Inteligencia Artificial Avanzada

Estado del Arte: Vehicle Routing Problem with Time Window

Daniel Tapia R.

15 de abril de 2019

Evaluación

Resumen (5 %):	_____
Introducción (5 %):	_____
Definición del Problema (10 %):	_____
Estado del Arte (35 %):	_____
Modelo Matemático (20 %):	_____
Conclusiones (20 %):	_____
Bibliografía (5 %):	_____
Nota Final (100 %):	_____

Resumen

Vehicle Routing Problem se define como el problema de organizar rutas para una cierta cantidad de vehículos en un sector con n nodos a recorrer, estos nodos deben ser todos visitados, los camiones no deben sobrepasar su capacidad al momento de hacer las entregas. La extensión llamada Time Windows define que los nodos anteriormente mencionados deben ser recorridos en una cantidad acotada de tiempo, del cual no se puede adelantar ni atrasarse. El objetivo de este problema es poder obtener rutas que minimicen el tiempo utilizado en recorrer las rutas mencionadas en el menor tiempo posible o , de forma análoga, minimizar la distancia recorrida por todos los vehículos.

1. Introducción

En el presente documento se presentará el problema Vehicle Routing Problem with Time Windows, el cual por simpleza se llamará VRPTW. Vehicle Routing Problem es una clásica simulación de logística, donde se busca repartir la mercadería desde un deposito central que es un punto fijo del cual se toma la carga y al final de la ruta se tiene que volver a este [4]. La variación Time Windows agrega además la condición de que estos puntos a repartir deben ser visitados en un intervalo de tiempo acotado, es decir, se tiene un horario el cual la ruta debe respetar para entregar correctamente la mercadería [2].

El problema descrito anteriormente se puede considerar una extensión de uno de los problemas más conocidos en el área de optimización: el Travelling Salesman Problem, también conocido

por sus siglas TSP. La explicación más simple de su extensión es que VRP puede ser considerado un m-TSP [2], donde m es la cantidad de "personas" disponibles para recorrer el camino general. Por lo dicho anteriormente, se puede extrapolar que VRPTW es un problema NP-Hard [5].

La principal motivación de estudiar este problema es temas de logística. Al ser un problema muy cercano a la realidad, este problema y su solución pueden ayudar bastante a las empresas a administrar sus repartos de mercadería y, por ende, bajar sus costos de envío. Sin embargo, no todo puede ser tan simple como modelar y tratar de resolver lo planteado en este documento, aquí solo se presentará una forma de resolver uno de los tantos factores que implica la logística en la vida real.

2. Definición del Problema

Para entender VRPTW de una forma clara, se procede a explicar ciertas cosas que se deben tener en mente al momento de describir el objetivo de este problema.

2.1. Travelling Salesman Problem (TSP)

TSP consiste en un viajero que debe recorrer n ciudades para repartir su mercadería y, por ende, debe recorrer un tour en el menor tiempo posible, tomando en cuenta que debe volver al mismo punto del cual inició [2, 8].

Este problema matemático fue descrito por primera vez por Sir William Rowan Hamilton and Thomas Penyngton Kirkman alrededor del siglo XIX, este consistía en recorrer los nodos de manera que no volver a visitar más de una vez el mismo nodo, ese tipo de movimiento es también conocido como "Camino Hamiltoniano" [8, 5].

Un modelo matemático que resuelva el TSP clásico debe contemplar como mínimo:

- El vendedor debe volver al mismo punto del cual inicio
- Se debe recorrer todos los nodos de la instancia
- No se puede pasar por el mismo nodo más de una vez

Teniendo esto en cuenta, el objetivo de TSP es minimizar el camino recorrido por el viajante en su ruta. Otras variantes pueden existir dependiendo de que es lo que varíe en el problema. En [5] se presenta un cuadro que muestra los problemas asociados a TSP que se obtienen al variar los objetivos y restricciones.

Demanda	Restricciones de capacidad	Nombre habitual del problema	Otras restricciones
Nodos	NO	Problema del Vendedor Viajero	
	SI	Problema de rutas de vehículos	Recogida/distribución
Arcos	NO	Problema del Cartero Chino (Componente conexa)	Ventanas de tiempo
		Problema del Cartero Rural (Varias componentes conexas)	Otras
	SI	Problema de rutas con capacidades	

Cuadro 1: Clasificación problemas de ruta, este cuadro puede apreciarse con más detalles en [5]

2.2. Vehicle Routing Problem (VRP)

VRP es considerado una variación de TSP, ya que cuando se extrapola el problema para m personas que recorran un mismo camino esto convierte el problema en Multiple Travelling Salesman Problem (m-TSP) [2]. En si, estos dos problemas en su versión clásica comparten restricciones que se deben cumplir, tales como:

- Todos los viajeros parten del mismo nodo.
- Todos deben volver al mismo punto del cual iniciaron.
- Cada nodo solo puede ser recorrido una vez por solo un viajero.
- Se debe recorrer todos los nodos de la instancia.

Como se puede apreciar, las diferencias entre las restricciones de TSP con VRP son mínimas, en general, cambian en que ahora son muchas personas las que salen a repartir. El objetivo también se mantiene para ambos problemas, este es el minimizar camino recorrido por todos los viajantes. A veces el objetivo puede cambiar a recorrer las rutas en el menor tiempo, pero esto depende de como se modele el problema.

Para VRP existen variaciones que vale la pena destacar, de las cuales son:

- *Open VRP*: Esta variación no limita a los vehículos a devolverse al deposito al terminar la ruta.
- *Capacitated VRP*: Ahora los vehículos tendrán una capacidad para llevar los objetos, la cual no deben superar.
- *VRP with Pick-up and Deliveries*: Los vehículos transportan la mercadería a los usuarios y estos pueden enviar mercadería a los depósitos y, por lo tanto, deben devolverse al deposito a dejar lo entregado por los usuarios.
- *Green VRP*: Al tener vehículos que emanan Dióxido de Carbono (CO_2) se trata de minimizar esta contaminación haciendo que su ruta sea más corta para los más contaminantes.
- *VRP with Time Windows*: Los vehículos tienen horarios en los cuales pasar por los nodos, los cuales deben ser respetados.
- *VRP with Multiple trips*: Los vehículos pueden hacer más de una ruta.

2.3. Vehicle Routing Problem with Time Windows (VRPTW)

VRPTW es una de las tantas variaciones que tiene VRP, este en particular contiene variables de tiempo cosa que, según el escritor, es más relevante en la vida real dado a los horarios que poseen los clientes al momento de repartir [2, 1].

Como fue mencionado en el ítem anterior, VRPTW es una extensión de VRP entonces sus restricciones deberían ser las mismas con la pequeña diferencia que Time Windows implica. En sí, la única restricción que cambia es:

- Se debe recorrer todos los nodos de la instancia en un intervalo de tiempo establecido para cada nodo.

Por otro lado, hay algunas variables que son incluidas al momento de experimentar, de las cuales los experimentadores agregan dependiendo de como buscan abarcar el problema. Algunos ejemplos son:

- Los nodos poseen demandas, las cuales deben considerarse como prioridad si es necesario.
- Los vehículos gastan tiempo descargando la mercadería en los nodos.
- Los vehículos tienen capacidad.

Estos son algunos ejemplos que se toman en cuenta en las instancias propuestas por Solomon para VRPTW [3]. Sin embargo, este no contempla el hecho de que se tengan n vehículos, sino que dependiendo de la cantidad de rutas óptimas logradas se consiguen los vehículos. Lo propuesto por Solomon con respecto a los vehículos, según el autor, no está muy apegado a la realidad, sin embargo, se procederá a usar estas instancias.

3. Estado del Arte

Generalmente en estos tipos de problemas NP, trata de probarse algoritmos conocidos para resolverlos y, dependiendo de la calidad del resultado en comparación con lo óptimo conocido, mejorar el resultado obtenido. Esto puede darse aplicando algoritmos reparadores.

Aplicar metaheurísticas para resolver problemas con grandes cantidades de datos pueden resultar útiles al momento de querer resultados decentes con poco procesamiento de estos datos, es decir, que no tome tanto tiempo obtener un resultado con "pocas" instancias. Dentro de las más conocidas metaheurística tenemos: simulated annealing, tabu search y algoritmos genéticos. Estos algoritmos no son los únicos, pero si son los más frecuentes en este problema, por lo tanto, son lo que se describirán a continuación [2].

3.1. Simulated annealing

Este algoritmo funciona de forma similar a un proceso físico de recocer el acero y cerámicas, de hecho, de ahí su nombre. Este algoritmo se puede dividir en dos partes: baja temperatura, que representa una intensificación dentro del algoritmo; y alta temperatura, que simula una diversificación en el espacio de búsqueda. Este algoritmo, al igual que la mayoría de su clase, no asegura una solución óptima global, es decir, no asegurará la mejor solución del espacio de búsqueda para el problema en el cual se este trabajando.

3.2. Tabu Search

Este algoritmo busca mejorar una solución ya dada, osea, es un algoritmo reparador. Sin embargo, si se utiliza una solución aleatoria al problema, se puede aplicar esta técnica para encontrar una solución mejor que la anterior. Al igual que el algoritmo anterior, este no asegura encontrar el óptimo global. Esta técnica funciona de forma similar a Hill Climbing, la diferencia está en una cola de la cual se ingresan los movimientos recientes, de forma que no puedan ser seleccionados hasta un buen tiempo asegurando que no se produzcan ciclos. En cuanto a la diversificación e intensificación del algoritmo, depende exclusivamente de la cola/lista tabú, ya que, si es pequeña, favorece la búsqueda local, en otras palabras, intensifica; si es grande, este tiende a formar caminos, permitiendo una exploración en el espacio de búsqueda.

3.3. Algoritmos genéticos

Esta técnica trata de simular la teoría de la evolución de Darwin, es decir, se toman dos individuos y se cruzan para lograr nuevos individuos que pueden ser de mejor calidad o peor. Este algoritmo, al igual que los anteriores, tiene su fase de intensificación y diversificación. El proceso de intensificación ocurre cuando al momento del cruzamiento solo se eligen soluciones

que ya son de buena calidad, entonces se espera que los individuos resultantes sean de mejor calidad (pero no lo asegura); por otro lado, la diversificación ocurre de manera contraria, en este se hacen cruzamientos con cualquier solución que sea considerada en ese momento (factible o infactible). Esta técnica, al igual que las mencionadas anteriormente, no asegura una solución que sea el óptimo global.

Luego de explicar los algoritmos más comunes que han sido usados para resolver este problema, no estaría demás decir algunas técnicas que mejoren la solución encontrada por los algoritmos anteriores. Si bien tabu search se puede considerar un algoritmo reparador, el hecho que no asegure un óptimo global hace que su solución tampoco sea del todo confiable. Por lo tanto, mejorar la solución encontrada por tabu search sigue siendo una buena idea.

Algunas de las técnicas de reparación más usadas en este problema son[7, 2]:

- *2-Opt*: Cambias un segmento de la ruta por otro segmento de una ruta distinta a la elegida en un comienzo.
- *Or-Opt*: Un segmento continuo de nodos/clientes se mueve de una posición de una ruta a otra.
- *Exchange Operator*: Intercambia dos nodos/clientes entre dos rutas.
- *Relocate Operator*: Mueve uno de los nodos/clientes desde una ruta a otra.

Estas son las técnicas más conocidas y usadas. Sin embargo todavia se pueden apreciar más en [7, 2].

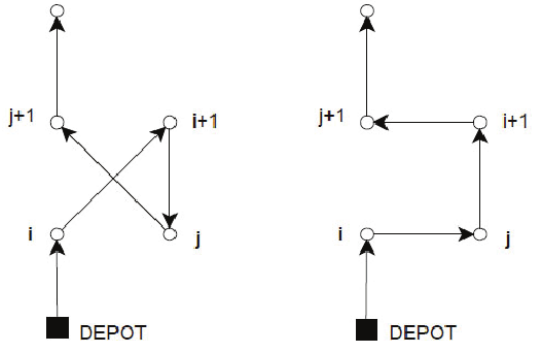

Figura 1: Ejemplo gráfico de como funciona 2-Opt

Figura 2: Ejemplo gráfico de como funciona Or-Opt

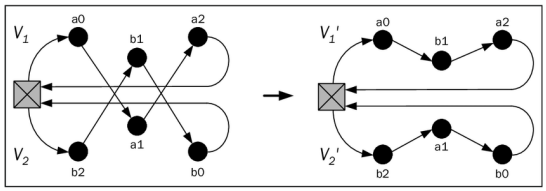

Figura 3: Ejemplo gráfico de como funciona Exchange Operator

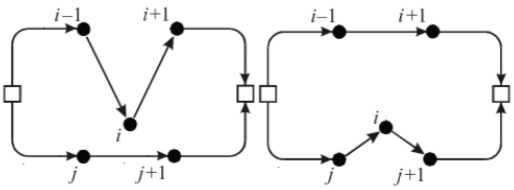

Figura 4: Ejemplo gráfico de como funciona Relocate Operator [7]

Para resolver este problema, suena mucho, recientemente, el uso de algoritmos genéticos. Al ser un algoritmo tan simple y poderoso, el uso de de esta técnica es bastante beneficioso para el estudio de este problema. Sin embargo, como fue anteriormente, al no asegurar una solución óptima global, no se puede considerar este algoritmo para obtener una solución totalmente buena como para establecer un óptimo recreable.

En [3] se puede apreciar los mejores resultados de VRPTW utilizando las instancias de Solomon. En ese articulo se puede ver que su función objetivo se enfocó en la distancia recorrida por el conjunto de vehículos K usados.

En cuanto a la obtención de los mejores resultados vistos en [3], estos fueron obtenidos en [6] utilizando un técnica llamada 2-Path Cut, obteniendo los mejores resultados para ciertas instancias propuestas por Solomon para este problema.

Para la explicación general del cómo se ha abordado este problema, es necesario explicar una instancia del problema y que se esperaría como salida de este:

3.3.1. Entrada

- *VEHICLE NUMBER*: Cantidad máxima de vehículos que se pueden utilizar.
- *CAPACITY*: Capacidad de los vehículos.
- *XCOORD*: Coordenada X de un nodo.
- *YCOORD*: Coordenada Y de un nodo.
- *DEMAND*: Demanda asociada del nodo.
- *READY TIME*: Tiempo de inicio al cual se le puede entregar al nodo.
- *DUE DATE*: Tiempo final al cual se le puede entregar al nodo.
- *SERVICE TIME*: Tiempo que tarda el vehículo en descargar la mercadería en el nodo.

Dependiendo de como se aborde el problema, se pueden ignorar algunos parámetros.

3.3.2. Salida

- K: Cantidad de vehículos que se necesitaron para la ruta mínima.
- *DIST*: Distancia que recorren los vehículos en total.
- *TIME*: Tiempo empleado en realizar la ruta. Como se realizan en paralelo, es elegir el vehículo que tarda más.

Las salidas pueden variar con respecto a la función objetivo elegida, sin embargo, la cantidad de vehículos utilizados es un datos que tiene que ir.

Lo presentado anteriormente es la estructura básica de las instancias propuestas por Solomon. Estas instancias poseen una cantidad de puntos a visitar distintos, variables entre 25, 50 y 100 puntos. Hay que enfatizar que estas instancias y las soluciones óptimas obtenidas en [3, 6] son cuando la distancia es considerada como distancia euclidiana.

4. Modelo Matemático

Luego de entender como funciona el problema y que variables puede llevar o ignorarse, se procede a modelar este problema de una manera matemática. Esto es por que al explicar las cosas con palabras, estas pueden quedar ambiguas, pero matemáticamente se espera que esa ambigüedad no exista.

Siguiendo lo propuesto por Nasser [2] en su modelo matemático, se mostrará lo planteado por el autor. Se presentará un orden con tal de entender finalmente lo que el autor quiere explicar.

4.1. Parámetros

- N: Las rutas que se han hecho.
- C: El conjunto de clientes.
- V: Los vehículos con las mismas condiciones, es decir, misma capacidad, velocidad, etc.
- q: Capacidad de los vehículos.
- c_{ij}: Costo asociado de viajar de i a j.
- d_i: Demanda del cliente i.
- s_{ik}: Servicio dado por el vehículo k al cliente i.
- t_{ij}: Tiempo que se lleva en servicio.

4.2. Variables

$$x_{ijk} = \begin{cases} 1 & \text{Si el vehiculo } k \text{ toma el camino de } i \text{ a } j. \\ 0 & \text{Si no.} \end{cases} \tag{1}$$

4.3. Función objetivo

$$\text{Min} \sum_{k \in V} \sum_{i \in N} \sum_{j \in N} c_{ij} x_{ijk} \tag{2}$$

4.4. Restricciones

$$\sum_{k \in V} \sum_{j \in N} x_{ijk} = 1;\ \forall i \in C \tag{3}$$

$$\sum_{i \in C} d_i \sum_{j \in N} x_{ijk} \leq q;\ \forall k \in V \tag{4}$$

$$\sum_{j \in N} x_{0jk} = 1;\ \forall k \in V \tag{5}$$

$$\sum_{i \in N} x_{ihk} - \sum_{j \in N} x_{hjk} = 0;\ \forall h \in C,\ \forall k \in V \tag{6}$$

$$\sum_{i \in N} x_{i,n+1,k} = 1;\ \forall k \in V \tag{7}$$

$$s_{ik} + t_{ij} - K(1 - x_{ijk}) \leq s_{jk};\ \forall i, j \in N,\ \forall k \in V \tag{8}$$

$$a_i \leq s_{ik} \leq b_i;\ \forall i \in N,\ \forall k \in V \tag{9}$$

$$x_{ijk} \in \{0, 1\},\ \forall i, j \in N,\ \forall k \in V \tag{10}$$

La ecuación (2) hace referencia a minimizar el costo de realizar las rutas. Este costo puede traducirse en distancia recorrida, tiempo utilizado en la ruta, etc. La ecuación (3) asegura que cada nodo/cliente es visitado solo una vez y la ecuación (4) dice que los vehículos no sobrepasaran su capacidad. Las ecuaciones (5), (6) y (7) aseguran que los vehiculos saldran del deposito,

despues de visitar a un nodo/cliente estos se irán y finalmente que su ultimo lugar de visita será el deposito. La ecuación (8) dice que un vehiculo no podrá llegar si se paso en su ventana de tiempo. (9) dicta que se cumpla el periodo de la ventana de tiempo y finalmente (10) asegura la integridad de las variables.

Como fue dicho anteriormente, este modelo puede ser visto de manera más detallada en [2].

5. Representación

Los datos de entrada en la instancia que fueron mencionados anteriormente, se explicarán para entender su dominio dentro del problema.

El problema se ubica en un camino que se puede representar como un plano cartesiano, es decir, posee una variable x e y por lo que su dominio cae en $(x, y) \in \mathbb{R}^2$. Por otro lado, el tiempo es una variable de una dimensión, por lo que su dominio es $t \in \mathbb{R}_0^+$. La demanda es una exigencia que tiene el deposito con sus clientes, por lo tanto esta también posee un dominio de tipo positivo $d \in \mathbb{R}_0^+$. Dado que si se incluye una capacidad a los vehículos este problema pasaría a ser Capacitated VRP with Time Windows (CVRPTW), por lo que la capacidad se ignorará en este problema.

La representación utilizada al momento de construir el algoritmo greedy se compone de 3 principales partes. La primera es un vector de que indica si el nodo i fue visitado, la segunda es una matriz de tiempos/distancia que se calcula a partir de la distancia euclidiana entre sus componentes y tercero son variables, una de tipo entero y otra de tipo flotante; la de tipo entero guarda la cantidad de vehículos necesitados para completar el circuito y; la de tipo flotante es para guardar la distancia recorrida por los vehículos en total.

6. Descripción del algoritmo

Este algoritmo se desarrollo bajo un concepto greedy, es decir, se opta por lo mejor de ese momento sin importar si a la larga puede afectar al resultado. Entendiendo esto, se necesita entender cual es la parte greedy del algoritmo propuesto: recorrer con un auto la mayor cantidad de clientes posibles, esto para reducir la cantidad de vehículos usados. Si se ve de una manera mas realista, contratar mas vehículos puede costar mucho, por lo que se trata de usar la menos cantidad posible y pagar menos por los transportes.

Entendiendo donde esta lo greedy en el problema, se trata de modelar este comportamiento, esto es, haciendo una búsqueda de clientes factibles con el vehículo actual, no agregando un nuevo vehículo hasta que el anterior no pueda cumplir con los tiempos y clientes establecidos por la instancia. Cuando este no pueda avanzar a un potencial cliente por falta de tiempo, se le obliga a esperar hasta que lo pueda atender. En caso que lo anterior no se pueda cumplir, se procede a enviar otro vehículo con la esperanza de completar el circuito con la menos cantidad de ellos.

Para la comparación con las soluciones óptimas encontradas para las instancias de Solomon, se procede a mostrar como salida del programa la distancia recorrida y la cantidad de vehículos utilizados.

7. Experimentos

Para obtener los valores de las instancias, se procede a modificar un poco las instancias de Solomon por la siguiente estructura:

7.1. Instancia Solomon

```
1  C102
2
3  VEHICLE
4  NUMBER     CAPACITY
5    25         200
6
7  CUSTOMER
8  CUST NO. XCOORD YCOORD DEMAND READY TIME DUE DATE SERVICE TIME
9
10    0        40     50       0         0        1236           0
11    1        45     68       10        0        1127          90
```

7.2. Instancia personal

```
1  26
2  40          50           0           0        1236           0
3  45          68          10         912         967          90
```

Para facilitar la lectura en el programa greedy propuesto, se eliminó el encabezado, los índices de los clientes y el deposito, el número de vehículos y se agregó un numero que indica la cantidad de nodos a utilizar.

En el presente documento se utilizaron instancias poco complejas pero algunas con gran cantidad de datos. El computador en el cual se realizan las pruebas no cuenta con el soporte necesario para procesar una instancia de datos grandes y complejos.

El computador utilizado tiene como características un procesador Intel Core i5 de 2,3Ghz; 4GB de RAM y una tarjeta de vídeo integrada Intel HD Graphics 3000 384 MB.

8. Resultados

Instancia	Personal		Óptimo		Diferencia
	Dist	K	Dist	K	
C101.25	600	3	191.3	3	3.13 %
C101.50	1642	5	362.4	5	4.53 %
C101.100	4964	10	827.3	10	6.00 %
C102.25	463	2	190.3	3	2.43 %
C201.25	330	1	214.7	2	1.54 %
R201.50	2022	1	791.9	6	2.55 %
RC101.50	2113	1	944	8	2.24 %
RC201.25	1356	1	360.2	3	3.76 %

Cuadro 2: Resultados obtenidos por greedy vs el óptimo conocido [3].

9. Conclusiones

Finalmente, se puede concluir algo que era esperable de este tipo de algoritmo, y es que greedy siempre conlleva un costo más caro que realizar una metaheurística conocida. Sin embargo, no es malo pensar de forma greedy para iniciar el ataque a un problema nuevo. Esta técnica no se encuentra nombrada mucho en la literatura de este problema, esto puede darse por el mal rendimiento que conlleva (2).

Si bien greedy no es una de las opciones más viables para resolver un problema NP, puede ser útil para obtener soluciones iniciales con el objetivo de aplicar técnicas reparadoras y obtener mejores soluciones que las dadas por greedy. Como fue dicho anteriormente, no se puede asegurar que se obtenga la solución óptima global pero si mejorar algo.

En cuanto al estado del arte, este problema ha sido estudiado de hace mucho tiempo [9]. Varios estudios, variantes, diferentes técnicas para abordar el mismo problema son algunas de las cosas que se pueden encontrar en VRP. Además este problema posee un acercamiento muy apegado a la realidad, ya que modela de muy buena manera la logística y, por lo tanto, su uso es muy cotizado por las grandes empresas.

Este problema, al tener tantas variantes distintas, puede llegar a ser un interesante problema. En efecto, existen variables de este problema que tratan de apegarse mucho a la realidad como CVRPTW, VRPMTW, entre otras variantes que mezclan algunas ya conocidas para agregar complejidad.

10. Bibliografía

Referencias

[1] Yaw Chang and Lin Chen. Solve the vehicle routing problem with time windows via a genetic algorithm. *University of North Carolina*, pages 240–249, 2007.

[2] Nasser A. El-Sherbeny. Vehicle routing with time windows: An overview of exact, heuristic and metaheuristic methods. *King Saud University*, pages 123–131, 2010.

[3] Marius M. Solomon Guy Desaulniers, Jacques Desrosiers. *Column Generation*. Springer US, 1 edition, 2005.

[4] Marius Solomon Martin Desrochers, Jacques Desrosiers. A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research*, 40(2):342–354, 1992.

[5] Aida Calviño Martínez. Cooperación en los problemas del viajante (tsp) y de rutas de vehículos (vrp): una panorámica. Master's thesis, Universidad de Santiago de Compostela, 2011.

[6] Marius M. Solomon Oli B. G. Madsen François Soumis Niklas Kohl, Jacques Desrosiers. 2-path cuts for the vehicle routing problem with time windows. *Transportation Science*, 33:101–116, 1999.

[7] Michel Gendreau Olli Bräysy. Vehicle routing problem with time windows, part i: Route construction and local search algorithms. *Transportation Science*, pages 104–118, 2002.

[8] Amanur Rahman Saiyed. The traveling salesman problem. *Indiana State University*, pages 1–15, 2012.

[9] Marius M. Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, pages 254–265, 1987.