Implementación de un Algoritmo Inmune Artificial basado en Selección Clonal para el UTRP

Cristopher Arenas
cristopher.arenas@alumnos.usm.cl

Departamento de Informática
Universidad Técnica Federico Santa María

Abstract. El diseño de un sistema de transporte público eficiente es un problema importante en la actualidad. El Urban Transit Routing Problem o UTRP, es un problema multi-objetivo planteado en la literatura, que intenta establecer un enrutamiento eficiente para satisfacer a pasajeros y operadores, minimizando costos temporales y restringiendo las rutas determinadas. En este trabajo se propondrá un Algoritmo Inmune Artificial o AIA, una propuesta multi-objetivo basada en el sistema inmune, el cual tiene procesos de transformación y selección con operadores asociados de manera de obtener un conjunto de soluciones que consideren los intereses de pasajeros y de operadores de una determinada red de transporte público.

1 Introducción

El diseño de un sistema de transporte que sea eficaz y eficiente es un problema importante en la actualidad y es de gran relevancia en las grandes urbes. porque implican translados de muchas personas en todo momento del día. Existen varios problemas asociados al diseño un buen sistema de transporte, los cuales se enfocan principalemente en establecer una buena planificación de horarios para conductores o buses, o definir enrutamientos que beneficien tanto a los usuarios como a las empresas operadoras de buses. El Urban Transit Routing Problem o UTRP [10] consiste en definir enrutamientos para una red de transporte público que sean capaces de reducir costos temporales tanto para pasajeros, como para operadores de buses. Es así como el UTRP se plantea como un problema multi-objetivo que requiere de técnicas que se encarguen de minimizar ambos costos.

En la literatura [12, 8] se han propuesto varios algoritmos, principalmente evolutivos, que tienen inspiración en fenómenos biológicos, para obtener soluciones en el UTRP. Los algoritmos propuestos establecen la evolución de soluciones iniciales definidas mediante estratégias o heurísticas, las cuales son modificadas mediante operadores de transformación, de manera de encontrar en pequeñas vecindades soluciones similares que puedan ser más beneficiosas para pasajeros u operadores de buses. Los Algoritmos Inmunes Artificiales o AIA [4,

15] corresponden a una familia de algoritmos que se basan en el sistema inmune de un organismo, donde un conjunto de anticuerpos debe enfrentarse a un virus o infección. Los anticuerpos funcionan por medio de procesos adaptativos, pues una vez que encuentran la mejor forma de curar una enfermedad, se capacitan para tomar acción ante futuras infecciones.

En el presente trabajo considera una propuesta de AIA, con sus respectivos procesos de transformación y selección. Esto es, mediante la definición de operadores de transformación específicos para el UTRP y por medio de la presentación de parámetros relevantes para los procesos de selección. En el capítulo 2 se presentará el estado del arte del UTRP y se mostrarán enfoques y técnicas utilizadas para encontrar soluciones. El capítulo 3 describe en detalle el UTRP, mencionando las variables, parámetros y restricciones que considera el problema, asi como los objetivos que deben minimizarse. El capítulo 4 muestra la representación de elementos importantes para el problema. Los capítulos 5, 6, 7 y 8 presentan el AIA propuesto, detallando los pasos necesarios para generar soluciones factibles, parámetros asociados, operadores de tranformación y otras consideraciones importantes. El capítulo 9 detalla una serie de experimentos que se realizarán sobre instancias utilizadas en la literatura, cuyos resultados obtenidos se mostrarán en el capítulo 10.

2 Estado del Arte

El problema de enrutamiento de tránsito urbano (UTRP por sus siglas en inglés) involucra la ideación de rutas para el transporte público. Se trata de un problema NP-Duro altamente complejo, y resolverlo involucra invariablemente un ciclo de generación y prueba de grupos de rutas candidatas. La mayor parte de literatura lo considera parte de un problema de escala mayor, el Urban Transit Network Design Problem (UTNDP por sus siglas), el cual es dividido tanto en el UTRP como UTSP (Urban Transit Scheduling Problem), y tal como se puede diferir de sus nombres, el UTSP tiene un enfoque de agendar los tiempos de llegada de los medios de transporte, mientras que el UTRP se enfoca en las rutas que estos utilizan, ambos para mejorar el sistema de transporte urbano en las ciudades. Los primeros acercamientos al UTRP lo tratan como un problema mono-objetivo. En [10] se comparan dos acercamientos de búsqueda local para resolver el problema. En este caso se consideran dos objetivos: la distancia acumulada de todos los pasajeros del bus y el número de trasbordos para la demanda completa. Se considera toda la demanda como satisfecha y el tiempo promedio que cada usuario destina en viajar es el menor posible. Además, se considera que cada ruta del conjunto está libre de ciclos y retrocesos; el conjunto de rutas está conectado; hay exactamente r rutas en el conjunto y el número de nodos en cada ruta debe ser mayor a uno y no debe exceder el valor máximo definido.
La función objetivo está dada por la suma ponderada de ambos objetivos. La inicialización se realiza de manera aleatoria respetando el largo establecido. Se utiliza el movimiento *Make-small-change*, que considera 3 posibilidades: Agre-

gar un nodo en la última posición de la ruta, borrar el primer nodo de la ruta e invertir el orden de nodos en la ruta. En [17] se propone un algoritmo memético cuyo objetivo es minimizar la suma del costo para los usuarios y la demanda insatisfecha para la red de rutas. La inicialización se reliza de manera aleatoria, se utiliza cruzamiento de rutas en un punto y mutación de una ruta por otra ruta factible. Para la búsqueda local se seleccionan uno o dos cromosomas aletatoriamente, y se combinan de acuerdo a: (1) movimiento 2-opt, (2) intercambio de dos paraderos y (3) reubicación de una parada. Al final del proceso se seleccionan los mejores μ cromosomas padre y λ cromosomas hijo. En [7] se utilizan colonias de hormigas para el UTRP. A diferencia de otros problemas resueltos con colonias de hormigas, las hormigas no deben recorrer todos los nodos, sino que deben ir desde cierto nodo hasta otro. Se toma como consideración que cada nodo podrá tener a lo más 4 vecinos y las conexiones con estos serán aquellas pertenceciente a un conjunto de arcos permitidos. Si en algún momento la ruta escogida por la hormiga no cumple alguna de las restricciones (por ejemplo, ciclos en el recorrido), la hormiga se declara muerta y se castiga el camino escogido.

Es posible encontrar acercamientos multi-objetivo para otros problemas similimares. Josefowiez, Semet y Talbi en [9] utilizan diversificación elitista y modelo de islas para el vehicle routing problem con dos objetivos: minimización del largo de las rutas y la diferencia entre el tamaño de la ruta más larga y la más corta. En este caso se utiliza ranking de dominancia para evaluar a los invididuos, donde los individuos no dominados de la población forman el conjunto E_1 de ranking 1 y el resto de los individuos se agrupan en conjuntos E_k donde cada elemento es dominado por todos los elementos de los conjuntos anteriores. En [14] se propone un MOEA híbrido que incorpora búsqueda local y el concepto de optimalidad de Pareto para el problema de encontrar una programación de rutas que cumpla con todas las entregas de una empresa de despacho, minimizando las distancias recorridas y el número de camiones. Al igual que en el caso anterior se utiliza un ranking de fitness de Pareto para evaluar la calidad de las soluciones. Zhang, Wang y Tang [3] investigan el problema de diseño de rutas para mega-eventos donde hay un gran tráfico de transporte público y se requieren rutas adicionales para el transporte de las personas que participan de un evento, a su lugar de destino. En [3] se utiliza un algoritmo genético cuya función de evaluación es la distancia total del conjunto de rutas. Además, se utilizan dos penalizaciones: la primera se aplica por cada nodo de destino que quedó fuera de ruta y la segunda se aplica si la distancia total de una ruta supera la máxima distancia permitida.

Acercamientos multi-objetivos específicos para el UTRP se pueden encontrar en [16] donde se utiliza un algoritmo evolutivo que usa el operador *Make-small-change*. En este caso se utiliza el concepto de dominancia de Pareto para construir el conjunto de soluciones. Este algoritmo entrega buen conjunto de rutas desde el punto de vista de los pasajeros y con mejores costos para operadores en comparación a un algoritmo mono-objetivo similar con el que se comparó. Mumford en [12] utiliza exactamente el mismo modelado, pero agrega un operador

de cruzamiento al algoritmo evolutivo. El operador de cruzamiento selecciona intercaladamente rutas de ambos padres de manera tal de tener una cantidad equitativa de rutas de cada padre en el hijo. Posteriormente, en [8] John, Mumford y Lewis presentan un acercamiento multi-objetivo mejorado basado en NSGAII, junto con los operadores de cruzamiento de [12] y ocho operadores de mutación. En [11] se considera el mismo modelo y función objetivo que los anteriores, pero se realizan diferentes operaciones sobre un set de rutas para tratar de mejorar su calidad. Entre las modificaciones se encuentran: una selección mediante ruleta de las rutas a modificar, una operación crossover entre dos rutas padres, y la revisión de factibilidad de los hijos, para finalmnente realizar una operación de mutación sobre dos rutas pertenecientes al mismo set, intercambiando nodos aleatorios.

Para hacerse una idea de la complejidad real del problema es posible mencionar que el sistema de transportes de la ciudad de Santiago, Chile, Transantiago cuenta con 372 recorridos, y 11272 paradas disponibles. Dichas paradas se distribuyen en siete unidades agrupadas por zonas [5]. Además, cuenta con 7 tipos de servicios entre los que se cuentan: servicios normales, cortos, expresos, variantes, nocturnos, especiales e inyectados que operan en distintas condiciones horarias y de capacidad. Los estudios realizados en el caso particular de Santiago coonsideran microsimulaciones de focos de congestión considerando diferentes escenarios de aumento de la capacidad vial, reversibilidad de las pistas y simulación de incidentes [1], mientras que en [2] se presenta un metodo de resolución para el problema de asignación de horarios, rutas y asignación de choferes para una de las siete empresas del Transantiago (STP Santiago) que cuenta con aproximadamente 300 buses. El autor aborda el problema mediante MIP (Programación Entera Mixta).

3 Descripción del problema

El problema considera inicialmente que se cuenta con algunas variables, tales como un grafo G; el cual representa una red de paraderos, lo que se puede entender, por ejemplo, como la estructura que se tiene en una ciudad. Cada vértice corresponde a un paradero, mientras que los arcos corresponden a las conexiones o caminos existentes entre dichos paraderos. Se considera que el grafo es no dirigido, lo que quiere decir que un arco entre dos vértices indica que hay una conexión en ambos sentidos entre los arcos que conecta. Las posiciones de cada vértice también son conocidas. Además, este grafo posee matrices simétricas, que poseen valores de tiempo de viaje entre paradas, demandas asociadas para cada vértice.

Como restricciones del problema se puede considerar lo siguiente [12]:

- Cada ruta del conjunto de rutas está libre de ciclos y retrocesos.
- El conjunto de rutas está conectado.
- Hay exactamente r rutas en el conjunto de rutas.

– El número de nodos en cada ruta debe ser mayor a uno y no debe exceder el valor máximo definido.

Con esto se asume que se sabe la cantidad de rutas tendrá el conjunto de rutas resultante, además de que tendrán un rango paraderos permitidos, los cuales están acotados por un valor mínimo y un valor máximo.

El UTRP consiste en satisfacer tanto a operadores de buses, como a los pasajeros que los utilizan. Es por esto que se tienen las siguientes funciones objetivo, dadas por las Ecuaciones (1) y (2) [12].

La Ecuación (1) considera una minimización de los costos para los pasajeros. d_{ij} representa la demanda entre las paradas i y j y $\alpha_{ij}(R)$ corresponde al camino más corto entre las paradas i y j usando el conjunto de rutas R. Es asi, como esta ecuación puede verse como una minimización del tiempo de viaje promedio de todos los pasajeros, considerando las demandas en las paradas.

$$min \ \frac{\sum_{i,j=1}^{n} d_{ij}\alpha_{ij}(R)}{\sum_{i,j=1}^{n} d_{ij}} \tag{1}$$

Por otra parte, en la Ecuación (2) se considera una minimización de los costos para los operadores de buses. t_{ij} corresponde al tiempo de viaje entre las paradas i y j. Entonces, esta ecuación corresponde simplemente a la minimización del tiempo de viaje considerando todas las rutas. En [12] se le llama longitud total de la ruta.

$$min \sum_{a=1}^{r} \sum_{(i,j)\in r} t_{i,j}(a) \tag{2}$$

Al considerar el UTRP como problema multiobjetivo, se plantea una minimización de costos tanto para pasajeros como para operadores de buses.

4 Representación

Dentro del contexto del UTRP, se pretende determinar un conjunto de rutas a partir de distintos paraderos de buses preestablecidos, los cuales deben presentar un beneficio para los pasajeros y para el operador del bus.

Para las paradas se utiliza un grafo no dirigido, cuyos vértices corresponden a las paradas y los arcos a un camino entre las paradas. Un ejemplo de esta representación es la red de Mandl, que puede apreciarse en la Figura 1.

La representación utilizada en [10] para manejar el conjunto de rutas consiste en un arreglo bi-dimensional de r filas y $m + 1$ columnas, con r la cantidad de rutas de la solución y m la máxima cantidad de paraderos permitida. Cada fila distinta será una ruta, donde la primera columna almacena el número de la ruta. Para las otras columnas de la fila se utlizan identificadores para los paraderos de buses que componen la ruta. Por otra parte, los paraderos se encuentran representados por una matriz de demandas, una matriz de tiempos de viaje y

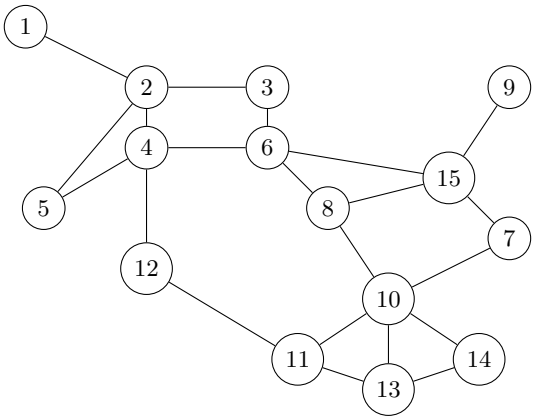
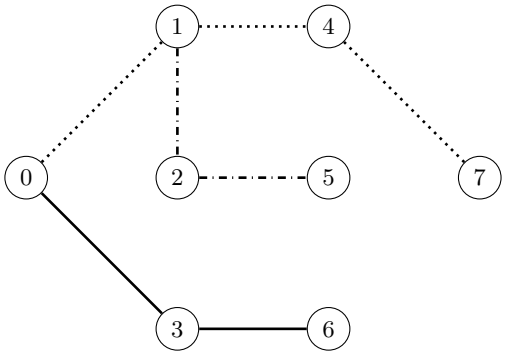


Fig. 1: Red de Mandl

por un sistema de coordenadas (este último para modelar la posición de cada paradero).



(a) Grafo con tres rutas.

R1	0	1	4	7
R2	0	3	6	∗
R3	1	2	5	∗

(b) Arreglo bi-dimensional con tres rutas.

Fig. 2: Representación para el conjunto de rutas.

Las matrices de tiempos de viaje y de demandas se representan mediante arreglos bidimensionales, donde la fila i y columna j aporta información entre el paradero i y el paradero j.

Un ejemplo representación, considerando un grafo de 7 paraderos y 3 rutas distintas, están dadas la estructura de la Figura 2. Se puede observar que la cantidad máxima de paraderos es 4, entonces la representación consiste de un arreglo de 3 filas y 5 columnas. El caracter asterisco al final de las rutas 2 y 3

denota que no hay un identificador de paradero, y por ende, la ruta tiene menos de cuatro paraderos.

5 Estructura del algoritmo

El Algoritmo Inmune Artificial implementado está basado en un algoritmo de selección clonal genérico, propuesto por de Castro y Von Zuben [13].

En el Algoritmo 1 se muestran un pseudocódigo del algoritmo implementado. Un anticuerpo se entiende dentro del contexto de los algoritmos inmunes artificiales como una solución candidata.

Algoritmo 1 Algoritmo Inmune Artificial basado en Selección Clonal

Entrada: Información del problema, información de instancia
Salida: Un conjunto de memoria M

```
1: Inicializar aleatoriamente población P de tamaño tam_pob
2: g ← 1
3: mientras g <=generaciones hacer
4:    para cada p perteneciente a la población P hacer
5:       Calcular aptitud de p
6:    fin para
7:    Eliminar de la población P anticuerpos dominados
8:    Seleccionar mp de los mejores anticuerpos de P
9:    Generar un conjunto de clones C de tamaño tam_clon con anticuerpos mp
10:   para cada clon c perteneciente al conjunto C hacer
11:      k ← numero de mutaciones a realizar de acuerdo a aptitud de c
12:      Mutar k veces el clon c
13:   fin para
14:   Copiar hasta mp clones a la población P, basándose en ranking de pareto
15:   Seleccionar mm anticuerpos de P
16:   Copiar los anticuerpos mm al conjunto de memoria M
17:   Reemplazar pp anticuerpos con anticuerpos generados aleatoriamente
18:   g ← g + 1
19: fin mientras
```

6 Parámetros del algoritmo

El algoritmo considera los siguientes parámetros:

- `generaciones`: criterio de término para el algoritmo inmune. Es la cantidad de iteraciones que deben ocurrir para que finalice el algoritmo. El rango de valores permitidos para este parámetro va de 0 a infinito.
- `alpha` y `beta`: representan variables de peso para ponderar las dos funciones objetivo y determinar una aptitud para una solución candidata. El rango de valores permitidos va de cero a uno. Además, la suma de ambas debe ser uno.

- `tam_pob`: representa el tamaño de la población P en cada generación. Una generación está compuesta por un conjunto de soluciones candidatas factibles. El rango de valores permitidos va de 1 a infinito.
- `tam_clon`: es el tamaño de la población de clones C que se generarán en cada generación a partir de las mejores soluciones candidatas. El rango de valores permitidos va de 1 a infinito.
- `porc_mejores`: Porcentaje de mejores soluciones de la población que serán seleccionadas para mutarse en cada generación. El rango de valores permitidos va de 0 a 1.
- `porc_clones`: porcentaje de mejores clones mutados que serán almacenados en el conjunto de memoria M. El rango de valores permitidos va de 0 a 1.
- `porc_reemplazo`: porcentaje de peores soluciones de la población que serán reemplazadas por soluciones aleatorias. El rango de valores permitidos va de 0 a 1.

7 Inicialización

El conjunto de soluciones iniciales es generado aleatoriamente. Una solución está compuesta por r rutas, donde cada una está acotada por un mínimo y máximo de paraderos. El algoritmo 2 muestra los pasos de la generación de la población inicial.

Como fue mencionado en la representación del problema, una solución se compone por un conjunto de rutas. A su vez cada ruta se compone de un conjunto de paraderos. Para la generación de soluciones se debe iniciar con la asignación de paraderos, los cuales al agruparse constituyen una ruta. Finalmente, un conjunto de rutas genera una solución.

Una ruta es generada escogiendo aleatoriamente la cantidad q de paraderos que tendrá (acotada por el mínimo y máximo de paraderos). Después se selecciona aleatoriamente un paradero. Luego, se busca una conexión directa desde este paradero a otros y se escoge aleatoriamente uno de ellos para formar parte de la ruta. Este proceso se repite hasta completar la cantidad q. Progresivamente, nuevas rutas son generadas de la misma forma, hasta completar r rutas. Un conjunto de soluciones se generará repitiendo los pasos anteriores de rutas y paraderos hasta llenar la población P con `tam_pob` individuos.

Las soluciones que se generan en este algoritmo siempre son factibles, ya que satisfacen las cuatro restricciones del problema:

- Cada ruta del conjunto de rutas está libre de ciclos y retrocesos: el conjunto de vecinos a considerar no contiene paraderos que ya estén en la ruta.
- El conjunto de rutas está conectado: es satisfecho considerando el conjunto de vecinos seleccionando solo aquellos paraderos con conexión directa, dado por el grafo de la red de paraderos. Si existe adyacencia entre dos vértices (paraderos) entonces se considera como un vecino.
- Hay exactamente r rutas en el conjunto de rutas: al generar solciones iniciales, se considera generar esta cantidad de rutas.

– El número de nodos en cada ruta debe ser mayor a uno y no debe exceder el valor máximo definido: el mínimo y máximo de paraderos está considerado al momento de generar una ruta. La cantidad de paraderos que tendrá una ruta se escoge aleatoriamente en un rango de valores posibles que es mayor o igual al mínimo de rutas y menor o igual al máximo de rutas.

Algoritmo 2 Inicialización de soluciones factibles para el UTRP

Entrada: tamaño de población `tam_pob`, cantidad r de rutas por solución, mínimo min de paredero para una ruta, máximo max de paraderos para una ruta
Salida: Conjunto P con población de soluciones factibles

```
1: P ← conjunto vacío
2: soluciones ← 0
3: mientras soluciones < tam_pob hacer
4:    rutas ← 0
5:    s ← nueva solución
6:    mientras rutas < r hacer
7:       nr ← nueva ruta
8:       q ← número aleatorio entre min y max
9:       i ← paradero seleccionado aleatoriamente
10:      agregar paradero i a ruta nr
11:      paraderos ← 1
12:      mientras paraderos < q hacer
13:         N ← vecinos de paradero i
14:         i ← paradero seleccionado aleatoriamente desde N
15:         agregar paradero i a ruta nr
16:         paraderos ← paraderos + 1
17:      fin mientras
18:      agregar ruta nr a solución s
19:      rutas ← rutas + 1
20:   fin mientras
21:   agregar solución s a población P
22:   soluciones ← soluciones + 1
23: fin mientras
```

8 Proceso iterativo

El algoritmo propuesto posee un proceso iterativo que se repite hasta satisfacer una cantidad de iteraciones o generaciones dadas por el parámetro `generaciones`. A continuación se detallarán los pasos realizados en cada una de estas generaciones.

8.1 Cálculo de aptitud

Cada elemento p de la población P es evaluado mediante una función de aptitud para determinar si es mejor o peor que otras soluciones. La aptitud de un

elemento p está dada por su conjunto de rutas, por lo tanto se deberá evaluar en dos funciones objetivo FO_1 y FO_2, dadas por las ecuaciones (3) y (4), que provienen de la minimización de costos para pasajeros y operadores, según se explicó en la definición del problema.

$$FO_1(p) = \frac{\sum_{i,j=1}^{n} d_{ij}\alpha_{ij}(R)}{\sum_{i,j=1}^{n} d_{ij}} \tag{3}$$

$$FO_2(p) = \sum_{a=1}^{r} \sum_{(i,j)\in r} t_{i,j}(a) \tag{4}$$

Dado que el orden de magnitud de las funciones objetivo es distinto, cada valor será normalizado por una cota inferior conocida para cada instancia. Así, se determinarán las magnitudes $\tilde{FO}_1$ y $\tilde{FO}_2$, según lo mostrado en las Ecuaciones (5) y (6).

$$\tilde{FO}_1(p) = \frac{FO_1(p)}{LB_{FO_1}} \tag{5}$$

$$\tilde{FO}_2(p) = \frac{FO_2(p)}{LB_{FO_2}} \tag{6}$$

Finalmente, la aptitud de p está dada por una poderación de $\tilde{FO}_1$ y $\tilde{FO}_2$, utilizando los parámetros `alpha` y `beta`. La función de aptitud entregará valores mayores o iguales a 1, siendo el valor 1 la mejor aptitud posible en el problema de minimización. La función de aptitud de un elemento p de la población se denotará por f_{apt} y se calculará mediante la ecuación (7).

$$f_{apt}(p) = \texttt{alpha} \cdot \tilde{FO}_1(p) + \texttt{beta} \cdot \tilde{FO}_2(p) \tag{7}$$

8.2 Eliminación de soluciones dominadas

El algoritmo inmune trabaja sobre soluciones factibles no dominadas. Al inicio de cada iteración, posteriormente al proceso de determinar la aptitud de las soluciones, se eliminan aquellas soluciones dominadas. Esto es, para realizar la transformación de soluciones que estén directamente en el frente de Pareto, con la intención de mejorarlas.

8.3 Selección de soluciones

Del conjunto de soluciones factibles no dominadas, se escoge un porcentaje de ellas para generar un conjunto de clones C. Las soluciones del conjunto P son ordenadas de acuerdo a su aptitud en orden ascendente, y luego las primeras mp soluciones se clonan de manera aleatoria en el conjunto C hasta completar el tamaño `tam_clon`. La cantidad mp es determinada utilizando el parámetro `porc_mejores` como se muestra en la Ecuación (8):

$$mp = \lceil |P| \cdot \texttt{porc\_mejores} \rceil \tag{8}$$

La cantidad mp se detemina multiplicando la cardinalidad del conjunto P por el parámetro `porc_mejores`. A esta operación se le aplica la función techo para aproximar al entero superior.

8.4 Operadores de transformación

Los operadores de transformación utilizados en este algoritmo consisten en tres operadores de mutación, diseñados para generar cambios distintos en las soluciones con la esperanza de mejorar la función de aptitud de los anticuerpos. Los operadores trabajan generando un movimiento sobre una ruta de una solución. Estos operadores son agregar un paradero a una ruta, eliminar un paradero de una ruta y reordenar una ruta mediante un punto de corte.

1. **Agregar una ruta**: Considerando una ruta r con n paraderos ordenados: $bs_1, bs_2, \ldots, bs_n$. Este operador busca un vecino del paradero bs_n y le agrega un nuevo paradero bs_{n+1} al final de la ruta, dejando la ruta con $n+1$ paraderos ordenados. Este operador es factible de realizar en gran parte de los casos, exceptuando las rutas que tienen el máximo de paraderos permitidos. La Figura 3 muestra un ejemplo con $n = 6$.

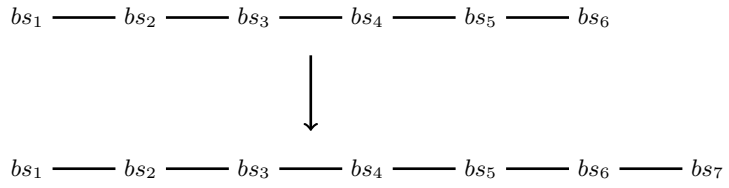


Fig. 3: Adición de un paradero al final de una ruta de 6 paraderos.

2. **Eliminar una ruta**: Considerando una ruta r con n paraderos ordenados: $bs_1, bs_2, \ldots, bs_n$. Este operador elimina el último paradero bs_n, dejando la ruta con $n-1$ paraderos ordenados. Este operador es factible de realizar en gran parte de los casos, exceptuando las rutas que tienen el mínimo de paraderos permitidos. La Figura 4 muestra un ejemplo con $n = 6$.
3. **Reordenar una ruta mediante un punto de corte**: Considerando una ruta r con n paraderos ordenados: $bs_1, bs_2, \ldots, bs_n$. Si los paraderos bs_1 y bs_n tienen conexión directa, entonces se genera un reordenamiento del orden de la ruta uniendo bs_1 y bs_n y seleccionando aleatoriamente un punto de corte en la conexión entre otras dos paradas. La ruta mantiene el número de paraderos. Este operador tiene una factibilidad menor a la de los otros operadores, ya que depende fuertemente de las conexiones entre paraderos. Se ha establecido

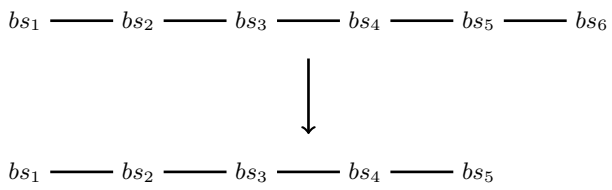


Fig. 4: Eliminación del último paradero en una ruta con 6 paraderos.

que se realizarán 20 intentos para intentar cambiar una solución mediante este operador. Menos intentos podrían descartar buenas opciones para la solución y más intentos impactarían en más tiempo de cómputo. En caso de fracaso, el operador entrega la misma solución que recibió inicialmente. La Figura 5 muestra un ejemplo con $n = 6$ y seleccionando como punto de corte la conexión entre el cuarto y quinto paradero.

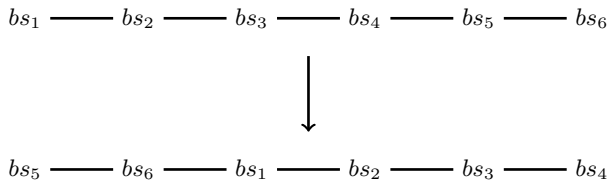


Fig. 5: Reordenamiento de una ruta con 6 paraderos, seleccionando como punto de corte la conexión entre bs_4 y bs_5.

La cantidad de veces que un clon debe ser mutado está determinado por su aptitud x con respecto a la mayor aptitud apt_{mayor} del conjunto de clones, la menor aptitud apt_{menor} del conjunto de clones y la cantidad de rutas n_{rutas} del clon. Para determinar un valor k, con la cantidad de veces que un clon debe ser mutado se utiliza la función $K(x)$, detallada en la Ecuación (9):

$$k = \lceil K(x) \rceil = \left\lceil 1 + \frac{n_{rutas} - 1}{apt_{menor} - apt_{mayor}} (x - apt_{menor}) \right\rceil \tag{9}$$

Dado que x puede tener valores entre la aptitud menor y la aptitud mayor, $apt_{menor} \leq x \leq apt_{mayor}$. Evaluando la función $K(x)$, se tiene que $K(apt_{menor}) = 1$ y $k(apt_{mayor}) = n_{rutas}$, por lo tanto $1 \leq K(x) \leq n_{rutas}$. En la Figura 6 se muestra la función $K(x)$ de manera gráfica. Esta función corresponde a una recta con pendiente positiva igual a $(n_{rutas} - 1)/(apt_{mayor} - apt_{menor})$. De esta forma, el clon dentro del conjunto de clones con mejor aptitud será mutado 1 vez, mientras que el clon con peor aptitud se mutará tantas veces como rutas

tenga. Clones que están en puntos intermedios se mutarán más a medida que tengan mayor aptitud. La cantidad k que determina la cantidad de veces que un clon se mutará utilizará esta función aproximada al entero inmediatamente superior. Con esta función se asegura la generación de diversidad de soluciones, donde todos los clones serán mutados al menos 1 vez y se mutarán más a medida que su aptitud sea mayor. Una consideración importante a destacar, para mantenerse en el terreno de las soluciones factibles, es que el operador de mutación 1 (agregar un paradero) está prohibido para rutas con el máximo de paraderos y el operador de mutación 2 (eliminar un paradero) está prohibido para rutas con el mínimo de paraderos.

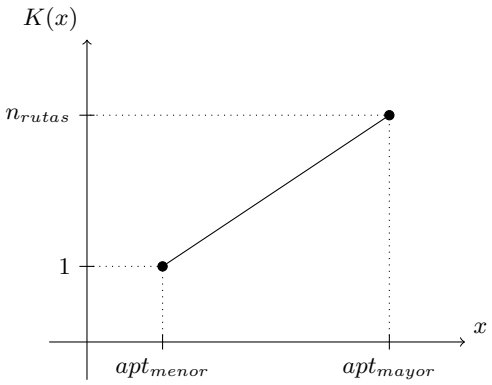


Fig. 6: Gráfica de la función $K(x)$

8.5 Selección de mejores soluciones para el conjunto de memoria

Luego de la mutación de clones, se copian hasta mp de los clones a la población P. El tope máximo de clones a copiar está acotado superiormente por el tamaño de la población `tam_pob`. El criterio escogido para copiar unos clones sobre otros es el ranking basado en la dominacia de pareto, mostrado en la Ecuación (10).

$$rank(x_i, t) = 1 + p_i^{(t)} \tag{10}$$

El ranking para una solución x_i con respecto a un conjunto t es $1 + p_i^{(t)}$, donde $p_i^{(t)}$ es la cantidad de soluciones que dominan a x_i en el conjunto t. Los clones seleccionados para pasar al conjunto P son escogidos por el valor de ranking que tengan con respecto al conjunto de clones. Los primeros clones en copiarse a la población P serán aquellos con menor valor de ranking, luego se seleccionarán clones en orden ascendente hasta completar la población `tam_pob` o hasta que se hayan copiado todos los clones en la población.

Los anticuerpos de la población P son ordenados de acuerdo a su aptitud, luego los primeros mm anticuerpos son copiados al conjunto de memoria M. Para determinar la cantidad mm se considera la cardinalidad de la población P y el parámetro `porc_clones`, según la Ecuación (11). mm es aproximado al entero inmediatamente superior.

$$mm = \lceil |P| \cdot \texttt{porc\_clones} \rceil \tag{11}$$

Al copiar soluciones al conjunto de memoria M se considera solo incluir soluciones no repetidas y no dominadas.

8.6 Selección de soluciones para conformar nueva población

Las peores *pp* soluciones (las últimas del conjunto ordenado P) son reemplazadas con nuevas soluciones generadas aleatoriamente, de acuerdo al Algoritmo 2. La cantidad *pp* es calculada considerando la cardinalidad de la población P y el parámetro `porc_reemplazo`, de acuerdo a la Ecuación 12. *pp* es aproximado al entero inmediatamente superior.

$$pp = \lceil |P| \cdot \texttt{porc\_reemplazo} \rceil \tag{12}$$

9 Experimentos

9.1 Datasets

Los datasets considerados para la experimentación corresponden a dos de los datasets mostrados en la literatura [8, 12]. El primero de ellos, Mandl, corresponde a una red conexa de 15 nodos y 20 arcos. El segundo, Mumford0, es una red de 30 nodos y 90 arcos. Dos instancias de los datasets se consideraron para la ejecución del algoritmo. En el caso de Mandl, se consideró una instancia con 6 rutas de entre 2 y 8 nodos por ruta. Para el caso de Mumford0, la instancia utilizada fue una de 12 rutas con 2 a 15 paraderos cada una. La Tabla 1 resume información de las instancias.

Instancia	Nodos	Arcos	Cant. Rutas	Paraderos/Ruta
Mandl	15	20	6	2-8
Mumford0	30	90	12	2-15

Tab. 1: Instancias a utilizar en la experimentación

Para normalizar la aptitud de las soluciones se utilizarán las cotas inferiores dadas en [12], las cuales se muestran en la Tabla 2.

Instancia	$LB_{FO_1}[min]$	$LB_{FO_2}[min]$
Mandl	10.0058	63
Mumford0	13.0121	94

Tab. 2: Cotas inferiores utilizados para normalizar la calidad de las soluciones.

9.2 Sintonización de Parámetros

Para encontrar los parámetros que obtienen mejores resultados se utilizará el sintonizador ParamILS [6], el cual determina la combinación de parámetros que minimizan un objetivo entregado por la instancia. El objetivo a minimizar por ParamILS será el hipervolumen. Sin embargo, ParamILS encontrará la combinación de parámetros para el menor hipervolumen negativo (el valor obtenido en una ejecución del algoritmo, multiplicado por menos uno) que es equivalente a maximizar el hipervolumen positivo del frente de pareto. Se sintonizarán 5 parámetros que tienen incidencia en la cantidad de soluciones seleccionadas en distintas etapas del algoritmo y en los tamaños de poblaciones de soluciones. Se fijará el parámetro `generaciones` en 250 para cada ejecución realizada por el sintonizador y se cambiarán los valores de los parámetros `alpha` y `beta` cada 50 iteraciones de manera aleatoria. En la Tabla 3 se muestran los parámetros a sintonizar, donde en cada caso, un valor por defecto será utilizado inicialmente y otros 3 valores adicionales serán entregados a ParamILS.

Parámetro	Defecto	Valores a utilizar			
`porc_mejores`	1.00	0.25	0.50	0.75	1.00
`porc_clones`	0.50	0.25	0.50	0.75	1.00
`porc_reemplazo`	0.30	0.10	0.30	0.50	0.70
`tam_pob`	100	50	100	150	200
`tam_clon`	150	50	100	150	200

Tab. 3: Valores a utilizar para la sintonización de parámetros.

9.3 Búsqueda de mejores soluciones

Una vez determinados los mejores parámetros por cada instancia, se realizarán 10 pruebas adicionales con estos parámetros usando 15 semillas distintas. Se fijará el parámetro `generaciones` en 1000 para cada ejecución del algoritmo y se cambiarán los valores de los parámetros `alpha` y `beta` cada 50 iteraciones de manera aleatoria.

10 Resultados

El sintonizador de parámetros obtuvo los mejores parámetros para las dos instancias analizadas, los cuales se muestran en la Tabla 4.

Instancia	`porc_mejores`	`porc_clones`	`porc_reemplazo`	`tam_pob`	`tam_clon`
Mandl	1.00	0.25	0.30	150	200
Mumford0	0.25	0.25	0.30	150	150

Tab. 4: Valores entregados por el sintonizador para cada instancia.

De acuerdo a estos resultados, se puede apreciar que el parámetro `porc_mejores` se mantuvo en el valor por defecto para Mandl, pero disminuyó al mínimo posible en Mumford0. Probablemente, esto se debe a que la red de Mumford0 es más grande y se tiene mejor esperanza de mejorar las soluciones si se toma una fracción de las mejores soluciones para pasar al proceso de mutación.

El mejor valor para el parámetro `porc_clones` fue asignado al mínimo posible en ambas instancias, lo que indica que se toma una fracción pequeña de valores para almacenar en el conjunto de memoria, lo que tiene una ventaja computacional al evitar almacenar soluciones que podrían ser eliminadas posteriormente.

El parámetro `porc_reemplazo` se mantuvo en el valor por defecto sugerido en ambas instancias y probablemente es el mejor valor para un *trade-off* entre exploración y explotación en una vecindad de soluciones candidatas.

Los tamaños de población, reflejados en los parámetros `tam_pob` y `tam_clon` aumentaron con respecto a los valores por defecto. Al comparar las cantidades de ambas poblaciones, se cumple que el tamaño de la población de clones siempre es mayor o igual al tamaño de la población de cada generación.

Al realizar experimentos para la instancia de Mandl con los mejores valores de los parámetros con 15 semillas distintas, se puede observar la variación de hipervolumen para todas ellas en la Figura 7. Se aprecia que el comportamiento de la gráfica es el esperado para un algoritmo que va mejorando sus soluciones en cada generación. Se observa que en todos los casos hay periodos de estancamiento en las soluciones obtenidas, pero en ocasiones se produce una mejora súbita para pasar nuevamente a otro periodo de estancamiento. En general, a partir de la generación 200 no se observa una gran mejora en las soluciones obtenidas, ya que el hipervolumen se mantiene en niveles estables para cada caso. La Figura 8 muestra la distibución de los valores del hipervolumen obtenidos con las 15 semillas en las generaciones 1, 50, 100 y 1000 del algoritmo para la instancia Mandl, utilizando los mejores valores de los parámetros para esta instancia. Se observa que a medida que aumentan las generaciones del algoritmo la mayoría de los valores se acerca a un valor de hipervolumen comprendido entre 800 y 850. Se puede decir entonces, que el algoritmo inmune propuesto no es afectado en gran medida por la aleatoridad introducida al utilizar distintas semillas en esta instancia.

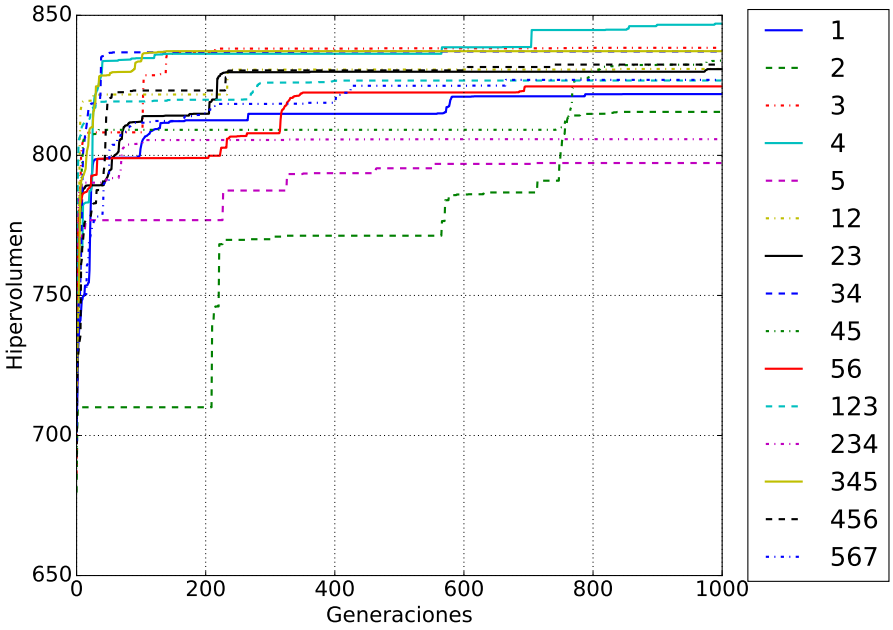


Fig. 7: Hipervolumen obtenido en 15 semillas distintas con la instancia Mandl con sintonización de parámetros.

A continuación, los frentes de Pareto resultantes de 4 semillas se pueden apreciar en las Figuras 9 y 10. Además, información adicional de apoyo referente a estos frentes se encuentra en la Tabla 5. En los casos mostrados, se observa que los frentes de Pareto se acercan a los mínimos de ambas funciones a medida que aumenta el número de generaciones. Adicionalmente, se puede ver la aparición de nuevas soluciones que no estaban definidas con anterioridad. El algoritmo propuesto es capaz de encontrar nuevas soluciones dentro de un vecindario de soluciones similares, producto de los operadores de mutación aplicados, los cual queda en evidencia con el aumento de soluciones del frente al aumentar el número de generaciones. Por otra parte, se obtienen mejoras en las soluciones ya encontradas, al visualizar el avance del frente de Pareto hacia el sector inferior izquierdo que tiene los mínimos de las dos funciones objetivo. Otro aspecto a notar en estas gráficas es la forma en que los frentes se van acercando al sector inferior izquierdo. Se observa claramente que el algoritmo propuesto obtiene muchas soluciones que son buenas para los operadores y pocas soluciones que son buenas para los pasajeros.

Por medio de la información de la Tabla 5 se destaca para este algoritmo en específico la aparición de muchas soluciones idénticas en los frentes de Pareto. Se puede observar, a modo general que desde la generación 50 y la generación

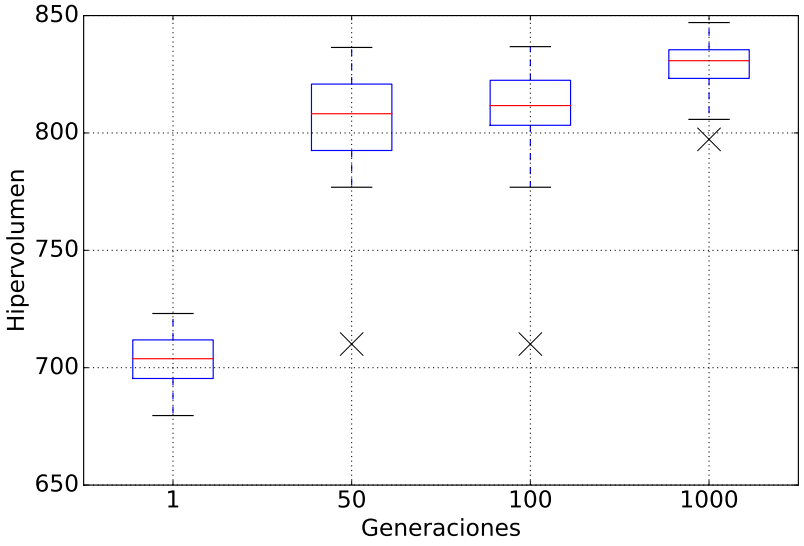


Fig. 8: Box plot del hipervolumen obtenido en distintas generaciones del algoritmo inmune para la instancia Mandl con sintonización de parámetros.

Semilla	Generación	Sol. Únicas	Sol. Totales	Hipervolumen	Tiempo [s]
4	1	**15**	**15**	702.207	26
45	1	14	14	692.819	25
12	1	12	12	**714.867**	**22**
34	1	13	13	704.389	**22**
4	50	34	34	833.746	334
45	50	**36**	**36**	809.110	344
12	50	27	27	821.737	347
34	50	19	19	**836.426**	**325**
4	100	35	65	834.683	645
45	100	**36**	**70**	809.110	663
12	100	27	27	821.737	683
34	100	20	37	**836.769**	**634**
4	1000	**37**	**335**	847.025	6770
45	1000	25	83	833.812	6533
12	1000	26	153	830.848	6675
34	1000	19	180	**837.087**	**6357**

Tab. 5: Cantidad de soluciones únicas/totales, hipervolumen y tiempo de cómputo en el frente de Pareto para 4 semillas de la instancia Mandl.

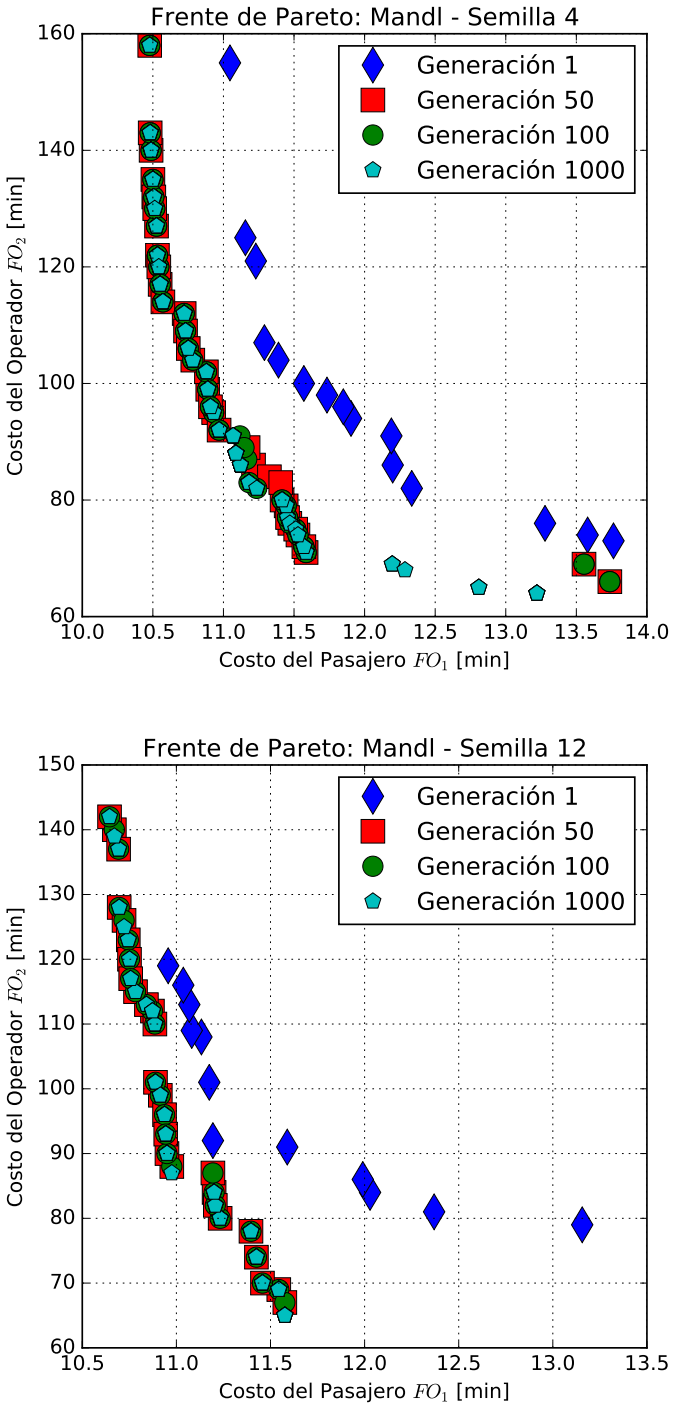


Fig. 9: Frente de Pareto para Mandl con semillas 4 y 12.

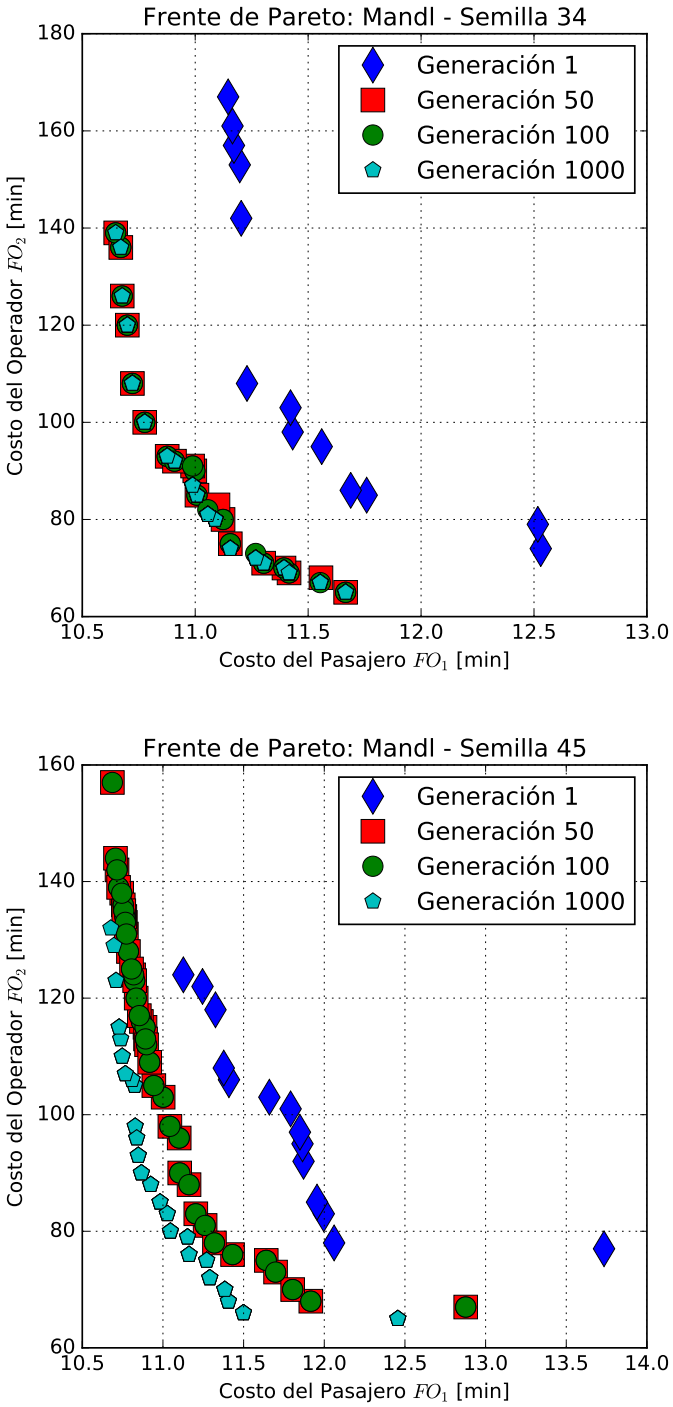


Fig. 10: Frente de Pareto para Mandl con semillas 34 y 45.

100 la soluciones totales son el doble de las soluciones únicas. En la generación 1000, las soluciones únicas representa entre un 10% y un 30% de las soluciones totales. En esta instancia se mantiene la tendencia de que la semilla que genera más soluciones únicas es la que tiene más soluciones totales. No existe una correlación entre un mayor hipervolumen con más soluciones únicas/totales en el frente. Finalmente, el tiempo para una instancia fue medido desde el inicio del algoritmo hasta la finalización de una generación. Tiempos menores en etapas prematuras del algoritmo suelen derivar en un tiempo total de cómputo menor para el algoritmo completo. Notar que al término de la generación 1 han transcurrido cerca de 20 segundos desde el inicio del algoritmo, donde este tiempo está destinado a la generación del conjunto inicial de soluciones factibles para iniciar el algoritmo.

En el caso de la instancia Mumford0, se realizó el mismo procedimiento, utilizando 15 semillas distintas con 1000 generaciones y los mejores valores para los parámetros. La Figura 11 muestra la evolución del hipervolumen para todas las semillas utilizadas y la Figura 12 muestra la distribución de los valores de hipervolumen para las semillas en las generaciones 1, 50, 100 y 1000 del algoritmo. En la Figura 11 se observa que el hipervolumen aumenta su valor en todas las semillas utilizadas, notando estancamientos prolongados en este valor desde las 200 generaciones aproximadamente. En el *box plot* de la Figura 12 se puede notar la aparición de *outliers*. Sin embargo, para todas las semillas se observa el mismo comportamiento creciente, visto anteriormente. A modo general, el algoritmo no se ve afectado por la aleatoridad de una semilla determinada y tiende a obtener un valor de hipervolumen acotado.

Las Figuras 13 y 14 muestran la evolución del frente de Pareto en 4 semillas distintas y la Tabla 6 muestra información adicional respecto a estos frentes. Se puede apreciar que la evolución del frente para semillas presenta un comportamiento favorable, ya que se acerca a la región de mínimos en ambas funciones objetivo. Un caso notable a mencionar ocurre con la semilla 4, donde se observa un gran avance a partir del frente inicial en la primera generación hasta el frente obtenido en la generación 50. A diferencia de la instancia Mandl, en estos casos se obtiene una buena cantidad de soluciones que son mejores para operadores y para pasajeros y el algoritmo tiene ciertas dificultades para acercarse a soluciones ubicadas en la zona inferior izquierda donde está el mínimo para operadores y pasajeros simultáneamente.

Al analizar los resultados de la Tabla 5 se puede observar que, en general, existe una repetición de soluciones candidatas que aumenta a medida que transcurren las generaciones. En esta instancia la cantidad de soluciones únicas representan una mayor parte de las soluciones totales que en la instancia Mandl. En las cuatro semillas mostradas, las soluciones únicas representan más del 40 % de las soluciones totales en la milésima generación. En la semilla 45 en específico, la cantidad de soluciones únicas es igual a la cantidad de soluciones totales. Nuevamente, no es posible relacionar la cantidad de soluciones con el hipervolumen obtenido para una misma semilla en una generación determinada. Además, tampoco es posible afirmar que una semilla que inicia con un buen

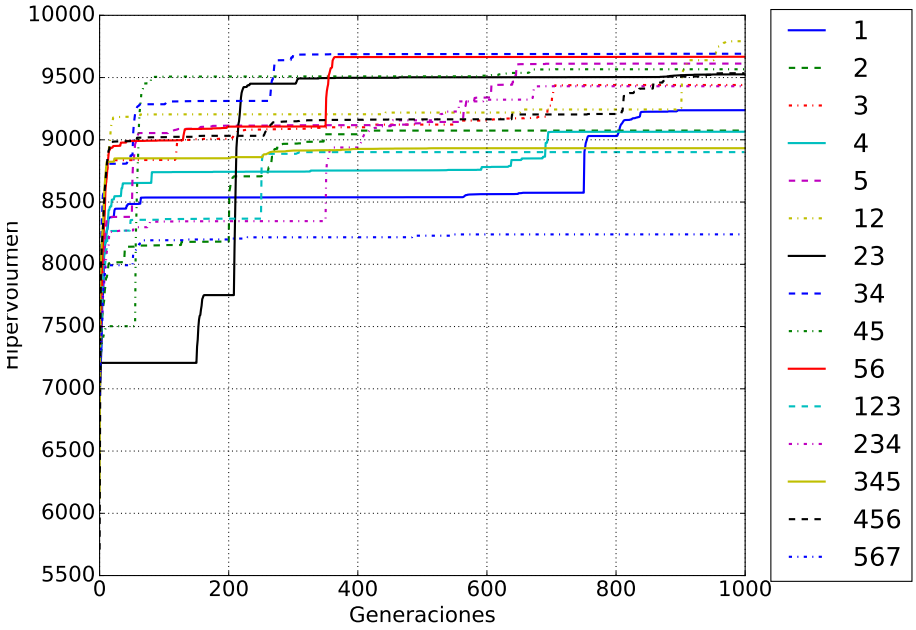


Fig. 11: Hipervolumen obtenido en 15 semillas distintas con la instancia Mumford0 con sintonización de parámetros.

valor de hipervolumen, seguirá siendo la mejor semilla en la generación 1000. Los tiempos registrados al finalizar una generación dan cuenta de la misma relación que en Mandl, donde semillas que tardan menos tiempo en llegar a una generación tempranamente mantendrán la tendencia y demorarán menos tiempo en terminar con las 1000 generaciones.

La Tabla 7 muestra la mejor solución obtenida para los pasajeros en cada instancia, utilizando los valores de los parámetros sintonizados. En ninguno de los dos casos se logró llegar a la cotas inferiores dadas por LB_{FO_1} según la Tabla 2, donde en el caso de Mandl la mejor solución tarda 0.47 minutos más que la cota inferior y en Mumford0 la mejor solución tarda 4.05 minutos más que la cota inferior. Las mejores soluciones para pasajeros están compuestas en su totalidad por rutas extensas.

Por otra parte, la Tabla 8 muestra la mejor solución para los operadores en cada instancia, utilizando los valores de parámetros sintonizados. Al comparar los valores obtenidos con las cotas inferiores de la Tabla 2, se logró llegar a la cota inferior de 63 minutos para la instancia Mandl, mientras que en Mumford0 se llegó a una solución que excede en 18 minutos a la cota inferior. Las soluciones obtenidas se caracterizan por ser rutas cortas en su mayoría. En cada caso una de las rutas del conjunto es considerablemente más extensa que el resto.

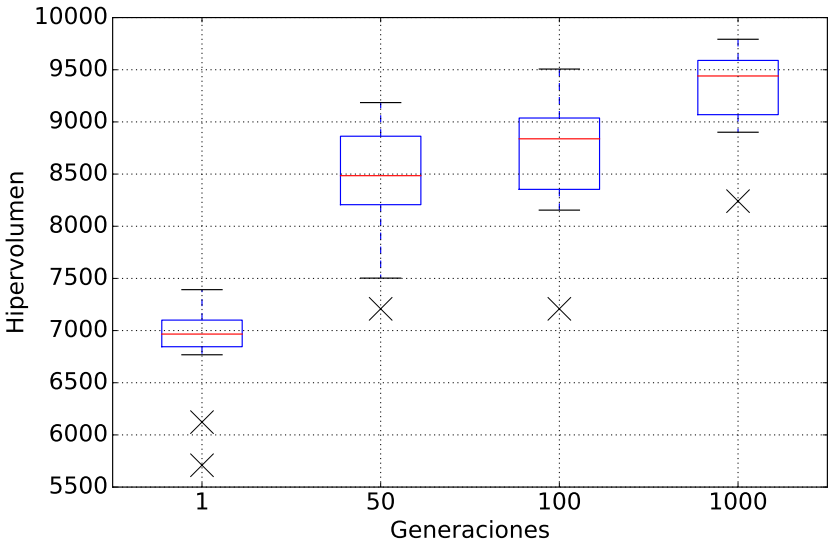


Fig. 12: Box plot del hipervolumen obtenido en distintas generaciones del algoritmo inmune para la instancia Mumford0 con sintonización de parámetros.

Semilla	Generación	Sol. Únicas	Sol. Totales	Hipervolumen	Tiempo [s]
4	1	**15**	**15**	7118.92	**46**
45	1	12	12	7054.21	52
12	1	9	9	7125.38	55
34	1	12	12	**7392.27**	49
4	50	23	23	8650.36	**1219**
45	50	20	20	7502.79	1742
12	50	**30**	**30**	**9185.21**	1349
34	50	22	22	8874.78	1276
4	100	26	33	8740.11	**2280**
45	100	38	38	**9506.56**	3062
12	100	27	30	9204.48	2622
34	100	**42**	**42**	9285.58	3158
4	1000	42	**103**	9064.71	**24308**
45	1000	38	38	9566.61	31216
12	1000	64	85	**9792.15**	27600
34	1000	**65**	83	9691.16	27324

Tab. 6: Cantidad de soluciones únicas/totales, hipervolumen y tiempo de cómputo en el frente de Pareto para 4 semillas de la instancia Mumford0.

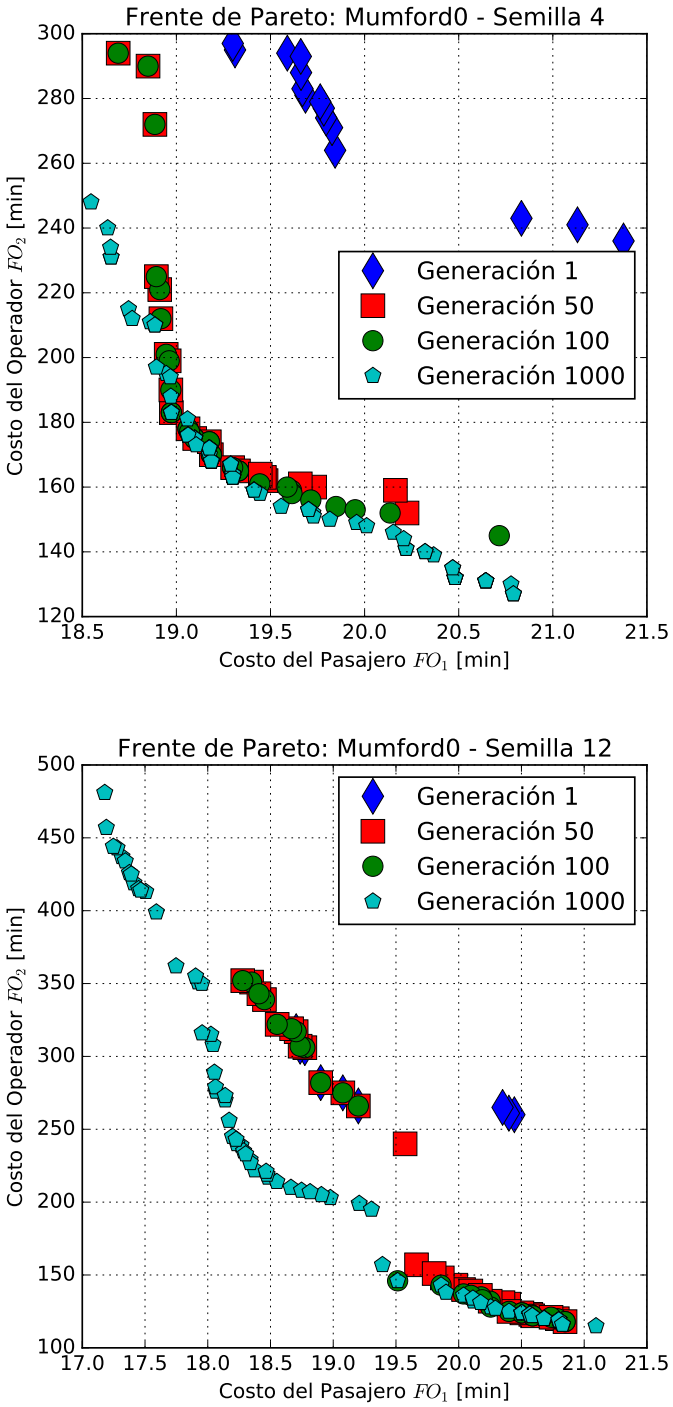


Fig. 13: Frente de Pareto para Mumford0 con semillas 4 y 12.

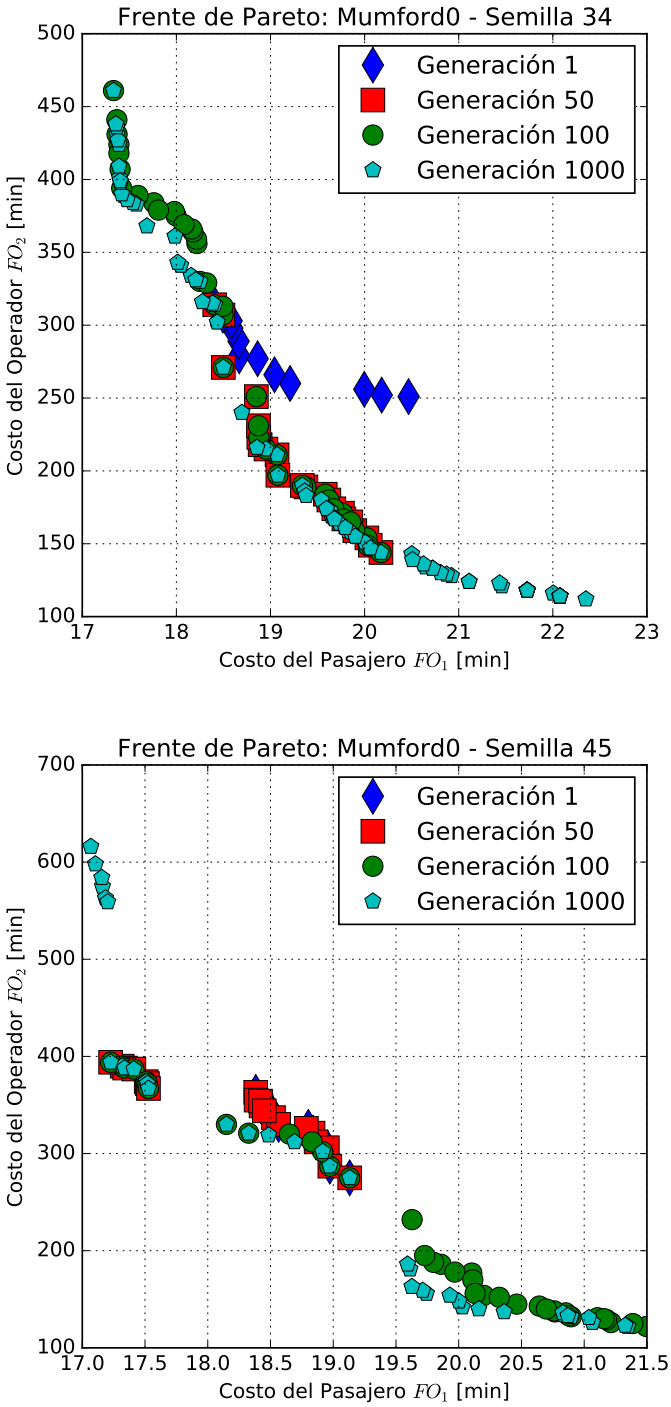


Fig. 14: Frente de Pareto para Mumford0 con semillas 34 y 45.

Instancia	Mejor solución para pasajeros (sintonización)	
	Costos	Rutas
Mandl	$FO_1 = 10.4778$ $FO_2 = 158$ (semilla=4)	9-15-6-3-2-1 3-2-5-4-6-8-10 4-2-3-6-15-7 11-10-8-6-3-2-1 9-15-7-10-11-12-4-2 11-13-14-10-7
Mumford0	$FO_1 = 17.0681$ $FO_2 = 616$ (semilla=45)	8-29-17-3-16-22-7-14-1-13-9-20-18-12 24-15-2-5-4-10 27-1-26-12-15-8-3-7-6-16-11 2-5-25-8-3-30-28-16-22-11-7-6 18-1-14-7-6-22-3-16-30-28-8-5-2 27-9-13-20-23-18-1-19-14-7-3-28-17-8 7-3-11-17-16-28-8-21-15-10 7-14-19-1-26-23-18-12-4-2-5 21-8-26-1-13-9 4-2-5-8-3-7-14-19-13-23-1-20-18-12-15 30-3-22-6-16-7-14-1-20-18-29-17-28-11-8 28-11-22-6-7-17-16-3-8-25-2-4-5-15

Tab. 7: Mejores soluciones para pasajeros por instancia.

Instancia	Mejor solución para operadores (sintonización)	
	Costos	Rutas
Mandl	$FO_1 = 13.0501$ $FO_2 = 63$ (semilla=567)	11-13-14 9-15-7 15-8-6-3-2-4-5 2-1 7-10 12-11-10
Mumford0	$FO_1 = 22.3516$ $FO_2 = 112$ (semilla=34)	15-21 5-25-15 3-30-11-7 22-11 28-17-7-6 29-26-12 19-1 16-7-14 2-4-10-24 2-5-8-29-18-19-13-20-9-27 1-23 14-1

Tab. 8: Mejores soluciones para operadores por instancia.

La Tabla 9 condensa los resultados asociados a los mejores valores de funciones objetivo obtenidos para Mandl y Mumford0 y los compara con las cotas inferiores de cada una de estas instancias.

Instancia	LB_{FO_1}	Mejor FO_1	LB_{FO_2}	Mejor FO_2
Mandl	10.0058	10.4778	**63**	**63**
Mumford0	13.0121	17.0681	94	112

Tab. 9: Tabla resumen comparativa con mejores valores de funciones objetivo para operadores y pasajeros.

Un último resultado comparativo se obtiene utilizando una población inicial de 200 y 200 generaciones, según se explica en [8]. Para esto se intentó replicar tanto como sea posible el mismo experimento, considerando las diferencias existentes con el algoritmo inmune propuesto. para esto se ejecutó el algoritmo inmune con 20 semillas, manteniendo los parámetros, con excepcion de `tam_pob` y `generaciones`, en los valores obtenidos en la sintonización de parámetros. La Tabla 10 muestra los resultados obtenidos con el MOEA propuesto por Mumford, mejorado con una heurística mencionada en [8] y denotado por MOEA-H con el algoritmo inmune propuesto, mencionado como AIA.

		Mandl		Mumford0	
		MOEA-H	AIA	MOEA-H	AIA
Mejor para pasajeros	FO_1	**10.25**	10.52	**15.40**	17.34
	FO_2	212	149	745	439
Mejor para operadores	FO_1	13.48	12.97	32.78	21.41
	FO_2	**63**	**63**	**95**	116

Tab. 10: Tabla comparativa con resultados obtenidos en [8].

Se puede observar que para la instancia Mandl, el algoritmo propuesto en la literatura es mejor que el algoritmo inmune considerando lo mejor para pasajeros. En la mejor solución para operadores, tanto el algoritmo MOEA-H como AIA logran llegar a la cota inferior de 63 minutos. No obstante, AIA logra mejorar el valor del costo de pasajeros en esta misma solución. En Mumford0, tanto para pasajeros, como para operadores el algoritmo de la literatura obtiene mejores resultados que el algoritmo inmune propuesto. A modo general, cuando se busca una mejor solución para pasajeros, AIA obtiene un menor costo para operadores que MOEA-H y cuando se busca la mejor solución para operadores, AIA obtiene un menor costo para pasajeros que MOEA-H.

11 Conclusiones

El problema de enrutamiento de tránsito urbano, o UTRP en inglés, es un tipo de problema complejo y contingente en los tiempos actuales, ya que intenta satisfacer las necesidades de las personas y de las empresas que se encargan de administrar el transporte público. Este es un problema NP-duro que involucra diseñar un conjunto de rutas a partir de un sistema compuesto por paraderos y rutas que los conectan. Este problema ha sido tratado como un problema mono-objetivo y más recientemente como un problema multi-objetivo por medio de heurísticas y algoritmos evolutivos.

El Algoritmo Inmune Artificial (AIA) toma como base un sistema inmune donde un conjunto de anticuerpos debe enfrentarse a las infecciones que atacan a un organismo y además almacenan información para evitar nuevas infecciones futuras. El AIA propuesto en este trabajo plantea resolver el UTRP como un problema multi-objetivo, de manera de reducir tanto los costos de los pasajeros que utilizan el transporte público asi como los costos de los operadores que manejan el sistema de transporte y obteniendo por tanto un conjunto de soluciones. Los costos asociados que se plantean minimizar están cuantificados por tiempos, ya que la información que se tiene de este problema consiste principalmente en tiempos de desplazamiento entre paradas. Se plantea una inicialización de soluciones factibles, de manera de satisfacer con las restricciones impuestas en el problema. Luego por medio de tres operadores de mutación, se realizan cambios en solcuciones candidatas y se almacena una parte de las mejores soluciones en un conjunto de memoria. Parámetros asociados a los tamaños de poblacion y a selecciones de soluciones que pasarán a otra etapa son considerados en este algoritmo. Una propuesta interesante para este trabajo tiene relación con la aplicación de operadores de transformación de manera proporcional a la aptitud de una soliución candidata y la selección de mejores soluciones utilizando el ranking de dominancia.

Dos instancias conocidas en la literatura fueron utilizadas para encontrar los mejores parámetros en cada caso por medio de un proceso de sintonización. La primera de ellas, Mandl, tiende a descartar menos soluciones que la segunda, Mumford0, al obtener un mayor valor en el parámetro que selecciona soluciones para mutar. El conjunto de soluciones fue cuantificado mediante la métrica de hipervolumen y fue mostrado gráficamente en frentes de Pareto. Para ambas instncias, el conjunto de soluciones no parece verse afectado por la elección de números aleatorios y entregan valores de hipervolumen estables y en rangos definidos. Adicionalmente, la métrica de hipervolumen crece a medida que transcurren las generaciones. Sin embargo, los frentes de Pareto se comportan de distinta forma. Como Mumford0 es una instancia considerablemente más grande que Mandl, los tiempos de cómputo por semilla son 4 a 5 veces mayores. En la instancia Mandl, hubo una tendencia a generar muchas soluciones que benefician a operadores y pocas soluciones que benefician a pasajeros, mientras que en Mumford0 no se observó este fenómeno.

Al hacer un anásisis de las mejores soluciones para pasajeros, en ambas instancias se obtuvieron rutas extensas, que suponen una minimización para el

tiempo de viaje promedio al evitar tener que hacer transbordos. En el caso de las mejores soluciones para operadores, las soluciones están conformadas mayormente por rutas cortas, lo que es consistente con la minimización de la función objetivo de operadores en alusión a minimizar el tiempo de viaje de todas las rutas. Al comparar las mejores soluciones con las cotas inferiores definidas para cada instancia, en Mandl se obtuvo una menor diferencia con estas cotas que en Mumford0. Un caso a notar es la mejor solución en Mandl para operadores, la cual no puede ser mejorada porque es la cota inferior de la instancia.

En la literatura se plantea un algoritmo evolutivo multi objetivo, MOEA, el cual es mejorado mediante una heurística que propone múltiples operadores de mutación. Si bien, el análisis no es exactamente comparable por las diferencias entre ambos acercamientos, el AIA obtuvo resultados cercanos a los planteados por esta técnica. Un aspecto a notar en la técnica MOEA de la literatura es el foco en mejorar las soluciones extremas del frente de Pareto y que el AIA propuesto no considera. Es por esto, que el AIA obtiene mejores valores de operadores en la mejor solución de pasajeros y viceversa.

Como trabajo a futuro, queda planteado un mejoramiento al AIA por medio de nuevos operadores de mutación, los que podrían ayudar a encontrar nuevos sectores inexplorados en el conjunto de soluciones. Por ejemplo, probar un operador que trabaje sobre dos rutas podría ser mejor que los tres operadores propuestos, aunque la factibilidad de que el operador sea exitoso y logre realizarse una tranformación podría ser menor.

12 Bibliografía

References

1. Victor Zuñiga Alarcón. *Uso de herramientas de microsimulación para la definición de estrategias de control de tránsito para la ciudad de Santiago*. PhD thesis, Universidad de Chile, 2010.
2. Cristián Cortés, Pablo Rey, Priscila Molina, Mario Recabal, and Sebastián Souyris. Uso de modelos de optimización para el apoyo a la operación de un alimentado de transantiago. 2009.
3. J. Tang D. Zhang, H. Wang. A hybrid genetic algorithm for the special transit route designing problem in mega-events. In *Control and Decision Conference, 25th Chinese*, pages 2426–2429, 2013.
4. Leandro Nunes de Castro y Jon Timmis. *An Introduction to Artificial Inmune Systems: A New Computational Intelligence Paradigm*. Springer-Verlag, 2002.
5. Gobierno de Chile. Portal de datos públicos. http://datos.gob.cl/datasets/ver/1587, Noviembre 2013.
6. Frank Hutter, Holger H. Hoos, Kevin Leyton-Brown, and Thomas Stutzle. Paramils: An automatic algorithm configuration framework. *Journal of Artificial Intelligence Research*, 36:267–306, October 2009.
7. Hong Jiang, Qingsong Yu, and Yong Huang. An improved ant colony algorithm for urban transit network optimization. In *Natural Computation (ICNC), 2010 Sixth International Conference on*, volume 5, pages 2739–2743. IEEE, 2010.

8. Matthew P John, Christine L Mumford, and Rhyd Lewis. An improved multi-objective algorithm for the urban transit routing problem. In *European Conference on Evolutionary Computation in Combinatorial Optimization*, pages 49–60. Springer, 2014.
9. Talbi Josefowiez, Semet. An evolutionary algorithm for the vehicle routing problem with route balancing. *European Journal of Operational Research*, 195:761–769, 2009.
10. Christine L. Mumford Lang Fan. A metaheuristic approach to the urban transit routing problem. *Journal of Heuristics*, 1(16):353–372, 2010.
11. Joanne Suk Chun Chew Lai Soon Lee. A genetic algorithm to the urban transit routing problem. *International Journal of Modern Physics*, 9, 2012.
12. Christine L. Mumford. New heuristic and evolutionary operators for the multi-objective urban transit routing problem. In *2013 IEEE Congress on Evolutionary Computation*, pages 939–946, 2013.
13. Mark Read, Paul S Andrews, and Jon Timmis. An introduction to artificial immune systems. In *Handbook of Natural Computing*, pages 1575–1597. Springer, 2012.
14. K.C. Tan. A hybrid multiobjective evolutionary algorithm for solving truck and trailer vehicle routing problems. *European Journal or Operational Research*, 172:855–885, 2006.
15. Nareli Cruz Cortés y Carlos A. Coello Coello. Un sistema inmune artificial para solucionar problemas de optimización multiobjetivo. Congreso Mexicano de Computación Evolutiva, 2003.
16. Jie Zhang, Huapu Lu, and Lang Fan. The multi-objective optimization algorithm to a simple model of urban transit routing problem. In *Natural Computation (ICNC), 2010 Sixth International Conference on*, volume 6, pages 2812–2815. IEEE, 2010.
17. Xie Binglei Zhao Hang. A memetic algorithm for optimization of urban transit route network. In *Seventth International Conference on Natural Computation*, pages 1899–1903, 2011.