Un enfoque al problema dinámico de múltiples agentes viajeros con programación heurística

Cornejo Acosta, José Alejandro *, Puga Soberanes, Héctor José; Carpio Valadez, Juan Martín
Ornelas Rodríguez, Manuel; Mancilla Espinoza, Luis Ernesto

División de Estudios de Posgrado e Investigación (DEPI)
Instituto Tecnológico de León
alex.cornejo@itleon.edu.mx

Recibido: *28 de octubre 2017*
Aceptado: *6 de febrero de 2018*

RESUMEN

El problema de múltiples agentes viajeros (mTSP) y el problema dinámico del agente viajero (DTSP) han sido abordados en el estado del arte de manera independiente. En este trabajo se hace la propuesta de abordar simultáneamente estos dos enfoques como un problema dinámico de múltiples agentes viajeros al que identificamos como DmTSP. En la propuesta se genera una solución inicial para mTSP utilizando el algoritmo del vecino más cercano y ésta es mejorada con búsqueda local iterada. Posteriormente, la actualización dinámica de las rutas se hace utilizando un proceso de optimización inter-route. Para simular un entorno dinámico, se hace uso de una matriz de tráfico que cambia aleatoriamente. Los resultados muestran la factibilidad del modelo propuesto.

Palabras claves: TSP, DTSP, mTSP, DmTSP, optimización

ABSTRACT

Multiple Traveling Salesmen Problem (mTSP) and Dynamic Traveling Salesman Problem (DTSP) have been discussed in the literature independently. This paper focuses in a simultaneous approach of DTSP and mTSP as a Dynamic Multiple Traveling Salesmen Problem (DmTSP). In this proposal, initially a solution for mTSP is generated using the Nearest Neighbor algorithm and this solution is improved using Iterated Local Search algorithm. After, the dynamic update of the tours is executed by an inter-route optimization process. In order to simulate a dynamic environment, we used a traffic factor matrix which is randomly changing. The results show the feasibility of the proposed model.

Keyword: TSP, DTSP, mTSP, DmTSP, optimization

1 INTRODUCCIÓN

El problema del agente viajero, conocido como TSP (Traveling Salesman Problem), ha sido estudiado por muchos investigadores en el área de optimización con enfoques de programación matemática y de programación heurística (Applegate et al., 2007)(Cook, 2012)(Johnson and McGeoch, 2007). Variantes de este problema también han sido discutidas en el estado del arte, dos de ellas son el problema de múltiples agentes viajeros conocido como mTSP (multiple Traveling Salesmen Problem) y el problema dinámico del agente viajero conocido como DTSP (Dynamic Traveling Salesman Problem). En el caso del DTSP se han encontrado artículos donde en su mayoría se utilizan algoritmos inspirados en hormigas para tratar de resolverlo (Wang et al., 2016) (Mavrovouniotis et al., 2017) (Soleimanian Gharehchopogh et al., 2012) (Li et al., 2006) (Eyckelhof and Snoek, 2002) (Dorigo and Stützle, 2004). Por otro lado, la investigación del mTSP es más sólida y hay varios enfoques y algoritmos que han sido utilizados para resolverlo, en el enfoque de programación heurística, los algoritmos genéticos han demostrado brindar buenas soluciones (Bolaños et al., 2016) (Zhou et al., 2018). En (Bektas, 2006) (Toro et al., 2014) se presenta una revisión bibliográfica sobre varias técnicas que se han utilizado para resolver mTSP, tales como heurísticas constructivas y de mejoramiento, recocido simulado, búsqueda tabú e incluso redes neuronales. Adicionalmente, para mTSP existen dos enfoques, la primera consiste en trabajar con un único depósito para los *m* agentes como es el caso de (Bolaños et al., 2016), y también existe el enfoque de mTSP con múltiples depósitos, el cual se trabaja en (Zhou et al., 2018). A pesar de que mTSP y DTSP han sido estudiados, no se encontraron investigaciones sobre algún enfoque o modelo que toque simultáneamente estas dos variantes de TSP. El propósito de este artículo es presentar un enfoque simultáneo sobre mTSP y DTSP, así como presentar un contexto general de estos problemas y como se podrían combinar en un mismo modelo. El modelo de DmTSP puede ser utilizado en entornos dinámicos como por ejemplo, en un sistema de distribución donde múltiples vehículos que reparten un mismo producto, tienen que visitar un conjunto de destinos específicos donde el costo de tiempo entre viajar de un lugar a otro cambia en diferentes horas del día.

Este artículo tiene la siguiente estructura, en la sección 2 se presenta un marco de referencia sobre mTSP y DTSP, también se mencionan los algoritmos heurísticos utilizados en este trabajo de investigación. La sección 3 presenta un modelo general sobre DmTSP y la metodología utilizada en el presente trabajo. La sección 4 presenta el diseño de experimentos, los resultados y el análisis de los mismos. Finalmente, la sección 5 presenta las conclusiones y opciones de trabajos futuros.

2 MARCO DE REFERENCIA

2.1 Múltiples TSP

En el problema de múltiples agentes viajeros (mTSP), es permitido que múltiples agentes viajeros trabajen en conjunto para resolver una instancia de TSP. Algunos investigadores han utilizado metaheurísticas como búsqueda tabú, recocido simulado y algoritmos genéticos para resolver este problema (Bektas, 2006). En (Bolaños et al., 2016) los autores proponen un algoritmo genético modificado llamado MCBGA (Modified Chu-Beasley Genetic Algorithm) para resolver mTSP.

El modelo de mTSP (Bolaños et al., 2016) (Bektas, 2006) considera un grafo completamente conectado $G (V, E)$ donde E es el conjunto de arcos y $V = \{1 \ldots n\}$ es el conjunto de vértices correspondientes a los nodos o ciudades. En este caso, el vértice 1 corresponde a la ciudad o nodo depósito. D es una matriz formada por términos de costo

d_{ij} asociados con cada arco $(i,j) \in E$. Los costos pueden representar distancia, tiempo o alguna otra medida. x_{ij} representa una variable binaria que puede tomar el valor *1* si el arco *(i, j)* forma parte de la solución o *0* en cualquier otro caso.
Se define:

$$x_{ij} = \begin{cases} 1, si\ se\ llega\ a\ la\ ciudad\ j\ desde\ la\ ciudad\ i \\ 0, de\ lo\ contrario \end{cases} \quad (1)$$

En resumen, el planteamiento general de mTSP consisten en, dado un conjunto de ciudades con *m* agentes viajeros, el objetivo es encontrar una ruta para cada agente viajero tal que:

- La suma total de los costos de las rutas de los agentes viajeros sea minimizada, como se muestra en la función objetivo de la ecuación (2).

$$z = \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij}\, x_{ij} \quad (2)$$

donde

$$d_{ij} = \infty\ para\ todas\ las\ i = j$$

- Todas las rutas tienen que iniciar y terminar en un mismo depósito.
- Cada ciudad tiene que ser visitada estrictamente una única vez por un solo agente, a excepción del depósito.

Sujeto a

$$\sum_{j=2}^{n} x_{1j} = m, \quad (3)$$

$$\sum_{i=2}^{n} x_{i1} = m, \quad (4)$$

$$\sum_{i=2}^{n} x_{ij} = 1, \qquad j = 2, \dots, n \quad (5)$$

$$\sum_{j=2}^{n} x_{ij} = 1, \qquad i = 2, \dots, n \quad (6)$$

Las restricciones (3) y (4) aseguran que exactamente *m* agentes inician su recorrido en el nodo 1 (depósito) y regresan a este mismo nodo. Las restricciones (5) y (6) garantizan que cada nodo quede conectado únicamente con otro nodo a excepción del nodo 1 que es el depósito.

2.2 TSP Dinámico

El TSP Dinámico (DTSP) fue introducido en 1998 por Psaraftis (Soleimanian Gharehchopogh et al., 2012), DTSP es un TSP en el cual ciudades pueden ser agregadas o eliminadas en tiempo real (Dorigo and Stützle, 2004), o cuando el costo de viajar entre ciudades puede cambiar (Wang et al., 2016). Algunos investigadores han trabajado

el DTSP con optimización con colonia de hormigas (ACO, Ant Colony Optimization) (Wang et al., 2016) (Soleimanian Gharehchopogh et al., 2012). El DTSP tratado desde un enfoque de optimización de colonia de hormigas tiene problemas con la evaporación de feromonas, incluso en (Wang et al., 2016) el autor propone una variante del algoritmo ACO con una función de evaporación de feromonas diferente a la tradicional para poder adaptar este algoritmo a DTSP.

Un TSP Dinámico es un TSP determinado por una matriz de costos dinámica (Li et al., 2006), como se muestra a continuación:

$$D(t) = d_{ij}(t)_{n\times n} \tag{7}$$

donde

$d_{ij} = Es\ el\ costo\ de\ viajar\ de\ la\ ciudad\ c_i\ a\ la\ ciudad\ c_j$

$t = Es\ un\ tiempo\ determinado$

2.3 Heurísticas de bajo nivel

Existen heurísticas de bajo nivel específicamente aplicadas a TSP y mTSP, las heurísticas de bajo nivel pueden utilizarse para perturbar o mutar una solución y se pueden clasificar de la siguiente manera:

2.3.1 Heurísticas intra-route

Son las heurísticas que se pueden aplicar dentro de una única ruta. Por ejemplo, permutar dos ciudades que están en una ruta.

Heurística de inversión para TSP

La heurística de inversión modifica la ruta de un agente viajero invirtiendo el orden de los nodos de un subrecorrido, se toman dos posiciones aleatorias *i* y *j* tal que *i*<*j*. El orden de esta secuencia es invertida para así obtener una nueva ruta (Taha, 2012).

2.3.2 Heurísticas inter-route

Son las heurísticas que se aplican entre dos rutas, por ejemplo permutar dos ciudades que están en dos rutas diferentes, como quitar una ciudad de una ruta y ponerla en otra, etc. En mTSP se utilizan heurísticas inter-route para mejorar una solución, a continuación se describen algunas heurísticas inter-route(Bolaños et al., 2016) (Toro et al., 2014).

Heurísticas de perturbación inter-route para mTSP

Shift(1, 0): Una ciudad p_k es transferida de una ruta *i* a una ruta *j*.

Shift(2, 0): Dos ciudades adyacentes p_k y p_{k+1} son transferidas de una ruta *i* a una ruta *j*.

Swap(1, 1): Dos ciudades son permutadas, la ciudad p_k de la ruta *i* es permutada con la ciudad p_l de la ruta *j*.

Swap(2, 1): Dos ciudades adyacentes p_k y p_{k+1} de la ruta *i* son permutadas con una ciudad p_l de la ruta *j*.

Swap(2, 2): Dos ciudades adyacentes p_k y p_{k+1} de la ruta *i* son permutadas con dos ciudades adyacentes p_l y p_{l+1} de la ruta *j*.

2.4 Búsqueda local *k-opt*

La heurística *k-opt* funciona de la siguiente manera, si tenemos *k=2*, entonces obtenemos dos arcos no adyacentes de una ruta, después los nodos de los arcos seleccionados se permutan con el objetivo de encontrar nuevos arcos y una nueva ruta con menor costo (Blazinskas and Misevicius, 2011). El proceso se repite hasta que no se encuentren mejoras. En el caso de tener un número *k* diferente, por ejemplo *k=3*, el proceso es el mismo, solo que ahora se toman tres arcos en lugar de dos.

2.5 Algoritmo de búsqueda local iterada

Es una mejora del algoritmo clásico de búsqueda local (Lourenço et al., 2010), es un algoritmo trayectorial que tiene como principio ejecutar varias búsquedas locales para tratar de encontrar un óptimo global dentro del espacio de búsqueda, este algoritmo trata de resolver el problema de la búsqueda local el cual consiste es quedarse estancado en un óptimo local. El algoritmo 1 muestra el pseudocódigo de una búsqueda local iterada.

Algoritmo 1 Búsqueda local iterada

```
1:  s0 = GenerarSolucionInicial
2:  s* = BusquedaLocal (s0)
3:  while no se cumple la condición de paro do
4:      s' = Perturbacion (s*, history)
5:      s*'= BusquedaLocal (s')
6:      s* = CriterioAceptacion (s*, s*', history)
7:  end while
```

2.6 Heurística del vecino más cercano

Como su nombre lo sugiere, una solución TSP puede hallarse comenzando con una ciudad aleatoria y luego conectándola con la ciudad no conectada más cercana. La ciudad que se acaba de agregar se conecta entonces con su ciudad no conectada más cercana. El proceso continúa hasta que se forma un recorrido (Taha, 2012).

3 METODOLOGÍA

La metodología propuesta para resolver un problema DmTSP consta esencialmente de 6 pasos que son mostrados en la figura (1). Pero antes, es necesario conocer las restricciones bajo las cuales esta propuesta está planteada.

3.1 Restricciones

El modelo presentado aquí asume las siguientes restricciones:

- Existe un único depósito.
- Hay *m* agentes viajeros y *m* es un valor conocido.
- Los agentes viajeros tienen aproximadamente la misma cantidad de ciudades asignadas a visitar.

3.2 Pasos de DmTSP

Para crear una solución de DmTSP se tiene que seguir una serie de pasos partiendo primero desde una solución tradicional de TSP. Los pasos de la figura (1) se describen a continuación.

1. **Crear una única ruta con todos los nodos de una instancia de TSP**

   Primero creamos una única ruta original con todas las ciudades de una instancia de TSP utilizando el algoritmo del vecino más cercano y utilizando un nodo aleatorio como depósito, hasta este punto tenemos una solución de TSP tradicional.

2. **Optimizar ruta con ILS y heurísticas intra-route**

   Optimizamos la ruta con el algoritmo de búsqueda local iterada (ILS) utilizando la heurística *inversión* como heurística de perturbación y utilizando la búsqueda local *2-opt* para intensificar la solución de TSP tradicional. El algoritmo se ejecuta con 100,000 llamadas a función.

3. **Dividir la ruta en *m* subrutas para obtener una solución de mTSP**

   Este paso consiste en dividir la solución de TSP tradicional en *m* subrutas (una subruta para agente viajero) con el objetivo de obtener una solución de mTSP. En este paso, se deben satisfacer las restricciones de c_{min} y c_{max}, ver expresión (8). Aquí se utiliza parte de la metodología propuesta en (Bolaños et al., 2016) la cual consiste en lo siguiente:

   El número de ciudades $|TSP_i|$ de un i-ésimo agente viajero tiene un límite c_{min}, que representa la cantidad mínima de ciudades que cada agente tiene que visitar y un límite c_{max} el cual representa la cantidad máxima de ciudades que cada agente tiene que visitar:

   $$c_{min} \leq |TSP_i| \leq c_{max} \tag{8}$$

   El valor de c_{max} es un parámetro de entrada del problema y su valor es aproximadamente $c_{max} \approx \frac{n}{m}$. c_{min} es calculado con la ecuación (9).

   $$c_{min} = \left\lfloor \frac{n}{TSP_{min} + 1} \right\rfloor \tag{9}$$

donde *n* es la cantidad de ciudades y TSP_{min} representa el número mínimo de agentes viajeros necesarios para cubrir todas las ciudades, este último se calcula con la ecuación (10).

$$TSP_{min} = \left\lceil \frac{n}{c_{max}} \right\rceil \quad (10)$$

Las *m* rutas asociadas a los *m* agentes son asignadas secuencialmente. Para asignar una ruta a un i-ésimo agente, se toman los primeros $|TSP_i|$ nodos de la ruta original. El proceso se repite secuencialmente para asignar las demás rutas a los demás agentes. Al final de este proceso se tiene una solución de mTSP.

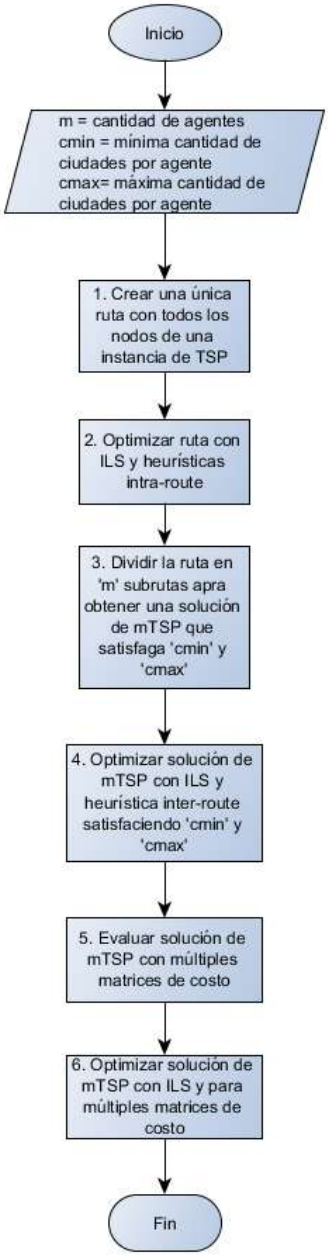

Figura 1: Pasos de DmTSP

4. **Optimizar la solución de mTSP con ILS heurísticas inter-route**

En este paso también se utiliza el algoritmo búsqueda local iterada (ILS), que es ejecutado con 100,000 llamadas a función y se utilizan las heurísticas inter-route presentadas en 2.3.2 como heurísticas de perturbación. ILS ejecuta la búsqueda local *2-opt* para cada ruta que forma parte de la solución de mTSP. De esta manera obtenemos una solución de mTSP más optimizada en comparación de la solución que se tenía en el paso anterior.

5. **Evaluar solución de mTSP con múltiples matrices de costo**

En todos los pasos anteriores, la optimización de rutas se hizo con base en una sola matriz de costos. En este paso, la solución de mTSP obtenida en el paso anterior es evaluado con múltiples matrices de costo.

6. **Optimizar solución de mTSP con ILS para múltiples matrices de costo**

Al igual que el paso 4, también se utiliza el algoritmo de búsqueda local iterada para para optimizar la solución de mTSP, la diferencia es que ahora se toman en cuenta múltiples matrices de costo para tomar en cuenta los cambios que hay en el entorno. El proceso se realiza con 100,000 llamadas a función por cada agente viajero que esté participando en el problema. Por ejemplo si tenemos 5 agentes viajeros $m = 5$, entones este paso se ejecuta con 500,000 llamadas a función.

3.3 Generación de las múltiples matrices de costo

Para simular el entorno dinámico, se hace uso de una matriz de costos *C*. Para construir *C* se implementa parte de la metodología presentada en (Wang et al., 2016), se considera una matriz de distancias *D* y una matriz de "factores de tráfico" *T* que es generada aleatoriamente, se calcula la matriz de costos *C* como:

$$C = D \times T \tag{11}$$

donde

$$c_{ij} = d_{ij} \times t_{ij}$$

y

d_{ij} = Es la distancia entre las ciudades *i* y *j*

t_{ij} = Es un factor de tráfico o escalamiento entre las ciudades *i* y *j*

c_{ij} = Es el costo de viajar de la ciudad *i* a la ciudad *j*

La matriz *T* cambia aleatoriamente de acuerdo a un umbral de probabilidad *p*, el valor de t_{ij} está limitado por $[T_L, T_H]$ donde T_L denota el menor valor del factor de escala de tráfico y T_H denota el mayor valor del factor de escala de tráfico.

En caso de que $p=1$, significa que *T* siempre cambiará. En caso de que $p=0$ significa que la matriz de costos nunca cambiará, lo que equivaldría a un mTSP.

4 RESULTADOS

A fin de mostrar la factibilidad del enfoque propuesto para DmTSP, se utilizaron seis instancias estándar del estado del arte obtenidas del sitio web de TSPLIB, http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/ , se utilizaron las mismas instancias que (Bolaños et al., 2016), estas instancias son simétricas y son pr76, pr152, pr226, pr299, pr439 y pr1002. La interpretación de cada instancia es pr*n*, donde *n* es la cantidad de nodos de la instancia.

En la experimentación, los límites de escalamiento de tráfico fueron $T_L=1$ y $T_H=4$. El umbral de probabilidad definido fue $p=1$.

Los valores de los parámetros de *m* y c_{max} fueron tomados como referencia de (Bolaños et al., 2016).

La ejecución de DmTSP por instancia se realizó 10 veces. En la tabla 1 se muestran los promedios de las 10 ejecuciones.

Tabla 1: **Resultados de DmTSP**

Instancia	m	c_{max}	Costo final sin actualizar las rutas	Costo final actualizando las rutas	% de reducción de costo
pr76	4	20	350,800.83	282,811.80	-18.96%
pr152	4	40	294,196.86	242,247.78	-17.68%
pr226	5	50	365,954.92	299,899.01	-17.92%
pr299	5	70	201,495.54	178,323.32	-11.38%
pr439	5	100	388,420.32	357,914.16	-7.85%
pr1002	5	220	294,196.86	242,247.78	-3.52%

La columna “Costo final sin actualizar las rutas” es el costo final que acumularían todos los agentes manteniendo las rutas iniciales durante todo el recorrido considerando los cambios de los costos entre las ciudades. La columna “Costo final actualizando las rutas” es el costo final que acumularían todos los agentes si actualizaran sus rutas respecto a los cambios que surgen en los costos entre las ciudades durante todo el recorrido. La columna “% de reducción de costo” es el porcentaje de reducción del costo que se logra cuando las rutas de los agentes son actualizadas. En esta experimentación el signo negativo significa que para todos los casos el costo final se redujo cuando las rutas de los agentes fueron actualizadas. Para todos los casos, cuando las rutas fueron actualizadas se puede observar que el costo disminuye. En los resultados también se puede observar que mientras más grande sea la instancia, el porcentaje de reducción de costo disminuye, esto puede deberse a que el espacio de búsqueda es más grande y por lo tanto es más difícil encontrar buenos movimientos para minimizar los costos de las rutas.

5 CONCLUSIONES Y TRABAJO A FUTURO.

Los resultados muestran la factibilidad de utilizar el modelo propuesto DmTSP para el problema dinámico de múltiples agentes viajeros. En el modelo propuesto se puede visualizar la importancia de actualizar las rutas cuando la matriz de costos cambia.

En los resultados obtenidos, se puede apreciar que aparentemente en las instancias con mayor cantidad de nodos, el porcentaje de reducción de costo disminuye.

Un trabajo a futuro es analizar la forma en que se relacionan el porcentaje de reducción de costos respecto a la cantidad de nodos y el número de agentes en las diferentes instancias de prueba.

Es importante mencionar que en los procesos del modelo DmTSP se utilizaron los algoritmos de búsqueda local iterada y el vecino más cercano, la metodología presentada no está limitada a éstos, ésta puede ser combinada con otros algoritmos que realicen la misma función para hacer más eficiente el modelo.

6 AGRADECIMIENTOS

El autor agradece al Consejo Nacional de Ciencia y Tecnología (CONACyT) por el apoyo económico brindado a través de becas de posgrado CVU 775090. También al Tecnológico Nacional de México, en particular al Departamento de Posgrado e Investigación (DEPI) del Instituto Tecnológico de León donde se desarrolló este trabajo.

7 REFERENCIAS BIBLIOGRÁFICAS

APPLEGATE, D. L., Bixby, R. E., Chvatal, V., and Cook, W. J. (2007). *The Traveling Salesman Problem: A Computational Study (Princeton Series in Applied Mathematics)*. Princeton University Press, Princeton, NJ, USA.

BEKTAS, T. (2006). The multiple traveling salesman problem: an overview of formulations and solution procedures. *Omega*, 34(3):209 – 219.

BLAZINSKAS and Misevicius, A. (2011). Combining 2-opt, 3-opt and 4-opt with k-swap-kick perturbations for the traveling salesman problem. En *Proceedings of the 17th international conference on Information and Software Technologies*.

BOLAÑOS et al., 2016. Bolaños, R., Toro, E., and Echeverri, M. (2016). A populationbased algorithm for the multi travelling salesman problem. *International Journal of Industrial Engineering Computations*, 7:245–256.

COOK, W. J. (2012). *In Pursuit of the Traveling Salesman: Mathematics at the Limits of Computation*. Princeton University Press.

DORIGO and Stützle, 2004. Dorigo, M. and Stützle, T. (2004). *Ant Colony Optimization*. Bradford Company, Scituate, MA, USA.

EYCKELHOF. Eyckelhof, C. J. and Snoek, M. (2002). Ant systems for a dynamic tsp. In Dorigo, M., Di Caro, G., and Sampels, M., editors, *Ant Algorithms*, páginas 88–99, Berlin, Heidelberg. Springer Berlin Heidelberg.

JOHNSON AND. and McGeoch, L. A. (2007). *Experimental Analysis of Heuristics for the STSP*, páginas 369–443. Springer US, Boston, MA.

LI, C., Yang, M., and Kang, L. (2006). A new approach to solving dynamic traveling salesman problems. In Wang, T.-D., Li, X., Chen, S.-H., Wang, X., Abbass, H., Iba, H., Chen, G.-L., and Yao, X., editors, *Simulated Evolution and Learning*, pages 236–243, Berlin, Heidelberg. Springer Berlin Heidelberg.

LOURENÇO, H. R., Martin, O. C., and Stützle, T. (2010). *Iterated Local Search: Framework and Applications*, pages 363–397. Springer US, Boston, MA.

MAVROVOUNIOTIS, M., Müller, F. M., and Yang, S. (2017). Ant colony optimization with local search for dynamic traveling salesman problems. *IEEE Transactions on Cybernetics*, 47(7):1743–1756.

SOLEIMANIAN GHAREHCHOPOGH, F., Maleki, I., and Farahmandian, M. (2012). New approach for solving dynamic traveling salesman problem with hybrid genetic algorithms and ant colony optimization. 53:39–44.

TAHA, H. A. (2012). *Investigación de operaciones*. Pearson, México, 9 edition.

TORO, E., Bolaños, R., and Echeverri, M. (2014). Solución del problema de múltiples agentes viajeros resuelto mediante técnicas heurísticas. *Scientia et Technica*, 19(2):174–182.

WANG, Y., Xu, Z., Sun, J., Han, F., Todo, Y., and Gao, S. (2016). Ant colony optimization with neighborhood search for dynamic tsp. In Tan, Y., Shi, Y., and Niu, B., editors, *Advances in Swarm Intelligence*, pages 434–442, Cham. Springer International Publishing.

ZHOU, H., Song, M., and Pedrycz, W. (2018). A comparative study of improved ga and pso in solving multiple traveling salesmen problem. *Applied Soft Computing*, 64:564 – 580.