

Ruta óptima dentro de Uniandes

Integrantes:

Julián Camilo Mora Valbuena y Luisa Fernanda Fuentes Ladino

Entrega 3: Algoritmo propuesto y Resultados del Algoritmo vs Modelo Matemático

Departamento de Ingeniería de Sistemas y Computación
Universidad de Los Andes
Bogotá, Colombia

Contenido

1	Contexto	1
2	Conjuntos, Parámetros y Variables	4
3	Función Objetivo y Restricciones	4
4	Implementación y resultados del Modelo Matemático.....	5
5	Algoritmo propuesto	8
6	Resultados del Algoritmo vs Modelo Matemático	10

1 Contexto

Desde su fundación en 1948, la universidad de los Andes se ha caracterizado por su plan constante de expansión en cuanto a su área predial y área construida. Actualmente, el campus de la universidad tiene más de 166.653 metros cuadrados construidos. El cual, está conformado por 83 bloques, espacios comunes, zonas verdes, terrazas y más. Adicionalmente, algunos de sus edificios se encuentran fuera del campus principal, por lo que es necesario salir de la universidad para acceder a ellos. Por tal motivo, movilizarse dentro de la universidad es un problema para cualquier miembro de la comunidad. No solo por su gran extensión, sino por la cantidad de personas que diariamente albergan las instalaciones de la universidad.

La intención de este proyecto es modelar la estructura de movilidad en la Universidad de los Andes, con el fin de encontrar cuáles son los caminos óptimos para llegar a un punto específico del campus. Para ello, se debe tener en cuenta diferentes variables, por ejemplo, el tiempo promedio de desplazamiento entre dos puntos de la universidad, a una velocidad constante de 5 km/h, ritmo medio de una persona caminando (Nike, 2021). Sin embargo, consideramos que es prudente manejar una velocidad constante de 3 km/h como el promedio para nuestro modelo. Además, es importante tener en cuenta la altitud de ambos puntos, para medir la diferencia entre ir a un punto por escaleras o en un ascensor. Lo cual lleva a otro aspecto interesante, saber el tiempo medio que implica esperar en un ascensor de la universidad, tomando en cuenta la cantidad de personas que se mueven en ciertos horarios específicos, como cambios de clase. Pues es la movilidad dentro de la universidad se ve muy alterada cuando salen todos los estudiantes de las clases. Por ejemplo, si en promedio en un salón del edificio Santo Domingo caben 60 personas, por piso hay una media de 8 salones y hay 6 pisos dedicados a salones, aproximadamente 2800 estudiantes se desplazan en su interior, salen y entran alrededor de cada hora y veinte minutos (duración estándar de una clase). Pero bueno, aparte del tráfico de personas, que claramente puede volver más demorado el hecho de subir en ascensor. Hay que tener en cuenta el tiempo que se demora una persona en promedio al utilizar el ascensor o las escaleras. Un estudio realizado por Canadian Medical Association Journal (2011), demostró que, en promedio, subir un piso en escaleras no toma más de 13 segundos, mientras que en ascensor los sujetos de prueba tardaron casi 40 segundos. Para nuestro

modelo, tomaremos que la distancia entre cada piso inmediato es de 5 metros, por lo que a 3 km/h, una persona se demorará aproximadamente 6 segundos en subir o bajar un piso. En ascensor, nosotros realizamos las mediciones de tiempo y, en el mejor caso, subir o bajar un piso solo demora 5 segundos. No obstante, este es un caso en el que no hay ningún tipo de demora, por lo que, en ciertos horarios con más congestión de personas, el tiempo de subir o bajar en ascensor será representado teniendo el peor caso, que el ascensor tenga que ir al otro extremo del edificio y volver. En ese caso, el tiempo de subir o bajar un piso será de 20 segundos multiplicado por el número de pisos que tenga el edificio.

Limitaciones del problema y qué es lo que se desea minimizar:

Tomando en cuenta las variables descritas anteriormente, la tarea principal del proyecto será desarrollar un modelo que determine cual es el camino óptimo que se debe seguir para minimizar el tiempo recorrido entre dos puntos de la universidad. Esto, teniendo en cuenta que solo consideraremos la partida desde un piso específico de un edificio hasta la llegada a otro piso específico de otro edificio, es decir, no tendremos en cuenta la distancia de cada salón hacia las escaleras o ascensores de acceso de cada punto. Además, consideraremos que el tiempo de desplazamiento entre cada piso es el mismo para las escaleras estáticas y las eléctricas. Así mismo, vamos a asumir que cada clase dura hora y treinta minutos, es decir 5 minutos antes y 5 minutos después de que empiece y termine cada clase va a haber una mayor concurrencia de personas. El horario académico comienza a las 6:30am y termina 12 horas después, por lo que vamos a considerar 8 intervalos de 90 minutos.

Figura 1. Modelo distancia D_{jp2}^{ip1}

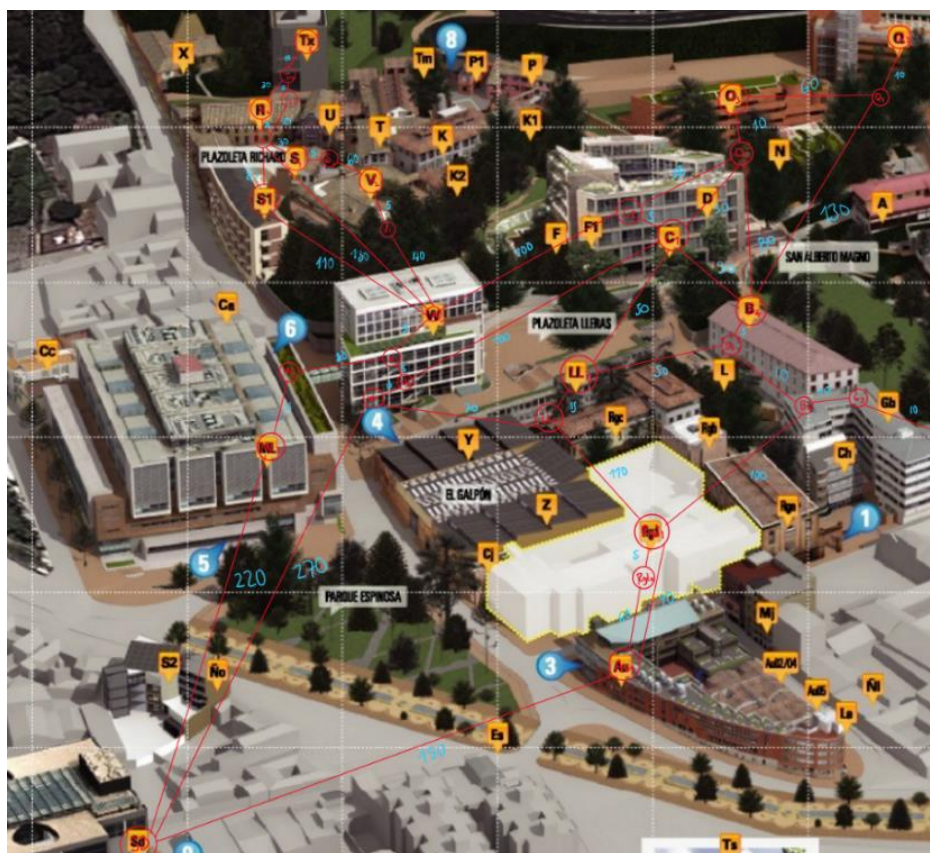
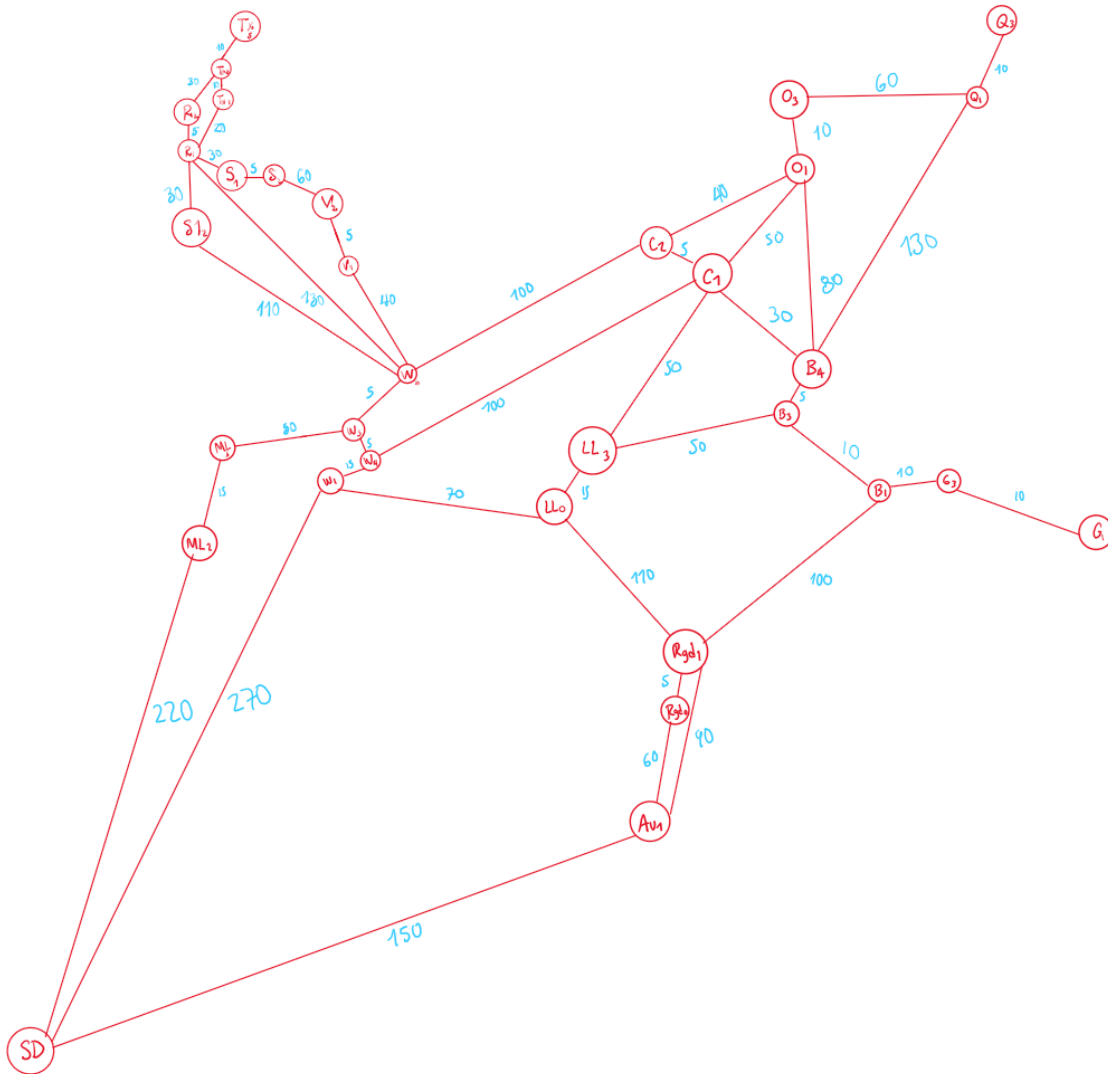


Figura 2. Abstracción del modelo



Para modelar el problema creamos un grafo cuyos nodos representan los diferentes edificios de la universidad, en algunos pisos clave. Además, su peso es representado por las distancias entre dos puntos específicos de la universidad. Las distancias están en metros y fueron obtenidas utilizando la herramienta de medición gratuita que está incorporada con Google Maps. Para representar la distancia entre los pisos, utilizamos un valor constante de 5 metros. Así, de un primer piso a un cuarto piso hay 15 metros, por ejemplo. Además, hay que aclarar que por motivos de la entrega solo se utilizaron los 16 edificios más representativos del campus principal, con los pisos que más se conectan entre sí.

2 Conjuntos, Parámetros y Variables

Table 1. Conjuntos y Parámetros

Sets and Parameters	Description
N	Set de nodos (edificios/lugares).
o	Nodo origen (nombre lugar)
d	Nodo destino (nombre lugar)
v	Set intervalos tiempo de clases
S_i	Número de pisos por nodo i
H_i	Hora partida desde el nodo i
D_{jp2}^{ip1}	Distancia entre un nodo i en el piso $p1$ y otro nodo j en el piso $p2$.

Table 2. Variables de decisión

Variables	Description
A	Tiempo tomado en ascensores (Variable Entera).
E	Tiempo tomado en escaleras (Variable Entera).
T_j^i	Tiempo que se demora una persona para caminar del nodo i al nodo j , a una velocidad constante de 5km/h
Y_j^i	Determina si el camino entre el nodo i y el nodo j fue seleccionado (Binary variable).

3 Función Objetivo y Restricciones

$$\min \sum_{i \in N} \sum_{j \in N} ((T_j^i + A + E) * Y_j^i) \quad (1)$$

$$\sum_{j \in N} Y_{ij} = 1 \quad \forall i | i = o \quad (2)$$

$$\sum_{i \in N} Y_{ij} = 1 \quad \forall j | j = d \quad (3)$$

$$\sum_{j \in N} Y_{ij} - \sum_{j \in N} Y_{ji} = 0 \quad \forall i | i \neq o, \forall j | j \neq d \quad (4)$$

$$Y_{ij} + Y_{ji} \leq 1 \quad \forall i | i \neq o, \forall j | j \neq d \quad (5)$$

(1) La función objetivo busca minimizar el tiempo recorrido desde un nodo fuente hasta un destino. Para esto se considera el tiempo que la persona tomará desplazándose, caminando desde un punto a otro, se suma con el tiempo que gasta tomando ascensores en el camino y subiendo escaleras. Todo este factor se multiplica por una variable binaria que nos indica si la persona debe o no pasar por el camino entre los nodos i y j .

(2) Esta restricción establece que se debe elegir el nodo origen para que el camino óptimo pase por él, es decir, la variable objetivo con los valores Y_j^o debe ser igual a 1. Por eso, solo se puede pasar una vez por este nodo.

(3) Esta restricción establece que se debe elegir el nodo destino para que el camino óptimo pase por él, es decir, la variable objetivo con los valores Y_d^i debe ser igual a 1. Por eso, solo se puede pasar una vez por este nodo.

(4) Esta restricción establece que se debe elegir el nodo intermedio para que el camino óptimo pase por él, es decir, la variable objetivo con los valores Y_j^i debe ser igual a 0. Este nodo debe ser distinto al nodo origen y el nodo destino y solo se puede pasar una vez por ahí.

(5) Esta restricción evita que se produzcan enlaces de conexión bidireccional entre dos nodos. De esta manera, el grafo se vuelve dirigido y se representa adecuadamente el camino real entre dos nodos, en el cuál la dirección sí importa.

4 Implementación y resultados del Modelo Matemático

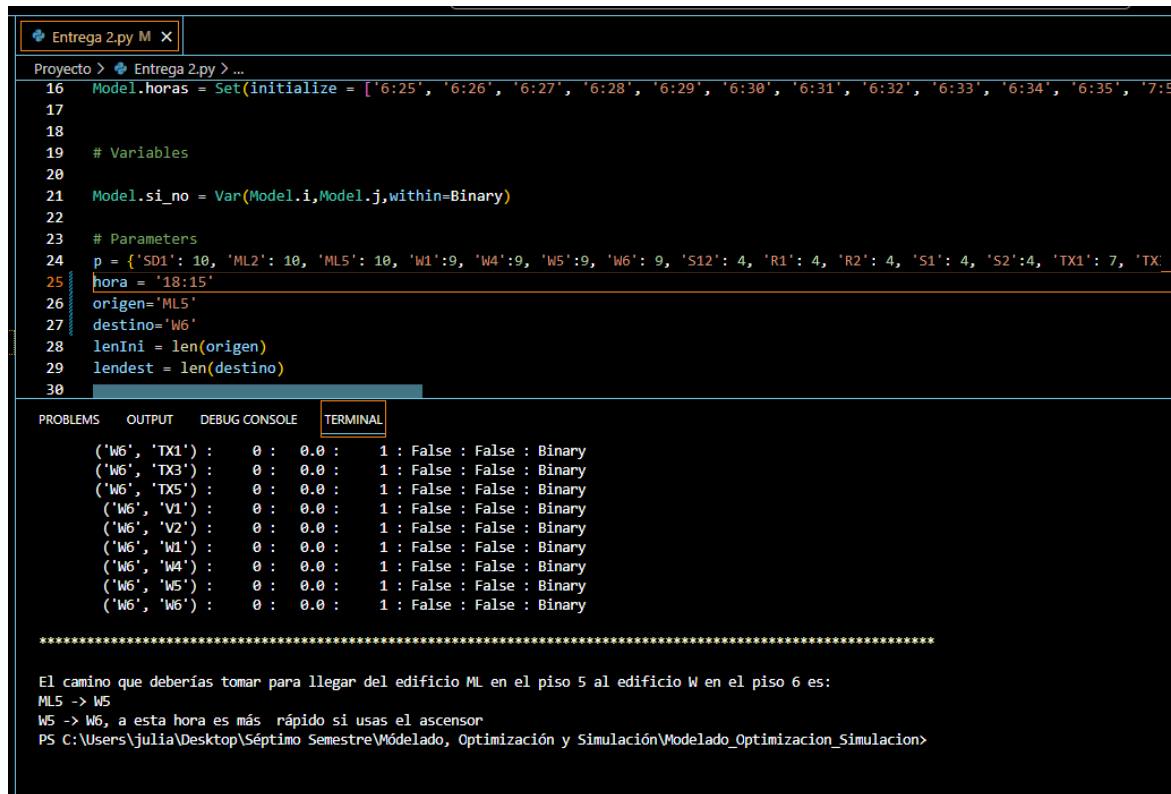
4.1 Escenario 1

Un estudiante de la universidad de los andes desea hacer un recorrido corto desde el quinto piso del edificio Mario Laserna al sexto piso del edificio contiguo, el W. El estudiante se va a desplazar a las 6:15 de la tarde y desea saber cuál es el camino que menos tiempo consumirá para su desplazamiento.

Table 3. Conjuntos y Parámetros

Sets and Parameters	Content
N	{SD, ML, W, AU, RGD, LL, G, R, S1, S, O, Q, C, TX, B, V}
o	ML
d	W
v	{6:30, 8:00, 9:30, 11:00, 12:30, 14:00, 15:30, 17:00, 18:30}
$9_W, 10_{ML}$	{ML 10, W 9}
$18:15_{ML}$	18:15

4.2 Resultados Escenario 1



```
Entrega 2.py M X
Proyecto > Entrega 2.py > ...
16 Model.horas = Set(initialize = ['6:25', '6:26', '6:27', '6:28', '6:29', '6:30', '6:31', '6:32', '6:33', '6:34', '6:35', '7:35'])
17
18
19 # Variables
20
21 Model.si_no = Var(Model.i,Model.j,within=Binary)
22
23 # Parameters
24 p = {'SD1': 10, 'ML2': 10, 'ML5': 10, 'W1':9, 'W4':9, 'W5':9, 'W6': 9, 'S12': 4, 'R1': 4, 'R2': 4, 'S1': 4, 'S2':4, 'TX1': 7, 'TX2': 7}
25 hora = '18:15'
26 origen='ML5'
27 destino='W6'
28 lenIni = len(origen)
29 lendest = len(destino)
30

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
('W6', 'TX1') : 0 : 0.0 : 1 : False : False : Binary
('W6', 'TX3') : 0 : 0.0 : 1 : False : False : Binary
('W6', 'TX5') : 0 : 0.0 : 1 : False : False : Binary
('W6', 'V1') : 0 : 0.0 : 1 : False : False : Binary
('W6', 'V2') : 0 : 0.0 : 1 : False : False : Binary
('W6', 'W1') : 0 : 0.0 : 1 : False : False : Binary
('W6', 'W4') : 0 : 0.0 : 1 : False : False : Binary
('W6', 'W5') : 0 : 0.0 : 1 : False : False : Binary
('W6', 'W6') : 0 : 0.0 : 1 : False : False : Binary

*****

El camino que deberías tomar para llegar del edificio ML en el piso 5 al edificio W en el piso 6 es:
ML5 -> W5
W5 -> W6, a esta hora es más rápido si usas el ascensor
PS C:\Users\julia\Desktop\Séptimo Semestre\Modelado, Optimización y Simulación\Modelado_Optimizacion_Simulacion>
```

Análisis de resultados:

El resultado fue el esperado, es decir, para ir del edificio ML en el quinto piso, al sexto piso del edificio W, el camino de costo mínimo es cruzar el puente del ML y el W en el quinto piso, pues esta es una conexión entre ambos nodos y, posteriormente subir al sexto piso del W. Ahora, ¿qué más se puede extraer de este caso? Por más sencillo que parezca, este caso nos está demostrando dos cosas fundamentales de nuestro problema. Primero, nosotros planteamos diversas formas de llegar del quinto piso del ML al sexto piso del W, una alternativa pudo ser bajar hasta el segundo piso del ML, cruzar al primer piso del W y tomar el subir al sexto piso del W. Sin embargo, este resultado nos demuestra que sí optó por el camino que tiene menos distancia. En segundo lugar, hay que resaltar que nuestro modelo le sugiere al usuario utilizar el ascensor para subir del quinto piso del W al sexto piso del W. ¿Por qué? Bueno, con los valores que le definimos, al viajar a las 6:15 pm, es decir, una hora poco congestionada, el programa reconoce que es será más eficiente subir en ascensor, pues como mencionamos anteriormente, en horas sin congestión tomamos el mejor caso, el cual dice que subir un piso en ascensor solo demora 5 segundos. Lo anterior, demuestra que está funcionando bien, pues como también se especificó al inicio, en nuestro modelo las personas caminan con una velocidad constante de 3 km/h, por lo que recorrer 5 metros para subir un piso les tardará más de 6 segundos. Por lo tanto, está bien que nuestro modelo sugiera viajar en ascensor.

4.3 Escenario 2

Una joven estudiante de la universidad de los andes tiene que llegar desde el primer piso del W, al quinto piso del edificio de TX. Ella quiere saber cómo puede desplazarse de la manera más eficiente, contando caminos cortos, escaleras o ascensores, si tiene clases a las 9:30 de la mañana

y va tarde. Por lo que es esencial para ella poder conocer la ruta más rápida. ¿Deberá esperar el ascensor? ¿Mejor sube a pie por las escaleras? De ser así, ¿cuáles escaleras?

Table 4. Conjuntos y Parámetros

Sets and Parameters	Content
N	{SD, ML, W, AU, RGD, LL, G, R, S1, S, O, Q, C, TX, B, V}
o	W
d	TX
v	{6:30, 8:00, 9:30, 11:00, 12:30, 14:00, 15:30, 17:00, 18:30}
$9_W, 7_{TX}$	{W 9, TX 7}
$9:30_{ML}$	9:30

4.4 Resultados Escenario 2

```

Entrega 2.py M X
Proyecto > Entrega 2.py > ...
19 # Variables
20
21 Model.si_no = Var(Model.i,Model.j,within=Binary)
22
23 # Parameters
24 p = {'SD1': 10, 'ML2': 10, 'ML5': 10, 'W1':9, 'W4':9, 'W5':9, 'W6': 9, 'S12': 4, 'R1': 4, 'R2': 4, 'S1': 4, 'S2':4, 'TX1': 7, 'TX3': 7,
25 hora = '9:30'
26 origen='W1'
27 destino='TX5'
28 lenIni = len(origen)
29 lendest = len(destino)
30
31 formas= []

```

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
('W6', 'G3') : 0 : 0.0 : 1 : False : False : Binary
('W6', 'LL0') : 0 : 0.0 : 1 : False : False : Binary
('W6', 'LL3') : 0 : 0.0 : 1 : False : False : Binary
('W6', 'ML2') : 0 : 0.0 : 1 : False : False : Binary
('W6', 'ML5') : 0 : 0.0 : 1 : False : False : Binary
('W6', 'O1') : 0 : 0.0 : 1 : False : False : Binary
('W6', 'O3') : 0 : 0.0 : 1 : False : False : Binary
('W6', 'Q1') : 0 : 0.0 : 1 : False : False : Binary
('W6', 'Q3') : 0 : 0.0 : 1 : False : False : Binary
('W6', 'R1') : 0 : 1.0 : 1 : False : False : Binary
('W6', 'R2') : 0 : 0.0 : 1 : False : False : Binary
('W6', 'RGD0') : 0 : 0.0 : 1 : False : False : Binary
('W6', 'RGD1') : 0 : 0.0 : 1 : False : False : Binary
('W6', 'S1') : 0 : 0.0 : 1 : False : False : Binary
('W6', 'S12') : 0 : 0.0 : 1 : False : False : Binary
('W6', 'S2') : 0 : 0.0 : 1 : False : False : Binary
('W6', 'SD1') : 0 : 0.0 : 1 : False : False : Binary
('W6', 'TX1') : 0 : 0.0 : 1 : False : False : Binary
('W6', 'TX3') : 0 : 0.0 : 1 : False : False : Binary
('W6', 'TX5') : 0 : 0.0 : 1 : False : False : Binary
('W6', 'V1') : 0 : 0.0 : 1 : False : False : Binary
('W6', 'V2') : 0 : 0.0 : 1 : False : False : Binary
('W6', 'W1') : 0 : 0.0 : 1 : False : False : Binary
('W6', 'W4') : 0 : 0.0 : 1 : False : False : Binary
('W6', 'W5') : 0 : 0.0 : 1 : False : False : Binary
('W6', 'W6') : 0 : 0.0 : 1 : False : False : Binary

*****

El camino que deberías tomar para llegar del edificio W en el piso 1 al edificio TX en el piso 5 es:
W1 -> W4, a esta hora es más rápido si usas las escaleras
W4 -> W5, a esta hora es más rápido si usas las escaleras
W5 -> W6, a esta hora es más rápido si usas las escaleras
W6 -> R1
R1 -> TX1
TX1 -> TX3, a esta hora es más rápido si usas las escaleras
TX3 -> TX5, a esta hora es más rápido si usas las escaleras
PS C:\Users\julia\Desktop\Séptimo Semestre\Modelado, Optimización y Simulación>Modelado_Optimizacion_Simulacion>

```

Análisis de resultados:

Este caso es un poco menos intuitivo, quizás para alguien que está acostumbrado al ambiente uniandino, esta respuesta puede sonar un poco evidente, pero no es tan fácil de percibir a simple vista. En primer lugar, sabemos que nuestra estudiante quiere viajar a las 9:30 de la mañana, una hora bastante congestionada en toda la universidad pues es justo en cambio de clases. Además, no es sorpresa para ningún miembro de nuestra comunidad que los ascensores del W en estos horarios tengan filas con más de 10 personas. ¿Por qué? Pues por el mismo caso que estamos considerando dentro de nuestra solución. Cuando hay un mayor flujo de usuarios, es más probable que el ascensor se demore más, pues la demanda aumenta y está constantemente subiendo y bajando, casi siempre parando en la mayoría de los pisos antes de llegar al destino final. Por lo que, en nuestro modelo, tomar el ascensor significa 20 segundos multiplicado por el número de pisos del edificio. Por esta razón, nuestro programa le recomienda a la joven estudiante que no se arriesgue y suba por las escaleras. Debido a nuestra forma de modelar el ejercicio, el programa diferencia viajar del primer piso al cuarto, después del cuarto al quinto y después al sexto, porque en cada uno de estos pisos podría tomar una decisión diferente, como ir al C, ir al ML o algo más. Sin embargo, el programa entiende que lo más eficiente es subir del primer al sexto piso en el W, por las escaleras. Pues, a una velocidad de 3 km/h, subir aquellos 25 metros inclinados no tardará más de un minuto, mientras que en ascensor este tiempo puede duplicarse. Siguiendo con la misma lógica, el programa evalúa posibles caminos para llegar del sexto piso del W al TX, pero decide que pasar por el R es lo más sensato. Finalmente, utilizando la misma lógica de la hora pico, el programa entiende que para subir al quinto piso del TX lo mejor es subir por las escaleras. En conclusión, se puede decir que ambos escenarios dieron resultados favorables.

Para los entregables, se enviaron dos archivos .py, los cuales representan cada uno de los casos respectivamente.

5 Algoritmo propuesto

La heurística implementada se basa en la adaptación del Algoritmo de Dijkstra de Jorge Ignacio Barrera Alviar. Este algoritmo fue utilizado para encontrar el camino más corto entre dos ubicaciones específicas en la Universidad de los Andes. Para implementar el algoritmo primero es necesario construir una matriz con los tiempos de recorrido entre cada par de nodos dependiendo de la hora de partida del usuario. Posteriormente, se ejecuta el algoritmo de Dijkstra pasándole como parámetros la matriz de tiempos, el nodo origen y el nodo destino.

5.1 Pseudocódigo del algoritmo

Algoritmo Dijkstra (t, origen, destino)

1. N : conjunto de nodos; S : conjunto de nodos visitados; s : nodo fuente; f : nodo destino; t : grafo;
2. $S = \{s\}$; $S' = N - \{s\}$;
3. $d[i] \leftarrow \text{infinito}$ para cada nodo i en N ;
4. $d[s] \leftarrow 0$;
5. $\text{pred}(s) \leftarrow \text{no determinado}$;
6. $P = 1$;
7. $\text{flag1} = 1$;
8. **while** $\text{flag1} == 1$
9. **while** $|S| < |N|$
10. $\text{Candidatos} = []$;
11. **for** cada nodo i de 1 a N


```

12.      if  $S[i] == 0$ 
13.           $Candidatos = [d[i]]$ ;
14.      if  $S[i]$  no es 0
15.           $Candidatos = [infinito]$ ;
16.      end if
17.  end for
18.   $u = \min (Candidatos)$ ;
19.   $S[u] = 1$ ;
20.  for cada nodo  $i$  de 1 a  $N$ 
21.      if  $d[u] + t[u][i] < d[i]$ 
22.           $d[i] = d[u] + t[u][i]$ ;
23.      end if
24.  end for
25.  end while
26.  if  $d[f] = infinito$ 
27.       $sp = [ ]$ ;
28.       $spcost = infinito$ ;
29.       $P = 0$ ;
30.  else
31.       $S[i] = S[f]$ 
32.       $sp = f$ ;
33.       $spcost = d[S[i]]$ ;
34.  end if
35.  while  $S[i] \neq S[s]$ 
36.      for cada Arco( $i, j$ ) existente donde  $i$  en  $V$  y  $j$  en  $V'$ 
37.          Encuentra los vecinos de  $S[i]$ ;
38.           $pred(j) = \min(d[vecinos])$ ;
39.           $pred(j) = i$ ;
40.          Agregar  $S[i]$  al inicio de  $sp$ ;
41.      end for
42.  end while
43.   $P = 1$ ;
44. end while
45.  $flag1 = 0$ ;
46. Devolver  $sp$  (camino más corto),  $spcost$  (costo del camino más corto);

```

En las líneas 1 a 7, se realiza la inicialización del código. Se definen las variables N (conjunto de nodos), S (conjunto de nodos visitados), s (nodo fuente), f (nodo destino) y t (grafo). Cabe aclarar que, t es una matriz de tiempos inicializada en el archivo "entrega3_JulianLuisa". Este grafo se realiza tomando en cuenta las distancias entre varios puntos específicos de la universidad y los tiempos que son necesarios para recorrer estos caminos, dependiendo de si el usuario toma las escaleras o los ascensores. La explicación detallada sobre esta construcción se encuentra en el apartado de la entrega 2. Retomando el tema principal del pseudocódigo, en estas líneas también se inicializan los conjuntos S y S' con el nodo fuente y los nodos restantes, respectivamente. Las distancias iniciales ($d[i]$) se establecen como infinito para cada nodo i en N , excepto para el nodo fuente s , cuya distancia se establece en 0. El predecesor del nodo fuente ($pred(s)$) se establece como "no determinado". La variable P se inicializa en 1 y $flag1$ se inicializa en 1.

A partir de la línea 8, comienza el bucle principal del código:

En las líneas 8 a 44 se ejecuta un bucle while que se repite mientras flag1 sea igual a 1. Este bucle contiene otro bucle while que se ejecuta mientras el tamaño del conjunto S sea menor que el tamaño del conjunto N. Para asegurarnos que se recorran todos los nodos.

En las líneas 11 a 17 se construye un conjunto de candidatos para determinar el nodo vecino más adecuado. Se itera sobre cada nodo i en N, y si el nodo i no ha sido visitado ($S[i] == 0$), se agrega su distancia actual ($d[i]$) al conjunto de candidatos. Si el nodo i ya ha sido visitado ($S[i]$ no es 0), se agrega infinito al conjunto de candidatos. Luego, se selecciona el nodo u con la distancia mínima de entre los candidatos y se marca como visitado asignando 1 al conjunto S en la posición correspondiente a u.

En las líneas 20 a 24, se actualizan las distancias de los nodos vecinos. Se itera sobre cada nodo i en N, y se verifica si la suma de la distancia $d[u]$ y el peso de la arista entre u e i ($t[u][i]$) es menor que la distancia actual $d[i]$. Si se cumple esta condición, se actualiza la distancia $d[i]$ con el nuevo valor.

En las líneas 26 a 30, se verifica si la distancia al nodo destino f es infinita. Si es así, se inicializan las variables sp (un array vacío), spcost (infinito) y P (0) para indicar que no se ha encontrado un camino válido hacia el nodo destino. En caso contrario, se copia el conjunto $S[f]$ en el conjunto $S[i]$, se asigna f al array sp y se asigna a spcost la distancia correspondiente al conjunto $S[i]$.

A partir de la línea 35, comienza otro bucle while que se repite mientras el nodo actual $S[i]$ sea diferente del nodo fuente $S[s]$.

En las líneas 36 a 41, se itera sobre cada arco (i, j) existente, donde i pertenece al conjunto de nodos visitados (V) y j pertenece al conjunto de nodos no visitados (V'). Se encuentran los vecinos del nodo $S[i]$, se actualiza el predecesor del nodo j con el mínimo de las distancias de los vecinos y se asigna el predecesor i al nodo j. Además, se agrega el nodo $S[i]$ al inicio del array sp.

En la línea 43, se vuelve a asignar 1 a la variable P, indicando que se ha encontrado un camino válido hacia el nodo destino.

Después de salir del segundo bucle while, se asigna 0 a flag1 para indicar que el bucle principal ha finalizado.

Finalmente, en la última línea del código, se devuelve el camino más corto (sp) y el costo del camino más corto (spcost) como resultado de la ejecución del algoritmo.

6 Resultados del Algoritmo vs Modelo Matemático

6.1 Escenario 1

Un estudiante de la universidad de los andes desea hacer un recorrido corto desde el quinto piso del edificio Mario Laserna al sexto piso del edificio contiguo, el W. El estudiante se va a desplazar a las 6:15 de la tarde y desea saber cuál es el camino que menos tiempo consumirá para su desplazamiento

Table 5. Conjuntos y Parámetros

Sets and Parameters	Content
N	{SD, ML, W, AU, RGD, LL, G, R, S1, S, O, Q, C, TX, B, V}
o	ML
d	W
v	{6:30, 8:00, 9:30, 11:00, 12:30, 14:00, 15:30, 17:00, 18:30}
$9_W, 10_{ML}$	{ML 10, W 9}
$18:15_{ML}$	18:15

Resultados Escenario 1

The screenshot shows a MATLAB script named 'Entrega3.m' and its execution results in the Command Window. The script implements a Dijkstra algorithm to find the shortest path from node ML5 to node W6. It converts a directed graph to an undirected one and then runs the algorithm. The output shows the shortest path as 'ML5 W5 W6' with a cost of 66.0000 seconds.

```

97         end
98     end
99 end
100 end
101
102 %Convertimos el grafo dirigido en no-dirigido.
103 for i=1:length(t)
104     for j=1:length(t)
105         if t(i,j) < inf
106             t(j,i)=t(i,j);
107         end
108     end
109 end
110 hora = '18:15';
111 origen='ML5';
112 destino='W6';
113 [sp, spcost] = dijkstra_v2(t, origen, destino) %sp: shortest path, spcost: shortest path cost

```

Command Window Output:

```

Tiempo entre nodos:
W6 - W5: 6.00
W5 - ML5: 60.00

sp =

    'ML5  W5  W6'

spcost =

    66.0000

>>

```

Los resultados obtenidos para este escenario fueron los mismos para la implementación de la heurística del algoritmo de Dijkstra que para la implementación del camino de mínimo costo en Pyomo. Al querer desplazarse del quinto piso del edificio ML al sexto piso del edificio W, se determinó que la ruta de menor costo consiste en cruzar el puente entre los edificios en el quinto piso y luego subir al sexto piso del edificio W. Este caso nos proporciona información adicional relevante. Aunque existen diferentes formas de llegar del quinto piso del edificio ML al sexto piso del edificio W, se eligió la opción con el tiempo en desplazamiento más corto en segundos (spcost = 66), en lugar de bajar al segundo piso del edificio ML, cruzar al primer piso del edificio W y luego subir al sexto piso del W. Esto demuestra que se optó por el camino con menor distancia. Basado en los parámetros definidos, como el horario de viaje a las 6:15 pm, que es un momento con poco congestionamiento, el programa reconoce que es más eficiente utilizar el ascensor. Dado que, en

horas sin congestión, el ascensor tarda solo 5 segundos en subir un piso, mientras que subir un piso por medio de las escaleras llevaría más de 6 segundos, conforme a la velocidad constante de 3 km/h establecida en el modelo. Por lo tanto, es apropiado que el modelo sugiera utilizar el ascensor.

6.2 Escenario 2

Una joven estudiante de la universidad de los andes tiene que llegar desde el primer piso del W, al quinto piso del edificio de TX. Ella quiere saber cómo puede desplazarse de la manera más eficiente, contando caminos cortos, escaleras o ascensores, si tiene clases a las 9:30 de la mañana y va tarde. Por lo que es esencial para ella poder conocer la ruta más rápida.

Table 6. Conjuntos y Parámetros

Sets and Parameters	Content
N	{SD, ML, W, AU, RGD, LL, G, R, S1, S, O, Q, C, TX, B, V}
o	W
d	TX
v	{6:30, 8:00, 9:30, 11:00, 12:30, 14:00, 15:30, 17:00, 18:30}
$9_{W, 7_{TX}}$	{W 9, TX 7}
$9:30_{ML}$	9:30

Resultados Escenario 2

```

97         end
98     end
99 end
100 end
101
102 %Convertimos el grafo dirigido en no-dirigido.
103 for i=1:length(t)
104     for j=1:length(t)
105         if t(i,j) < inf
106             t(j,i)=t(i,j);
107         end
108     end
109 end
110 hora = '9:30';
111 origen='W1';
112 destino='TX5';
113 [sp, spcost] = dijkstra_v2(t, origen, destino) %sp: shortest path, spcost: shortest path cost

```

```

Command Window

Tiempo entre nodos:
TX5 - TX3: 12.00
TX3 - R2: 36.00
R2 - R1: 6.00
R1 - W6: 156.00
W6 - W5: 6.00
W5 - W4: 6.00
W4 - W1: 18.00

sp =

    'W1W4W5W6R1R2TX3TX5'

spcost =

    234.0000

>>

```

Los resultados obtenidos en este escenario fueron consistentes tanto para la implementación de la heurística del algoritmo de Dijkstra como para la implementación del camino de mínimo costo en Pyomo. Aun así, este caso puede resultar menos intuitivo, especialmente para aquellos que no están familiarizados con el entorno de la Universidad de los Andes. A simple vista, puede no ser evidente la respuesta. En primer lugar, sabemos que nuestra estudiante desea viajar a las 9:30 de la mañana, un horario en el que la universidad está bastante congestionada debido al cambio de clases. Además, no sorprende a ningún miembro de nuestra comunidad que los ascensores del edificio W en estos horarios tengan filas con más de 10 personas. Cuando hay un mayor flujo de usuarios, es más probable que los ascensores se demoren, ya que la demanda aumenta y se detienen en la mayoría de los pisos antes de llegar al destino final. En nuestro modelo, tomar el ascensor implica un tiempo de espera de 20 segundos multiplicado por el número de pisos del edificio. Por esta razón, nuestro programa le recomienda a la estudiante que no pierda tiempo y que suba por las escaleras.

Debido a nuestra forma de modelar el ejercicio, el programa diferencia entre viajar desde el primer piso al cuarto, luego del cuarto al quinto y finalmente al sexto, porque en cada uno de estos pisos se pueden tomar decisiones diferentes, como ir al edificio C, al edificio ML u otro lugar. Sin embargo, el programa reconoce que lo más eficiente es subir del primer al sexto piso en el edificio W utilizando las escaleras. A una velocidad de 3 km/h, subir esos 25 metros inclinados le tomará medio minuto, mientras que en ascensor este tiempo puede ser mucho mayor.

Siguiendo la misma lógica, el programa evalúa posibles caminos para llegar desde el sexto piso del edificio W hasta el edificio TX, pero determina que pasar por el edificio R es lo óptimo. Finalmente, utilizando la misma lógica en horas pico, el programa concluye que la mejor opción para subir al quinto piso del edificio TX es utilizar las escaleras con el tiempo total de desplazamiento más corto en segundos (spcost = 234).