediante un gráfico el comportamiento de esta ley para diferentes porcentajes de porciones no paralelizables o partes secuenciales de un determinado programa. Las líneas demuestran que entre mayor es la parte no paralelizable, es más rápido el incremento de la ganancia en tiempo a medida que aumenta el número de threads que se ejecutan en paralelo.

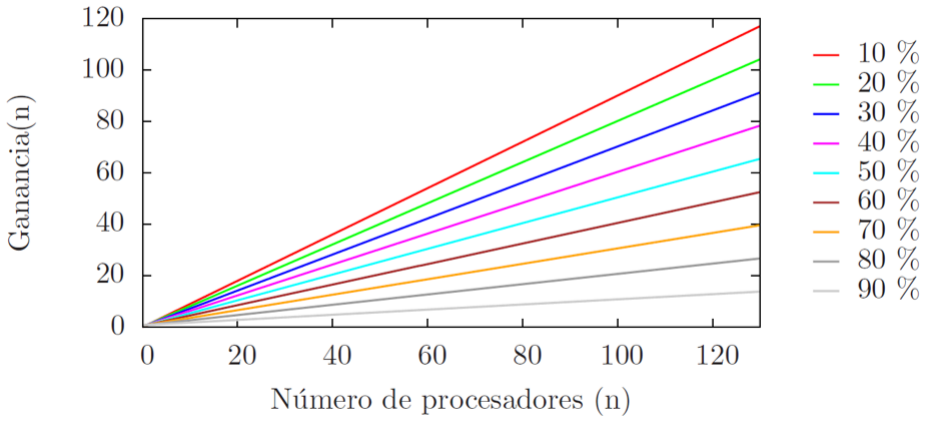

Figura 2.12: El gráfico muestra la evolución de la ley de Gustafson a medida que aumenta el número de threads para distintos porcentajes de porciones no paralelizables de cierto problema.

Capítulo 3

Descripción del Algoritmo

En este capítulo se describe el algoritmo creado con el objetivo de encontrar soluciones de alta calidad para la sincronización de los trenes durante un turno en la mina, de modo que se logre incrementar el flujo de mineral acarreado entre la zona interior mina y el sector de descarga, el que está ubicado sobre la superficie a un costado de la mina. El algoritmo permite probar con rapidez distintas combinaciones para coordinar a los trenes, entregando en poco tiempo una solución cercana al óptimo, sin asegurar necesariamente la optimalidad global del problema.

El algoritmo puede aplicarse a turnos con diferentes características, siempre que se tengan los valores de los parámetros necesarios, tanto si se tiene certeza de ellos (turnos pasados) como si se cuenta con una estimación (turnos futuros). Para efecto de esta tesis se considera como un parámetro más la asignación de los trenes a las entregas de viajes en los piques. A pesar de eso, el algoritmo tiene la suficiente rapidez como para permitir que en un tiempo razonable los usuarios ejecuten distintas asignaciones y puedan comparar los resultados, identificando oportunidades de mejora en ese ámbito.

En consecuencia, algoritmo se encarga de determinar las rutas, movimientos y detenciones de los trenes a través de la red ferroviaria para cumplir con una asignación dada de trenes a piques. Dicho problema tiene una alta complejidad, pues para un turno normal existen en torno a 10^{12} combinaciones distintas para coordinar a los trenes. Justamente la relevancia de resolver este problema se justifica por la gran cantidad de posibilidades, ya que una persona no tiene la capacidad de evaluarlas todas y eso abre un espacio de oportunidades de mejora que el algoritmo puede aprovechar para obtener mayor cantidad de entregas acarreadas durante un turno gracias a una buena sincronización de los trenes, tal como se pudo ver en un caso mostrado en el **Capítulo 1.3** a modo de ejemplo.

3.1. Representación de la Red Ferroviaria

Los trenes circulan de forma continua a través de las líneas de la red ferroviaria, sin embargo, al modelar el algoritmo resulta difícil replicar ese movimiento continuo, de manera

que se opta por discretizar el desplazamiento de los trenes y así hacer menos complejo el problema a resolver. Para ello se construye una simplificación de la red original que fue mostrada en la Figura 1.2, creando una red compuesta por un conjunto de puntos discretos por los que circularán los trenes.

La modificación de la red está compuesta por 37 vértices [1] unidos por aristas [2], donde cada vértice representa un tramo de la red original y cada arista es un enlace entre dos tramos. Eso permite que los trenes transiten solamente a través de 37 puntos perfectamente identificados, consiguiendo hacer discreto su traslado.

Todo vértice tiene un tiempo asociado, el que indica los minutos transcurridos para que un tren atraviese el tramo de la red original representado por ese vértice. Eso quiere decir que si en la red original un tren se demora 5 minutos en cruzar un segmento de vía, entonces cuando el tren ingrese al vértice que simula ese tramo, permanecerá como mínimo 5 minutos ahí antes de moverse a otro vértice. El tiempo es distinto en ambas direcciones, pues la red tiene pendientes y los trenes transitan con diferente peso dependiendo del sentido de movimiento, lo que afecta la velocidad a la que circulan.

En la Figura 3.1 se muestra la red con 37 vértices, donde se hace una división para distinguir las tres zonas principales: interior mina, túnel y zona de descarga.

En el **sector interior mina** hay 26 vértices que representan la totalidad de los tramos de vía férrea existentes. En ninguno de ellos puede haber dos o más trenes en el mismo instante, dado que los tramos no tienen la longitud suficiente para que pueda ingresar más de un tren. Los 10 vértices terminales de la red corresponden a aquellos segmentos de vía que alojan a los piques principales, por lo tanto, son los puntos donde los trenes cargan el mineral. A ellos se les suma el vértice llamado XC12/13 que también contiene piques, a pesar de no estar ubicado en un extremo.

El **túnel principal** está compuesto por 5 vértices. Dos de ellos indican ambos puntos extremos del túnel, otros dos la zona media, mientras que el quinto vértice corresponde al desvío interior del túnel, lugar que puede ser utilizado solo por un único tren que se desplace en dirección mina, para permanecer en espera hasta que el túnel se desocupe de los trenes que van hacia el sector de descarga. Al igual que en el sector interior mina, cada vértice representa un tramo original donde no puede haber más de un tren a la vez.

Finalmente se encuentra la **zona de descarga**. Como en este sector la red real tiene una gran cantidad de tramos de vía diferentes, si todos se replicaran fielmente en la red modificada el número de vértices existentes aumentaría considerablemente. Además, al haber muchos tramos disponibles, los trenes poseen distintas alternativas para desplazarse entre los chancadores y el inicio del túnel en un tiempo muy similar. Lo anterior permite que los trenes circulen entre los chancadores y el túnel sin mayores interferencias, por lo tanto, la coordinación en ese sector no es tan compleja como en las otras dos zonas.

[1] Es necesario aclarar que el término "vértice" es equivalente a "nodo" en la terminología de los grafos, sin embargo, la palabra vértice se aplicará únicamente a la red férrea simplificada, mientras que nodo será utilizado para el desarrollo del algoritmo y su árbol de decisión.

[2] Al igual que en el caso anterior, "arista" es equivalente a "arco" en un grafo, pero arista se usará solamente para la red ferroviaria y la palabra arco se aplicará en el algoritmo y su árbol de decisión.

Por las razones expuestas se redujo la red del sector de descarga a solo 6 vértices, los que agrupan varios segmentos de vía, dando la posibilidad de que haya más de un tren en cualquiera de esos vértices. Los dos primeros vértices corresponden a las líneas de descarga, existiendo uno para chancadores primarios y otro para chancadores secundarios. En cuanto al tiempo de permanencia de los trenes en los vértices de esta zona, consiste en un promedio de los minutos de traslado por los tramos originales que agrupa cada vértice.

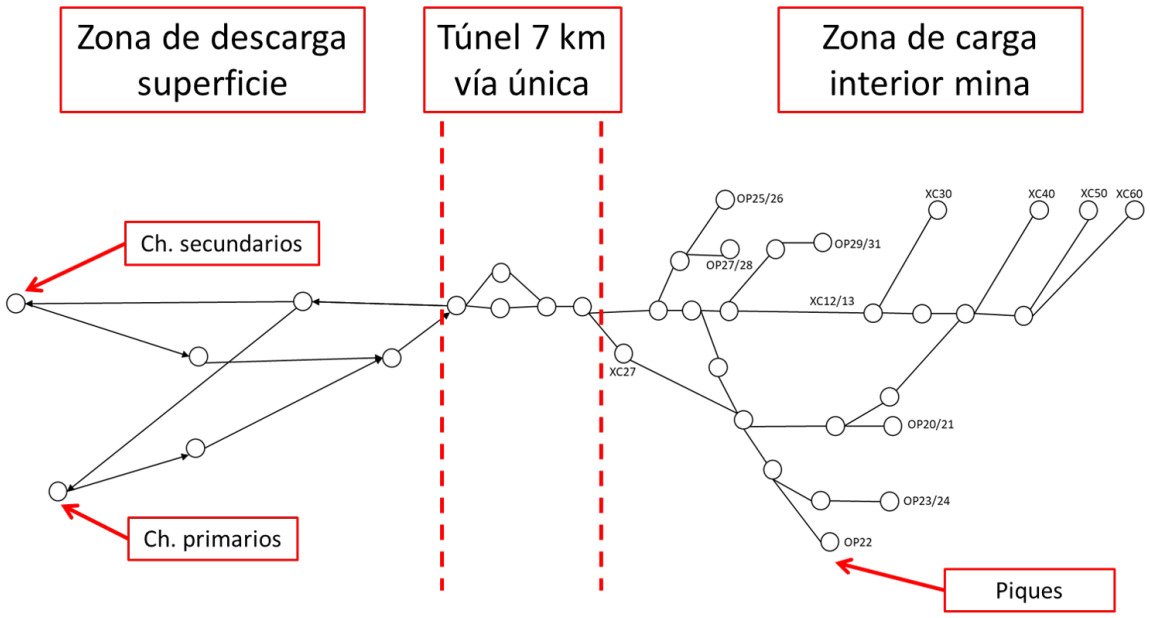

Figura 3.1: Red ferroviaria simplificada para efectos del algoritmo.

3.2. Árbol de Decisión Binario

A medida que un tren va circulando por las vías, constantemente se enfrenta a interferencias con otros trenes que intentan desplazarse por el mismo tramo en sentido contrario. En todos esos casos se debe tomar una decisión que tiene dos alternativas: dejar avanzar al tren que se dirige hacia el interior de la mina o dejar avanzar al tren que se dirige hacia los chancadores de la zona de descarga, obligando al segundo tren a esperar en un vértice de la red donde no intervenga en el camino del tren que obtiene la preferencia. Durante un turno completo se puede producir una gran cantidad de interferencias, generando una larga secuencia de decisiones con dos opciones de elección.

Dadas las características del problema y basado en los conceptos explicados en el marco teórico de esta tesis (**Capítulo 2**), el algoritmo es construido según la estructura de un árbol de decisión binario. La raíz del árbol será el nodo n_0, que marca el comienzo del turno en evaluación y que contiene los parámetros con toda la información inicial. A partir de ese nodo se genera la primera decisión y, por ende, la primera ramificación.

3.2.1. Ramificación del Árbol

El nodo n_0 guarda los datos iniciales del turno, tales como la cantidad de trenes y su numeración (ID), el vértice de la red en el que comienza cada tren y los piques a los que debe dirigirse en cada vuelta, según la asignación ingresada como parámetro.

Dentro de toda vuelta un tren tiene dos destinos: un pique y luego una línea de descarga. Para que los trenes arriben a los destinos designados, se les otorga una ruta según la secuencia de vértices que le permita al tren llegar en menos tiempo. Esas rutas fueron determinadas previamente y están dentro de la información recibida como input, por lo tanto, el algoritmo no necesita ejecutar el problema del camino mínimo para otorgarlas.

En la Figura 3.2 se muestran 3 recorridos distintos para que el tren T4 pueda desplazarse desde el sector de descarga hacia la mina, específicamente hasta el vértice que fue identificado por la letra M en el primer cuadro de la imagen. En la **opción A**, el tren cruza en línea recta el túnel principal y luego utiliza el vértice llamado desvío XC27 para llegar a su destino. En la **opción B**, el T4 pasa por el desvío del túnel principal y después cruza el desvío XC27. Por último, en la **opción C** el tren no usa el desvío del túnel y luego prefiere otros vértices de la zona interior mina en lugar del desvío XC27. Solo el recorrido de menor tiempo será el predeterminado por el algoritmo, mientras que el resto se mantendrá como rutas alternativas en caso de ser necesarias. Es importante notar que los 3 recorridos tienen muchos vértices en común, diferenciándose solo en pocos tramos.

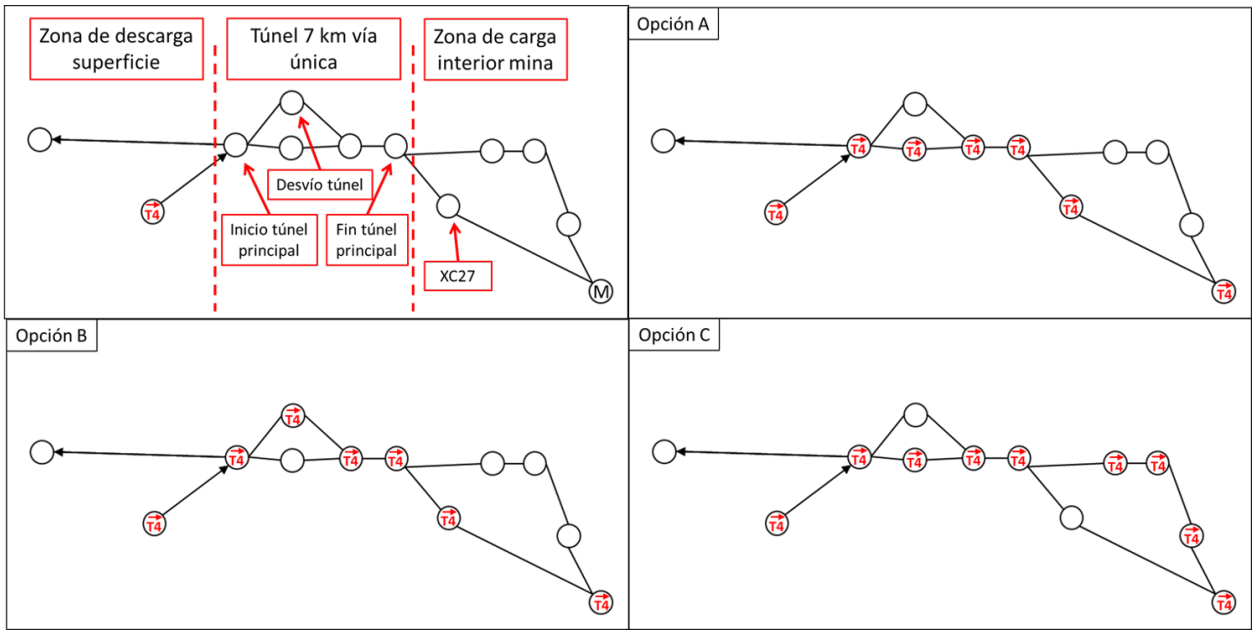

Figura 3.2: Se muestran 3 rutas diferentes para que el T4 se traslade desde el sector de descarga hasta el vértice de la mina denotado por M en el primer cuadro de la figura.

Desde el instante inicial del turno todos los trenes comienzan conjuntamente a hacer su recorrido por los vértices de la ruta asignada. Como el movimiento por la red modificada es discreto, una vez que los trenes ingresan a un vértice permanecen ahí un tiempo determinado

según los parámetros, el que representa los minutos que demora un tren en atravesar el tramo respectivo de la red original. Cuando un tren cumple con el tiempo de permanencia, se mueve hacia el siguiente vértice de su ruta.

Debido a que las diferentes rutas tienen en común muchos tramos de la red, especialmente segmentos del túnel principal y de la mina, en algún momento uno de los trenes avanza hacia un vértice en el que ya se encuentra otro tren circulando en sentido contrario. En aquellos casos se busca una ruta alternativa que esté desocupada para que uno de los dos trenes involucrados en la interferencia cambie su recorrido. Si para algún tren existe esa ruta, se le reasigna y todo el proceso continúa con normalidad. Pero, en caso de no existir alternativas, se produce un "choque" de dos trenes, tal como se detalla en el Algoritmo 1.

Algorithm 1 Desplazamiento de los trenes

1: //Itera dentro de un mismo nodo del árbol, hasta que se genera un "choque"
2: **Conjuntos**
3: T : conjunto de todos los trenes
4: C : conjunto de dos trenes que chocan
5: **Parámetros**
6: t_{d} : dirección de movimiento del tren t
7: t_v : vértice actual del tren t
8: t_m : minutos que le quedan al tren t en el vértice actual
9: t^* : primer tren en avanzar a siguiente vértice (tren de menor t_m)
10: c : 1 si se produce "choque" en el nodo actual del árbol y 0 si no
11: **Inicialización**
12: $C := \{\}$ //Se comienza sin "choques" dentro del nodo actual del árbol
13: $c := 0$
14: **while** $c \neq 1$ **do**
15: $t^* := j \in T \mid j_m = \mathrm{mín}\{t_m, t \in T\}$ //Busca tren de menor t_m
16: **if** $t^* = null$ **then**
17: Break //Todos los trenes finalizaron el turno y se termina el nodo
18: **else**
19: $t^*_v := t^*_{v+1}$ //El tren avanza al siguiente vértice de su ruta
20: Actualizar t^*_m
21: //Si hay otro tren en el mismo vértice moviéndose en dirección contraria
22: **if** $\exists\ t \in T \mid t \neq t^*, t_v = t^*_v, t_{\mathrm{d}} \neq t^*_{\mathrm{d}}$ **then**
23: **if** $\exists$ ruta alternativa para tren t^* **then**
24: Cambiar ruta de t^*
25: **else if** $\exists$ ruta alternativa para tren t **then**
26: Cambiar ruta de t
27: **else**
28: $c = 1$ //Se produce "choque"
29: $C := \{t, t^*\}$ //Se guarda la información de los trenes involucrados en el "choque"
30: **end if**
31: **end if**
32: **end if**
33: **end while**
34: **return** C

Tiempos de Espera

Al ocurrir un "choque", uno de los dos trenes debe retroceder hasta llegar a un vértice en que ya no interfiera con el camino del otro tren, mientras que el segundo tren obtiene prioridad para seguir avanzando por su ruta. El punto en el que culmina el retroceso se denomina "vértice de espera", lugar donde se le adiciona un tiempo de espera al tren, cuyo valor es igual a los minutos que pasan hasta que el tren con preferencia despeja el tramo de la ruta que tienen en común y donde se obstruyen el paso.

Además de las esperas producto de los "choques", también se asignan esperas cuando existen filas para avanzar en algunos vértices, particularmente en la entrada del túnel desde el sector de descarga hacia la mina y en los buzones de vaciado cuando los trenes entran llenos de mineral, ya que se cuenta solo con dos líneas de descarga por tipo de chancado. A todo el conjunto de situaciones donde un tren tiene que esperar debido a la interferencia con otros trenes, tanto por "choques" como por filas, se les denomina "esperas operacionales".

Por otra parte, si un tren está listo para ir a buscar mineral en un pique pero la entrega aún no está disponible, el tren debe permanecer en la zona de descarga hasta que el pique tenga el mineral suficiente para llenarlo, de modo que se le agrega una "espera por entrega". La Figura 3.3 indica el vértice donde se mantiene un tren en caso de tener asignada una espera de ese tipo.

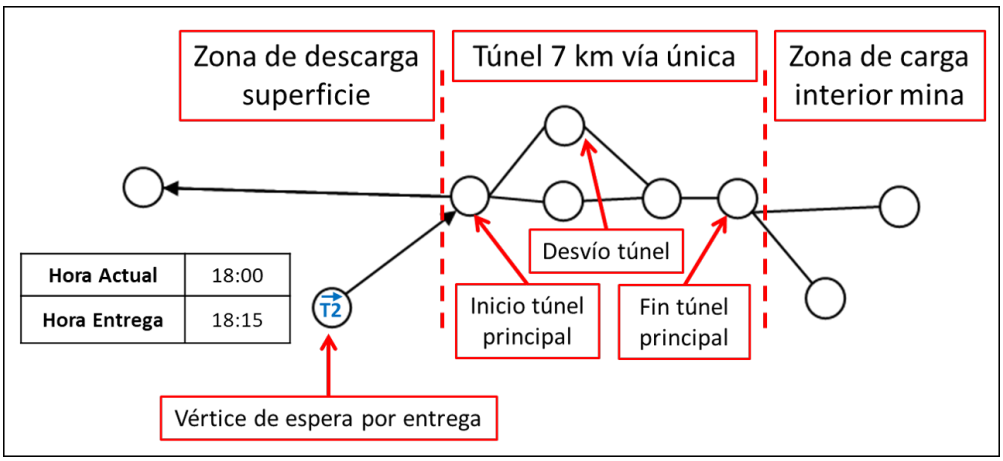

Figura 3.3: Vértice de la zona de descarga donde los trenes esperan cuando su entrega correspondiente todavía no está disponible para ser acarreada.

Los nodos del árbol van guardando toda la información que describe el desplazamiento de los trenes, sin embargo, cuando un tren retrocede producto de un "choque", su ida y vuelta hasta el vértice de espera es eliminado de la información del recorrido, dado que no es un movimiento lógico de realizar en la realidad. Al instante en que se entrega la solución del problema, simplemente se observará que el tren llega en un momento al vértice de espera y se le agrega inmediatamente el tiempo extra para evitar el futuro "choque", sin que el incidente se concrete efectivamente.

Lo que si se mantiene en la información guardada por los nodos es la ocurrencia del eventual "choque" entre ambos trenes, suceso que es fundamental para la construcción del

árbol. Cabe señalar que la mayoría de los "choques" tienen lugar en el túnel de vía única, el que cuenta solamente con un desvío con capacidad para un tren. Además, generalmente los trenes comienzan el turno juntos en la zona de descarga y avanzan en caravana durante toda la primera vuelta, sin embargo, desde la segunda vuelta en adelante los trenes se desordenan y empiezan a circular en sentido contrario por el túnel, produciéndose muchas obstrucciones.

Con la idea de dar mayor claridad a la parte inicial del método de ramificación del árbol, se describe a continuación un ejemplo donde ocurre un "choque" de dos trenes al interior del túnel principal, obligando a uno de ellos a permanecer en un vértice de espera hasta que su camino sea desocupado. La Figura 3.4 muestra una serie de imágenes con los eventos más relevantes del caso.

En el cuadro A de la ilustración se observa un tramo de la red simplificada, con todos los vértices del túnel y algunos vértices aledaños. Para identificar fácilmente los vértices durante la explicación, se les agregó una numeración que aplica solo en este ejemplo.

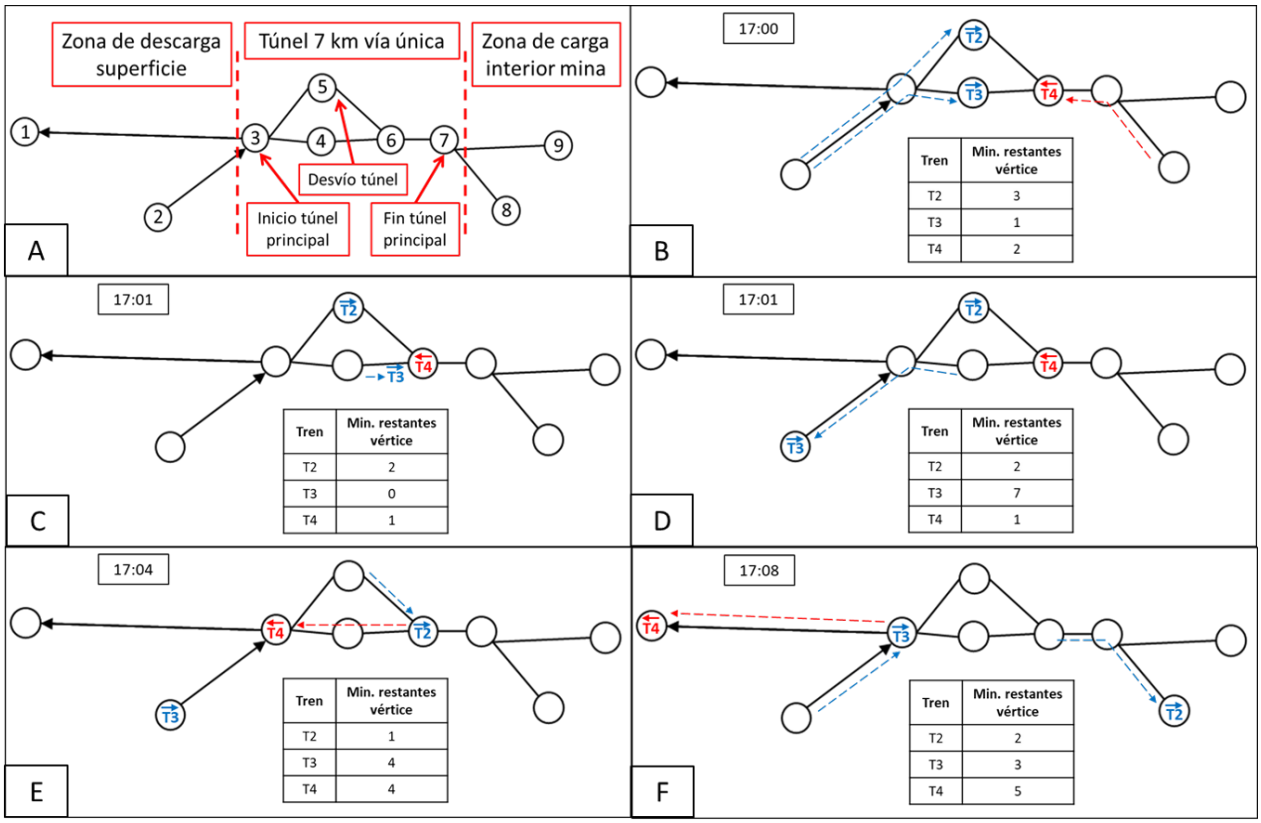

Figura 3.4: Caso en que dos trenes tienen un "choque" al interior del túnel.

- El caso explicativo comienza a las 17:00 (cuadro B), cuando el T2 se encuentra en el desvío del túnel desplazándose hacia la mina, mientras que el T3 circula por el túnel en sentido contrario al T4. De los tres trenes, el T3 es el más próximo a cambiar de vértice, pues le queda solo un minuto para pasar desde el vértice 4 al 6.
- Al cumplirse las 17:01 (cuadro C), el T3 avanza al vértice 6 y se encuentra de frente con el T4. No hay ruta alternativa disponible, dado que el desvío del túnel está ocupado

por el T2, por lo tanto, se produce un "choque" y uno de los trenes debe retroceder.

- Si bien existen dos opciones, en este caso se hará retroceder al T3, regresándolo hasta el vértice 2 donde no interviene en la ruta del T4 (cuadro D). Al T3 se le asigna una espera de 7 minutos, correspondiente al tiempo que le toma al T4 llegar al vértice 1, desocupando el túnel. Ese retroceso se borra del recorrido, así que en la solución final al T3 se le suma previamente el tiempo de espera en el vértice 3 para evitar el "choque".
- Finalmente, el T4 llega al vértice 3 a las 17:04 (cuadro E), desocupándolo 4 minutos más tarde. De esa manera, a las 17:08 (cuadro F) el T3 puede entrar al túnel y continuar su desplazamiento sin problemas.

En el ejemplo se le dio preferencia al T4 por sobre el T3, lo que corresponde a solo una de las dos alteratívas que se pueden escoger ante la ocurrencia de un "choque". Justamente los "choques" son los eventos que desencadenan una ramificación del árbol de decisión en dos nuevas ramas. La rama de la izquierda se genera si se da preferencia al tren que se dirige hacia el sector de descarga, denotando esa decisión con el número 1, mientras que la rama de la derecha nace al otorgar prioridad al tren desplazándose hacia la mina, lo que es denotado con el número 2.

Ambas ramas generan escenarios diferentes entre ellos y derivan en la formación de dos nuevos nodos del árbol, los que almacenan todos los parámetros y la historia del movimiento de los trenes hasta ese momento, incluyendo las decisiones que se han tomado en los "choques" previos, así como los trenes involucrados en esos eventos. Por ejemplo, el nodo n_{012} se forma luego de ocurrir un segundo "choque", donde en el primer "choque" se escogió darle preferencia al tren que va hacia la zona de descarga y en el segundo se decidió dejar avanzar al tren en dirección mina. En la Figura 3.5 se observan las primeras ramificaciones del árbol decisión.

Los nodos del árbol se pueden seguir ramificando hasta llegar a un nodo infactible o a un nodo terminal. Un nodo se declara infactible si el tren que debe retroceder no cuenta con un vértice de espera en el que no obstruya el camino del tren con preferencia, produciéndose de todas formas el "choque". Por otra parte, un nodo terminal se genera cuando ya no pueden suceder más "choques" debido a que se cumple alguno de los dos criterios que determinan el fin del turno: si todas las entregas disponibles en piques para el turno completo ya fueron acarreadas, o si se llega a la hora final del turno. Todos los nodos terminales otorgan una sincronización de trenes factible, así que son candidatos a ser la solución final del problema.

3.2.2. Método para Recorrer el Árbol

Durante un turno normal en el que no existan grandes problemas en la red ferroviaria, plantas o piques, se trabaja con 8 trenes y cada tren realiza 4 o 5 vueltas entre la mina y la zona de descarga. Bajo esas condiciones surgen alrededor de 40 "choques" [3] durante todo el turno a medida que se desplazan los trenes, generando $2^{40} \approx 10^{12}$ ramas diferentes en el árbol de decisión. La cantidad de combinaciones posibles incentiva a buscar una técnica eficiente para realizar el recorrido por las ramas, pues construir completamente un árbol de

[3] Cifra empírica calculada como un promedio de "choques" generados durante todo el turno en casos donde existen 8 trenes funcionando y cada tren completa entre 4 y 5 vueltas.

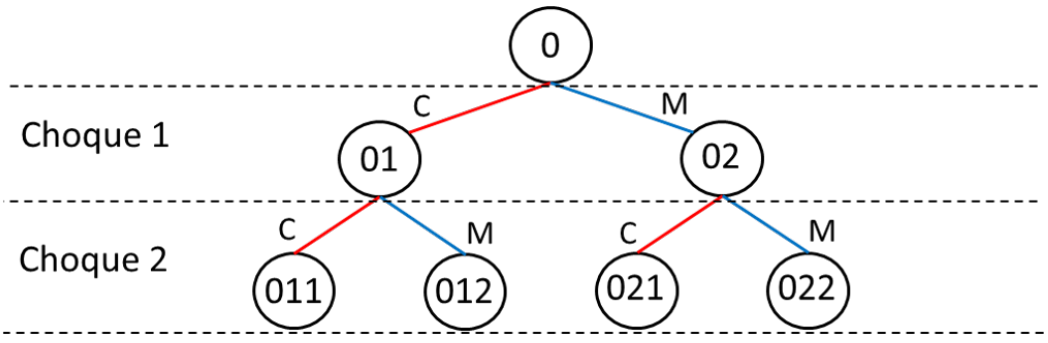

Figura 3.5: Primeras cuatro ramas del árbol de decisión binario. Los arcos de color rojo, denotados por "C", indican las ramificaciones donde se le da preferencia al tren que va hacia la zona de descarga. Los arcos de color azul, denotados por "M", indican las ramificaciones donde se le da preferencia al tren en dirección mina.

ese tamaño toma un tiempo demasiado alto, lo que haría impracticable su uso. Hablando en cifras más concretas, el tiempo en recorrer ese número de ramas supera los 1.200 años, considerando que el algoritmo necesita $3,65 \times 10^{-2}$ segundos para crear un solo nodo del árbol.

El algoritmo Branch & Bound aporta con dos métodos clásicos de recorrido: en anchura o en profundidad. Si bien en Branch & Bound ninguna de las dos formas es siempre superior a la otra, pues la opción más eficiente depende del problema específico, para el algoritmo de esta tesis alguna podría funcionar mejor de manera global.

Para corroborar lo anterior, ambos métodos fueron probados en el algoritmo, resultando más eficiente y rápido el recorrido en profundidad. Eso se explica porque el recorrido en anchura necesita prácticamente armar el árbol entero para llegar a algún nodo terminal con una solución factible, lo que requiere un tiempo excesivo dado el número de "choques" ocurridos en un turno. En cambio, el recorrido en profundidad permite encontrar muy rápido las primeras soluciones factibles del problema, ya que siempre itera en una sola dirección hasta llegar a alguna hoja del árbol, dejando en pausa el resto de los nodos. Además, al arrojar sincronizaciones factibles desde el comienzo de la ejecución, se pueden obtener cotas para realizar tempranamente podas sobre las ramas del árbol y así optimizar el recorrido, tal como en Branch & Bound.

El recorrido en profundidad tiene tres formas de ser aplicado: profundizar hacia la izquierda, hacia la derecha o de manera aleatoria. Para elegir la forma de hacerlo, nuevamente se realizaron experimentos y se comprobó que existe mayor eficiencia si se profundiza hacia la izquierda, lo que se traduce en recorrer inicialmente las ramas denotadas por el número 1, en las que se prioriza el tren con dirección hacia la zona de descarga.

Eso quiere decir que, según el método escogido en este trabajo para hacer el recorrido, el árbol se ramifica hacia la izquierda hasta que se encuentra con un nodo infactible o un nodo terminal. Recién en ese instante el algoritmo retrocede un nodo y ramifica una vez hacia la derecha, para luego seguir privilegiando la izquierda hasta tener que detenerse nuevamente. En la Figura 3.6 se muestra la manera de recorrer los nodos en un árbol con tres "choques", utilizando un recorrido en profundidad hacia la izquierda. Los nodos poseen una etiqueta a

un costado para indicar el orden en el que son visitados.

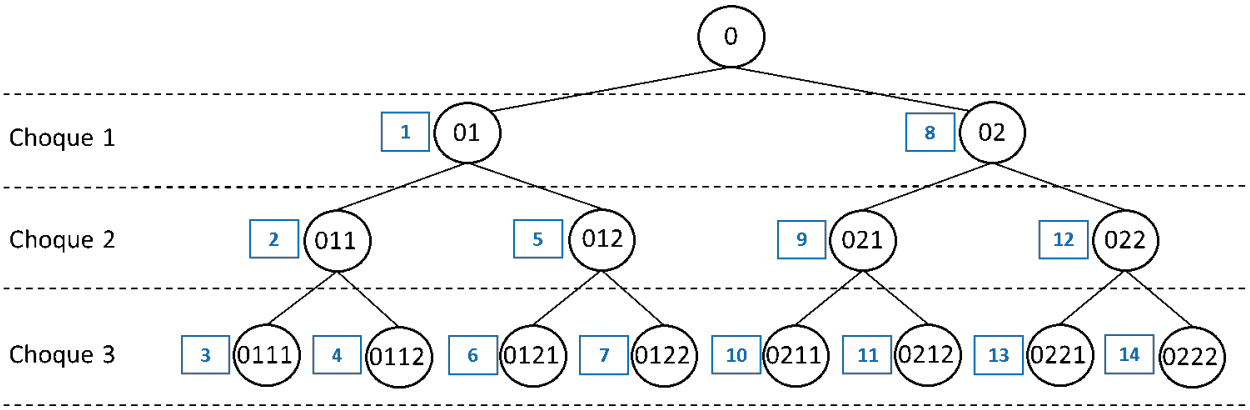

Figura 3.6: Ramificación en profundidad hacia la izquierda. Las etiquetas indican el orden en el que son recorridos los nodos.

La explicación del método de ramificación para el árbol se encuentra en forma de pseudo-código en el Algoritmo 2.

Algorithm 2 Ramificación del árbol

1: //Cada vez que se genera un "choque" de trenes dentro de un nodo del árbol se realiza una ramificación, creando dos nuevos nodos a partir del nodo actual
2: **Conjuntos**
3: N : conjunto de todos los nodos (se agregan nodos a lo largo de las iteraciones)
4: NT : conjunto de nodos terminales (se agregan nodos a lo largo de las iteraciones)
5: NI : conjunto de nodos infactibles (se agregan nodos a lo largo de las iteraciones)
6: C : conjunto de dos trenes que chocan (varía dependiendo del "choque")
7: **Parámetros**
8: n_k : nodo actual en el que se está iterando, con k su secuencia numérica de decisiones
9: k_{i} : número (1 o 2) en la posición i > 0 de la secuencia numérica k (siempre $k_0 = 0$)
10: x : largo de la secuencia k (igual al número de "choques" correspondientes al nivel del árbol en que está el nodo)
11: n_r : nodo del árbol con secuencia numérica r
12: c : 1 si se produce "choque" en el nodo actual y 0 si no
13: p : 1 si se activa **externamente** un criterio de parada del algoritmo y 0 si no
14: **Inicialización**
15: $N := \{n_0\}$, $n_k := n_0$
16: $p := 0$
17: **while** $p \neq 1$ **do**
18: **if** $(n_k \notin NT)$ and $(n_k \notin NI)$ **then**
19: **if** $c = 1$ **then**
20: $C = \{t_1, t_2\}$ // Con t_1 tren que choca en dirección mina y t_2 en dirección opuesta
21: $N := N \cup \{n_{\mathrm{i}}\}$, i $:= (0, k_1, k_2, ..., k_x, 1)$ //Ramifica a la izquierda y lo agrega a N
22: $N := N \cup \{n_j\}$, $j := (0, k_1, k_2, ..., k_x, 2)$ //Ramifica a la derecha y lo agrega a N
23: **if** t_1 tiene vértice de espera **then**
24: Actualizar vértice de t_1 y agregarle tiempo de espera
25: Dejar avanzar a t_2
26: **else**
27: $NI := NI \cup \{n_{\mathrm{i}}\}$ //t_1 no tiene vértice de espera y n_{i} es infactible
28: **end if**
29: **if** t_2 tiene vértice de espera **then**
30: Actualizar vértice de t_2 y agregarle tiempo de espera
31: Dejar avanzar a t_1
32: **else**
33: $NI := NI \cup \{n_j\}$ //t_2 no tiene vértice de espera y n_j es infactible
34: **end if**
35: $n_k := n_{\mathrm{i}}$ //Se continúa iterando hacia la rama de la izquierda
36: **end if**
37: **else**
38: **if** $k_x = 1$ **then**
39: $n_k := n_r$, $r := (0, k_1, k_2, ..., k_{x-1}, 2)$ //Recorre el nodo de la derecha del mismo nivel que el nodo actual
40: **else**
41: $n_k := n_r$, $r := (0, k_1, k_2, ..., k_{x-2}, 2)$ //Sube un nivel y luego recorre el nodo de la derecha
42: **end if**
43: **end if**
44: **end while**

3.2.3. Solución del Árbol de Decisión

Tal como se ha mencionado anteriormente, el algoritmo desarrollado está inspirado en la estructura de un árbol de decisión, sin embargo, dicha metodología requiere que el árbol sea construido completamente para encontrar la solución óptima, lo que para este problema en particular es imposible de lograr en un tiempo razonable, pues su dimensión es del orden de 10^{12} ramas que pueden demorar más de 1.200 años en ser recorridas en su totalidad. Por lo tanto, se elabora un método diferente de obtener la solución del problema.

El recorrido en profundidad del árbol va generando constantemente nuevos nodos terminales con soluciones factibles, lo que es aprovechado por el algoritmo para compararlas a medida que se crean y así poder determinar la solución general del problema. Para ello se utiliza un incumbente, al igual que en Branch & Bound, el que se actualiza cada vez que se llega a nodos terminales con sincronizaciones que sean mejores.

Previo a comparar soluciones, es necesario tener una función objetivo que establezca claramente los criterios de comparación. En este problema se pretende realizar la mayor cantidad de viajes posibles entre la mina y los chancadores durante un turno, por lo tanto, el número de vueltas es el primer indicador para decretar que una solución es mejor que otra. Ante la igualdad en ese valor, se pasa al segundo indicador, correspondiente al tiempo transcurrido desde el inicio del turno hasta que el último tren finaliza de vaciar el mineral en un buzón del sector de descarga, siendo mejor una solución mientras menor sea ese tiempo. Así las soluciones quedan caracterizadas por un par de indicadores $[v, t]$, con v el número de vueltas logradas y t los minutos necesarios para que todos esos viajes sean completados.

El algoritmo termina de correr cuando se cumple con alguno de los siguientes tres criterios de parada: si se sobrepasa un tiempo de ejecución límite, si se llega a una cantidad máxima de ramas recorridas o si se recorren todas las ramas del árbol. Los dos primeros criterios son parámetros ingresados como input al programa, mientras que el último se podría cumplir luego de muchas horas de ejecución, dependiendo del número de "choques" generados.

Cualquiera sea el caso, el algoritmo escoge como solución del problema al nodo del árbol que sea el incumbente vigente n_I en el momento de finalizar la ejecución, donde se define el incumbente de la misma forma que en Branch & Bound, es decir, la mejor solución encontrada hasta el instante actual. Al output entregado por el algoritmo se le denomina "sincronización del turno" y corresponde a la coordinación de los trenes asociada al nodo escogido como solución del problema, la que contiene los desplazamientos por la red, los procesos de carga y descarga de mineral y las esperas tanto operacionales como por entregas de todos los trenes a través de su recorrido durante el turno completo.

El algoritmo 3 contiene el procedimiento explicado anteriormente para obtener la solución del problema.

Algorithm 3 Solución del árbol

//Siempre que surge un nuevo nodo terminal del árbol de decisión, se compara su solución con el incumbente. Si la solución es mejor, se actualiza el incumbente.
//Entrega como solución del problema al incumbente vigente en el momento de activarse un criterio de parada del algoritmo
Conjuntos
NT : conjunto de nodos terminales (cuando surge un nuevo nodo terminal, se agrega)
Parámetros
n_I : nodo incumbente (nodo con la mejor solución actual)
V : número de vueltas realizadas por el nodo incumbente
T : tiempo en que el nodo incumbente completa las vueltas
v_{i} : número de vueltas realizadas por nodo terminal n_{i}
t_{i} : tiempo en que nodo terminal n_{i} completa las vueltas
p : 1 si se activa **externamente** un criterio de parada del algoritmo y 0 si no
Inicialización
$[V, T] := [0, \infty]$
$p := 0$
while $p \neq 1$ **do**
  if se agrega nuevo nodo terminal n_{i} en el conjunto NT **then**
    if $v_{\mathrm{i}} > V$ **then**
      $n_I := n_{\mathrm{i}}$ //Mayor número de vueltas, se actualiza incumbente
      $[V, T] := [v_{\mathrm{i}}, t_{\mathrm{i}}]$
    else if $(v_{\mathrm{i}} = V)$ and $(t_{\mathrm{i}} < T)$ **then**
      $n_I := n_{\mathrm{i}}$ //Igual número de vueltas y menor tiempo, se actualiza incumbente
      $[V, T] := [v_{\mathrm{i}}, t_{\mathrm{i}}]$
    end if
  end if
end while
return n_I //Al activar un criterio de parada, retorna el incumbente como solución

3.3. Método de Poda de Ramas

La creación computacional de un solo nodo del árbol requiere alrededor $3,65 \times 10^{-2}$ segundos, un tiempo bastante alto teniendo en cuenta que pueden generarse del orden de 10^{12} ramas, las que demorarían más de 1.200 años en construirse totalmente. Como el objetivo del algoritmo es apoyar la labor de los despachadores, es necesario encontrar soluciones de alta calidad en solo algunos minutos de ejecución, por lo tanto, se debe evitar gastar tiempo y recursos del PC elaborando ramas que deriven en sincronizaciones de baja calidad.

El algoritmo de Branch & Bound se hace cargo de esa problemática podando todas aquellas ramas que no tienen un resultado superior al incumbente, ahorrando tiempo de ejecución e iteraciones para encontrar el óptimo. Sin embargo, existe una gran diferencia entre el árbol de Branch & Bound y el árbol de esta tesis, que imposibilita replicar en exactitud esa técnica.

En Branch & Bound todos los nodos poseen una solución completa del problema, pudiendo ser factible o infactible, lo que otorga la certeza de que al descender en el árbol las soluciones irán empeorando y la seguridad de estar podando únicamente nodos peores que el incumbente. En cambio, en el algoritmo de este trabajo solo los nodos terminales tienen soluciones completas para sincronizar el turno en su totalidad, mientras que los nodos de niveles superiores cuentan con una sincronización parcial de los trenes, sin haber llegado aún al final del turno.

Dicho detalle impide que los nodos no terminales del árbol puedan ser podados mediante la comparación directa con el incumbente, pues no tienen una coordinación para todo el turno. De todas formas, se pueden establecer cotas para el conjunto de nodos terminales que descienden desde un nodo de un nivel superior, de modo que se tenga una referencia de la calidad de las potenciales soluciones originadas a partir de ese nodo no terminal.

Por ejemplo, suponer que en la resolución de un caso el nodo incumbente indica que cada tren completó cuatro vueltas en todo el turno, mientras que dentro de un nodo parcial particular del árbol los trenes se encuentran realizando su tercera vuelta a falta de dos horas para llegar al cambio de turno. Como ambos nodos están en instantes temporales distintos, en esas condiciones no hay manera de compararlos. Para hacerlo, se debe proyectar una cota del desempeño de ese nodo parcial desde su situación actual (tercera vuelta) hasta el final del turno, obteniendo en definitiva una cota para las soluciones de todos los nodos terminales que descienden a partir de aquel, la que sí es comparable con el incumbente del problema.

3.3.1. Generación de Cotas para Poda de Nodos No Terminales

Durante toda la ejecución del algoritmo el incumbente representa la mejor sincronización encontrada hasta el minuto, es decir, la secuencia de decisiones respecto a las rutas, desplazamientos y esperas de los trenes que hasta el instante actual permite realizar la mayor cantidad de vueltas dentro del turno y en el menor tiempo. La idea de la poda es descartar rápidamente los nodos no terminales que al ramificarse van a generar con total seguridad soluciones peores que el incumbente, para así enfocar el recorrido en ramas con mayores probabilidades de superarlo.

Al podar el nodo n_x, automáticamente se eliminan todos aquellos potenciales nodos terminales descendientes de él, los que tienen que ser considerados en el criterio de poda para no descartar de forma errónea una solución mejor que el incumbente. En consecuencia, la condición de poda deber ser: "si el mejor de los potenciales nodos terminales descendientes de n_x es peor que el incumbente, podar a n_x".

Como es imposible saber a priori el resultado exacto del mejor nodo terminal originado desde n_x sin recorrer exhaustivamente todas sus ramas descendientes, se trabaja con una cota superior que engloba a todas esas eventuales soluciones. Al ser cota superior existe la certeza de que no se podarán nodos mejores que el incumbente, pues no hay soluciones descendientes de n_x que sean mejores que su cota. Entonces, el criterio de poda cambia a: "si la cota superior de las soluciones descendientes de n_x es peor que el incumbente, podar a n_x".

Para establecer la cota se realiza una proyección optimista del movimiento de los trenes, desde el instante en que está parado el nodo n_x, hasta el final del turno. La cota no necesariamente tiene que ser una sincronización factible, sino que basta con que su resultado sea igual o mejor (más vueltas realizadas en el turno o igual cantidad de vueltas pero completadas en menor tiempo) que las soluciones factibles a las que abarca.

De esa forma, se construye una primera cota otorgándole a los trenes rutas sin detenciones por interferencias con otros trenes, entre el minuto actual del nodo n_x y el resto del turno. Eso quiere decir que los trenes pueden pasar simultáneamente por un mismo vértice en sentido contrario, acción evidentemente infactible, pero que siempre tiene un resultado mejor que las soluciones factibles acotadas, en las que sí hay detenciones operacionales.

En el ejemplo de la Figura 3.7 se muestra una secuencia de imágenes que ilustran la manera de estimar la cota para un nodo no terminal.

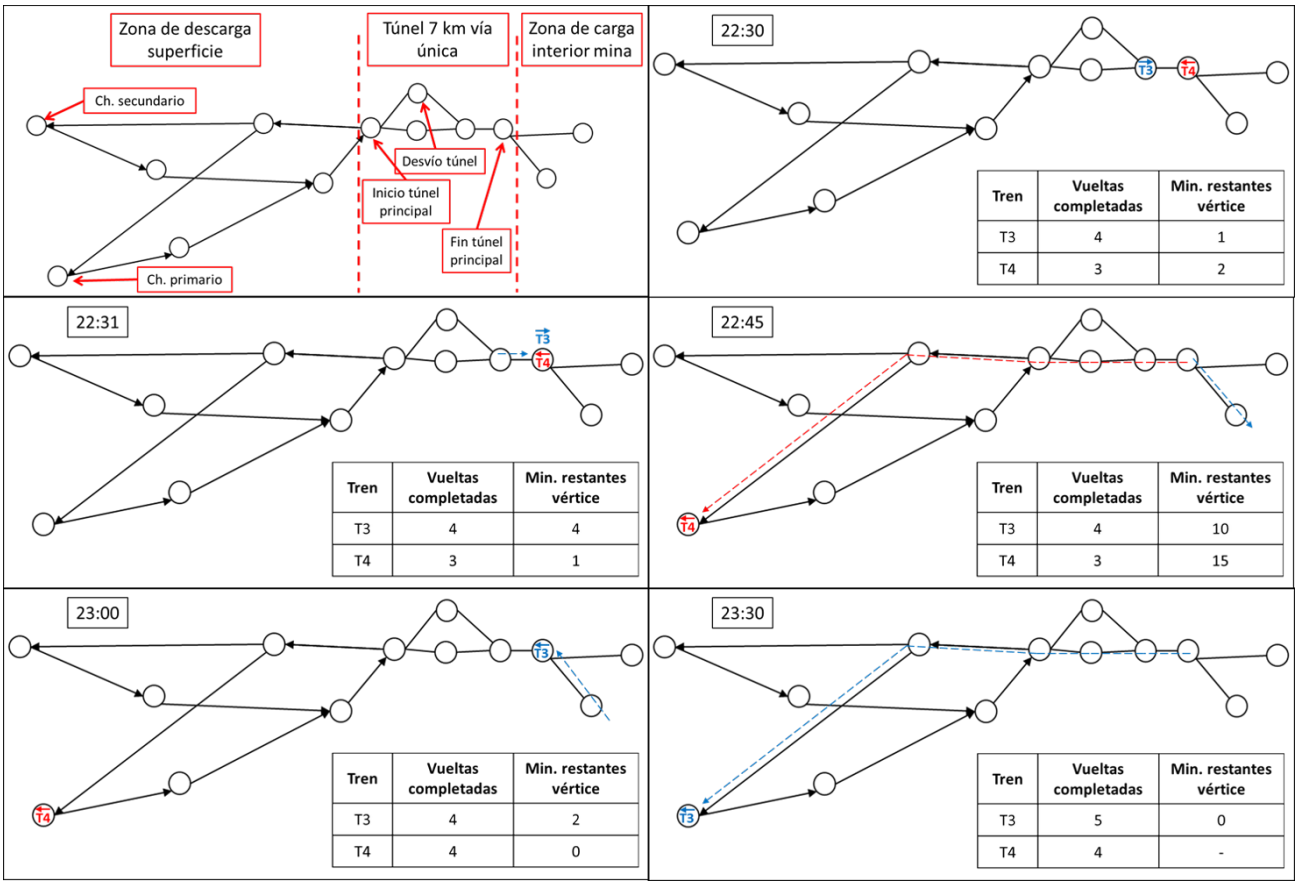

Figura 3.7: Caso de ejemplo que muestra el método para estimar la cota de un nodo no terminal del árbol. El segundo cuadro representa la situación actual de un nodo parcial y desde ese momento en adelante los trenes se mueven por la red sin considerar interferencias, hasta el final del turno. Las tablas muestran la cantidad de vueltas completadas por los trenes en cada instante y los minutos que les restan para permanecer en el vértice en el que se encuentran.

- El cuadro inicial de la figura contiene un segmento de la red ferroviaria utilizada por

el algoritmo, el que abarca desde la zona de descarga hasta los dos vértices de la mina que se localizan justo a la salida del túnel principal. En este ejemplo se asume que para cruzar el túnel principal son necesarios 10 minutos, mientras que para recorrer el conjunto de vértices de la mina o los del sector de descarga se requieren 5 minutos en total, independiente de la dirección de movimiento. En cuanto a los procesos de carga y descarga, cada uno demora 15 minutos.

- El segundo cuadro representa la situación actual de un nodo parcial del árbol. Dicho nodo se ubica temporalmente a las 22:30 de un turno B y su cota será estimada desde ese instante en adelante bajo las reglas descritas previamente. Para este caso solo hay dos trenes en funcionamiento: el T3 con 4 vueltas completadas y el T4 con 3 vueltas realizadas hasta el momento.
- Ambos trenes se mueven en dirección contraria por el túnel principal, de manera que si siguen sus respectivas rutas eventualmente chocarán. Pero, en el procedimiento de construcción de la cota para los nodos no terminales los "choques" son ignorados. Entonces, a las 22:31 el T3 avanza al mismo vértice donde permanece el T4, sin que los trenes deban retroceder ni esperar.
- Luego de cruzarse en un vértice del túnel, los dos trenes siguen circulando libremente por la red ferroviaria hasta completar sus vueltas asignadas. El T4 finaliza su cuarto viaje a las 23:00, mientras que el T3 culmina de descargar su quinta vuelta a las 23:30. Como en un turno B la hora límite de ingreso al túnel principal desde la zona de descarga es a las 23:00, ninguno de los trenes alcanza a completar otra entrega.
- Finalmente, se calcula del número de vueltas realizadas por los trenes y el tiempo necesario para terminarlas. En el ejemplo se logró completar 9 vueltas entre ambos trenes, finalizando la descarga del último tren (T3) a las 23:30. Ambas cifras dan forma a la cota que abarca a todos los nodos terminales originados desde el nodo parcial en evaluación y que sirve para comparar aquel nodo no terminal con el incumbente del problema.

El hecho de que la cota permita a todos los trenes desplazarse libremente sin detenciones, genera una estimación demasiado optimista, haciendo indispensable considerar algunos tiempos de espera para llegar a cifras más próximas a la realidad. Dado que se conoce la hora exacta en que cada entrega se encuentra disponible en su pique, se agregan en primer lugar esperas por entrega, las que se producen cuando un tren está preparado para comenzar un nuevo viaje pero el pique asignado aún no tiene lista la entrega de mineral. La adición de estos tiempos claramente empeora el resultado de la cota, permitiendo que se acerque a los valores de las soluciones factibles envueltas. Como la hora a la que se liberan las entregas de mineral son parámetros fijos previamente establecidos, las esperas por entrega afectan de igual manera a cualquier solución del árbol, de modo que la incorporación de aquellos tiempos sigue asegurando que la cota tiene un mejor resultado en relación a las soluciones factibles acotadas, es decir, las soluciones factibles que descienden del nodo desde donde se realiza la estimación.

A pesar de haber sumado esperas por entregas, al no poseer esperas operacionales la cota todavía puede estar muy alejada de las soluciones factibles englobadas, así que es conveniente agregar algunas detenciones de ese tipo si con seguridad se sabe de su ocurrencia. En efecto, se incluye una proyección de las esperas operacionales que tendrá cada tren dentro de la

vuelta que actualmente está recorriendo, producto de interferencias con otros trenes en la red ferroviaria y de filas generadas particularmente en las líneas de descarga de mineral y en la entrada del túnel principal. Recordando que durante un turno normal cada tren puede completar entre 4 y 5 viajes, la razón para considerar en la proyección de las esperas operacionales solamente la vuelta actual del tren, es porque desde ahí en adelante la gran cantidad de posibilidades no permite afirmar que efectivamente se concreten las esperas estimadas, pudiendo afectar la calidad de la cota superior.

En el método ideado para proyectar las esperas operacionales de los trenes durante la vuelta actual, inicialmente todos los trenes tienen como destino y posición sobre la red ferroviaria lo indicado por la situación en la que se encuentra el nodo parcial estudiado. Luego, cada tren se somete a una evaluación de su recorrido a través de la actual vuelta, identificando si se producen interferencias con otros trenes en alguno de los siguientes sectores: en los piques, en la entrada del túnel principal o en las líneas de descarga de mineral. La manera de abordar estos tres tipos de eventos es explicada a continuación:

1. **Interferencias en piques:** estos incidentes se analizan cuando dos o más trenes deben llegar durante la vuelta actual a piques ubicados en el mismo cruzado de la mina. Hay que tener en cuenta que cada vértice de la red que aloja piques tiene cierto tramo de vía con una ruta única para acceder a él, siendo aquellos segmentos donde se produce esta clase de interferencias. Por ejemplo, en la Figura 3.8 se destacan con color rojo los vértices que de manera inevitable tienen que recorrer los trenes para llegar a los piques OP23 y OP24, pues no existen más alternativas.

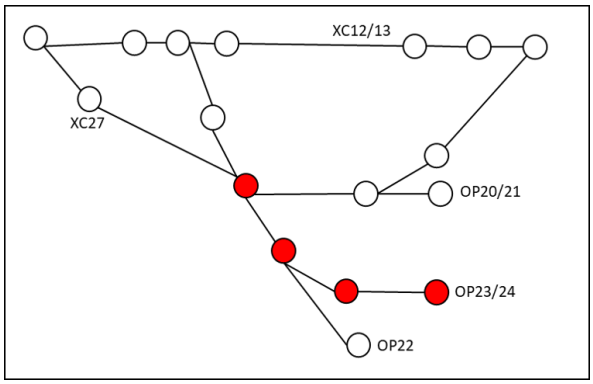

Figura 3.8: En color rojo se destaca el segmento de ruta única para acceder a los piques principales OP23 y OP24.

   Con el fin de estimar si un tren tendrá una espera gracias a una interferencia de estas características, para todos los trenes que van a cargar al mismo cruzado se proyecta su hora de entrada y salida del tramo único de acceso al pique respectivo. Si la hora de salida proyectada de uno de los trenes es posterior que la hora de entrada de otro tren que llega después al segmento exclusivo, entonces la diferencia en minutos será la espera que deberá efectuar el segundo tren antes de ingresar a aquel tramo, con lo que se evita un posible "choque".
2. **Interferencias en entrada del túnel principal:** recordar que según la red ferroviaria utilizada por el algoritmo, los vértices que representan al sector de descarga pueden recibir a más de un tren simultáneamente, ya que agrupan varios segmentos diferentes

e independientes de la red original. Por otra parte, en los vértices del túnel principal solo se permite la permanencia de un tren a la vez, lo que provoca un cuello de botella en el ingreso al túnel desde la zona de descarga y una posible fila de trenes a la espera de su oportunidad para avanzar. En la Figura 3.9 se muestra precisamente una cola de 4 trenes que se mantienen a la espera en el sector de descarga para poder ingresar al primer vértice del túnel, el que se encuentra ocupado por el T2. Cuando el T2 avance hacia el próximo vértice, el T3 podrá desplazarse al inicio del túnel y la fila quedará compuesta por 3 trenes. De igual forma ocurrirá con los siguientes trenes de la cola a medida que los trenes que van a la cabeza sigan avanzando vértices.

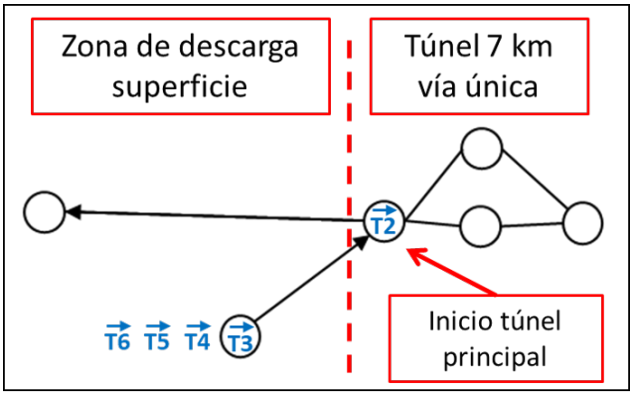

Figura 3.9: Fila de trenes a la espera para avanzar desde la zona de descarga hacia el túnel principal. El T3 puede entrar al túnel recién cuando el T2 se mueva al próximo vértice. Lo mismo ocurre con los siguientes trenes de la cola.

El tiempo de espera provocado por estas interferencias se estima calculando la hora en que cada tren llegaría al vértice inicial del túnel y la hora de desplazamiento hacia el siguiente vértice del recorrido. Si el tren que está primero en la fila generada en la zona de descarga (T3 en el ejemplo) tiene una hora de entrada al túnel menor que la hora en que el tren ubicado en el inicio del túnel (T2 en el ejemplo) cambia de vértice, entonces la diferencia entre ambos tiempos será la espera obligada del primer tren de la cola (T3), lo que repercutirá obviamente en la espera de los demás trenes de la fila. De ese modo, según la distribución de la cola, se estiman secuencialmente los minutos de espera de todos los trenes que forman parte de ella.

3. **Interferencias en líneas de descarga:** al igual que en la entrada del túnel principal, en las líneas de descarga se producen filas, especialmente en las dos líneas que reciben a los 6 trenes de mineral grueso. En el ejemplo de la Figura 3.10 se observa una cola de trenes esperando que el T4 desocupe una línea de descarga de mineral grueso, suponiendo, a modo de ejemplo, que en ese momento es la única habilitada debido a que la otra línea se encuentra con trabajos de mantenimiento.

   El cálculo de las esperas se realiza de manera similar a los dos casos anteriores. Primero se proyecta la hora de entrada y salida de la línea de descarga para cada tren. Cuando hay dos líneas de descarga disponibles y dos trenes utilizándolas, entonces se agrega una espera si otro tren llega a descargar antes que desocupe su línea el tren que finaliza primero el proceso de vaciado. En caso de que solo haya una línea habilitada, se suma una espera si la hora de llegada de un tren es anterior que la hora de salida del tren descargando mineral. En ambas situaciones el valor de la espera es igual a la diferencia

de los tiempos mencionados y se aplica el mismo cálculo sucesivamente para el resto de trenes en la cola.

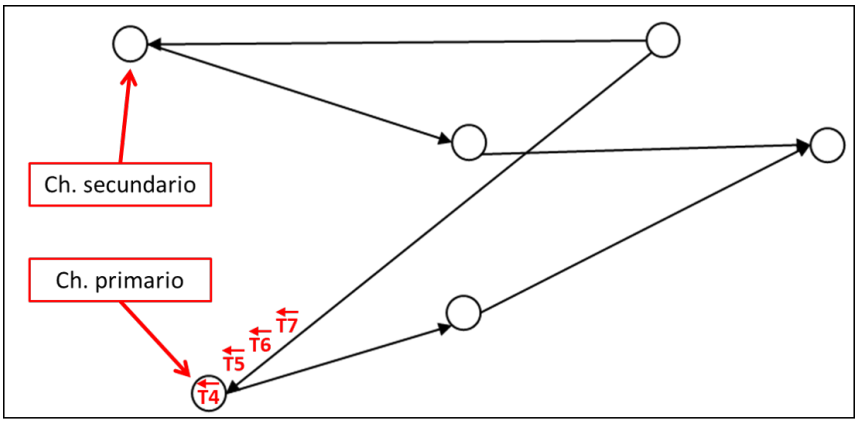

Figura 3.10: Fila de trenes en una línea de descarga de mineral grueso, suponiendo que es la única habilitada. El T4 está descargando, mientras los demás trenes se mantienen en espera.

A modo de resumen, a continuación se describe cada paso necesario para la creación de la cota de un nodo no terminal n_x, es decir, un nodo en el que existen trenes que aún están recorriendo alguna vuelta y no han finalizado el turno:

- La estimación de la cota comienza desde la situación actual de los trenes en el nodo n_x. Toda la historia previa del nodo se asume como un dato fijo, incluyendo las vueltas que han completado los trenes hasta el momento. Luego, el procedimiento se divide en dos partes: vuelta actual y vueltas siguientes.
- Vuelta actual: se proyectan las esperas operacionales que con seguridad afectarán a los trenes en la presente vuelta, ya sea por "choques", en el caso de las interferencias en los piques principales, o por filas producidas en las líneas de descarga de mineral y en la entrada del túnel en dirección mina. Hecho eso, se calcula el minuto en que cada tren finaliza la vuelta actual omitiendo cualquier espera diferente a las esperas por interferencias antes proyectadas y las esperas por entregas. En particular, no se consideran esperas por "choques" de trenes diferentes a los "choques" proyectados en el sector donde se encuentran los piques principales.
- Vueltas siguientes: posterior a completar la presente vuelta, se proyectan los siguientes viajes de cada tren hasta que se llegue al final del turno, según la disponibilidad de entregas en piques y cumpliendo con la restricción sobre la hora de entrada y salida del túnel principal. En dichas estimaciones solo se agregan esperas por entregas y se ignora cualquier tipo de interferencias con otros trenes, como si cada tren fuera el único desplazándose por la red ferroviaria.
- Finalmente, sumando las vueltas guardadas en la historia del nodo n_x y las vueltas estimadas, se obtiene la cota $[v_x^{cota}, t_x^{cota}]$, compuesta por la cantidad de viajes conseguidos en el turno entre la totalidad de los trenes y el tiempo que fue necesario para lograrlos, el que se cuenta hasta que el último tren finaliza el proceso de descarga de su vuelta final.

3.3.2. Poda de Ramas

Una vez que el nodo n_x obtiene su cota superior $[v_x^{cota}, t_x^{cota}]$, que representa una cota optimista e infactible para la cantidad de vueltas y para el tiempo, se realiza una evaluación del nodo con la finalidad de determinar si debe ser podado o no. El mejor punto de comparación que se puede tener es el nodo incumbente, ya que es la mejor solución factible completa encontrada hasta el momento. Se denota como V al número de vueltas asociadas al nodo incumbente y como T al tiempo requerido para completarlas, escribiendo como $[V, T]$ ambas cifras. Luego, los nodos se analizan según las siguientes reglas de poda:

Algorithm 4 Poda de ramas

Parámetros
n_x : nodo evaluado
V : número de vueltas realizadas por el nodo incumbente
T : tiempo requerido para realizar vueltas por el nodo incumbente
v_x^{cota} : cota de número de vueltas para nodo n_x
t_x^{cota} : cota de tiempo para nodo n_x
if $v_x^{cota} < V$ **then**
    Podar n_x //Debido a un menor número de vueltas
else if $(v_x^{cota} = V)$ and $(t_x^{cota} \geq T)$ **then**
    Podar n_x //Debido a igual número de vueltas, pero mayor tiempo
end if

La estimación de cotas es un proceso que necesita una capacidad computacional menor que la generación de ramas, por lo tanto, cada vez que se podan nodos de baja calidad aquello se traduce en un gran ahorro de tiempo que el algoritmo puede utilizar en recorrer mejores ramas, sobretodo considerando que desde un nodo sin podar pueden surgir millones de ramas descendientes antes de llegar a un nodo terminal con una solución factible mejor que el incumbente.

3.3.3. Gap de la Solución

Si bien en esta tesis el método de cotas fue hecho pensando en la poda de ramas, en muchos otros problemas de optimización las cotas se usan con la idea de obtener un gap para la solución.

La finalidad de un gap es tener una referencia de qué tan cerca o alejada está la mejor solución factible encontrada respecto al óptimo del problema, sin que se conozca evidentemente el óptimo, lo que ayuda a discriminar si la solución es de alta o de baja calidad. El valor del gap se define como la diferencia simple o porcentual entre el resultado de la mejor solución factible obtenida y la mejor cota superior estimada para el óptimo del problema, donde la mejor cota superior corresponde a la cifra más cercana al óptimo que se pueda encontrar y de la cuál se tenga certeza que sea siempre mejor que este. Por ejemplo, en programación entera la solución de la relajación lineal se considera como una buena cota superior inicial al

calcular el gap.

Al momento de escoger la cota del problema de esta tesis, se necesita que esta englobe absolutamente a todas las soluciones factibles del árbol. Una primera opción es estimar la cota (bajo el mismo procedimiento de las cotas asociadas a la poda de ramas) justo antes de que el nodo inicial n_0 se ramifique, es decir, en el instante previo a que ocurra el primer "choque" de trenes, pues desde aquel nodo desciende cualquier rama del árbol y su cota limita a todas las soluciones factibles. Sin embargo, como se calcula en el nivel más alto del árbol, ese valor puede estar muy distante de las soluciones factibles del problema. Es por ello que resulta conveniente buscar una forma de estimarla desde niveles más bajos del árbol.

Para lograrlo, inicialmente se construyen todos los nodos del árbol de decisión hasta un nivel o profundidad m, número que se elige de manera que se pueda recorrer en un tiempo razonable todo el árbol hasta aquel nivel. Luego, a partir de cada nodo de profundidad m se estima una cota optimista para las soluciones factibles de sus nodos descendientes, obteniendo un máximo de 2^m cotas diferentes, que en la práctica pueden ser mucho menos debido a la existencia de nodos infactibles en el nivel m y a que el valor de la cota se podría repetir para dos o más nodos.

Como se busca tener la mejor cota para el árbol completo, se toma la más optimista entre las 2^m cotas calculadas, asegurando que la cifra sea siempre mejor o igual que todas las soluciones factibles del árbol. Eso se traduce en elegir la cota con mayor cantidad de vueltas realizadas durante el turno y seleccionar en caso de empate la solución que complete esos viajes en menor tiempo. Finalmente, se llega a un par de cifras $[v_m^{cota}, t_m^{cota}]$ correspondientes a la cota para el número de vueltas realizadas y para el tiempo requerido en terminarlas, valores que son más realistas que los entregados por la cota estimada en la raíz del árbol.

De esa manera, asumiendo que el nodo n_x tiene la mejor solución factible encontrada (incumbente actual en el instante en que finaliza la ejecución del algoritmo), el gap se obtiene calculando la diferencia entre el número de vueltas logradas por la cota y por la mejor solución factible $(v_m^{cota} - v_x)$. Si la resta es mayor que cero, se decreta ese número como el gap de la solución, pero en caso de ser igual a cero, entonces el gap quedará determinado por la diferencia del tiempo en completar esas vueltas $(t_x - t_m^{cota})$.

Posteriormente se puede utilizar el gap para diversos análisis, teniendo en cuenta que entre menor es su valor más se acerca la solución encontrada al óptimo del problema, así que mayor es su calidad, lo que se puede demostrar con cifras concretas gracias al gap.

Capítulo 4

Resolución en Paralelo

4.1. División del Problema en Subárboles

La técnica para recorrer el árbol de decisión binario presentada en el **Capítulo 3.2.2** genera un sesgo en la solución del problema, pues prioriza en todo momento las ramificaciones hacia la izquierda y posterga el lado derecho del árbol, donde podrían existir nodos terminales con sincronizaciones de alta calidad.

Una buena opción para reducir ese sesgo, es dividir el problema en un número n de subárboles originados equitativamente desde ambos sectores del árbol de decisión y obligar a que cada subárbol sea recorrido con la misma prioridad. Eso permite construir tempranamente ramificaciones en la zona derecha, que de otra manera solo serían visitadas luego de bastantes iteraciones y tiempo, considerando que el árbol cuenta con alrededor de 10^{12} ramas diferentes para un turno típico. En la Figura 4.1 se muestra un caso donde el problema se divide en $n = 8$ subárboles, posibilitando que con rapidez se comiencen a recorrer las ramas el sector derecho, tal como las ramas que descienden desde el nodo 0222, por ejemplo.

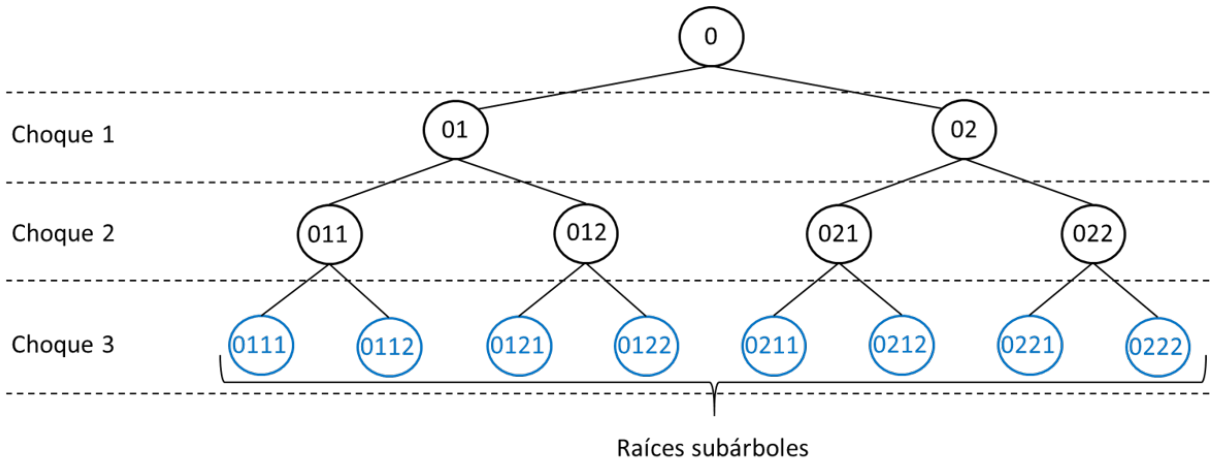

Figura 4.1: División de problema en 8 subárboles.

Sin embargo, a la hora de aplicar la división del problema, el actual método de ejecución en serie del algoritmo genera importantes dificultades, pues con la finalidad de ramificar a todos los subárboles con la misma prioridad, el recorrido se tendría que hacer descendiendo alternadamente de un nodo a la vez por cada subárbol. Por ejemplo, si se aplicara una ejecución en serie al caso mostrado en la Figura 4.1, para recorrer cada subárbol equitativamente se tendría que empezar construyendo la rama 001111 desde el primer subárbol, luego la 01121 desde el segundo subárbol y así sucesivamente, llegando a una lógica muy similar al recorrido en anchura, el que demostró empíricamente ser peor que el recorrido en profundidad dentro del problema de este trabajo.

Como la ejecución en serie no es una técnica eficiente cuando se divide el problema, se puede recurrir al método de paralelización explicado en el **Capítulo 2.5**. Para aplicarlo, el algoritmo debe ser programado de manera tal que cada uno de los n subárboles sea asignado a un thread, donde cada thread se ejecuta de forma independiente y recorre ramas del árbol completamente diferentes. En consecuencia, como los subárboles son entregados a threads independientes, todos son construidos simultáneamente de manera equitativa sin necesidad de hacer un recorrido en anchura.

En teoría, un computador puede ejecutar sin inconvenientes tantos subárboles en paralelo como subprocesos [1] tenga, pues desde ese punto en adelante el kernel deberá priorizar algunos threads por sobre otros dada las limitaciones de aquel hardware. En este caso se tiene a disposición un computador con 64 GB de memoria RAM y con dos procesadores Intel Xeon E5-2620 v4 de 8 núcleos y 16 subprocesos, sumando en total 16 núcleos y 32 subprocesos, así que el algoritmo debiese al menos poder ejecutarse con 32 subárboles en paralelo.

Además de asegurar que todos los subárboles se prioricen de forma igualitaria, la ejecución en paralelo tiene la ventaja de aprovechar la capacidad computacional disponible. Al ejecutar el algoritmo en serie utilizando el computador descrito en el párrafo anterior, se ocupa solamente el 10 % de la CPU y el 10 % de la memoria RAM, dejando un 90 % de ambas capacidades sin ser usadas, mientras que al ejecutar 32 threads en paralelo se utiliza el 100 % de la CPU y el 40 % de la memoria RAM. Como el número de subprocesos permite correr al menos 32 threads, el algoritmo visita una mayor cantidad de nodos del árbol en el mismo tiempo, generando una ganancia en eficiencia a medida que se aumenta el número de threads. De todas maneras, es preciso tener presente que dicha ganancia siempre está limitada por las pérdidas de rendimiento que existen producto de las partes no paralelizables del código, de la constante comunicación que debe existir entre los distintos threads y del número de subprocesos disponibles para la ejecución de los threads.

La cifra n de subárboles en los que será fraccionado el árbol debe ser determinado previo a realizar la ejecución, dentro del límite de capacidad del computador de modo que este no colapse o que no baje demasiado su rendimiento. Con el objetivo de que la división del árbol sea uniforme y considerando que se trata de un árbol binario, se establece que el valor de n debe ser una potencia de 2, por ejemplo 2, 4, 8, 16, 32, etc.

Dado que n es una potencia de 2, todos los nodos que son asignados como raíz de uno de los subárboles tienen igual profundidad (distancia entre el nodo y la raíz del árbol general). En

[1] Subprocesos: unidades de procesamiento en las que se divide el o los procesadores del computador.

efecto, la profundidad de las raíces de los subárboles respecto al árbol de decisión completo, es $\log_2 n$. Por ejemplo, si $n = 16$, la profundidad de la raíces es $\log_2 16 = 4$. En el caso de la Figura 4.1, el problema se divide en $n = 8$ subárboles que serán resueltos en paralelo, donde las raíces de los subárboles tienen profundidad $\log_2 8 = 3$.

Una vez que comienza a ejecutarse el programa, se construyen en anchura las primeras ramas del árbol hasta el nivel $\log_2 n$ y luego cada nodo de aquel nivel pasa a ser la raíz de uno de los n subárboles. Todos los threads recorren solamente las ramas del subárbol respectivo, bajo las mismas reglas de recorrido en profundidad y hacia la izquierda explicadas en el **Capítulo 3.2.2**.

4.1.1. Solución del Árbol con Paralelización

Cada thread construye exclusivamente las ramas del subárbol respectivo, de manera que todos llegan a diferentes nodos terminales, los que pueden tener distintas soluciones factibles. Como se quiere obtener solo una solución para el problema, no tiene sentido que los threads realicen la búsqueda de forma totalmente independiente, de modo que se trabaja con un incumbente en común para todos ellos. Ese valor se cambia a medida que se encuentran mejores soluciones completas factibles dentro de los n subárboles, sujeto al criterio de actualización explicado en el **Algoritmo 3**. Es importante mencionar que el incumbente en común es utilizado en la condición para la poda de ramas, lo cuál hace aún más eficiente el método de resolución en paralelo porque los subárboles obtienen cotas no solo desde sus ramas, sino que desde todos los subárboles.

Finalmente, la solución del problema puede provenir desde cualquiera de los n subárboles recorridos y se escoge la mejor sincronización encontrada hasta el momento en que se cumple alguna condición de detención del algoritmo, es decir, se elige el incumbente vigente al finalizar la ejecución del programa.

4.2. Método de Sondajes

La división del problema en subárboles y su ejecución en paralelo significa un gran avance en la reducción del sesgo que arrastra la lógica de recorrido en profundidad aplicada en este algoritmo. A pesar de aquello, cada subárbol todavía tiene una dimensión demasiado grande como para poder recorrerlo completo en un tiempo razonable, lo que revive la problemática inicial del sesgo que manifiestan las soluciones, dado que con baja probabilidad se llega a visitar las ramas de la zona derecha del subárbol.

Asumiendo que se escoge un número n de subárboles a ejecutar en paralelo de modo tal que se utiliza el máximo de la capacidad computacional, sería ineficiente seguir dividiendo aquellos subárboles para crear más threads, pues el kernel del PC deberá priorizar algunos subárboles por sobre otros. Entonces, se requiere indagar en otra solución para mitigar ese sesgo.

A partir de esa necesidad se desarrolla un método de recorrido más equitativo dentro de los subárboles, el que fue denominado bajo el nombre de sondajes. Este inicia con la determinación de un nivel m del árbol, que será el nivel desde el cuál se realizarán los sondajes y que debe cumplir con la restricción $m > \log_2 n$, para que puedan ser hechos en niveles más bajos que la raíz de los n subárboles a ejecutarse en paralelo. Luego, se comienza a construir cada subárbol de forma normal mediante el recorrido en profundidad y hacia la izquierda, hasta llegar al primer nodo del nivel m. Desde aquel nodo se realiza el primer sondaje, el que consiste en recorrer el subárbol según las mismas reglas, pero solo hasta que se completen k nodos visitados, número que se define dependiendo de la magnitud del turno a evaluar de manera que permita obtener un número razonable de soluciones factibles dentro de cada sondaje. En ese momento se da por concluido el primer sondaje y se pasa al siguiente, el que empieza desde el segundo nodo del nivel m del respectivo subárbol. De ese modo se sigue iterando sucesivamente, siempre realizando k iteraciones por cada sondaje.

En la Figura 4.2 se muestra un ejemplo donde el problema se divide en $n = 2$ subárboles a recorrer en paralelo y se fija el nivel $m = 3$ para aplicar los sondajes. El primer sondaje del subárbol 1 se hace desde el nodo 0111, recorriendo k nodos a partir de ese punto, siempre respetando el método de recorrido en profundidad hacia la izquierda. Una vez completado, se pasa al segundo sondaje que parte desde el nodo 0112, iterando k veces nuevamente. Los mismo ocurrirá con los siguientes 2 sondajes del subárol 1 y con los 4 sondajes del subárbol 2, el que se ejecuta en paralelo.

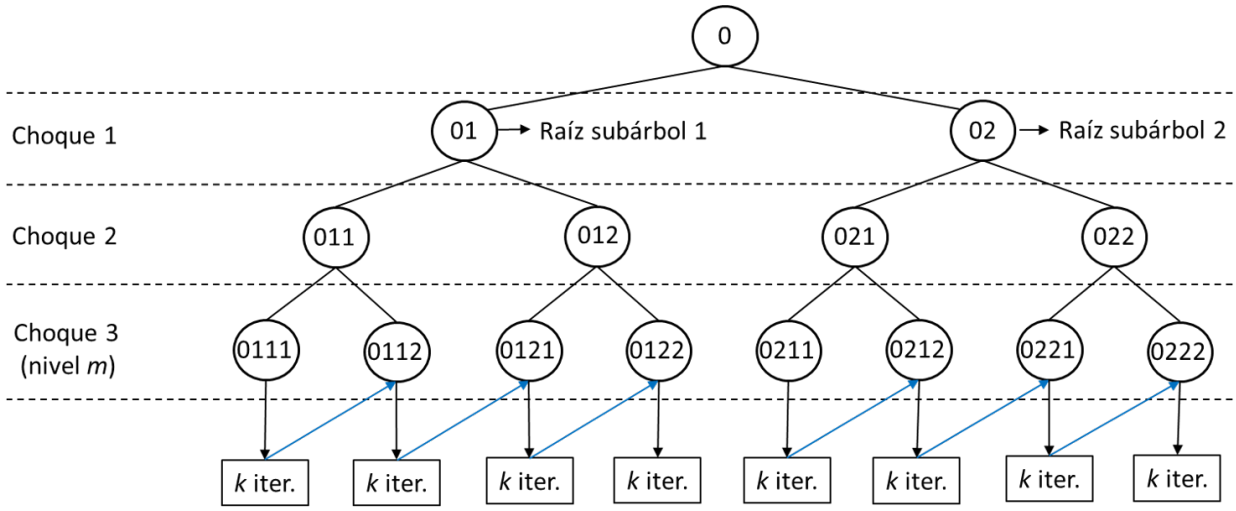

Figura 4.2: Sondajes de k iteraciones realizados a partir del nivel $m = 3$ en un problema ejecutado en paralelo con $n = 2$ subárboles.

Esta técnica ayuda a reducir aún más el sesgo que afecta al algoritmo, permitiendo que no se gaste tanto tiempo recorriendo solo una zona específica del subárbol, sino que se construya un número limitado de nodos en cada zona y que luego el recorrido cambie de sector, explorando nuevas ramas que podrían contener buenas soluciones factibles del problema.

Capítulo 5

Resultados

Este capítulo muestra la validación y calibración del modelo desarrollado en este trabajo, además de los principales resultados conseguidos al aplicarlo. Para realizar los análisis se utilizan dos casos de estudio: un turno laboral real pasado y un caso creado con el objetivo de comprobar una hipótesis teórica. En el primero se compara el número de vueltas logradas bajo la coordinación del personal de la mina versus las completadas gracias a la sincronización propuesta por el algoritmo para dicho turno, mientras que en el segundo se simula un turno sin ningún tipo de fallas para evaluar si es posible alcanzar el máximo teórico de 40 vueltas dentro de un turno de 8 horas.

Es importante destacar que el algoritmo fue puesto a prueba en un mayor número de casos, tanto reales como hipotéticos, pero solo se detallarán los dos antes mencionados, pues a partir de aquellos casos se obtienen resultados de especial interés para esta tesis.

Para ejecutar el algoritmo fue utilizado un computador con 64 GB de memoria RAM y con dos procesadores Intel Xeon E5-2620 v4 de 8 núcleos y 16 subprocesos. Dada la gran capacidad computacional disponible, se aplicó el método de resolución en paralelo.

5.1. Calibración y Validación del Modelo

5.1.1. Calibración

El modelo de optimización creado en esta tesis se caracteriza por utilizar una red ferroviaria discreta conformada por 37 vértices, los que plasman la totalidad de los segmentos de vía pertenecientes a la red original. Todos aquellos vértices tienen asociado un tiempo de traslado de los trenes en ambos sentidos de movimiento, el que refleja los minutos de desplazamiento para los trenes a través de los tramos de la red original representados por esos vértices.

Los tiempos de traslado de cada vértice fueron determinados mediante un trabajo con los datos recogidos por los sistemas que supervisan a los trenes de la etapa de transporte

principal. En primer lugar, se obtuvo un tiempo promedio de desplazamiento para todos los tramos asociados a cada vértice de la red modificada. Luego, se realizó lo mismo con los minutos requeridos en el proceso de carga de mineral. Finalmente, se calculó el promedio de los tiempos de descarga, diferenciando si la descarga se lleva a cabo en buzones para chancado primario o secundario.

A pesar de que los tiempos fueron extraídos desde los sistemas de información oficiales de la mina, la modificación de la red ferroviaria y potenciales errores en las cifras o en los cálculos, podrían derivar en ciertas diferencias entre los tiempos obtenidos y los tiempos de traslado reales, lo que eventualmente se puede traducir en que dentro del modelo los trenes se desplacen más rápido o más lento, perdiendo calidad y precisión en la herramienta desarrollada. Dado eso, con el objetivo de ajustar a la realidad los tiempos asociados a los vértices de la red férrea usada por el algoritmo, se debe realizar una calibración de aquellos números.

Para efectuar la calibración de dichos parámetros, se trabajó con la primera vuelta de 10 turnos diferentes. Además, se utilizó como herramienta de apoyo los videos históricos que se almacenan en los sistemas de la mina, los que muestran sobre un layout de la red el desplazamiento de los trenes en cada instante para cualquier turno. De esa manera, se ejecutó el algoritmo en cada una de las 10 vueltas configurando en los parámetros iniciales del modelo todas sus condiciones reales, tales como entregas en piques y disponibilidad de las líneas descarga. Posteriormente se comparó la herramienta versus los videos, mirando tramo a tramo el traslado de los trenes, los minutos gastados en los procesos de carga y los utilizados en la etapa de descarga.

Una vez finalizada la comparación, se identificaron los procesos y vértices de la red donde se producían mayores inconsistencias entre el modelo y los videos, las que variaban entre 0,5 y 2 minutos aproximadamente, procediendo a ajustar aquellos tiempos y así fijarlos en valores más realistas.

5.1.2. Validación

Además de calibrar los principales parámetros, correspondientes a los tiempos empleados en el desplazamiento por los vértices y en los procesos de carga y descarga, con la finalidad de verificar si el algoritmo entrega soluciones razonables y ajustadas a la realidad, es sumamente necesario validar el desempeño de la herramienta, para que así el modelo sea confiable y pueda ser aplicado por los despachadores de la mina. Claramente de nada sirve llegar a muy buenas soluciones, pero obtenidas bajo supuestos y parámetros demasiado optimistas. Para hacer la validación, se requiere evaluar un turno pasado donde se disponga de toda la información relevante, incluyendo el registro de fallas y situaciones especiales. El caso elegido corresponde al turno A del 02/06/2017, ya que se cuenta con la totalidad de los datos y carece de la ocurrencia de eventos fuera de lo común.

El método de validación aplicado consiste en replicar con exactitud las condiciones y parámetros del turno dentro del algoritmo, para luego comparar durante un intervalo de tiempo determinado el resultado de la herramienta versus el resultado original. En este caso se optó

por analizar las dos primeras vueltas del turno, en las cuáles los trenes se desplazan generalmente en caravana, de modo que se produce una baja cantidad de "choques" y así fácilmente se pueden comparar las soluciones de ambos casos. El hecho de que las posibilidades de decisión sean muy acotadas durante ese período ayuda a efectuar una buena evaluación, pues de lo contrario se llegaría a un elevado número de interferencias entre trenes y el algoritmo podría encontrar una solución muy distinta a la del despachador, lo que haría los dos resultados incomparables para efectos de la validación.

Como parámetros se utilizaron los tiempos de desplazamiento, carga y descarga de mineral ya calibrados y se ingresaron todas las situaciones particulares del turno, especialmente los tiempos de carga y descarga que demoraron más de lo habitual debido a problemas en los piques principales y en los chancadores, respectivamente. El turno estudiado inicia a las 6:51 del día 02/06/2017 y al cabo de 2 horas y 25 minutos (a las 9:16) se realiza la comparación de los escenarios, momento en que se desarrolla la segunda vuelta de los trenes.

En la Figura 5.1 se muestra la posición de los trenes dentro del túnel principal y de la mina luego de 2 horas y 25 minutos, según las decisiones tomadas por el despachador de trenes del turno. Por otra parte, en la Figura 5.2 está ilustrada la distribución de los trenes dentro de la red producto de la solución escogida por el algoritmo para el mismo período. En las dos figuras se muestran con color azul los nombres de todos los trenes que se encuentran descargados y que se dirigen hacia un pique principal, mientras que en color rojo aparecen los nombres de los trenes cargados de mineral que se desplazan hacia el sector de descarga.

Al analizar ambas distribuciones de trenes, se observa que después de casi 2,5 horas de iniciado el turno el modelo llega a una solución muy parecida a lo ocurrido realmente. 6 trenes mantienen idéntica posición y dirección, donde las excepciones son el tren T4 y el tren T6, ya que el algoritmo tomó una decisión diferente debido a un potencial "choque" que podría haber ocurrido entre ellos al interior de la mina.

Esta similitud entre los resultados luego de las dos primeras vueltas permite afirmar que el algoritmo de optimización desarrollado se ajusta bastante a la realidad, que trabaja con tiempos de desplazamientos realistas y que toma decisiones razonables y aplicables en la operación diaria. Por lo tanto, su uso por los despachadores de trenes es totalmente factible y asegura soluciones que pueden ser ejecutadas en la práctica.

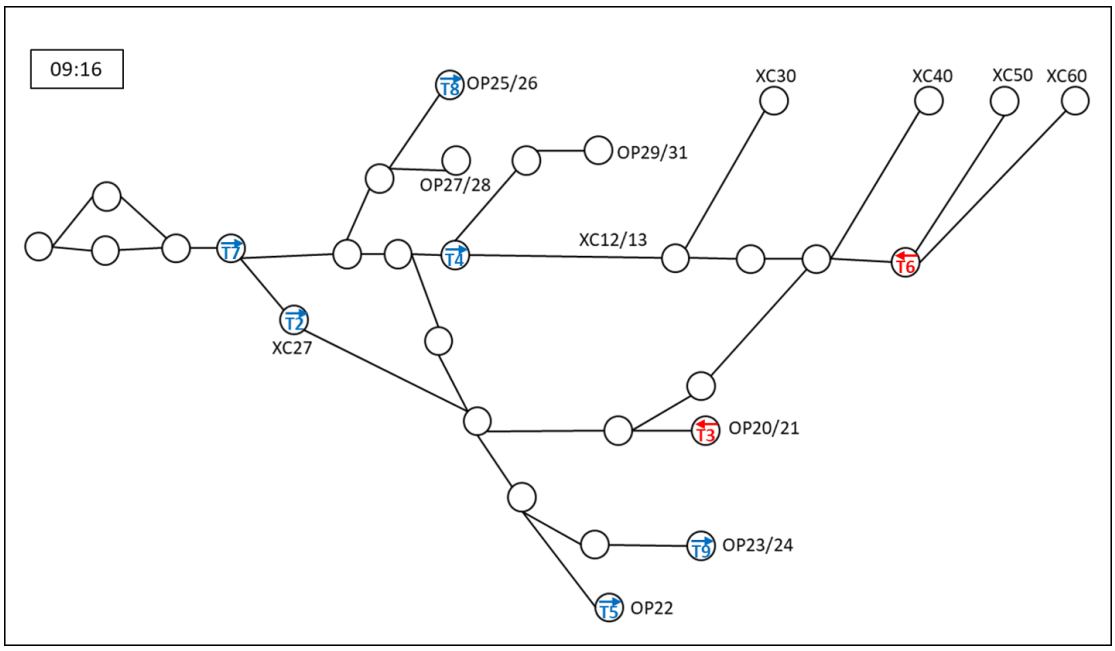

Figura 5.1: Solución del despachador de trenes para un intervalo de 2 horas y 25 minutos del turno A 02/06/2017.

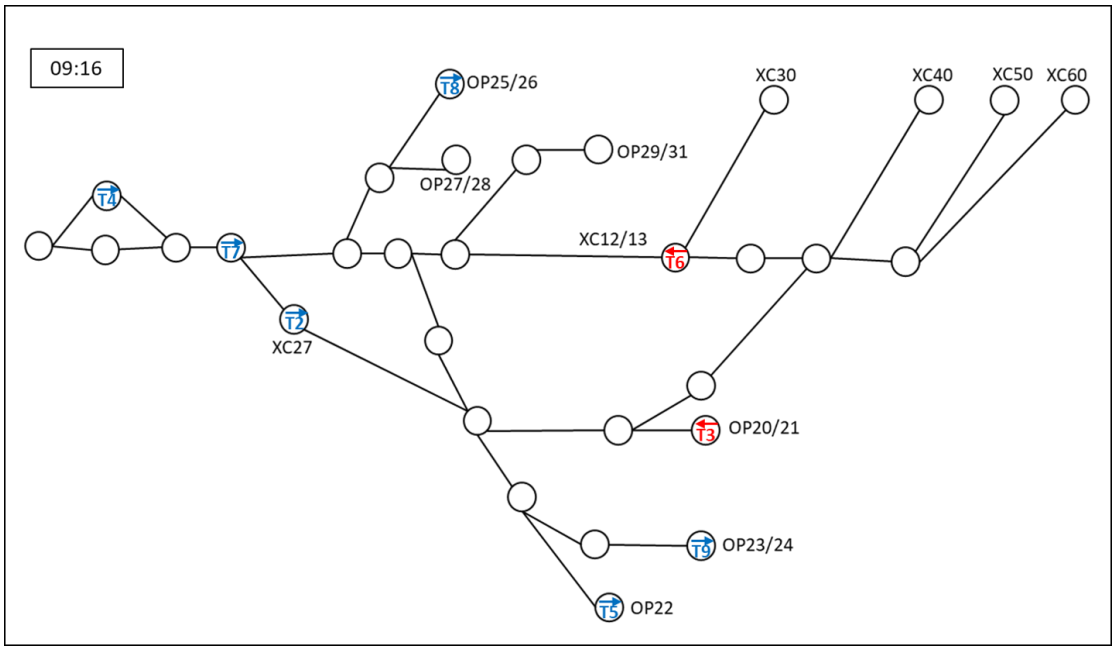

Figura 5.2: Solución del algoritmo para un intervalo de 2 horas y 25 minutos del turno A 02/06/2017.

5.2. Elección del Número de Threads a Ejecutar

El algoritmo desarrollado para sincronizar a los trenes de la mina se basa en la construcción de un árbol de decisión binario que se va ramificando cada vez que se produce un "choque" de trenes. Gracias al método de paralelización el problema puede ser dividido en varios

subárboles, donde cada uno es asignado a un thread que se ejecuta de manera paralela en el procesador del PC, aprovechando la capacidad computacional y estableciendo un recorrido más homogéneo a través del árbol de decisión.

Teóricamente se pueden correr sin inconvenientes tantos threads como subprocesos tenga el procesador del computador utilizado, por lo tanto, la cantidad óptima de threads a ejecutar depende principalmente de las características del procesador disponible. Para este trabajo se tiene un PC con 64 GB de memoria RAM y dos procesadores, los que juntos suman un total de 16 núcleos y 32 subprocesos. Dados los recursos del hardware, se deberían ejecutar perfectamente 32 threads de manera eficiente, lo que no necesariamente se tiene que cumplir, pues la asignación de subprocesos a threads por parte del kernel y la constante comunicación que debe existir entre los threads son factores que limitan los incrementos de eficiencia a medida que aumenta la cantidad de threads que se corren en paralelo, tal como se explicó en el **Capítulo 2.5.1**.

Entonces, para determinar con certeza el número óptimo de threads a ejecutar en aquel computador específico, es necesario verificar experimentalmente ese supuesto teórico. Teniendo en cuenta que la cifra debe ser una potencia de 2 por la naturaleza binaria del árbol, se ejecutó 8 veces el algoritmo aplicando 1, 2, 4, 8, 16, 32, 64 y 128 threads, donde cada corrida tuvo una duración de 1 hora.

5.2.1. Utilización del Hardware

En primer lugar, es interesante evaluar la utilización efectiva que se les da a los hardwares del PC. Con el fin de analizar ese aspecto, en el gráfico de la Figura 5.3 se muestra el porcentaje de la CPU y de la memoria RAM que se mantuvo en uso durante la ejecución del algoritmo para los distintos números de threads. Hay que considerar que en su estado inicial, donde no se ejecutaron softwares externos al sistema operativo, el PC se encontraba con su CPU totalmente disponible, pero con un 8 % de utilización en la memoria RAM.

Desde el gráfico se observa que a medida que aumenta el número de threads ejecutándose en paralelo, incrementa proporcionalmente el uso de ambos hardwares. La CPU alcanza su capacidad máxima con 32 threads, debido a que es la cantidad de subprocesos con los que cuenta el procesador del computador. Desde ahí en adelante se mantiene una utilización del 100 %, pero el kernel del sistema operativo debe establecer ciertos criterios para priorizar algunos threads por sobre otros. En cuanto a la memoria RAM, solo con 128 threads se utiliza completamente el recurso y probablemente para esa cantidad el kernel tiene que gestionar la asignación del hardware a los threads dado que es requerido por sobre su disponibilidad. Notar que en realidad para 128 threads el gráfico muestra un uso del 98 % de la memoria RAM, pero esto equivale a un 100 % de la RAM utilizable, pues el PC deja un 2 % de memoria libre para darle otro tipo de uso.

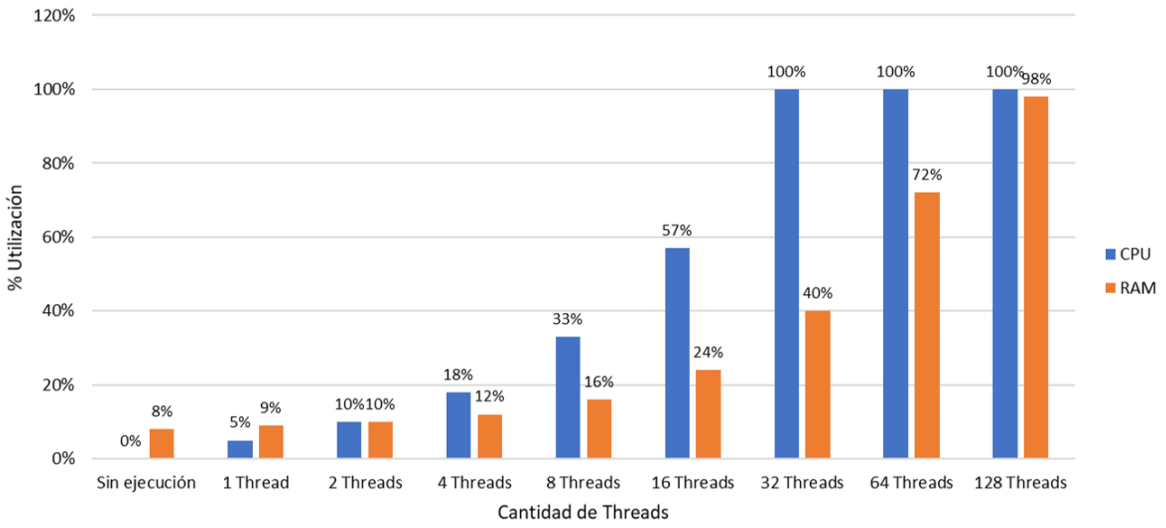

Figura 5.3: Porcentaje de uso de la CPU y de la memoria RAM para distintas cantidades de threads ejecutándose en paralelo.

5.2.2. Rendimiento del Algoritmo

A partir de la Figura 5.3 se pudo ver que efectivamente la capacidad computacional llega a su límite máximo cuando se corren en paralelo 32 threads del programa, punto en el que existe un emparejamiento perfecto entre los threads y los subprocesos de los que disponen los procesadores. Sin embargo, eso no asegura que dicha cifra sea la cantidad óptima de threads que se debe ejecutar para lograr un mejor rendimiento del algoritmo, por lo tanto, se requiere medir algún indicador de desempeño de la herramienta.

Una métrica que cumple con dicho objetivo es la cantidad de nodos del árbol de decisión recorridos durante el período de ejecución, es decir, a lo largo de una hora. En la Figura 5.4 se muestra un gráfico con el número de nodos creados por hora para las distintas cantidades de threads, donde se observa que la mayor rapidez de recorrido se obtiene al ejecutar 32 threads en paralelo, con una tasa de un poco más de 2 millones de nodos construidos por hora.

Además, inicialmente en el gráfico es posible ver que mientras sube el número de threads también lo hace la tasa de recorrido de nodos, lo que se cumple hasta los 32 threads. Luego, la velocidad de creación disminuye a pesar de aumentar la cantidad de threads en ejecución. Eso se justifica porque existe una sobrecarga de requerimientos de hardware, pues se necesitan más subprocesos que los disponibles, así que el kernel debe gastar recursos para constantemente estar decidiendo cómo priorizar algunos threads en desmedro de otros.

Estas cifras permiten afirmar que para el computador descrito previamente, la cantidad óptima de threads a ejecutar en correr es de 32, donde aquel número asegura una velocidad de recorrido a través de las ramas del árbol de decisión mayor que la lograda con otro número de threads.

Una vez hecho ese análisis, también es importante revisar la cantidad de nodos creados en promedio por cada thread en ejecución durante una hora, lo que se puede ver en la Figura

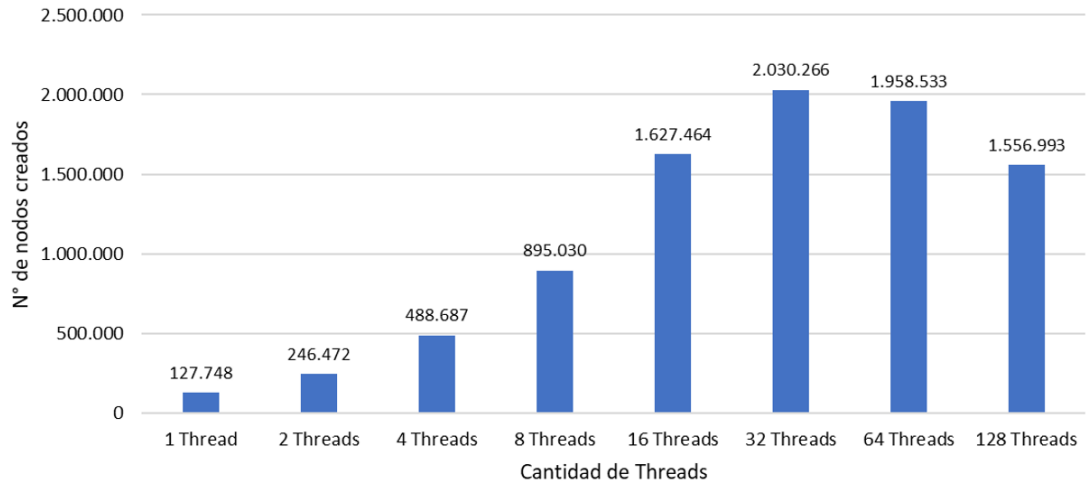

Figura 5.4: Cantidad de nodos del árbol de decisión creados durante una hora para ejecuciones con distinto número de threads, donde cada barra representa una ejecución para la respectiva cantidad de threads.

5.5. En aquel gráfico se observa que la velocidad promedio de creación de nodos por thread siempre disminuye si se aumenta el número de threads. A pesar de que en su conjunto la ejecución con 32 threads sea más eficiente, individualmente cada uno de esos 32 threads es más lento que el único thread de la ejecución con 1 solo thread, por ejemplo.

Lo anterior se explica porque a medida que aumenta la cantidad de threads, también sube la complejidad de la comunicación que debe existir entre todos los threads que están corriendo en paralelo, lo que absorbe una porción importante de los recursos computacionales y también establece un cuello de botella que disminuye la velocidad de recorrido de cada thread.

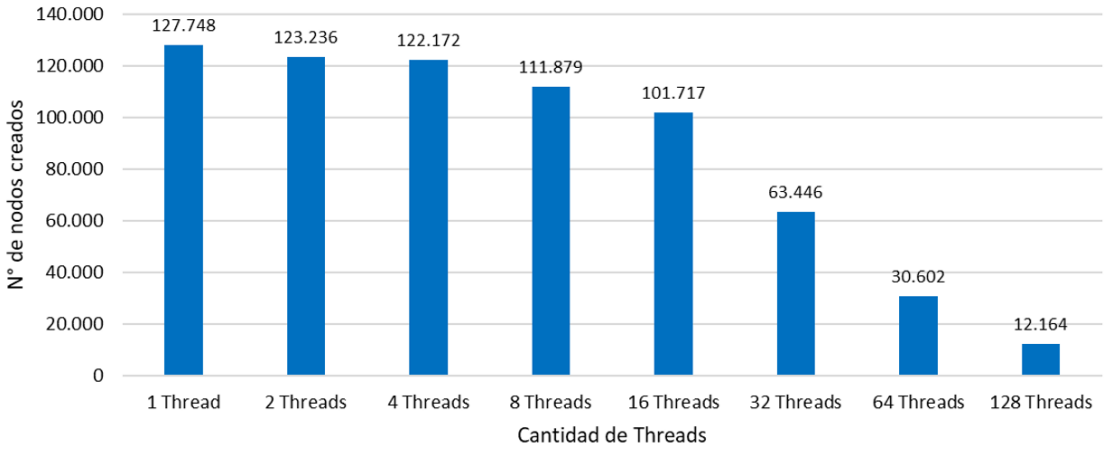

Figura 5.5: Cantidad promedio de nodos creados del árbol de decisión por cada thread durante una hora para ejecuciones con distinto número de threads, donde cada barra representa una ejecución para la respectiva cantidad de threads.

5.3. Caso de Estudio: Turno C 23/01/2017

Con la finalidad de medir el funcionamiento y los resultados del algoritmo, se tomó un caso de estudio que cumpliera con ciertas características que permitieran realizar un completo análisis de la herramienta desarrollada en esta tesis. Específicamente, el objetivo es evaluar si la aplicación del algoritmo para un caso real logra entregar una solución con un mayor numero de vueltas acarreadas durante el turno respecto a lo conseguido por el despachador de trenes bajo las mismas condiciones y variables originales, además de verificar si la rapidez de la ejecución hace factible o no su uso práctico en la mina.

El caso fue escogido desde un grupo de turnos pasados, los que se desarrollaron entre agosto del 2016 y marzo del 2017, pues para ese período se dispone de toda la información necesaria para construir los inputs de la herramienta. Luego, a partir de aquel grupo se elaboró una lista con turnos catalogados como candidatos ideales para el estudio, cuya principal característica es que la etapa de transporte principal fue el cuello de botella del proceso, de manera que quedaron entregas de viajes sin ser acarreadas durante el turno y que en consecuencia, existía la posibilidad teórica de realizar un mayor número de viajes mediante una mejor coordinación de los trenes.

Para que los turnos del listado cumplieran con dicha característica, se establecieron las siguientes condiciones y requisitos:

- Una o más entregas de viajes disponibles durante el turno que no hayan sido acarreadas. El respectivo pique principal de la entrega debe haber permanecido sin desperfectos técnicos.
- 30 o más vueltas completadas en total durante el turno real.
- Cantidad de viajes concretados menor que el número previamente planificado para el turno.
- Al menos un chancador primario y uno secundario en funcionamiento durante el 100 % del turno.
- Ambos chancadores primarios y ambos chancadores secundarios funcionando por lo menos el 40 % del turno.
- Sin observaciones en el registro de novedades del transporte principal.

De la lista de candidatos fue seleccionado el turno C del día 23 de enero del 2017, el que se llevó a cabo entre las 23:46 del día anterior (22 de enero) y las 07:00 horas.

5.3.1. Parámetros

Previo a ejecutar el algoritmo es necesario ingresar una serie de parámetros, los que se dividen en dos tipos: en primer lugar está la información exclusiva del turno, como las entregas en piques o disponibilidad de los chancadores, y por otra parte están aquellos datos generales que son estáticos y que conservan las mismas cifras para cualquier turno. A continuación se detallan los valores para ambas clases de parámetros:

Parámetros propios del turno

- **Hora de partida:** los trenes comienzan a ingresar al túnel en dirección mina a las 23:46 del día 22/01/2017.
- **Trenes:** 8 trenes en funcionamiento, 2 de mineral fino (T2 y T3) y 6 de mineral grueso (T4, T5, T6, T7, T8 Y T9). A excepción del T5 que comienza a trabajar a las 02:15, el resto de los trenes parten con normalidad a las 23:46.
- **Posición inicial de trenes:** los 8 trenes se distribuyen inicialmente en la zona de descarga. El T9 comienza en la zona de los chancadores primarios descargando una entrega pendiente del turno anterior, mientras que el T5 parte el turno en mantenimiento, lo que se extiende hasta las 02:15. Los 6 trenes restantes inician descargados y listos para empezar a acarrear las entregas del turno
- **Viajes programados:** se programaron 35 viajes para el turno, 10 de mineral fino y 25 de mineral grueso.
- **Viajes realizados:** los trenes realizaron 32 viajes, 9 de mineral fino y 23 de mineral grueso.
- **Asignación de trenes a entregas:** se utiliza la misma asignación de trenes a entregas en piques principales determinada por el despachador del turno.
- **Entregas sin acarrear:** 3 entregas de mineral grueso sin ser cargadas durante el turno.
- **Chancadores primarios:** 85 % del turno con los 2 disponibles y 15 % con solo uno funcionando (chancador primario número 2).
- **Chancadores secundarios:** 46 % del turno con ambos trabajando y 54 % con un único chancador.
- **Situaciones especiales:** en la primera vuelta del T9 el proceso de carga de mineral demora 40 minutos, cifra mayor que lo habitual, así que se considera ese antecedente dentro de los parámetros. Respecto a la red ferroviaria y a los piques principales, no presentaron inconvenientes durante el desarrollo del turno.

Parámetros generales

- **Tiempo de carga:** 18 minutos para que un tren cargue una entrega.
- **Tiempo de descarga:** los trenes de mineral fino descargan en 10 minutos y los de grueso lo hacen en 12 minutos.
- **Hora límite de ingreso al túnel:** con el objetivo de que el cambio del personal que opera los trenes se realice sin atrasos para el siguiente turno, durante un turno C ningún tren puede ingresar al túnel en dirección mina luego de las 06:05.
- **Velocidad de trenes:** la velocidad a la que se desplazan los trenes en cada tramo de la red para ambos sentidos de movimiento se determinó basándose en datos históricos, números que luego fueron verificados y calibrados.

5.3.2. Ejecución del Algoritmo y Resultados

Una de las principales características que permitieron que este caso fuese escogido para ser estudiado, es que en la realidad se completaron 32 vueltas en 434 minutos (desde las 23:46

hasta las 07:00), quedando 3 entregas de mineral grueso sin ser acarreadas. Esas entregas representan una gran oportunidad de lograr una o más vueltas adicionales mediante una mejor sincronización de los trenes, que es precisamente el objetivo del algoritmo. Como en este trabajo se aplica la misma asignación de trenes a entregas de viajes en los piques principales que la escogida por el despachador del turno y considerando que las entregas sin acarrear no les fueron asignadas a trenes durante el turno en estudio, se decidió otorgárselas a los trenes T6 y T8, que hicieron 4 viajes cada uno, y al T9 que consiguió dar 3 vueltas, números menores en relación al máximo de 5 viajes que puede realizar un solo tren dentro de un mismo turno.

La ejecución del algoritmo se realizó corriendo 32 threads en paralelo a lo largo de una hora, pues en el computador utilizado esa es la cantidad de threads con la que se alcanza una mayor velocidad de recorrido del árbol de decisión y que asegura construir un poco más de 2 millones de nodos en ese período.

Inicialmente, con el objetivo de posteriormente obtener un gap de la solución, se estimó una cota superior para el óptimo del problema según el procedimiento expuesto en el **Capítulo 3.3.3**, la que indicó que como máximo se podrían conseguir 34 vueltas en un tiempo mínimo de 432 minutos. Luego se comenzó con la ejecución de la instancia, la que se realizó durante una hora, intervalo de tiempo donde el incumbente (mejor solución factible para el turno completo encontrada hasta el momento) se fue actualizando constantemente a medida que alguno de los 32 threads llegaba a una mejor solución. La evolución del número de vueltas logradas por el incumbente se muestra en el gráfico de la Figura 5.6, el que además contiene la cota superior estimada para la cantidad de vueltas.

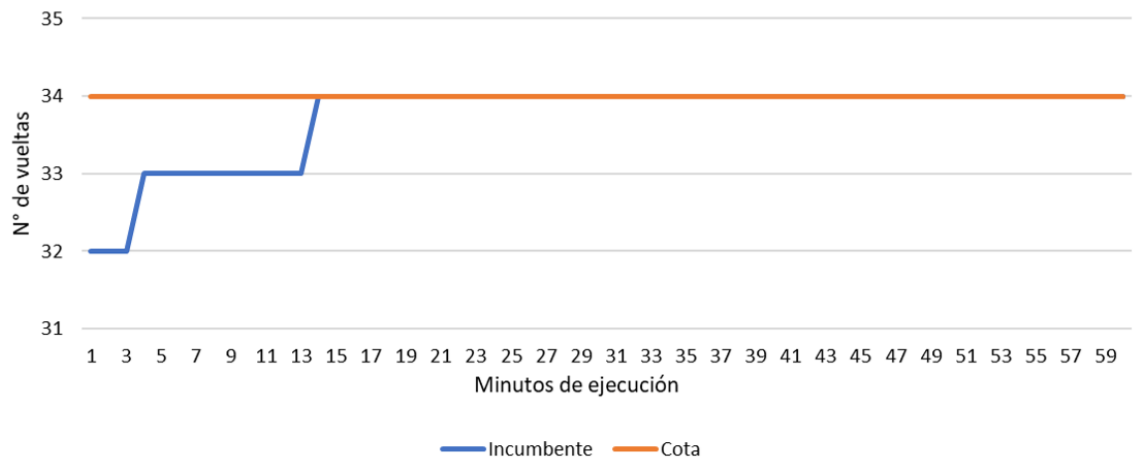

Figura 5.6: Evolución del incumbente para la cantidad de vueltas completadas versus la cota máxima para el mismo indicador.

En el gráfico se observa que al comienzo de la ejecución el algoritmo encuentra rápidamente una solución que iguala la cantidad de viajes logrados mediante la coordinación del despachador de trenes. Transcurridos 4 minutos, se llega a una sincronización de trenes factible en la que se completan 33 vueltas, superando en un viaje el desempeño original. Finalmente, luego de 14 minutos desde iniciada la corrida se encuentra una solución con 34 vueltas, logrando realizar 2 viajes más que en la realidad y alcanzando la cota superior establecida para el caso.

De esa manera, en menos de 15 minutos el algoritmo pudo obtener una solución cuyo gap es cero en cuanto al principal indicador que es la cantidad de vueltas completadas.

Si bien en el minuto 14 de iniciada la corrida el incumbente alcanza la cota máxima para el número de viajes, durante los 46 minutos restantes se producen algunas variaciones en el segundo indicador de la solución, correspondiente al tiempo necesario para que los trenes completen las 34 vueltas. En la Figura 5.7 se ilustra la evolución del incumbente en cuanto a dicho indicador, donde el tiempo se mide desde el comienzo del turno (23:45) hasta que el último tren finaliza el vaciado de mineral en los buzones de descarga.

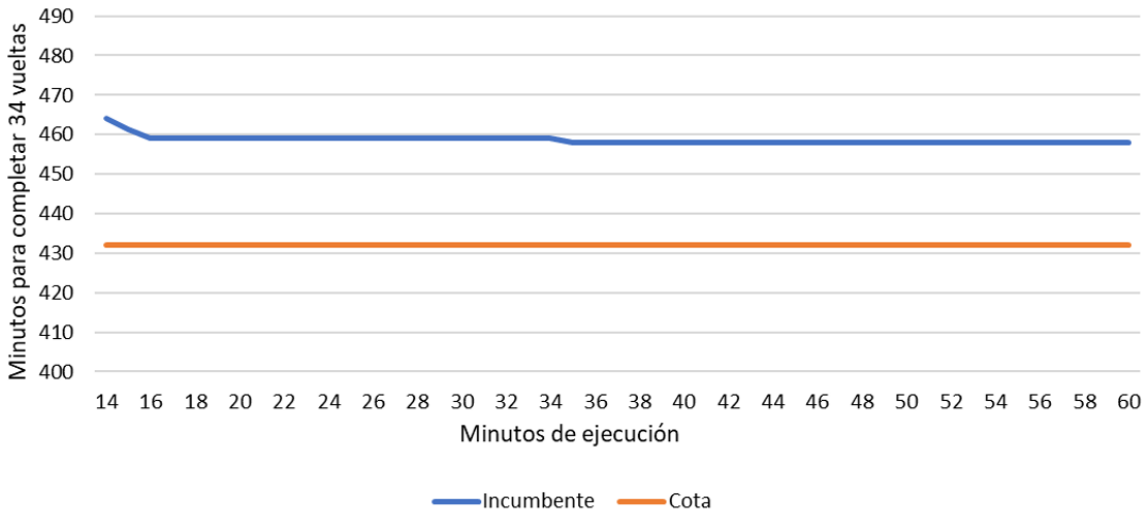

Figura 5.7: Evolución del incumbente para el tiempo necesario en completar las 34 vueltas durante el turno.

La primera solución de 34 vueltas, encontrada a los 14 minutos de ejecución, logra completar esa cantidad de viajes en 464 minutos. Dos minutos de ejecución más tarde se llega a una sincronización con la misma cantidad de vueltas y con un tiempo de 459 minutos. Finalmente, a los 35 minutos de ejecución se llega a la solución definitiva de 458 minutos donde el último tren termina la descarga de mineral a las 07:24, tiempo que no mejora antes de completar una hora desde el comienzo de la corrida. En la imagen también se muestra la cota del problema cuyo valor es de 432 minutos, lo que significa que el problema tiene un gap de 26 minutos en el indicador que mide el tiempo requerido para completar las 34 vueltas.

Durante la hora en que el algoritmo corrió con 32 threads, se crearon alrededor de 2 millones de nodos. Tal como se muestra en la Figura 5.8, del total de nodos creados el 41,5 % fueron podados debido a que la proyección de sus nodos hijos estimaba un peor resultado que el incumbente, el 12,8 % correspondían a nodos con decisiones infactibles y solo el 2 % de los nodos construidos llegó a una solución factible del problema. El 43,7 % restante representa a los nodos de niveles intermedios que pudieron ramificarse.

Para tener una idea de la fracción del árbol que fue recorrida durante los 60 minutos, en primera instancia se estimó el tamaño total del árbol de decisión. Basándose en el nivel promedio en el que se ubicaban los nodos terminales con soluciones factibles, donde en particular la mejor solución encontrada pertenece al nivel 46, se llegó a que una buena aproximación

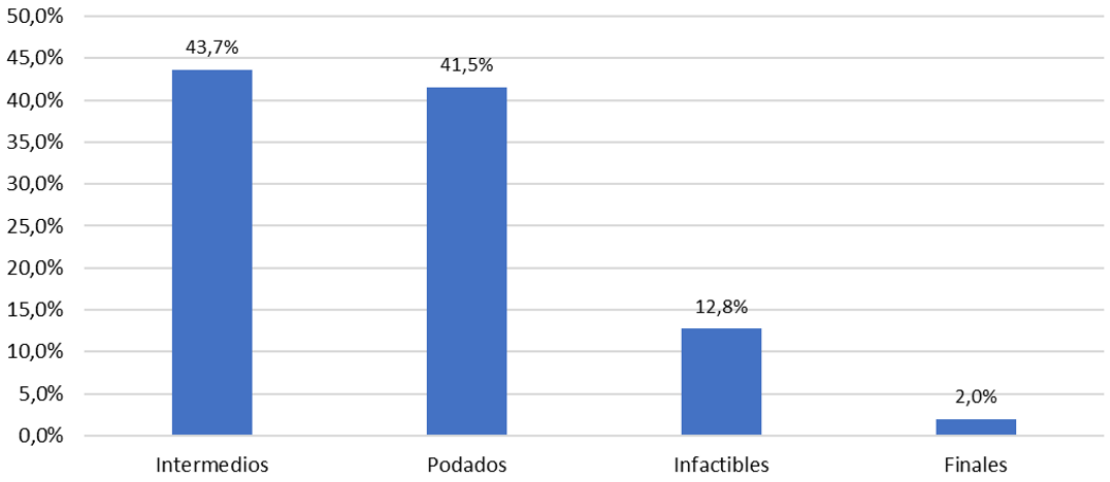

Figura 5.8: Distribución porcentual de los nodos del árbol de decisión creados durante una hora de ejecución según tipo.

es suponer que el árbol tiene una profundidad de 50 niveles. Dada esa cifra, se estima que el árbol completo cuenta con $\sum_{i=0}^{50} 2^i \approx 2,25 \times 10^{15}$ nodos. Debido a que el rendimiento del algoritmo al ejecutarse con 32 threads es de 2.030.266 nodos creados por hora, tal como se mostró en la Figura 5.4, el tiempo estimado necesario para recorrer completamente los $2,25 \times 10^{15}$ nodos es de alrededor de 126.000 años, sin considerar que la cantidad de nodos reales puede ser más baja debido a la existencia de nodos del árbol infactibles y al método de poda de ramas que descarta del recorrido cierto porcentaje de subárboles.

Teniendo en consideración ese número de nodos, en la Figura 5.9 se muestra la evolución del porcentaje del árbol recorrido a lo largo de una hora. Ahí se observa que rápidamente se descarta cerca del 60 % de la cifra estimada, lo que se explica por las ramas infactibles encontradas en los primeros niveles del árbol, pues si aparece un nodo infactible en el nivel 2, por ejemplo, se descartan inmediatamente todos sus descendientes hasta el nivel 50, es decir, la mitad del árbol calculado. Es posible notar en el gráfico que en los primeros minutos no existe una mayor evolución en el recorrido del árbol, dado que el incumbente inicial es de 32 vueltas y luego sube a 33, por lo tanto, la tasa de poda aun es baja. Sin embargo, a partir del minutos 14, que es donde se encuentra la primera solución de 34 vueltas, el porcentaje del árbol revisado sufre importantes alzas, lo que se debe tanto a nodos infactibles como a un incremento considerable en la tasa de poda gracias a un incumbente más exigente, el que permite podar en niveles superiores y así descartar en cada poda un mayor porcentaje del árbol. En la última media hora de ejecución la fracción del árbol recorrido se mantiene con un alza relativamente constante dado que el incumbente no sufre grandes modificaciones.

En definitiva, al cabo de una hora se recorre aproximadamente el 67 % del árbol estimado, lo que es una cantidad alta considerando la magnitud del árbol total.

Si se quiere tener una idea cuánto tiempo más se requiere para recorrer el 100 % del árbol estimado, se puede tomar la evolución del porcentaje recorrido entre el minuto 59 y 60 y proyectarlo como una tasa constante hacia el futuro. Para este caso dicha tasa es de 0,089 % del árbol recorrido durante un minuto, por lo tanto, para completar el 33 % restante se

necesitarían 371 minutos, es decir, 6 horas y 11 minutos más de ejecución. A pesar de que el tiempo estimado es bajo en comparación a los 126 mil años proyectados inicialmente, la tasa de recorrido podría ir disminuyendo posteriormente, lo que aumentaría el tiempo requerido para recorrer la totalidad del árbol de decisión.

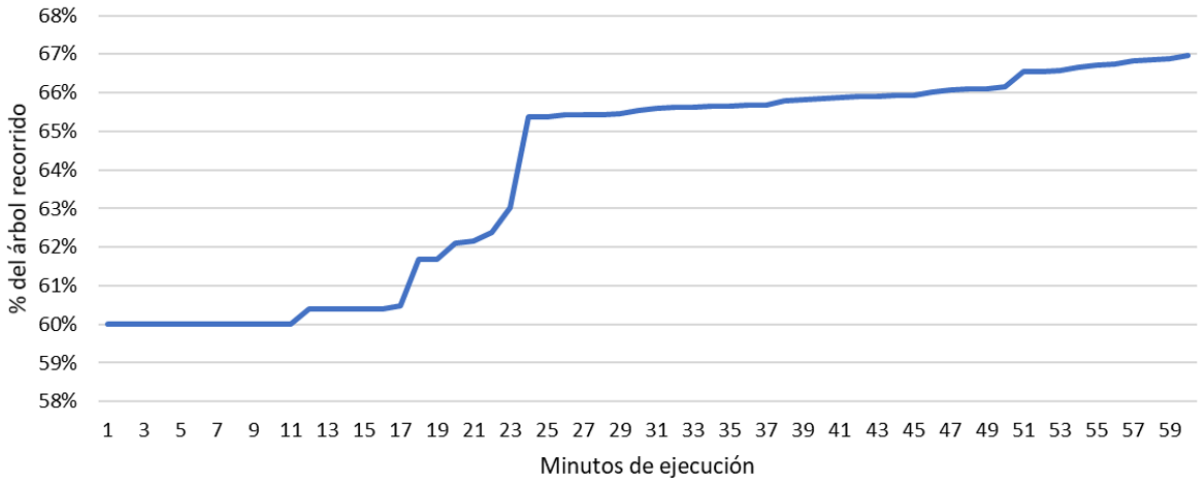

Figura 5.9: Evolución del porcentaje estimado del árbol recorrido durante una hora de ejecución.

5.3.3. Análisis Operacional de la Solución

Con el objetivo de analizar con claridad la solución obtenida, se elaboró un esquema para resumir la sincronización original de los trenes (Figura 5.10) y otro para resumir la sincronización propuesta por el algoritmo (Figura 5.11).

Dentro de los esquemas, las vueltas que realizan los trenes están indicadas por barras de colores. Guiándose por la línea temporal mostrada en la parte superior de las figuras, cada barra comienza en el minuto en que el tren cruza el inicio del túnel para llegar hasta la mina y finaliza cuando culmina de vaciar el mineral en un buzón de descarga. Las barras tienen detallado el nombre del pique al que se dirige el tren en la respectiva vuelta y además contienen una línea gris vertical que marca el instante en que atraviesa el inicio del túnel en dirección hacia el sector de descarga. Las barras rojas expresan los intervalos donde el tren debe esperar en la zona de descarga por su entrega que aún no está disponible dado que el pique todavía no tiene el mineral suficiente, mientras que los espacios blancos corresponden al trayecto entre los chancadores y el inicio del túnel para comenzar una nueva vuelta.

Comparando ambos resúmenes se logra visualizar que las dos vueltas ganadas son dadas por el T6 y el T9. En el primer caso, el viaje extra se explica principalmente por una disminución en el tiempo utilizado para completar la tercera vuelta, cuyo destino es el OP27/28, lo que deja el tiempo justo para que en la última hora del turno el tren pueda concretar su quinta entrega en un pique del XC12/13. Por otra parte, la vuelta adicional del T9 se debe claramente a que se emplearon menos minutos en el primer viaje gracias a menos interferencias operacionales, provocando que la segunda vuelta se iniciara una hora antes que en la

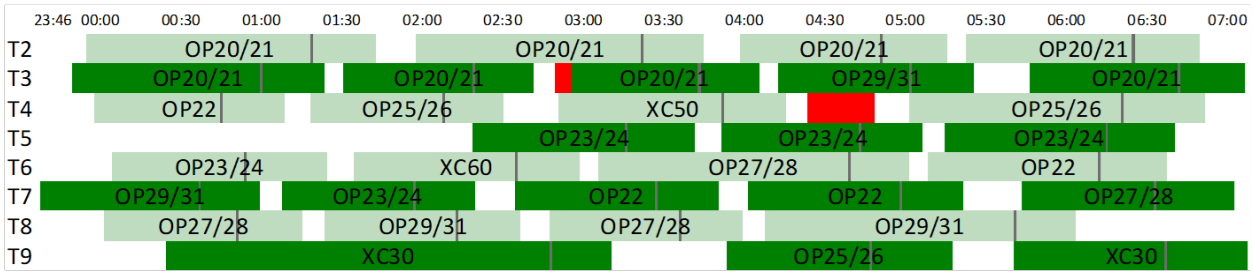

Figura 5.10: Esquema que resume la sincronización de trenes determinada por el despachador del turno.

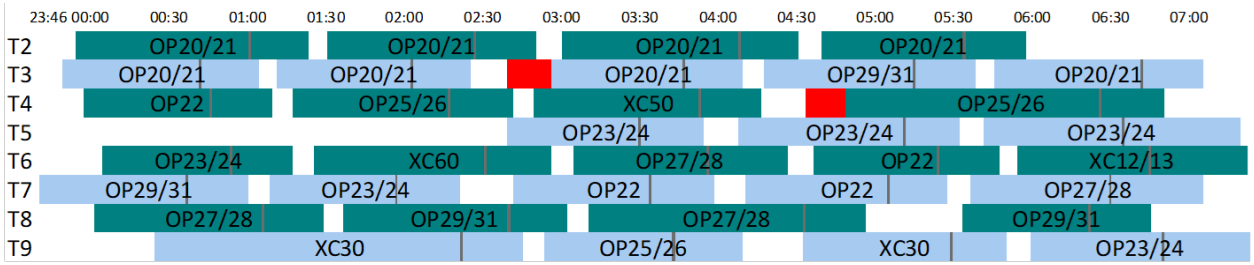

Figura 5.11: Esquema que resume la sincronización de trenes determinada por el algoritmo.

realidad y que, por ende, quedara el espacio temporal suficiente para realizar un cuarto viaje al pique OP23/24 antes de alcanzar la hora límite de entrada al túnel.

El hecho de que se obtuviera un resultado donde se acortó el tiempo de algunas vueltas para luego ganar 2 viajes adicionales, se debe a una serie de decisiones en las que el algoritmo tomó una alternativa diferente que lo hecho en la realidad, gracias a su capacidad de probar muchas combinaciones. Con la finalidad de ejemplificar lo mencionado, en la Figura 5.12 se presenta una captura de una herramienta desarrollada para ver en formato de video la sincronización entregada por el algoritmo. Los trenes se mueven a través de los vértices de la red, donde se muestran en color rojo los nombres (T1, T2, etc.) de aquellos trenes cargados que se dirigen hacia la zona de descarga y en color azul los nombres de los que van descargados hacia la mina.

En la situación mostrada en la Figura 5.12, que corresponde al minuto 44 (00:30) cuando se lleva a cabo la primera vuelta del turno, el T9 permanece en el desvío del túnel y se dirige hacia su respectivo pique, sin embargo, los trenes T7 y T3 ya se encuentran cargados y listos para trasladarse a descargar el mineral, lo que genera una interferencia en el túnel. El resto de los trenes también se mantienen ubicados en la mina y están a solo minutos de finalizar el proceso de carga para luego emprender camino a los buzones, a excepción del T2 que recién está ingresando a su pique.

Lo que se hizo en la realidad fue priorizar en el uso del túnel a los 7 trenes que se encontraban al interior de la mina y una vez que todos sobrepasaron el desvío, recién se dejó al T9 seguir su camino. En cuanto a la decisión tomada por la herramienta desarrollada, como al T2 le faltaba bastante tiempo para terminar de cargar, el algoritmo aprovechó ese

espacio y le permitió al T9 avanzar antes de que el T2 ingresara cargado al túnel en dirección hacia la zona de descarga, ganando minutos claves para que ese tren lograra acarrear una cuarta vuelta. Esta situación es solo una de las tantas decisiones que marcaron la diferencia entre el resultado obtenido por el despachador del turno y el resultado del algoritmo.

Si se analiza económicamente la ganancia de las dos vuelta, a la empresa esos viajes extra le reportan un ingreso adicional aproximado de US$150.000 por la venta de cobre fino, considerando la ley del mineral específica de los piques respectivos y asumiendo un precio de US$3 por libra de cobre.

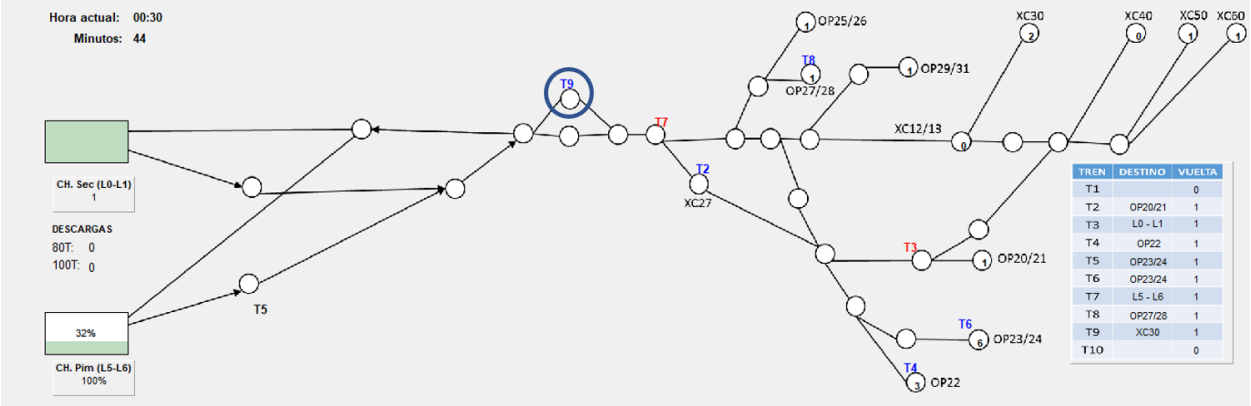

Figura 5.12: Representación gráfica de un instante de la solución del caso entregada por el algoritmo. En esa situación el T9 debe dirigirse a la mina, pero los demás trenes tienen que utilizar el túnel para ir hacia la zona de descarga.

En resumen, cuando se aplicó el algoritmo sobre este caso particular en el que existían oportunidades de realizar una mayor cantidad de vueltas en comparación a la realidad, se pudo conseguir al instante igualar el resultado logrado por el despachador de turno. Además, luego de 4 minutos de ejecución se superó en una vuelta el resultado original y al cabo de 14 minutos se llegó a una solución donde fue posible completar 2 viajes adicionales, igualando la cota superior previamente obtenida para el problema. También se demostró que al mejorar el incumbente del problema aumenta la tasa de poda del algoritmo, permitiendo llegar en menor tiempo a ramas con mayores posibilidades de tener mejores soluciones que la actual.

Esto prueba que es factible correr el algoritmo en un tiempo razonable y llegar a sincronizaciones con mejores resultados de lo que se puede conseguir sin la ayuda de una herramienta de estas características, lo que demuestra el valor que le agrega a la operación de la mina el algoritmo desarrollado.

5.4. Caso de Estudio: Turno de 40 Vueltas

Teóricamente en un turno normal de 8 horas con condiciones ideales, con suficientes entregas de mineral de modo que ningún tren deba esperar por su entrega y sin problemas técnicos en los trenes, chancadores, piques principales ni en la red ferroviaria, se puede alcanzar un

máximo 40 viajes [1] utilizando 8 trenes que acarreen 5 entregas cada uno. En la práctica, solo se tiene registro de un caso donde se realizaron 39 vueltas, correspondiente al turno B que se desarrolló entre las 16:00 y las 00:00 del día 14 de octubre del 2016.

El objetivo de este ejercicio es probar con el apoyo del algoritmo si es factible lograr ese número de viajes en un turno con condiciones óptimas.

Antes de ejecutar la prueba, los parámetros del problema son configurados para que todos los elementos funcionen sin inconvenientes, asegurando también que los 8 trenes cuenten con las 5 entregas que deberían acarrear para lograr las 40 vueltas, donde cada entrega tiene que estar disponible mucho antes de que el tren esté listo para cargarla, de modo que los trenes nunca necesiten realizar detenciones para esperar su respectiva entrega.

5.4.1. Parámetros

A continuación, se detallan los parámetros bajo los que se configuró al caso de estudio:

- **Hora de partida:** se aplicará el caso en un turno B, donde el primer tren ingresa al túnel en dirección mina a las 16:15.
- **Trenes:** 8 trenes en funcionamiento, 2 de mineral fino (T2 y T3) y 6 de mineral grueso (T4, T5, T6, T7, T8 Y T9). Todos los trenes inician sus funciones con normalidad y se mantienen trabajando sin problemas durante el turno entero.
- **Posición inicial de trenes:** los 8 trenes parten el turno descargados y distribuidos en los vértices del sector de descarga.
- **Entregas del turno:** se disponen 40 entregas, 10 de mineral fino y 30 de mineral grueso. Todas esas entregas se configuran para que estén listas en una hora que impida que los trenes deban esperarlas. Además, las entregas se asignan en los piques principales considerando la proporción de la producción correspondiente a cada sector minero y la distribución espacial de los cruzados en los que se encuentran los piques, de manera que dos trenes no deban ir a cargar al mismo cruzado en la misma vuelta, evitando esa clase de interferencias.
- **Asignación de trenes a entregas:** se realiza una asignación de trenes a entregas basándose en la posición inicial de los trenes, tratando de impedir que se produzcan mayores obstrucciones al menos en las primeras vueltas.
- **Chancadores primarios:** 100 % del turno con 2 chancadores disponibles.
- **Chancadores secundarios:** 100 % del turno con ambos trabajando.
- **Situaciones especiales:** no existen fallas en los elementos partícipes de la etapa de transporte principal.

El resto de los datos corresponde a los parámetros generales, por lo tanto, no hay cambios en comparación a los valores expuestos en el **Capítulo 5.3.1**.

[1] En un turno especial con una duración mayor a 8 horas es posible completar más de 40 viajes.

5.4.2. Ejecución del Algoritmo y Resultados

Una vez que se establecieron los parámetros del caso, se procedió a ejecutar el algoritmo durante un período de una hora dividiendo el problema en 32 subárboles, que es la cantidad óptima de threads a correr en el computador utilizado en el contexto de este trabajo.

La cota superior que se estimó a priori para la solución del caso es de 40 vueltas que se completan en un tiempo mínimo de 449 minutos. Dicha cota se refleja en el gráfico de la Figura 5.13, donde además se muestra la evolución del incumbente respecto al número de viajes que los trenes logran acarrear dentro del turno.

Desde la imagen es posible observar que en el instante inicial el algoritmo llega a una sincronización factible en la que se realizan 39 vueltas, igualando la cantidad de viajes del mejor turno del que se tenía registro. El incumbente se mantuvo con ese mismo valor durante 32 minutos, luego de lo cuál se encuentra una solución de 40 vueltas.

La sincronización obtenida permite concretar los 40 viajes en 483 minutos, lo que se traduce en que el último tren termina de vaciar el mineral en un buzón del sector de descarga a las 00:16. Después de llegar a esa solución el incumbente no sufrió modificaciones antes de cumplir la hora de ejecución, dado que no se llegó a sincronizaciones factibles que completaran las 40 vueltas en un menor tiempo. Por lo tanto, con la solución entregada por el algoritmo se genera un gap de 0 vueltas y 34 minutos en relación a la cota.

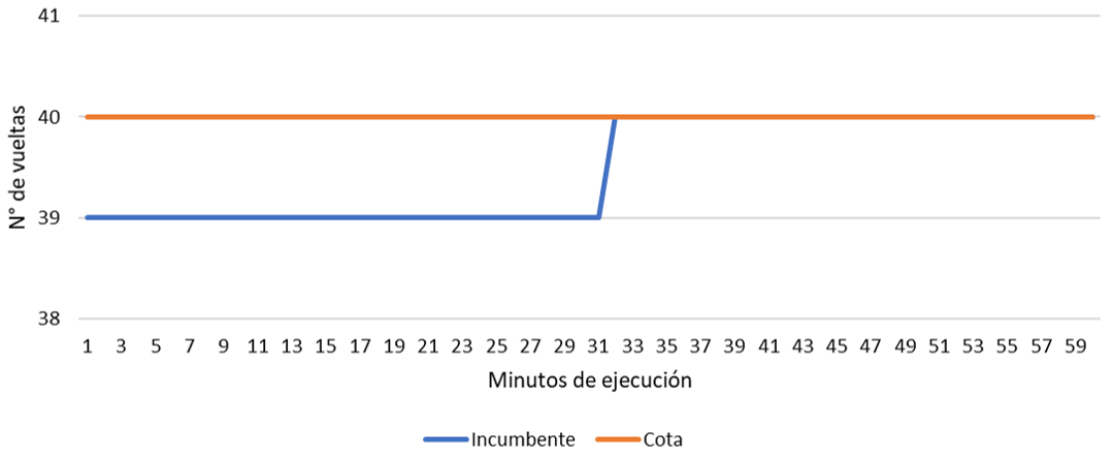

Figura 5.13: Evolución del incumbente para la cantidad de vueltas completadas versus la cota máxima para el mismo indicador.

Dentro de los 60 minutos en que el caso estuvo corriendo, se crearon un poco más de 2 millones de nodos del árbol de decisión, sumando el trabajo de los 32 threads que se ejecutaron en paralelo. Como se observa en la Figura 5.14, de ese total de nodos el 45,1 % fue podado porque su proyección arrojó un peor desempeño que el incumbente, el 9,3 % resultó ser infactible y solo un 0,2 % de los nodos llegó a soluciones factibles para el problema. El otro 45,5 % engloba a todos aquellos nodos intermedios del árbol que se ramificaron.

Al igual que en el caso de estudio del Turno C 23/01/2017, en esta instancia se realiza una estimación del tamaño completo del árbol, lo que se calcula en base a un promedio

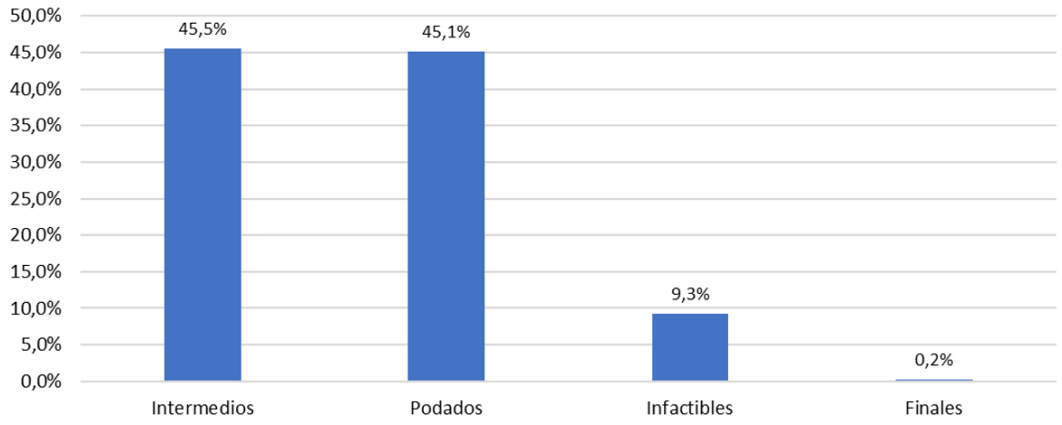

Figura 5.14: Distribución porcentual de los nodos del árbol creados según tipo.

de la profundidad de los nodos terminales que se obtuvieron en diferentes casos. Con esa cifra se estima que el árbol tiene aproximadamente 50 niveles y que, por lo tanto, tiene una dimensión de $2,25 \times 10^{15}$ nodos, los que podrían ser construidos totalmente luego de 126.000 años, sin considerar, al igual que en el caso de estudio anterior, que la cantidad real de nodos probablemente es más baja debido a la existencia de nodos del árbol infactibles y al método de poda de ramas que descarta del recorrido cierto porcentaje de subárboles.

La Figura 5.15 muestra el porcentaje del árbol recorrido a medida que se avanza temporalmente en la ejecución del caso. Al comienzo se poda cerca del 65 % del árbol debido a nodos infactibles ubicados en niveles superiores, sin embargo, durante la primera media hora solo existen alzas muy leves en la fracción del árbol total recorrida. Por otra parte, en la segunda media hora se producen importantes avances, debido principalmente a una tasa de poda más alta gracias a un incumbente de 40 vueltas, lo que impone un criterio de poda bastante estricto y permite descartar nodos en niveles más altos del árbol.

Finalmente, al cabo de una hora el algoritmo recorrió aproximadamente un 71 % del total del árbol estimado, considerando los nodos construidos y todos aquellos subárboles que fueron rápidamente descartados dado que resultaron infactibles o fueron podados.

De igual manera que en el caso anterior, con el objetivo de dimensionar cuánto tiempo más se requiere para recorrer el 100 % del árbol estimado, se toma la tasa de recorrido del último minuto de ejecución, la que en este caso es igual a 0,000034 % del árbol recorrido por minuto. Con dicho valor, se proyecta una tasa constante hacia el futuro y se calcula que se necesitan 852.941 minutos para completar el 29 % restante del árbol de decisión, lo que equivale a alrededor de 1 año y 7 meses. Evidentemente la tasa no necesariamente es constante, por lo que el tiempo que en realidad falta dependerá de la manera en que evoluciona la tasa de recorrido a lo largo de la ejecución.

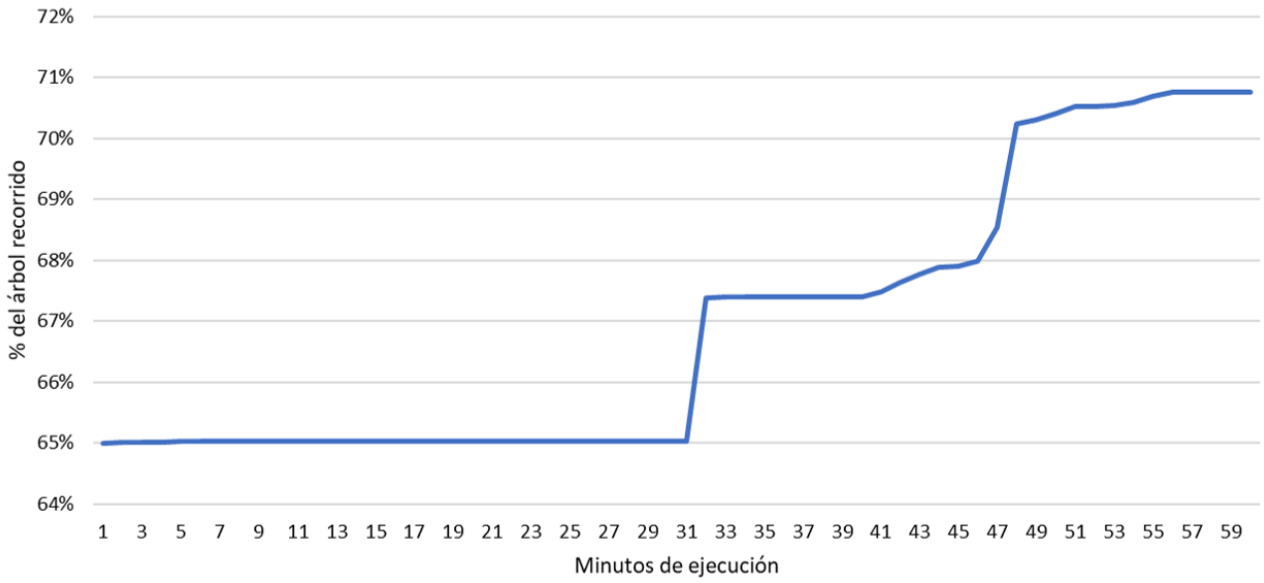

Figura 5.15: Evolución del porcentaje estimado del árbol recorrido durante una hora de ejecución.

5.4.3. Análisis Operacional de la Solución

Para tener una visión general de la forma en que se desplazaron y coordinaron los 8 trenes a lo largo del turno, en la Figura 5.16 se muestra el resumen de la sincronización propuesta por el algoritmo. A partir del esquema se puede observar que los trenes realizan las 5 vueltas relativamente juntos, entrando y saliendo de la mina en caravana durante la mayoría del turno, lo que evita una gran cantidad de interferencias que se producen con frecuencia cuando los trenes transitan en sentido contrario por el túnel principal. Esa buena sincronización permite que ningún tren se quede atrás y que en la última vuelta todos alcancen a ingresar al túnel antes de cumplir la hora límite (23:00 para un turno B), para así completar su quinto viaje.

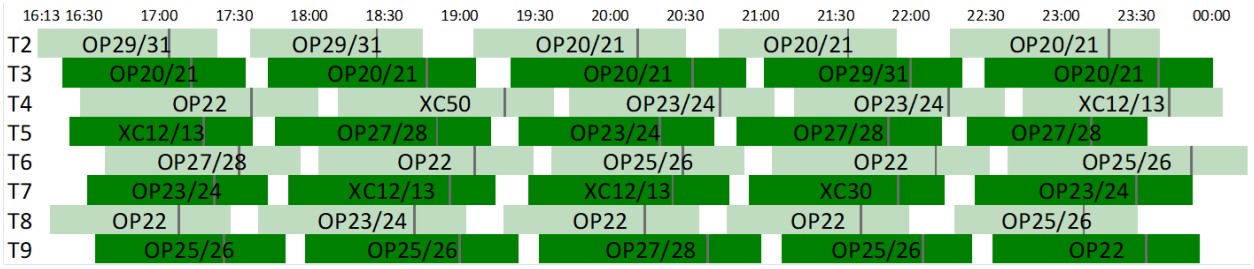

Figura 5.16: Esquema que resume la sincronización de trenes determinada por el algoritmo para un turno con 40 viajes completados.

Para ejemplificar lo descrito previamente, en la Figura 5.17 y en la Figura 5.18 se muestran capturas de una herramienta visual desarrollada para reproducir las soluciones del algoritmo. Dichas imágenes representan dos instantes distintos del desplazamiento de los trenes a lo largo del turno, recordando que los trenes cuyo nombre se encuentra en color rojo son los que

se dirigen hacia la zona de descarga, mientras que los trenes con nombre en azul se trasladan vacíos hacia el interior de la mina.

En el primer caso los trenes se desplazan en fila hacia la zona de descarga para efectuar el vaciado de la tercera vuelta del turno. Es posible notar que el T2 y el T8 ya arribaron a las líneas de descarga, mientras que el resto de los trenes se mueven ordenadamente en cola para ingresar a aquel sector, por lo que no se generan obstrucciones debido a trenes moviéndose en dirección contraria a través de la red ferroviaria.

En cuanto a la segunda figura, todos los trenes están desplazándose hacia el interior de la mina para completar la carga de la quinta vuelta del turno. Eso permite que los últimos trenes tengan tiempo suficiente para ingresar al túnel y trasladarse hasta el interior de la mina sin que sufran interferencias ni esperas.

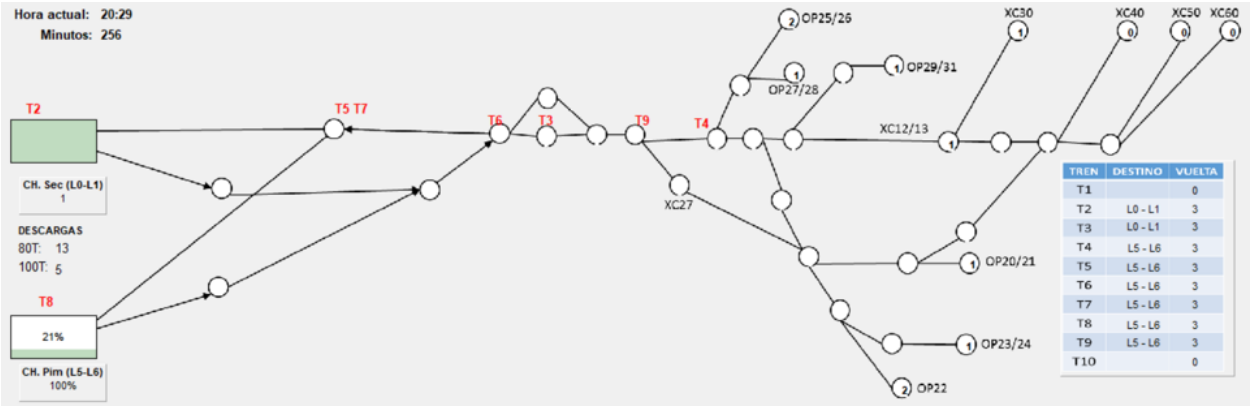

Figura 5.17: Captura del minuto 256 del turno (20:29) cuando los trenes se desplazan en caravana hacia la zona de descarga para vaciar la tercera vuelta.

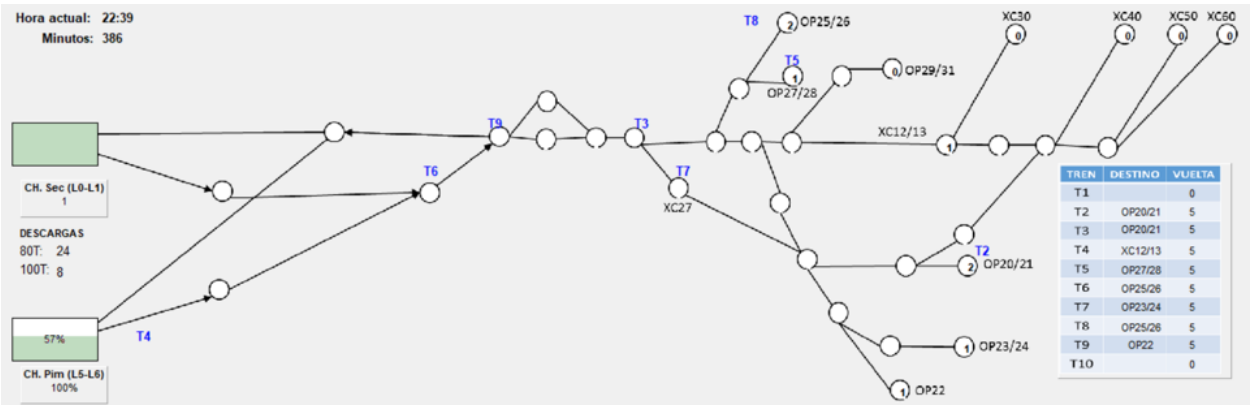

Figura 5.18: Captura del minuto 386 del turno (22:39) cuando los trenes se desplazan en caravana hacia el interior de la mina para realizar la carga de la quinta vuelta.

Este caso desarrollado con el objetivo de verificar la factibilidad de completar 40 viajes en un turno con óptimas condiciones, tuvo resultados positivos, ya que demostró que mediante el apoyo del algoritmo es posible obtener una buena sincronización de los trenes de manera

que logren realizar los 40 viajes. En particular, la herramienta consiguió coordinar a los trenes de forma tal que transitaran la mayor parte del turno en caravana, lo que minimizó las interferencias al interior del túnel y las esperas operacionales de los trenes, algo que es muy difícil de lograr sin la colaboración de una herramienta como la presentada en esta tesis.

La solución de este caso sin duda reafirma el valor que otorga el algoritmo a la operación diaria de la mina, pues prueba que la herramienta es capaz de conseguir resultados que han sido bastante complicados de alcanzar en la práctica.

Capítulo 6

Conclusiones

En este capítulo se describen las principales conclusiones y reflexiones acerca de posibles trabajos futuros originadas a partir del desarrollo del algoritmo y del análisis de las soluciones obtenidas en los casos de estudio.

6.1. Conclusiones Generales

La mina estudiada presenta un gran desafío operacional al momento de coordinar los trenes que diariamente se desplazan a través de la limitada red ferroviaria con la que se cuenta. Hasta ahora esa compleja labor ha sido abordada de forma completamente humana, pero hoy en día se hace necesario responder a las grandes demandas de la industria, por lo que es indispensable involucrar elementos tecnológicos para apoyar el trabajo de los despachadores de turno, dado que la enorme cantidad de combinaciones posibles solo pueden ser procesadas por herramientas computacionales.

Esta tesis tuvo por objetivo desarrollar un algoritmo para colaborar con la coordinación de los trenes, de manera que se logre planificar sincronizaciones en las que se realicen más vueltas de trenes que en la actualidad, donde cada viaje extra tiene un importante impacto económico para la empresa.

El algoritmo se basó en la estructura de los árboles de decisión. En este caso el árbol se ramifica en dos cada vez que un par de trenes se interfieren en la red férrea. Una de las particularidades es que se incorporó un método de poda de ramas mediante la elaboración de cotas, inspirado en Branch & Bound, lo que hace mucho más eficiente el recorrido del algoritmo al descartar tempranamente nodos de baja calidad y priorizar aquellas ramas con mejores expectativas de derivar en buenas soluciones.

Con la idea de aumentar la velocidad de recorrido a través de las ramas del árbol y aprovechar al máximo la capacidad computacional, se aplicó una técnica de paralelización, donde se divide el problema en varios subárboles que se ejecutan en paralelo en diferentes threads. Se realizaron pruebas para establecer el número óptimo de threads a ejecutar, llegando a la

conclusión de que en el computador utilizado en este trabajo se lograba una mayor rapidez de recorrido dividiendo el problema en 32 subárboles, que es igual a la cantidad de subprocesos con los que cuenta dicho PC.

La construcción de las ramas del árbol se hace bajo una lógica de recorrido en profundidad hacia la izquierda, que demostró ser la más eficiente en este problema, pero que genera un sesgo en las soluciones del árbol. La división del problema en subárboles mitiga en parte dicha problemática, pero no es suficiente. Dado eso, con la finalidad de reducir aun más el sesgo se desarrolló un método de sondajes, el que consiste en explorar de manera limitada cada sector particular del árbol para luego pasar a otras zonas que también podrían contener soluciones de alta calidad, impidiendo que el algoritmo se quede estancado iterando solo en un sector del árbol de decisión.

Al ejecutar el algoritmo con 32 threads en un computador con 64 GB de memoria RAM y con dos procesadores Intel Xeon E5-2620 v4 de 8 núcleos y 16 subprocesos, se generan un poco más de 2 millones de nodos en una hora, de los cuáles se poda aproximadamente un 40 %, un 10 % resulta infactible, alrededor del 1 % llegan a ser nodos terminales con sincronizaciones factibles, mientras que el porcentaje restante corresponde a los nodos que logran ramificarse, es decir, nodos intermedios o no terminales del árbol. La alta tasa de poda y la baja cantidad de nodos terminales es un hecho muy positivo, ya que permite enfocar el recorrido por ramas con grandes probabilidades de tener soluciones de alta calidad y así llegar solo a un selecto grupo de buenas sincronizaciones factibles para el turno.

En esta tesis se presentaron dos casos de estudio que permitieron probar la funcionalidad y rendimiento del algoritmo, mostrando excelentes resultados, pues en el primer caso la herramienta consiguió en menos de 15 minutos de ejecución encontrar una solución donde se completan dos vueltas más que en la coordinación aplicada en la realidad y en el segundo se evidenció que es factible alcanzar el máximo teórico de 40 vueltas en un mismo turno bajo condiciones ideales.

Los resultados obtenidos al utilizar el algoritmo en dos casos diferentes demostraron claramente su efectividad, dado que en un tiempo razonable la herramienta entrega soluciones de alta calidad, superando lo que puede ser realizado por una persona sin la ayuda de un elemento tecnológico. En consecuencia, se concluye que el trabajo de esta tesis cumplió con sus propósitos y se convierte en un importante apoyo para los despachadores de turnos en la operación diaria, con la idea de incrementar el flujo de mineral acarreado y los ingresos económicos de la empresa.

6.2. Limitaciones y Trabajos Futuros

Si bien el algoritmo mostró un excelente desempeño y cumplió con las expectativas, existen ciertas deficiencias en su funcionamiento que podrían ser abordadas en siguientes trabajos.

En primer lugar, el algoritmo está limitado por una asignación de trenes a piques fijada previamente, lo que dificulta su utilización, pues el usuario siempre tiene que ingresar esa

información como parámetro del turno. Lo anterior se hace especialmente complicado al aplicar el algoritmo en turnos futuros donde no existe asignación previa. Además, se tiene conocimiento de que en algunos casos también hay oportunidades de mejora en la asignación, las que no están siendo aprovechadas por el algoritmo. Dicho problema puede ser abordado mediante un método heurístico simple que requiera poca capacidad computacional y tiempo, pues el problema de la asignación es de menor impacto en el resultado final del turno en comparación a la coordinación de los trenes. De hecho, la búsqueda de una asignación cercana al óptimo solo tiene sentido cuando se obliga a trenes a esperar una entrega de mineral específica siendo que existe otra entrega disponible para ser acarreada, donde una buena asignación permitiría a los trenes ahorrarse dichos tiempos de espera y llegar a una solución mejor.

Por otra parte, el método de resolución en paralelo puede presentar una complicación a la hora de implementarse, pues a veces los nodos escogidos como raíz de un subárbol, son infactibles. Eso implica que no se ejecuten los n subárboles en paralelo, sino que un número menor, dejando sin uso cierta parte de la capacidad computacional disponible. Como forma de solucionar dicha problemática, en el futuro se puede desarrollar una lógica dinámica de elección de raíces, donde el algoritmo identifique si una raíz es infactible y le reasigne esa misión a otro nodo del árbol, para que así se construya siempre la cantidad de subárboles definida al comienzo. Dicha lógica dinámica es de poca complejidad y de fácil implementación, pero podría significar un impacto positivo en el rendimiento del algoritmo en aquellos árboles con muchos nodos infactibles en sus niveles superiores, casos que en realidad representan un porcentaje menor del total de turnos en los que se probó la herramienta.

A pesar del buen resultado de la poda de ramas, también se propone indagar en mejores métodos de creación de cotas, ya que actualmente representan casos optimistas y en algunas ocasiones su valor podría estar muy alejado de las soluciones factibles de los nodos descendientes de aquel nodo en que se estimó la cota. Eso ayudará a perfeccionar el método de poda y a obtener un gap más realista, para así realizar mejores análisis sobre las soluciones.

Finalmente, es preciso comentar que gracias a que la tecnología va evolucionando constantemente, existe la posibilidad de cada vez adquirir computadores con mejores procesadores, en los que se podría ejecutar el algoritmo con un rendimiento mucho mayor al mostrado en la actualidad. Por ejemplo, el procesador Intel Xeon E5-2620 v4 de 8 núcleos y 16 subprocesos (el computador utilizado en esta tesis cuenta con dos de estos procesadores) fue lanzado en los primeros meses del año 2016, pero hoy en día se vende, por ejemplo, por aproximadamente 7.000 dólares un procesador Intel Xeon E5-4669 v4 de 22 núcleos y 44 subprocesos. Eso significa que con un solo procesador se pueden ejecutar 12 threads en paralelo más que la suma de los dos procesadores con los que se cuenta hoy en día. Esto permite que a medida que se perfeccionan las tecnologías se puedan obtener soluciones cercanas al óptimo en un menor tiempo, lo que es fundamental para que la herramienta sea aplicada en la operación diaria de la mina.

Bibliografía

[1] Dimitris Bertsimas and John Tsitsiklis. *Introduction to Linear Optimization.* Athena Scientific, Belmont, Massachusetts, 1997.

[2] Cristián Cortés. *Apunte de Cátedra Logística y Producción.* Departamento de Ingeniería Civil, Universidad de Chile, 2015.

[3] Jorge Alarcón Díaz. *Evaluación de la resolución en paralelo de un problema estocástico de planificación minera de largo plazo.* Memoria de Ingeniero Civil Industrial. Santiago, Universidad de Chile, Facultad de Ciencias Físicas y Matemáticas, 2012.

[4] Benjamín Bustos y Patricio Poblete. *Apunte de Cátedra Algoritmos y Estructura de Datos.* Departamento de Ciencias de la Computación, Universidad de Chile, 2004.

[5] René Caldentey y Susana Mondschein. *Modelos de Decisión en Ambientes Inciertos.* Departamento de Ingeniería Industrial, Universidad de Chile, 1999.