Algoritmos Inmunes Artificiales en el UTRP

Cristopher Arenas
cristopher.arenas@alumnos.usm.cl

Departamento de Informática
Universidad Técnica Federico Santa María

Abstract. Los algoritmos inmunes artificiales (AIA) son estrategias de resolución de problemas evolutivas, lo que quiere decir que cambian con el tiempo y representan analogías de ciertos comportamientos biológicos. Estos algoritmos se puede utilizar en ciertos problemas, como por ejemplo el Urban Transit Routing Problem UTRP, en el cual se pretende encontrar rutas que satisfagan a operadores de los buses y a las personas que transportan. Por medio de un modelamiento, una representación del problema y la resolución por medio de un AIA se encontrará una solución que pueda satisfacer un conjunto de restricciones y trabajar con ciertos objetivos.

1 Introducción

Existen acercamientos que se encargan de resolver el Urban Transit Routing Problem (UTRP) [8]. Entre ellos se encuentran algoritmos evolutivos que emulan ciertos comportamientos de la naturaleza y los organismos de manera de encontrar soluciones a este tipo de problemas.

Dentro del grupo de los algorimos evolutivos, existen algunos que implementan ciertos comportamientos observados en los organismos de las personas. Los algoritmos inmunes artificiales representan una metáfora del sistema inmune donde se tiene un conjunto de anticuerpos que tienen la misión de encargarse de los virus y bacterias de enfermedades que atacan a uu organismo. Los anticuerpos funcionan a través de procesos adaptativos, pues una vez que encuentran la mejor forma de curar una enfermedad se capacitan para futuras infecciones [4] [12].

En el presente documento se implementará un algoritmo evolutivo que se enfoca en simular el comportamiento del sistema inmune para resolver el UTRP. Posteriormete, se escogeran parámetros adecuados para el problema mediante un proceso de sintonnización. Finalmente, se mostrarán algunos experimentos con instancias y las conclusiones del trabajo realizado.

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 URSP 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 [8] 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: Agregar 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 [14] 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 [6] 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 [7] 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 [11] 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 [13] 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 [10] 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. En [9] se considera el mismo modelo y función objetivo que los anteriores, pero es 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 como la estructura que se tiene en una ciudad por ejemlo. 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 [10]:

- 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 cuantas rutas tendrá el conjunto de rutas resultante, además de que tendrán un rango paraderos permitidos, los cuales están acotados por un mínimo y un 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 (1) (2) [10].

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

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

La función objetivo (1) considera los costos del operador. d_{ij} representa la demanda entre las paradas i y j y α_{ij} corresponde al camino más corto entre las dos paradas. Por otra parte, la función objetivo (2) corresponde a los costos del pasajero. t_{ij} corresponde al tiempo de viaje entre las paradas i y j. Los costos que se pretende satisfacer es la demanda, por parte de los operadores y los tiempos de viaje por parte de los pasajeros.

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, la cual puede apreciarse en la Figura 1.

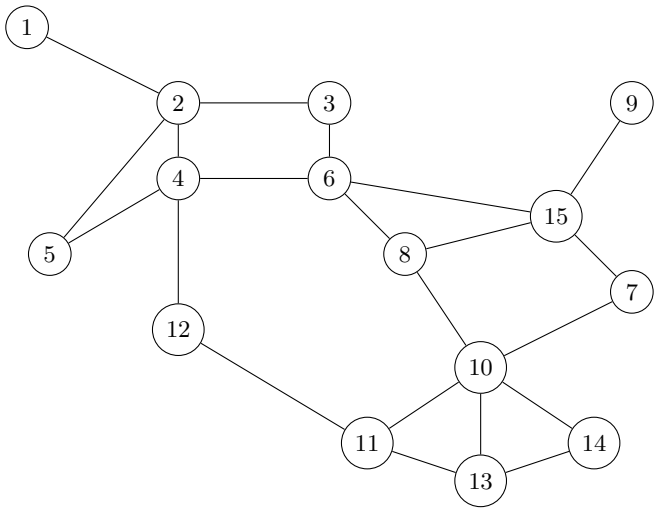

Fig. 1. Red de Mandl

La representación escogida para manejar el UTRP es mediante un arreglo bi-dimensional que representa un conjunto de rutas. Cada fila será una ruta, la cual contiene un identificador asociado al número de ruta y posteriormente identificadores para las paraderos de buses. Estos paraderos se encuentran representados por una matriz de demandas, una matriz de tiempos de viaje y por un sistema de coordenadas (este último para modelar la posición de cada paradero).

En la Figura 2 se muestra un ejemplo de una representación de rutas el cual se pudo obtener a partir de la literatura existente sobre el UTRP [8].

R1	0	1	4	7
R2	0	3	6	$*$
R3	1	2	5	$*$

Fig. 2. Representación para las 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.

Las soluciones para el algoritmo inmune artificial propuesto están dadas por un conjunto de rutas que siguen la estructura de la Figura 2.

5 Estructura del algoritmo

El Algoritmo 1 muestra a grandes rasgos el pseudocódigo del algorimto artificial inmune implementado. Un anticuerpo se entiende dentro del contexto de los algoritmos inmunes artificiales como una solución candidata.

Algoritmo 1 Algoritmo Inmune Artificial

```
1: Inicializar población de anticuerpos aleatoriamente con sets de rutas factibles
2: g = 0
3: mientras GENERACIONES> g hacer
4:    Eliminar de la población anticuerpos dominados en cada set
5:    Calcular afinidad de cada miembro de la población
6:    Clonar a de los mejores anticuerpos hsata llegar a un tamaño CLON_SIZE
7:    para cada Clon hacer
8:       Mutar Clon
9:    fin para
10:   Guardar b mejores anticuerpos
11:   Eliminar exceso de anticuerpos
12:   Reemplazar peores anticuerpos con anticuerpos generados aleatoriamente
13:   g = g + 1
14: fin mientras
```

En el pseudocódigo se pueden disinguir ciertas etapas que serán detalladas a continuación.

6 Inicialización

La inicialización del conjunto de soluciones inicial es determinado de manera aleatoria. De tal forma de tener contar con distintos lugares del espacio de búsqueda. Cada solución del conjunto contiene un conjunto de rutas, escogido de la siguiente manera: aleatoriamente se escoge un largo de ruta, el cual está comprendido entre el mínimo y máximo establecido. Se escoge aleatoriemente un punto de partida y se genera un camino mediante las conexiones con paradas vecinas hasta completar el largo. Se realiza este procedimiento hasta tener un conjunto de rutas que posea todos los paraderos disponibles o hasta que se agote un número de intentos, lo que puede ocurrir cuando el número de paraderos es muy grande en comparación con la cantidad de paraderos que serán destinados a una ruta o si el número de rutas a generar es muy pequeño. Las instancias con las que se trabajará siempre tratarán de incluír a todos los paraderos.

Se genera una población de soluciones de manera de tener varias soluciones candidatas. Este conjunto de soluciones inciales es un conjunto de soluciones factible, lo que quiere decir que las rutas generadas son conexas y además no presentan ciclos ni backtrackings. El algoritmo trabaja con soluciones factibles y al generar nuevas soluciones se comprueban las condiciones necesarias para que también sean factibles.

7 Proceso de transformación

7.1 Selección de soluciones

El primer paso en el proceso de transformación del algoritmo consiste en realizar una selección clonal. Esto es, seleccionar a los mejores anticuerpos en base a una función de aptitud (3).

$$f_{aptitud} = \alpha \cdot FO_1 + \beta \cdot FO_2 \tag{3}$$

Donde FO_1 y FO_2 corresponden a las funciones objetivo dadas por las ecuaciones (1) y (2), respectivamente. Un porcentaje de estas soluciones son escogidas de acuerdo al resultado de $f_{aptitud}$ para pasar a la siguiente etapa del algoritmo.

7.2 Operadores de transformación

Los mejores anticuerpos son clonados hasta llegar a una cierta población. Esto consiste básicamente en generar muchas copias de los mejores anticuerpos. El operador de selección toma un anticuerpo y genera uno igual. La manera de decidir cual de los mejores anticuerpos se clona se realiza de manera aleatoria. De esta forma cada uno de los anticuerpos tiene la misma probabilidad de ser clonado.

Posteriormente, se utiliza un operador de mutación, el cual consiste en introducir pequeños cambios al conjunto de rutas para producir cambios en sus funciones objetivo, y por ende en la función de aptitud. La mutación consiste en seleccionar aleatoriamente una ruta de una solucion e introducir un nuevo vertice en ella, siempre y cuando no exceda el límite superior.

7.3 Selección de soluciones para conformar nueva población

Luego, se aplica una reducción de la población de los clones, dejando los más aptos (con una mejor función de aptitud). Se procede a reemplazar a los peores anticuerpos con anticuerpos generados aleatoriamente, siguiendo las mismas restricciones de la generación inicial al generar soluciones candidatas factibles.

8 Parámetros del algoritmo

El algoritmo considera los siguientes parámetros:

- `POP_SIZE`: Representa el tamaño de la población en cada generación. Una generación está compuesta por un conjunto de soluciones candidatas. Posee valores de 0 a infinito.
- `ALPHA` y `BETA`: representan variables de peso para ponderar las dos funciones objetivo y determinar una aptitud para la solución candidata. Posee valores entre cero y uno. La suma de ambas debe ser uno.
- `CLON_SIZE`: cantidad de clones que se generarán a partir de las mejores soluciones candidatas. Posee valores de 0 a infinito.
- `AFINIDAD`: Porcentaje de mejores soluciones seleccionadas para mutarse. Posee valores entre 0 y 1.
- `GENERACIONES`: Cantidad de generaciones que se producirán hasta la finalización del algoritmo. Posee valores de 0 a infinito.
- `CLONES`: porcentaje de clones que sobrevivirá después de realizar el operador de mutación. Posee valores entre 0 y 1.
- `REEMPLAZO`: porcentaje de anticuerpos que serán reemplazados por nuevos anticuerpos aleatorios. Posee valores entre 0 y 1.

9 Sintonización de Parámetros

Para la sintonización de parámetros se escogieron 4 de los parámetros y el resto quedó con un valor fijo. Entre los parámetros con valor fijo se utilizaron los valores de la Tabla 9. La instancia utilizada fue la de `Mandl 6:2:8`, cuyo grafo está representado en la Figura 1. Se buscan 6 rutas con longitudes que varían en el rango de 2 a 8 paradas.

`ALPHA`	1.0
`BETA`	1.0
`AFINIDAD`	1
`REEMPLAZO`	0.01

Tab. 1. Parámetros seteados en un valor fijo para Mandl.

Los parámetros `POP_SIZE`, `CLON_SIZE`, `GENERACIONES` y `CLONES` fueron puestos a prueba al utilizar ParamILS, un sintonizador de parámetros que utiliza un intérprete del código y ejecuta el algoritmo utilizando un rango de valores para los 4 parámetros escogidos.

En la Figura 3 `ps`, `cs`, `it` y `pc` corresponden a `POP_SIZE`, `CLON_SIZE`, `GENERACIONES` y `CLONES` respectivamente.

```
ps {20, 40, 100, 160, 200}[100]
cs {5, 15, 25}[15]
it {10, 30, 50, 70, 100}[30]
pc {10, 30, 50}[10]
```

Fig. 3. Valores utilizados en los parámetros sintonizados en Mandl.

Al ejecutar ParamILS se obtuvo la siguiente salida:

```
Final best parameter configuration: cs=5, it=100, pc=10, ps=200
=====================================================================
Active parameters: cs=5, it=100, pc=10, ps=200
=====================================================================
Training quality of this final best found parameter configuration:
-704.619631901841, based on 163 runs with cutoff 1000000000.0
Test quality of this final best found parameter configuration:
-703.96, based on 50 independent runs with cutoff 1000000000.0
```

Para este valores de parámetros se obtuvo un valor máximo de hipervolumen igual a 703.96.

10 Experimentos

El experimento con los valores de parámetros fijados según la Tabla 9 y los obtenidos mediante ParamILS, según lo mostroado a continuación:

POP SIZE	200
ALPHA	1.0
BETA	1.0
CLON SIZE	5
AFINIDAD	1
GENERACIONES	100
CLONES	0.1
REEMPLAZO	0.01

Tab. 2. Valores utilizados para el experimento con la instancia de Mandl.

Notar que el porcentaje de clones fue manejado en ParamILS con valores entre 0 y 100 y pasado posteriormente a valores entre 0 y 1.

Además, los mejores valores para las funciones objetivo corresponden a:

- Mejor FO1: 11.5087
- Mejor FO2: 76

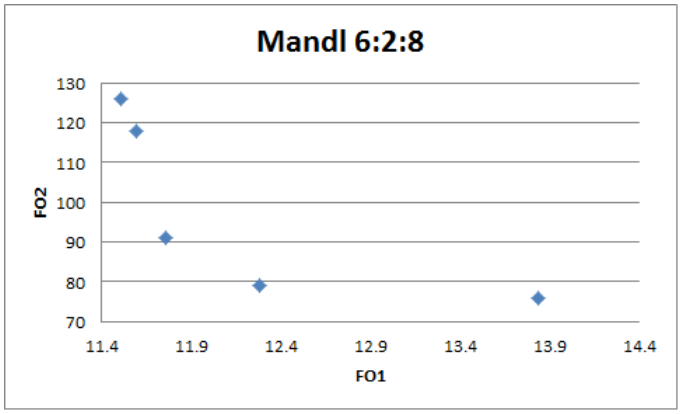

Fig. 4. Frente de Pareto de la instancia de Mandl.

Hipervolumen	698.452
Tiempo de ejecución	23 segundos

Tab. 3. Hipervolumen y tiempo de ejecución.

El conjunto de rutas que presenta el menor valor para la función de aptitud se muestra a continuación:

```
6-3
9-15-6-8
3-2-4-12
8-6-3
8-15-7-10-14-13-11
1-2-5
```

Se puede concluir que estos resultados no reflejan exactamente a lo obtenido mediante la sintonización de parámetros porque esto corresponde solo a una ejecución del algoritmo. Por otra parte, ParamILS realiza una cantidad considerable de experimentos probando con distintos valores y logra encontrar el mejor valor de hipervolumen. En este caso el valor es cercano al mejor valor obtenido, sin embargo si se realizara otro experimento podría originar un valor más cercano o más lejano.

11 Conclusiones

Los algoritmos inmunes artificiales utilizan analogías de los organismos vivos. Cuando un virus ataca al organismo los anticuerpos son capaces de interactuar con este y los más capaces son duplicados para atacar y eliminar a una enfermedad que cause. Luego se guarda en memoria una cantidad menor de este

anticuerpo para que esté preparado a nuevas infecciones del mismo virus o uno similar. Esta idea es la que permite encontrar soluciones mediante la clonación y mutación de los mejores individuos. El proceso de búsqueda de soluciones es un proceso adaptativo, lo que quiere decir que los anticuerpos se capacitan para enfrentarse a infecciones futuras.

Mediante esta implementación, la cual está compuesta por variables, restricciones, funciones objetivo, función de aptitud, operadores de selección y transformación para una población de anticuerpos se pudo encontrar un conjunto de soluciones no dominadas según Pareto, obteniendo mejores soluciones en tiempos relativamente cortos.

El proceso de sintonización de parámetros fue útil para poder encontrar valores para parámetros y obtener un mejor resultado, el cual está reflejado en un mayor valor de hipervolumen.

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. 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.
7. Talbi Josefowiez, Semet. An evolutionary algorithm for the vehicle routing problem with route balancing. *European Journal of Operational Research*, 195:761–769, 2009.
8. Christine L. Mumford Lang Fan. A metaheuristic approach to the urban transit routing problem. *Journal of Heuristics*, 1(16):353–372, 2010.
9. Joanne Suk Chun Chew Lai Soon Lee. A genetic algorithm to the urban transit routing problem. *International Journal of Modern Physics*, 9, 2012.
10. 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.

11. 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.
12. Nareli Cruz Cortés y Carlos A. Coello Coello. Un sistema inmune artificial para solucionar problemas de optimización multiobjetivo. Congreso Mexicano de Computación Evolutiva, 2003.
13. 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.
14. 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.