Facultad de Ingeniería
Departamento de Ingeniería Industrial
Flujo en Redes – IIND 4115

Proyecto 1

Integrante 1: Nicolás Mateo Hernández Rojas ***Código:*** 201412420

Integrante 2: Nicolás Vásquez Murcia ***Código:*** 201424562

Parte 1 (50 puntos): Marguerite Shuttle

a) El problema de Marguerite puede ser planteado y modelado como una red que represente los movimientos de cualquier persona por medio de las diferentes estaciones y a lo largo de los horarios que se ofrecen. En específico, los dos componentes del modelaje son:

- Los nodos, que representan las paradas de los buses en las puertas de las estaciones a una específica. Por ejemplo, la parada que se hace a las 7:20 A.M. en la puerta *(Campus Dr)* de la estación *Li Ka Shing* por medio de la ruta *1050 Arastadero* es un nodo, y la parada que se hace en esa misma estación y puerta *Li Ka Shing Center (Campus Dr)* 20 minutos después (7:40 A.M.) es otro nodo diferente. Así mismo para cada hora de parada, en cada estación y con cada ruta. En particular se modelaron 3 tipos de nodo: los anteriormente explicados, los generales y los finales. Los **nodos generales** representan un nodo de una puerta en una estación sin diferenciar por hora ni ruta; i.e., *Main Quad (Memorial Church)* es un nodo general diferente a *Main Quad (Campus Oval Side)*. Finalmente, los **nodos finales** identifican una estación entera y no diferencia por puerta: *Main Quad* sería el nodo final relacionado con los nodos generales ejemplificados anteriormente.

- Los arcos son la conexión que permite al modelo representar el movimiento entre los nodos. Por construcción, entonces, cada arco puede representar el desplazamiento entre dos paradas al usar el bus de una ruta, el trasbordo en una misma estación hacia distintas puertas para tomar otras rutas, o la espera de un usuario moviéndose entre nodos que sólo se diferencian por la hora. En particular, todos los nodos específicos (aquellos que tienen hora, puerta, estación y ruta) tienen un arco hacia el nodo general relacionado; y, todos los nodos generales (aquellos que son estación y puerta) tienen arcos que los conectan hacia el nodo final de la estación. El costo[1] entre los arcos es el tiempo que toma dicho desplazamiento, espera o trasbordo. Es importante notar que el uso de los nodos generales y finales sólo son usados para modelar trasbordos entre puertas y las **llegadas a una estación** (dado que no se conoce la hora de llegada de un viaje), por lo que físicamente son lo mismo y no toma ningún tiempo moverse de nodos específicos a generales, ni de generales a finales ($costo = 0$). Por esto, de los nodos finales **no** salen arcos.

- No hay nodos 'iniciales' porque un viaje sí tiene hora de inicio, y puede identificarse como nodo inicial aquel que esté en la estación y puerta de partida a la hora más cercana posible (e.g. el nodo inicial de un viaje que empieza a las 5:40 en *Main Quad (Memorial Church)* es la parada en esa puerta de la estación con la ruta *C* a la hora 5:41 A.M.).

Puede visualizarse (véase *Ilustración 1* en *Anexos*) un **fragmento** del grafo modelado para el problema. Es notable que existen nodos finales, generales y específicos en el fragmento, y que así mismo hay un nodo que representa, en forma de ejemplo, un trasbordo entre dos rutas en la parte inferior izquierda de la ilustración: 'Li Ka Shing Center (Campos Dr) 7:13 AM Ruta C' es un trasbordo hacia la Ruta C desde el nodo 'Li Ka Shing Center (Campos Dr) 7:00 AM Ruta 1050 Arastadero'.

b) Formulación del problema de programación lineal:

- Conjuntos y nodos especiales:

$$G: nodos\ generales$$
$$F: nodos\ finales$$
$$E: nodos\ específicos$$
$$N: conjunto\ de\ nodos\ \{G \cup F \cup E\}$$
$$A: conjunto\ de\ arcos\ (i,j)\ \forall\ i,j \in N$$
$$S: nodo\ de\ partida \in N$$
$$T: nodo\ de\ llegada \in N$$

- Parámetros:

$$C_{i,j}: tiempo\ en\ minutos\ del\ movimiento\ (i,j) \in A$$
$$b_i: oferta/demanda\ de\ personas\ desde/para\ un\ nodo\ i \in N$$

- Valores particulares de los parámetros:

$$C_{i,j} = 0\ \forall\ (i,j) \in A \mid (j \in F\ \vee\ j \in G)$$
$$b_i = 1\ \forall\ i \in N \mid (i = S)$$

[1] El costo de los arcos está en unidades de minutos. No hay costos negativos.

$$b_i = -1 \; \forall \; i \in N \mid (i = T)$$
$$b_i = 0 \; \forall \; i \in N \mid (i \neq T \wedge i \neq S)$$

- Variables de decisión:

$$x_{i,j}: binario\ de\ si\ debe\ realizarse\ (1)\ o\ no\ (0)\ el\ movimiento\ que\ representa\ el\ arco\ (i,j) \in A$$

- Restricciones:

$$Balance: \sum_{(i,j)\in A} x_{i,j} - \sum_{(k,i)\in A} x_{k,i} = b_i \; \forall \; i \in N$$

- Función objetivo:

$$\min \sum_{(i,j)\in A} x_{i,j} * C_{i,j}$$

Puede evidenciarse que la formulación sobre el modelaje del problema de Marguerite representa un problema sobre flujo en redes de **ruta más corta**. Por esta razón no es necesario establecer restricciones de capacidad, porque el hecho de enmarcar el problema con sólo una unidad de oferta y demanda en los nodos respectivos (S y T) permite que ningún arco tome un valor mayor a 1 o menor a 0, comportándose como la variable binaria que, en efecto, queremos modelar.

En el archivo de soporte '*Punto1.mos*' está el procedimiento computacional que resuelve el problema de ruta más corta dado unos nodos específicos de partida y de llegada. El proceso necesita un archivo de entrada '*b.txt*' que contiene la referencia a ambos nodos y un archivo de salida '*Ruta más corta.txt*' que contendrá el plan óptimo y el recorrido mínimo para llegar al destino. Es imperativo recordar que el nodo de partida debe ser un **nodo específico** (porque debe contener una hora de salida) y el de llegada un **nodo final**. Con esto, se otorga el archivo de soporte '*Grafo Marguerite.xlsm*' que tiene en la hoja '*Equivalencia de nodos*' la lista de todos los nodos modelados en el problema, el número que debe estar en el archivo de entrada '*b.txt*' es el registrado en la columna '*Fila*' del archivo del grafo.

c) El recorrido óptimo encontrado para los diferentes escenarios fue:

- **Escenario 1:**

  Parámetros: $(Estación\ origen = Main\ Quad\ (Campus\ Oval\ Side); Hora\ de\ salida = 7{:}03\ a.m.; Estación\ destino = Palo\ Alto\ Transit\ Center)$, en $b.txt\ (inicio, 216 \quad final, 1398)$:

  Recorrido óptimo: **(1)** Pasa por estación Main-Quad-(Campus-Oval-Side) a las 7:13 AM tomando la ruta C **(2)** Pasa por estación Li-Ka-Shing-Center-(Campus-Dr) a las 7:16 AM tomando la ruta C **(3)** Trasbordo en estación Li-Ka-Shing-Center-(Campus-Dr) a las 7:20 AM tomando la ruta 1050-Arastradero **(4)** Pasa por estación Y2E2-(Via-Ortega) a las 7:22 AM tomando la ruta 1050-Arastradero **(5)** Trasbordo en estación Y2E2-(Via-Ortega) a las 7:35 AM tomando la ruta Y **(6)** Pasa por estación Medical-Center-(Roth-Way) a las 7:37 AM tomando la ruta Y **(7)** Pasa por estación Palo-Alto-Transit-Center-(Caltrain-Platform) a las 7:48 AM tomando la ruta Y.

  Tiempo total del recorrido: 45 minutos.

- **Escenario 2:**

  Parámetros: $(Estación\ origen = Oak\ Creek\ Apartments; Hora\ de\ salida = 7{:}25\ a.m.; Estación\ destino = Campus\ Dr)$, en $b.txt\ (inicio, 696 \quad final, 1392)$:

  Recorrido Óptimo: **(1)** Pasa por estación Oak-Creek-Apartments-(Across-Street-On-Sand-Hill-Rd) a las 8:01 AM tomando la ruta S **(2)** Pasa por estación Stanford-West-Apartments-(Clark-Way-Across-Street-From-Office) a las 8:03 AM tomando la ruta S **(3)** Trasbordo en la estación Stanford-West-Apartments-(Clark-Way) a las 8:04 AM tomando la ruta C **(4)** Pasa por estación Across-Street-From-Lucas-Center-(Welch-Rd) a las 8:08 AM tomando la ruta C **(5)** Pasa por estación Stock-Farm-Garage a las 8:14 AM tomando la ruta C **(6)** Pasa por estación Campus-Dr-(Across-Street-from-Li-Ka-Shing-Center) a las 8:17 AM tomando la ruta C.

  Tiempo total del recorrido: 52 minutos.

- **Escenario 3:**

  Parámetros: $(Estación\ origen = Y2E2\ (Via\ Ortega); Hora\ de\ salida = 8{:}50\ a.m.; Estación\ destino = 1000\ Welch\ Rd)$, en $b.txt\ (inicio, 74 \quad final, 1402)$:

  Recorrido óptimo: **(1)** Pasa por estación Y2E2-(Via-Ortega) a las 9:02 AM tomando la ruta 1050-Arastradero **(2)** Trasbordo en estación Y2E2-(Via-Ortega) a las 9:12 AM tomando la ruta Y **(3)** Pasa por estación Medical-Center-(Roth-Way) a las 9:14 AM

tomando la ruta Y **(4)** Pasa por estación Palo-Alto-Transit-Center-(Caltrain-Platform) a las 9:25 AM tomando la ruta Y **(5)** Trasbordo en estación Palo-Alto-Transit-Center-(Caltrain-Platform) a las 9:37 AM tomando la ruta X **(6)** Pasa por estación Stanford-Shopping-Center-(Quarry-Rd-Near-Pear-Ln) a las 9:40 AM tomando la ruta X **(7)** Trasbordo en la estación Stanford-Shopping-Center-(Quarry-Rd-Near-Pear-Ln) a las 9:43 AM tomando la ruta Medical Center **(8)** Pasa por estación Stanford-Barn-(700-Welch-Rd) a las 9:46 AM tomando la ruta Medical Center **(9)** Pasa por estación 780-Welch-Rd a las 9:47 AM tomando la ruta Medical Center **(10)** Pasa por estación 1000-Welch-Rd a las 9:48 AM tomando la ruta Medical Center.

Tiempo total del recorrido: 58 minutos.

- **Escenario 4:**

Parámetros: $(Estación\ origen = Palo\ Alto\ Transit\ Center; Hora\ de\ salida = 9{:}01\ a.m.; Estación\ destino = Oak\ Creek\ Apartments)$, en $b.txt$ $(inicio, 708 \quad final, 1410)$:

Recorrido Óptimo: **(1)** Pasa por estación Palo-Alto-Transit-Center-(Bus-Depot-Drop-Off) a las 9:03 AM tomando la ruta S **(2)** Pasa por estación Palo-Alto-Transit-Center-(Bus-Depot-Pick-Up) a las 9:04 AM tomando la ruta S **(3)** Pasa por estación Crate-&-Barrel-(Quarry-Rd) a las 9:07 AM tomando la ruta S **(4)** Pasa por estación Sand-Hill-Rd-@-Clark-Way a las 9:09 AM tomando la ruta S **(5)** Pasa por estación Oak-Creek-Apartments-(Sand-Hill-Rd) a las 9:10 AM tomando la ruta S.

Tiempo total del recorrido: 9 minutos.

- **Escenario 5:**

Parámetros: $(Estación\ origen = 3170\ Porter\ Dr; Hora\ de\ salida = 9{:}10\ a.m.; Estación\ destino = 3475\ Deer\ Creek\ Rd)$, en $b.txt$ $(inicio, 648 \quad final, 1409)$:

Recorrido Óptimo: **(1)** Pasa por estación 3170-Porter-Dr a las 9:13 AM tomando la ruta Research-Park **(2)** Pasa por estación Hillview-Ave-@-Coyote-Hill-Rd a las 9:17 AM tomando la ruta Research-Park **(3)** Pasa por estación 1050/1070-Arastradero-Rd a las 9:19 AM tomando la ruta Research-Park **(4)** Pasa por estación 3475-Deer-Creek-Rd a las 9:26 AM tomando la ruta Research-Park.

Tiempo total del recorrido: 16 minutos.

Nota: el archivo de salida arroja un tiempo total de recorrido desde el punto (1) hasta el punto final, por lo que debe sumársele el tiempo entre la hora de salida y el primer nodo visitado. Los tiempos totales de los escenarios registrados anteriormente en este documento ya tienen este aspecto en cuenta y representan el tiempo neto del recorrido. Además, los archivos salida de los 5 escenarios están adjuntos al entregable con el nombre '*Escenario X.txt*' donde X es el número del escenario.

d) Para determinar la cantidad de gente que puede desplazarse entre las estaciones *Main Quad (Memorial Church)* y *Palo Alto Transit Center* se reformuló el problema modelado para que dejara de ser uno de ruta más corta, sino de flujo máximo. Esto permite conocer cuál es la mayor cantidad de flujo que puede enviarse desde un punto en particular hasta otro en algún horario específico. Lo anterior implica algunos cambios en el modelaje:

- Todos los costos son 0 sin importar el tipo de nodo al que llega el arco.

$$C_{i,j} = 0 \ \forall \ (i,j) \in A$$

- La oferta y demanda de todos los nodos es de 0 porque no debe establecerse un objetivo de transporte, sino que permitir al algoritmo transportar lo máximo posible.

$$b_i = 0 \ \forall \ i \in N$$

- Se agrega un arco nuevo que va desde el nodo de llegada hasta el nodo de salida y va a ser el único que tenga un costo diferente a 0. En particular va a tener un costo de -1 para que la función de optimización busque enviar todo lo que pueda por allí con el fin de minimizar el objetivo.

$$C_{i,j} = -1 \ \forall \ (i,j) \in A \mid (i = T \ \wedge j = S)$$

- Con el fin de imponer el límite de movilización a las 8:00 a.m. se eliminan todos los arcos que conectan los nodos específicos que tienen horarios mayores con los generales respectivos.

- Se agrega un con junto de restricciones extra que permiten modelas la capacidad de los buses y el número máximo de pasajeros que puede transportarse por medio de los arcos.

$$U_{i,j}: número\ máximo\ de\ pasajeros\ del\ movimiento\ (i,j) \in A$$
$$U_{i,j} = 70 \ \forall \ (i,j) \in A$$

$$U_{i,j} = \infty \; \forall \; (i,j) \in A \mid (i = T \; \wedge j = S)$$
$$Capacidad: x_{i,j} \leq U_{i,j} \; \forall \; (i,j) \in A$$

El archivo de soporte '*Punto1fMax.mos*' modela computacionalmente el problema y su solución de optimización lineal. Usa el mismo archivo de entrada '*b.txt*' pero genera un archivo de salida '*Flujo Máximo.txt*' con los valores de los arcos y el flujo máximo entre los nodos especificados de salida y llegada.

Solución de evacuación

Entre las 7:00 a.m. y las 8:00 a.m. el flujo máximo desde la estación Main Quad (Memorial Chuch) hasta la estación Palo Alto Transit Center **es de 0 personas**. No hay rutas que logren llegar desde esa estación de partida al destino en cuestión antes de las 8:00 de la mañana, por lo que no se puede desplazar ni evacuar personas en este intervalo de tiempo específico. **Parámetros en el archivo '*b.txt*':** $(inicio, 15\, final, 1398)$.

Parte 2 (40 puntos): MOPTA 2018

Para este punto, primero se planteó el grafo como un problema de flujo en redes de ruta más corta para encontrar el menor costo posible entre la bodega y los clientes para que solo se pase una sola vez por cada cliente. Ahora bien, como era necesario tener en cuenta que cada cliente tenía una demanda, también era necesario formular un problema de flujo de costo mínimo donde la se tuvieran en cuenta las ofertas como cada vehículo y las demandas en cada cliente. Como estos dos problemas son diferentes, pero ambos de flujo en redes, se utilizó una formulación de flujo en redes con "*comodities*" utilizando la descomposición de Dantzig-Wolfe para resolverlo utilizando el algoritmo de Simplex en vez de un problema entero MIP, gracias a que Simplex es mucho más eficiente computacionalmente.

En primer lugar, se planteó el grafo del problema. Para esto, se creó un nodo por cada vehículo, con el motivo de representar de una mejor manera que se pueden generar varias rutas diferentes. Adicionalmente, cada uno de estos nodos oferta una cantidad de helados b_i igual a la capacidad máxima del vehículo. Adicionalmente, se creó un nodo por cada cliente, donde cada uno tiene una demanda b_i (valor negativo). Por último, y dado que la suma de la capacidad de todos los vehículos disponibles es mayor a la demanda total, se crea un nodo final a donde se dirigen todas las unidades que sobran. Cabe resaltar que ningún arco sale de este nodo y que para todo nodo existe un arco hacia este último.

Ahora bien, en este grafo es necesario considerar dos problemas diferentes. Uno, en el que un solo vehículo puede pasar por cada nodo, y que con la menor cantidad de vehículos posible se deben recorrer todos los clientes con el mínimo costo posible. Por lo tanto, se definieron los costos entre cada arco del grafo como: $c_{ij} = L_{ij} * m_v + T_{ij} * m_t \; \forall \; i,j \; Nodos \backslash vehiculos \; \cup UnidadesSobra$. Por otro lado, para todo grafo entre un vehículo y un cliente se adiciona un costo fijo: $c_{ij} = L_{ij} * m_v + T_{ij} * m_t + M_F \; \forall \; i \in vehiculos \; , j \in Clientes$. Puesto que esto simula que se incurre en un costo fijo para el vehículo i, en caso de que este sea utilizado en el problema. Por otro lado, esto implica que para los arcos entre los vehículos y el nodo de sobra el costo es 0, debido a que si hay flujo por este arco significa que el vehículo no fue utilizado: $c_{ij} = 0 \; \forall \; i \in vehiculos \; , j = UnidadesSobra$. No obstante, este costo solo está asociado al transporte de vehículos, puesto que no existe un costo asociado a transportar un número específico de helados. Por último, el costo desde cualquier cliente hasta el nodo de unidades de sobra es igual al costo de ir desde cualquier cliente hacia la bodega, puesto que este arco representa que se acaba la ruta para el vehículo y retorna a la bodega. Este grafo, se muestra en el Anexo 2 (*Ilustración* 2). Adicionalmente, los parámetros de este problema brindados por MOPTA, fueron procesados en el proyecto de Eclipse llamado "proyecto1FlujoEnRedes". Al ejecutar el archivo Solucion2.java, se generan los parámetros que recibe Xpress para resolver el problema. Cabe resaltar, que se creó un arco entre la bodega y los clientes 4 y 6 tomando el mínimo costo para llegar a estos nodos, pasando por algún otro cliente, esto, con el motivo de que los vehículos no tuvieran que pasar más de una vez por un cliente.

Ahora bien, se deben formular dos problemas sobre este grafo. Uno, el problema de suplir la demanda para todos los nodos, puesto que es un requisito para el problema, aunque esto no cambia los costos de la función objetivo. Además, se debe cumplir que para todo cliente solo debe haber un vehículo que lo visite, excepto el nodo final a donde llegan todos los vehículos. Por otro lado, se debe enviar una cantidad determinada de helados hacia cada cliente, tomando en cuenta la capacidad de cada vehículo. Se debe cumplir que lo que llega a cada cliente, menos lo que sale debe ser igual a lo que este demande de helados. Por lo tanto, tenemos una formulación de flujo en redes con dos "*comodities*" una, los vehículos y otra, los helados que se mueven por cada arco. Por lo tanto, tenemos la siguiente formulación:

Conjuntos:

$$N: Nodos$$
$$A: Arcos$$
$$Nv: Subconjunto\ de\ Nodos\ que\ son\ vehículos$$
$$Nc: Subconjunto\ de\ Nodos\ que\ son\ clientes$$
$$K: \{\text{Vehículos,helados}\}$$

Parámetros:

$$c_{ij}: Costo\ de\ mover\ un\ vehículo\ de\ nodo\ i \in N\ a\ j \in N$$
$$n: Número\ de\ clientes$$
$$V: Número\ de\ vehículos$$
$$b_{j,vehiculo}: Parámetro\ para\ lograr\ ecuación\ de\ balance\ j \in N$$
$$b_{j,helados}: Oferta\ o\ demanda\ \ de\ cada\ nodo\ j \in N$$
$$C: Capacidad\ máxima\ de\ los\ vehículos$$

Variable de decisión

$$x_{ijk}: Cuánto\ flujo\ debe\ pasar\ entre\ nodo\ i \in N\ y\ nodo\ j \in N\ del\ comodity\ k$$

Restricciones:

1. Restricción de balance para todos los nodos:

$$\sum_{j|(i,j)\in A} x_{ijk} - \sum_{j|(j,i)\in A} x_{jik} = b_{ik} \ \ \forall\ i \in N, k \in K$$

2. Restricción para vehículos. Solo puede entrar y salir un arco de cada nodo. Es decir, solo debe pasar un vehículo por cada cliente

$$\sum_{j|(i,j)\in A} x_{ij,vehiculos} = 1\ \forall\ i \in N$$

$$\sum_{j|(j,i)\in A} x_{ji,vehiculos} = 1\ \forall\ i \in N$$

3. Restricción para que solo haya flujo de helados por un arco sí y sólo sí hay un vehículo que pasa por el mismo arco:

$$x_{ij,helados} \leq x_{ij,vehiculos} * C \quad \forall\ i,j \in N$$

4. Naturaleza de las variables. Para un problema de flujo en redes, la unimodularidad y triangularidad de la matriz A, no es necesario definir como enteras las variables. Sin embargo, solo las restricciones 1 y 2 conforman matrices de este tipo, mientras que la restricción 3 arruina la estructura de la matriz. Por lo tanto, con esta formulación general MIP, en vez de tener:

$$x_{ijk} \geq 0$$

Tenemos que:

$$x_{ij,helados} \geq 0$$
$$x_{ij,vehiculos} \in \{0,1\}$$

Función objetivo

$$\min \sum_{i,j\ |(i,j)\in A} x_{ij,vehiculos} * c_{ij}$$

Para conservar la estructura de flujo en redes, es posible realizar la descomposición Dantzig-Wolfe. Donde se conserva la estructura de flujo en redes para problemas auxiliares más pequeños que se resuelven de manera iterativa, y generan columnas de la matriz A del problema general para encontrar una solución óptima para problemas escalables. Este problema, se soluciona utilizando los puntos extremos de la región convexa que forma el espacio de soluciones factibles de cada problema. v_j. De esta manera, la variable de decisión estará conformada por una combinación convexa dada por λ_j (el peso de cada punto extremo).

$$\boldsymbol{X} = \sum_j v_j \lambda_j$$

$$\min\ c^T * \sum_j v_j \lambda_j \quad s.a.$$
$$\boldsymbol{A_0} \sum_j \lambda_j\, v_j = 1$$

$$\sum_j \lambda_j = 1$$

$$\lambda_j \geq 0 \quad \forall\ j$$

En este caso, **A0** consiste en la porción de la matriz de restricciones del problema planteado anteriormente, conformado únicamente por la restricción 3, puesto que esta conecta los 2 subproblemas. Ahora bien, para el problema modificado (problema maestro) se pueden generar columnas factibles para la resolución del problema de manera eficiente, a partir de (en este caso) dos problemas auxiliares; uno para cada "$comodity$". A continuación, se muestra el problema auxiliar:

$$\min(c^T - {\pi_0}^T A_0) v_q \quad s.a.$$

Donde v_q es uno de los vértices de la región lineal convexa y ${\pi_0}^T$ corresponde al valor de las variables duales del problema maestro:

$$\boldsymbol{A_1 x = b_1}$$

Donde **A1** corresponde a la restricción 1 para el problema de los helados, y las restricciones 1 y 2 para el problema de los vehículos. Nótese que estos problemas auxiliares abstraen únicamente las restricciones que conforman la estructura de un problema de flujo en redes que se puede resolver por medio del algoritmo de Simplex, y por lo tanto se solucionan de manera muy eficiente y sencilla. Esta descomposición, permite solucionar estos problemas sencillos, y añadir columnas a la matriz del problema maestro fácilmente, para que, de manera iterativa se encuentre una solución al problema original, utilizando algoritmos de baja complejidad operacional, en vez de un algoritmo como $Branch\ and\ Bound$. Los problemas punto2Vehiculos.mos y punto2Helados.mos son los problemas auxiliares planteados y tardan muy pocas iteraciones y muy poco tiempo en encontrar el óptimo. En cambio, la solución con formulación entera MIP se demora más tiempo encontrando la solución óptima.

Por último, se encontró la solución óptima para el problema con los parámetros dados:

- Sale camión 1 hasta cliente 3, Se devuelve hacia la bodega desde cliente 3
- No se utiliza el camión 2
- Sale camión 3 hasta cliente 6, Va de cliente 6 hasta cliente 5, Se devuelve hacia la bodega desde cliente 5
- Sale camión 4 hasta cliente 9, Va de cliente 9 hasta cliente 1, Se devuelve hacia la bodega desde cliente 1
- No se utiliza el camión 5
- No se utiliza el camión 6
- Sale camión 7 hasta cliente 4, va de cliente 4 hasta cliente 7, va de cliente 7 hasta cliente 2, se devuelve hacia la bodega desde cliente 2
- Sale camión 8 hasta cliente 8, Va de cliente 8 hasta cliente 10, Se devuelve hacia la bodega desde cliente 10

El costo total de transporte sin incluir demoras es de $5931.33

Finalmente, la formulación de flujo en redes tiene muchas ventajas en el momento de resolver problemas de este tipo, gracias a que, por medio de una formulación simple, es posible obtener soluciones de manera eficiente, con un número de iteraciones bajo y que no crece tanto a medida que crece el problema (es decir, es escalable). Las soluciones con esta formulación suelen ser mucho más rápidas que problemas de optimización lineal utilizando variables enteras que resuelven el problema utilizando $Branch\ and\ Bound$, y todo gracias a una propiedad en la matriz de restricciones que permite relajar la naturaleza de las variables. Ahora bien, para un problema de un grafo, el número de nodos y arcos siempre va a crecer de manera exponencial, a medida que crezca el problema, sin embargo, el tiempo computacional no se verá tan afectado como en un problema de optimización lineal utilizando variables enteras. Adicionalmente, en caso de que se tengan redes con flujos de diferentes tipos, se pueden descomponer en diferentes problemas que conserven la formulación en Simplex, y de esta manera conservar la solución de manera eficiente. Sin embargo, una de las desventajas de la formulación de este tipo, es que no en todos los casos se puede formular, por lo que puede ser una solución muy útil, pero no se garantiza que resuelva todos los problemas.

Parte 3 (10 puntos): Revisión de literatura

Finding long chains in kidney exchange using the traveling salesman problem:

Adicionalmente, modifica este problema para llegar a una solución más eficiente, tal y como lo muestra en varios de sus resultados, comparando la solución de la formulación general TSP y la solución propuesta PC-TSP. El problema que desea solucionar consta de una estructura de un grafo, donde los nodos representan los donantes de órganos, y se propone maximizar la cantidad de trasplantes realizados. Es decir, se desea conectar de la mejor manera posible los donantes de órganos con pacientes que lo puedan necesitar, dado una prioridad para donar, donantes que desean donar a pacientes incompatibles, los datos de compatibilidad entre donantes y pacientes y el máximo número de conexiones con las que puede contar un ciclo.

En primer lugar, proponen una solución con una formulación TSP, con la variable binaria y_e que representa los arcos entre los diferentes nodos. Toma el valor de 1 si hay una conexión por el arco e y 0 de lo contrario, es decir que si toma valor se realiza una conexión entre un donante y un paciente. En esta formulación, encontramos la estructura de un problema de flujo en redes de las ecuaciones de balance (restricción [1], Anexo 3), en donde se garantiza que la cantidad de flujo de salida f_v^o para todos los arcos de un nodo v sea igual al flujo de entrada f_v^i. Además, esto varía de acuerdo con si es un paciente o no, lo cual es equivalente a la restricción de balance. Adicionalmente, cuenta con la restricción para eliminar sub-ciclos de longitud k, que hace que la solución conste de un algoritmo recursivo que corta los sub-ciclos en cada iteración para encontrar la solución óptima.

Por otro lado, proponen una mejor a esta formulación llamada PC-TSP, para que la solución tenga una menor complejidad computacional, manteniendo algunas restricciones de balance, pero cambiando la eliminación recursiva de sub-ciclos. Para esto, introducen una variable adicional Z_C para cada ciclo C que se esté utilizando en la solución. Esta variable toma el valor de 1 si se está utilizando y 0 de lo contrario. En primer lugar, adecúan una ecuación de balance, para que cuando el nodo v esté en un ciclo Z_C, los arcos y_e en el ciclo C no tomen valor. Por último, cambian la restricción que corta los ciclos de longitud k, utilizando la nueva variable. Esta restricción (restricción [7] Anexo 5) consiste en que siempre que haya un donante, y, por lo tanto, un flujo de entrada con el valor de 1 en un paciente en un subconjunto de pacientes S, debe haber un arco que tenga la cola afuera de S y la cabeza en S, de tal manera, se asegura que no existen sub-ciclos en S.

Finalmente, comparan esta solución incluyendo una nueva variable que incluye los ciclos, con la formulación original de TSP, como se muestra en la tabla S3 del artículo (Anexo 4) y encuentran que a medida que crece el tamaño del grafo, la solución recursiva crece en tiempo de ejecución exponencialmente y la formulación PC-TSP no lo hace en casi todos los casos. Por lo tanto, esta formulación es más eficiente para resolver problemas de sub-ciclos en redes.

Network models for vehicle and crew scheduling:

En el artículo de Carraresi et al. (1984) se evoca la importancia de los modelos en redes y su aplicación en distinta variedad de problemas que pueden ser solucionados eficientemente gracias a ellos. El tipo de problemas a los que hace alusión el artículo es aquel que involucra redes de

transporte que tienen un componente horario. Con esto se plantean diferentes modelos que pueden facilitar la formulación y solución de estos problemas para aprovechar el máximo la utilidad de los componentes de una red.

Principalmente busca definir los elementos de un grafo en función de las características de un problema horario de transporte. Por ejemplo, definen como viaje un conjunto de información que se conoce acerca de un movimiento entre dos estaciones cualquiera y que debe contener, por lo menos, tiempo de partida, duración, estación de origen y estación de destino. Así, se define entonces una propiedad entre los viajes que permite identificar dos de ellos compatibles si se cumple que el momento en que se llega al destino con uno de los viajes permite hacer abordaje al otro. Es decir, si v_i y v_j son pareja compatible de viajes si $tiempo\ de\ partida_i + duración_i + tiempo\ de\ trasbordo\ (i,j) \leq tiempo\ de\ partida_j$ en el grafo.

La idea de lo anterior es formar un **grafo bipartito** que tenga dos conjuntos principales de nodos (S y T) que representan los viajes como v_i o v_j en una pareja compatible de viajes (v_i, v_j), es decir, todo arco que haya entre un nodo de S y uno de T representa la compatibilidad entre los dos viajes conectados. Con un grafo que tenga un conjunto de nodos que sólo tenga arcos de salida, y otro que sólo tenga arcos de llegada, se puede utilizar estrategias de asignación, transporte o similares para facilitar el algoritmo de optimización. En particular, clasifica los arcos dentro de dos conjuntos A^* y A a los que les asigna costos 1 y 0 respectivamente para representar arcos de viajes que son compatibles (A) o no (A^*) con el fin de minimizar la función objetivo y representar decisiones de trasbordo o movimientos entre estaciones de manera efectiva.

Con todo lo anterior se permite modelar problemas de transporte con restricciones y límites de tiempo entre los movimientos de las unidades, ya sea con un depósito de abastecimiento o no. Además de lo anterior, mencionan la formulación que debería hacerse en caso de tener varios depósitos para diferenciar las unidades de cada uno de ellos por medio de distintas redes (modelo *multicomoditie*). Se le agrega, así mismo, componentes de turnos entre los recorridos que se hacen sobre el grafo modelado y que harán uso de descomposición eventualmente para poder involucrar las distintas asignaciones de conductores y tener subproblemas que acoten la solución final factible.

Referencias

Anderson, R., Ashlagi, I., Gamarnik, D., & Roth, A. E. (2015). Finding long chains in kidney exchange using the traveling salesman problem. Proceedings of the National Academy of Sciences of the United States of America. 112(3): 663–668.

Carraresi, P., Gallo, G. (1984). Network models for vehicle and crew scheduling. European Journal of Operational Research. 16(2): 39-151.

Anexos

Anexo 1:

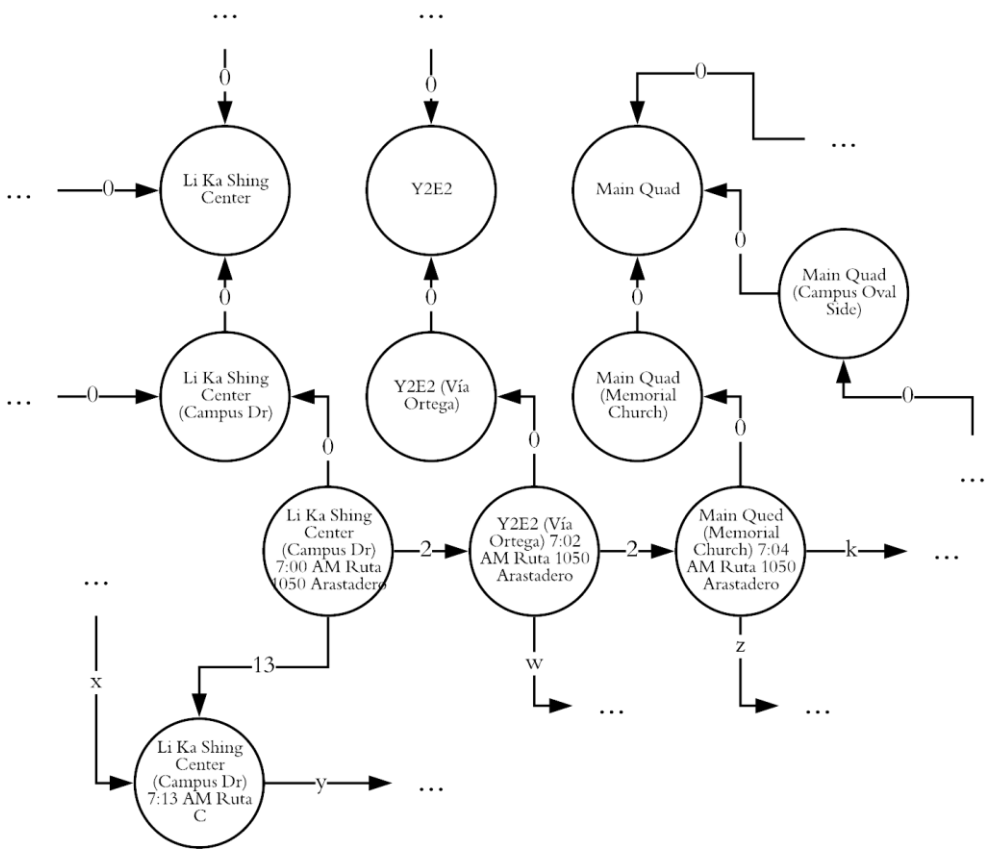

Ilustración 1. Fragmento del grafo modelado para Marguerite.

Anexo 2:

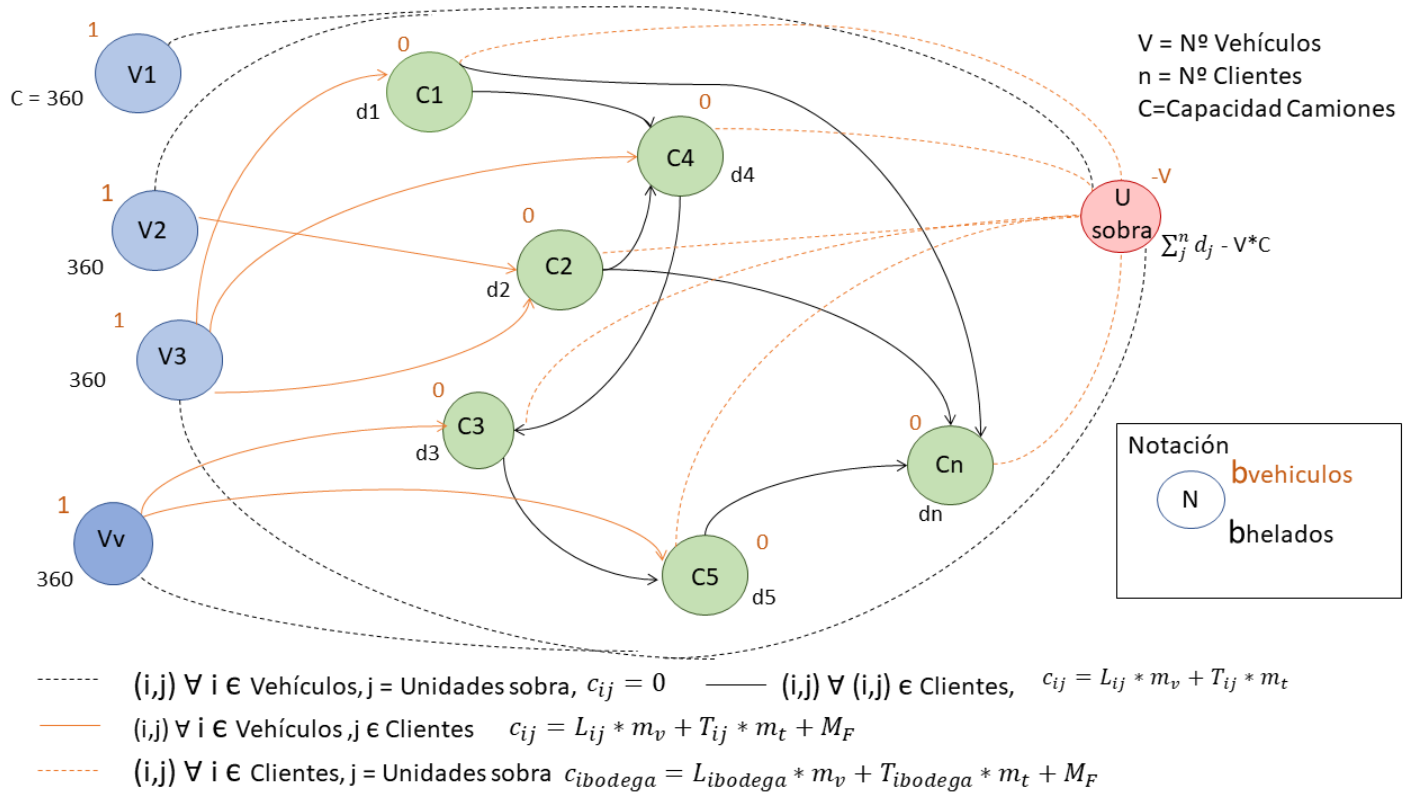

Ilustración 2. Grafo para el problema de MOPTA.

Anexo 3:

$$\max \quad \sum_{e \in E} w_e y_e$$

$$\text{s.t.} \quad \sum_{e \in \delta^-(v)} y_e = f_v^i \quad v \in V \qquad [1]$$

$$\sum_{e \in \delta^+(v)} y_e = f_v^o \quad v \in V$$

$$f_v^o \le f_v^i \le 1 \quad v \in P, \qquad [2]$$

$$f_v^o \le 1 \quad v \in N, \qquad [3]$$

$$\sum_{e \in C} y_e \le |C| - 1 \quad C \in \mathcal{C} \backslash \mathcal{C}_k, \qquad [4]$$

$$y_e \in \{0, 1\} \quad e \in E. \qquad [5]$$

Anexo 4:

Table S3. Performance of the recursive and TSP algorithms for "difficult" real-data KEP instances

			Running time, s	
NDDs	Patient–donor pairs	Edges	Recursive	TSP
3	202	4,706	0.148	0.031
10	156	1,109	13.093	0.022
6	263	8,939	59.158	1.655
5	284	10,126	71.066	0.807
6	324	13,175	418.27	0.981
6	328	13,711	474.947	1.947
6	312	13,045	1,200*	0.157*
10	152	1,125	48.56	0.054
3	269	2,642	40.506	0.134
10	257	2,461	67.783	0.258
7	255	2,390	85.475	0.268
6	215	6,145	248.46	0.532
10	255	2,550	216.48	0.126
1	310	4,463	721.66	0.555
11	257	2,502	1,039.105	0.125
6	261	8,915	1,200	4.435
10	256	2,411	587.238	0.114
6	330	13,399	1,200	1.621
10	256	2,347	1,200	0.305
7	291	3,771	1,200*	0.221
8	275	3,158	1,200*	0.224
4	289	3,499	1,200*	0.2
3	199	2,581	1,200*	0.041
7	198	4,882	1,200*	8.204
2	389	8,346	1,200*	0.096

Timeouts (optimal solution not found) are indicated by an asterisk.

Anexo 5:

$$\begin{aligned}
\max \quad & \sum_{e\in E} w_e y_e + \sum_{C\in\mathcal{C}_k} w_C z_C \\
\text{s.t.} \quad & \sum_{e\in\delta^-(v)} y_e = f_v^i \quad v\in V \\
& \sum_{e\in\delta^+(v)} y_e = f_v^o \quad v\in V \\
& f_v^o + \sum_{C\in\mathcal{C}_k(v)} z_C \le f_v^i + \sum_{C\in\mathcal{C}_k(v)} z_C \le 1 \quad v\in P,
\end{aligned} \tag{6}$$

$$\begin{aligned}
& f_v^o \le 1 \quad v\in N, \\
& \sum_{e\in\delta^-(S)} y_e \ge f_v^i \quad S\subseteq P,\ v\in S \\
& y_e\in\{0,1\} \quad e\in E, \\
& z_C\in\{0,1\} \quad C\in\mathcal{C}_k.
\end{aligned} \tag{7}$$