Implementación del Algoritmo Simplex
Programación Lineal para Optimización de Ingresos en Bicicletas Compartidas

Yalidt Díaz[141394], Yedam Fortiz[119523], María Fernanda Rubio[130441], and José Reyes[142207]

Instituto Tecnológico Autónomo de México

1. Introducción

Citi Bike es el sistema de bicicletas compartidas de la ciudad de Nueva York y el más grande del país. Citi Bike tuvo su apertura en en mayo de 2013 y se ha convertido en una parte esencial de la red de transporte de NY. Está disponible para su uso las 24 horas del día, los 7 días de la semana, los 365 días del año, y los ciclistas tienen acceso a miles de bicicletas en cientos de estaciones en Manhattan, Brooklyn, Queens y Jersey City.

El problema que a continuación se va a resolver es con programación lineal y se desea optimizar los ingresos dependiendo del tipo de pases que ofrecen y sus respectivos precios. Este planteamiento se genera a partir de la nueva entrada de competidores al mercado lo cual puede afectar significativamente a la empresa Citi Bike y de igual manera tienen como objetivo evaluar opciones de reestructurara de precios para ver si es posible incrementar sus ingresos.

2. Características del Problema de Optimización

2.1. Problema de Negocio

Citi Bike tiene un esquema de pago vigente que consiste en tres tipos de pases:

- Anual membership: consiste en pagar \$180 dlls al año por 45 minutos de viaje. Si se exceden los 45 minutos se cobra una tarifa extra de \$0.12 dlls por minuto.
- Day pass: consiste en viajes de 30 minutos durante 24 horas por \$15 dlls, si se excede el tiempo se cobran \$4 dlls por cada 15 minutos extra.
- Single ride: es un solo viaje de 30 minutos por \$3.5 dlls, si se excede el tiempo se cobran \$0.18 dlls extra por minuto.

El objetivo es revisar diferentes esquemas de pago para definir si el esquema actual es óptimo, o si podemos hacer cambios para maximizar las ganancias de Citi Bike. Utilizando información del segundo semestre de 2019 (considerando que 2020 y 2021 han sido años atípicos), se consideraron las siguientes variables para la modelación del problema:

- *trip_id* - Número de viaje
- *duration* - Duración de viaje en minutos
- *starttime* - Fecha de inicio de viaje
- *stoptime* - Fecha de término de viaje
- *trip_category* - Tipo de viaje : round_trip o one way
- *type_pass* - Tipo de pase : *annually*, *daily* y *single_trip*

Para construir un problema de optimización lineal que pretende maximizar la función:

$$income = 3{,}5 * z_1 + 2{,}7 * z_2 + 1{,}8 * z_3 + z_4$$

donde:

- z_1 =passes_sold_single
- z_2 =total_15min_blocks_post_free of daily passes
- z_3 =total_15min_blocks_post_free for anual passes
- z_4 =passes_sold * previous_pass_prices (fix income)

2.2. Métodos de Solución

En este caso, se busca resolver un problema de programación lineal de enteros en donde las variables objetivos toman valores binarios de 0 y 1. Este tipo de problemas se les nombra programaciòn entera binaria y se utilizan para tomar decisiones óptimas para llevar a cabo o no una actividad. El valor de 1 en la variable a optimizar representa que sí se lleva a cabo y un valor de 0 que no. Cuando el número de variables a optimizar es pequeño, puede hacerse una búsqueda de todas las soluciones factibles y evaluarlas para determinar la combinación óptima. Sin embargo, cuando el número de variables es muy grande, las combinaciones son del orden de 2^n siendo n el número de variables. Dado lo anterior, es recomendable utilizar algún método de programación entera para resolver el problema.

El método simplex puede utilizarse para resolver problemas de programación entera; sin embargo, deben cumplirse ciertos supuestos para asegurar una solución entera y óptima. Dado un problema de maximización (o minimización) con restricciones tal que :

$$max_x c^T x \text{ Sujeto a } \quad Ax = b$$

en donde A y b son enteros y A es totalmente unimodular[1], entonces se garantiza que la solución básica factible será un entero. Se dice que A es totalmente unimodular si es una matriz de $m \times n$ cuyos renglones pueden particionarse en dos conjuntos disjuntos B y C tal que cumpla con:

- Cada entrada de A es $0, 1$ o -1.
- Cada columna de A contiene a lo más dos entradas diferentes de cero.

[1] https://en.wikipedia.org/wiki/Integer-programmingUsing-total-unimodularity

- Si dos entradas distintas de cero en una columna tienen el mismo signo, entonces la fila de una de ellas estará en B y la otra en C.
- Si dos entradas distintas de cero en una columna de A tienen signos contrarios, entonces ambas filas estarán en B o en C.

Este será el enfoque que se utilizará para resolver el problema de optimización.

Existen otro tipo de métodos para resolver problemas de programación entera, algunos de ellos son algoritmos heurísticos de búsqueda. La desventaja es que no siempre logran encontrar la solución óptima o no logran diferenciar cuando existe más de una solución.

Un ejemplo de estos algoritmos, es el algoritmo genético que replica procesos de selección natural para encontrar una solución óptima (o casi óptima). El algoritmo funciona de la siguiente manera. Se establece un cromosoma que contiene un número genes igual al número de variables a optimizar. En este caso, cada gen puede tomar valores de 0 y 1. Se establece una población que es un número N de cromosomas. Se evalúan las restricciones y aquellos cromosomas que no las cumplan, son descartados. Después se evalúa la función objetivo para cada cromosoma factible identificando a los mejores y se reproducen". Generalmente este en este proceso se combinan los primeros m genes de uno con los últimos z de otro. Cada "hijo"puede tener una mutación aleatoria en alguno de sus genes, cambiando su valor. Tanto los cromosomas "padresçomo los "hijos"sobreviven a la siguiente generación y se repite el proceso. El número de cromosomas en la población siempre debe ser el mismo. El criterio de paro del algoritmo es que la población haya convergido y no existen mejoras en los genes.

3. Método Simplex

El método simplex es un algoritmo de optimización que busca establecer cúales restricciones son activas y cuáles inactivas en la solución. Actualiza dichas restricciones y realiza cambios en cada iteración del algoritmo.

El objetivo es iterar eligiendo algún valor del conjunto de variables no básicas y sustituirlo por un valor de las variables básicas, a este proceso se le conoce como pivoteo. Es importante mencionar que la elección de qué variable se sustituirá depende de la existencia de soluciones básicas factibles que mejoren la función objetivo y para ello se utiliza un criterio de optimalidad.

3.1. Algoritmo

Para cada paso dentro del método simplex se realizará lo siguiente:[2]

[2] https://itam-ds.github.io/analisis-numerico-computo-cientifico/IV.optimizacion-en-redes-y-prog-lineal/4.2/Programacion-lineal-y-metodo-simplex.htmlalgoritmo-para-un-paso-del-metodo-simplex

Dados $\mathcal{B}, \mathcal{N}, x_B = B^{-1}b \geq 0, x_N = 0$

> Resolver $B^T v = c_B$ para v
>
> Calcular $\lambda_N = c_N - N^T v$
>
> Si $\lambda \geq 0$ se encontró un punto óptimo, si no:
>
> > Seleccionar $nb \in \mathcal{N}$ con $\lambda_{nb} < 0$ como el índice que entra
> >
> > Resolver $Bd = A_{nb}$ para d.
> >
> > Si $d \leq 0$ detenerse, el problema es no acotado.
> >
> > Calcular $x_{nb}^{+} = \min\{\frac{x_{B_i}}{d_i} : d_i > 0\}$ y sea ba el índice que minimiza.
> >
> > Actualizar $x_B^{+} = x_B - dx_{nb}^{+}, x_N^{+} = (0, \dots, 0, x_{nb}^{+}, 0, \dots, 0)^T$
> >
> > Cambiar $\mathcal{B}$ al añadir nb y remover la variable básica correspondiente a la columna ba de B.

4. Paquete Simplex en Python

Se ha desarrollado un paquete en Python para resolver problemas de programación lineal utilizando el método Simplex, el cual, se utilizó para resolver este problema. Dicho paquete, permite maximizar o minimizar una función objetivo lineal, sujeto a restricciones lineales, asegurando una solución no negativa. Por otro lado, si el problema no cumple con los supuestos para que el algoritmo Simplex lo pueda resolver, retornará un error.

Este paquete instalará todas las dependencias necesarias para su utilización. Además, cuenta con dos módulos, 'Simplex' y 'SimplexC', que resuelven el problema de la misma forma pero existen unas diferencias en cuanto a su compilación que se presentaran más adelante.

Para utilizar este paquete es necesario instalarlo desde la línea de comandos con la siguiente instrucción:

```
pip install --quiet "git+https://github.com/optimizacion
    -2-2021-1-gh-classroom/practica-1-segunda-parte-yefovar.
    git#egg=Simplex&subdirectory=src"
```

De forma automática se instalarán ambos módulos y podrán utilizarse para resolver este o cualquier otro problema de optimización simplex. Se recomienda realizar la instalación dentro de un ambiente virtual o de lo contrario, puede utilizarse la imagen de *Docker* desarrollada para esta finalidad [3]

Los módulos, tanto Simplex como SimplexC, contienen un objeto llamado Simplex, el cual debe debe definirse para poder resolver el problema. Los atributos de este objeto son $c, A, b, problem$ y $verbose$. El atributo problema, toma

[3] Desde *Docker Hub*, puede descargarse *yalidt/pkg_optimizacion:0.1* que contiene el paquete instalado junto con *Jupyter Notebook*. Ademas, puede descargarse *ferubio/pkg_optim_kale:0.1* que cuenta con *Kale* y *Kubeflow* para resolver que requieran de un *pipeline* más complejo de forma eficiente

valores de Max o Min dependiendo si el problema es de maximización o minimización. El atributo verbose toma valores booleanos, si es verdadero, se imprimirán los resultados y algunas otras evaluaciones intermedias del algoritmo, de lo contrario sólo regresará el resultado de la maximización. Para el caso de c, A y b, se requieren arreglos tal que c es la función objetivo a maximizar, A es una matriz de costos y b un arreglo de restricciones. Estas matrices y vectores deben cumplir con las características de la forma canónica de un problema de programación lineal. Para el caso de maximización se tiene:

$$\max_x (-c)^T x \;\; S.A$$

$$Ax \leq b$$

$$x \geq 0 \;\; con:$$

$$c, x \in R^n$$

$$A \in R^{m \times n}$$

$$b \in R^m$$

y para el caso de un problema de minimización:

$$\min_x c^T x \;\; S.A$$

$$Ax \geq b$$

$$x \geq 0 \;\; con:$$

$$c, x \in R^n$$

$$A \in R^{m \times n}$$

$$b \in R^m$$

Una vez declarado el objeto, se utiliza el método $solve()$ para resolver el problema mencionado. A continuación se presenta un breve ejemplo de la utilización del paquete en Python:

```
import Simplex
c = [3, 5]
b = [4, 12, 18]
A = [[1,  0],
     [0,  2],
     [3, 2]]

problema = Simplex.Simplex(c,A,b,problem='Max')
problema.solve()

problema = SimplexC.Simplex(c,A,b,problem='Max')
problema.solve()
```

La diferencia entre el paquete *Simplex* y *SimplexC* es que este este último se encuentra compilado en *C* utilizando *Cython*. Esta compilación permite hacer más eficiente ciertas secciones de código para que el algoritmo sea más rápido. Ene cambio, *Simplex* únicamente contiene código en Python pero se encuentra optimizado con la implementación de operaciones de álgebra lineal básica de la librería *OpenBlas*. Para determinar la eficiencia de los paquetes, se realizó un perilamiento de tiempo y de memoria que puede consultarse en *Github*[4]

5. Solución del Problema de Optimización

5.1. Procesamiento de los Datos

La base de datos de CitiBike contaba originalmente con las siguientes variables: *tripduration, starttime, stoptime, start station id, start station name, start station latitude, start station longitude, end station id, end station name, end station latitude, end station longitude, bikeid, usertype, gender.* Las cuales nos ayudaron a hacer un pequeño análisis exploratorio de datos, pero fue necesario agregar 4 varables para pasar a la parte del modelado, con lo cual se agregó : **id_trip, trip_category, duration, pass_type**.

En primer lugar, para la variable **id_trip** únicamente creamos una nueva columna con un número único de viaje. En segundo lugar, para **trip_category** se utilizaron como referencia los porcentajes del sistema de viajes compartido LA Bike Share para crear esta variable que nos indicara que tipo de viaje realizaron las personas: one_way o round_trip. De acuerdo a la referencia, se mantuvo un 19.2 % para round_trip y la diferencia para one_way (únicamente sobre los pases anuales o diarios, ya que los viajes con pase single_trip siempre son one_way). En tercer lugar, se creó una nueva variable a partir de la variable *tripduration*, ya que el tiempo venía en segundos y para poder hacer el cálculo de las traifas adicionales se necesitaba en minutos, únicamente creamos una nueva columa **duration** que realizaba dicha conversión del tiempo. Por último, se creó una variable a partir de **usertype**, la cual fue nombrada **type_pass**. Esto fue necesario porque la base original no diferenciaba entre pases diarios o sencillos, únicamente diferenciaba los anuales. La información se obtuvo a partir de los reportes anuales de la empresa en donde mencionan el número de usuarios con cada tipo de pase : *daily_pass* 31.6 % y *single_trip* 68.4 % respectivamente.

5.2. Definición de Variables

El sistema de precios de Citi Bike contempla un costo por cada pase en cual cubre los primeros 30 o 45 minutos dependiendo del tipo de pase, después de exceder estos tiempos se cobra una tarifa adicional que varía dependiendo del tipo de pase. Por simplicidad se decidió calcular los excedentes de tiempo en bloques de 15 minutos y para el cálculo del cobro de esta tarifa adicional se crearon

[4] https://github.com/optimizacion-2-2021-1-gh-classroom/practica-2-segunda-parte-yefovar

dos nuevas variables *total_free_blocks* y *time_block_count_post_free*. Apartir de la creación de estas nuevas variables se observaron *outliers* que tenían un exceso en el número de bloques de 15 minutos porque los consideramos errores en la base de datos.

Adicionalmente, se realizó limpieza general en los datos como : convertir a minúsculas todas las variables, se cambio el formato de la fecha en las variables *starttime y stoptime*.

Por último, se calcularon los puntos de equilibro para los pases *daily* y *anually*, tomando como referencia el pase *single_trip*, para obtener el número de viajes que debería realizar las personas como mínimo para que les convenga adquirir un pase anual o diario respectivamente.

5.3. Experimentos y Resultados

Se realizaron tres experimentos para la optimización con el fin de saber cuál es el mejor escenario para la empresa suponiendo que el *single pass* siempre se vende, dejamos abierta la opción de que deje de venderse alguno otro de los pases. Esto se traduce en las siguientes restricciones para el modelo de optimización:

$$day\_yes + day\_no = 1$$

$$annual\_yes + annual\_no = 1$$

Para el periodo de tiempo examinado, tenemos los viajes registrados en la tabla:

Type	Total trips	Total free blocks	Total 15min blocks post free	Total minutes
annually	8,230,870	9,354,633	1,765,973	108,470,986
daily	872,372	927,538	76,574	30,526,375

Cuadro 1. Viajes registrados de Junio a Diciembre 2019

Escenario de precios actuales El primer escenario se basa en los costos actuales. Los resultados indican que los ingresos alcanzan su valor óptimo si se siguen ofreciendo los pases anuales y diarios y el valor maximizado de los ingresos es $\$40,709,327$.

Type	Costo del pase	Tarifa adicional (15 min)
annually	\$180 dlls/año	\$1.8 dlls
daily	\$15 dlls/día	\$2.7 dlls
single	\$3.5 dlls/viaje	\$2.7 dlls

Cuadro 2. Precios vigentes en 2019

Escenario precios 2018 El segundo escenario se basa en los costos anteriores, vigentes durante 2018. Los resultados indican que los ingresos alcanzan su valor óptimo si se siguen ofreciendo los pases anuales y dejan de venderse los diarios y el valor maximizado de los ingresos es $\$35,716,840$.

Type	Costo del pase	Tarifa adicional (15 min)
annually	\$169 dlls/año	\$2.5 dlls
daily	\$14.95 dlls/día	\$4 dlls
single	\$3 dlls/viaje	\$2.7 dlls

Cuadro 3. Precios vigentes en 2018

Escenario precios 2020 Para el último escenario proponemos un precio que creemos que hará que el pase diario deje de considerarse en la solución óptima. Los resultados indican que los ingresos alcanzan su valor óptimo si se siguen ofreciendo los pases anuales y diarios y el valor maximizado de los ingresos es $\$39,879,308$.

Type	Costo del pase	Tarifa adicional (15 min)
annually	\$180 dlls/año	\$1.8 dlls
daily	\$5 dlls/día	\$2.7 dlls
single	\$3.5 dlls/viaje	\$2.7 dlls

Cuadro 4. Precios propuestos en 2020

Para realizar estos experimentos, se utilizó infraestructura montada en *AWS*, utilizando *Kale* y Minikube para ejecutar el pipeline. La carga de información se realiza en forma paralela, ya que las bases de datos eran de gran tamaño. Después, se realizó un proceso de preparación, limpieza y construcción de las matrices necesarias para el proceso de optimización. Por último, se realizaron los tres experimentos en paralelo y se obtuvieron los resultados.

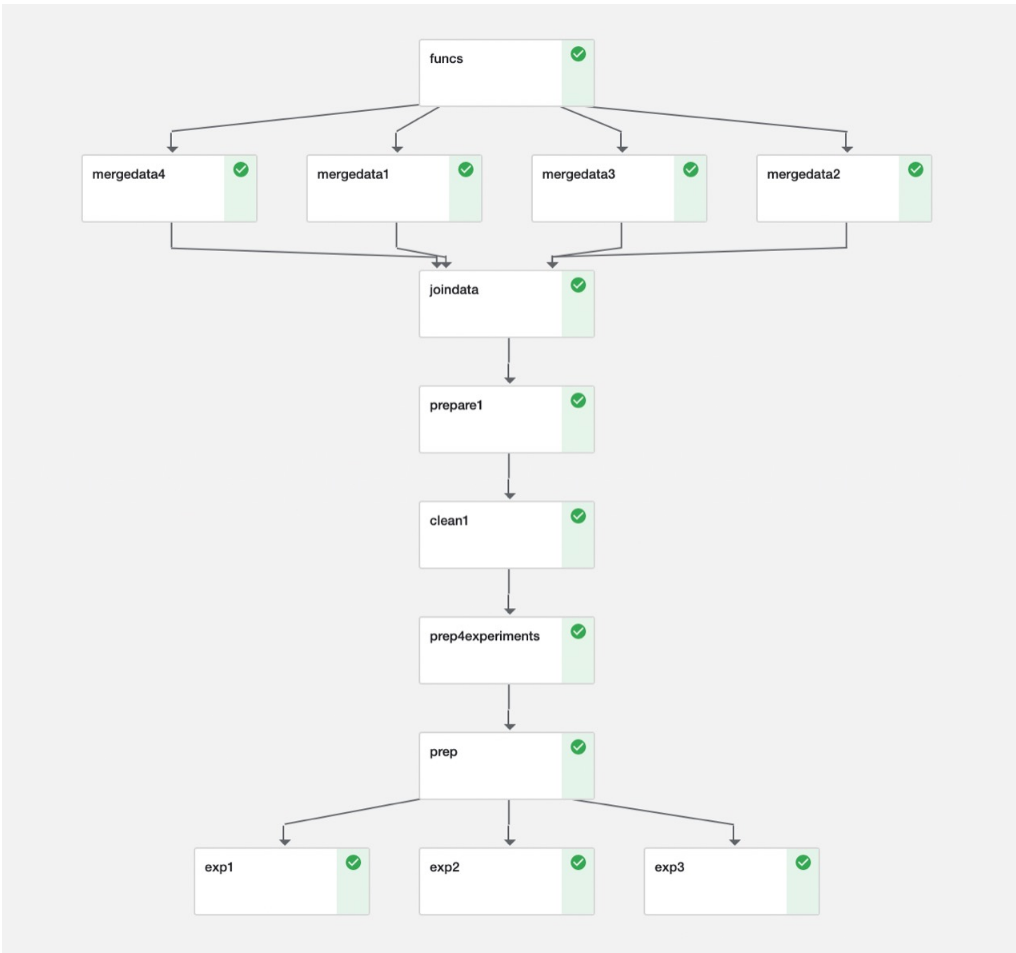

Figura. 1 Pipeline del problema de optimización en Kale

6. Conclusiones

- De acuerdo a los tres experimentos que se evaluaron, la empresa debe continuar ofreciendo el pase anual y el pase diario, ya que aunque exista un incremento en los precios de los pases y de las tarifas extras, los usuarios continuarán adquiriendo los pases y seguirá siendo rentable.
- Citi Bike podría implementar un tipo de pase mensual, como lo hacen otros sistemas de bicicletas compartidas. La evaluación de si es o no rentable se puede hacer mediante el método expuesto considerando supuestos sobre las variables.

- Nos ayudó bastante el uso de Kale y minikube para el desarrollo de la optimización con vlos últimos 6 meses del año 2019, ya que el uso de contenedores de docker lo hizo posible para juntar la base de datos poco a poco, pues en el jupyter notebook no era posible la ejecución con esos datos de gran escala.
- En el avance que tuvimos del perfilamiento y de las prácticas anteriores nos ayudaron a hacer más óptimo el algoritmo en tiempos de ejecución y en presentación de resultados al usuario final.

Referencias

[1] Citi Bike, oficial page: https://www.citibikenyc.com/pricing

[2] Citi Bike, data: https://www.citibikenyc.com/system-data

[3] Algoritmos genéticos : https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3: :text=A

[4] Libro de Estudio, Palacios Moreno Erick: https://itam-ds.github.io/analisis-numerico-computo-cientifico/README.html