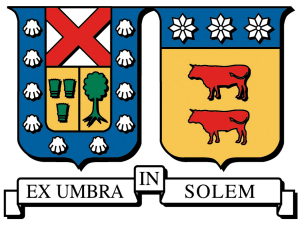
Universidad Técnica Federico Santa María
Departamento de Informática
Valparaíso, Chile



Algorithms based on heuristics for solving the Train Departure Matching Problem

Alondra Valentina Rojas Ruz

Tesis para optar al Grado de
Magíster en Ciencias de la Ingeniería Informática

Profesor Guía: María Cristina Riff
Profesor Correferente Interno: Carlos Castro
Profesor Correferente Externo: Carlos Coello
Presidente Comisión: Marcelo Mendoza

5 de abril de 2018

Dedicado a mis amadas madre y hermana

Agradecimientos

Quisiera agradecer a mi hermosa familia por apoyarme y alentarme siempre. A mi mamá, un gran ejemplo de mujer, y a mi amada hermana Valentina, que día a día alegra mis días. A mis abuelas Nora y Mami, que no se que haría sin ellas.

También quiero agradecer a mis amigos y compañeros en este proceso: Alejndro y Andrea, los adoro infinito, muchas gracias por su apoyo a lo largo de este proceso. Alejandro, simplemente te amo.

A los chicos del lab: Eli, Nico y Javi, por toda su alegría y buena onda, el poquito tiempo que estuve trabajando con ustedes fue genial.

Por último, a la profesora María Cristina Riff por guiarme e incentivarme en todo este proceso, gracias.

Resumen

El problema de asignación de salidas de trenes (DMP por sus siglas en inglés) es un problema de planificación de trenes, donde el objetivo es encontrar la mejor manera de asignar trenes a salidas programadas, sujetas a un conjunto de restricciones relacionadas con la gestión ferroviaria. Este es un problema que surge del *rolling stock units management on railway sites problem* presentado en el Desafío ROADEF/EURO 2014, el cual se suele tratar como dos sub-problemas: El DMP y un problema de planificación de rutas. En este trabajo se presenta formalmente el DMP, indicando su modelo, los estudios realizados a la fecha sobre este problema y otros problemas de gestión de trenes relacionados, para luego presentar la propuesta de un algoritmo tipo GRASP (G-DMP) y un Algoritmo Evolutivo (AE-DMP) para resolver este problema y luego evaluar su desempeño por medio de diferentes experimentos.

Abstract

The Train Departure Matching Problem (DMP) is a problem of train scheduling where the objective is find the best way to assign trains to scheduled departures subject to a set of constraints related to railway management. It is a problem that arises from the rolling stock units management on railway sites problem, presented in the ROADEF/EURO Challenge 2014, which is usually treated as two sub-problems: The DMP and the routing problem. In this work the DMP is formally presented: the model of the DMP, a review of the literature of this problem and other train scheduling problems, then present the GRASP algorithm (G-DMP) and theEvolutionary Algorithm (AE-DMP) capable of solving this problem and then evaluate its performance in different experiments.

Índice general

Índice de Tablas

Índice de Figuras

Índice de Algoritmos

Capítulo 1

Introducción

El *Train Departure Matching Problem* DMP es un problema de asignación de trenes de distintas categorías, que son asignados a salidas programadas. Este problema nace a partir del problema de manejo de trenes en ferrocarriles (RSUM[1]), el cual consiste en encontrar la mejor manera de gestionar los trenes entre sus llegadas y salidas en estaciones terminales (plataformas), tomando en cuenta diferentes restricciones y costos asociados a los diferentes eventos que ocurren y los recursos de infraestructura que posee un sistema ferroviario, por donde pasará el tren mientras permanezca en el sistema. Este problema se reparte entre varios departamentos de la SNCF[2]. En el año 2014 se planteó este problema, descrito muy cercano a la realidad, como un desafío a resolver en la conferencia ROADEF/EURO.

Para abordar el RSUM, los completidores lo trataron como dos subproblemas que podían ser resueltos por separado: el DMP y el *Routing Problem*, que tiene que ver con planificar las rutas que realizan los trenes dentro de dichos recursos de infraestructura que posee el sistema ferroviario. El RSUM busca ser lo más cercano a la realidad.

En esta tesis se presentarán dos algoritmos basados en heurísticas que resuelvan el DMP: un algoritmo basado en GRASP: G-DMP, y un Algoritmo Evolutivo: AE-DMP.

1.1. Objetivos

El desarrollo de este trabajo de tesis, se enmarca en los siguientes objetivos.

1.1.1. Objetivo Principal

El objetivo principal es proponer algoritmos basados en heurísticas, para resolver el problema de *matchings* de salidas DMP, que sean capaces de incluir todos los

[1]Rolling stock units management on rileways sites

[2]*Société Nationale des Chemins de Fer Français*

costos involucrados.

Para lograr este objetivo es necesario cumplir con los siguientes objetivos específicos:

1.1.2. Objetivos Específicos

Los objetivos específicos, definidos para llevar a cabo el objetivo principal son:

- Investigar el problema y las soluciones ya propuestas, además de otros problemas similares, relacionados con el agendamiento de trenes, para así conocer qué técnicas han tenido buenos resultados con este tipo de problemas y tener nociones de heurísticas a utilizar que resulten útiles a la hora de diseñar el algoritmo.
- Diseñar e implementar los algoritmos basados en heurísticas que sean capaces de resolver el problema mencionado.
- Realizar una evaluación de las técnicas propuestas y la comparación de los resultados obtenidos con los existentes en la literatura.

1.2. Estructura del Documento

El presente documento está estructurado de la siguiente manera. En el Capítulo 2 se describirá el DMP de manera detallada, indicando sus componentes, restricciones y costos que definen el modelo. En el Capítulo 3 se analizan diversas técnicas y enfoques utilizados para abordar distintos problemas de gestión y agendamiento de trenes, incluyendo al RSUM y al DMP. En el Capítulo 4 se detalla el G-DMP, algoritmo basado en *Greedy Randomized Adaptive Search Procedure* (GRASP), diseñado para la resolución del problema. En el Capítulo 5 se detallan los experimentos realizados y se presentan los resultados obtenidos por G-DMP. En el Capítulo 6 se detalla el AE-DMP, un algoritmo evolutivo diseñado para resolver el DMP. En el Capítulo 7, se presentan los experimentos y resultados obtenidos por el AE-DMP, y se comparan con los resultados del G-DMP y los resultados existentes la literatura. Finalmente, en el Capítulo 8 se señalan las conclusiones obtenidas en este trabajo y se plantean las proyecciones para un trabajo futuro.

Capítulo 2

Definición del Problema

El problema de asignación de salidas de trenes, *Train Departure Matching Problem* (DMP) es un problema de optimización que consiste en encontrar la mejor manera de asignar trenes de distintas categorías a salidas agendadas, considerando que los trenes pueden estar ubicados inicialmente en el sistema ferroviario o ingresar al sistema por medio de llegadas, además de considerar aspectos prácticos de la gestión de trenes, como el mantenimiento, la unión o separación de trenes y los tiempos de permanencia en los recursos.

En el DMP los eventos ocurren en un horizonte de tiempo discreto y variable, dentro de un sistema ferroviario que posee diferentes recursos de infraestructura, como se muestra en la Figura 2.1. Los eventos más importantes en el problema son las llegadas (*arrivals*) y salidas (*departures*), que son realizadas en plataformas. Los grupos de vías (*track groups*) conectan todos los recursos, además existen vías simples donde dos trenes pueden ensamblarse o separarse, además de los talleres de mantenimiento (*maintenance facilities*) donde los trenes llevan a cabo sus mantenimientos. Los trenes pueden estacionarse en patios de estacionamiento (*yards*). Por último, aquellos trenes que están inicialmente en el sistema pueden estar en un *yard* o en una vía simple, al comienzo del horizonte de tiempo.

2.1. Componentes del Problema

Para definir completamente este problema, es necesario conocer en detalle los siguientes conceptos:

Horizonte de tiempo *(Planning Horizon)*

Es el tiempo donde transcurre un evento. No se especifica una fecha exacta, así que se hablan de días $d_1, d_2, \ldots d_{nbDays}$ que van desde el día 1 hasta el día $nbDays$.

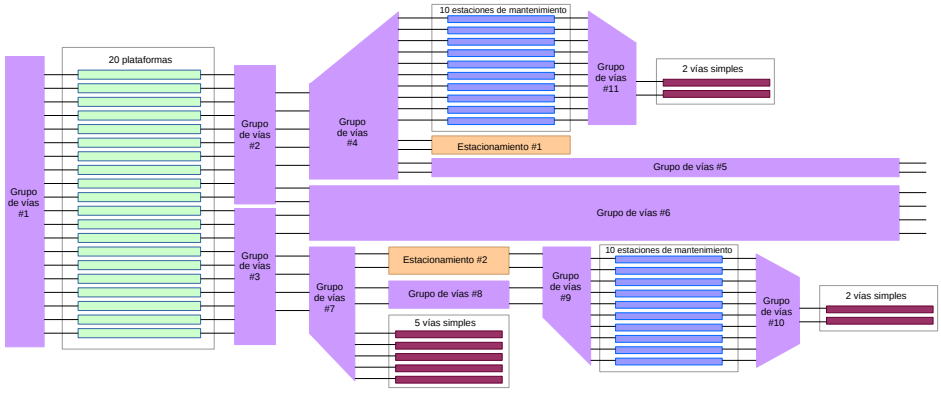


Figura 2.1: Recursos de Infraestructura de una estación

Es una variable discreta donde la precisión es alta, ya que el menor instante de tiempo considerado es el segundo.
Un tiempo o *time*, se refiere a un instante de tiempo dentro del horizonte y se describe como: "$d_i\ hh:mm:ss$', donde $i \in [\![1, nbDays]\!]$ y d_i es el día, $hh \in [\![0, 23]\!]$ se refiere a la hora, $mm \in [\![0, 59]\!]$ a los minutos y $ss \in [\![0, 59]\!]$ a los segundos.
$\mathcal{H}$ es conjunto de todos los instantes de tiempo dentro del horizonte.

Llegadas (*Arrivals*)

Una llegada es el fin del viaje para los pasajeros, ya que el tren ingresa al sistema y llega a una plataforma. Los tiempos de llegada son *inputs* no modificables y corresponden al ingreso del tren en la plataforma.

Se define el conjunto $\mathcal{A}$ como el conjunto de llegadas durante el horizonte de tiempo. Una llegada $a \in \mathcal{A}$ se define según los siguientes atributos:

- El tren asociado $arrTrain_a$.
- El tiempo de llegada $arrTime_a$.
- El tiempo ideal de permanencia en una plataforma $idealDwell_a$ y el tiempo máximo de permanencia en una plataforma $maxDwell_a$.
- La distancia restante antes de una mantención DBM[1], $remDBM_a$ y el tiempo restante antes de una mantención TBM[2], $remTBM_a$.

[1] *Distance before maintenance*
[2] *Time before maintenance*

- $linkedDep_a$, que corresponde a una salida asociada. Cuando hay una salida asociada es porque el tren indicado, físicamente salió del sistema con otro id y volvió a ingresar. Como previamente realizó una salida, los datos del input ya no son los que corresponden a ese tren, sino que deben actualizarse según el tren que fue utilizado para cubrir dicha salida.

También puede darse el caso de una llegada en conjunto, que es cuando un tren compuesto de la unión de dos o más trenes ingresa a una plataforma. Esta situación se explicará posteriormente con más detalles.

Salidas (*Departures*)

Es donde comienza el viaje para los pasajeros, es decir, cuando el tren se va de la plataforma. En este modelo es cuando el tren sale del sistema.
Un tren puede ser asignado a lo más a una salida o *departure*, donde $depTrain_d$ es la asignación de un tren a la salida d. Las salidas no cubiertas (*uncovered departures*) constituyen una parte importante de la función objetivo.
Se define el conjunto $\mathcal{D}$ como el conjunto de salidas durante el horizonte de tiempo. Una llegada $d \in \mathcal{D}$ se define según los siguientes atributos:

- El tiempo de salida $depTime_d$.
- El conjunto de categorías de trenes compatibles con la salida $compCatDep_d$.
- El tiempo ideal de permanencia en una plataforma $idealDwell_d$ y el tiempo máximo de permanencia en una plataforma $maxDwell_d$ después de una salida.
- La distancia $reqDBM_d$ y el tiempo $reqTBM_d$ del viaje después de la salida. Estos valores se comparan con los DBM y TBM restantes para un tren $t \in \mathcal{T}$ para determinar si necesita mantenimiento o no. Las operaciones deben realizarse antes de $depTime_d$.

Al igual que en las llegadas, puede darse el caso que trenes unidos salgan de la plataforma.

Trenes

Sea $\mathcal{T}$ el conjunto que contiene todos los trenes. Este conjunto de trenes se divide en:

- Trenes que están actualmente en el sistema, es decir tienen algún recurso en el tiempo de inicio del horizonte. Estos trenes pertenecen al conjunto $\mathcal{T}_{\mathcal{I}} \subset \mathcal{T}$
- Trenes que se asocian a una llegada. Estos trenes pertencen al conjunto $\mathcal{T}_{\mathcal{A}} \subset \mathcal{T}$.

Un tren $t \in \mathcal{T}$ tiene una categoría $cat_t \in \mathcal{C}$ que define algunas características técnicas que tiene en común con otros trenes de la misma categoría.

A continuación se definen aspectos importantes de los trenes:

- **Trenes inicialmente en el sistema:** Los trenes que están inicialmente en el sistema poseen los siguientes atributos:
  - res_t: son los recursos usados por un tren t el d_1 $00:00:00$,
  - $remDBM_t$: la distancia restante antes de la mantención de t,
  - $remTBM_t$: el tiempo restante antes de la mantención de t.

  Así como algunas llegadas pueden estar no cubiertas, se puede optar por no usar algunos trenes inicialmente en el sistema, por ejemplo si se utilizan todos los trenes de las llegadas para cubrir salidas, no es necesario utilizar estos trenes iniciales.

- **Categorías de trenes:** Tal como se mencionó, los trenes pueden compartir características en común y se agrupan en categorías. Una categoría $c \in \mathcal{C}$ donde $\mathcal{C}$ es el conjunto de todas las categorías de trenes, se define por:
  - El largo $length_c$.
  - El grupo de compatibilidad $catGroup_c$ (características físicas y técnicas que permiten unión de trenes).
  - Distancia máxima antes de la mantención $maxDBM_c$. Es el máximo número de kilómetros de un tren de categoría c puede recorrer entre dos mantenimientos de tipo D.
  - Similar al punto anterior, el tiempo máximo antes de la mantención $maxTBM_c$.
  - El tiempo $maintTimeD_c$ requerido para realizar una mantención de tipo D.
  - El tiempo $maintTimeT_c$ requerido para realizar una mantención de tipo T.

  Por simplicidad se usará que las características de un tren se referirán a las de su categoría.

- **Preferencia por la reutilización de trenes:** Para asegurar la factibilidad de la gestión de los trenes, se planean algunas acciones de antemano, dentro de estas está la reutilización de algunos trenes, donde se busca predeterminar los trenes que están llegando a una estación a la que se les asignará una salida, si bien no es lo óptimo en casos donde hay cambios en la planificación, es más realista con la información que comparten los trabajadores. Estas preferencias están dadas como *input* en algunas instancias. Una reutilización $u \in \mathcal{U}$, donde $\mathcal{U}$ es el conjunto de todas las preferencias, se define por una llegada arr_u y una salida dep_u. Si no se satisfacen estas preferencias hay un costo asociado.

Llegadas en conjunto y salidas en conjunto (*Joint-arrivals and joint-departures*)

Puede darse el caso en que mucha gente quiera viajar hacia un cierto lugar, y necesiten unirse dos o más trenes, tal como se muestra en la Figura 2.2.

Se denomina llegada en conjunto a la llegada de n trenes unidos al sistema, donde $\mathcal{J}_{arr}$ es el conjunto de todas las llegadas en conjunto del sistema. Análogo para una salida en conjunto, $\mathcal{J}_{dep}$ es el conjunto de todas las salidas en conjunto del sistema. El orden de las uniones es importante para saber qué tren corre primero, estos se especifican en listas que indican el orden de los trenes.

Así como pueden ensamblarse los trenes, también existe la operación inversa, que es separar los trenes (*disjunction*), como se muestra en la Figura 2.3.

Estas operaciones tienen costos asociados, $junCost$ y $disjCost$, para uniones y separaciones respectivamente, y necesitan un tiempo $junTime$ para la unión y $disjTime$ para la separación.

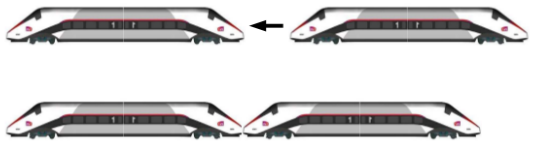


Figura 2.2: Unión de dos trenes

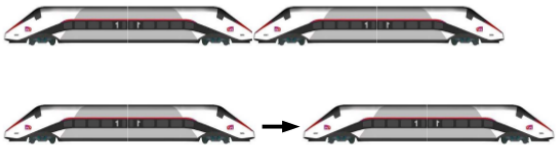


Figura 2.3: Separación de dos trenes

Mantenimiento

Los trenes deben pasar por un proceso de mantenimiento para poder cumplir con las condiciones de seguridad y comodidad. En este problema se presentan dos tipos de mantenciones relativas al tiempo que puede correr un tren t después de una mantención, y a la distancia que recorre t después de la mantención.

Una mantención de tipo D permite restaurar el DBM (*Distance Before Maintenance*) de un tren t a $maxDBM_t$, mientras que una mantención de tipo T permite restaurar el TBM (*Time Before Maintenance*) de un tren t a $maxTBM_t$. Las operaciones de mantenimiento sólo pueden ser realizadas en el edificio de mantenimiento especificado en el problema, para cada tipo de mantención. Un tren puede realizar

a lo más una mantención de cada tipo, si realizan más de una no tendrá efecto pues TBM y DBM ya fueron restauradas a sus valores máximos.

La duración de las operaciones de mantenimiento dependerá de la categoría de los trenes: para trenes de categoría $c \in \mathcal{C}$ las operaciones de tipo D tendrán una duración de $maintTimeD_c$, y las operaciones de tipo T tardarán un tiempo $maintTimeT_c$.

Por último, el sistema tiene un límite de mantenciones que pueden realizarse diariamente en un edificio de mantenimiento. Este valor es conocido como $maxMaint$.

2.2. Restricciones

Las restricciones definirán la factibilidad de una solución. Una solución es factible sí y sólo sí las restricciones descritas en esta sección son satisfechas.

Estas restricciones tienen que ver con los agendamientos, los tipos de asignaciones, el uso de los recursos y con los trenes que requieren de unión o separación.

Se utilizará la notación $\mathcal{T}^+$ para referirse a los trenes que son parte de la solución.

Propiedades de los agendamientos

- Llegadas: Un agendamiento $sched_t$ para cada tren $t \in \mathcal{T}^+$ debe comenzar cuando el tren entra al sistema.
  - Si hay un tren inicialmente en el sistema, es decir, $t \in \mathcal{T}_\mathcal{I}$, éste entra en el sistema en el tiempo: d_1 $00:00:00$.
- Salidas: Los agendamientos $sched_t$ de cada tren $t \in \mathcal{T}^+$ terminan cuando el tren sale del sistema.
  - Si un tren no se asigna a una salida, se asume que la salida del sistema ocurre al final del horizonte de tiempo, es decir, en d_{nbDays} $23:59:59$.
- Operaciones de mantenimiento: Una planificación $sched_t$ puede incluir operaciones de mantenimiento. Éstas operaciones son realizadas en un recurso de infraestructura exclusivo, donde el tipo debe ser consistente con el tipo de mantenimiento a realizar (tipo T o tipo D). A lo más un mantenimiento de cada tipo está permitido para cada tren.
- Operaciones de unión y separación: Una planificación $sched_t$ puede incluir operaciones de unión o separacipon de trenes. Estas operaciones deben ser realizadas en ciertos recursos de infraestructura permitidos y en los tiempos indicados para cada una de estas operaciones. Dos o más trenes pueden unirse o separarse sólo si están disponibles en el instante requerido.

Asignaciones

- A lo más un tren asignado a cada salida: No es posible asignar más de un tren a una misma salida (excepto si se habla de salidas en conjunto, las cuales son cubiertas por uniones de trenes).
- DBM requerido para una salida: Para asignar un tren a una salida, el DBM remanente debe ser mayor o igual que el DBM requerido o en caso de ser menor se debe realizar un mantenimiento de tipo D para restaurar el DBM a su valor máximo. Acá el tiempo de la salida debe considerar el tiempo requerido para realizar el mantenimiento, ya que de poder realizarse mantenimiento, éste debe ser realizado antes de la salida.
  - Si el tren t realiza un mantenimiento de tipo D, entonces:
    $$depTrain_d = t \Rightarrow maxDBM_t \geq reqDBM_d$$
  - Si t no se realiza el mantenimiento de tipo D, entonces:
    $$depTrain_d = t \Rightarrow remDBM_t \geq reqDBM_d$$

  Además, si una llegada tenía una salida asociada *linkedDep* y ésta fue realizada, el valor del DBM debe actualizarse dependiendo del tren que realizó la salida asociada.
  $$remTBM_{t''} = remDBM_{t'} - reqDBM_{d'}$$
- TBM requerido para una salida: Análogamente, la asignación de un tren a una salida, implica que el TBM remanente debe ser mayor o igual que el TBM requerido para una salida.
  - Si el tren t realiza un mantenimiento de tipo T, entonces:
    $$depTrain_d = t \Rightarrow maxTBM_t \geq reqTBM_d$$
  - Si t no se realiza el mantenimiento de tipo T, entonces:
    $$depTrain_d = t \Rightarrow remTBM_t \geq reqTBM_d$$

  Además, si una llegada tenía un *linkedDep* (es decir, una salida asociada) y ésta fue realizada, el valor del TBM debe actualizarse dependiendo del tren que realizó la salida asociada.
  $$remTBM_{t''} = remTBM_{t'} - reqTBM_{d'}$$
- Salidas y categorías compatibles: Las categorías de los trenes deben ser compatibles con la salida asignada.
- Uniones y categorías compatibles: Las categorías de trenes deben ser compatibles si van a realizar una salida en conjunto.

Uso de recursos

Si bien este problema no tinene que ver con la asignación de recursos, hay que asegurarse de respetar los tiempos necesarios en los que un tren debiese permanecer en un recursos, especialmente si se quiere que el *output* de este problema sirva como un *input* para el problema de rutas de trenes.

- Máximo uso de la plataforma: Un tren no puede estar en una plataforma más de $maxDwellTime$. Si la plataforma es usada para una llegada, ésta no puede estar más de un tiempo $maxDwell_a$. Para una salida, no debe excederse el tiempo $maxDwell_d$. Si hay una llegada seguida de una salida, el tiempo máximo que puede estar un tren en la plataforma es el máximo entre ambos tiempos, es decir máx $(maxDwell_a, maxDwell_d)$.

- Tiempo mínimo de uso: Existe un tiempo mínimo de uso de recursos $minResTime$, en nuestro problema, esto quiere decir que entre un tiempo de salida y un tiempo de llegada debe haber al menos un tiempo $minResTime$

- Capacidad de los recursos de mantenimiento: Pueden haber a lo más $maxMaint$ operaciones de mantenimiento cada día.

$$\sum_{t \in \mathcal{T}^+} maintD_{t,i} + maintT_{t,i} \leq maxMaint, \forall i \in \{1, nbDays\}$$

  donde $maintD_{t,i}$ indica si el mantenimiento de tipo D en el día i, y de igual manera, $maintT_{t,i}$ representa el mantenimiento tipo T en el día i.

Trenes ensamblados

- Salidas en conjunto: Cuando dos o más trenes son asignados a una salida en conjunto, entonces deben ser ensamblados al menos en un tiempo $minAsbTime$ antes del tiempo de salida.

- Llegadas en conjuto: Después de la llegada de los trenes ensamblados, los trenes deben separarse si es que van a cubrir salidas que no son en conjunto.

2.2.1. Objetivos

El objetivo principal de este problema es minimizar los costos asociados con el manejo de llegadas y salidas de trenes. Se define la función objetivo f del problema presentado en este trabajo, como la suma ponderada de los siguientes costos:

- Costo de salidas sin cubrir f^{uncov}
  Corresponde al costo de no cubrir una salida, es decir, de no asignar ningún tren a la salida.

$$f^{uncov} = d \in \mathcal{D}; depTrain_d = O \tag{2.1}$$

- Reuso de trenes no satisfecho f^{reuse}
  Si no se satisface una condición de reuso $u \in \mathcal{U}$ se considera el costo $reuseCost$

$$f^{reuse} = \sum_{\substack{u \in \mathcal{U} \\ depTrain_{dep_u} \neq arrTrain_{arr_u}}} reuseCost \tag{2.2}$$

- Costo de sobremantenimiento f^{maint}
  Con el fin de evitar el sobremantenimiento, la diferencia entre los valores máximos y los requeridos asociados a mantenimiento son penalizados. Aquí los costos $remDCost$ y $remTCost$ están en kilometros y segundos respectivamente.

$$f^{maint} = \sum_{\substack{e \in \mathcal{E} \\ y_e=\texttt{BegMaintenance} \\ \text{and } c_e=\text{“D”}}} remDCost \cdot remDBM_{t_e} + \sum_{\substack{e \in \mathcal{E} \\ y_e=\texttt{BegMaintenance} \\ \text{and } c_e=\text{“T”}}} remTCost \cdot remTBM_{t_e} \tag{2.3}$$

  donde $remDBM_{t_e}$ corresponde a la diferencia entre el máximo DBM y el requerido, mientras $remTBM_{t_e}$ es la diferencia entre el máximo TBM y el requerido.

- Costo de las operaciones de unión y separación f^{jun}
  Si un conjunto de trenes se une o se separa se aplican los costos respectivos:

$$f^{jun} = \sum_{\substack{e \in \mathcal{E} \\ y_e=\texttt{BegJunction}}} junCost + \sum_{\substack{e \in \mathcal{E} \\ y_e=\texttt{BegDisjunction}}} disjCost \tag{2.4}$$

Finalmente, la función objetivo puede resumirse como:

$$f^* = f^{uncov} + f^{reuse} + f^{maint} + f^{jun} \tag{2.5}$$

Es importante remarcar, que en la definición del DMP dada en [12] proponen como función objetivo solo el costo de salidas sin cubrir y de no cumplimiento de los reusos, como se puede observar en la ecuación 2.6, sin embargo en este trabajo se consideran también los dos últimos con el fin de abordar más aspectos relacionados a la gestión de trenes.

$$f = f^{uncov} + f^{reuse} \tag{2.6}$$

2.3. Resumen del Capítulo

En este capítulo se presentó el modelo del DMP. El DMP posee una gran cantidad de componentes y restricciones que se presentan en la gestión ferroviaria. Si bien en este problema no se realiza la asignación de recursos ni rutas de los trenes, es necesario conocer las restricciones de dichos recursos para tomar en cuenta los tiempos que deben haber entre el ingreso de los trenes al sistema y las salidas.

En este modelo se consideran todas las componentes, al igual que en el problema RSUM del desafío ROADEF, incluyendo todos los costos de la función objetivo que se utilizan en la asignación.

En base a este modelo se trabajará con una representación adecuada, que permita tomar en cuenta todos los aspectos del problema para plantear la solución.

Capítulo 3

Estado del Arte

En este capítulo se presenta el Estado del Arte de este trabajo, donde se estudian distintos problemas de gestión de trenes y sus soluciones, incluyendo al problema presentado en la competencia ROADEF/EURO 2014 de donde nace el DMP.

3.1. Problemas de Gestión de Trenes

En la literatura existe una amplia variedad de problemas que tienen que ver con la gestión de trenes, con diferentes objetivos, como maximizar la comodidad de los pasajeros, minimizar costos o mejorar los tiempos de respuesta ante eventos inesperados.

En [6] se define el *Train Scheduling Problem*[1] y se demuestra su NP-completitud. Se menciona como un problema que a menudo requiere ser resuelto en tiempo real, por lo que es necesario una heurística rápida que permita obtener una buena solución factible en un número finito de pasos.

En [5] resuelven el problema de enrutamiento y agendamiento de las vías usando *Simulated Annealing* (SA) y lo comparan con un algoritmo genético (AG). En este estudio mencionan que este tipo de problema es uno de los más difíciles dentro de la optimización combinatoria, y concluyen que es mucho más fácil para este tipo de problemas la implementación de SA que de algoritmos genéticos, y que los resultados obtenidos con SA son mejores.

En [22], se resuelve el problema de Manejo de Desviación de tráfico de trenes mediante *Tabu Search* (TS) y SA. En esta investigación se busca resolver el problema de manejar el tráfico en las vías cuando hay perturbaciones, donde debe hacerse un re-agendamiento. El problema se separa en diferentes niveles: El nivel superior se encarga de la orden que se da en el encuentro y adelanto de trenes en tramos de la pista, mientras que el nivel inferior determina las horas de inicio y fin para cada tren. SA y TS se ocupan por separado en el nivel superior.

[1] Problema de Agendamiento de Trenes

Chang & Chung Min Kwan en [9] aplican técnicas de la computación evolutiva al problema de optimización multiobjetivo de la organización de trenes[2], que se basa en los trenes *Medium sized mass rapid transit* (MRT), que son medios de transporte masivos rápidos y de tamaño medio en Singapur. Los objetivos de este problema son maximizar el nivel de comodidad de los pasajeros y minimizar los costos. Estos objetivos entran en conflito y son afectados por distintos factores, como la frecuencia de los trenes, los tiempos de espera de los pasajeros en las estaciones y la comodidad del viaje. Para abordarlo, definen un algoritmo genérico: a partir de éste utilizaron 3 técnicas a comparar: AG, *Particle Swarm Optimization* (PSO), y *Differential Evolution* (DE). En sus resultados obtuvieron que los algoritmos basados en técnicas evolutivas funcionaron bien con el problema y sus instancias, realizando un análisis en dos partes, primero enfocándose en un objetivo, y luego en ambos objetivos: minimizar los costos de operación y minimizar el nivel de incomodidad de los pasajeros. Las instancias de prueba se basaron en el estudio de 26 estaciones en la hora punta de la mañana. Estos mismos autores en [15] presentan una heurística basada en Algoritmos Evolutivos para el problema de optimizar el agendamiento de los trenes de Singapur. Este problema se identifica como altamente no lineal y con muchas variables conectadas entre sí, donde los objetivos son los mismos que en el estudio anterior: maximizar comodidad de los pasajeros y disminuir los costos operacionales. Los autores tuvieron resultados exitosos al compararlos con el sistema utilizado para la gestión de estos trenes, y se propone utilizarlo para la gestión de otros medios de transporte, como aviones y buses.

En [8], Chang & Sim proponen un algoritmo genético para optimizar la circulación de trenes (también MRT) utilizando un sistema de operación automática de los trenes[3]. Los aspectos que se consideran en este problema son los horarios del tren, la carga de pasajeros y el voltaje necesario para las pistas. Cada sistema de control tiene la información necesaria para el tren, antes de que el tren salga hacia una estación designada. Esa información es codificada para poder ser manejada en los cromosomas.

Kwan et al. en [16] presentan un algoritmo co-evolutivo para el problema de horarios de trenes. En este estudio se plantea el caso de Reino Unido, donde existen muchas empresas que operan los trenes de manera independiente y comparten la capacidad limitada de la red ferroviaria, además de un único operador que debe coordinar los trenes en la red. Presentaron una aproximación para la generación de los horarios de los trenes, descomponiendo el problema en módulos. El algoritmo fue probado con 7 instancias generadas, donde cada instancia poseía 150 trenes a ser agendados en un período de 2 horas, de los cuales 50 ya tenían sus horarios de salida agendados. Si bien la generación de estos horarios fue exitosa, se puede mejorar el modelo, considerando las ubicaciones de las plataformas y las vías, similar al problema de la competencia ROADEF.

[2] *Evaluation of Evolutionary Algorithms for Multi-objective Train Schedule Optimization*

[3] *Automatic Train Operation ATO*

En [20], presentan un Algoritmo Evolutivo (AE) basado en permutación, que se basa en una heurística *"semi-greedy"*, que reconstruye de forma gradual el horario de trenes mediante la inserción de un tren después de la permutación. El objetivo del AE es encontrar la permutación que genere un agendamiento óptimo. Para las instancias se usaron datos reales proporcionados por la SNCF, donde cada instancia contaba con cerca de 500 nodos y 541 trenes, lo que produce un total de 1 millón de variables y 2 millones de restricciones. El algoritmo propuesto obtuvo buenas soluciones para una instancia grande y compleja, comparado con CPLEX obtuvo mejores resultados en términos de tiempo de ejecución y calidad de la solución, pero su robustez aún no se ha demostrado porque debe probarse con más instancias.

En [23] se estudia el problema de gestión de trenes en la red ferroviaria holandesa, donde el objetivo es desarrollar un algoritmo que pueda construir planes de derivación durante un período de 24 horas en base al diseño del servicio: un horario de trenes de llegada y salida y una lista de tareas. Se presenta un modelo de búsqueda local basado en SA, donde los resultados obtenidos son comparados con el *software* usado por *Nederlandse Spoorwegen*, la principal compañía operadora de servicios ferroviarios de pasajeros de los Países Bajos, llamado OPG.

Para evaluar el desempeño del algoritmo, se generaron casos de prueba artificiales y otros reales, basados en las distribuciones de horarios y servicios realizados en un día de semana normal. La heurística presentada en este estudio fue aplicada de manera exitosa para los dos tipos de casos de prueba, donde el SA propuesto tuvo mejor desempeño que OPG para las instancias creadas y fue capaz de planificar más trenes que el OPG. Como resultado de esta investigación, NedTrain incorporó este enfoque de búsqueda local en el software utilizado para calcular la capacidad del sitio de servicio.

En [13], se estudian distintos enfoques usados en problemas de gestión de trenes en general, incluyendo la descripción del DMP, realizada por el mismo autor.

En [17] proponen un algoritmo *Branch-and-Price* para los reagendamientos de trenes en sistemas ferroviarios. Cuando ocurre una interrupción en un sistema ferroviario es necesario hacer un re-agendamiento, esto se transforma en un problema de optimización para los operadores de vías, el cual, a pesar de su complejidad, muchas veces es resuelto de forma manual por los operadores. Con este enfoque, la formulación se extiende más fácilmente para manejar ciertas familias de restricciones, como por ejemplo las restricciones de mantenimiento. Para medir la calidad de las soluciones, se usan instancias basadas en datos reales proporcionados por el operador ferroviario suburbano en Copenhague, DSB S-tog. La metodología propuesta se puede utilizar para la planificación táctica, donde encuentra resultados cercanos al óptimo en tiempos de CPU del orden de minutos.

3.2. Problemas de Gestión de Trenes: Desafío ROADEF

En el desafío ROADEF/EURO 2014 *Trains don't vanish!*, se presenta el problema de manejo de trenes en ferrocarriles (RSUM, por sus siglas en inglés). El desafío estaba abierto a todo público, en particular a los investigadores jóvenes. Se definieron dos categorías: *junior*, donde los investigadores jóvenes pueden enfrentarse a un problema complejo de optimización industrial, y *senior*, donde investigadores más experimentados pueden poner en práctica sus conocimientos y demostrar su experiencia en problemas prácticos [3].

El problema se centra en la gestión de trenes entre sus llegadas y salidas a las plataformas, pasando por diferentes recursos de infraestructura.

En la competencia participaron 13 grupos en la categoría *junior* y 22 grupos en la categoría *senior* [1], con integrantes de diferentes países, que se distribuyen como se muestra en la Figura 3.1, que se encuentra en el sitio del desafío [3].

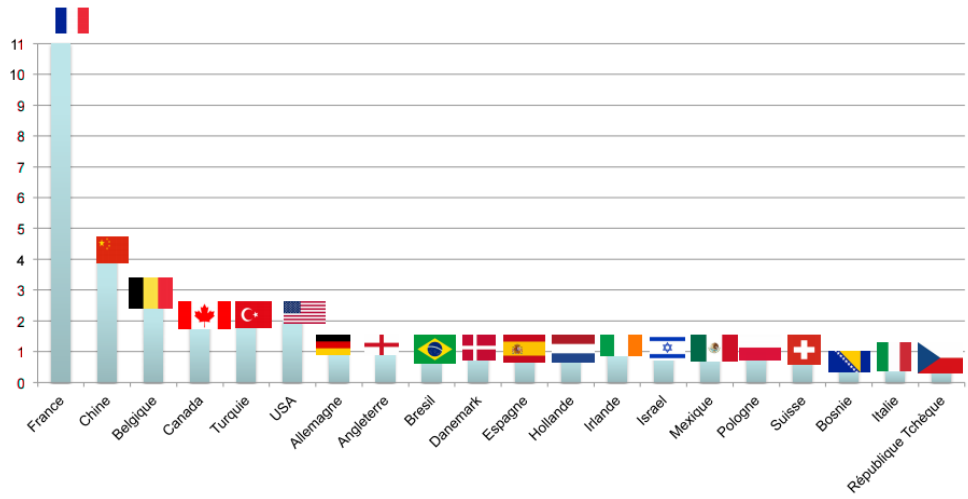


Figura 3.1: Competidores en Desafío ROADEF 2014

Estos participantes debieron seguir una serie de reglas publicadas en un documento oficial en su sitio web [4].

Los resultados finales de los ganadores se encuentran publicados en el sitio [2], donde hubo un único ganador para la categoría *senior* y tres lugares en la categoría *junior*.

En cuanto a los algoritmos utilizados por los competidores no hay mayor información en el sitio de la competencia, sin embargo, algunos de ellos han publicado sus trabajos, por lo que a continuación se revisa el estado del arte de los algoritmos que resuelven el Desafío ROADEF.

En [7], Hadrien Cambazard y Nicolás Catusse (participantes de la categoría *senior)*, proponen una metodología fuertemente basada en el modelado de programación entera mixta *(MIP: Mixed-Integer Programming)* y un modelo de programación por restricciones *(CP: Constraint Programming)*. En esta investigación se prefiere trabajar más el modelo que un algoritmo que lo resuelva, puesto que los autores consideran que una ingeniería robusta es más fácil de lograr apoyándose en modelos, que del código dedicado. El modelamiento se centra en dos ideas principales: los límites de mantenimiento diarios y las salidas en conjunto. La idea es separar el problema en dos etapas de decisión:

- Manejar la asignación de los trenes asociados a llegadas a las salidas en la plataforma elegida. Esta decisión es realizada mediande un *MIP solver*. En este punto se consideran los mantenimientos diarios y llegadas en conjunto.
- Se considera que las rutas de los trenes se puede calcular sin la estricta necesidad de realizar el paso anterior, donde se hacen las asignaciones. Desde el punto de vista de ingeniería, así como la eficiencia, consideran que CP es la mejor tecnología para realizar esta etapa y generar la ruta dentro de los límites de tiempo.

Estos pasos se resumen en el esquema presentado en la Figura 3.2.

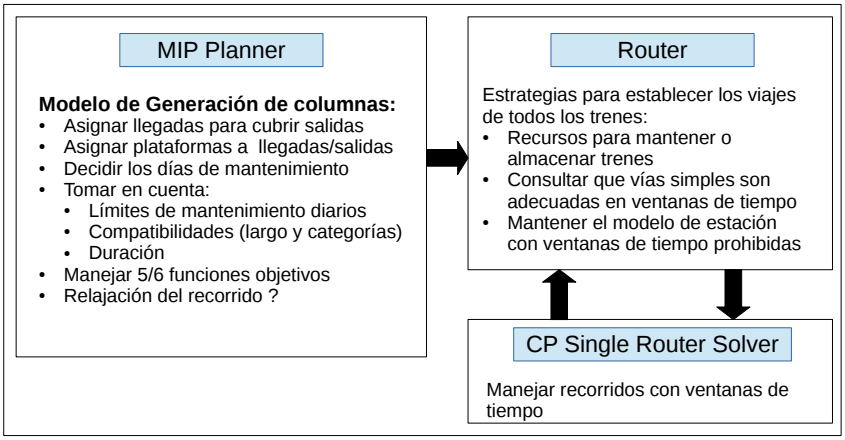


Figura 3.2: Esquema del modelo basado en MIP y CP

Los autores de este estudio concluyen que su metodología propuesta sólo se mantiene si las rutas se consideran separadas de la asignación, pero sigue siendo una pregunta abierta si se considera como restricciones duras los conflictos en los grupos de vías. De todos modos se propone una relajación fuerte en la decisión de las rutas para ser incluída en la parte de asignación del modelo.

En [11], Haahr & Bull (segundo lugar en la categoría *junior* de la competencia), presentan una heurística matemática, que se basa en métodos exactos y el método heurístico basado en *Simulated Annealing* (SA), siguiendo el esquema que se presenta en la Figura 3.3

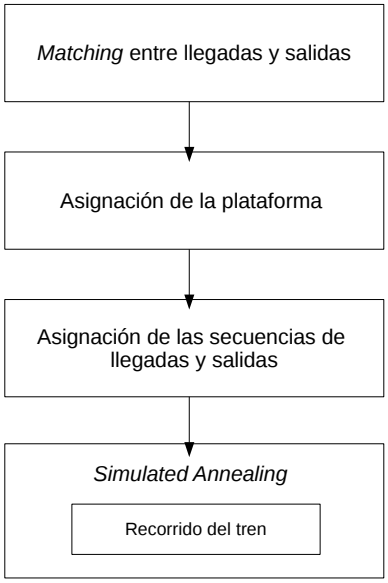


Figura 3.3: Heurística Matemática utilizada para resolver el RSUM

El núcleo de esta propuesta es *Simulated Annealing* (SA) que iterativamente destruye y construye soluciones en rutas de trenes aleatorias. En la Figura 3.3 se pueden ver todos los pasos y elementos principales del *framework* propuesto. En el primer paso, llegadas y salidas son emparejadas con el fin de asegurar los mejores *matchings* (o emparejamientos) y reducir el número de cancelaciones (salidas sin tren que las cubra). El siguiente paso es escoger un espacio en la plataforma que se asignará a esas llegadas y salidas, seleccionando, mientras sea posible, entre las plataformas que tienen preferencia. Luego se asignan las pistas (*Track Groups*) según las secuencias de llegada y secuencias de salida de modo que no existan conflictos con el consumo impuesto de los recursos. El siguiente paso verifica que no se traslapen los trenes en los espacios de mantenimiento. Por último, mediante SA, iterativamente se remueven y se rearman las rutas, reasignando un grupo de trenes con otros *matchings*. Como el problema presentado es tan difícil, los autores de este *paper* han decidido modificar el problema y no considerar las llegadas y salidas en conjunto, debido a la falta de tiempo y al alto costo de implementar el *framework* de solución.

En [18], Mulders y Scholliers (competidores en la categoría *junior*), en su tesis, estudian este problema y cómo fue abordado en la competencia, y destacan tres enfoques:

1. *Matching:* Donde se busca maximizar los calces entre las llegadas y salidas, que comúnmente se ha resuelto mediante MIP, pero proponen utilizar propiedades de los grafos bipartitos. Luego se combina con otras técnicas, como SA para completar las soluciones.

2. Programación con restricciones (CP): Utilizan este paradigma para dos situaciones: como parte del problema de *matching* y luego para resolver el problema completo.

3. *Greedy*: Donde se divide el horizonte de tiempo en días independientes y se realizan agendamientos para cada uno de esos días.

Los dos primeros enfoques incluyen una mezcla de técnicas, como es el caso de utilizar MIP+SA, o MIP + CP, para resolver los distintos subproblemas.

Luego de estudiar los diferentes enfoques, proponen su propio algoritmo, también enfocándose en resolver primero el problema de *matching*, ya que así se minimiza el número de las salidas sin cubrir, que es una de las componentes más relevantes de la función objetivo, y concluyen que al enfocarse en resolver el primer subproblema se pueden encontrar mejores soluciones para el problema completo.

3.3. Problemas de Gestión de Trenes: DMP

Haahr y Bull en [12] presentan cómo resolver el DMP con métodos exactos. En el estudio se presenta formalmente este subproblema y se demuestra que es *NP-Hard*.

Primero, se presenta una formulación basada en Programación Entera Mixa (MIP), con la cual resulta difícil resolver la mayoría de las instancias del problema, debido a problemas de memoria. Para solucionar este problema, proponen el método de generación de columnas (GC), el cual ayuda a manejar el caso de las llegadas con salidas enlazadas, el cual se contrasta con el méotodo de no utilizar generación de columnas (No GC), pero ignorar todas aquellas salidas que estén enlazadas a alguna llegada, lo cual repercute en el valor de la función objetivo. Para evaluar esta propuesta, ejecutan cada instancia con un límite de tiempo máximo de 20 minutos. En el caso de GC, se ocupan los primeros 10 minutos en la generación de columnas, y posteriormente se resuelve el modelo con `CPLEX`. El fin de aplicar esta heurística de generación de columnas es hacer más rápido el desempeño de la resolución del problema, y no resolver de manera exacta la optimización, los resultados obtenidos por estos enfoques se presentan en la Tabla 3.1.

Para calcular los costos de los *matching*, los autores utilizan la parte de la función objetivo asociada a los costos de salidas sin cubrir y costos de reusos. En cuanto a seleccionar las parejas de salidas y trenes, toma en cuenta las restricciones de tiempo entre salida y llegada, tiempo necesario para realizar mantenimientos, restricción de número máximo de mantenimientos diarios y compatibilidades entre categorías

y salidas, pero deja de lado aspectos que podrían considerarse en esta parte del problema, como son los costos de unión y separación, o costos de sobremantenimiento.

Inst.	MIP		GC			No GC	
	f	t[s]	#columns	f	t[s]	f	t[s]
B1	-	-	278229	**93200**	1207	254600	1213
B2	-	-	383937	**61000**	1200	228000	1201
B3	**0**	1199.4	0	**0**	26	**0**	5
B4	-	-	300143	**209300**	1162	379000	333
B5	-	-	248919	**221100**	1203	370900	87
B6	-	-	285498	**211800**	1201	379000	341
B7	**3400**	31.3	30245	4300	78	73200	81
B8	**3400**	31.0	27942	4400	86	73200	81
B9	-	-	277485	**252900**	1207	456700	1215
B10	**1000**	56.0	5908	2000	5	42400	26
B11	-	-	171188	**80600**	1215	258000	1210
B12	43900	1204.5	71442	**14300**	1206	135600	1205

Tabla 3.1: Comparación de desempeño entre enfoque MIP y enfoques usando GC y No GC (sin permitir salidas enlazadas)

De este estudio concluyen que las instancias propuestas en el desafío son demasiado difíciles de resolver con técnicas completas, y sugieren que no es posible obtener una solución satisfactoria para el problema completo, ya que no es posible encontrar soluciones que no posean salidas sin cubrir con las instancias entregadas en la competencia.

Haahr, en [13], estudia distintos problemas relacionados a la gestión de trenes y los enfoques utilizados, se incluye la descripción de DMP y el enfoque de generación de columnas presentado anteriormente, donde se destaca adicionalmente la importancia de la búsqueda de instancias que se pueden resolver sin salidas sin cubrir.

3.4. Resumen del Capítulo

En este capítulo se presentó una revisión de problemas similares al de gestión de trenes de ROADEF existentes en la literatura, que poseen diferentes componentes y objetivos. Es importante analizar cómo se han abordado problemas similares a la hora de diseñar un algoritmo.

El DMP, es un problema reciente, por lo que en la literatura ha sido menos estudiado que otros problemas más clásicos como los presentados en la sección 3.1.

En general el problema definido en la competencia ROADEF, se ha enfrentado como la división de dos subproblemas, uno que tiene que ver con las asignaciones de entradas/salidas (*matching*), que corresponde al problema a resolver en este trabajo,

y el subproblema de las rutas, que maneja todo el itinerario de los trenes entre estas llegadas y salidas. En esta investigación se pudo apreciar las principales dificultades de los competidores al momento de abordar el problema completo, donde concluyeron que no es posible encontrar soluciones sin cancelaciones para este problema con las instancias dadas en la competencia[12].

En el siguiente capítulo se presenta el diseño del algoritmo para resolver el DMP.

Capítulo 4

Diseño de la Propuesta: G-DMP

A diferencia de lo que se observa en la literatura, donde los autores resuelven el subproblema DMP utilizando programación entera mixta, se propone encontrar los *matchings* mediante un algoritmo basado en GRASP[1] que considere todos los elementos del problema. Se propone una técnica incompleta que sea capaz de considerar todos los costos involucrados en el problema, ya que hasta ahora, las técnicas propuestas son muy costosas computacionalmente y para disminuir esos costos tienen que hacer grandes cambios en el modelo, como relajar restricciones o eliminar aspectos importantes del problema como son las llegadas y salidas en conjunto, no resolviendo así el problema original, como se menciona en [11].

GRASP es una metaheurística de construcción que se ha aplicado con éxito en la resolución de problemas de optimización combinatoria. Busca construir soluciones diversas y aproximadas que le permitan a un algoritmo de búsqueda local mejorar la calidad de estas soluciones. Esta técnica fue propuesta por Resende y Feo en [10].

4.1. GRASP

GRASP se divide en tres fases:

Fase de Pre-procesamiento:

En esta fase se detecta si hay subestructuras que formen parte de la solución óptima, para así fijarlas y reducir los tiempos de ejecución, al tratar estas subestructuras se trata carga adicional.

Fase de Construcción:

En la fase de construcción, se van escogiendo diferentes candidatos por medio de una función miope. Los mejores candidatos se encuentran en una lista ordenada

[1] *Greedy Randomized Adaptive Search Procedure*

según los beneficios que entreguen los elementos a escoger, y donde serán escogidos de manera aleatoria (*randomized*). Esta lista se conoce como *restricted candidate list* o lista restringida de candidatos (LRC).

Fase de Post-procesamiento:

Se utiliza una técnica de búsqueda local que mejore la calidad de la solución construída en la fase anterior.

Al completar las fases, se guarda la solución obtenida y se realiza otra iteración, así hasta que se cumpla el criterio de término y manteniendo la mejor solución que se encuentre hasta el momento.

En el Algoritmo 1 se presenta una estructura general de un GRASP:

Algoritmo 1 GRASP

```
while !condición de término do
  preProc(solución)
  construcción(solución)
  postProc(solución)
end while
return Mejor Solución
```

Las componentes del GRASP propuesto se presentan a continuación:

4.2. Algoritmo Propuesto G-DMP

A continuación se presentan las componentes principales del algoritmo propuesto.

4.2.1. Representación

La representación consiste en un vector de vectores de *string*, de tamaño dos, que contiene siempre en la primera posición el índice de la salida a cubrir y en la segunda posición el índice del tren que se está utilizando (ya sea un tren de una llegada o un tren que se encuentra inicialmente en el sistema), como en el ejemplo presentado en la Tabla 4.1.

Dep_3	Arr_1
Dep_5	Arr_3
Dep_6	Arr_4
Dep_7	$Train_{10}$

Tabla 4.1: Ejemplo de un vector de *matching*

Esta representación permite insertar fácilmente un nuevo *matching* cada vez que se encuentra, y al conocer los índices de la salida y su tren respectivo, se puede acceder a toda la información necesaria (compatibilidad de la salida, compatibilidad con los recursos, categoría del tren, si es necesaria una unión o separación y si se requiere de un mantenimiento de tipo T o D, además de los tiempos entre los que ocurren los eventos).

Esta representación tiene la ventaja de ser rápida de revisar, y además cubre las restricciones de a lo más un tren asignado para cada salida.

4.2.2. Fase de Pre-procesamiento

En esta implementación de GRASP no se consideró una fase de pre-procesamiento, ya que no se identificó alguna información a priori que puede ser usada.

4.2.3. Función Miope

La función miope propuesta está orientada a la minimización de los costos de salidas sin cubrir y reusos, por lo que se generará una lista de trenes candidatos para cubrir las salidas correspondientes. Para que un tren sea considerado como candidato, se revisan las siguientes restricciones en orden:

- El tren candidato es compatible con la salida
- El tiempo de llegada del tren debe ser menor que el tiempo de salida (o es un tren inicial en el sistema).
- El tiempo entre la salida y llegada es mayor que el tiempo $minResTime$.
- Si no necesita mantenimiento o si necesita y además no se ha llegado al límite de mantenimientos diarios, y el tiempo entre salida y llegada es mayor que el tiempo de mantenimiento.
- Si es que se realizará una salida en conjunto, el siguiente tren que cubra la salida unida tiene que ser también compatible, y si la llegada es en conjunto, que sea posible la separación de estos trenes para cubrir sus respectivas salidas.

Si se cumplen todas estras restricciones entonces el tren candidato pertenecerá a la lista restringida. Siempre se realiza en orden de tiempos, es decir que la diferencia entre el tiempo de salida y llegada sea lo menor posible, pero siempre y cuando no viole las otras restricciones de tiempo de uso. Esto minimiza el costo general de permanencia en la plataforma, por lo que los primeros en la lista tendrán menor costo. Además, siempre se guardan primero en la lista los trenes de la lista de preferencias de reutilización, siempre y cuando cumplan con las restricciones mencionadas. Al final se agrega la opción de que ningún tren cubra esa salida, con el fin de que pueda ser reparada en la fase de post-procesamiento.

El detalle de la función miope se muestra en el Algoritmo 2.

Algoritmo 2 Función Miope

```
for tren in trenes_dispobibles do
  for salida in salidas_sin_cubrir do
    while rcl.size() ≤ maxLen do
      if reuso(tren,salida) then
        rcl.push_back(tren,salida)
      else if factible(tren) then
        salidas_sin_cubrir -= salida
        trenes_disponibles -= tren
        rcl.push_back(salida, tren)
      end if
    end while
  end for
end for
```

4.2.4. Fase Constructiva

En esta fase se busca asignar a una salida sin cubrir algún tren que no haya sido utilizado. Para esto se tiene una lista restringida de candidatos (LRC) que, dado un largo máximo, guarda diferentes trenes que podrían cubrir esa salida, seleccionados con la función miope descrita en la Sección 4.2.3. Se escoge de manera aleatoria uno de estos candidatos, y si es que corresponde a algún tren, se marca la salida como cubierta y se inserta la salida con el tren en el vector de *matching*. Además, esa salida debe eliminarse del vector de salidas sin cubrir, y el tren también se elimina del conjunto de trenes sin utilizar, dada la restricción que un tren puede cubrir a lo más una salida. Además, se reduce el costo total, y se aplican los otros costos asociados.

A diferencia del modelo presentado en [12], este modelo considera las llegadas y salidas en conjunto, por lo que en estos casos la plataforma a utilizar es la misma, pero se define el orden de los trenes en la unión.

Como una salida debe cubrirse por a lo más un tren, puede ocurrir que no sea cubierta (si es que no se repara en la fase de post-procesamiento), en tal caso no se descuenta el costo.

Al final de esta fase se tiene el vector con los *matchings* actuales del sistema, y se pasa a la siguiente fase.

El detalle de la fase constructiva se presenta en el Algoritmo 3.

4.2.5. Fase de Post-procesamiento

Para el post-procesamiento se propone un algoritmo tipo *Hill Climbing* (HC). HC es un algoritmo reparador, que busca mejorar una solución inicial de manera iterativa aplicando movimientos en la solución. En cada iteración se intenta obten-

Algoritmo 3 Fase Constructiva

```
for salida_actual in salidas_sin_cubrir do
  función_miope()
  candidato = LRC[random]
  if candidato != 0 then
    costo_total -= uncovCost
    costo_total += costos_asociados
    salidas_sin_cubrir -= salida_actual
    trenes_sin_usar -= candidato
    matching(salida_actual, candidato)
    MATCH.push_back(matching)
  end if
end for
postProcesamiento(MATCH)
```

er una solución nueva. Existen dos versiones de este algoritmo: *Hill Climbing* con "alguna mejora"[2] y *Hill Climbing* con "mejor mejora"[3]. Para la versión con alguna mejora, el algoritmo genera el vecindario hasta encontrar una solución que sea mejor a la actual, y la reemplaza, mientras que en la "mejor mejora" se genera el vecindario completo y se reemplaza la solución con la mejor solución encontrada.

En este caso se utiliza un algoritmo tipo *Hill Climbing* con alguna mejora, por lo que no será necesario generar todo el vecindario, sino que se generará hasta que se encuentre un vecino que resulte de mejor calidad. La función de evaluación corresponde a la misma función objetivo completa del problema f^* en la ecuación 2.5.

Estructura de Hill Climbing

Se trabaja en el ámbito de lo factible, por lo que el movimiento debe mantener la factibilidad de la solución generada en la fase constructiva. Se utiliza un movimiento de inserción similar al descrito en [24]. Este movimiento consiste en revisar el conjunto de trenes sin utilizar y la lista de salidas sin cubrir, que ocurran el mismo día de la llegada del tren (a menos que sea un tren inicial en el sistema), luego se calculan los costos parciales (de mantenimiento, unión o separación y si es que no se cumple la preferencia de reuso), y si el costo original menos el costo de salida sin cubrir más los costos parciales es menor, entonces se agrega esa solución al vector de *matching* y se actualiza la solución. Además, para mantener la factibilidad, se debe revisar que el tren que va a cubrir la salida cumpla las siguientes restricciones:

- El tren a utilizar es compatible con la salida

[2]First Improvement
[3]Best Improvement

- El tiempo de llegada del tren debe ser menor que el tiempo de salida (o es un tren inicial).
- El tiempo entre la salida y llegada es mayor que el tiempo $minResTime$.
- Si no necesita mantenimiento o si necesita y además no se ha llegado al límite de mantenimientos diarios, y el tiempo entre salida y llegada es mayor que el tiempo de mantenimiento.
- Si es que se realizará una salida en conjunto, el siguiente tren que cubra la salida unida tiene que ser compatible también, y si la llegada es en conjunto, que sea posible la separación de estos trenes para cubrir sus respectivas salidas.

Por lo tanto, el nuevo *matching* se ingresa si y sólo si se cumplen todas las restricciones mencionadas y el costo de esa nueva solución es menor que el costo anterior. El costo calculado es el costo completo f^*.

En el Algoritmo 4 se presenta esta etapa de post-procesamiento:

Algoritmo 4 Post-Procesamiento con HC

```
solución_actual = solución_inicial
while !condición_de_término do
  for trenes_sin_usar do
    for salidas_sin_cubrir do
      nueva_solución = Insert(solución_actual, salida)
      if f(nueva_solución) <f(solución_actual) then
        solución_actual = nueva_solución
        break
      end if
    end for
  end for
end while
return solución_actual
```

Este algoritmo tiene como único parámetro el largo de la lista restringida de candidatos, ya que se definió el número de iteraciones directamente proporcional a la cantidad de salidas que hay en la instancia. En cada paso se intenta asignar un tren a una salida no cubierta.

4.3. Conclusiones del Capítulo

Se presentó un algoritmo GRASP para resolver el DMP, que usa una representación enfocada en los *matching* entre salidas y llegadas o trenes iniciales en el sistema. Esta representación permite obtener toda la información necesaria del problema, ya que conociendo los índices de los trenes y las salidas se puede deducir el resto de la información. Además, esta representación asegura que se asigne a lo más un tren a cada salida. La solución generada en la fase de construcción es factible, ya que se van revisando las restricciones al minuto de escoger un elemento que sea parte de la solución. Esta factibilidad se mantiene en la fase de post-procesamiento, ya que los movimientos no alteran el resto de la solución, y los elementos a insertar para cubrir una salida, también se escogen tras revisar las restricciones asociadas. Esto es posible, porque tanto en la fase constructiva, con la función miope, como en el movimiento de la búsqueda local se inserta un tren, si y solo si, cumple con que el tren esté disponible, que no requiera mantenimiento, y si lo requiere pueda ser realizado (ya sea porque hay un tiempo suficiente para realizar dicha operación, y porque no se haya alcanzado el límite máximo), si es que hay que unir o separar trenes existe el tiempo suficiente para hacerlo y se cumplen las compatibilidades ente las categorías de los trenes y por último, si el tren a asignar es compatible con la salida.

A diferencia de lo encontrado en la literatura, el algoritmo propuesto trabaja sin alterar el modelo propuesto en la competencia, ya que no necesita relajar restricciones. Además, es capaz de calcular todos los costos que se ven involucrados en el problema.

En el siguiente capítulo se presentan los experimentos realizados y los resultados obtenidos.

Capítulo 5

Experimentación y Resultados para G-DMP

Para poder evaluar la implementación del algoritmo propuesto, que consiste en resolver el DMP utilizando un algoritmo basado en GRASP, se realizan pruebas utilizando instancias generadas aleatoriamente además de las instancias del set B de la competencia ROADEF.

En primer lugar se analizarán las características del algoritmo propuesto, para posteriormente comparar los resultados obtenidos con los resultados presentados en la literatura.

En este capítulo se describirán los experimentos, resultados obtenidos y las conclusiones respectivas.

5.1. Entorno de experimentación

El algoritmo fue desarrollado utilizando el lenguaje C^{++} con el compilador gcc versión 6.4.1 .

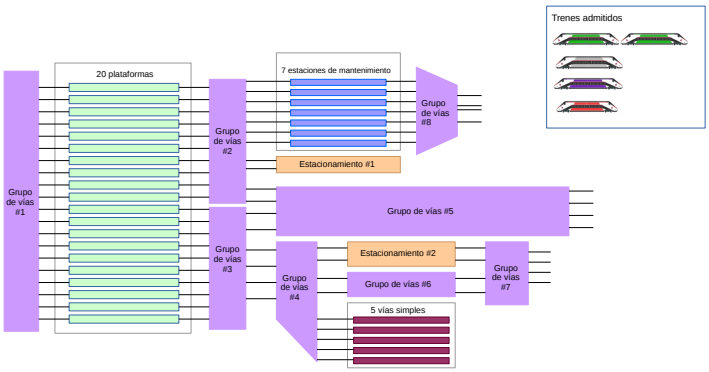
Las pruebas se realizaron en un computador con procesador Intel(R) Core(TM) i5-2400 CPU @ 3.10GHz, con 4GB de memoria RAM y sistema operativo Fedora 25.

5.2. Experimentos

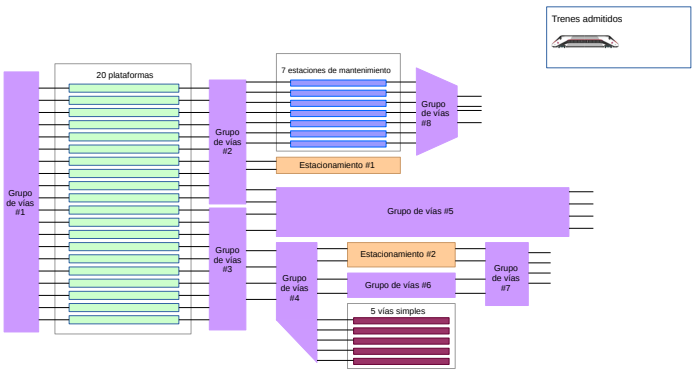
Para evaluar el desempeño del algoritmo, se utilizó un grupo de instancias generadas de manera aleatoria, creadas en un principio para poder hacer un análisis sobre las componentes del algoritmo. Posteriormente, se realizan pruebas con instancias del desafío ROADEF para medir el desempeño del algoritmo y compararlo con los resultados obtenidos en la literatura.

Para el conjunto de instancias generadas aleatoriamente, se tienen 4 conjuntos de 300, 500, 700 y 1000 salidas respectivamente. Cada conjunto posee 10 instancias

con el número de salidas respectivo, sin embargo la cantidad de trenes iniciales varía. Se divide el análisis en instancias denominadas completas, que consideran todos los aspectos del problema descritos en la competencia, e instancias simples, que son instancias donde no hay llegadas ni salidas en conjunto, y hay una única categoría para cada tren. En la Figura 5.1, se puede observar que en cuanto a recursos de infraestructura, en ambos casos se pueden tener los mismos recursos, sin embargo la instancia completa admite trenes de diferentes categorías, además de trenes unidos. Al tener distintas categorías, cada tren podrá pasar sólo por los recursos que sean compatibles con él, mientras que en una instancia simple, donde hay una única categoría y no hay uniones, los trenes podrán pasar por cualquier recurso del sistema. mientras éste esté disponible.



(a) Instancia completa



(b) Instancia simple

Figura 5.1: Esquema Instancias Completas y Simples

En cada grupo de las instancias generadas aleatoriamente se encuentran 5 instancias completas y 5 simples.

En cuanto a las instancias de la competencia ROADEF, se usarán las del set B, que fueron utilizadas por los competidores para evaluar sus resultados finales en la competencia. Estas instancias tienen diferentes tamaños y características, como se observa en la Tabla 5.1, donde se indica el número de llegadas y salidas, el número de llegadas asociadas, que son aquellas cuyo tren fue reutilizado previamente para una salida y que deben actualizarse sus datos si es que ocurrió esa salida previa, el número de salidas y el número de reusos.

Inst.	#Llegadas	#Asociadas	#Salidas	#Reusos	%Asociadas	%Reusos
B1	1235	475	1235	804	38 %	65 %
B2	1235	475	1235	0	38 %	0 %
B3	1235	0	1235	0	0 %	0 %
B4	1780	722	1780	1089	41 %	61 %
B5	2153	720	2153	1089	33 %	51 %
B6	1780	722	1780	1089	41 %	61 %
B7	304	144	304	187	47 %	62 %
B8	304	144	304	187	47 %	62 %
B9	1967	860	1967	1226	44 %	62 %
B10	196	89	196	123	45 %	63 %
B11	1122	486	1122	726	43 %	65 %
B12	570	263	570	377	46 %	66 %

Tabla 5.1: Características de las instancias del grupo B

Se utilizan estas instancias con el fin de medir el desempeño del algoritmo propuesto, y comparar los resultados con los obtenidos en [12], donde utilizan diferentes métodos exactos para obtener resultados para el DMP.

Se define el número de iteraciones proporcional al número de salidas, ya que el algoritmo itera sobre el número de salidas sin cubrir, por lo tanto, si se sobrepasa este número de iteraciones no se encontrarán mejores soluciones.

El número de restarts utilizados en GRASP es de 20, es decir, cada instancia se prueba con 20 semillas diferentes, y se presenta la mejor solución obtenida.

5.3. Resultados Obtenidos

En esta sección se muestran los resultados obtenidos usando G-DMP para resolver el DMP. En primer lugar se hace un análisis del efecto de los parámetros del algoritmo, y luego se realiza la comparación del desempeño del algoritmo propuesto con los existentes en la literatura.

G-DMP requiere de la sintonización de un solo parámetro: el tamaño de la lista restringida de candidatos LRC. Cada lista de trenes candidatos se ordena de la siguiente manera: primero, aquellos que reducen el costo de reuso (aquellos que

pertenecen a la lista de preferencias de reuso) y luego los trenes restantes ordenados en relación al tiempo de salida (se priorizan las llegadas que tienen tiempos más cercanos al tiempo de salida). Para estos experimentos se fijó el número de restarts a 20.

5.3.1. Instancias generadas aleatoriamente

La Figura 5.2 muestra el *boxplot* del desempeño obtenido por G-DMP usando instancias generadas aleatoriamente con listas restringidas de candidatos de tamaño 2, 5 y 10. Se muestra la distancia porcentual respecto a la mejor solución encontrada en todas las ejecuciones. Para cada grupo, las tres primeras cajas muestran el desempeño utilizando las instancias de tipo simple, mientras que las tres restantes muestran el desempeño utilizando instancias completas.

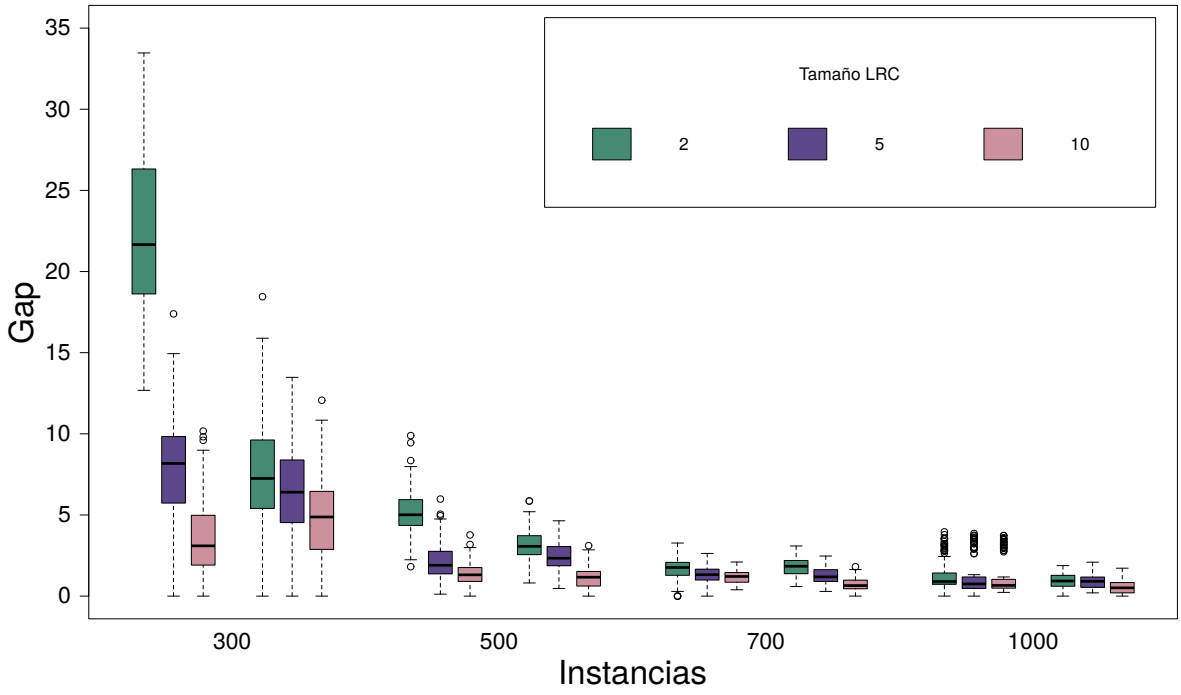


Figura 5.2: Gap porcentual entre la mejor solución encontrada para instancias aleatorias acorde al tamaño de la lista restringida de candidatos.

Se puede observar que usando un tamaño de lista 10, G-DMP encuentra soluciones de mejor calidad.

En la Tabla 5.2 se muestra el test de Friedman para comparar el efecto del valor LRC en el desempeño de G-DMP para las instancias generadas aleatoriamente. Acorde al resultado del test, con un 95 % de confianza, el mejor tamaño LRC es 10.

LRC	Rango Medio
10	1.32
5	2.07
2	2.60

Tabla 5.2: Test de Friedman para LRC con instancias aleatorias

Con el fin de analizar la importancia de cada fase en G-DMP se mide la calidad de las soluciones obtenidas a partir de:

1. La fase constructiva sin Búsqueda Local: "Construcción".

2. Búsqueda Local aplicada a una solución factible generada aleatoriamente, es decir, sin la fase constructiva: "Búsqueda Local".

En la Tabla 5.3 para las instancias simples, y en la Tabla 5.4 para las instancias completas, se muestra la calidad promedio de las soluciones obtenidas en la fase constructiva y la fase de Búsqueda Local, que corresponde a G-DMP, además de las soluciones obtenidas de manera aleatoria, y luego de aplicar Búsqueda Local a dichas soluciones aleatorias.

La calidad aquí se muestra como la distancia porcentual a la mejor solución encontrada en todas las ejecuciones.

Categoría	Construcción	t[s]	G-DMP	t[s]	Sol. Inicial	t[s]	B. Local	t[s]
300	64.79	13.1	**17.84**	44.9	110.15	4.0	50.74	47.1
500	67.27	40.6	**11.17**	203.0	78.13	6.8	17.66	259.6
700	51.49	80.5	**10.13**	474.1	53.71	8.8	11.95	532.4
1000	31.68	184.8	**7.15**	982.4	32.37	13.7	7.76	1212.1

Tabla 5.3: Comparación del desempeño por componentes para instancias simples generadas aleatoriamente

Categoría	Construcción	t[s]	G-DMP	t[s]	Sol. inicial	t[s]	B. Local	t[s]
300	99.57	16.0	**28.21**	60.6	188.54	4.4	28.73	70.0
500	76.44	47.4	**12.40**	426.7	113.13	7.0	13.31	535.0
700	54.31	88.4	**10.21**	1364.8	75.83	11.4	10.95	1699.2
1000	36.68	188.6	**10.59**	3972.6	47.35	14.0	10.68	4234.5

Tabla 5.4: Comparación del desempeño por componentes para instancias completas generadas aleatoriamente

A partir de estos resultados se puede observar que la fase de Búsqueda Local es siempre capaz de mejorar la calidad de las soluciones, ya sean creadas con la fase constructiva del GRASP o aleatoriamente.

Además se observa que el mejor desempeño se logra usando ambas fases con el algoritmo G-DMP.

5.3.2. Instancias ROADEF

La Figura 5.3 muestra el desempeño obtenido por G-DMP utilizando las instancias del set B de la competencia ROADEF, considerando los tamaños de la lista LRC $\in \{2, 5, 10\}$.

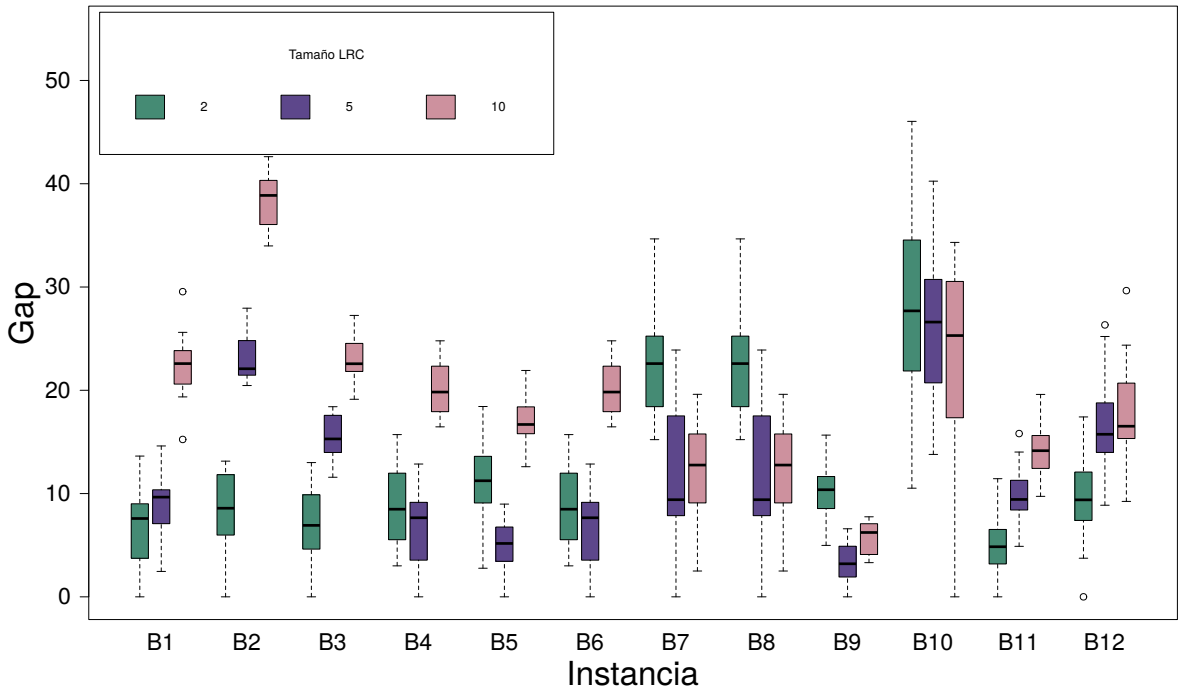


Figura 5.3: Gap porcentual entre la mejor solución encontrada para instancias ROADEF acorde al tamaño de la lista restringida de candidatos.

En la Tabla 5.5 se muestran los resultados del Test de Friedman que comparan el efecto del valor LRC en el desempeño de G-DMP para las instancias ROADEF.

Para estas instancias, el mejor largo de lista que permite a la fase constructiva generar soluciones de calidad capaces de ser mejoradas en la fase de búsqueda local es 5.

También se analiza la relevancia de cada fase de G-DMP utilizando el conjunto de instancias de la competencia ROADEF.

LRC	Rango Medio
5	1.72
2	1.84
10	2.44

Tabla 5.5: Test de Friedman para LRC con instancias ROADEF

La Tabla 5.6 muesta la calidad promedio de las soluciones obtenidas después de la fase constructiva, la fase de búsqueda local (G-DMP), una solución inicial factible construida de manera aleatoria y la búsqueda local aplicada a esta solución aleatoria.

Se observa que para las instancias B2 y B3 la aplicación de una búsqueda local a una solución generada aleatoriamente obtiene mejores resultados que utilizar G-DMP, esto se debe a que esas instancias no tienen preferencias de reusos. En general, para todas las instancias la búsqueda local significa una mejora para las soluciones iniciales, sin embargo, le toma más tiempo trabajar sobre las soluciones iniciales aleatorias.

I	Construcción	t[s]	G-DMP	t[s]	Sol. Inicial	t[s]	B. Local	t[s]
B1	80.92	38	**4.74**	109	295.46	4	23.86	237
B2	118.94	43	7.97	86	360.35	4	**6.06**	142
B3	108.05	57	15.64	88	333.07	4	**4.53**	140
B4	96.95	79	**7.83**	213	428.25	7	52.44	563
B5	102.41	122	**3.79**	338	433.85	9	51.93	957
B6	96.95	87	**7.83**	203	428.25	7	52.44	569
B7	145.71	3	**17.32**	6	340.92	1	66.64	10
B8	145.71	3	**17.32**	6	340.92	1	66.64	10
B9	43.19	57	**4.59**	326	210.62	7	26.11	1046
B10	247.03	1	**22.41**	4	433.68	7	44.45	3
B11	95.45	58	**7.84**	152	254.13	3	8.34	160
B12	123.62	11	**8.16**	28	276.12	1	18.52	35

Tabla 5.6: Comparación del desempeño por componentes para instancias ROADEF

5.3.3. Resultados Obtenidos para G-DMP

La Tabla 5.7 resume los resultados obtenidos al ejecutar G-DMP con las instancias ROADEF. Se realizaron pruebas con 20 semillas distintas. Se pueden ver la cantidad de salidas sin cubrir $\#u$, los mejores valores obtenidos y los valores promedios, para los costos simplificados f y los costos totales del problema, f^*, y el tiempo de ejecución con las 20 semillas.

	Mejor Encontrado			Promedio			Tiempo
I	$\#u$	f	f^*	$\#u$	f	f^*	t[s]
B1	92	147900	312174.0	102	160035	327758.5	109
B2	177	177000	260880.0	189	189000	271367.5	86
B3	188	188000	298409.0	200	200150	309270.0	88
B4	63	143200	323176.0	88	169935	348654.2	213
B5	102	185100	384776.0	119	202060	399657.9	338
B6	63	143200	323176.0	87	169055	347526.0	203
B7	33	43600	76590.6	42	54290	90140.3	6
B8	33	43600	76590.6	42	54290	90140.3	6
B9	162	260700	648433.0	194	282765	679076.9	326
B10	13	18000	44206.4	17	22545	54113.7	4
B11	138	170100	339863.0	156	190020	368185.4	152
B12	63	80500	168120.0	74	91895	182234.6	28

Tabla 5.7: Resultados G-DMP con 20 restart

En la Figura 5.4 el gap porcentual del costo total respecto al mejor encontrado en cada instancia, considerando las ejecuciones de la sección anterior.

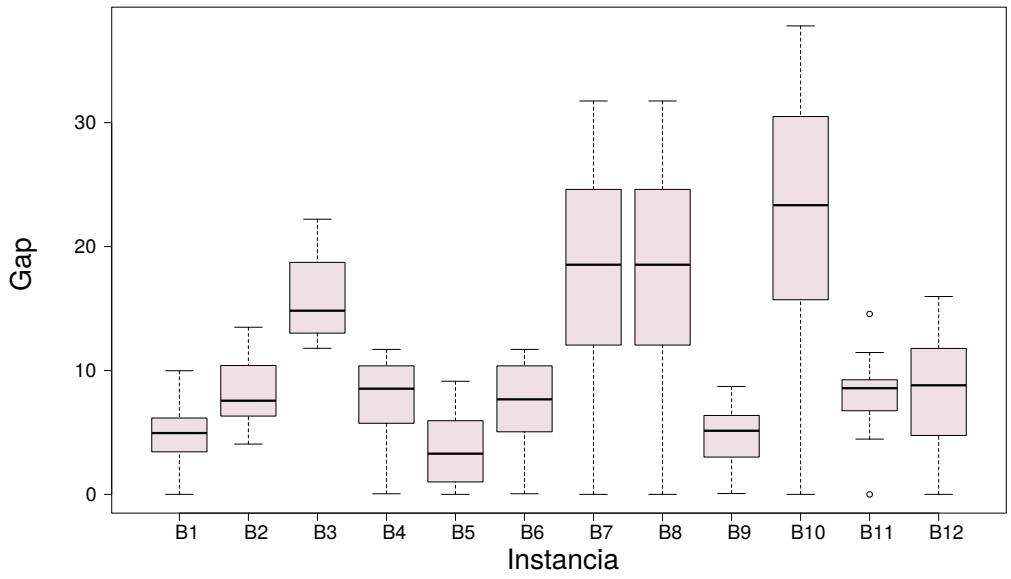


Figura 5.4: Gap porcentual entre la mejor solución encontrada por todas las ejecuciones para instancias ROADEF

5.3.4. Comparación de G-DMP con la literatura

Para comparar los resultados con los presentados en la literatura se utilizan las instancias de ROADEF.

Para abordar el problema, en [12] estudian 3 tipos de enfoque: Un enfoque MIP, que usa 6 tipos de variables: m_t^d indica si un tren t tiene un *matching* con una salida d; cat_t^i indica si un tren t posee la categoría i; dbm_t/tbm_t indica la distancia/tiempo antes del mantenimiento (DBM y TBM, respectivamente) y f_t^d/g_t^d si es que el tren t realiza un mantenimiento para la salida d. Un parametro $\omega_{t,day}^d$ indica el día en que se realiza cada mantenimiento. Este modelo minimiza los costos de las salidas sin cubrir y los costos de reusos no satisfechos. En este enfoque, cada tren se trata como un tren asociado, las restricciones se relajan para los trenes que no tienen salidas asociadas o son trenes iniciales en el sistema.

El enfoque de Generación de Columnas presentado en [12] permite incluir en el modelo posibles agendamientos para trenes que incluyen estos patrones (salidas asociadas), indicando con la variable c_d si una salida d fue cancelada, y λ_p si es que el patrón p fue usado. El enfoque sin Generación de columnas no permite ninguna salida asociada.

Por lo tanto, estos resultados presentados en [12] resultan de una versión relajada y simplificada del DMP, que no considera salidas ni llegadas en conjunto, tomando como que cada salida o llegada es realizada por un solo tren. Además, si hay llegadas en conjunto, son tomadas como llegadas separadas que ocurren en un mismo instante de tiempo, por lo que no consideran las restricciones asociadas a realizar las operaciones de separación. En cuanto a los mantenimientos, solo se considera la restricción de límite de mantenimientos diarios, pero no los costos asociados a estas operaciones. Por otra parte, estos enfoques sólo consideran los costos de salidas sin cubrir y reusos como parte de su función objetivo, a diferencia de G-DMP, que considera también los costos de unir y separar trenes y costos y restricciones asociadas al mantenimiento.

En la Tabla 5.8 se muestra la comparación entre los desempeños de estos enfoques vs. G-DMP, utilizando las instancias B de ROADEF. Sólo G-DMP calcula el costo total f^*, mientras que los otros enfoques solo calculan un costo reducido f.

Para cada instancia se indica el mejor costo encontrado, y el tiempo en segundos requerido para alcanzar ese valor. En las pruebas presentadas en [12] se fijó un tiempo máximo de ejecución de 20 minutos, en este caso se considerará ese mismo tiempo como máximo, indicando el número de *restarts* #r utilizados.

I	G-DMP				MIP		GC	
	f	f^*	#r	t[s]	f	t[s]	f	t[s]
B1	138400	305056.0	200	1115	-	-	**93200**	1207
B2	169000	253669.0	200	1092	-	-	**61000**	1200
B3	188000	298409.0	1	5	**0**	1199	**0**	26
B4	**143200**	323176.0	30	324	-	-	209300	1162
B5	**185100**	384776.0	5	83	-	-	221100	1203
B6	**143200**	323176.0	32	334	-	-	211800	1201
B7	41900	75783.2	95	31	**3400**	31	4300	78
B8	41900	75783.2	85	31	**3400**	31	4400	86
B9	260700	648433.0	85	1214	-	-	**252900**	1207
B10	14500	42032.6	200	34	**1000**	56	2000	5
B11	161700	344677.0	160	1198	-	-	**80600**	1215
B12	72500	159193.0	815	1203	43900	1205	**14300**	1206

Tabla 5.8: Comparación de resultados

Para G-DMP se calculan los costos reducidos f (para compararlos con la literatura) y los costos totales f^*, que consideran todos los costos mencionados en la Sección 2.2.1.

Se puede apreciar que el enfoque MIP es capaz de obtener resultados óptimos para la función relajada en las instancias $B3$, $B7$, $B8$ y $B10$ y que es capaz de llegar a una buena solución en $B12$, pero que no puede encontrar soluciones en el resto de las instancias. El enfoque con Generación de Columnas permite encontrar buenas soluciones, en términos de la función relajada, para las instancias $B1$, $B2$, $B3$, $B9$, $B11$ y $B12$ en un tiempo de ejecución de máximo 20 minutos.

G-DMP es capaz de encontrar soluciones de calidad para aquellas instancias que tienen un gran número de llegadas con salidas asociadas y preferencias de reusos, gracias a la representación que utiliza, ya que permite manejar sin problemas estas propiedades, a diferencia de los enfoques exactos presentes en la literatura.

5.4. Conclusiones del Capítulo

En este capítulo se presentaron los experimentos realizados al implementar un algoritmo GRASP que resuelve el DMP.

En primer lugar se hicieron pruebas con instancias aleatorias de diferentes tamaños, con el fin de analizar el impacto del tamaño de la lista del GRASP. Además se comparó en diferentes tipos de instancias: simples, que no considera salidas o llegadas en conjunto, y completas, que toman en cuenta todos los aspectos presentados en la definición del problema. Para eso, se evaluó el diseño de G-DMP en diferentes escenarios y se estudió la importancia de sus componentes y su configuración de parámetros. A partir de los resultados, tanto la fase constructiva, como la de Búsqueda Local,

demostraron ser cruciales en la calidad de las soluciones encontradas. El mejor valor del parámetro LRC, dependerá de la instancia del problema que se está resolviendo, esto se debe principalmente al movimiento utilizado durante la fase de Búsqueda Local.

Se realizó una comparación del algoritmo GRASP con resultados existentes en la literatura para el DMP, resuelto con métodos exactos, donde GRASP obtuvo mejores resultados para instancias que contienen un mayor número de llegadas asociadas, ya que éstas afectaban al modelo de programación entera mixta, debido a las simplificaciones que deben hacer del problema para poder abordarlo.

Además, GRASP, para buscar las soluciones, considera en su función evaluación los costos de sobremantenimiento y de unión y separación, para que al momento de resolver el problema completo, se minimice el impacto al agregar estos costos.

Los resultados de la investigación presentados en este capítulo fueron publicados en [19].

Capítulo 6

Diseño de la Propuesta: AE-DMP

En este capítulo se propone un Algoritmo Evolutivo (AE) para resolver el DMP: AE-DMP. Si bien G-DMP fue capaz de encontrar buenas soluciones en algunas instancias, no pudo superar al enfoque de Generación de Columnas en otras, en cuanto alos costos reducidos. Con el algoritmo evolutivo se pueden obtener soluciones distintas para un mismo número de salidas sin cubrir, por lo tanto se manejan más opciones para la planificación, mientras que con G-DMP, solo se puede encontrar una en cada ejecución. Ya que en el AE-DMP, varios individuos van revisando el espacio de búsqueda, es necesario que las operaciones no sean costosas.

La idea principal es tener poblaciones con diferentes individuos que corresponden a los *matchings* entre salidas y sus respectivos trenes, las cuales irán evolucionando en el tiempo. Se comenzará con una población inicial, donde un porcentaje de la población será generada de manera aleatoria, mientras que la mitad restante se generará por medio de un algoritmo *greedy* con componentes aleatorios, similar a la fase constructiva del algoritmo GRASP presentado en el Capítulo 4, con el fin de generar una población inicial que sea diversa que no se estanque rápidamente en óptimos locales. Luego, en cada generación se aplicarán los operadores genéticos correspondientes, sujetos a sus respectivas probabilidades, donde los individuos serán seleccionados utilizando ruleta. En cada generación se escogerá al individuo que tenga mejor función evaluación (*fitness*), que corresponderá al número de salidas cubiertas. Se trata de maximizar el número de salidas cubiertas. El AE tiene un post-procesamiento de búsqueda local con el fin de mejorar la solución final.

A lo largo de este capítulo se describirán en detalle las diferentes componentes del algoritmo desarrollado para resolver el problema.

6.1. Algoritmos Evolutivos

Los algoritmos evolutivos se basan en el principio de la Evolución de Darwin [21], donde a partir de una población inicial de individuos, mediante un proceso de selección natural y un proceso de transformación, los individuos evolucionan a través

de las generaciones.

Es necesaria una función de evaluación o *fitness* que mida que tan apto es un individuo y que guíe la búsqueda del algoritmo, y un proceso de selección que determine si su material genético será parte de los individuos en las siguientes generaciones. Para la transformación, existen los operadores genéticos, que actúan sobre uno o más individuos. En este caso se tienen dos operadores: mutación y cruzamiento. En la mutación se selecciona un solo individuo el cual será modificado, mientras que en el cruzamiento se aplicará a dos individuos, generando dos hijos que posean una combinación de ambos padres.

A continuación se presentan las componentes del Algoritmo Evolutivo presentado en esta propuesta:

6.2. Algoritmo Propuesto

En esta propuesta se trabajará siempre en el espacio factible.

6.2.1. Análisis de costos de las instancias

A diferencia de G-DMP, para el diseño de AE-DMP se utiliza como función de evaluación la maximización de las salidas cubiertas, con el objetivo que el algoritmo enfoque su búsqueda en satisfacer las salidas cubiertas dado que es el costo más relevante asociado, como se ve en la Tabla 6.1, por lo tanto, reducir las salidas sin cubrir podría ayudar a reducir los costos totales del problema, tal como se menciona en [12].

I	$uncovCost$	$reuseCost$	$junCost$	$disjCost$	$remDCost$	$remTCost$
B1	1000	100	100	100	0.2	5×10^{-5}
B2	1000	100	100	100	0.2	5×10^{-5}
B3	1000	100	100	100	0.2	5×10^{-5}
B4	1000	100	100	100	0.2	5×10^{-5}
B5	1000	100	100	100	0.2	5×10^{-5}
B6	1000	100	100	100	0.2	5×10^{-5}
B7	1000	100	100	100	0.2	3×10^{-3}
B8	1000	100	100	100	0.2	3×10^{-3}
B9	1000	100	100	100	0.2	3×10^{-3}
B10	1000	100	100	100	0.2	3×10^{-3}
B11	1000	100	100	100	0.2	3×10^{-3}
B12	1000	100	100	100	0.2	3×10^{-3}

Tabla 6.1: Costos utilizados en las instancias

El algoritmo evolutivo se enfocará en minimizar el costo de salidas sin cubrir, sin embargo, en la etapa de post-procesamiento se realiza una búsqueda local que considerará buscar una mejor solución desde el punto de vista total de los costos asociados.

6.2.2. Representación

En esta propuesta, tanto el genotipo como el fenotipo utilizan la misma representación. Al igual que en la representación utilizada para G-DMP explicada en la Sección 4.2.1 Cada individuo está representado por un vector de *matching*.

En esto caso, los *matching* corresponden a estructuras que tienen dos campos enteros: *idDep* que corresponde al identificador de la salida, e $idTrain$ que corresponde al indentificador del tren que va a cubrir esa salida. Para representar que salida no está cubiera se usa $idTrain = 0$. En la Figura 6.1, se puede ver un ejemplo de individuo.

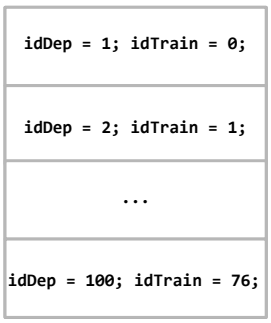


Figura 6.1: Ejemplo de un individuo

En este ejemplo se puede ver que la salida 1 no está cubierta, mientras que la salida 2 es realizada por el tren 1. En el caso de la salida 100, ésta es realizada por el tren 76.

6.2.3. Función *fitness*

Se utiliza el número de salidas cubiertas, es decir, las salidas donde idTrain $\neq 0$, el cual busca maximizarse. Se utiliza este valor como función de evaluación, ya que dentro de este problema, el costo con mayor peso es el de las salidas sin cubrir: *uncovCost*. Por ejemplo, la función *fitness* del ejemplo de la Figura 6.2 es 2, ya que solo 2 salidas tienen un tren asignado.

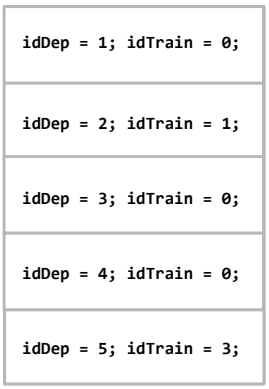


Figura 6.2: Ejemplo función *fitness* de un individuo

6.2.4. Generación de la población inicial

Con el fin de generar poblaciones iniciales más variadas, se evalúan dos procedimientos, uno basado en GRASP y otro aleatorio.

GRASP

Se utiliza la misma fase constructiva de G-DMP, presentada en la Sección 4.2.4. Acá se toman en cuenta directamente las preferencias de reusos, por lo que en este paso se considera la disminución de la función asociada al costo $reuseCost$, además del costo de salidas sin cubrir $uncovCost$.

Random

En este caso, los individuos de la población inicial se generan de manera aleatoria. Se itera entre los trenes disponibles y se escoge una salida sin cubrir al azar, se revisa la factibilidad, y si cumple se ingresa el tren al *matching*, tal como se muestra en el Algoritmo 5.

6.2.5. Algoritmo de selección

Para elegir los individuos que podrán participar en el proceso de transformación, es necesario contar con un algoritmo de selección. En este caso se utiliza la selección por ruleta, como la descrita en [25]. Para esto, se calcula la función *fitness* de cada individuo y se calcula su frecuencia $f_i = fitness(i)/suma\_total$, donde $suma\_total$ es la suma de los *fitness* de cada individuo. Acá, los individuos con mejores fitness tienen mayor probabilidad, ya que su frecuencia será mayor.

En el Algoritmo 6 se resume el proceso de selección por ruleta.

Algoritmo 5 Soluciones iniciales aleatorias

```
for tren in trenes_disponibles do
  salida = random(salidas_sin_cubrir)
  if factible(tren) then
    salidas_sin_cubrir -= salida
    trenes_disponibles -= tren
    match(salida, tren)
    matching.push_back(match)
  end if
end for
return matching;
```

Algoritmo 6 Selección por Ruleta

```
suma_total = sumar_fitness(población)
F_0 = 0
for i in población do
  f_i = fitness(i)/suma_total
  F_i+ = f_i
end for
roulette = random(0, 1)
for i in población do
  if F_i-1 ≤ roulette < F_i then
    return i
  end if
end for
```

6.2.6. Elitismo

En esta propuesta se considera el elitismo, ya que siempre el individuo con mejor función de evaluación pasa a formar parte de la población en la siguiente generación.

6.2.7. Operadores Genéticos

Operador Asexual

Este operador es tipo mutación, sin embargo está enfocado en la explotación. Modifica a un individuo seleccionado y tiene como objetivo cubrir salidas que aún no han sido cubiertas dentro de un individuo, con el fin de reducir el valor del costo *uncovCost* de su función de evaluación. Esta mutación se basa en una búsqueda local, que usa el mismo movimiento del post-procesamiento de G-DMP, presentado en el Algoritmo 4, en la Sección 4.1, y revisa las mismas restricciones al momento de realizar el movimiento. Se va revisando en el individuo si es que tiene alguna salida sin cubrir, si es así, elige aleatoriamente un tren y lo asigna, si es que es factible. A diferencia del algoritmo utilizado para G-DMP, acá se elige de manera aleatoria el tren a utilizar, y no de manera secuencial, además el criterio de aceptación es si se cubre la salida o no, mientras que el G-DMP utiliza como criterio la disminución del costo total. En el Algoritmo 7 se presenta la mutación utilizada por AE-DMP.

Algoritmo 7 Mutación AE-DMP

```
for match in individuo do
  if match.tren == 0 then
    tren = selección_random(trenes_disponibles)
    if factible(tren) then
      trenes_disponibles -= tren
      match = INSERT(tren, salida)
      mutado = cambiar_match(match, individuo)
    end if
  end if
end for
return mutado
```

Operador Bisexual

Este operador es de tipo cruzamiento, sin embargo, en este algoritmo está orientado a explorar. Tiene como objetivo combinar el material genético de dos individuos, y en este caso poder tener individuos diversos (ya que se generarán distintos *matchings*). Se realiza un cruzamiento en un punto, donde se escoge una porción de un

padre con la porción restante del otro padre, de esta manera se comparte la información entre ambos vectores de *matching* distintos, como se puede observar en la Figura 6.3.

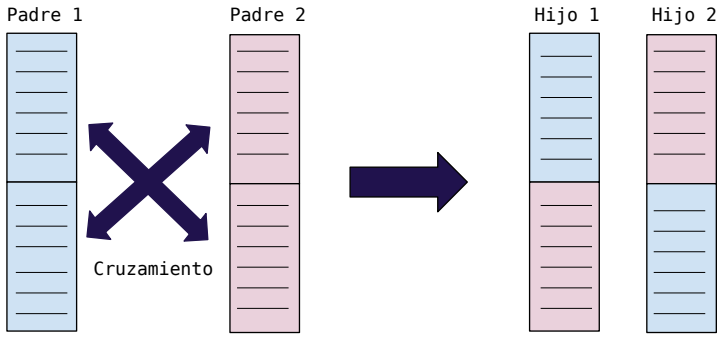


Figura 6.3: Esquema de Cruzamiento

El Operador de cruzamiento utilizado en AE-DMP puede resumirse en el Algoritmo 8. En este caso se escoge la mitad del individuo como punto de cruzamiento.

Algoritmo 8 Cruzamiento AE-DMP

```
cross_point = pop_size/2
for i < cross_point do
    hijo_1 = padre_1
    hijo_2 = padre_2
end for
for i ≤ cross_point do
    hijo_1 = padre_2
    hijo_2 = padre_1
end for
return hijo_1,hijo_2
```

Para mantener la factibilidad de la solución, en la generación de la población inicial, y en la mutación, los trenes utilizados para asignar las salidas se separan en dos conjuntos, de manera de asignar a las salidas siempre trenes que pertenezcan al mismo conjunto, y así no violar la restricción de no asignar más de una vez un tren. En la Figura 6.4 se presenta un esquema de esta selección, donde los trenes de cada color deben asignarse con salidas de su color correspondiente.

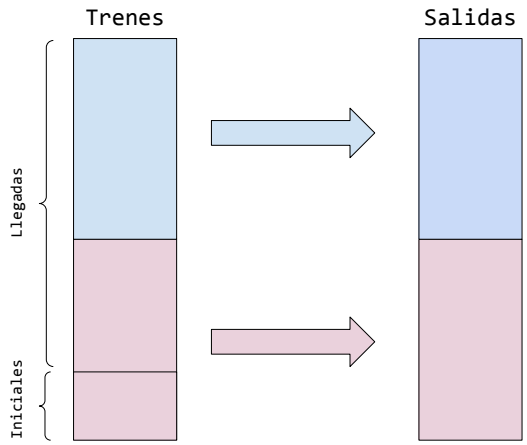


Figura 6.4: Selección de trenes

Esto no afecta a las soluciones, ya que las llegadas y salidas están ordenadas según sus respectivos tiempos, y así se evita cubrir salidas con trenes que tienen una gran diferencia de tiempo, por ejemplo una salida del día 7, con un tren que llega el día 1.

6.2.8. Post-Procesamiento

Una vez terminado el Algoritmo Evolutivo, se obtiene el o los mejores inidividuos (en el caso de tener el mismo valor de su fitness) y se les realiza una búsqueda local con *Hill Climbing first improvement*. En este caso el movimiento busca cubrir más salidas vía inserción como lo descrito en la Sección 4.1, sin embargo el criterio de aceptación está enfocado en la mejora del costo total.

En esta etapa, los trenes disponibles corresponden a todo el conjunto de trenes que aún no se han asignado, ya que es una operación final y no afectará la factibilidad de la solución.

6.3. AE-DMP

Tomando en cuenta las componentes detalladas anteriormente, se puede resumir en el Algoritmo 9 el AE-DMP.

Se puede observar que el AE-DMP necesita 5 parámetros:

1. `pop_size`: El tamaño de la población, que será fijo en cada generación.
2. `# generations`: Corresponde al número de generaciones del AE.
3. α: Probabilidad de mutación.

4. β: Probabilidad de cruzamiento.

5. `LRC`: Tamaño de la lista restringida de candidatos utilizada en el algoritmo GRASP para generar poblaciones iniciales.

Estos parámetros serán sintonizados con el programa `ParamILS` [14].

Algoritmo 9 AE-DMP

```
for pop_size do
  individuo = (greedy || random)
  población.push_back(individuo)
end for
while #generaciones do
  mejor_individuo = mejor(población)
  población_nueva
  población_nueva.push_back(mejor_individuo)
  r = random(0,1)
  while población_nueva.size() <pop_size do
    if r <α then
      individuo = ruleta(población)
      individuo_mutado = mutación(individuo)
      población_nueva.push_back(individuo_mutado)
    else if r >1 − β then
      padre1, padre2 = ruleta(población)
      hijo1, hijo2 = cruzamiento(padre1, padre2)
      población_nueva.push_back(hijo1)
      if población_nueva.size() <pop_size then
        población_nueva.push_back(hijo2)
      end if
    else
      individuo = ruleta(población)
      población_nueva.push_back(individuo)
    end if
  end while
  población = población_nueva
end while
mejor_solución = mejor(población)
solución_final = post-proc(mejor_solución)
```

6.4. Conclusiones del Capítulo

En este capítulo se presentó el diseño un Algoritmo Evolutivo para resolver el DMP: AE-DMP. Para el diseño del algoritmo se elige la misma representación usada en G-DMP, donde no se ve afectada la factibilidad de la solución al aplicar los operadores de tipo mutación y cruzamiento. La representación de cada individuo corresponde a un vector de *matching*, donde en cada elemento se identifica la salida y el tren que la cubre, considerando también el caso donde la salida no esté cubierta.

Para generar la población inicial se evaluaron dos opciones de algoritmos constructivos: un GRASP y una construcción aleatoria, con el fin de contar con un conjunto de individuos de buena calidad y diversos.

Los operadores genéticos definidos son uno de tipo mutación y otro de tipo cruzamiento. En la mutación se busca explotar, aumentando el número de salidas cubiertas, para eso, se revisa en cada individuo sus salidas sin cubrir, y de manera aleatoria se escoge un tren, que cumple con las restricciones del problema y se asigna. Si se asigna, se retorna el individuo mutado con una nueva asignación, disminuyendo el costo de las salidas sin cubrir. Por otra parte, el cruzamiento está enfocado en exploración, usa la combinación de dos individuos. Se realiza cruzamiento en un punto, donde para no afectar la factibilidad de la solución en los pasos previos, como la generación de soluciones iniciales, y en la mutación, los matching se realizan con un subconjunto definido de trenes.

Como la búsqueda del algoritmo evolutivo se centra en cubrir la máxima cantidad de salidas, se definió una búsqueda local como post-procesamiento para refinar los resultados obtenidos y considerando la totalidad de los costos asociados.

En el siguiente capítulo se presenta la evaluación del algoritmo propuesto.

Capítulo 7

Experimentación y Resultados para AE-DMP

Para poder evaluar la implementación del algoritmo propuesto, que consiste en resolver el DMP utilizando un algortimo evolutivo, se realizan pruebas utilizando las instancias del set B de la competencia ROADEF. En primer lugar se analizan las características del algoritmo propuesto, y después se comparan los resultados obtenidos con los presentes en la literatura.

A lo largo del presente capítulo se describirán los experimentos, resultados obtenidos y las conclusiones respectivas.

7.1. Entorno de experimentación

El algoritmo fue desarrollado utilizando el lenguaje C^{++} con el compilador gcc versión 6.4.1 .

Las pruebas se realizaron en un computador con procesador Intel(R) Core(TM) i5-2400 CPU @ 3.10GHz, con 4GB de memoria RAM y sistema operativo Fedora 25.

7.2. Experimentos

Para los experimentos presentados en este trabajo, se utilizan las instancias del set B de la competencia ROADEF 2014. Estas instancias fueron utilizadas por los competidores para evaluar sus resultados finales en la competencia. Estas instancias tienen diferentes tamaños y características, como se observa en la Tabla 5.1 de la Sección 5.2, donde se indica el número de llegadas y salidas, el número de llegadas asociadas, que son aquellas cuyo tren fue reutilizado previamente para una salida y cuyos datos deben actualizarse si es que ocurrió esa salida previa, el número de salidas y el número de reusos.

El fin de esta sección es analizar el algoritmo propuesto, y comparar los resultados con los obtenidos en [12], donde utilizan diferentes métodos exactos para obtener resultados para el DMP, y los obtenidos en [19], donde se utilizó un enfoque GRASP.

7.3. Proceso de Sintonización

Se utilizó el sintonizador de parámetros `ParamILS`[14], que es un algoritmo iterativo de búsqueda local, cuyo objetivo es encontrar los mejores valores de parámetros basado en los resultados obtenidos de la ejecución del algoritmo usando un subconjunto de instancias denominadas instancias de entrenamiento. En este caso, se utilizaron las instancias del set B de la competencia ROADEF, las mismas usadas para medir el desempeño del algoritmo. Se utilizó el *gap* a los mejores resultados encontrados hasta la fecha con el fin de escalar diferentes valores de la función de costos f, correspondiente a las salidas sin cubrir y reusos (mismos costos que los usados en la literatura). Los parámetros a sintonizar fueron α y β, que corresponden a las probabilidades de mutación y cruzamiento, respectivamente, y el largo de la lista LRC, utilizado en la generación de soluciones iniciales. El tamaño de la población fue fijado en 10, y se ejecutó con 100 generaciones.

En el diseño del AE-DMP se propone generar las poblaciones iniciales de dos formas: de manera aleatoria (*random*) o construirlas con un algoritmo de tipo *greedy*. Para encontrar cual de estas opciones tiene un mejor efecto en la función objetivo, se incluye en la sintonización el valor $iconf$, donde cada valor corresponde al tipo de solución inicial que se va a utilizar:

- $iconf = 1$ corresponde a soluciones iniciales generadas únicamente de manera aleatoria.
- $iconf = 2$ corresponde a soluciones iniciales generadas únicamente con GRASP.
- $iconf = 3$ corresponde a generar la mitad de la población con GRASP y la otra mitad de manera aleatoria.

El conjunto de posibles valores de los parámetros entregados a `paramILS` es:

- $\alpha = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$
- $\beta = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$
- LRC $= \{1, 2, 3\}$
- $iconf = \{1, 2, 3\}$

Sintonizar estos parámetros tomó un tiempo de aproximadamente 48 minutos. La configuración de parámetros utilizada en los experimentos, en base a lo entregado por `ParamILS` se resume en la Tabla 7.1.

pop_size	#generations	α	β	LRC	$iconf$
10	100	0.3	0.5	1	2

Tabla 7.1: Configuración de parámetros obtenida

Al obtener LRC = 1, se ve que la mejor opción es realizar un algoritmo *greedy* en lugar de GRASP. Esto se debe a que siempre se escoge el mejor candidato, lo que resulta útil cuando hay reusos, ya que así se ahorra el costo de reusos, disminuyendo, como consecuencia, el costo total.

El mejor valor obtenido para $iconf$ es 2, esto quiere decir que todas las soluciones iniciales serán generadas con el algoritmo *greedy*, que cuenta con una función miope orientada a las preferencias de reuso.

7.4. Efecto del Post-Procesamiento

En la Tabla 7.2 se presenta la comparación de los costos promedio obtenidos al no aplicar el post-procesamiento, versus aplicar el post-procesamiento. Para obtener estos resultados, se generaron las soluciones iniciales utilizando *greedy* y se realizaron pruebas con 20 semillas distintas, además de la configuración de parámetros descrita en 7.1. Se compara el número de salidas sin cubrir $\#u$, el costo reducido f y el costo total f^* y los tiempos de ejecución respectivos.

	Sin Post-Procesamiento				Post-Procesamiento			
I	$\#u$	f	f^*	t[s]	$\#u$	f	f^*	t[s]
B1	397.7	398900	475503.8	14.9	6.55	7750	186410.10	32.0
B2	1153.4	1153350	1163324.5	13.0	47.45	47450	355661.95	786.8
B3	1153.9	1153850	1165035.5	12.6	26.30	26300	359212.15	655.9
B4	651.6	654400	701543.7	23.9	0.50	3300	209632.05	53.3
B5	1020.4	1020000	1070000.0	36.2	0.00	3400	307590.35	100.3
B6	651.6	654400	701543.7	24.0	0.50	3300	209632.05	54.1
B7	83.1	85750	129512.0	4.3	6.45	9150	71779.24	5.7
B8	83.1	85750	129512.0	4.3	6.45	9150	71779.24	5.8
B9	700.9	703100	961749.8	20.6	9.10	11300	468830.75	47.2
B10	37.0	38600	61705.4	2.1	4.95	6550	37531.26	2.7
B11	364.4	365250	438672.1	8.6	0.00	900	206132.65	16.6
B12	160.7	161900	207844.9	5.5	0.00	1200	100941.19	7.7

Tabla 7.2: Comparación de costos promedio de AE-DMP al utilizar post-procesamiento

Se puede observar que en todas las instancias es importante el uso del post-procesamiento, logrando grandes mejoras en la reducción de salidas sin cubrir y de los costos reducidos y totales. Esta mejora se debe a que la búsqueda local realizada revisa por completo el mejor individuo obtenido, y va cubriendo las salidas sin cubrir

con todos los trenes restantes que tiene, y que cumplen las restricciones. Además, como se realiza sobre el último de los individuos, no genera tiempos de ejecución extra en el algoritmo, y hace que no se requiera una gran cantidad de generaciones.

Al momento de realizar la sintonización de parámetros, se consideró esta etapa dentro del algoritmo, por lo tanto los parámetros utilizados funcionan bien en conjunto con el post-procesamiento.

7.5. Resultados Obtenidos

A continuación se presentan los resultados obtenidos por el algoritmo AE-DMP propuesto. Se ejecutaron pruebas con 20 semillas distintas para cada una de las instancias. La configuración de parámetros utilizada fue la descrita en la Tabla 7.1. Los resultados se presentan en la Tabla 7.3, donde se muestra el número de salidas sin cubrir $\#u$, el costo reducido $f = f^{uncov} + f^{reuse}$ y el costo total $f^* = f^{uncov} + f^{reuse} + f^{over} + f^{jun}$ y los tiempos de ejecución con todas las semillas. Se muestra el mejor valor obtenido y el valor promedio, para cada instancia.

	Mejor			Promedio			Tiempo
I	$\#u$	f	f^*	$\#u$	f	f^*	t[s]
B1	4	5200	183439.0	6.55	7750	186410.10	32.0
B2	45	45000	351110.0	47.45	47450	355661.95	786.8
B3	26	26000	353459.0	26.30	26300	359212.15	655.9
B4	0	2800	207718.0	0.50	3300	209632.05	53.3
B5	0	3400	306030.0	0.00	3400	307590.35	100.3
B6	0	2800	207718.0	0.50	3300	209632.05	54.1
B7	2	4700	68150.4	6.45	9150	71779.24	5.7
B8	2	4700	68150.4	6.45	9150	71779.24	5.8
B9	9	11200	460425.0	9.10	11300	468830.75	47.2
B10	0	1600	34033.6	4.95	6550	37531.26	2.7
B11	0	900	203782.0	0.00	900	206132.65	16.6
B12	0	1200	99301.0	0.00	1200	100941.19	7.7

Tabla 7.3: Resultados AE-DMP

Se puede apreciar que en las instancias B4, B5, B6, B10, B11 y B12 se logran soluciones que no poseen salidas sin cubrir, una de las principales dificultades de los métodos exactos presentados en [12].

En la Figura 7.1 se presenta el gap porcentual de los costos totales respecto al mejor encontrado en cada instancia. Se puede ver que en general los datos se encuentran cercanos al mejor encontrado por el algoritmo. En el caso de las instancias B7, B8 y B10 hay mayor dispersión, debido a que son instancias pequeñas, y que en proporción, se ven más afectadas si es que aumentan las salidas sin cubrir.

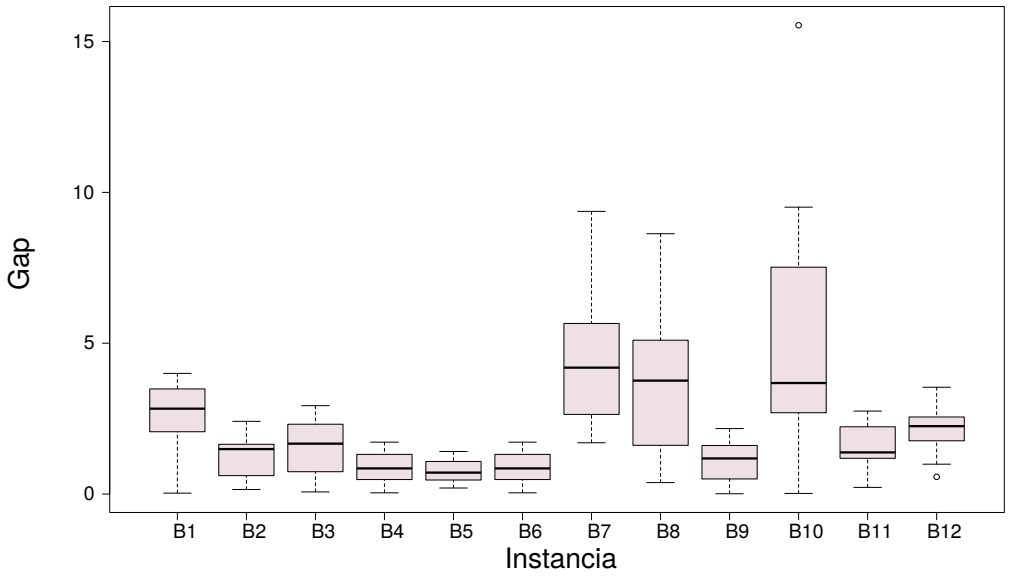


Figura 7.1: Gap porcentual entre la mejor solución encontrada para instancias ROADEF

7.6. Comparación estadística entre G-DMP y AE-DMP

En esta sección se presentan los resultados obtenidos al utilizar la prueba de los rangos con signo de Wilcoxon para comparar el desempeño obtenido por las propuestas presentadas.

En la Tabla 7.4 se puede ver que los *p-value* obtenidos demuestran que existen diferencias estadísticas significativas para probar que AE-DMP encuentra mejores resultados que G-DMP en la mayoría de las instancias considerando el costo total $f = f^{uncov} + f^{reuse} + f^{over} + f^{maint}$, con un nivel de confianza de 95 %.

Comparación		Rangos Pos.	Rangos Neg.	Empates	*p-value*
B1 AE-DMP	B1 G-DMP	0	20	0	1.907×10^{-6}
B2 AE-DMP	B2 G-DMP	20	0	0	1.907×10^{-6}
B2 AE-DMP	B2 G-DMP	20	0	0	1.907×10^{-6}
B4 AE-DMP	B4 G-DMP	0	20	0	1.907×10^{-6}
B5 AE-DMP	B5 G-DMP	0	20	0	1.907×10^{-6}
B6 AE-DMP	B6 G-DMP	0	20	0	1.907×10^{-6}
B7 AE-DMP	B7 G-DMP	0	20	0	1.907×10^{-6}
B8 AE-DMP	B8 G-DMP	0	20	0	1.907×10^{-6}
B9 AE-DMP	B9 G-DMP	0	20	0	1.907×10^{-6}
B10 AE-DMP	B10 G-DMP	0	20	0	1.907×10^{-6}
B11 AE-DMP	B11 G-DMP	0	20	0	1.907×10^{-6}
B12 AE-DMP	B12 G-DMP	0	20	0	1.907×10^{-6}

Tabla 7.4: Test de Wilcoxon

7.7. Comparación con la literatura

A continuación se compara el desempeño de AE-DMP con los mejores resultados presentes en la literatura y con G-DMP. Para estos resultados, se utilizaron los parámetros sintonizados según la Tabla 7.1.

Se puede observar en la Tabla 7.5 que AE-DMP obtiene muy buenos resultados respecto a los encontrados en la literatura y en comparación a G-DMP. En los costos reducidos, supera a G-DMP en todas las instancias, y en los costos totales en la mayoría de las instancias. En las instancias B2 y B3 G-DMP obtiene mejores resultados en cuanto al costo total, esto se debe a que estas instancias no tienen reusos, y el *greedy* usado para generar la población inicial se enfoca en agregar siempre las preferencias de reuso a la solución.

En cuanto a los modelos exactos, MIP solo obtiene resultados para la instancia B3, que no posee ni reusos ni salidas enlazadas, y para las instancias que son más pequeñas, como la B7, B8, B10 y B12. AE-DMP, obtiene mejores resultados para B12, que las técnicas exactas, y no se aleja mucho de los resultados de B7, B8 y B10.

Al igual que en G-DMP, AE-DMP considera la totalidad de los costos.

	AE-DMP			G-DMP			MIP		GC	
Inst.	f	f^*	t[s]	f	f^*	t[s]	f	t[s]	f	t[s]
B1	**5200**	**183439.0**	32.0	138400	305056.0	109	-	-	93200	1207
B2	**45000**	351110.0	786.8	169000	**253669.0**	86	-	-	61000	1200
B3	23000	353459.0	655.9	188000	**298409.0**	88	**0**	1199	**0**	26
B4	**2800**	**207718.0**	53.3	143200	323176.0	213	-	-	209300	1162
B5	**3400**	**306030.0**	100.3	185100	384776.0	338	-	-	221100	1203
B6	**2800**	**207718.0**	54.1	143200	323176.0	203	-	-	211800	1201
B7	4700	**68150.4**	5.7	41900	75783.2	6	**3400**	31	4300	78
B8	4700	**68150.4**	5.8	41900	75783.2	6	**3400**	31	4400	86
B9	**11200**	**460425.0**	47.2	260700	648433.0	326	-	-	252900	1207
B10	1600	**34033.6**	2.7	14500	42032.6	4	**1000**	56	2000	5
B11	**900**	**203782.0**	16.6	161700	344677.0	152	-	-	80600	1215
B12	**1200**	**99301.4**	7.7	72500	159193.0	28	14700	23115	14300	1206

Tabla 7.5: Comparación de desempeño con otros acercamientos de la literatura

7.8. Conclusiones del Capítulo

Se propuso un algoritmo evolutivo que es una alternativa para la resolución del DMP cuando las técnicas completas no sean capaces de resolverlo.

Los resultados obtenidos son satisfactorios, ya que en tiempos de ejecución rápido se pueden lograr mejores resultados que los obtenidos en la literatura y en G-DMP, en la mayoría de las instancias.

Al igual que en G-DMP, AE-DMP considera la totalidad de los costos, y no hace una relajación de restricciones, esto le permite poder encontrar soluciones que no tengan salidas sin cubrir, a diferencia de los métodos exactos.

Uno de los objetivos al proponer un algoritmo evolutivo es contar con soluciones alternativas en cuanto las salidas sin cubrir. Esto permitirá la elección por parte de un usuario de acuerdo a algún crriterio propio de selección. La mejora realizada en el postprocesamiento permite incluir el resto de los costos para mejorar aún más la(s) solucion(es) obtenida(s).

Otra ventaja de este enfoque, es que la representación es sencilla, y permite tomar en cuenta aspectos del problema que son difíciles de considerar utilizando técnicas completas.

Capítulo 8

Conclusiones

Los problemas de asignación, y en particular de asignación de trenes son problemas estudiados en la literatura y que poseen una gran complejidad debido a que son problemas aplicables a la vida real. El RSUM, planteado en el desafío ROADEF 2014, presenta una complejidad mayor, respecto de los clásicos problemas de *scheduling*, ya que combina varias restricciones que consideran diferentes aspectos en un solo problema. Los investigadores y participantes del desafío concuerdan en que es un problema difícil, que para ser resuelto de manera holística, es necesario abordar por separado sus principales componentes. Para este problema, los investigadores se han enfocado principalmente en el modelo, con el fin de simplificarlo para resolverlo, más que en un algoritmo que lo resuelva. La estrategia común que se utiliza para abordar este problema es dividirlo en dos: *Departure Matching Problem*, que consiste en encontrar los mejores emparejamientos entre salidas y trenes (ya sean de llegadas o inicialmente en el sistema), y *Routing Problem*, donde se buscan rutas factibles entre las llegadas y salidas.

El DMP es un problema interesante por sí solo, ya que también toma en cuenta aspectos de la infraestructura ferroviaria para realizar las asignaciones de trenes, como lo son los tiempos en que se realizan operaciones de mantenimiento o de unión y separación de trenes, o relacionados a las capacidades de infraestructura, como lo es el límite de mantenimientos diario.

En este trabajo de tesis se propuso dos algoritmos basados en heurísticas para resolver el DMP: un algoritmo basado en GRASP: G-DMP, y un algoritmo evolutivo: AE-DMP, los cuales consideran todos los aspectos necesarios para resolver el problema, principalmente sus restricciones y costos.

Si bien se ha podido resolver el DMP con métodos exactos, éstos sólo consideran dos de las componentes de la función objetivo, que son las salidas sin cubrir, o cancelaciones, y las reutilizaciones, que están directamente ligadas al *matching*, sin embargo, aunque consideran las restricciones del límite de mantenimientos diarios, no toman en cuenta el costo de realizar estos mantenimientos. Otra parte que no se considera son los costos de unión y separación, que están asociados a las llegadas y salidas en conjunto.

Si bien G-DMP considera todos estos aspectos, a diferencia de las técnicas exactas, AE-DMP puede obtener mejores resultados, ya que los operadores genéticos y la representación utilizada permiten visitar más lugares del espacio de búsqueda. Incluso utilizando una lista de candidatos muy restringida para el caso de las soluciones iniciales.

Es importante remarcar el uso de un post-procesamiento, que intenta en último término asignar en un individuo completo aquellas salidas que quedaron sin cubrir con la visión de minimizar la totalidad de los costos involucrados.

Las relajaciones y cambios al modelo, se ven reflejadas en los resultados, donde los métodos exactos sólo encuentran mejores soluciones en instancias que poseen menor cantidad de reusos y llegadas con asociaciones, mientras que las instancias con un alto número de reusos y asociaciones se vuelven inmanejables. Incluso se puede ver, que a excepción de la instancia B3, que posee 0 reusos y asociaciones, AE-DMP se aproxima a los valores encontrados con las técnicas exactas.

Si bien AE-DMP posee varios parámetros, su rápida ejecución permite sintonizarlos en un tiempo razonable de aproximadamente 48 minutos. Esta configuración, entregada por `ParamILS`, permitió que el algoritmo pudiese trabajar de mejor manera, entregando no solo valores para los parámetros, sino trambién para configurar el diseño de la población inicial.

8.1. Trabajo Futuro

Se propone continuar esta investigación tomando los resultados obtenidos por estos algoritmos y transformarlos en *input* para resolver el problema de las rutas de ROADEF.

Bibliografía

[1] Challenge ROADEF/EURO 2014 - registered teams (15th jan 2014). `http://challenge.roadef.org/2014/en/participants.php`. Consultado: 18-03-2018.

[2] Challenge ROADEF/EURO 2014 final results. `http://challenge.roadef.org/2014/en/finalResults.php`. Consultado: 18-03-2018.

[3] ROADEF/EURO challenge 2014: Trains don't vanish! `http://challenge.roadef.org/2014/en/index.php`. Consultado: 18-03-2018.

[4] ROADEF/EURO challenge 2014: Trains don't vanish! `http://challenge.roadef.org/2014/files/Challenge_ROADEF_EURO_SNCF_2014_rules_280414.pdf`. Consultado: 18-03-2018.

[5] D. E. Brown, C. L. Huntley, B. P. Markowicz, and D. E. Sappington. Rail network routing and scheduling using simulated annealing. In *IEEE International Conference on Systems, Man and Cybernetics*, pages 589–592. IEEE, 1992.

[6] X. Cai and C. Goh. A fast heuristic for the train scheduling problem. *Computers & Operations Research*, 21(5):499–510, 1994.

[7] H. Cambazard and N. Catusse. ROADEF challenge 2014: A modeling approach.

[8] C. Chang and S. Sim. Optimising train movements through coast control using genetic algorithms. In *IEE Proceedings, Electric Power Applications*, volume 144, pages 65–73. IET, 1997.

[9] C. S. Chang and C. M. Kwan. Evaluation of evolutionary algorithms for multi-objective train schedule optimization. In *Advances in Artificial Intelligence*, pages 803–815. Springer, 2004.

[10] T. A. Feo and M. G. Resende. Greedy randomized adaptive search procedures. *Journal of global optimization*, 6(2):109–133, 1995.

[11] J. T. Haahr and S. H. Bull. A math-heuristic framework for the ROADEF/EURO challenge 2014. Technical report, DTU Management Engineering, 2014.

[12] J. T. Haahr and S. H. Bull. Exact methods for solving the train departure matching problem. Technical report, DTU Management Engineering, 2015.

[13] J. T. Haahr, D. Pisinger, and J. Larsen. *Reactive Robustness and Integrated Approaches for Railway Optimization Problems.* PhD thesis, Technical University of DenmarkDanmarks Tekniske Universitet, Department of Informatics and Mathematical ModelingInstitut for Informatik og Matematisk Modellering, 2015.

[14] F. Hutter, H. H. Hoos, K. Leyton-Brown, and T. Stützle. Paramils: an automatic algorithm configuration framework. *Journal of Artificial Intelligence Research*, 36(1):267–306, 2009.

[15] C. M. Kwan and C. S. Chang. Application of evolutionary algorithm on a transportation scheduling problem-the mass rapid transit. In *IEEE Congress on Evolutionary Computation*, volume 2, pages 987–994. IEEE, 2005.

[16] R. S. Kwan and P. Mistry. A co-evolutionary algorithm for train timetabling. In *Congress on Evolutionary Computation*, volume 3, pages 2142–2148. IEEE, 2003.

[17] R. M. Lusby, J. T. Haahr, J. Larsen, and D. Pisinger. A branch-and-price algorithm for railway rolling stock rescheduling. *Transportation Research Part B: Methodological*, 99:228–250, 2017.

[18] S. Mulders and Y.-L. Scholliers. Train scheduling and resource management in a railway station. Master's thesis, Université Catholique de Louvain, 2014.

[19] A. Rojas, E. Montero, and M.-C. Riff. G-DMP: An algorithm without constraint relaxation to solve the train departure matching problem. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 2382–2389. IEEE, 2017.

[20] Y. Semet and M. Schoenauer. An efficient memetic, permutation-based evolutionary algorithm for real-world train timetabling. In *IEEE Congress on Evolutionary Computation*, volume 3, pages 2752–2759. IEEE, 2005.

[21] B. Thomas, F. David, and M. Zbigniew. Handbook of evolutionary computation, 1997.

[22] J. Törnquist and J. A. Persson. Train traffic deviation handling using tabu search and simulated annealing. In *HICSS'05. Proceedings of the 38th Annual Hawaii International Conference on System Sciences.* IEEE, 2005.

[23] R. W. van den Broek. Train shunting and service scheduling: an integrated local search approach. Master's thesis, 2016.

[24] P. Vansteenwegen, W. Souffriau, and D. V. Oudheusden. The orienteering problem: A survey. *European Journal of Operational Research*, 209(1):1–10, 2011.

[25] J. Zhong, X. Hu, J. Zhang, and M. Gu. Comparison of performance between different selection strategies on simple genetic algorithms. In *International Conference on Computational Intelligence for Modelling, Control and Automation*, volume 2, pages 1115–1121. IEEE, 2005.