UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA MATEMÁTICA

MULTIPLE TRAVELING SALESMAN PROBLEM WITH HANDLING TIMES ON PATHS AND SPIDERS

TESIS PARA OPTAR AL GRADO DE MAGÍSTER EN CIENCIAS DE LA INGENIERÍA, MENCIÓN MATEMÁTICAS APLICADAS.
MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL MATEMÁTICO.

IAN ANDRÉS VIDAL MORALES

PROFESOR GUÍA:
IVÁN RAPAPORT ZIMERMANN
PROFESOR CO-GUÍA:
JOSÉ SOTO SAN MARTÍN

MIEMBROS DE LA COMISIÓN:
MARTÍN MATAMALA VÁSQUEZ

Este trabajo ha sido parcialmente financiado por Proyecto Regular Fondecyt 1170021 y CMM-Conicyt PIA AFB170001

SANTIAGO DE CHILE
2019

RESUMEN DE LA MEMORIA PARA OPTAR AL TÍTULO
DE MAGÍSTER EN CIENCIAS DE LA INGENIERÍA, MENCIÓN MATEMÁTICAS APLICADAS
POR: IAN ANDRÉS VIDAL MORALES
FECHA: JULIO 2019
PROF. GUÍA: IVÁN RAPAPORT ZIMERMANN JOSÉ SOTO SAN MARTÍN

MULTIPLE TRAVELING SALESMAN PROBLEM WITH HANDLING TIMES ON PATHS AND SPIDERS

Dado un conjunto finito de N vértices, el tiempo de viaje entre cada uno de ellos, una cantidad m de vehículos y un punto de origen llamado *depósito*, en el *multiple traveling salesman problem* se desea encontrar m rutas que, partiendo desde el *depósito*, visiten a todos los vértices, a los que llamaremos clientes. En este trabajo tendremos en consideración el tiempo de visita a los clientes, el cual supondremos el mismo para cada uno de ellos.
Se estudian dos variantes del problema: la primera es minimizar el máximo tiempo de ruta de los vehículos y, la segunda, es minimizar el tiempo de completación, que corresponde a minimizar el tiempo en que se atiende al último cliente.
Se estudió el problema cuando la red de *clientes-depósito* es un camino, un ciclo y una *araña*. Para el camino, el ciclo y la araña se encontraron soluciones óptimas estructuradas. Para el camino y el ciclo, algoritmos polinomiales. Para la araña con 2 vehículos se encontró una 3/2-aproximación para el tiempo de ruta y una 5/3-aproximación para el tiempo de completación. Finalmente, para la araña con una cantidad m fija de vehículos, se dedujo una $(1+\varepsilon)$-aproximación en tiempo $N^{O(m/\varepsilon)}$ tanto para el tiempo de completación como para el tiempo de ruta.

A mi familia, amigos y a todos los que me apoyaron en este viaje.

Agradecimientos

Quiero agradecer a todas las personas que estuvieron a mi lado durante mi paso por la universidad, las cuales hicieron posible que éste trabajo exista.

A mi familia que me han dado su apoyo incondicional a todo lo que intentado hacer.
A mi mamá, que todos los días me preparó el almuerzo para que me alimentara de buena manera, que en la mañana se preocupaba de que no fuera a llegar tarde sin importarle que le dijera que no lo hiciera ni que algunos días no tuviera que llegar temprano.
A mi papá, que siempre se preocupó que estuviera bien, que cada vez que el creía que algo me podía estar afectando se acercaba a hablarme, aunque sus conversaciones eran demasiado largas para una simple falta de sueño.
A mi hermana, la que en esos días en los que me sentía consumido por los estudios o que creía que no podía pensar en nada más, me hablaba de su día a día en el colegio, haciendo que por al menos una hora me desconectara de todo sin importarle su cansancio, ni lo ocupada que estaba ella o lo ocupado que estaba yo, ni siquiera que en una conversación deben intervenir al menos 2 personas.
A mi hermano, que me enseño que en los momentos más difíciles es en los momentos que más tranquilo se debe estar, y que gracias a su tranquilidad, en más de alguna ocasión llegué tarde.

A mis amigos, que han estado conmigo desde el colegio.
A Byron, que cuando necesité hablar, al otro día me estaba esperando con pizza.
A Cuevas y a Loyola, que estuvieron presente en casi todas mis salidas universitarias.

A mi profesor guía Iván, porque siempre se hizo el tiempo cuando necesité hablar con él, porque al final de cada reunión siempre hubo una palabra de ánimo.
A mi profesor co-guía José, porque su consejo en más de alguna ocasión me sacó de los que creí caminos sin salida.
A todo el equipo docente del DIM, en especial a Silvia, a su gran paciencia y su buena voluntad. Sin ella, este trabajo fácilmente se pudo haber atrasado meses.

Y quiero dar gracias a todas las personas que no mencioné pero que estuvieron a mi lado, por que me tomaría un libro completo agradecer a cada una de ellas.

Tabla de Contenido

Introducción **1**

1. Preliminares **4**
1.1. Grafos 4
1.2. Algoritmos 5
1.3. Problema del vendedor viajero 5

2. mTSPHT* en el camino **8**
2.1. Depósito en un extremo del camino 9
2.1.1. Depósito en extremo : caso $m = 1$ 9
2.1.2. Depósito en extremo : caso $m = 2$ 10
2.1.3. Depósito en extremo : caso $m > 2$ 13
2.2. Depósito libre 15
2.2.1. Depósito libre : $m = 1$ 15
2.2.2. Depósito libre : caso $m \geqslant 2$ 17
2.3. mTSPHT* en el ciclo 25

3. mTSPHT* en la araña **32**
3.1. Aproximación para $m = 2$ 39
3.1.1. Tiempo de ruta 39
3.1.2. Tiempo de completación 42
3.1.3. PTAS para m fijo 47

4. Conclusión **54**

A. Reducción de mTSPHT a mTSP en árboles: variante tiempo de ruta **56**

Bibliografía **58**

Índice de Ilustraciones

2.1. Ejemplo notación 9
2.2. Ejemplo de la ejecución de MOVEMENT-1 11
2.3. Ejemplo de orden parcial y rutas concurrentes 18
2.4. Ejemplo de la ejecución de MOVEMENT-2 19
2.5. Ejemplo de la ejecución de SHIFT 22
2.6. Ejemplo notación ciclo 26

3.1. Ejemplo notación Araña 32

Introducción

El problema del vendedor viajero (TSP) es uno de los problemas más icónicos de estudio debido a su antigüedad y su amplio espectro de aplicaciones, que va desde la logística hasta la electrónica (fabricación de circuitos electrónicos).
El *Multiple Traveling Salesman Problem* (mTSP) es una generalización de TSP, mientras que en (TSP) se cuenta con un vendedor que debe visitar a los clientes, en (mTSP) contamos con m vendedores para visitar a los clientes. Cabe destacar que (mTSP) se puede ver como un caso particular de *Vehicle Routing Problem* (VRP) cuando no se considera la capacidad de los vehículos, a esta variante la denotaremos simplemente por (VRP).
Más especificamente, suponga que tenemos un grafo con $N + 1$ nodos, en N de ellos hay un cliente que necesita ser atendido y además contamos con un nodo de origen al que llamamos *depósito*. Para satisfacer a los clientes, contamos con m vendedores (o vehículos) disponibles que tienen su punto de partida en el *depósito*. El *m-traveling salesman problem* desea encontrar m rutas que los vendedores deben seguir, partiendo desde el *depósito*, para visitar a cada cliente de manera que la distancia más larga recorrida por algún vendedor sea lo más pequeña posible posible. Algunas veces en vez de la distancia recorrida, se minimiza el tiempo que le toma al último vendedor completar su ruta.
Como ha sido mencionado, (mTSP) puede verse como un caso particular de (VRP) y es precisamente ese el enfoque que se ha tomado: a los vértices del grafo, con excepción del *depósito*, serán llamados clientes. Además, en vez de hablar de vendedores, se hablará de vehículos. Por otro lado, consideramos que el tiempo de atención a los clientes no puede ser ignorado al momento de planificar la ruta de cada vehículo, he aquí la razón de estudiar la variante de (mTSP) con *handling times* (o tiempos de visita en español), a esta variante la denotaremos por (mTSPHT). Además, si los tiempos de visita son los mismos para todos los clientes, se denotará a la variante del problema por (mTSPHT*).

Motivación y Trabajos relacionados

Éste trabajo intenta estudiar dos visiones: dar una buena satisfacción a los clientes, sin dejar de lado el beneficio de los vehículos, he de aquí que se desea minimizar el máximo largo de ruta (se quiere que el tiempo en que el último vehículo vuelve al depósito sea lo más pequeño posible) y minimizar el máximo tiempo de completación (se quiere minimizar el tiempo en que el último cliente sea satisfecho).
Se han realizado algunas simplificaciones con respecto al problema original: supondremos

que el tiempo de atención de cada cliente es el mismo para todos, por ejemplo éste puede ser el tiempo de atención promedio por cliente. Además, se estudia el caso cuando la red de *clientes-depósito* tiene cierta topología, ya sea un camino, un ciclo o una araña.

Durante esta sección notaremos por n a la cantidad de clientes o vértices y por m a la cantidad de vehículos disponibles o rutas.
En [6] , [7] y [5] se estudia el multi-vehicle scheduling problem (VSP), con handling times y release time, en la línea. Karuno y Nagamochi entregan [6] [7] una 2-aproximación y además, para cada m fijo, entregan una una $(1+\varepsilon)$-aproximación en tiempo $O((1+2/\varepsilon)m^2n^2(n+2^{1+2/\varepsilon})(mn^3(1+2/\varepsilon)/\varepsilon)^{m(1+2/\varepsilon)}$, mientras que Gaur et al. [5] entregan una $\frac{5}{3}$-aproximación.
En [12], Yu y Liu estudian dos variantes de (VRP) con release time en la línea, la versión tour y la versión path. En la versión tour se busca minimizar el máximo makespan de los vehículos, mientras que en la versión path se busca minimizar el máximo tiempo de completación de los clientes. Yu y Liu muestran que tanto en la versión tour del problema como en la versión path, es posible encontrar la solución en tiempo polinomial.
En [1], Bao y Liu estudian el (VSP) con release time y handling times en el ciclo y en el árbol. Para el problema en el ciclo, tanto para la version tour como para la version path entregan una 12/7 aproximación. Mientras que para el problema en el árbol entregan una 9/5-aproximación para la versión de tour y una 27/14-aproximación para la versión path.
En [11], para (VSP) con handling times en grafos generales, Xu et.al encuentran una máx$\{3-2/m, 2\}$-aproximación.
Yu y Liu [13] estudian el min-max cycle cover problem (MMCP) y el rooted min-max cycle cover problem (RMMCP), encontrando una 5-aproximación para el (MMCCP) y muestran que si se tiene una α-aproximación para (MMCCP) es posible obtener una $(1+\alpha)$-aproximación para (RMMCCP), por lo que entregan una 6-aproximación para (RMMCCP). Mientras que Xu et.al [10] encuentran una $(6\frac{1}{3}+\varepsilon)$-aproximación en tiempo $O((n^2m^2+m^5)(\log n+\log\frac{1}{\varepsilon}))$ para el (RMMCCP).
Por último, en [9] y [2] se estudia el (mTSP), minimizando máximo el makespan, en el árbol. Xu et.al [9], para cada m fijo, encuentran un algoritmo pseudo-polinomial y una $(1+\varepsilon)$-aproximación (manteniendo m fijo) en tiempo $O(n^{2m-1}/\varepsilon^{2k-2})$. Mientras que Becker y Paul [2] logran una $(1+\varepsilon)$ aproximación en tiempo $n^{O(\varepsilon^{-8})}$.

Organización

En el primer capitulo se introducen los conceptos básicos para el desarrollo de la tesis. Se definen los conceptos básicos de teoría de grafos. Se define los conceptos de algoritmos de aproximación. Se describe el problema del vendedor viajero, para luego describir nuestro problema. Se define el concepto de ruta, tiempo de ruta y tiempo de completación.

En el segundo capitulo se realiza el estudio de (mTSPHT*) en el camino y en el ciclo. Se prueba que podemos reducir el rango de búsqueda a ciertas soluciones, y luego se caracteriza un tipo de solución óptima lo que permite obtener algoritmos para resolver el problema.

En el tercer capítulo se estudia (mTSPHT*) en la araña, otra vez se caracteriza un tipo de solución óptima, generalizando el caso del camino. Se demuestra que el problema es **NP-difícil**. Luego se obtiene un algoritmo de aproximación para el caso de 2 vehículos, tanto para el tiempo de completación como para el tiempo de ruta. Luego, se obtiene un **PTAS** para el tiempo de completación y el tiempo de ruta.

Resultados

Los resultados principales de la tesis son: mostrar que es posible resolver (mTSPHT*) en un camino, con N clientes y m vehículos, en tiempo $O(mN^4)$ y que (mTSPHT*) en un ciclo, con N clientes y m vehículos, se puede resolver en tiempo $O(mN^5)$.
Se prueba que (mTSPHT*) en una araña, sin límite en el número de ramas, es **NP-difícil**. Además, cuando $m = 2$ se entrega una $\frac{3}{2}$-aproximación para el tiempo de ruta y una $\frac{5}{3}$-aproximación para el tiempo de completación. Y por último, para (mTSPHT*) en una araña, con N clientes y m vehículos, se entrega una $(1+\varepsilon)$-aproximación en tiempo $N^{O(m/\varepsilon)}$ para el tiempo de ruta y el tiempo de completación.

Capítulo 1

Preliminares

Este capitulo esta dedicado a introducir las definiciones y notaciones que se utilizarán en el resto de este trabajo. Se comienza entregando definiciones básicas de grafos y algoritmos, para luego entrar a definir el problema central de estudio.

1.1. Grafos

Un *grafo no dirigido*, de ahora en adelante un *grafo*, es un par de conjuntos $G = (V, E)$ donde V es un conjunto finito, llamado conjunto de vértices, y $E \subseteq \binom{V}{2}$, llamado conjunto de aristas. Al conjunto de vértices de G lo notaremos por $V(G)$, y de la misma manera, al conjunto de aristas de G los notaremos por $E(G)$. Estas convenciones son independientes de los nombres reales de ambos conjuntos: El conjunto de vértices W de un grafo $H = (W, F)$ se seguirá escribiendo $V(H)$ y no como $W(H)$.
Diremos que dos vértices u, v son *adyacentes* o *vecinos* si uv es una arista de G. Por otro lado, diremos que una arista e es incidente a un vértice v si $v \in e$. Al conjunto de vecinos de u lo notaremos por $N(u)$ y al cardinal de $N(u)$ le llamaremos el *grado* de u. En general, si $U \subseteq V$, al conjunto de vecinos de los vértices de U en $V \setminus U$ lo notaremos por $N(U)$.
Sea $G' = (V', E')$, si $V' \subseteq V$ y $E' \subseteq E$, entonces diremos que G' es un *subgrafo* de G, de manera informal, diremos que G contiene a G'.
Sea $U \subseteq V$, escribimos $G - U$ para describir al grafo obtenido al borrar todos los vértices de U y sus aristas incidentes. Llamaremos $G[U]$ al subgrafo de G *inducido* por los vértices de U. Es decir, para todo $u, v \in U$, la arista uv pertenece a $G[U]$ si y sólo si $uv \in G$. Por otro lado, si H es un subgrafo de G, notaremos por $G[H]$ al subgrafo inducido por $G[V(H)]$.
Para $F \subseteq \binom{V}{2}$, escribiremos

$$G - F := (V, E \setminus F)$$
$$G + F := (V, E \cup F)$$

Con las nociones básicas anteriores, ya podemos entrar a definir algunos tipos especiales de grafos.

Definición 1.1 Camino
Un camino es un grafo no vacío $P = (V, E)$ de la forma:

$$\begin{aligned} V &= \{v_0, v_1, \ldots, v_k\}, \text{ con todos los } v_i \text{ distintos,} \\ E &= \{v_0v_1, v_1v_2, \ldots, v_{k-1}v_k\} \end{aligned}$$

A menudo nos referiremos a un camino por la secuencia natural de sus vértices, es decir, $P = v_0v_1, \ldots, v_k$ y diremos que P es un camino desde v_0 a v_k.

Definición 1.2 Ciclo
Si $P = v_0, \ldots, v_{k-1}$ es un camino y $k \geqslant 3$, entonces el grafo $C := P + v_{k-1}v_0$ es un ciclo.

Un grafo no vacío G es llamado *conexo* si todo par de vértices existe un camino camino en G. Un grafo sin ciclos es llamado *bosque*. Un bosque conexo es llamado ***árbol***.

Definición 1.3 Araña
Un grafo G se dice araña si G es un árbol con sólo un vértice de grado mayor a dos, al cual llamaremos el centro de la araña.

1.2. Algoritmos

Vamos a asumir que el lector sabe lo que es un algoritmo y es familiar con la técnica de programación dinámica. Además, tiene conocimientos básicos de complejidad computacional; a saber: conoce la notación asintótica, sabe realizar análisis de complejidad de algoritmos y conoce las clases de complejidad **P** y **NP**.
Las siguientes definiciones fueron extraídas desde [8].

Definición 1.4 Algoritmo de α-aproximación *Un α algoritmo de aproximación para un problema de optimización es un algoritmo en tiempo polinomial tal que para todas las instancias del problema produce una solución cuyo valor está dentro de un factor α del valor de una solución óptima.*

Definición 1.5 Polynomial-time approximation scheme (PTAS)
Un Polynomial-time approximation scheme *es una familia de algoritmos $\{A_\varepsilon\}$, en la cual existe un algoritmo para cada $\varepsilon > 0$ tal que A_ε es un $(1+\varepsilon)$-algoritmo de aproximación (para problemas de minimización) o un $(1-\varepsilon)$-algoritmo de aproximación (para problemas de maximización).*

1.3. Problema del vendedor viajero

Dado un conjunto de ciudades, junto con el costo de viaje entre cada una de ellas, el problema del vendedor viajero, o (TSP), es encontrar la forma más económica de visitar todas las ciudades y volver al punto de inicio. La "forma de visitar todas las ciudades" es

simplemente el orden en cual las ciudades son visitadas [3].
El problema ha tratar en este trabajo es una generalización del (TSP): en (TSP) existe un vendedor, desde ahora vehículo, que debe visitar a las ciudades, desde ahora clientes. A diferencia del problema de estudio en el cual se tienen m vehículos para visitar a los clientes. Además, al momento de planificar el orden de visita, se considerará el costo (tiempo) de visita que toma cada cliente.
Para definir el problema, empezaremos por dar la terminología a utilizar: dado un grafo $G = (V \cup \{D\}, E)$ donde V son los clientes a visitar y D denota al depósito, se incluye una función $t : V \to \mathbb{R}_+$ que corresponde al tiempo de visita de los clientes y una función $t : E \to \mathbb{R}_+$, que representa el tiempo de viaje entre cada par de clientes unidos por alguna arista.

Observación El desplazamiento entre u y v a través del grafo será a través de un camino mínimo entre ambos clientes, el cual lo supondremos conocido de antemano. Extenderemos la notación de $t : E \to \mathbb{R}_+$ a $t : \binom{V}{2} \to \mathbb{R}_+$ denotando el tiempo de viaje entre v y u (valor del camino mínimo entre u y v) como $t(u, v)$.

Definición 1.6 Ruta
Una ruta S es una secuencia ordenada de vértices de $V \cup \{D\}$, $(s_j)_{j=0}^{n}$ (en donde el número de clientes visitados en una ruta S es n) con $s_0 = D$.

Observación s_1 y s_n corresponden al primer y último cliente visitado respectivamente por S. Notaremos por $C(S) = \{v \in V : \exists! i \in \{1, \dots, n\}, s_i = v\}$ al conjunto de clientes de la ruta S. Cuando no haya confusión, utilizaremos $|S|$ para referirnos a $|C(S)|$.

Definición 1.7 Solución factible
Diremos que $\hat{S}$ es una solución factible del problema con m vehículos si:

1. *$\hat{S} = (S^1, S^2, \dots, S^m)$ donde S^k es ruta $\forall k \in \{1, \dots, m\}$.*
2. *$V = C(S^1) \dot{\cup} \dots \dot{\cup} C(S^m)$.*

Observación Cuando $m = 1$, omitiremos los súper-índices de la solución. Es decir, si $\hat{S}$ es solución factible con un vehículo, $\hat{S} = (S)$.

Definición 1.8 Tiempo de completación
Sea $\hat{S} = (S^1, S^2, \dots, S^m)$ una solución factible. Definimos el tiempo de completación de un cliente v visitado por el vehículo k ($v \in C(S^k)$), como:

$$T_{S^k}(v) = \sum_{j=1}^{n_v^k} [t(s_{j-1}^k, s_j^k) + t(s_j)]$$

donde $S^k = (s_j^k)_{j=0}^{n_k}$ es tal que $v \in C(S^k)$ y $s_{n_v^k}^k = v$.
Cuando no haya confusiones, denotaremos el tiempo de completación de v por $T(v)$. Dado que el tiempo de completación $T(s_n^k)$ es creciente en función de n, podemos definir el tiempo de completación de S^k como:

$$T(S^k) = T_{S^k}(s_{|C(S^k)|}^k)$$

Observación El tiempo de completación de dos clientes en una misma ruta es no-decreciente con respecto al orden de visita. Es decir, si $S = (s_j)_{j=0}^{n_S}$ es una ruta, entonces $(j \leqslant j' \Rightarrow T(s_j) \leqslant T(s_{j'}))$

Definición 1.9 Tiempo de ruta
Sea $\hat{S}$ una solución factible, definimos el tiempo de ruta de S^k, $Tr(S^k)$ como:

$$Tr(S^k) = \sum_{j=1}^{n_k} [t(s_{j-1}^k, s_j^k) + t(s_j^k)] + t(s_{n_k}^k, D) = T(S^k) + t(s_{|C(S^k)|}^k, D)$$

Ahora estamos en condiciones de definir el problema

Definición 1.10 Multiple Traveling Salesman Problem with Handling Times (*mTSPHT*)
Dado un grafo $G = (V \cup \{D\}, E)$, $m \in \mathbb{N}$, $t : V \to R_+$ y $t : E \to R_+$, con t denotando el tiempo de visita y la función de tiempo de viaje respectivamente. Se desea encontrar una solución factible que ***minimice el máximo tiempo de completación/tiempo de ruta****. A este problema le llamaremos* multiple Traveling Salesman Problem With Handling Times*.*

Como es posible observar, (mTSP) tiene un nivel de dificultad mayor que (TSP), ya que aparte de tener que buscar el orden de visita, hay que repartir los clientes a los vehículos, razón por la cual vamos a realizar ciertas simplificaciones al problema:

1. El tiempo de visita de cada cliente es el mismo para todos los clientes.
2. Asumiremos cierta estructura sobre la red.

Por lo tanto, el problema que vamos a estudiar durante este trabajo corresponde a (mTSPHT) con tiempos de visita constantes, al cual nos referiremos por (mTSPHT*), en distintos tipos de redes.

Observación Es posible que en algunas redes, tales como caminos, sea imposible dejar de pasar por un cliente dos veces, por lo que haremos una distinción entre *visitar* y *transitar* por un cliente. Todos los clientes deben ser visitados solo una vez (es ahí donde consideramos el tiempo de visita), pero no hay límite para el número de veces en que se puede transitar por ellos.

Capítulo 2

mTSPHT* en el camino

Empezaremos estudiando el problema cuando la red de clientes y el depósito están distribuidos en un camino. Recuerde que estamos considerando que el tiempo de visita de cada cliente es constante, es decir, que $t(v) = t_0, \forall v \in V$. Vamos a definir un etiquetado que nos será de gran utilidad. Para ello es conveniente hablar de *ramas*.

Definición 2.1 Rama

- *Llamaremos rama a un camino maximal B que tiene por inicio el depósito.*
- *Denotaremos por $C(B)$ a los clientes de la rama B, y por $|B|$ al número de clientes en la rama B.*

Sea G el grafo que representa a la red de *depósito-clientes*. Si el grafo tiene 2 ramas, a la rama derecha la denotaremos por B_1 y a la izquierda por B_2 (ver Figura 2.1).
Con respecto a los clientes, enumeraremos con sub-índices a los clientes de una rama de acuerdo a su cercanía al *depósito* y con súper-índices asignaremos la rama a la que pertenecen. Es decir, para la rama B_b, el cliente más cercano al depósito será v_1^b y el más lejano será $v_{|B_b|}^b$.

Definición 2.2 label(v)
Definimos el label(v) *como el sub-índice de v. Es decir,* label(v) $= j$ *si* $v = v_j^b$ *con* $b \in \{1, 2\}$.

Por simplicidad, se empezará estudiando el caso en el que el *depósito* se encuentra en uno de los extremos del camino, para luego analizar el caso general.

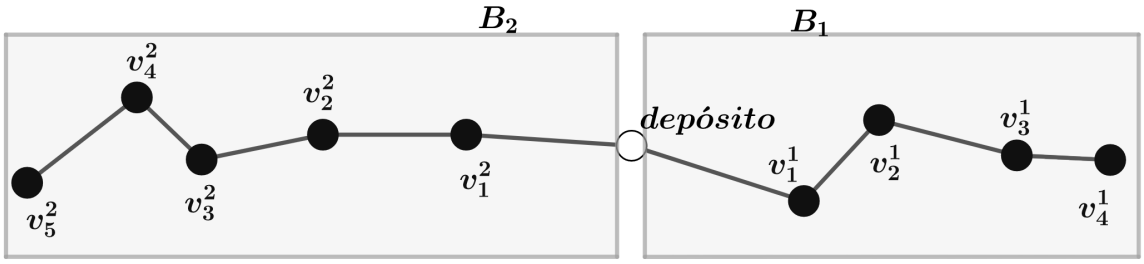


Figura 2.1: Las ramas B_1 y B_2 de (mTSPHT*) en un camino. El *depósito* está coloreado de blanco.

2.1. Depósito en un extremo del camino

Para fijar ideas, supondremos que el *depósito* se encuentra en el extremo izquierdo del camino. El otro caso (*depósito* en el extremo derecho) es análogo. Estudiaremos el problema de acuerdo a la cantidad de vehículos, es decir, de acuerdo al valor de m.

2.1.1. Depósito en extremo : caso $m = 1$

Partiremos por observar que el tiempo de completación de los clientes es una función creciente con respecto al orden de los clientes visitados. Es decir, si en una ruta S, v es visitado después de u entonces $T_S(u) \leqslant T_S(v)$. Es decir, el mayor máximo de completación corresponde al último cliente visitado.

Ahora volveremos a realizar una subdivisión de acuerdo a la función objetivo a considerar.

Min-Max Tiempo de Completación:

En este caso toda solución factible consta sólo de una ruta y ésta debe visitar a todos los clientes. Es decir, si S es ruta, entonces $|C(S)| = N$.

Ahora, analizando la función objetivo (recordar que suponemos $t(v) = t_0, \forall v \in V$)

$$\begin{aligned}\min_{S:S \text{ Ruta}} \max_{v \in C(S)} T(v) &= \min_{S:S \text{ Ruta}} \sum_{j=1}^{N} [t(v_{s_{j-1}}, v_{s_j}) + t_0] \\ &= Nt_0 + \min_{S:S \text{ Ruta}} \sum_{j=1}^{N} [t(v_{s_{j-1}}, v_{s_j})]\end{aligned}$$

con lo que el problema se reduce a minimizar $\sum_{j=1}^{N} t(v_{s_{j-1}}, v_{s_j})$. Pero basta notar que lo anterior siempre es mayor o igual a $t(D, v_N)$ (pues se tiene que visitar al cliente más lejano del depósito), y esto se alcanza cuando se visitan los clientes "de izquierda a derecha", por lo que la ruta óptima corresponde a visitar a los clientes en orden creciente de etiquetado.

Min-Max Tiempo de Ruta:
Al igual que antes, empezamos por analizar la función objetivo:

$$\begin{aligned} \min_{S:S \text{ Ruta}} Tr(S) &= t(v_{s_N}, D) + \sum_{j=1}^{N} [t(v_{s_{j-1}}, v_{s_j}) + t_0] \\ &\geqslant 2t(D, v_N) + Nt_0 \end{aligned}$$

donde la última desigualdad se debe a que al ser una ruta factible, debe visitar a todos los vértices (Nt_0), además debe visitar al vértice más lejano y volver al *depósito* ($2t(D, v_N)$). Sin embargo, recorrer los clientes de acuerdo a su etiquetado alcanza la cota, por lo que ésta es una manera óptima de recorrerlos. Es decir, en este caso también corresponde a recorrer a los clientes de izquierda a derecha.

Observación Si conociéramos los clientes que cierto vehículo debe visitar, el resultado anterior nos dice que podemos asumir conocido el orden de visita de ellos. Más aún, vamos a reducir nuestro estudio a soluciones cuyas rutas visiten a sus clientes por orden de etiquetado. Por lo que para los problemas con $m \geqslant 2$ sólo debemos estudiar la asignación de los clientes a los m vehículos.

Definiremos una operación que será usada con bastante frecuencia en este trabajo, la cual, dadas dos rutas distintas, realiza un intercambio de clientes. **Para utilizarla, es necesario saber reconstruir la ruta a partir de los clientes que debe visitar**.

Algorithm 1: EXCHANGE - intercambio entre dos clientes de dos rutas

Input: (S^1, v, S^2, u) con S^1, S^2 rutas y $v \in C(S^1), u \in C(S^2)$
Output: (S'_1, S'_2) resultantes al intercambiar v por u en (S^1, S^2)
1 $C(S'_1) \leftarrow C(S^1) - v + u$
2 $C(S'_2) \leftarrow C(S^2) + v - u$
3 **return** (S'^1, S'^2)

2.1.2. Depósito en extremo : caso $m = 2$

Tal como se dijo anteriormente, el problema que debemos abordar es el de la asignación de clientes. Para ello probaremos la siguiente proposición.

Proposición 2.3 *Para* (mTSPHT*) *en la línea con 2 vehículos, en la cual el depósito se encuentra en un extremo, existe una solución óptima* $S^* = (S^1, S^2)$ *en la cual* $C(S^i) = \{v_1, v_2, \ldots, v_{n_1}\}$ *y* $C(S^j) = \{v_{n_1+1}, v_{n_1+2}, \ldots, v_N\}$ *y* $N = |V|$.

Utilizaremos el algoritmo MOVEMENT-1, el cual dadas dos rutas S^1 y S^2 y una rama B, para fijar ideas, supondremos que el último cliente que visita S^2 está más alejado que el último cliente que visita S^1. Entonces, intercambiaremos clientes de los dos camiones de manera que ambos camiones visiten a clientes consecutivos, manteniendo el hecho que el último cliente que visita S^2 está más alejado que el último cliente que visita S^1 (ver figura 2.2).

Algorithm 2: MOVEMENT-1

Input: S^1, S^2 rutas factibles y B rama.
Output: S'^1, S'^2 par de rutas factibles para el problema

```
S'^1 ← S^1
S'^2 ← S^2
if label(s'^1_{n_1}) < label(s'^2_{n_2}) then
    while (∃u ∈ C(S'^2) : label(u) < label(s'^1_{n_1})) do
        (S'^1, S'^2) ← EXCHANGE(S'^1, s'^1_{n_1}, S'^2, u)
else
    Repeat: 4–5 cambiando los roles de S'^1 y S'^2
return (S'^1, S'^2)
```

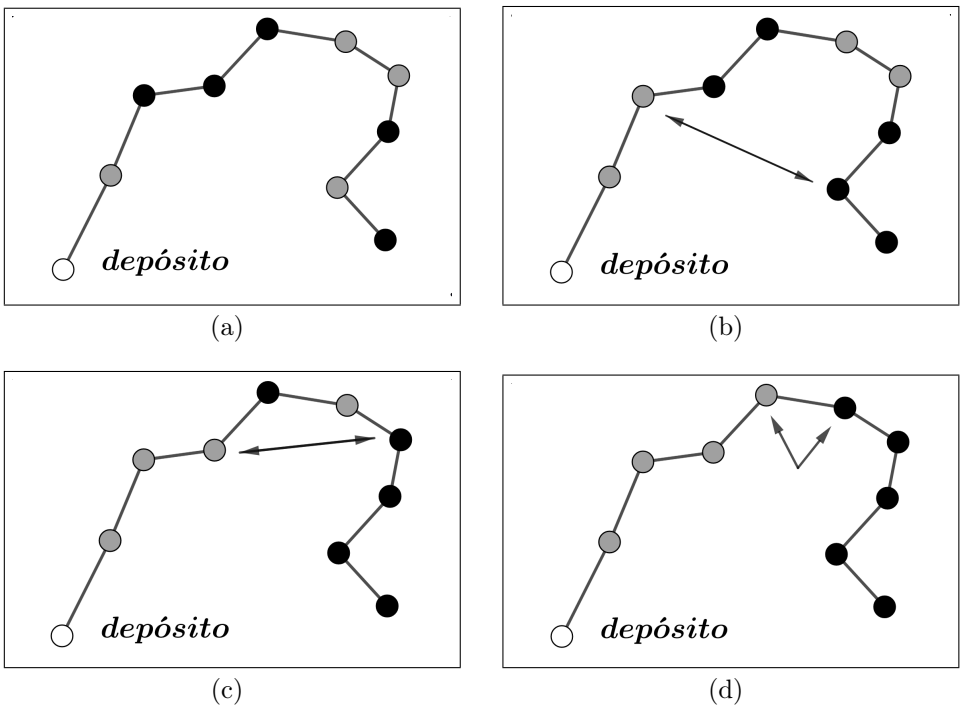


Figura 2.2: Ejemplo de la ejecución de MOVEMENT-1. Hay 2 rutas, una representada por el color gris y otra por el negro, el depósito esta pintado de blanco. En (a) se muestra el estado inicial, mientras que (b), (c) muestran estados intermedios. Finalmente, (d) muestra el estado final.

Lema 2.4 *En* (mTSPHT*)*, en el cual la red de* clientes-depósito *forman un camino y el depósito se encuentra en uno de los extremos del camino, las rutas obtenidas* (S'^1, S'^2) *al aplicar* MOVEMENT-1 *a un par de rutas* (S^1, S^2)*,cuyos vértices son visitados en orden creciente de enumeración, cumplen que el tiempo de completación/tiempo de ruta de* S'^i *no es mayor que* S^i *con* $i \in \{1, 2\}$*.*

Observación En MOVEMENT-1 no necesariamente $C(S^1) \cup C(S^2) = V$. Es decir, está pensado para ser usado cuando $m > 2$.

DEMOSTRACIÓN LEMA 2.4. Basta probar que en cada iteración (línea 4 de MOVEMENT-1) no aumenta ni el tiempo de completación ni el tiempo de ruta.

Sean S^1, S^2 un par de rutas cuyos vértices son visitados en orden creciente de enumeración, y sin pérdida de generalidad suponga que $label(s^1_{n_1}) < label(s^2_{n_2})$ (en el otro caso, intercambiamos los roles de S^1 y S^2). Si no existe $u \in C(S^2)$ tal que $label(u) < label(s^1_{n_1})$, entonces el algoritmo termina y las rutas no se ven modificadas. En caso contrario, se aplica EXCHANGE entre $(S^1, s^1_{n_1}, S^2, u)$, obteniendo (S'^1, S'^2).

Veamos que el tiempo de completación y el tiempo de ruta de S'^1, S'^2 no son más grandes que los de S^1, S^2.

Notar que luego de la operación EXCHANGE, se tiene que $|C(S^1)| = |C(S'^1)|, |C(S^2)| = |C(S'^2)|$. Todo vértice que estaba cubierto por $C(S^1 \cup S^2)$ lo sigue estando por $C(S'_1 \cup S'_2)$ (EXCHANGE intercambia las rutas a las que pertenecen los clientes por lo que cualquier cliente previamente cubierto lo sigue estando). Además, $label(s'^1_{n_1}) < label(s^1_{n_1})$, por lo que $t(D, s'^1_{n_1}) \leq t(D, s^1_{n_1})$, con lo que se tiene:

- *Tiempo de completación*

$$\begin{aligned}
T(S'^2) &= \sum_{j=1}^{n_2} [t(s'^2_{j-1}, s'^2_j) + t(s'^2_{n_2})] \\
&= t(D, s'^2_{n_2}) + |C(S'^2)|t_0 \\
&= t(D, s'^2_{n_2}) + |C(S^2)|t_0 \\
&= t(D, s^2_{n_2}) + |C(S^2)|t_0 \\
&= T(S^2) \\
T(S'^1) &= \sum_{j=1}^{n_1}]t(s'^1_{j-1}, s'^1_j) + t(s'^1_{n_1})] \\
&= t(D, s'^1_{n_1}) + |C(S'^1)|t_0 \\
&= t(D, s'^1_{n_1}) + |C(S^1)|t_0 \\
&\leq t(D, s^1_{n_1}) + |C(S^1)|t_0 \\
&= T(S^1)
\end{aligned}$$

- *Tiempo de ruta*

$$\begin{aligned}
Tr(S'^2) &= \sum_{j=1}^{n_2} [t(s'^2_{j-1}, s'^2_j) + t(s'^2_{n_2})] + t(D, s'^2_{n_2}) \\
&= t(D, s'^2_{n2}) + |C(S'^2)|t_0 + t(D, s'^2_{n_2}) \\
&= 2t(D, s'^2_{n_2}) + |C(S^2)|t_0 \\
&= 2t(D, s^2_{n_2}) + |C(S^2)|t_0 \\
&= T(S^2)
\end{aligned}$$

$$\begin{aligned} Tr(S'^1) &= \sum_{j=1}^{n_1}[t(s'^1_{j-1}, s'^1_j) + t(s'^1_{n_1})] + t(D, s'^1_{n_1}) \\ &= t(D, s'^1_{n_1}) + |C(S'^1)|t_0 + t(D, s'^1_{n_1}) \\ &= 2t(D, s'^1_{n_1}) + |C(S^1)|t_0 \\ &\leq 2t(D, s^1_{n_1}) + |C(S^1)|t_0 \\ &= T(S^1) \end{aligned}$$

Luego, ni el tiempo de completación ni el tiempo de ruta de S'^i son mayores que los de S^i, con $i \in \{1, 2\}$. Por lo tanto, en cada iteración, no aumentan ni el tiempo de ruta ni el tiempo de completación.
Es decir, las rutas (S'^1, S'^2), obtenidas con MOVEMENT-1 a partir de (S^1, S^2) cumplen que tanto su tiempo de completación como su tiempo de ruta son menores o iguales a las originales. □

Habiendo probado el Lema 2.5, tenemos las herramientas necesarias para caracterizar una solución óptima.

DEMOSTRACIÓN PROP 2.4. Sea $\hat{S}^* = (S^{*1}, S^{*2})$ una solución óptima del problema. Llamando $\hat{S} = (S^1, S^2)$ al par de rutas (en este caso solución) obtenidas al aplicar MOVEMENT-1 a (S^{*1}, S^{*2}, B), demostraremos que ésta es la solución buscada.

Por LEMA 2.4 se tiene que $\hat{S}$ también es una solución óptima del problema. Ahora debemos revisar el conjunto de clientes $C(S^1)$ y $C(S^2)$. Para analizar estos conjuntos, sin pérdida de generalidad, supondremos que $label(s^1_{n_1}) < label(s^2_{n_2})$.

En este caso veremos que

$$C(S^1) = \{v_1, v_2, \ldots, v_{i-1}\}, C(S^2) = \{v_i, v_{i+1}, v_{i+2}, \ldots, v_N\}$$

En efecto, supongamos que $\exists u \in C(S^2) : label(u) < label(s^1_{n_1})$. Esto no puede ocurrir pues es el resultado de MOVEMENT-1 y la existencia de u hubiera sido corregida en la línea 4 del algoritmo. Luego, como S es una solución factible, se cumple que

$$(\forall v \in V, \exists! j \in \{1, 2\} : v \in S^j)$$

y, por lo anterior,

$$\forall u, label(u) < label(s^1_{n_1}) \Rightarrow u \in C(S^1)$$

Luego

$$\begin{aligned} C(S^1) &= \{v_1, v_2, \ldots, v_{n_1}\} \\ C(S^2) &= V - \{D\} - C(S^1) \end{aligned}$$

Es decir, $C(S^2) = \{v_{n_1+1}, v_{n_1+1}, \ldots, v_N\}$ □

2.1.3. Depósito en extremo : caso $m > 2$

Ahora se estudiará el caso general. Para ello se utilizarán las técnicas empleadas para los casos $m = 1, 2$. Más específicamente, probaremos que existe una solución óptima tal que

$\forall k \in \{1,\dots,m\}, C(S^k) = \{v_{\mathrm{i}}, v_{\mathrm{i+1}}, v_{\mathrm{i+2}}, \dots, v_{\mathrm{i}+n_k-1}\}$. Es decir, el conjunto de clientes de cada vehículo es un conjunto de vértices tal que su etiquetado es un intervalo de $\mathbb{N}$. A esta última propiedad la llamaremos la **"propiedad de intervalos"**.

Proposición 2.5 *Para* (mTSPHT*)*en la línea, en la cual el depósito se encuentra en un extremo, con tiempos de procesamientos constantes, dada una solución factible $\hat{S}$, existe una solución $\hat{R} = (R^1, \dots, R^m)$, con el mismo tiempo de ruta/tiempo de completación que $\hat{S}$, en la cual, para cada $i \in \{1,\dots,m\}$, $C(R^i)$ tiene la "propiedad de intervalos".*

Demostración. Sea $\hat{S}$ una solución factible del problema. Si $m \in \{1,2\}$ sabemos que existe una solución que cumple lo buscado, por lo que consideramos $m > 2$. Como la red de clientes y depósito es una línea, existe sólo una rama (la cual denotaremos por B). Ahora, considere la solución obtenida mediante el siguiente procedimiento (recordar que estamos considerando sólo soluciones que son recorridas de forma óptima, es decir, por orden de etiquetado de sus vértices):

Algorithm 3: LocalSorting

Input: $\hat{S}$ solución factible.
Output: $\hat{R}$ solución factible con la *"propiedad de intervalos"*.

1 $\hat{R} \leftarrow \hat{S}$
2 Ordenar las rutas de R de manera que $label(r^{l_i}_{|R^{l_i}|}) < label(r^{l_j}_{|R^{l_j}|})$ si $i < j$.
3 **for** $i = 1, \dots, m-1$ **do**
4 **for** $j = i+1, \dots, m$ **do**
5 $(R^{l_i}, R^{l_j}) \leftarrow$ Movement-1(R^{l_i}, R^{l_j}, B)
6 **return** $\hat{R}$

Veamos que $\hat{R}$ es la solución que buscamos. Para simplificar notación supondremos que las rutas de R estás ordenadas, de manera creciente, de acuerdo al label$(r^i_{|R^i|})$. Vamos a utilizar inducción para probar que para cualquier $i \in \{1,\dots,m-1\}$, luego de las líneas 4 y 5, R^i posee la *"propiedad de intervalos"*: vamos a ver que al terminar la primera iteración, $C(R^1)$ posee la *"propiedad de intervalos"*. Luego, como al terminar la i-ésima iteración (línea 3 del algoritmo) las iteraciones posteriores afectan a las rutas R^j con $j > i$, el argumento para el caso base sigue siendo válido cuando $i > 1$ con lo que habremos probado la proposición.

En efecto, suponga que al terminar $\exists u \notin C(R^1) : label(u) < label(r^1_{n_1})$. Como $\hat{R}$ originalmente era óptima, en particular era factible, esto es que $u \in R^k$ para algún $k \in \{2,\dots,m\}$. Notar que no puede ocurrir que $k = 1$ pues, Movement-1 sólo "saca" de $C(R^1)$ el vértice de mayor etiquetado, y como $label(u) < label(r^1_{n_1})$ entonces el etiquetado de u era menor que el vértice de mayor etiquetado de R^1 original (previo a la iteración), ya que las rutas de $\hat{R}$ están ordenadas de acuerdo al label$(\cdot)$. Además, si $k \neq 1$ entonces u no puede existir pues al momento de aplicar el Movement-1 al par (R^1, R^k, B), u debió pasar de R^k a R^1, con lo que $(\nexists u \in V : label(u) < label(r^1_{n_1}) \wedge u \notin C(R^1))$. Por lo tanto, $C(R^1)$ tiene la *"propiedad de intervalos"*.
Además, el tiempo de completación/tiempo de ruta de $\hat{R}$ no es mayor que el de $\hat{S}$, pues sólo se utiliza Movement-1 reiteradas veces, lo cual no aumenta el valor de la solución.

Cuando $i > 1$, usando los mimos argumentos se prueba que ($\nexists u \in V - \cup_{l=1}^{i-1} C(R^l) : \text{label}(u) < \text{label}(r^i_{n_i}) \wedge u \notin C(R^i)$). Es decir, $\forall u : \text{label}(u) < \text{label}(r^i_{n_i}), u \in \cup_{l=1}^{i} C(R^l)$. Por otro lado, como para $l < i$ las rutas R^l no fueron modificadas, sumado a que ellas poseen la *"propiedad de intervalos"*, se tiene que R^i también la posee.
$\therefore R$ es una solución factible donde que cada ruta cumple con la propiedad deseada. □

Corolario 2.6 *Para* (mTSPHT*) *en la línea, en la cual el depósito se encuentra en un extremo, con tiempos de procesamientos constantes, existe una solución óptima* $\hat{S}^*$ *tal que* $C(S^*_i)$ *tiene la "propiedad de intervalos".*

2.2. Depósito libre

Ahora consideramos el caso en que el *depósito* puede estar en cualquier vértice del camino, no necesariamente en los extremos. De manera similar al caso en que el depósito estaba en un extremo del camino, en un principio empezaremos por restringir nuestro estudio al problema con un único vehículo.

2.2.1. Depósito libre : $m = 1$

En este caso tenemos dos ramas, la rama derecha y la rama izquierda, etiquetadas respectivamente por B_1 y B_2 (Ver Figura 2.1).

Sea $G = (V, E)$ un camino que representa la red de *depósito-clientes*, donde el *depósito* se encuentra en un vértice no extremal. Observe que para una solución factible $\hat{S} = (S)$ con un vehículo, se tiene que:

$$\begin{aligned} T(S) &= \sum_{i=1}^{N} [t(s_{i-1}, s_i) + t(s_i)] \\ &= Nt_0 + \sum_{i=1}^{N} [t(s_{i-1}, s_i)] \\ &\geq Nt_0 + \text{mín}\{t(D, v^{B_1}_{|B_1|}), t(D, v^{B_2}_{|B_2|})\} + t(v^{B_1}_{|B_1|}, v^{B_2}_{|B_2|}), \end{aligned}$$

en donde $|B|$ representa la cantidad de clientes en la rama B, y la última desigualdad viene de la factibilidad de la solución, pues ambos extremos de cada rama deben ser visitados. Es decir, al menos se recorre la distancia del depósito al extremo más cercano y desde ahí se debe llegar al otro extremo.
Con lo anterior, existe una ruta con la cual se alcanza la cota inferior presentada: partiendo del depósito, escoger la rama con el extremo más cercano al punto de partida y visitar cada cliente por orden de etiquetado hasta terminar la rama, luego visitar los clientes de la otra rama de la misma manera. Es decir, por orden de etiquetado, hasta terminar la rama.

Ocurre algo similar con el tiempo de ruta:

$$\begin{aligned} Tr(S) &= t(s_N, D) + \sum_{i=1}^{N} [t(s_{i-1}, s_i) + t(s_i)] \\ &= t(s_N, D) + Nt_0 + \sum_{i=1}^{N} [t(s_{i-1}, s_i)] \\ &\geq Nt_0 + t(D, v^{B_1}_{|B_1|}) + t(D, v^{B_2}_{|B_2|}) + t(v^{B_1}_{|B_1|}, v^{B_2}_{|B_2|}) \end{aligned}$$

Aquí, la última desigualdad viene de que además de visitar los extremos se debe regresar al depósito, con lo que la distancia del depósito a ambos extremos se debe recorrer al menos una vez y, además, al llegar a uno de ellos se debe ir al otro. De ahí viene $t(v^{B_1}_{|B_1|}, v^{B_2}_{|B_2|})$. Notar que una de las formas de alcanzar esta cota es similar a cuando consideramos el tiempo de completación: partiendo del depósito visitamos a los clientes de la rama con el extremo más cercano y luego hacemos lo mismo con la siguiente. Pero, la diferencia con el caso anterior, es que en esta ocasión luego de realizar el recorrido se regresa directamente al depósito (en el caso anterior se terminaba luego de visitar ambos extremos).
Por lo tanto, supondremos conocido el orden óptimo de visita de los clientes. Es decir, si a un vehículo j le corresponde visitar un conjunto X de clientes, supondremos conocida su ruta S (sabemos el orden de visita óptimo de los X clientes).
Antes de seguir, introduciremos una extensión de la notación utilizada,

Observación Dada una ruta S, su secuencia de visita puede ser descrita de acuerdo a los clientes que visita en cada rama, lo que induce una notación natural:

$$S = (S^{B_1}, S^{B_2})$$

En donde B_l representa a la l-ésima rama y de manera que si s_j es el primer cliente visitado de la rama l, con n_l el número de clientes en la rama B_l visitado por S, entonces:

$$S^{B_l} = \{s_{j+i}\}_{i=0}^{n_l} = \{s_i^{B_l}\}_{i=0}^{n_l}$$

Con $s_0^{B_l} = D, \forall l \in \{1, 2\}$.

Esta notación no genera problemas cuando un conjunto de clientes es recorrido de manera óptima, ya que el reparador visita sólo una vez cada rama y lo hace en orden creciente de etiquetado. Además, como cada reparador parte desde el *depósito* y para cambiar de rama debe pasar por él, dada una ruta S, en la descripción por ramas de S podemos escribir $s_0^{B_l} = D$.

Al igual que cuando el *depósito* está en un extremo, analizaremos el problema de acuerdo a los valores de m.

2.2.2. Depósito libre : caso $m \geqslant 2$

Por simplicidad, denotaremos $n(S, B)$ al sub-índice del último cliente a visitar en la rama B de la ruta S.

Sea $\hat{S} = \{S^1, \ldots, S^m\}$ una solución del problema. Analizando la función objetivo para el tiempo de completación, de acuerdo al caso $m = 1$, se tiene:

$$\begin{aligned}
T(S^j) &= \min\{t(D, s^{j,B_1}_{n(S^j,B_1)}), t(D, s^{j,B_2}_{n(S^j,B_2)})\} + t(s^{j,B_1}_{n(S^j,B_1)|}, s^{j,B_2}_{n(S^j,B_2)}) + |C(S^j)|t_0 \\
&= \min\{t(D, s^{j,B_1}_{n(S^j,B_1)}), t(D, s^{j,B_2}_{n(S^j,B_2)})\} + t(D, s^{j,B_1}_{n(S^j,B_1)}) + t(D, s^{j,B_2}_{n(S^j,B_2)}) + n_j t_0 \\
&= \min\{t(D, s^{j,B_1}_{n(S^j,B_1)}), t(D, s^{j,B_2}_{n(S^j,B_2)})\} + t(D, s^{j,B_1}_{n(S^j,B_1)}) \\
&\quad + t(D, s^{j,B_2}_{n(S^j,B_2)}) + t_0(|C(S^j) \cap B_1| + |C(S^j) \cap B_2|) \\
&= \min\{t(D, s^{j,B_1}_{n(S^j,B_1)}), t(D, s^{j,B_2}_{n(S^j,B_2)})\} + t(D, s^{j,B_1}_{n(S^j,B_1)}) + t_0|C(S^j) \cap B_1| \\
&\quad + t(D, s^{j,B_2}_{n(S^j,B_2)}) + t_0|C(S^j) \cap B_2| \\
&= \min\{t(D, s^{j,B_1}_{n(S^j,B_1)}), t(D, s^{j,B_2}_{n(S^j,B_2)})\} + T(S^{j,B_1}) + T(S^{j,B_2})
\end{aligned}$$

Es decir, $T(S^j) = \min\{t(D, s^{j,B_1}_{n(S^j,B_1)}), t(D, s^{j,B_2}_{n(S^j,B_2)})\} + T(S^{j,B_1}) + T(S^{j,B_2})$.
En donde $T(S^{j,B_l})$ corresponde al tiempo de completación de la ruta S^j restringido a la rama l, y viene dado por:

$$T(S^{j,B_l}) = t(D, s^{j,B_l}_{n(S^j,B_1)}) + t_0|(C(S^j) \cap B_l)|$$

Y para el tiempo de ruta, haciendo el mismo análisis se obtiene algo similar,

$$\begin{aligned}
Tr(S^j) &= t(D, s^{j,B_1}_{n(S^j,B_1)}) + t(D, s^{j,B_2}_{n(S^j,B_2)}) + t(s^{j,B_1}_{n(S^j,B_1)}, s^{j,B_2}_{n(S^j,B_2)}) + n_j t_0 \\
&= Tr(S^{1,B_1}) + Tr(S^{1,B_2})
\end{aligned}$$

En donde $Tr(S^{j,B_l})$ es el tiempo de ruta de S^j restringido a la rama l.

Por lo que podemos restringir el análisis a las ramas del grafo, siguiendo este enfoque se presenta el siguiente resultado:

Proposición 2.7 *Para* (mTSPHT*) *en el camino, existe una solución óptima* $\hat{S} = (S^1, \ldots, S^m)$ *tal que para cada ruta* k *y para cada rama* B, $C(S^{k,B})$ *tiene la "propiedad de intervalos".*

Demostración. Sea $\hat{S} = (S^1, S^2, \ldots, S^m)$ una solución óptima. Para cada rama B considere el problema auxiliar en el cual $G_B = G[B]$, los costos y las distancias son las derivadas del problema original y considere la solución factible $\hat{S}^B = (S^{1,B}, \ldots, S^{m,B})$. Luego, aplicamos LocalSorting a $\hat{S}^B$ con lo que se obtienen nuevas rutas tal que su tiempo de completación y tiempo de ruta no aumentan (con respecto a la rutas de $\hat{S}^B$). La nueva solución de problema original, derivada a partir de $\hat{S}$ cuyas rutas en la rama B han sido modificadas y que seguiremos denotando por $\hat{S}$, es óptima. Además $\forall j \in \{1, \ldots, m\}$, $C(S^{j,B})$ posee la *"propiedad de intervalos"*.
Al realizar la operación para cada rama B, la solución que se obtiene es óptima y cumple que para cada rama B y $\forall j \in \{1, \ldots, m\}$ se tiene que $C(S^{j,B})$ posee la *"propiedad de intervalos"*. □

Tenemos la existencia de una solución óptima en la cual los clientes de cada ruta, restringidos a una rama, poseen la *"propiedad de intervalos"*, lo que aún no es suficiente para tener

un buen algoritmo. Ahora, encontraremos una solución óptima con mayor estructura, para ello definiremos un orden entre las rutas de una solución.

Definición 2.8 ***Orden*** $(\preccurlyeq)$
Sea $G = \{V, E\}$ *una red clientes-depósito en forma de camino. Sean* S^1 *y* S^2 *dos rutas del problema, diremos que* $S^1 \preccurlyeq S^2$ *si* $\forall B$ *rama*, $\forall u \in C(S^{1,B}), \forall v \in C(S^{2,B}) : label(u) < label(v)$.

Definición 2.9 ***Rutas concurrentes***
Diremos que dos rutas S^1 *y* S^2 *son concurrentes si* $\exists B$ *rama* $: C(S^{1,B}) \neq \emptyset \wedge C(S^{2,B}) \neq \emptyset$.

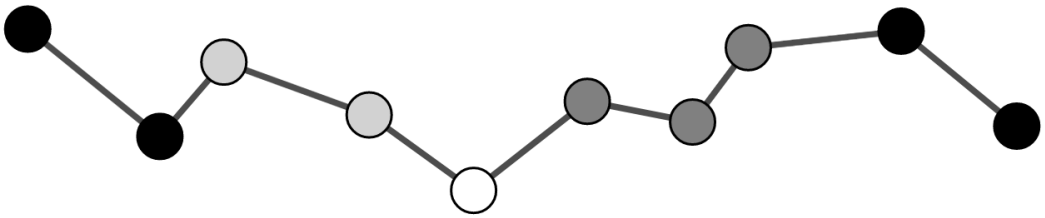


Figura 2.3: En la figura se muestran 3 rutas, una pintada de negro, una de gris oscuro y una de gris claro, el depósito esta pintado de blanco. Las ruta negra y la ruta gris claro son concurrentes, al igual que la ruta negra y la ruta gris oscuro. Sin embargo, las rutas gris oscuro y gris claro no son concurrentes. La ruta negra es mayor que la ruta gris oscuro y que la ruta gris claro.

Definición 2.10 ***Ruta estándar***
Diremos que una ruta S^j *es una ruta estándar si:*

1. *Es recorrida de manera óptima de acuerdo a la función objetivo.*
2. $C(S^{j,B})$ *tiene la "propiedad de intervalos" para cada rama* B.

Una solución en donde todas sus rutas son estándar, diremos que es una ***solución estándar***.

Proposición 2.11 *Para* (mTSPHT*) *en la línea existe una solución óptima tal que:*

1. *La solución es estándar.*
2. *Para cada par de rutas concurrentes* S^1, S^2 *se cumple que* $S^1 \preccurlyeq S^2 \vee S^2 \preccurlyeq S^1$

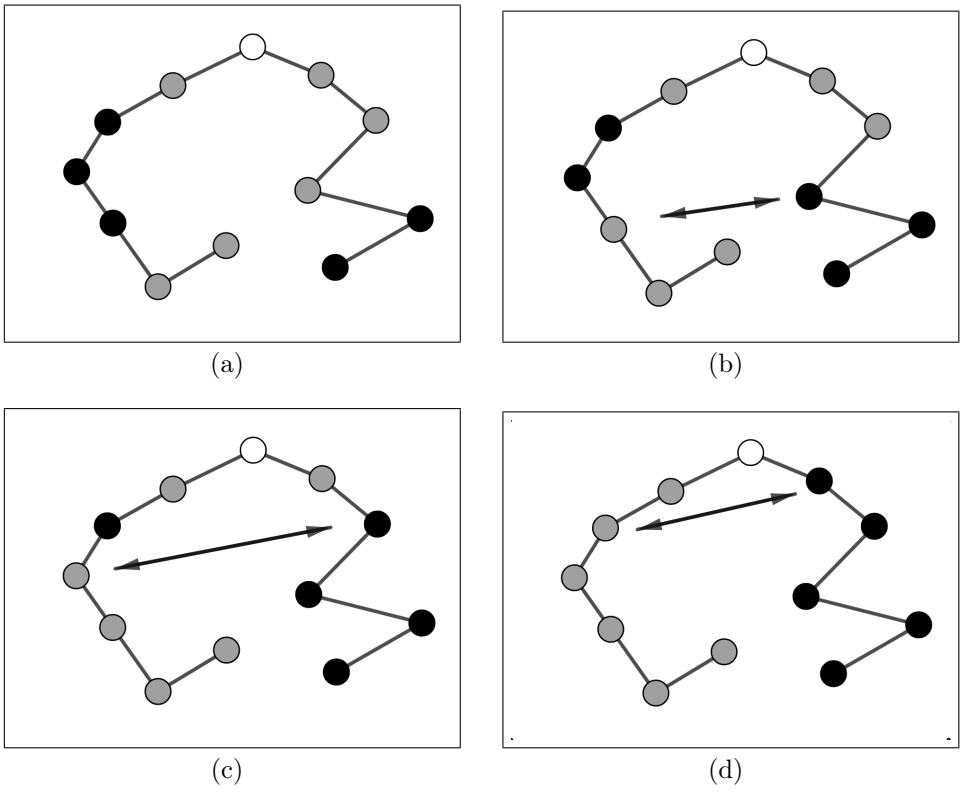
En la demostración de la proposición lo que haremos será probar 1), y luego utilizaremos el algoritmo MOVEMENT-2 para probar 2). MOVEMENT-2 recibe como input dos rutas estándares (S^1, S^2) y dos ramas (B_1, B_2), modifica las rutas haciendo que haya algún orden entre ellas (restringidas a las ramas B_1 y B_2). Suponga que en B_1, el último cliente de S^1 es más cercano al *depósito* que el primer cliente de S^2, entonces lo que hacemos es intercambiar los clientes de S^1 en B_1 por clientes en S^2 en B_2, empezando por los clientes más alejados

del *depósito* hasta que $C(S^{1,B_1}) = \emptyset$ o $C(S^{2,B_2}) = \emptyset$ (ver Figura 2.4).

Algorithm 4: MOVEMENT-2

Input: S^1, S^2 rutas estándar y B_1, B_2 ramas.
Output: R^1, R^2 con $R^1 \preccurlyeq R^2 \vee R^2 \preccurlyeq R^1$.

$R^1 \leftarrow S^1$
$R^2 \leftarrow S^2$
if $((R^1 \not\preccurlyeq R^2) \wedge (R^1 \not\succcurlyeq R^2))$ **then**
    if $(label(s^{1,B_1}_{n(S^1,B_1)}) \leqslant label(s^{2,B_1}_{n(S^2,B_1)}))$ **then**
        $\bar{i} \leftarrow 1, \bar{j} \leftarrow 2$
    else
        $\bar{i} \leftarrow 2, \bar{j} \leftarrow 1$
    while $((C(R^{\bar{i}}, B_1) \neq \emptyset) \wedge (C(R^{\bar{j}}, B_2) \neq \emptyset))$ **do**
        $(R^{\bar{i}}, R^{\bar{j}}) \leftarrow$ EXCHANGE$(R^{\bar{i}}, r^{\bar{i},B_1}_{n(R^{\bar{i}},B_1)}, R^{\bar{j}}, r^{\bar{j},B_2}_{n(R^{\bar{i}},B_2)})$
return (R^1, R^2)



(a) (b) (c) (d)

Figura 2.4: Ejemplo de la ejecución de MOVEMENT-2. Hay 2 rutas, una representada por el color negro y otra por el color gris, el depósito esta pintado de blanco. En (a) se muestra el estado inicial, mientras que (b), (c) muestran estados intermedios. Finalmente (d) muestra el output de MOVEMENT-2.

Proposición 2.12 MOVEMENT-2 *termina y aplicarlo a un par de rutas estándar concurrentes* (S^1, S^2) *y a las ramas* (B_1, B_2) *da como resultado* (R^1, R^2)*, que cumplen:*

1. $C(S^1) \cup C(S^2) = C(R^1) \cup C(R^2)$
2. *Restringidas a* (B_1, B_2)*,* R^1 *y* R^2 *no son concurrentes o bien* $R^1 \preccurlyeq R^2 \vee R^2 \preccurlyeq R^1$*.*

Además, si la red de clientes-depósito *es una línea, se cumple que:*

3. $\text{mín}_{i\in\{1,2\}} Tr(R^i) \leqslant \text{mín}_{i\in\{1,2\}} Tr(S^i)$
4. $\text{mín}_{i\in\{1,2\}} T(R^i) \leqslant \text{mín}_{i\in\{1,2\}} T(S^i)$

DEMOSTRACIÓN. Para fijar supondremos que

$$\text{label}(s^{1,B_1}_{n(S^1,B_1)}) \leqslant \text{label}(s^{2,B_1}_{n(S^2,B_1)})$$

Es decir, en la rama B_1 el último cliente visitado por la ruta S^1 está más cerca del depósito que el último cliente de la ruta S^2. Por otro lado, en la rama B_2 la situación es al revés: el último cliente visitado por S^2 está mas cerca del depósito que el primer cliente visitado por la ruta S^1. Esto se debe a que no hay un orden entre S^1 y S^2.

Veamos que el algoritmo termina. En cada iteración de la línea 8, R^1 cede un cliente de B_1 y R^2 cede un cliente de B_2. Es decir, en cada iteración se disminuye en uno el número de clientes de B_1 que visita S^1, lo mismo ocurre en B_2 con S^2, por lo que en algún momento se cumple que:

$$(C(R^1, B_1) = \emptyset) \vee (C(R^2, B_2) = \emptyset)$$

Con lo que se rompe el invariante y el algoritmo termina.
Para probar 1, basta notar que MOVEMENT-2 solo realiza intercambios de clientes entre ambas rutas, por lo que todos los clientes que eran servidos por S^1 o S^2 están cubiertos por R^1 o R^2. Además, se cumple que:

$$\begin{aligned} |C(R^1)| &= |C(S^1)| \\ |C(R^2)| &= |C(S^2)| \end{aligned}$$

Para probar 2, supongamos que ocurre que $C(R^1, B_1) = \emptyset$, de ser así en B_1 no hay clientes servidos por S^1 lo que puede ocurrir en dos situaciones:

- $C(R^2, B_2) = \emptyset$, en éste caso B_1 y B_2 no son concurrentes.
- $C(R^2, B_2) \neq \emptyset$, para éste caso basta notar que en la penúltima iteración se seguía cumpliendo el invariante y por ser la penúltima, sólo queda un cliente a servir en B_1 correspondiente a R^1, por lo que al cederlo se llega a que R^1 no tiene clientes en B_1. Sin embargo, en B_2, R^1 obtiene el último cliente que sirve R^2, por lo que se tiene que $R^2 \preccurlyeq R^1$.

Cuando ocurre $C(R^1, B_1) = \emptyset$, el análisis es el mismo pero intercambiando los roles de R^1 y R^2.

Finalmente, probaremos 3 y 4. Para ello, analizaremos el caso cuando la red es un camino. Recordemos que para una ruta S^j se cumple que

$$\begin{aligned} T(S^j) &= \min_{b\in\{1,2\}} t(s^{j,B_b}_{n(S^j,B_b)}, D) + T(S^{j,B_1}) + T(S^{j,B_2}) \\ &= 2t(D, s^{j,B_1}_{|B_1|}) + t(D, s^{j,B_2}_{|B_2|}) + t_0|C(S^j)| \\ Tr(S^j) &= Tr(S^1, B_1) + Tr(S^{1,B_2}) \\ &= 2t(D, s^{j,B_1}_{|B_1|}) + 2t(D, s^{j,B_2}_{|B_2|}) + t_0|C(S^j)| \end{aligned}$$

De dónde es fácil ver que tanto $Tr(\cdot)$ como $T(\cdot)$ dependen de la cantidad de clientes que la ruta S sirva y la distancia del último cliente visitado en cada rama al depósito. Ahora veamos que se cumple el siguiente invariante: el cliente más lejano del depósito de S^1 en B_1 es más cercano al depósito que el primer cliente visitado por S^2 en B_1, mientras que en B_2 ocurre lo mismo, salvo que se intercambian los roles de S^1 y S^2. Ahora, en la línea 9 del algoritmo en la cual se realiza la operación EXCHANGE, se cede el cliente más lejano de R^1 en la rama B_1 y, a su vez, R^2 cede su cliente más lejano correspondiente a B_2. Realizar este intercambio mantiene el invariante mientras que se cumpla la condición de la linea 8 del algoritmo, es decir, $(C(R^{\bar{i}}, B_1) \neq \emptyset) \wedge (C(R^{\bar{j}}, B_2) \neq \emptyset)$.

Observe que el intercambio anteriormente mencionado no aumenta la distancia del último cliente visitado por R^2 en B_1, sin embargo, disminuye la distancia del último cliente de R^1 en B_1, mientras se mantiene la cantidad de clientes visitados por ambas rutas. El mismo análisis es válido para la rama B_2 intercambiando los roles de R^1 y R^2. Como la distancia en B_1 (resp. B_2) del último cliente de R^1 (resp. R^2) al depósito no aumenta por lo que se cumple que

$$\begin{aligned} T(R^1) \leqslant T(S^1) \text{ (resp. } T(R^2) \leqslant T(S^2)) \\ Tr(R^1) \leqslant Tr(S^1) \text{ (resp. } Tr(R^2) \leqslant Tr(S^2)) \end{aligned}$$

□

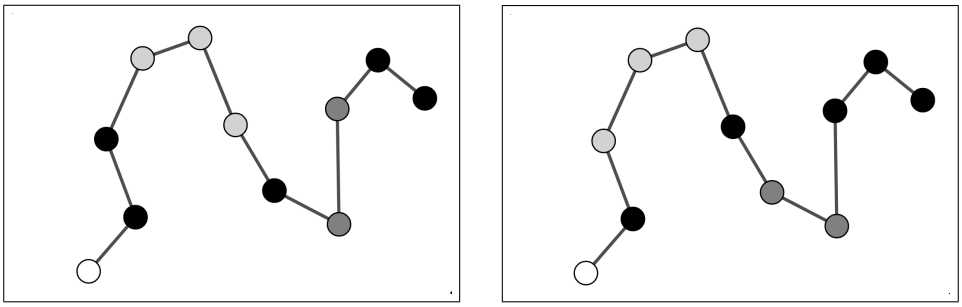
También necesitaremos la siguiente operación, SHIFT, la cual dada una solución factible $\hat{S}$, un índice de ruta j y una rama B, tal que $\forall k \in \{1, \ldots, m\} - j$, $S^{k,B}$ tiene la propiedad de intervalos. SHIFT hace que todas las rutas de la solución $\hat{S}$ que estén dentro de S^j en la rama B, es decir, que estén entre dos clientes que pertenezcan a S^j, se acerquen un cliente al *depósito*. Más específicamente, si S^k está dentro de S^j y $C(S^k) = \{v^B_l, v^B_{l+1}, \ldots, v^B_{l+r}\}$ antes de aplicar el algoritmo, después de aplicarlo, se tiene que $C(S^k) = \{v^B_{l-1}, v^B_l, \ldots, v^B_{l+r-1}\}$ (ver Figura 2.5).

Algorithm 5: SHIFT

Input: Solución factible $\hat{S}$, índice de ruta j y rama B. $C(S^{k,B})$ tiene propiedad de intervalos $\forall k \in \{1,\ldots,m\} - j$.

Output: $\hat{R}$ solución factible.

$\hat{R} \leftarrow \emptyset$
for $(i \in \{1,\ldots,m\} - j)$ **do**
  if $(\text{label}(s_1^{i,B}) > \text{label}((s_1^{j,B}))) \wedge (\text{label}(s_{n(S^i,b)}^{i,B}) < \text{label}((s_{n(S^j,B)}^{j,B})))$ **then**
    for $(v \in C(S^i, B))$ **do**
      $C(R^i) \leftarrow u_{\text{label}(v)-1}^{i,B}$
  else
    $R^i \leftarrow S^i$
$r \leftarrow$**find**$(\min r > 0 : [\text{label}(s_{r+1}^{j,B}) \neq \text{label}(s_r^{j,B}) + 1] \wedge [r < |C(S^{j,B}|])$
$r' \leftarrow$**find**$(\max r > 0 : [\text{label}(s_{r+1}^{j,B}) \neq \text{label}(s_r^{j,B}) + 1] \wedge [r < |C(S^{j,B}|])$
if $(r \neq +\infty)$ **then**
  for $(v \in C(S^j, B))$ **do**
    if $([\text{label}(v) > \text{label}(s_r^{j,B})] \wedge [\text{label}(v) < \text{label}(s_{r'+1}^{j,B})])$ **then**
      $C(R^j) \leftarrow u_{\text{label}(v)-1}^{j,B}$
    else
      $C(R^j) \leftarrow C(R^j) + v$
  $C(R^j) \leftarrow C(R^j) + v_{\text{label}(s_{r'+1}^{j,B})-1}^{B} - s_r^{j,B}$
else
  $C(R^j) \leftarrow C(S^j)$
return $\hat{R}$



(a) (b)

Figura 2.5: Ejemplo de la ejecución de SHIFT. Se muestra sólo la rama B considerada, el índice de ruta entregado es el correspondiente a la ruta negra y el depósito esta pintado de blanco.(a) muestra el estado inicial y (b) muestra el estado luego de la ejecución del algoritmo.

Demostración Prop 2.11. Sea $\hat{S}$ una solución óptima del problema, aplicaremos el siguiente algoritmo a $\hat{S}$:

Algorithm 6: GlobalSorting(Path-Version)

Input: Solución factible $\hat{S}$
Output: $\hat{R}$ solución factible

```
1  R̂ ← S
2  for (b: b rama) do
3      R^b ← LocalSorting(R^b, b)
4  for i = 1...m do
5      for j = i...m do
6          (R^i, R^j) ← Movement-2(R^i, R^j, B_1, B_2)
7          for l ∈ {1,2} do
8              while (∃n > 0 : label(s^{i,B_l}_{n+1}) ≠ label(s^{i,B_l}_n) + 1) do
9                  R̂ ← Shift(R̂, i, B_l)
10             while (∃n > 0 : label(s^{j,B_l}_{n+1}) ≠ label(s^{j,B_l}_n) + 1) do
11                 R̂ ← Shift(R̂, j, B_l)
12 return R̂
```

Primero veamos que el algoritmo termina, para ello basta ver que en las líneas 8 y 10 a lo más toma tiempo N, pues al cabo de a lo más N operaciones de Shift$(\hat{R}, i, B)$, no puede existir una ruta que este dentro de R^i.
Ahora veamos que $\hat{R}$ es la solución que buscamos. Observe que en las líneas 2-3 ni el tiempo de completación ni el tiempo de ruta de $\hat{R}$ no aumentan, pues son múltiples aplicaciones de LocalSorting, por lo que al terminar las iteraciones correspondientes se tiene que hasta ese momento $\hat{R}$ es una solución óptima en donde $C(R_i^{B_b})$ es un intervalo de $\mathbb{N}$, con $b \in \{1,2\}$ y $i \in \{1,\ldots,m\}$, luego sólo queda lidiar con el orden parcial.

Entre las líneas 4-10 se realizan las modificaciones a R para garantizar el orden parcial de la solución y para ello sólo basta analizar la primera iteración, esto es porque la operación Shift sólo afecta a las rutas que están "entre" dos clientes de la ruta j en la rama B_b, pero sin alterar la estructura de la ruta. Es decir, si una ruta i servía a clientes consecutivos ($i \neq j$), luego de aplicar Shift, i sigue sirviendo a clientes consecutivos (directo de la definición de Shift).
Recordemos que en este momento $C(R_i^{B_b})$ tiene la *"propiedad de intervalos"*, con $b \in \{1,2\}$ y $i \in \{1,\ldots,m\}$. Si $R^1 \preccurlyeq R^2$ o $R^2 \preccurlyeq R^1$ se continua a la siguiente iteración, en caso contrario:

$$\text{label}(r^{1,B_b}_{n(R^1,B_b)}) < \text{label}(r^{2,B_b}_{n(R^2,B_b)}) \text{ y } \text{label}(r^{2,B_{b'}}_{n(R^2,B_{b'})}) < \text{label}(r^{1,B_{b'}}_{n(R^1,B_{b'})})$$

con $b, b' \in \{1,2\} : b \neq b'$, para fijar ideas suponga que $b = 1$ y $b' = 2$. Luego al aplicar Movement-2 ocurre que alguna de las dos rutas que fueron entregadas como input, R^1 o R^2, no tiene clientes por visitar en alguna de las dos ramas con lo que se cumple $R^1 \preccurlyeq R^2$ o $R^2 \preccurlyeq R^1$. Sin embargo, no se puede asegurar que los clientes que visiten sean consecutivos.
Ahora, sin pérdida de generalidad suponga que $C(R^1, B_2) = \emptyset$
En este caso puede pasar que $C(R^2, B_1) = \emptyset$ o bien $C(R^2, B_1) \neq \emptyset$. En el primer caso no hay

que probar nada pues no se ha modificado ninguna ruta aparte de R^1 y R^2, mientras que en el otro caso ocurre que sólo clientes de R^2 en B_1 han dejado de ser un conjunto de clientes consecutivos, a lo que las líneas 7-11 del algoritmo entran en juego; Luego si existe algún cliente servido por R^2 en B_1 tal que el cliente que le sigue, v, no corresponda al cliente que le siga en la línea (contando desde el depósito al cliente final de B_2), se aplicará SHIFT de manera de corregir este evento. Observe que en estas condiciones SHIFT no aumenta el valor y sin embargo reordena la solución de manera que recupere la propiedad de que los clientes son visitados de manera consecutiva en cada rama por cada ruta. En efecto, la reordenación de los clientes es directa de la definición de SHIFT y que SHIFT no aumente el tiempo de completación ni el tiempo de ruta de la solución viene de que las rutas que están dentro de R^2 mantienen la cantidad de clientes que están visitando y a su vez, disminuye la distancia entre el último cliente a visitar (en la rama B_1) y el depósito, mientras que para R^2 ni la cantidad de clientes que visita ni la distancia entre el depósito y el último cliente que visita en B_1 cambian, manteniendo así el valor para la ruta R^2.
El análisis es el mismo en el caso que $C(R^1, B_2) = \emptyset$, intercambiando los roles de R^1 por R^2 y de B_2 por B_1.
Entonces, después de la primera iteración el valor de $\hat{R}$ no aumenta. Además, $R^1 \preccurlyeq R^2$ o bien $R^2 \preccurlyeq R^1$.
$\therefore$ La solución $\hat{R}$ entregada por GLOBALSORTING(PATH-VERSION) es una solución óptima estándar y existe un orden parcial entre rutas. □

Habiendo caracterizado la estructura de una solución óptima para el problema, es posible hacer la siguiente afirmación

Corolario 2.13 (mTSPHT*) *en el camino se puede resolver en tiempo polinomial.*

DEMOSTRACIÓN. En efecto, por la Proposición 2.11 existe una solución óptima $\hat{S}$ para (mTSPHT*) en el camino tal que:

1. Para toda rama B y $k \in \{1, \ldots, m\}$, $C(S^{k,B})$ tiene la *"propiedad de intervalos"*.
2. Para cada par de rutas concurrentes S^1, S^2 se cumple que $S^1 \preccurlyeq S^2 \vee S^2 \preccurlyeq S^1$. Es decir, $\hat{S}$ tiene un orden parcial.

Por lo que el problema se reduce a encontrar el *intervalo* de clientes que cada ruta debe visitar. Ahora considere la siguiente función:

$$F(L,R,m) = \begin{cases} p(v_L^{B_2}, v_R^{B_1}) + (L+R)t_0 & m = 1 \\ \min\limits_{i=0\ldots L} \min\limits_{j=0\ldots R} \left\{ \max(F(L-i, R-j, m-1), p(v_L^{B_2}, v_R^{B_1}) + (i+j)t_0) \right\} & m > 1 \end{cases}$$

En donde:

- $L = |B_2|$
- $R = |B_1|$
- m es el número de vehículos

- p es una función que depende de la función objetivo que se considera, si se minimiza el tiempo de ruta:
$$p(v_L^{B_2}, v_R^{B_1}) = 2t(v_L^{B_2}, D) + 2t(v_R^{B_1}, D)$$
Si lo que se minimiza es el tiempo de completación:
$$p(v_L^{B_2}, v_R^{B_1}) = 2t(v_L^{B_2}, D) + 2t(v_R^{B_1}, D) - \text{máx}(t(v_L^{B_2}, D), t(D, v_R^{B_1}))$$

Luego $F(|B_2|, |B_1|, m)$ encuentra la solución óptima en $O(N^4 m)$. En efecto, hay $O(N^2 m)$ estados (posibles valores de L, R, m) y en cada estado se busca el mínimo entre $O(N^2)$ valores, por lo tanto $F(|B_2|, |B_1|, m)$ se puede calcular en tiempo $O(N^4 m)$.
$\therefore$ (mTSPHT*) en el camino se puede resolver en tiempo polinomial. □

2.3. mTSPHT* en el ciclo

Ahora se estudiará cuando la red de *clientes-depósito* forma un ciclo. Observe que esta vez, dados dos clientes u, v, tenemos dos alternativas para ir de u a v a través del ciclo. Una es recorrer el ciclo en sentido positivo (a favor de las manillas del reloj) y en sentido negativo (en contra de las manillas del reloj). Como uno de nuestros parámetros a minimizar es el tiempo de viaje, vamos a considerar el tiempo de viaje $t(u, v)$ como el tiempo que toma ir de u a v tomando el camino más corto (entre el camino positivo y el negativo). Luego, es natural definir los siguientes conjuntos:

Definición 2.14 *Definiremos $C_+(S^i)$ (resp. $C_-(S^i)$) como el conjunto de clientes $v \in C(S^i)$ de manera que si $v = s_j^i$ entonces el camino más corto entre s_{j-1}^i y v es positivo (resp. negativo).*

Al igual que en el caso de la línea vamos a etiquetar los vértices de acuerdo a su lejanía del depósito. Sin embargo, esta vez cada cliente tiene dos etiquetas, los denotaremos v_j^+ y v_j^- (al *j-ésimo* cliente del ciclo recorrido a favor y en contra de las manillas del reloj respectivamente. Ver Figura 2.5).

Como en secciones anteriores lo primero que haremos será estudiar el orden de visita de los clientes de un vehículo dado el conjunto de clientes a visitar. Además, vamos a extender la idea de label$(\cdot)$.

Definición 2.15 label$_+(\cdot)$/ label$_-(\cdot)$
Para una red depósito-cliente etiquetada y un vértice v, definimos label$_+(v)$/ label$_-(v)$ *como la etiqueta de que tiene v con respecto al etiquetado en sentido positivo/negativo. Vamos a denotar $n(S, -)$ al último cliente en S que es visitado en sentido negativo.*

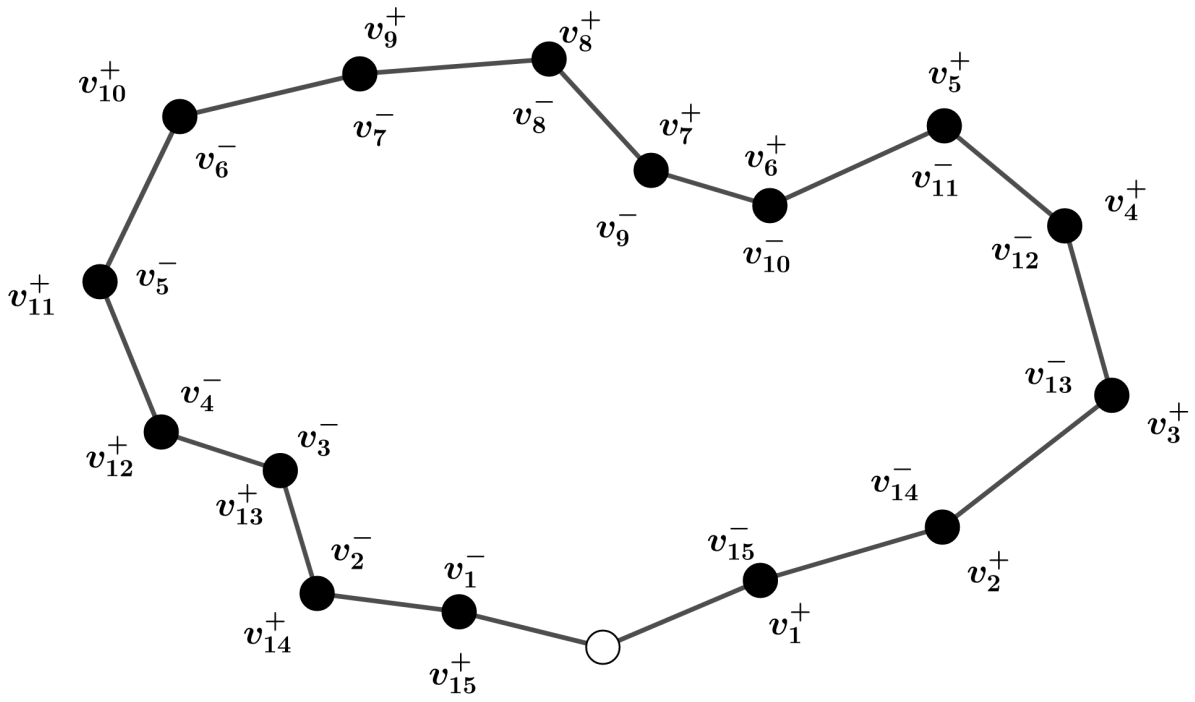


Figura 2.6: Ejemplo etiquetado de (mTSPHT*) en un ciclo. El *depósito* esta coloreado de blanco y los clientes de negro. Además, se muestran las dos etiquetas correspondientes a los clientes.

Proposición 2.16 *Sea S una ruta de* (mTSPHT*) *en el ciclo. Existe una ruta R, con $C(R) = C(S)$, tal que no se "salta" clientes. Es decir, si $R = r_0, r_1, \dots r_{n_R}$ y existe un cliente $v \in C(R)$ en el camino entre r_i y r_{i+1}, entonces $v = r_j$ con $j < i$. Y además,*

$$Tr(R) < Tr(S)$$
$$T(R) < T(S)$$

Demostración. En efecto, considere una ruta $S = s_0 s_1 \dots s_{n_S} D$. Como definimos el tiempo de viaje $t(u, v)$ entre dos clientes u, v como el camino más corto entre ellos, entonces t cumple la desigualdad triangular. Ahora, suponga que $v^* = s_j$ es el cliente de menor índice i, tal que v^* se encuentra en el camino entre s_i y s_{i+1}, y $j > i$. Es decir, el primer cliente que la ruta S se salta. Sea $R = r_0, r_1 \dots, r_{n_S}$ (recuerde que $n_S = n_R$), con:

$$r_l = \begin{cases} s_l, & l \leqslant i \\ v^*, & l = i+1 \\ s_{l-1}, & i+1 < l \leqslant j \\ s_l, & l > j \end{cases}$$

Entonces se cumple que:

$$\begin{aligned}
Tr(S) &= n_S t_0 + \sum_{l=0}^{n_S+1} [t(s_l, s_{l+1})] \\
&\geqslant n_S t_0 + t(s_i, v^*) + t(v^*, s_{i+1}) + t(s_{j-1}, s_{j+1}) \\
&\quad + \sum_{l=0}^{i-1} [t(s_l, s_{l+1})] + \sum_{l=i+1}^{j-1} [t(s_l, s_{l+1})] + \sum_{l=j+1}^{n_S+1} [t(s_l, s_{l+1})] \\
&= n_S t_0 + t(r_i, r_{i+1}) + t(r_{i+1}, r_{i+2}) + t(r_j, r_{j+1}) \\
&\quad + \sum_{l=0}^{i-1} [t(r_l, r_{l+1})] + \sum_{l=i+2}^{j} [t(r_l, r_{l+1})] + \sum_{l=j+1}^{n_S+1} [t(r_l, r_{l+1})] \\
&= n_S t_0 + \sum_{l=0}^{n_R+1} [t(r_l, r_{l+1})] \\
&= Tr(R)
\end{aligned}$$

La primera desigualdad viene de que por un lado tenemos que $t(s_i, v^*)+t(v^*, s_{i+1}) = t(s_i, s_{i+1})$ pues v^* está en el camino entre s_i y s_{i+1}, y por otro lado, $t(s_{j-1}, v^*)+t(v^*, s_{j+1}) \leqslant t(s_{j-1}, s_{j+11})$ por desigualdad triangular.
Si todavía existe un cliente v^* que R se salte, entonces se vuelve a repetir el argumento. Note que cada vez se reduce en al menos uno el número de clientes que R se salta, pues por la elección de v^* que estamos haciendo, al terminar el "arreglo" de la ruta R, tenemos que entre $r_0 r_1 \dots v^* = s_j$ no existe $u \in C(R)$ tal que u este en el camino de $r_i, r - i + 1$ para todo $i \in \{0, \dots, j-1\}$, por lo que si sigue existiendo un cliente u^* que genere problemas, entonces $u^* = r_k$ con $k > j$. Por lo que este procedimiento a lo más se realiza n_S veces. Luego, tenemos una ruta R que no se salta cliente, y cumple que $Tr(R) \leqslant Tr(S)$. Para el tiempo de completación, la demostración es análoga, salvo que las rutas en vez de ser $S = s_0 s_1 \dots s_{n_S} D$ y $R = r_0 r_1 \dots r_{n_S} D$, estas son $S = s_0 s_1 \dots s_{n_S}$ y $R = r_0 r_1 \dots r_{n_S}$, en donde R esta definida de la misma manera a partir de S. □

Por la propiedad anterior, consideraremos **solo rutas que no se salten clientes**. Vamos a denotar por $s^+_{n(S,+)}$ al último cliente de $C_+(S)$ en ser visitado y por $s^-_{n(S,-)}$ al último cliente de $C_-(S)$ en ser visitado.

Corolario 2.17 *Sea S una ruta, si conocemos los conjuntos $C_+(S)$ y $C_-(S)$, entonces saber el orden en que los clientes de S son recorridos de manera de minimizar su tiempo de completación y tiempo de ruta.*

Demostración. En efecto, por la propiedad anterior sabemos que la ruta S no se salta clientes, por lo que si $C_+(S) = \emptyset$, entonces todos los clientes son visitados, en sentido negativo, sin saltarse ninguno. Es decir, los clientes son visitados en orden creciente de su label$_-(\cdot)$ (no existe otra opción). Cuando $C_-(S) = \emptyset$ ocurre lo mismo, pero los clientes son visitados en sentido positivo.
Ahora, cuando $C_-(S) \neq \emptyset \wedge C_+(S) \neq \emptyset$, como la ruta no salta clientes, se tiene que primero se visitan o todos los clientes de $C_+(S)$ y luego los de $C_-(S)$, o al revés, se visitan todos los

clientes de $C_-(S)$ y luego los de $C_+(S)$. Para el tiempo de completación, hay dos alternativas: la primera es que el camino desde el *depósito* a $s^+_{n(S,+)}$ es recorrido dos veces, y la segunda es que el camino desde el *depósito* a $s^-_{n(S,-)}$ es recorrido dos veces, esto es debido a que S no se salta clientes. Por lo que la manera óptima de recorrer los clientes es recorrer dos veces el camino cuya distancia entre D y el último cliente, ya sea en sentido negativo o en sentido positivo, sea la menor. Es decir, si $t(D, s^+_{n(S,+)}) < t(D, s^-_{n(S,-)})$ se parte recorriendo en sentido positivo y luego en sentido negativo, de lo contrario se parte recorriendo en sentido negativo y luego en sentido positivo.
Para el tiempo de ruta, no importa si se parte recorriendo en sentido negativo o positivo, el camino entre D y $s^-_{n(S,-)}$ y el camino entre D y $s^+_{n(S,+)}$ son recorridos dos veces. Pero para ser consistentes con el tiempo de completación, vamos a elegir recorrer partir por camino que cumpla que su distancia entre D y el último cliente, recorrido en su sentido correspondiente, sea la menor. Es decir, si $t(D, s^+_{n(S,+)}) < t(D, s^-_{n(S,-)})$ se parte recorriendo en sentido positivo y luego en sentido negativo, de lo contrario se parte recorriendo en sentido negativo y luego en sentido positivo. □

Por el corolario anterior, además de **considerar sólo rutas que no se saltan clientes**, además vamos a dar por **conocido el orden en que los clientes de una ruta son recorridos, si es que sabemos los clientes que son visitados en sentido positivo y en sentido negativo**. Antes de dar el resultado principal de esta sección, vamos a probar que, salvo ciertas ocasiones, existe una solución óptima en la cual una de las aristas del ciclo nunca es transitada por ninguna ruta.

Proposición 2.18 *Para toda solución factible $\hat{S}$, existe una solución factible $\hat{R}$ tal que $\forall k \in \{1, \dots, m\}$ si $v \in C_+(R^k)$ con $v = v_j^-$ entonces $\forall i \in \{1 \dots m\},\ j >$ label$_-(r^{-,i}_{n(R^i,-)})$. Es decir, para cualquier vértice visitado por alguna ruta en sentido positivo, su* label$_-$ *es mayor al* label$_-$ *de cualquier vértice visitado en sentido negativo. A ésta propiedad la que denotaremos por (P). Además se cumple que tanto los tiempos de ruta y de completación de $\hat{R}$ no son mayores que los de $\hat{S}$.*

Demostración. Sea $\hat{S}$ una solución factible, suponga que $\exists k \in \{1, \dots, m\}$ y $v \in C_+(R^k)$ con $v = v_j^-$ tal que $\exists i \in \{1 \dots m\},\ j <$ label$_-(r^{-,i}_{n(R^i,-)})$. Considere el siguiente procedimiento:

Algorithm 7: CrossSorting

Input: Solución factible $\hat{S}$
Output: Solución factible $\hat{R}$ que cumple la propiedad (P)

1 $R \leftarrow S$
2 **while** $(\exists v : v$ *hace que no se cumpla* $(P))$ **do**
3 Sea v^* el cliente que genera problemas que posee el etiquetado más grande en dirección positiva. Sea i^* tal que $v^* \in R^{i^*}$
4 Sea j^* el índice de la ruta cuyo último cliente visitado en dirección negativa es el más alejado del depósito(considerando la dirección negativa).
5 $C(R^{i^*,+}) \leftarrow C(R^{i^*,+}) - v^* + r^{-,j^*}_{n(R^{j^*},-)}$
6 $C(R^{j^*,-}) \leftarrow C(R^{j^*,-}) + v^* - r^{-,j^*}_{n(R^{j^*},-)}$
7 **return** R

Si el algoritmo termina, sólo nos falta probar que $T(\hat{R}), Tr(\hat{R}) \leqslant T(\hat{S}), Tr(\hat{S})$, lo cual viene del hecho que $\forall i \in \{1, \ldots, m\}$, la cantidad de clientes que sirve S^i es la misma que sirve R^i y además se tiene que $t(D, r^{+,i}_{n(R^i,+)}) \leqslant t(D, s^{+,i}_{n(S^i,+)})$ y $t(D, r^{-,i}_{n(R^i,-)}) \leqslant t(D, s^{-,i}_{n(S^i,-)})$. Por lo que R es la solución que estamos buscando.
Ahora sólo falta ver que el algoritmo termina, en efecto, el intercambiar el cliente v^* por el cliente más alejado visitado en sentido negativo hace que en la nueva solución, el etiquetado del cliente más alejado del *depósito* visitado en sentido negativo, digamos u, sea menor por al menos uno, además en el intercambio no se agregan clientes a recorrer (en sentido positivo) al camino negativo entre D y u. □

Corolario 2.19 *Si $\hat{S}$ es una solución en la cual sus rutas no saltan clientes y cumple la propiedad (P) entonces, cuando se minimiza el tiempo de completación, existe una arista del ciclo que no es transitada por ninguna ruta de $\hat{S}$. Mientras que para el tiempo de ruta, si existen clientes visitados tanto en sentido positivo como en sentido negativo, existe una arista del ciclo que no es transitada por ninguna ruta de $\hat{S}$.*

Demostración. Sea $\hat{S}$ una solución cuyas rutas no saltan clientes y que posee la propiedad (P). Supongamos primero que existen clientes que son recorridos en sentido negativo y otros que son recorridos en sentido positivo.
Observe que como $\hat{S}$ es una solución, $\forall v \in V, \exists k : v \in C(S^k)$. Por otro lado, $\hat{S}$ tiene la propiedad (P), entonces

$$\min_{k=1,\ldots,m} \text{label}_-(s^{+,k}_{n(S^k,+)}) > \max_{k=1,\ldots,m} \text{label}_-(s^{-,k}_{n(S^k,-)})$$

Por lo que el vértice de $\text{label}_- = j^* = \min_{k=1,\ldots,m} \text{label}_-(s^{+,k}_{n(S^k,+)}) - 1$, debe ser recorrido en sentido negativo, luego la arista $\{v^-_{j^*}, v^-_{j^*-1}\}$ nunca es recorrida.
En el caso en que no existan clientes recorridos en sentido positivo, cuando se minimiza el tiempo de completación con N clientes, la arista $\{v^-_N, D\}$ nunca es visitada. Cuando no existen clientes recorridos en sentido negativo, cuando se minimiza el tiempo de completación con N clientes, la arista $\{v^+_N, D\}$ nunca es visitada. □

Ahora vamos a ver que es lo que sucede cuando se minimiza el tiempo de ruta, cuando en una solución factible $\hat{S}$, ningún cliente es visitado en sentido negativo o ningún cliente es visitado en sentido positivo.

Proposición 2.20 *Sea $\hat{S}$ una solución factible cuando se minimiza el tiempo de ruta, que cumple visita a todos sus clientes en sentido positivo (resp. negativo). Entonces existe una solución $\hat{R}$ tal que:*

1. *$Tr(\hat{R}) \leqslant Tr(\hat{S})$*
2. *Para todo $k \in \{1, \ldots, m\}$, $C_+(R^k)$ (resp. $C_-(R^k)$) tiene la propiedad de intervalos*

Demostración. Partiremos por definir la suma de las aristas totales del ciclo, sea N la cantidad de clientes de la instancia,

$$t_C = \frac{\sum_{i=0}^{N} t(v^+_i, v^+_{i+1})}{2}$$

Sea j^* el label$_+$ más pequeño del cliente que cumple que $\sum_{i=0}^{j^*} t(v_i^+, v_{i+1}^+) \leqslant t_C$. Note que para cualquier ruta S^k cuyo último cliente a visitar tenga un label$_+$ mayor que j^* se va a devolver al *depósito* en sentido positivo (pues es el camino mas corto), luego su tiempo de ruta es $2t_C + |S^k|t_0$. Por otro lado, para cualquier ruta S^k cuyo último cliente a visitar tenga un label$_+$ menor que j^* se va a devolver al *depósito* en sentido negativo, por lo que su tiempo de completación será de $2d(s^k_{|S^k|}, D) + |S^k|t_0$. De lo que podemos ver que para calcular el tiempo de ruta de la solución sólo interesa el último cliente a visitar y número de clientes a visitar.
Ahora, utilizaremos LocalSorting sobre $\hat{S}$, considerando la rama $B = Dv_1^+ \dots v_N^+$. Observe que LocalSorting no aumenta el label de los últimos clientes a visitar por las rutas, luego el utilizarlo sobre $\hat{S}$ no aumenta el label$_+$ de los últimos clientes de cada ruta y la solución $\hat{R}$ entregada tiene un tiempo de ruta menor o igual a $\hat{S}$, con lo que se cumple el punto 1. Y además, posee la "propiedad de intervalos", con lo que se tiene el punto 2.
El caso cuando se visita a todos los vértices en sentido negativo, la demostración es análoga.
□

Con la propiedad anterior, podemos encontrar una función que nos permita encontrar la solución óptima en caso que de alguna manera sepamos que todos los clientes son visitados en sentido positivo o sentido negativo, al igual que en el caso del camino, vamos a ver que "intervalo" de clientes le corresponde a cada ruta:

$$F_+(N,m) = \begin{cases} Nt_0 + t(v_N^+, D) + \sum_{i=0}^{N} t(v_i^+, v_{i+1}^+) & m = 1 \\ \min\limits_{j=1\dots N} (F_+(N-j, m-1), (N-j)t_0 + t(v_N^+, D) + \sum_{i=0}^{N} t(v_i^+, v_{i+1}^+)) & m > 1 \end{cases}$$

Luego, utilizando $F_+(N,m)$ cuando todos los clientes son visitados en sentido positivo, con N el número de clientes y m el número de vehículos, nos entrega la solución óptima.
Cuando, los clientes son visitados en sentido negativo, vamos definir $F_-(N,m)$ de manera análoga, simplemente utilizando los label$_-$ en vez de los label$_+$. Observe que esta función tiene a lo más $O(Nm)$ estados (posibles valores de m y N) y en cada estado busca el mínimo entre $O(N)$ valores, por lo que la complejidad de la función es de $O(mN^2)$.

Utilizando el Cor. 2.19, la Prop. 2.20 junto con la función $F_+(N,m)$ y $F_-(N,m)$, podemos resolver (mTSPHT*) en el ciclo "adivinando" los clientes visitados en sentido negativo y en sentido positivo, lo cual se muestra en el siguiente teorema.

Teorema 2.21 *Es posible resolver* (mTSPHT*) *en el ciclo en tiempo polinomial.*

Demostración. Considere una instancia de (mTSPHT*) en el ciclo. Considere el siguiente algoritmo:

Algorithm 8: Cycle-Solver

Input: Red de *clientes-depósito* G, tiempos de viaje, tiempo de visita t_0 y función objetivo.

Output: Solución óptima del problema

1 Etiquete todos los vértices del ciclo en dirección negativa y dirección positiva

2 **for** $(i \in \{0 \dots N\})$ **do**

3 Genere una instancia del problema en el camino de la siguiente manera:

4 La rama derecha constara de los vértices etiquetados desde 1 a i en dirección positiva, el tiempo de viaje entre dos vértices consecutivos de esta rama viene dada por el tiempo de viaje en sentido positivo en el ciclo.

5 La rama izquierda constara de los vértices etiquetados desde 1 a $N - i$ en dirección negativa, el tiempo de viaje entre dos vértices consecutivos de esta rama viene dada por el tiempo de viaje en sentido negativo en el ciclo.

6 En el caso, de tiempo de ruta, si alguna de las ramas es vacía, utilizaremos la función F_+ o F_- para resolver el problema. En caso contrario, resuelva el problema en el camino y guarde la solución.

7 Devuelva la solución de menor valor entre las $N + 1$ soluciones que se calcularon.

Veamos que la solución entregada por el algoritmo en realmente la solución de nuestro problema. Observe que en la solución óptima de problema se cumple una de las siguientes afirmaciones: no existen clientes recorridos en sentido negativo, no existen clientes recorridos en sentido positivo o existen clientes recorridos tanto en sentido positivo como en sentido negativo. Si estamos minimizando el tiempo de ruta, y la solución óptima no tiene clientes en sentido positivo o en sentido negativo, sabemos como resolverlo (mediante F_- o F_+ respectivamente). Si no estamos minimizando el tiempo de ruta, por el Cor. 2.19 sabemos que existe una solución óptima $\hat{S}$ tal que una arista en el ciclo no es nunca transitada, por lo que el algoritmo es capaz de encontrar esta solución óptima al probar la mejor solución posible al no transitar una arista. Por lo tanto, el algoritmo siempre encuentra una solución óptima.
El algoritmo complejidad de tiempo $O(mN^5)$, pues resolver cada sub-problema de (mTSPHT*) en el camino toma $O(mN^4)$, recuerde que la complejidad de F_+ y de F_- es $O(mN^2)$ por lo que su complejidad igual es de $O(mN^2)$, y se realizan N iteraciones.
Observe que estamos considerando solo el tiempo de solución de los sub-problemas debido a que la generación de estos puede ser realizada en tiempo $O(N^2)$.
$\therefore$ (mTSPHT*) se puede resolver en tiempo polinomial. □

Capítulo 3

mTSPHT* en la araña

En este capitulo vamos a estudiar (mTSPHT*) cuando la red de *clientes-depósito* forma un grafo araña con el depósito en su centro. Vamos a etiquetar a las ramas del grafo sin seguir ningún criterio en específico: si hay M ramas, a la primera rama la llamaremos B_1 y a la última B_M. Supondremos conocido el conjunto de ramas $\hat{B} = \{B_1, B_2, \ldots, B_M\}$. Teniendo la notación para las ramas. el resto de las notaciones de (mTSPHT*) en el camino se extienden de manera natural a la araña.

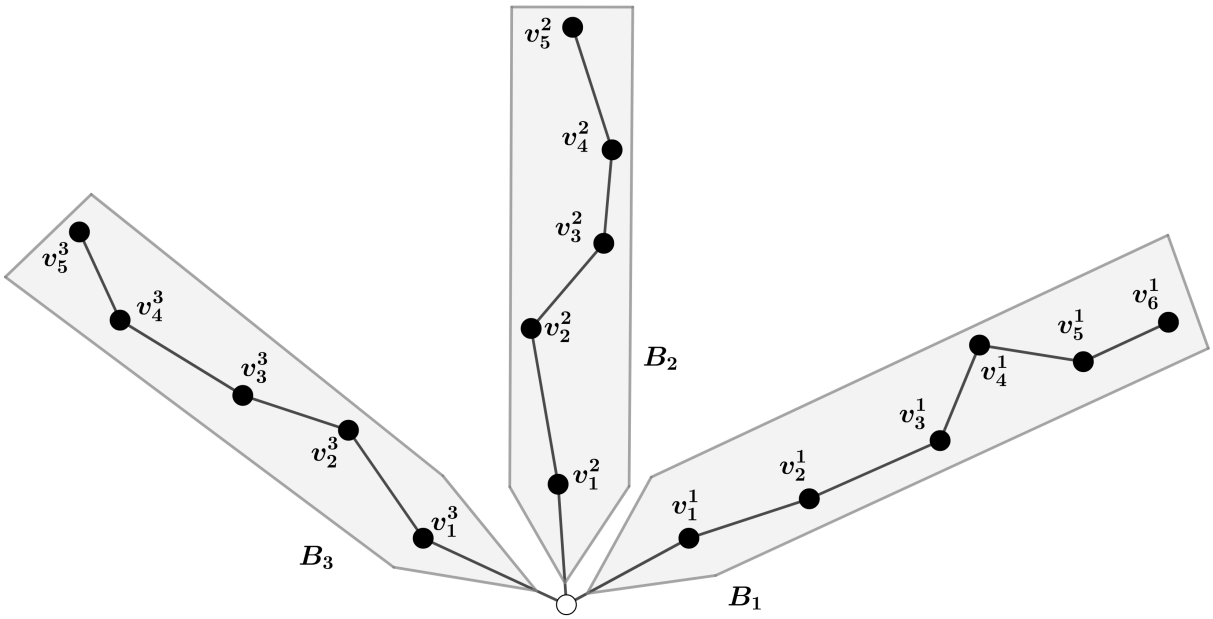


Figura 3.1: Ejemplo notación en instancia de (mTSPHT) en una araña compuesta por 3 ramas. El *depósito* esta coloreado de blanco.

Utilizaremos la misma estrategia del capitulo anterior. Vamos a empezar por estudiar el orden de visita de los clientes suponiendo conocidos los clientes que cada vehículo debe visitar, para luego encontrar una solución óptima que tenga cierta estructura que especificaremos más adelante.

Suponga que se conocen los clientes que la ruta S debe visitar. Vamos a dividir el análisis según la función objetivo a considerar. Vamos a suponer que la cantidad de clientes que tiene que visitar S es N.

- **Tiempo de Ruta:**
  Como cada cliente debe ser visitado, y como además, para cambiar de rama se debe pasar por el depósito y se debe terminar en él, todas las aristas deben ser recorridas al menos 2 veces. Es decir, cada rama es recorrida, al menos dos veces. Con esto,

$$\begin{aligned} Tr(S) &= \sum_{i=0}^{N}[t(s_i, s_{i+1}) + t_0] + t(s_N, D) \\ &= \sum_{i=0}^{N}[t(s_i, s_{i+1})] + t(s_N, D) + Nt_0 \\ &\geqslant 2\sum_{e\in E} t(e) + Nt_0 \\ &= 2\sum_{B:\text{B rama}}\sum_{e\in B} t(e) + Nt_0 \\ &= 2\sum_{B:\text{B rama}}[t(D, s_{|B|})] + Nt_0 \end{aligned}$$

  Ahora, si cada rama es recorrida una vez, los clientes son visitados en orden de etiquetado, desde el mas pequeño al más grande, se alcanza la cota anterior.
- **Tiempo de Completación:**
  El análisis es similar al caso anterior. Esta vez, sin embargo, no se debe regresar al depósito. Podemos suponer que toda arista se recorre al menos 2 veces, salvo las que corresponden al camino entre el *depósito* y el último cliente de la ruta.

$$\begin{aligned} T(S) &= \sum_{i=0}^{N}[t(s_i, s_{i+1}) + t_0] \\ &= \sum_{i=0}^{N}[t(s_i, s_{i+1})] + t(s_N, s_0) - t(s_N, s_0) + Nt_0 \\ &\geqslant 2\sum_{e\in E} t(e) + -t(s_N, s_0)Nt_0 \\ &= 2\sum_{B:\text{B rama}}[\sum_{e\in B} t(e)] - t(s_N, s_0) + Nt_0 \\ &\geqslant 2\sum_{B:\text{B rama}}[t(D, s_{|B|})] - \max_{B:\text{B rama}} t(v^B_{|B|}, D) + Nt_0 \end{aligned}$$

  Si recorremos cada rama, empezando por la más corta a la más larga, visitando a los clientes en orden de etiquetado, desde el más pequeño al más grande, se alcanza la cota anterior.

En ambos casos un orden de visita óptimo es el siguiente: empezando por la rama más corta a la rama más larga, se visitan los clientes en orden de etiquetado, desde el más pequeño al más grande. A esta secuencia de visita le diremos optimal y, en adelante, supondremos que todas los camiones visitan a sus clientes de manera optimal. Es decir, si conocemos los clientes que un vehículo debe visitar, daremos por conocida su ruta.

Observación El tiempo de ruta y tiempo de completación de una ruta S^k, dependen sólo de la cantidad de clientes que visita S^k y la distancia del *depósito* al cliente (de S^k) más alejado del *depósito* de cada rama.

Corolario 3.1 *Las solución resultante $\hat{R}$ al aplicar* MOVEMENT-2 *a un par de rutas (S^i, S^j) de una solución $\hat{S}$, es decir, $\hat{R} = (S_1, \ldots, S_{i-1}, R_i, \ldots, S_{j-1}, R_j, \ldots, S^m)$ cumple que:*

$$Tr(\hat{R}) \leqslant Tr(\hat{S}) \text{ y } T(\hat{R}) \leqslant T(\hat{S})$$

DEMOSTRACIÓN. Basta notar que aplicar MOVEMENT-2 al par de rutas (S^i, S^j) no cambia la cantidad de clientes que visita cada una de las rutas, ni la distancia del *depósito* a los clientes más alejados de cada rama de cada ruta, por lo que sumado a la observación anterior, tenemos lo deseado. □

A diferencia de (mTSPHT*) en el camino, (mTSPHT*) en la araña sin establecer condiciones en el número de ramas, para cada $m \geqslant 2$, es **NP-difícil**.

Proposición 3.2 (mTSPHT*) *minimizando el máximo largo de ruta en la araña es **NP-difícil**.*

DEMOSTRACIÓN. Vamos a probar que *Multiprocessor Scheduling [4]* (MS) se puede reducir a nuestro problema. En *Multiprocessor Scheduling* se tienen N trabajos, donde cada trabajo w_i tiene un largo l_i, y hay m procesadores. El objetivo es encontrar la asignación de trabajos a los procesadores minimizando el tiempo necesario para resolver los N trabajos, sin que haya superposición entre ellos. Es un problema **NP-difícil**, para cada $m \geqslant 2$.
Considere una instancia de (MS). Sea W el conjunto de trabajos, $|W| = N$, y $m \in \mathbb{N}$ el número de procesadores disponibles. Con respecto a estos datos vamos a construir el siguiente grafo G:

- Por cada trabajo en W agregamos un vértice a G. Además, agregaremos un vértice adicional, D.
- Para cada vértice v distinto a D agregamos la arista $\{D, v\}$.
- Sea w_z el trabajo más corto de W. A cada cliente asignaremos el largo de w_z, l_z, como el tiempo de visita. Es decir, $t_0 = l_z$.
- Para cada arista $e = \{u, D\}$ se le asignará la distancia de la siguiente forma: si el vértice u fue generado por el trabajo w_u, entonces $t(e) = \frac{l_u - l_z}{2}$.
- Consideraremos m camiones, los cuales parten desde D.

El grafo construido es una araña pues sólo D tiene grado mayor a 2, mientras que los otros vértices tienen grado 1. Observe que toda solución J de (MS) induce una solución $\hat{S}$ en la instancia de (mTSPHT*) de manera natural: para cada procesador k, se asignará una ruta S^k, la cual visita a los clientes generados por los trabajos de k.

Observe que

$$\begin{aligned} Tr(S^k) &= \sum_{v \in k} [2t(v, D) + t_0] \\ &= \sum_{v \in k} [l_v - l_z] + |S^k| l_z \\ &= \sum_{v \in k} [l_v] - |S^k| l_z + |S^k| l_z \\ &= \sum_{v \in k} [l_v] \\ &= \text{Makespan}(k) \end{aligned}$$

En donde Makespan(k) corresponde al tiempo que le toma al procesador k terminar sus trabajos. Entonces,

$$\max_{k \in \{1,\ldots,m\}} Tr(S^k) = \max_{k \in \{1,\ldots,m\}} \text{Makespan}(k),$$

Por lo que OPT(instancia de (MS)) $\geqslant$ OPT(instancia de (mTSPHT*)).
Considere la solución óptima de la instancia de (mTSPHT*). Cada cliente es visitado una vez por un vehículo, con lo que es posible derivar una solución de (MS): para la ruta S^k tome el procesador k y añada los trabajos que corresponden a los clientes visitados por la ruta k. Entonces,

$$\begin{aligned} \max_{k \in \{1,\ldots,m\}} Tr(S^k) &= \sum_{v \in C(S^k)} [2t(v, D) + t_0] \\ &= \sum_{v \in C(S^k)} [l_v] \\ &= \text{Makespan}(k) \end{aligned}$$

Con lo que OPT(instancia de (MS)) $\leqslant$ OPT(instancia de (mTSPHT*)).
$\therefore$ OPT(instancia de (MS)) = OPT(instancia de (mTSPHT*)).
$\therefore$ (MS) puede reducirse a (mTSPHT*).
$\therefore$ (mTSPHT*) en la araña, minimizando el tiempo de ruta, es **NP-difícil** . □

Corolario 3.3 (mTSPHT*) *minimizando el tiempo de completación en la araña es* ***NP-difícil****.*

Demostración. Considere una instancia de (mTSPHT*) minimizando el tiempo de ruta, defina H como:

$$H = Nt_0 + \sum_{\{u,v\} \in E} [2t(u, v)]$$

En donde N es el número de clientes totales, t_0 el tiempo de procesamiento de cada cliente y consideramos m camiones.

Generaremos una instancia para (mTSPHT*) minimizando el tiempo de completación. Vamos a hacer una copia de la red de *clientes-depósito* y añadiremos m vértices conectados al *depósito* a través de una arista. Además, para cada cliente v agregado, $t(v, D) = 2H$. A estos clientes les diremos *clientes-carga*.
Primero, observar que en toda solución óptima, correspondiente a la instancia de tiempo de completación, cada vehículo visita exactamente a un *cliente-carga*. En efecto, si existe algún vehículo que no visita a uno de ellos, entonces al menos un vehículo debe visitar a 2 *clientes-carga*, por lo que el valor de la solución es al menos $5H + t_0$. Sin embargo, considere la siguiente asignación: cada vehículo visita exactamente un *cliente-carga* y el resto de los clientes es visitado por un único vehículo, con lo que el valor de la solución es de $3H + t_0$. Luego, la solución anterior no puede ser óptima.
$\therefore$ Por lo tanto, en toda solución óptima, cada vehículo visita exactamente un *cliente-carga*.
Sea $\hat{R}$ una solución óptima para la instancia generada artificialmente. Vamos a construir una solución $\hat{S}$ para el problema original en base a $\hat{R}$ de la siguiente manera: como para cada ruta R^k el último cliente a ser visitado es un *cliente-carga* (pues la distancia al *depósito* es más grande que cualquier otra rama salvo las ramas correspondiente a *clientes-carga*), la ruta S^k va a consistir en los clientes de R^k salvo el ultimo cliente a visitar. Es decir, si el último cliente en ser visitado de R^k es v^*, entonces $C(S^k) = C(R^k) - v^*$. Luego, se tiene que:

$$\begin{aligned} T(R^k) &= Tr(S^k) + t(D, v^*) + t_0 \\ &= Tr(S^k) + H + t_0 \end{aligned}$$

Entonces, si deseamos buscar la ruta con el mayor valor de $T(R^k)$,

$$\begin{aligned} \max_{k=1,\dots,m} T(R^k) &= \max_{k=1,\dots,m} [Tr(S^k) + t(D, v^*) + t_0] \\ &= \max_{k=1,\dots,m} [Tr(S^k)] + H + t_0 \end{aligned}$$

Por otro lado, considere una solución óptima del problema original $\hat{S}'$. Vamos a generar una solución $\hat{R}'$ para la instancia de tiempo de completación agregando a cada ruta un *cliente-carga*, luego

$$\begin{aligned} \max_{k=1,\dots,m} T(R'_k) &= \max_{k=1,\dots,m} [Tr(S'_k)] + H + t_0 \\ &\leqslant \max_{k=1,\dots,m} [Tr(S^k)] + H + t_0 \\ &= \max_{k=1,\dots,m} T(R^k) \end{aligned}$$

La desigualdad anterior es debido a que $\hat{S}'$ es una solución óptima del problema original, mientras que $\hat{S}$ es una solución factible. Pero, como $\hat{R}$ es una solución óptima de la instancia creada, se cumple que

$$\max_{k=1,\dots,m} T(R^k) \leqslant \max_{k=1,\dots,m} T(R'_k),$$

con lo que

$$\max_{k=1,\dots,m}[Tr(S'_k)] + H + t_0 = \max_{k=1,\dots,m}[Tr(S^k)] + H + t_0$$
$$\max_{k=1,\dots,m}[Tr(S'_k)] = \max_{k=1,\dots,m}[Tr(S^k)].$$

Como $\hat{S}'$ es una solución óptima de problema original, entonces $\hat{S}$ también es una solución óptima. Luego, (mTSPHT*) minimizando tiempo de ruta puede reducirse a (mTSPHT*) minimizando tiempo de completación.
$\therefore$ (mTSPHT*) minimizando tiempo de completación es **NP-difícil**. □

Ahora vamos a caracterizar una solución óptima de manera de encontrar un algoritmo para resolver el problema (de manera óptima). Para ello vamos a utilizar GlobalSorting(Spider-Version). Este algoritmo recibe una solución factible $\hat{S}$ del problema, y entrega una solución factible $\hat{R}$. Para construir la solución $\hat{R}$ a partir de $\hat{S}$, el algoritmo primero ordena cada rama de $\hat{S}$ de manera que cada ruta, restringida a B, tenga la propiedad de intervalos, para toda rama B. Luego para cada par de rutas y para cada par de ramas se utiliza Movement-2, la cual rompe con la propiedad de intervalos y para restituirla se usa Shift.

Algorithm 9: GlobalSorting(Spider-Version)

Input: Solución factible $\hat{S}$
Output: $\hat{R}$ solución factible

$\hat{R} \leftarrow S$
for $(b \in \{1,\dots,m\})$ **do**
  $R^b \leftarrow LocalSorting(R^{B_b}, B_b)$
for $i = 1,\dots,m$ **do**
  for $j = i,\dots,m$ **do**
    for $b_1 = 1,\dots,m$ **do**
      for $b_2 = b_1+1,\dots,m$ **do**
        $(R^i, R^j) \leftarrow$ Movement-2$(R^i, R^j, B_{b_1}, B_{b_2})$
        for $l \in \{b_1, b_2\}$ **do**
          while $(\exists n > 0 : \text{label}(s_{n+1}^{i,B_l}) \neq \text{label}(s_n^{i,B_l}) + 1)$ **do**
            $R \leftarrow$ Shift(R, i, B_l)
          while $(\exists n > 0 : \text{label}(s_{n+1}^{j,B_l}) \neq \text{label}(s_n^{j,B_l}) + 1)$ **do**
            $R \leftarrow$ Shift(R, j, B_l)
return $\hat{R}$

Proposición 3.4 *Para* (mTSPHT*) *en la araña existe una solución óptima tal que:*

1. *La solución es estándar.*
2. *Para cada par de rutas concurrentes S^1 y S^2 se cumple que $S^1 \preccurlyeq S^2 \vee S^1 \succcurlyeq S^2$.*

Demostración. Sea $\hat{S}$ una solución óptima del problema y sea M la cantidad de ramas del grafo. Vamos a aplicar GlobalSorting(Spider-Version) a $\hat{S}$. Dado que es el mismo

procedimiento utilizado en el caso del camino, salvo que vamos iterando a través de cada rama, sólo queda revisar que $\hat{R}$ tiene un orden parcial. En efecto, recordar que en cada iteración las rutas afectadas son sólo las que participan en MOVEMENT-2, y se cumple (en cada rama B_{b_1} y B_{b_2}) que las rutas que estaban más cerca (lejos) del *depósito* que $R^i \vee R^j$ al término de la iteración siguen estando más cerca (lejos) que $R^i \vee R^j$. Es decir, si en la rama B_{b_1}, la ruta R^l está más cerca (lejos) del *depósito* que R^i, y al final de MOVEMENT-2, R^i sigue sirviendo clientes en B_{b_1}, entonces R^l sigue estando más cerca(lejos) del *depósito* que R^i. Por lo que ir ordenando las ramas de par en par, en orden y sin alterar los resultados de iteraciones anteriores, hace que la solución $\hat{R}$ posea orden parcial.
$\hat{R}$ es una solución óptima pues tanto como SHIFT y MOVEMENT-2 no aumentan ni el tiempo de completación ni el tiempo ruta.
$\therefore \hat{R}$ es una solución óptima que cumple con las condiciones 1. y 2. □

Tener una solución óptima estructurada nos permite proponer una primera forma de resolver el problema de manera exacta: vamos a suponer que la red de *clientes-depósito* posee M ramas. Consideremos la siguiente función, la cual es una extensión de la construida en el caso del camino.

$$F_M(|B_1|,\ldots,|B_M|,m) = \begin{cases} p(v_{B_1}^{B_1},\ldots,v_{B_M}^{B_M}) + \sum_{i=1}^{M} |B_i| t_0 & m = 1 \\ \min_{\vec{i}=(i_1,\ldots,i_M)\in [B_1]\times\cdots\times[B_M]} \{\max(F_M(B_1 - i_1,\ldots,B_M - i_M, m-1), & \\ p(v_{B_1}^{B_1} - i_1,\ldots,v_{B_M}^{B_M} - i_M) + sum(\vec{i})t_0)\} & m > 1 \end{cases}$$

En donde,

- $[B_j] = \{1,2,3,\ldots,B_j\}$.
- m es el número de camiones
- p es una función que depende de la función objetivo que se considera. Si se minimiza el tiempo de ruta:

$$p(v_{B_1}^{B_1},\ldots,v_{B_M}^{B_M}) = 2\sum_{b=1}^{M} t(D, v_{|B_b|}^{B_b})$$

  Si lo que se minimiza es el tiempo de completación:

$$p(v_{B_1}^{B_1},\ldots,v_{B_M}^{B_M}) = 2\sum_{b=1}^{M} [t(D, v_{|B_b|}^{B_b})] - \max_{b\in\{1,\ldots,m\}} t(D, v_{|B_b|}^{B_b})$$

Lo que hace esta función es, mediante programación dinámica, encontrar los "intervalos de clientes" (conjunto de clientes consecutivos) en cada rama que las rutas deben visitar. Note que a diferencia del caso del camino, nuestra función depende de manera exponencial de M. Más específicamente, encontrar la solución con esta función toma $O(mN^{2M})$ (recordar que notamos por N a la cantidad de clientes a servir).

Si bien utilizar la función $F_M(|B_1|,\ldots,|B_M|,m)$ para resolver el problema no es eficiente, vamos a utilizarla para aproximarlo. Vamos a presentar una aproximación cuando hay dos vehículos y un **PTAS** cuando el número de camiones es fijo.

3.1. Aproximación para $m = 2$

Recordar que si sabemos qué clientes deben ser visitados por cada vehículo, entonces damos por conocida la secuencia de visita. Supondremos que tenemos una instancia de (mTSPHT*) en una araña ($G = (V \cup \{D\}, E)$ con 2 camiones, $N = |V|$ clientes y M ramas. Por conveniencia, vamos a definir el **largo** y **peso** de una rama. El largo de la rama B viene dado por $t_B = \sum_{(u,v)\in B} t(u,v)$ y corresponde al tiempo de viaje del *depósito* al último cliente de B, mientras que el peso de una rama B viene dado por $\text{peso}(B) = 2t_B + |B|t_0$ y representa el tiempo que le toma a un vehículo visitar a todos los clientes de B y volver al depósito. A la rama más pesada la notaremos por $\bar{B}$.

Si $\hat{B}$ es un conjunto de ramas, entonces:

$$\text{peso}(\hat{B}) = \sum_{B\in\hat{B}} \text{peso}(B)$$
$$\text{largo}(\hat{B}) = \sum_{B\in\hat{B}} \text{largo}(B)$$

3.1.1. Tiempo de ruta

Cada cliente debe ser servido y los camiones deben volver al *depósito*, por lo que cada arista debe ser recorrida al menos 2 veces. Vamos a definir H como:

$$2H = \sum_{v\in V} t(v) + 2\sum_{(u,v)\in E} t(u,v) = Nt_0 + 2\sum_{(u,v)\in E} t(u,v)$$

Notar que H es una cota inferior para OPT, con OPT el valor de la solución óptima.En efecto, sea $\hat{S} = (S^1, S^2)$ una solución óptima. Como cada cliente debe ser servido por S^1 o S^2, y cada arista es recorrida al menos 2 veces por algún vehículo, se tiene que:

$$Tr(S^1) + Tr(S^2) \geqslant 2H$$

Con lo que $OPT = \text{máx}_{i\in\{1,2\}} Tr(S^i) \geqslant H$. Utilizando esta cota encontraremos una $\frac{3}{2}$-aproximación.

Para aproximar el tiempo de ruta, vamos a tomar una acción de acuerdo peso de $\bar{B}$. Si es muy pesada, vamos a repartir los clientes de la rama a optimalidad y las ramas restantes serán distribuidas de manera glotona entre los dos camiones. Si el peso de $\bar{B}$ es mediano, un vehículo va a visitar $\bar{B}$ mientras el otro visita las restantes. Si el peso de $\bar{B}$ es pequeño, vamos a repartir las ramas entre los camiones de manera glotona.

Algorithm 10: APPROXIMATION-TR

Input: Instancia del problema.
Output: $\frac{3}{2}$-aproximación $\hat{S}$.

$\hat{S} \leftarrow \emptyset$
Crear conjunto $\hat{B}$ de ramas.
Calcular el peso de cada rama B y calcular H.
case $(peso(\bar{B}) > \frac{3}{2}H)$ **do**
    Genere el siguiente sub-problema: $G_2 = G[\bar{B}]$, las distancias serán heredadas del problema original al igual que el tiempo de procesamiento. Resolveremos el problema (mTSPHT*) en el camino con 2 vehículos y sea $\hat{R}$ la solución encontrada.
    $C(S^1) \leftarrow C(R^1),\ C(S^2) \leftarrow C(R^2)$.
    if $(Tr(S^1) > Tr(S^2))$ **then**
        $C(S^2) \leftarrow C(S^2) + V \setminus V[\bar{B}]$
    else
        $C(S^1) \leftarrow C(S^1) + V \setminus V[\bar{B}]$
case $(\frac{H}{2} < peso(\bar{B}) \leqslant \frac{3}{2}H)$ **do**
    $C(S^1) \leftarrow V[\bar{B}] \setminus \{D\}$
    $C(S^2) \leftarrow V \setminus V[\bar{B}]$
otherwise do
    $(C(S^1), C(S^2)) \leftarrow$ GREEDYASSIGNMENT-1$(\hat{B})$
return $\hat{S} = (S^1, S^2)$

Algorithm 11: GREEDYASSIGNMENT-1

Input: $\hat{B}$ conjunto de ramas con sus pesos.
Output: $(C(S^1), C(S^2))$ partición de vértices del grafo en dos conjuntos

$C(S^1) \leftarrow \emptyset, C(S^2) \leftarrow \emptyset$.
Ordenar las ramas de $\hat{B}$ en orden decreciente de peso.
for $B \in \hat{B}$ *(siguiendo el orden)* **do**
    if $Tr(S^1) > Tr(S^2)$ **then**
        $C(S^1) \leftarrow C(S^1) + V[B] - D$
    else
        $C(S^2) \leftarrow C(S^2) + V[B] - D$

Análisis de Approximation-Tr

1. Complejidad del algoritmo:
   El primer caso, $(\text{peso}(\bar{B}) > \frac{3}{2}H)$, es $O(N)$ pues tanto generar el grafo como resolver el sub-problema toma $O(N)$. Luego las líneas 6-10 son asignaciones que realizan en $O(N)$. El segundo caso, $(\frac{H}{2} < \text{peso}(\bar{B}) \leqslant \frac{3}{2}H)$, es una simple asignación que toma $O(N)$ y el último caso, $(peso(\bar{B}) \leqslant \frac{H}{2})$, corresponde a utilizar GreedyAssignment-1, el cual toma $O(N\log(N))$, pues debe ordenar las ramas y luego iterar sobre ellas.
   Por lo que el algoritmo es $O(N\log(N))$
2. Correctitud del algoritmo:
   A la solución entregada por el algoritmos la denotaremos por $\hat{S}$ y a su valor por ALG. Análogamente el valor de una solución óptima la notaremos por OPT. Analizaremos las soluciones de acuerdo a los casos correspondientes,
   - Caso: $\text{peso}(\bar{B}) > \frac{3}{2}H$
     Llamando **resto** a las ramas $\hat{B} - \bar{B}$,

     $$\begin{aligned}\text{peso(resto)} &= 2H - \text{peso}(\bar{B}) \\ &\leqslant 2H - \frac{3}{2}H \\ &\leqslant \frac{H}{2}\end{aligned}$$

     Por otro lado, el valor OPT restringido a $\bar{B}$ es mayor o igual que el valor del sub-problema considerando sólo $\bar{B}$ con dos camiones. Luego,

     $$\text{ALG} \leqslant \text{OPT} + \frac{H}{2} \leqslant \frac{3}{2}\,\text{OPT}$$

   - Caso: $\frac{H}{2} < \text{peso}(\bar{B}) \leqslant \frac{3}{2}H$
     Retomando la noción de resto,

     $$\begin{aligned}\text{peso(resto)} &= 2H - \text{peso}(\bar{B}) \\ &\leqslant 2H - \frac{H}{2} \\ &\leqslant \frac{3}{2}H\end{aligned}$$

     Con lo que, para $i \in \{1, 2\}$

     $$Tr(S^i) \leqslant \frac{3}{2}H \leqslant \frac{3}{2}\,\text{OPT}$$

   - Caso: $\text{peso}(\bar{B}) \leqslant \frac{H}{2}$
     Considere S^1 y S^2 al finalizar la asignación de las ramas en GreedyAssignment-1. Observe que $Tr(S^1) + Tr(S^2) = 2H$, pues cada rama es visitada por sólo un vehículo, luego cada arista se recorre exactamente 2 veces y cada vértice es visitado sólo una vez. Así, si $Tr(S^1) = \min\limits_{i\in\{1,2\}} Tr(S^i)$. Entonces,

     $$Tr(S^1) \leqslant H$$

Por otro lado, como la rama más pesada de $\hat{B}$ pesa a lo más $\frac{H}{2}$ y la forma de repartir las ramas es de manera glotona, al momento en que $Tr(S^2) > H$ no se le vuelven a asignar más ramas en GreedyAssignment-1 (debido a que $Tr(S^1) + Tr(S^2) = 2H$, luego si $Tr(S^2) > H$ entonces no hay forma de que $Tr(S^1) < Tr(S^2)$ no importa que ramas queden por agregar). Entonces, la penúltima rama a ser agregada a S^2 ocurre cuando $Tr(S^2) \leqslant H$ y ésta rama pesa a lo más $\frac{H}{2}$, se tiene que:

$$Tr(S^2) \leqslant \frac{2H}{2} + \frac{H}{2} = H + \frac{H}{2} = \frac{3H}{2}$$

Si $Tr(S^2) = \min_{i\in\{1,2\}} Tr(S^i)$, el análisis es el mismo, cambiando los roles de S^1 y S^2.

Por lo que en este caso también $\text{ALG} \leqslant \frac{3}{2}H \leqslant \frac{3}{2}\,\text{OPT}$.

Finalmente, en los tres casos posibles $\text{ALG} \leqslant \frac{3}{2}\,\text{OPT}$, con lo cual nuestro algoritmo es una $\frac{3}{2}$-aproximación.

3.1.2. Tiempo de completación

Vamos a mantener las notación del caso anterior y algunas más. Denotaremos por $\bar{B}_1$ y $\bar{B}_2$ a la rama más larga y a la segunda rama más larga respectivamente. A diferencia de tiempo de ruta, los camiones no se devuelve al *depósito*, por lo que si definimos H_1 como:

$$\begin{aligned} 2H_1 &= \sum_{v\in V}[t(v)] + t_{\bar{B}_1} + t_{\bar{B}_2} + 2\sum_{B\in\hat{B}-\{\bar{B}_1,\bar{B}_2\}} t_B \\ &= Nt_0 + t_{\bar{B}_1} + t_{\bar{B}_2} + 2\sum_{B\in\hat{B}-\{\bar{B}_1,\bar{B}_2\}} t_B \end{aligned}$$

Entonces H_1 es una cota inferior para OPT. En efecto, sea $\hat{S} = (S^1, S^2)$ una solución óptima, como cada cliente debe ser visitado y cada rama debe ser visitada a lo menos 2 veces, salvo a lo mas 2 ramas que son visitadas 1 vez, luego se tiene que:

$$\begin{aligned} 2H_1 &\leqslant T(S^1) + T(S^2) \\ H_1 &\leqslant \max_{i\in\{1,2\}} T(S^i) \\ &= \text{OPT} \end{aligned}$$

Además, vamos a definir H_2 como:

$$2H_2 = \sum_{v\in V} t(v) + 2\sum_{B\in\hat{B}-\{\bar{B}_1\}} t_B$$

Dónde H_2 es cota inferior pues $H_2 \leqslant H_1$.Utilizando estas cotas encontraremos una $\frac{5}{3}$-aproximación. Para ello definiremos peso* de una rama B como peso$^*(B) = t_B + |B|t_0$.

Análogamente al aproximar el tiempo de ruta, vamos a tomar una acción de acuerdo al peso* de $\bar{B}_1$ (en vez del peso de $\bar{B}$). Si $\bar{B}_1$ es muy *-pesada,vamos a repartir los clientes de $\bar{B}_1$ a optimalidad entre los 2 vehículos y el resto de las ramas serán asignadas de manera

glotona. Si es medianamente *-pesada, un vehículo recorrerá $\bar{B}_1$ y otro vehículo recorrerá las ramas faltantes. Si $\bar{B}_1$ es *-pequeña, volveremos a dividir nuestro curso de acción: si la rama más pesada y la rama más larga coinciden, vamos a asignar las ramas de manera glotona. De lo contrario vamos a volver a repetir lo que hacíamos en la aproximación del tiempo de ruta (considerando que esta vez la función objetivo es el tiempo de completación)

Algorithm 12: APPROXIMATION-T

Input: Instancia del problema.
Output: $\hat{S}$ solución factible.

$S \leftarrow \emptyset$
Crear conjunto $\hat{B}$ de ramas.
Calcular el peso, peso* y largo de cada rama.
Obtener $\bar{B}_1, \bar{B}_2, t_{\bar{B}_1}, t_{\bar{B}_2}, \bar{B}$, calcular $2H_1$, $2H_2$.
case $(\text{peso}^*(\bar{B}_1) > \frac{5}{3}H_1)$ **do**
    Generar sub-problema: $G_2 = G[\bar{B}_1]$, las distancias y tiempo de procesamiento serán heredades del problema original. Resolveremos el problema ((mTSPHT*) en el camino con 2 vehículos) y sea $\hat{R}$ la solución encontrada.
    $C(S^1) \leftarrow C(R^1) \ \wedge \ C(S^2) \leftarrow C(R^2)$
    for $(B \in \hat{B} - \bar{B}_1)$ **do**
        if $(T(S^1) \leqslant T(S^2))$ **then**
            $C(S^1) \leftarrow C(S^1) + V[B] - D$
        else
            $C(S^2) \leftarrow C(S^2) + V[B] - D$
case $\frac{H_1}{3} < \text{peso}^*(\bar{B}_1) \leqslant \frac{5}{3}H_1$ **do**
    $S^1 \leftarrow V[\bar{B}_1] - D$
    $S^2 \leftarrow V - V[\bar{B}_1] - D$
otherwise do
    case $\bar{B}_1 = \bar{B}$ **do**
        $(S^1, S^2) \leftarrow$ GREEDYASSIGNMENT-2$(\hat{B})$
    otherwise do
        case $\text{peso}(\bar{B}_1) \geqslant \frac{5}{3}H_2$ **do**
            Repetir 6-12, pero con $G_2 = \bar{B}$.
        case $\frac{2}{3}H_2 < \text{peso}(\bar{B}_1) \leqslant \frac{5}{3}H_2$ **do**
            $S^1 = \{\bar{B}_1\}$, $S^2 = \hat{B} - \bar{B}_1$
        otherwise do
            $(S^1, S^2) \leftarrow$ GREEDYASSIGNMENT-2$(\hat{B})$
return $\hat{S} = (S^1, S^2)$

Algorithm 13: GreedyAssignment-2

Input: B conjunto de ramas con sus pesos y largos respectivos
Output: $(C(S^1), C(S^2))$ partición de vértices del grafo en dos conjuntos

```
1 Ordenar las ramas de acuerdo a su largo de manera decreciente
2 C(S^1) ← B̄_1, C(S^2) ← B̄_2.
3 for (B ∈ B̂ − {B̄_1, B̄_2}, siguiendo el orden) do
4     if (T(S^1) ⩽ T(S^2)) then
5         C(S^1) ← C(S^1) + B
6     else
7         C(S^2) ← C(S^2) + B
8 return (C(S^1), C(S^2))
```

Análisis de Approximation-T

1. Complejidad del algoritmo:
   El caso de la línea 5 toma $O(N)$ ya que tanto generar el grafo como resolver el sub-problema demora $O(N)$. El ciclo de la línea 8 toma a lo más $O(N)$, pues hay a lo más N ramas. El caso de la linea 15 toma $O(N\log(N))$ ya que sub-caso en 16, GreedyAssignment-2, toma $O(N\log(N))$, que es lo que tarde en ordenar las ramas y sólo $O(N)$ al ciclar a través de ellas. Volviendo a Approximation-T, el sub-caso de la línea 18 toma $O(N\log(N))$ pues es análogo a lo realizado en 15.
   Por lo que el algoritmo es de orden $O(N\log(N))$.
2. Correctitud del algoritmo:
   Separaremos el análisis en casos,
   - Caso: $\text{largo}(\bar{B}_1) > \frac{5}{3}H_1$
     Resolver el problema con $G_2 = G[\bar{B}_1]$ entrega una solución de valor menor que OPT pues la solución óptima restringida a $\bar{B}_1$, es una solución factible del sub-problema. Por lo que los conjuntos obtenidos en este paso previo tienen un valor menor a OPT. Además si resto $= \hat{B} - \bar{B}_1$,

$$\begin{aligned}\text{peso(resto)} &= 2H_1 - \text{peso}^*(\bar{B}_1) + t_{\bar{B}_2} \\ &\leqslant \frac{H_1}{3} + t_{\bar{B}_2} \\ &\leqslant \frac{H_1}{3} + \frac{H_1}{3} \\ &= \frac{2}{3}H_1\end{aligned}$$

     Note que $2H_1 \geqslant \text{peso}^*(\bar{B}_1) + t_{\bar{B}_2}$ de ahí la segunda desigualdad. Por lo que agregar las ramas restantes a alguno de los conjuntos aumenta su valor en a lo mas $\frac{2}{3}H_1$. Por lo que en este caso ALG $\leqslant \frac{5}{3}$ OPT.
   - Caso: $\frac{H_1}{3} < \text{peso}^*(\bar{B}_1) \leqslant \frac{5}{3}H_1$

$$\begin{aligned} Tr(S^1) &\leqslant \frac{5}{3}H_1 \\ Tr(S^2) &= 2H_1 - peso^*(\bar{B}_1) \\ &\leqslant \frac{5}{3}H_1 \end{aligned}$$

Por lo que ALG $\leqslant \frac{5}{3}$ OPT

- Caso: peso$^*(\bar{B}_1) \leqslant \frac{H}{3}$
  Volveremos a subdividir en los casos correspondientes,
  - $\bar{B}_1 = \bar{B}$, es decir, la rama más pesada y la rama más larga son la misma.

    * Sub-Caso: $peso^*(\bar{B}_1) \leqslant \frac{H_1}{3}$
      Observe que en GreedyAssignment-2 justo antes de agregar la última rama al par de conjuntos $(C(S^1), C(S^2))$, estos son tales que cada rama aparece sólo en uno de ellos y falta sólo una rama, por lo que $T(S^1) + T(S^2) \leqslant 2H_1$ (por la forma del algoritmo, al completar la asignación de todas las ramas esa es una igualdad). Luego, el más pequeño debe ser menor que H_1 y por lo tanto menor que OPT. Además, el agregar la rama restante al más pequeño hace que éste último aumente en a lo más $\frac{H_1}{3} + t_{\bar{B}_1} \leqslant \frac{H_1}{3} + \frac{H_1}{3} \leqslant \frac{2}{3}$ OPT. Con lo que esta ruta es a lo más $H_1 + \frac{2}{3}OPT \leqslant \frac{5H}{3}$
      Por otro lado, la ruta con T más grande, en la penúltima iteración no puede ser mayor que $H_1 +$ peso$^*(\bar{B}_1)$, pues al momento de pasar el valor H_1 ya no se le siguen asignando más ramas a recorrer, y como la rama más pesada es $\bar{B}_1$ se tiene la cota. Luego esta ruta a lo más aumenta en $\leqslant \frac{H_1}{3} + \frac{H_1}{3}$. Con lo que esta ruta es a lo más $H_1 + \frac{2}{3}$ OPT $\leqslant \frac{5H}{3}$
      Por lo que en este caso se obtiene una $\frac{5}{3}$-aproximación.
  - $\bar{B}_1 \neq \bar{B}$, es decir, la rama más pesada y la rama más larga son distintas

    * Sub-Caso: peso$(\bar{B}) > \frac{5}{3}H_2$

$$\begin{aligned} peso^*(\hat{B} - \bar{B}) &= 2H - peso(\bar{B}) - t_{\bar{B}_1} \\ &= 2H_2 + 2t_{\bar{B}_1} - peso(\bar{B}) - t_{\bar{B}_1} \\ &\leqslant \frac{H_2}{3} + t_{\bar{B}_1} \\ &\leqslant \frac{H_2}{3} + \frac{H_1}{3} \\ &\leqslant \frac{2}{3}\,\text{OPT} \end{aligned}$$

      Recordar que como $\bar{B}$ no es la rama más larga, entonces $\bar{B}_1$ se encuentra en $\hat{B} - \bar{B}$. Los argumentos en este caso son idénticos a los anteriores, es decir, la solución del problema restringido a $\bar{B}$ es menor al óptimo y el agregar $V - \bar{B}$ a cualquiera de los camiones aumenta en a lo más $\frac{2}{3}H_2$ el valor de la solución. Por lo que se tiene una $\frac{5}{3}$-aproximación.

* Sub-Caso: $\frac{2}{3}H_2 \leqslant \text{peso}(\bar{B}) \leqslant \frac{5}{3}H_2$

$$\begin{aligned} T(S^1) &\leqslant \frac{5}{3}H_2 \\ T(S^2) &= 2H - \text{peso}(\bar{B}) - t_{\bar{B}_1} \\ &= 2H_2 + 2t_{\bar{B}_1} - \text{peso}(\bar{B}) - t_{\bar{B}_1} \\ &\leqslant \frac{4}{3}H_2 + t_{\bar{B}_1} \\ &\leqslant \frac{4}{3}H_2 + \frac{1}{3}H_1 \\ &\leqslant \frac{5}{3}\,\text{OPT} \end{aligned}$$

  Por lo que en este caso se tiene una $\frac{5}{3}$-aprox.

* Sub-Caso: $\text{peso}(\bar{B}) \leqslant \frac{2}{3}H_2$
  Note que en GreedyAssignment-2 justo antes de agregar la última rama al par de conjuntos S^1, S^2, estos son tales que cada rama aparece sólo en uno de ellos y falta una única rama. Por lo que $T(S^1)+T(S^2) \leqslant 2H_1$(por la forma del algoritmo, al completar la asignación de todas las ramas esa es una igualdad) luego la ruta con el T menor debe ser más pequeña que H_1, por lo tanto menor que OPT. Al agregar la rama restante al más pequeño aumenta su T en a lo más $\text{peso}(\bar{B}) \leqslant \frac{2}{3}H_2 \leqslant \frac{2}{3}\,\text{OPT}$. Por lo que el tiempo de completación de éste es a lo más $\frac{5}{3}\,\text{OPT}$. Luego, la ruta con T más grande (con respecto a la penúltima iteración), debe ser menor que $H_1 + \text{peso}(\bar{B})$, pues una vez que haya superado H_1 no se le siguen asignando más ramas a visitar. Entonces, a lo más se le puede agregar una rama cuando es $\leqslant H_1$. Con lo que su tiempo de completación es a lo más es $\frac{5}{3}\,\text{OPT}$.
  Por lo tanto, en este caso se obtiene una $\frac{5}{3}$-aprox.

Por lo que el algoritmo entrega una $\frac{5}{3}$-aproximación.

3.1.3. PTAS para m fijo

Vamos a presentar un **PTAS** para m fijo, pero antes recordar que si tenemos una instancia del problema con M ramas y m camiones tenemos la siguiente función para resolver el problema:

$$F_M(|B_1|\dots|B_M|,m)=\begin{cases}p(v_{B_1}^{B_1}\dots v_{B_M}^{B_M})+\sum_{i=1}^{M}|B_i|t_0 & m=1\\ \min_{\vec{i}=(i_1\dots i_M)\in\left[B_1\right]\times\cdots\times\left[B_M\right]}\{\max(F_M(B_1-i_1,\dots,B_M-i_M,m-1),\\ p(v_{B_1}^{B_1}-i_1,\dots,v_{B_M}^{B_M}-i_M)+sum(\vec{i})t_0)\} & m>1\end{cases}$$

La idea del **PTAS** es generar una instancia nueva, resolverla y a partir de la solución encontrada derivar una solución del problema original. La nueva instancia se genera primero ordenando las ramas de acuerdo a su peso, para luego considerar una cierta cantidad de las ramas más pesadas, que dependen de la precisión deseada. Las ramas restantes juntarlas, en medida que el peso combinado no sobrepase cierto límite, en nuevo un cliente al que llamaremos cliente *cluster* de tal manera que si un vehículo visita a este cliente *cluster*, significa que visitará a todas las ramas que lo componen. Luego, con esta nueva instancia resolveremos el problema con la función F_M, con M adecuado, para luego generar la solución del problema original. Si bien el procedimiento es el mismo para ambas funciones objetivo, la aproximación obtenida es diferente en ambos casos.
Al igual que en la aproximación para 2 vehículos, vamos a definir una cota inferior, que depende de m, H_m:

$$mH_m = Nt_0 + 2\sum_{B\in\hat{B}}[t(D,v_{|B|}^{B})]$$

Observar que H_m es una cota inferior para el problema de minimizar el largo de ruta con m repartidores. En efecto, como cada vértice debe ser cubierto, cada arista es recorrida por lo menos 2 veces y se considerar el tiempo de espera de cada cliente (Nt_0). Entonces, si $\hat{S}=(S^1,\dots S^m)$ es una solución factible,

$$\sum_{r=1}^{m} Tr(S^r) \geqslant mH_m$$

Por lo que $Tr(\hat{S}) = \max_{r\in\{1,\dots,m\}} Tr(S^r) \geqslant H_m$.
Para el tiempo de completación vamos a considerar la siguiente cota inferior, que depende de m, H'_m:

$$mH'_m = Nt_0 + \sum_{B\in\hat{B}}[t(D,v_{|B|}^{B})]$$

Cabe notar que $H_m \leqslant 2H'_m$. Veamos que H'_m es una cota inferior. Considere una solución factible $\hat{S}$, como cada cliente debe ser visitado, cada arista es usada por lo menos una vez y debemos considerar el tiempo de espera de cada cliente (Nt_0). Entonces,

$$\textstyle\sum_{r=1}^{m} T(S^r) \geqslant mH'_m$$

Por lo que $T(\hat{S}) = \max_{r\in\{1,\dots,m\}} T(S^r) \geqslant H'_m$.

Algorithm 14: GreedyAssignment-3

Input: Conjunto de ramas $\hat{B}$, con sus pesos y una cota superior L, tiempo de espera t_0, entero M.

Output: $(\hat{B}', Z)$:Conjunto de ramas $\hat{B}'$, donde cada rama pesa a lo más L. $Z = (z_1, \ldots, z_{2M})$ donde z_i indica el conjunto de ramas del grafo original representadas por la rama B'_i de $\hat{B}'$.

$\hat{B}', Z \leftarrow \emptyset, T \leftarrow \hat{B}$.
for $(i = 1, \ldots, 2M)$ **do**
  if $(T = \emptyset)$ **then**
    break
  else
    $d_i \leftarrow 0,\ z_i \leftarrow \emptyset$
    for $(B \in T)$ **do**
      if $(\text{peso}(B) + d_i - t_0 \leqslant L)$ **then**
        $d_i \leftarrow d_i + \text{peso}(B)$
        $z_{v_{c_i}} \leftarrow z_{v_{c_i}} \cup \{B\}$
        $T \leftarrow T - B$
        $B' \leftarrow (D, v_{c_i}),\ t(D, v_{c_i}) \leftarrow d_i$
        $Z \leftarrow Z \cup \{z_{v_{c_i}}\},\ \hat{B}' \leftarrow \hat{B}' \cup \{B'\}$
return $(\hat{B}', Z)$

Algorithm 15: PTAS para (mTSPHT*) en la araña

Input: Instancia de (mTSPHT*) en la araña, con la precisión $\bar{\varepsilon}$ deseada.

Output: $(1 + \bar{\varepsilon})$-aproximación para (mTSPHT*)

$S = (S^1, \ldots, S^m)$, con $S^i = \emptyset, \forall i$. Generar el conjunto de ramas $\hat{B}$, calcular sus pesos y H_m. Si se minimiza el tiempo de completación, $\varepsilon \leftarrow \frac{\bar{\varepsilon}}{4m}$, de lo contrario, $\varepsilon \leftarrow \frac{\bar{\varepsilon}}{m}$.
$M \leftarrow \lceil \frac{1}{\varepsilon} \rceil$
if $(|\hat{B}| \leqslant M)$ **then**
  Resolver el problema con $F_{|\hat{B}|}(|B_1|, \ldots, |B_{|\hat{B}|}|, m)$.
else
  $(\hat{B}', Z) \leftarrow$ GreedyAssignment-3$(\{B_{M+1}, \ldots, B_{|\hat{B}|}\}, \frac{mH_m}{M}, t_0, M)$.
  Generar el sub-problema:
  Construir la red de *depósito-clientes* la cual corresponde a las ramas $\hat{B}_{sub} = \{B_1, \ldots, B_M\} \cup \hat{B}'$ y el depósito $\{D\}$.
  Resolver el problema con $F_{|\hat{B}_{sub}|}$, a la solución encontrada llamarla $\hat{R} = (R^1, \ldots, R^m)$.
  Construir una solución del problema original a partir de $\hat{R}$:
  for $(k \in \{1, \ldots, m\})$ **do**
    $C(S^i) \leftarrow C(R^i)$
    while $(\exists v_c \in C(S^i) : v_c \notin V)$ **do**
      $C(S^i) \leftarrow C(S^i) - \{v_c\} + z_{v_c}$
return $\hat{S}$

Análisis de PTAS para (mTSPHT*) en la araña

1. **Complejidad del Algoritmo:**
   Para el problema con $M = \lceil \frac{1}{\varepsilon} \rceil$, la creación del conjunto de ramas $\hat{B}$, junto con sus pesos y el computo de H es de orden $O(N)$.

   Si $|\hat{B}| \leqslant M$ entonces resolver el problema con F_M toma $O(mN^{2M})$, y como $m \leqslant N$, entonces podemos decir que toma $O(N^{2M+1}) = O(N^{\frac{2}{\varepsilon}+3})$. Poro otro lado, si $|\hat{B}| > M$, necesitamos saber cuanto toma GreedyAssignment-3, pero es fácil ver que es $O(MN) = O(N^2)$ (recordar que $M \leqslant N$), ya que se itera en torno al número de ramas y entre 1 a M.

   Generar el sub-problema no toma más que $O(N^2)$ y resolver el problema con $F_{|B_{sub}|} = F_{3M}$ toma a lo más $O(mN^{2(3M)}) = O(mN^{6M}) = O(N^{6M+1})$.

   Reconstruir la solución, con ayuda de Z toma $O(m(2M)) = O(mN)$. Finalmente el algoritmo toma en total $O(N^{6M+1}) = O(N^{\frac{6}{\varepsilon}+7})$. Por lo que para tiempo de completación, el algoritmo toma tiempo $O(N^{\frac{24m}{\varepsilon}+7}) = N^{O(\frac{m}{\varepsilon})}$, y para el tiempo de ruta, el algoritmo toma tiempo $O(N^{\frac{6m}{\varepsilon}+7}) = N^{O(\frac{m}{\varepsilon})}$.

2. **Correctitud del Algoritmo:**
   Vamos a empezar por la correctitud de GreedyAssignment-3 cuando es llamado en PTAS para (mTSPHT*) en la araña. Observe que si ordenamos las ramas por pesos, la M-ésima rama pesa a lo más $\frac{mH_m}{M}$, esto es debido a que el peso total de la red es de mH_m. Por lo que las ramas restantes pesan a lo más $\frac{mH_m}{M}$. Se agrupan las ramas de manera que el peso del cúmulo de ellas sea menor o igual a $\frac{mH_m}{M}$. Se necesitan a lo más $2M$ grupos, ya que si hay por lo menos $2M+1$ grupos al menos 2 de ellos pesarían menos que $\frac{mH_m}{2M}$ pero se podrían juntar en sólo uno, reduciendo la cantidad de grupos totales. Luego, como GreedyAssignment-3 va llenando cada grupo en orden, y admite hasta $2M$ grupos, es posible agrupar todas las ramas entregadas en el input sin dejar ninguna fuera y cada grupo pesa a lo más $\frac{mH_m}{M}$. A continuación por cada grupo, se añade una rama correspondiente a una arista con un cliente, de manera que que el tiempo de completación del cliente sea el mismo que el del problema y la distancia del *depósito* al cliente agregado es igual al peso del grupo menos el tiempo de completación, por lo que la rama agregada tiene un peso equivalente al grupo que la conforma. Así, GreedyAssignment-3, cuando es llamado desde el algoritmo entrega un conjunto de ramas, donde cada una de ellas pesa a lo más $\frac{mH_m}{M}$.

   Volviendo al algoritmo principal, si la cantidad de ramas es menor que M, entonces resolvemos a optimalidad. Por otro lado, si la cantidad de ramas es mayor que M, luego de utilizar GreedyAssignment-3, generar el nuevo problema y resolverlo, vamos a estudiar el valor de la solución óptima del problema (modificado), con respecto a la solución óptima del problema original. Para ello, vamos a separar nuestro análisis de acuerdo a la función óptima a considerar:

   - **Tiempo de Ruta:**
     Probaremos que la diferencia entre la solución del problema modificado y el problema original es menor o igual que $\frac{mH_m}{M}$.

     Claim: Si llamamos $\hat{S}_{\text{OPT}}$ a la solución óptima del problema original, entonces existe una solución del problema modificado $\hat{R}_{\text{mod}}$ tal que

     $$Tr_{\text{mod}}(\hat{R}_{\text{mod}}) \leq Tr_{\text{original}}(\hat{S}_{\text{OPT}}) + \tfrac{mH_m}{M},$$

     donde Tr_{original} calcula el tiempo de ruta en el problema original y Tr_{mod} el tiempo

de ruta del problema modificado.

Demostración. Sea $\hat{S}_{\text{OPT}}$ la solución óptima del problema original. Considere las ramas que fueron agrupadas. Es decir, $\{B_{M+1}, B_{M+2}, \ldots, B_{|\hat{B}|}\}$. Vamos a definir las cantidades A_i como el tiempo de ruta del vehículo i en las ramas $\{B_{M+1}, B_{M+2}, \ldots, B_{|\hat{B}|}\}$, es decir,

$$A_i = \sum_{B \in \{B_{M+1}, B_{M+2}, \ldots, B_{|\hat{B}|}\}} [Tr_{\text{original}}(S^{i,B}_{\text{OPT}})]$$

Ahora, vamos a construir un problema con los $A'_i s$ y las ramas de clientes *clusters* $\hat{B}'$. Por cada rama $B \in \hat{B}'$ vamos a construir un nodo v_B y vamos a darle una utilidad $U(v_B) = \text{peso}(B)$. El objetivo del problema es asignar cada nodo sólo uno de los m vehículos, de manera de minimizar:

$$\max_{k \in \{1,\ldots,m\}} \sum_{v: v \text{ fue asignado a k}} [U(v)] - A_k$$

Lo cual puede ser interpretado como en cuanto aumenta la solución óptima si cambiamos la solución en las ramas $\{B_{M+1}, B_{M+2}, \ldots, B_{|\hat{B}|}\}$, por alguna asignación de clientes *cluster* a algún vehículo.
Si bien no vamos a resolver el problema, vamos a probar que existe una solución tal que

$$\max_{k \in \{1,\ldots,m\}} \sum_{v: v \text{ fue asignado a k}} [U(v)] - A_k \leq \frac{mH_m}{M}$$

Definiremos A como:

$$A = \sum_{B \in \{B_{M+1}, B_{M+2}, \ldots, B_{|\hat{B}|}\}} [\text{peso}(B)]$$

Entonces, se tiene que $A < \sum_{k \in 1,\ldots,m} A_k$, pues en la suma de los A_k se pueden estar contando más de dos veces cada arista, mientras que en A cada arista se cuenta sólo 2 veces. Además, se tiene que:

$$A = \sum_{k \in \{1,\ldots,m\}} \sum_{v: v \text{ fue asignado a k}} [U(v)]$$

Luego, considere una solución factible T del problema

$$A \leqslant \sum_{k \in 1,\ldots,m} A_k$$

$$\sum_{k \in \{1,\ldots,m\}} \sum_{v: v \text{ fue asignado a k}} U(v) \leq \sum_{k \in 1,\ldots,m} A_k$$

$$\sum_{k \in \{1,\ldots,m\}} \sum_{v: v \text{ fue asignado a k}} [U(v)] - \sum_{k \in 1,\ldots,m} A_k \leq 0$$

$$\sum_{k \in \{1,\ldots,m\}} \Big[\sum_{v: v \text{ fue asignado a k}} [U(v)] - A_k \Big] \leq 0$$

Por lo tanto, existe al menos índice k tal que

$$\sum_{v:v \text{ fue asignado a k}} [U(v)] - A_k \leq 0$$

Ahora, considere una solución factible T del problema, observe que siempre podemos asignar todos los vértices al vehículo 1 y con ello tenemos una solución factible. Suponga que en T existe un índice k tal que

$$\sum_{v:v \text{ fue asignado a k}} [U(v)] - A_k > \frac{mH_m}{M}$$

Entonces, por la proposición anterior existe j tal que

$$\sum_{v:v \text{ fue asignado a j}} [U(v)] - A_j \leq 0$$

Tomando el nodo con mayor utilidad de k, lo vamos a asignar al vehículo j, observe que $\sum_{v:v \text{ fue asignado a k}}[U(v)]-A_k$ disminuyó, mientras que $\sum_{v:v \text{ fue asignado a j}}[U(v)]-A_j$ aumentó. Sin embargo, sigue siendo menor que $\frac{mH_m}{M}$, pues el peso máximo de los nodos esta acotado por $\frac{mH_m}{M}$. Repetimos el proceso mientras exista la índice k que genere problemas. Es posible realizarlo debido a lo demostrado.
El procedimiento termina, pues cada nodo es transferido a lo más 1 vez entre cada índice debido a que si un índice recibe un vértice en algún momento, el valor correspondiente a éste índice nunca va a sobrepasar la cota de $\frac{mH_m}{M}$.
Hemos probado que existe una solución tal que

$$\max_{k\in\{1,\ldots,m\}} \sum_{v:v \text{ fue asignado a k}} [U(v)] - A_k \leq \frac{mH_m}{M}$$

Ahora, volviendo al problema de ruteo, en el problema modificado vamos a crear una solución $\hat{R}_{\text{mod}}$ en donde cada vehículo tiene asignado a los clientes de $\{B_1, \ldots, B_M\}$ según $\hat{S}^{\text{OPT}}$ y en las ramas de $\hat{B}'$, tendrá asignados los clientes según la solución del problema de los $A'_k s$ (solución en la cual $\max_{k\in\{1,\ldots,m\}} \sum_{v:v \text{ fue asignado a k}}[U(v)]-A_k \leq \frac{mH_m}{M}$). Por lo tanto, el tiempo de ruta de cada vehículo aumenta a lo más en $\frac{mH_m}{M}$ con respecto al tiempo de ruta del mismo vehículo en $\hat{S}^{\text{OPT}}$.
Finalmente, se tiene que:

$$Tr_{\text{mod}}(\hat{R}_{\text{mod}}) \leqslant Tr_{\text{original}}(\hat{S}_{\text{OPT}}) + \frac{mH_m}{M}$$

□

Habiendo probado el **Claim** anterior, si llamamos $\hat{S}_{\text{mod}}$ a la solución óptima del problema modificado, se tiene que:

$$\begin{aligned} Tr_{\text{mod}}(\hat{S}_{\text{mod}}) &\leq Tr_{\text{mod}}(\hat{R}_{\text{mod}}) \\ &\leq Tr_{\text{original}}(\hat{S}_{\text{OPT}}) + \frac{mH_m}{M} \\ &\leq Tr_{\text{original}}(\hat{S}_{\text{OPT}}) + \frac{mTr_{\text{original}}(\hat{S}_{\text{OPT}})}{M} \end{aligned}$$

Para finalizar, basta notar que la construcción de la solución del problema original en base al problema modificado no altera su tiempo de ruta, por lo que el algoritmo es una $(1 + m\varepsilon)$-aproximación.
Por lo tanto, el algoritmo es una $(1 + \bar{\varepsilon})$-aproximación.

- **Tiempo de Completación:**
En este caso vamos a probar que la diferencia entre la solución del problema modificado y el problema original es menor o igual que $\frac{mH_m}{M} + \frac{mH_m}{2M}$. Para ellos recuerde el análisis del **Tiempo de Ruta**, en él se encontraba una solución $\hat{R}_{\text{mod}}$ que estaba a lo más $\frac{mH_m}{M}$ del óptimo, el único problema es que en este caso no se vuelve al depósito, por lo que necesitamos refinar un poco más el argumento.
Considere un vehículo k fijo, con respecto a la solución óptima del problema original puede ocurrir que en la solución original y en $\hat{R}_{\text{mod}}$ la rama con la mayor distancia (tiempo) que recorre k corresponde a una de las M primeras ramas. Si esto ocurre, el análisis del tiempo de ruta es el mismo pues todas las ramas que fueron cambiadas son recorridas de ida y vuelta.
De lo contrario, la distancia entre el *depósito* y el último cliente de la ruta k de la solución original y la distancia entre el *depósito* y el último cliente de la ruta k de $\hat{R}_{\text{mod}}$ difieren en a lo más la distancia del *depósito* al último cliente de la rama más larga de $\{B_{M+1}, \ldots, B_{|\hat{B}|}\}$, es decir, en a lo más $\frac{mH_m}{2M}$. Por lo que la solución óptima del problema modificado aumenta en a lo más $\frac{mH_m}{M} + \frac{mH_m}{2M}$ (debido a juntar las ramas en clientes *clusters* + cambiar la distancia del último cliente visitado y el *depósito*).
Sin embargo, nuestro análisis aún no termina, ya que ahora cuando generamos una solución del problema original, a partir de la solución óptima del problema modificado, puede volver a cambiar el valor de la solución: si en $\hat{S}_{\text{mod}}$ (solución óptima del problema modificado), la última rama que visita el vehículo k corresponde a una de las M primeras, el valor de la solución no cambia. Si la última rama que visita el vehículo k corresponde a una de las ramas de $\hat{B}'$ entonces al generar la solución del problema original puede aumentar en a lo más $\frac{mH_m}{2M}$.
En efecto, como el tiempo de completación se puede escribir como el tiempo de ruta menos la distancia que recorre el vehículo en la rama más larga, en este caso el valor disminuye, pero no puede disminuir mas allá que el largo de la rama del problema modificado, es decir, que$\frac{mH_m}{2M}$. Luego el valor de la solución puede volver aumentar en a lo más $\frac{mH_m}{2M}$. Con lo que tenemos que, si $\hat{R}_{\text{original}}$ es la solución generada a partir de $\hat{S}_{\text{mod}}$:

$$\begin{aligned} T_{\text{original}}(\hat{R}_{\text{original}}) &\leq T_{\text{mod}}(\hat{S}_{\text{mod}}) + \frac{mH_m}{2M} \\ &\leq (T_{\text{original}}(\hat{S}_{\text{original}}) + \frac{mH_m}{M} + \frac{mH_m}{2M}) + \frac{mH_m}{2M} \end{aligned}$$

Como consecuencia,

$$\begin{aligned} T_{\text{original}}(\hat{R}_{\text{original}}) &\leq T_{\text{original}}(\hat{S}_{\text{original}}) + \frac{2mH_m}{M} \\ &\leq T_{\text{original}}(\hat{S}_{\text{original}}) + \frac{4mH'_m}{M} \\ &\leq T_{\text{original}}(\hat{S}_{\text{original}})(1 + 4m\varepsilon) \end{aligned}$$

Finalmente,

$$T_{\text{original}}(\hat{R}_{\text{original}}) \leq T_{\text{original}}(\hat{S}_{\text{original}})(1 + 4m\varepsilon)$$

Luego, la solución encontrada por el algoritmo es una $(1 + 4m\varepsilon)$-aproximación. Por lo tanto, el algoritmo es una $(1 + \bar{\varepsilon})$-aproximación.

Capítulo 4

Conclusión

En el presente trabajo se estudio (mTSPHT*) en el camino, en el ciclo y en el árbol. En primer lugar, se estudió el problema en el camino cuando el *depósito* es uno de los extremos del camino, para luego estudiar el caso camino sin condiciones sobre el lugar donde se encuentra el *depósito*. A continuación, se estudió el problema en el ciclo, utilizando los resultados del problema en el camino. Y finalmente, se estudió el problema en la araña, se entregaron algoritmos de aproximación y un **PTAS** para ambas variantes del problema (minimizar el máximo tiempo de ruta y minimizar el máximo tiempo de completación).

En la primera parte de esta tesis se estudio (mTSPHT*) en el camino, en el caso particular cuando el depósito está en alguno de los extremos del camino. Este enfoque permitió encontrar una solución óptima estructurada para este caso: las rutas de esta solución tienen la *"propiedad de intervalos"*.
A continuación, se estudió el problema en el camino, sin establecer condiciones para el *depósito*. Se extendió la noción de *"propiedad de intervalos"* a cada ruta, pero restringida a cada rama del camino. Además, se definió una noción de orden entre las rutas de una solución. Luego, se encontró una solución óptima estructurada para este problema. Ésta solución, cumple que cada ruta, restringida a una rama, posee la *"propiedad de intervalos"* y además, existe un orden parcial entre las rutas de la solución. Gracias a este resultado se entregó un algoritmo que resuelve el problema en tiempo $O(mN^4)$.
Extendiendo éste resultado, se estudió (mTSPHT*) en el ciclo. Se demostró que ciertas soluciones del problema tienen la misma estructura que las soluciones óptimas estructuradas del problema en el camino. Es decir, algunas soluciones del problema en el ciclo, desde cierto punto de vista, poseen la *"propiedad de intervalos"* y el orden parcial. Para aquellas soluciones del problema que no posean la estructura anterior, se encontró una forma rápida de calcularlas. Luego, mezclando estos resultados fue posible obtener un algoritmo polinomial que resuelve (mTSPHT*) en el ciclo en tiempo $O(mN^5)$.

La segunda parte de ésta tesis corresponde al estudio de (mTSPHT*) en la araña. Se demostró que para cualquier $m \geqslant 2$, (mTSPHT*) es **NP-difícil**. Además, se extendieron los resultados del camino. Es decir, se probó que existe una solución óptima del problema en la

cual cada ruta, restringida a una rama, posee la *"propiedad de intervalos"* y que existe un orden parcial entre las rutas de la solución. Gracias a esto se entregó un algoritmo, que para un número de ramas fijo es polinomial, para resolver problema.
A continuación, se entrego un $\frac{5}{3}$-algoritmo de aproximación para (mTSPHT*) en la araña, con $m = 2$, minimizando el tiempo de completación y un $\frac{3}{2}$-algoritmo de aproximación para (mTSPHT*) en la araña, con $m = 2$, minimizando el tiempo de ruta.
Para finalizar, se entregó un $(1 + \varepsilon)$-algoritmo de aproximación en tiempo $N^{O(m/\varepsilon)}$ para el tiempo de completación y el tiempo de ruta.

En ésta memoria se abarcaron redes de cliente que tienen forma de camino, ciclo y araña. Como trabajo a futuro, se podría tratar de extender la idea de buscar una solución estructurada en el árbol. Es decir, ver si se sigue cumpliendo alguna propiedad tipo *"propiedad de intervalos"* y, si fuera posible un algún tipo de orden. Luego, en base a ello crear un algoritmo que resuelva el problema.
Si se desea estudiar la variante correspondiente al tiempo de ruta, quizás sea una mejor idea estudiar el problema mTSP (Vea Anexo A), por lo que el estudio de mTSPHT* en el árbol, para el tiempo de completación, es uno de los posibles temas de estudio a futuro.

Apéndice A

Reducción de mTSPHT a mTSP en árboles: variante tiempo de ruta

Se probará que resolver el problema, minimizando el tiempo de ruta, en un árbol se puede reducir a resolver (mTSP) tan sólo duplicando la cantidad de clientes, incluso cuando los tiempo de visita de cada cliente son distintos.

Proposición A.1 *Si podemos resolver (*mTSP*), minimizando el tiempo de ruta, en tiempo* $g(N)$*. Entonces podemos resolver* (mTSPHT) *en* $g(2N)$*, en dónde* N *es el número de clientes a servir.*

Demostración. Consideremos una instancia I de (mTSPHT) en un árbol y a partir de ella generaremos la siguiente red de *clientes-depósito*: por cada cliente v vamos a añadir un nuevo vértice u_v, junto con la arista $e = \{v, u_v\}$ con $t(e) = \frac{t(v)}{2}$ (recordar que $t(v)$ es el tiempo de visita del vértice v), con lo que la nueva red tiene el doble de clientes a visitar. Ahora, el problema que vamos a resolver será el de mTSP en la red de *clientes-depósito*, problema al que denotaremos por I'.
Vamos a demostrar que para toda solución de I existe una solución de I' del mismo valor. En efecto, considere una solución $\hat{S}$ del problema I, vamos a crear una solución del mismo valor, $\hat{S}'$, en el problema I' de la siguiente forma: sea $\hat{S} = (S^1, \dots S^m)$, $S^k = s_0^k s_k^k \dots s_{n_k}^k s_{n_k+}^k 1$, con $s_0^k = s_{n_k+1}^k = D$ y s_{n_k} el último cliente visitado por el vehículo k , para cada k. Definiremos $\hat{S}' = (S'_1 \dots S'_m)$, con $S'_k = s_0^k s_1^k v_{s_1^k} s_2^k v_{s_2^k} \dots s_{n_k}^k v_{s_{n_k}^k} s_{n_k+1}^k$, para cada k.
Observe que $t(v_{s_i^k}, s_{i+1}^k) = t(s_i^k, s_{i+1}^k) + \frac{t(v)}{2}$, pues para ir al cliente s_{i+1}^k desde $v_{s_i^k}$, primero se debe ir a s_i^k y luego a s_{i+1}^k. Ahora, veamos cuanto es el tiempo de ruta de S'_k.

$$
\begin{aligned}
Tr_{I'}(S'^k) &= \sum_{i=0}^{2n_k+2} t(s_i'^k, s_{i+1}'^k) \\
&= \sum_{i=1}^{n_k+1} [t(s_i^k, v_{s_i}) + t(v_{s_i^k}, s_{i+1}^k)] + t(D, s_1^k) \\
&= \sum_{i=1}^{n_k+1} [\frac{t(s_i^k)}{2} + \frac{t(s_i^k)}{2} + t(s_i^k, s_{i+1}^k)] + t(D, s_1^k) \\
&= \sum_{i=1}^{n_k+1} [t(s_i^k, s_{i+1}^k) + t(s_i^k)] + t(D, s_1^k) \\
&= Tr_I(S^k)
\end{aligned}
$$

Por lo tanto, para cualquier solución de I, hay una solución de mismo valor en I'.
Suponga ahora que $\hat{S}'$ es una solución óptima de I'. Vamos a armar una solución $\hat{S}$ en I de un valor menor o igual, de la siguiente manera: para $v \in C(S'^k) \cap V$ (según el orden de visita de la solución S'^k), si u_v también esta en $C(S'^k)$, vamos a añadir v a $C(S^k)$. Esto lo realizaremos para cada $k \in \{1, \ldots, m\}$.
Observe que cada cliente en V esta cubierto por al menos un vehículo en $\hat{S}$, en efecto, suponga que existe un cliente $v \in V : v \notin C(S^k), \forall k \in \{1, \ldots, m\}$, entonces u_v no es cubierto por ninguna ruta en $\hat{S}'$, pues de lo contrario v estaría cubierto por $\hat{S}$, lo cual no puede ser pues $\hat{S}'$ es una solución óptima.
El orden de visita de S^k vendrá dado por el orden que fueron agregados los clientes a la ruta, partiendo y terminando en el *depósito*. Por lo tanto, si $s_{r_j}'^k$ es el j-ésimo cliente que fue agregado,

$$
\begin{aligned}
Tr_{I'}(S_k') &\geq \sum_{i=1}^{|C(S'^k) \cap V|} [2t(s_{r_i}'^k, s_{r_{i+1}}'^k))] + 2t(s_{r_1}'^k, D) + \sum_{v \in C(S^k)} 2t(v, u_v) \\
&= \sum_{i=0}^{|C(S_k)|} [2t(s_i^k, s_{i+1}^k))] + \sum_{v \in C(S^k)} 2\frac{t(v)}{2} \\
&= Tr_I(S^k)
\end{aligned}
$$

Con lo que $Tr_I(\hat{S}) \leq Tr_{I'}(\hat{S}')$, pero por otro lado, teníamos que el óptimo de I era mayor o igual que el óptimo de I'. Por lo tanto, la solución $\hat{S}$ que derivamos a partir de $\hat{S}'$ es óptima en I.
Con lo anterior acabamos de probar que el valor óptimo en $\hat{S}'$ y en $\hat{S}$ es el mismo, y a su vez la forma de construir una solución óptima en I a partir de una solución óptima en I' en tiempo $O(2N)$, por lo que si podemos resolver (mTSP) en tiempo $g(N)$, entonces podemos resolver (mTSPHT) en tiempo $g(2N)$. □

Bibliografía

[1] Xiaoguang Bao and Zhaohui Liu. Approximation algorithms for single vehicle scheduling problems with release and service times on a tree or cycle. *Theor. Comput. Sci.*, 434:1–10, 2012.

[2] Amariah Becker and Alice Paul. A ptas for minimum makespan vehicle routing in trees. *ArXiv*, abs/1807.04308, 2018.

[3] V. Chvátal D. Applegate, R. Bixby and W. Cook. *The Traveling Salesman Problem - A Computational Study*. Princeton University Press, 2006.

[4] Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.

[5] Daya Gaur, Arvind Gupta, and Ramesh Krishnamurti. A 5/3-approximation algorithm for scheduling vehicles on a path with release and handling times. *Inf. Process. Lett.*, 86:87–91, 04 2003.

[6] Yoshiyuki Karuno and Hiroshi Nagamochi. A 2-approximation algorithm for the multi-vehicle scheduling problem on a path with release and handling times. In Friedhelm Meyer auf der Heide, editor, *Algorithms — ESA 2001*, pages 218–229, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[7] Yoshiyuki Karuno and Hiroshi Nagamochi. An approximability result of the multi-vehicle scheduling problem on a path with release and handling times. *Theoretical Computer Science*, 312:267–280, 01 2004.

[8] David P. Williamson and David Shmoys. The design of approximation algorithms. *The Design of Approximation Algorithms*, 01 2011.

[9] Liang Xu, Zhou Xu, and Dongsheng Xu. Exact and approximation algorithms for the min–max k-traveling salesmen problem on a tree. *European Journal of Operational Research*, 227(2):284–292, 2013.

[10] Wenzheng Xu, Weifa Liang, and Xiaola Lin. Approximation algorithms for min-max cycle cover problems. *IEEE Transactions on Computers*, 64:600–613, 03 2015.

[11] Zhou Xu and Liang Xu. Approximation algorithms for min-max path cover problems with service handling time. In Yingfei Dong, Ding-Zhu Du, and Oscar Ibarra, editors,

Algorithms and Computation, pages 383–392, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[12] Wei Yu and Zhaohui Liu. Vehicle routing problems on a line-shaped network with release time constraints. *Oper. Res. Lett.*, 37:85–88, 03 2009.

[13] Wei Yu and Zhaohui Liu. Improved approximation algorithms for min-max and minimum vehicle routing problems. In Dachuan Xu, Donglei Du, and Dingzhu Du, editors, *Computing and Combinatorics*, pages 147–158, Cham, 2015. Springer International Publishing.