Optimización determinista

Fase 2 Reto

Víctor Augusto Godínez Velasco - A01656075
Íker Sebastian Bali Elizalde - A01656437
Dabria Camila Carrillo Meneses - A01656716
Alan Uriel Merlan Esquivel - A01656612
Takeshi Nakauma Aguirre - A01656026

Marzo, 2023

Abstract

Teniendo como objetivo central el problema de asignación de profesores con restricciones específicas dado por el socio formador, en este documento se abarcan diferentes temas de varios tipos de asignación así como temas de programacion lineal y herramientas, como las librerías **pulp**[1] y **munkres**[2] en Python. En este proyecto se logró aprender los elementos básicos de la optimización y la complejidad que puede llegar a tener, así como plasmar diferentes ideas recopiladas en equipo en un código que ayude a resolver el reto de la mejor manera.

Contents

1 Planteamiento del reto

El reto consiste en realizar un programa/algoritmo que ayude y permita la asignación óptima de materias por departamento. En dicho algoritmo existen diferentes restricciones como que cada materia tiene que ser impartida por un sólo profesor en un solo horario. También, un profesor puede dar más de una materia, pero lo tendría que hacer en horarios distintos. Un profesor no puede estar dando dos materias distintas en el mismo horario, etc. Aunque es deseable que todos los profesores tengan materia, no es estrictamente necesario. El programa debe de funcionar con cualquier lista de profesores, materias, horarios, salones,y restricciones. Se sabe que la organización de horarios de una universidad puede ser bastante complicado debido a que se trabaja con insumos, objetivos, contratiempos, cambios, casos especiales y muchos problemas más, además de tener como otros retos la sistematización de la programación, ligar de manera más eficiente los archivos y tener una lista de profesores. Dentro del trabajo se buscó llegar a la mejor solución basándonos en los objetivos planteados y los obstáculos que se confrota la universidad semestre con semestre.

2 Simplificaciones que se hicieron al reto en su equipo

Durante la realización del equipo , las simplificaciones que se hicieron fueron las siguientes:

[1]Ver `https://coin-or.github.io/pulp/`

[2]Ver `https://pypi.or/project/munkres`

- Se contempla que una materia solo puede ser dada por un profesor , sin embargo, en la aplicación real, las materias pueden ser dadas por más de un profesor.
- Se contempla que la cantidad de demanda de una materia no es necesaria para impartirla.
- Se contempla que cada materia viene ligada a un solo grupo, por lo que no se puede repetir, cuando en realidad, una materia puede impartirse varias veces en distintos grupos.
- Se contempla que existe un máximo de 10 profesores, materias, horarios y salones con sus respectivas restricciones.
- Se contempla que la importancia de cada materia es la misma, es decir, que no se le debe dar preferencia a ninguna.

3 Logros del programa elaborado

Respecto a los logros del código hecho por el equipo, el equipo logró lo siguiente:

- Ejecutar el programa cubriendo los costos de materias, profesores, horarios y salones.
- Establecer restricciones adecuadas al modelo que se pedía al principio.
- Usar el programa como un API en excel.
- Hacer un problema de programación lineal para la optimización de horarios con una complejidad intermedia.
- Reforzamos nuestras habilidades necesarias para realizar este algoritmo.

4 Áreas de mejora del programa

Dentro del proyecto se tienen distintas áreas de mejora, tomando es cuenta que el tiempo de trabajo fue muy reducido y la fecha de entrega muy pronta, algunas de ellas son:

- Hacer que tenga más eficiencia espacial en relación a la memoria. Se ocupó mucha memoria debido a que tiene demasiadas variables, consideramos que hay maneras de sintetizar el número de variables usadas.
- Aumentar y detallar las restricciones en el programa con el fin de ser más precisos en lo que puede hacer el código con el fin de facilitar la forma de trabajo del socio formador.
- Se podrían agregar extensiones al programa, por ejemplo vincular el código a la página web del Tec o a dónde se realizan los horarios con el fin de generar y recibir actualizaciones del proceso.
- Tomar en cuenta las utilidades que podría generar el programa y tener un pronóstico de comportamiento de estas.
- Personalizar los horarios de los alumnos por medio del programa, tomando las necesidades de la persona y generar una satisfacción al usuario tanto al personal como al usuario.

5 Ejemplos de uso del programa

Para los ejemplos de prueba del código, se usarán los siguientes casos:

Profesores	Materias	Horarios	Salones	Materias que puede dar cada profesor	Horario que puede dar cada profesor	Costos Profesores	Costos Materias	Costos Horarios	Costos Salones
A	h	1	a	hr	1	100	50	10	30
B	j	2	b	joe	2	200		30	30
C	o	3		hjo	12	100	40	30	20
	r	4	d	ho	13	100		20	20
E	e	5		oj	21	100	50		
		6		oh	13	101	40		
				jh	31				
				j	3	103			
I				o	1	104			
---	---	---	---	---	---	---	---	---	---
hasta aquí									
				Materias Default	Horarios Default	Costos Default			
				hjore	123456	1000			

Figure 1: Datos Caso 1

Resultados				
Profesor	Materia	Horario	Salon	
A	r	1	b	
B	e	2	a	Costo
C	h	2	d	5860
C	j	1	d	
E	o	1	a	

Figure 2: Resultado Caso 1 Obtuvo la respuesta correcta

Profesores	Materias	Horarios	Salones	Materias que puede dar cada profesor	Horario que puede dar cada profesor	Costos Profesores	Costos Materias	Costos Horarios	Costos Salones
A	h	1	a	hre	1356	100	50	10	30
	j	2	b	joe	2	200		30	30
C	o	3		hjo	1234	100	40	30	20
	r	4	d	ho	13	100		20	20
	e	5		oj	21	100	50		
		6		oh	13	101	40		
				jh	31				
				j	3	103			
				o	1	104			
---	---	---	---	---	---	---	---	---	---
hasta aquí									
				Materias Default	Horarios Default	Costos Default			
				hjore	123456	1000			

Figure 3: Datos Caso 2

Resultados				
Profesor	Materia	Horario	Salon	
A	r	3	d	
A	e	1	b	Costo
C	h	1	d	3810
C	j	2	d	
C	o	4	d	

Figure 4: Resultado Caso 2 Obtuvo la respuesta correcta

Resultados				
Profesor	Materia	Horario	Salon	
A	r	1	a	
B	e	2	d	Costo
C	o	4	d	3710
E	j	1	d	

Figure 5: Datos Caso 3

Profesores	Materias	Horarios	Salones	Materias que puede dar cada profesor	Horario que puede dar cada profesor	Costos Profesores	Costos Materias	Costos Horarios	Costos Salones
A		1	a	re	1	100	50	10	30
B	j	2	b	joe	2	200		30	30
C	o	3		jo	4	100	40	30	20
D	r	4	d	o	13	100		20	20
E	e	5		oj	21	100	50		
F		6		o	13	101	40		
G				j	31				
H				j	5	103			
I				o	6	104			
----------	----------	----------	----------	----------	----------	----------	----------	----------	----------
hasta aquí									
				Materias Default	Horarios Default	Costos Default			
				jore	123456	1000			

Figure 6: Resultado Caso 3 Obtuvo la respuesta correcta

Profesores	Materias	Horarios	Salones	Materias que puede dar cada profesor	Horario que puede dar cada profesor	Costos Profesores	Costos Materias	Costos Horarios	Costos Salones
A		1	a	re	1		50	10	30
B	j		b	joe	15		10	30	30
C	o			jo	4		40	30	20
D	r	4	d	o	14		10	20	20
E	e	5		oj	1		50		
F		6		o	15		40		
G				j	61				
H				j	5				
I				o	6				
----------	----------	----------	----------	----------	----------	----------	----------	----------	----------
hasta aquí									
				Materias Default	Horarios Default	Costos Default			
				jore	1456	1000			

Figure 7: Datos Caso 4

Resultados				
Profesor	Materia	Horario	Salon	
A	r	1	a	
B	e	1	b	Costo
C	o	4	d	6210
E	j	1	d	

Figure 8: Resultado Caso 4 Obtuvo la respuesta correcta

Profesores	Materias	Horarios	Salones	Materias que puede dar cada profesor	Horario que puede dar cada profesor	Costos Profesores	Costos Materias	Costos Horarios	Costos Salones
A		1	a	re	1				
B	j		b	joe	15				
C	o			jo	4				
D	r	4	d	o	14				
E	e	5		oj	1				
F		6		o	15				
G				j	61				
H				j	5				
I				o	6				
----------	----------	----------	----------	----------	----------	----------	----------	----------	----------
hasta aquí									
				Materias Default	Horarios Default	Costos Default			
				jore	1456	1000			

Figure 9: Datos Caso 5

Resultados				
Profesor	Materia	Horario	Salon	
A	r	1	b	
B	j	5	b	Costo
B	e	1	a	16000
I	o	6	d	

Figure 10: Resultado Caso 5 Obtuvo la respuesta correcta

Profesores	Materias	Horarios	Salones	Materias que puede dar cada profesor	Horario que puede dar cada profesor	Costos Profesores	Costos Materias	Costos Horarios	Costos Salones
A		1	a						
B	j		b						
C	o								
D	r	4	d						
E	e	5							
F		6							
G									
H									
I									
----------	----------	----------	----------	----------	----------	----------	----------	----------	----------
hasta aquí									
				Materias Default	Horarios Default	Costos Default			
				jore	1456	1000			

Figure 11: Datos Caso 6

Resultados				
Profesor	Materia	Horario	Salon	
A	o	6	d	
A	e	5	d	Costo
B	j	1	a	16000
C	r	6	a	

Figure 12: Resultado Caso 6 Obtuvo la respuesta correcta

Profesores	Materias	Horarios	Salones	Materias que puede dar cada profesor	Horario que puede dar cada profesor	Costos Profesores	Costos Materias	Costos Horarios	Costos Salones
A	h	1	a	hr	1	100	50	10	
B	j	2	b	joe	2	200			30
C	o	3	c	hjo	12	100	40	30	20
D	r	4	d	ho	13	100			20
E	e	5		oj	21	100			
F		6		oh	13	101			
G				jh	31				
H				j	3	103			
I				o	1	104			
---	---	---	---	---	---	---	---	---	---
hasta aquí									
				Materias Default	Horarios Default	Costos Default			
				hjore	123456	1000			

Figure 13: Datos Caso 7

	Resultados					
	Profesor	Materia	Horario	Salon		
	A	r	1	d		
	B	e	2	c	Costo	
	C	j	1	c	5820	
	D	h	3	c		
	D	o	1	b		

Figure 14: Resultado Caso 7 Obtuvo la respuesta correcta

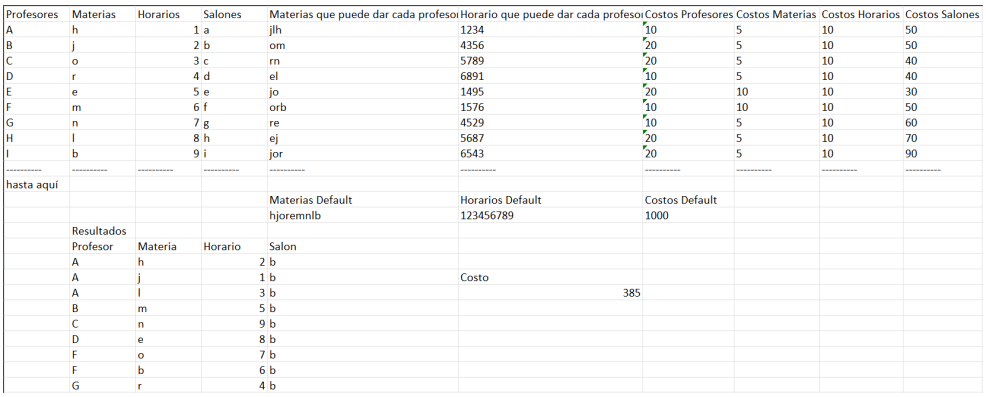
6 Interfaz gráfica

Para esta parte del trabajo, se nos pedía una interfaz grafica tipo API, para que el usuario que quiera realizar pruebas en nuestro programa se le hiciera más fácil en cuanto a la interacción. Lo que hicimos nosotros fue realizar con ayuda de la liberia de xlwings una interfaz grafica en excel en donde nosotros le podemos poner las restricciones de los profesores, materias, horarios y salones donde indiquemos cuantos hay de cada uno y por ejemplo que profesores pueden dar que materia.
Aquí incluimos una imágen de como se ve nuestra interfaz gráfica vacía en donde le vamos a meter los datos previamente establecidos.

Profesores	Materias	Horarios	Salones	Materias que puede dar cada profesor	Horario que puede dar cada profesor	Costos Profesores	Costos Materias	Costos Horarios	Costos Salones
---	---	---	---	---	---	---	---	---	---
hasta aquí									
				Materias Default	Horarios Default	Costos Default			
						1000			
	Resultados								
	Profesor	Materia	Horario	Salon					
					Costo				

Figure 15: Interfaz gráfica para el usuario

Este es un ejemplo de como puede quedar la API despues de que el usuario le haya añadido los inputs correspondientes y en el cual nos arroja un resultado de la solucion al problema que planteamos en un principio y como extra el costo de implementarlo.



Profesores	Materias	Horarios	Salones	Materias que puede dar cada profesor	Horario que puede dar cada profesor	Costos Profesores	Costos Materias	Costos Horarios	Costos Salones
A	h	1	a	jlh	1234	10	5	10	50
B	j	2	b	om	4356	20	5	10	50
C	o	3	c	rn	5789	20	5	10	40
D	r	4	d	el	6891	10	5	10	40
E	e	5	e	jo	1495	20	10	10	30
F	m	6	f	orb	1576	10	10	10	50
G	n	7	g	re	4529	10	5	10	60
H	l	8	h	ej	5687	20	5	10	70
I	b	9	i	jor	6543	20	5	10	90
----------	----------	----------	----------	----------	----------	----------	----------	----------	----------
hasta aquí									
				Materias Default	Horarios Default	Costos Default			
				hjoremnlb	123456789	1000			
	Resultados								
	Profesor	Materia	Horario	Salon					
	A	h	2	b					
	A	j	1	b	Costo				
	A	l	3	b	385				
	B	m	5	b					
	C	n	9	b					
	D	e	8	b					
	F	o	7	b					
	F	b	6	b					
	G	r	4	b					

Figure 16: Interfaz gráfica con valores

Los únicos requisitos que hay que tener para poder hacer uso de la API, es tener instalado en su computadora las librerías correspondientes que se usaron en el programa (pulp, xlwings) y consecuentemente tambien tener instalado Microsoft Excel para poder introducir los inputs que se requieran.
Algunos consejos que debemos de darles para hacer uso de la API es que al momento de meter materias, tenemos que asegurarnos de que todas las materias estén asignadas al menos a 1 profesor, ya que, si no se realiza esto puede ser que el código nos marque error y no funcione. Otro consejo es que si quitamos alguna materia de la lista de materia, tenemos que asegurarnos que no esté asignada a ningún profesor después, porque si es así estaríamos diciendo que un profesor puede dar una materia que no existe y por lo tanto va a marcar error. Estos son algunos de los consejos que tenemos para el funcionamiento correcto del programa.
Otros consejos para que sirva bien la interfaz es que si por ejemplo queremos poner un caso en donde haya muy pocos profesores y muchas materias, tenemos que tener muchas opciones de horarios y salones ya que vamos a recurrir que los profesores se les asignen varias materias y la única manera de que pueda pasar eso es que si tienen flexibilidad en sus horarios. Podemos poner casos en donde no se ponga ningun valor de los costos de ninguna de las cosas, ya que pusimos valores por 'default' los cuáles se van a agarrar si no les ponemos las restricciones correspondientes las cuáles van a significar que todos los profes pueden dar todas las materias, en todos los horarios y van a tener el costo máximo.

7 Apéndice

Aquí anexamos el código íntegro de Python en el que implementamos el programa solucionador del problema.

```
tabsize

Programa de optimizaci n de horarios
Creamos las fuciones que vamos a utilizar para generar las restricciones extraidas de la API
def qn(list): # funcion para quitar los elementos NONE que puedan aparecer en nuestra API
 list2 = []
 for val in list:
  if val != None:
   list2.append(val)
 return list2

def aresm(dictp,listp): # funcion para agregar las restricciones de las materias
 for val in listp:
  for i in range(9):
   for j in range(10):
    if val == c[i+1,j].value:
```

```
     if c[i+1,j+4].value == None:
      dictp[val] = list((c.range('E14').value))
     else:
      dictp[val] = list((c[i+1,j+4].value))
   del j
  del i
 del val
 return dictp

def aresh(dicth,listp): # funcion para agregar las restricciones de los horarios
 for val in listp:
  for i in range(9):
   for j in range(10):
    if val == c[i+1,j].value:
     if c[i+1,j+5].value == None:
      dicth[val] = list(str(c.range('F14').value))
     else:
      dicth[val] = list(str(int(c[i+1,j+5].value)))
   del j
  del i
 del val
 return dicth

def acos(dict,lista,t): # funcion para agregar los costos
 for val in lista:
  if t == 's':
   for i in range(len(lista)+1):
    for j in range(10):
     if val == c[i+1,j].value:
      lista_temp = []
      if c[i+1,j+6].value == None:
       lista_temp.append((c.range('G14').value))
      else:
       lista_temp.append((c[i+1,j+6].value))
      dict[val] = list(map(int,lista_temp))
    del j
   del i
  else:
   for i in range(10):
    for j in range(len(lista)+1):
     if val == c[i+1,j].value:
      lista_temp = []
      if c[i+1,j+6].value == None:
       lista_temp.append((c.range('G14').value))
      else:
       lista_temp.append((c[i+1,j+6].value))
      dict[val] = list(map(int,lista_temp))
    del j
   del i
 del val
 return dict

#Importamos las librerias
import pulp
import xlwings as xl

# creamos nuestra funci n que har  el problema de optimizaci n
def asignaciones(t_l,m_l,h_l,s_l,r_profe_materia,r_profe_horario,costo_profe,costo_salon,costo_horario,costo_materia):

 # Iniciamos el problema en pulp de tal forma que minimice nuestra funci n objetivo
 prob = pulp.LpProblem("Teachers", sense = pulp.const.LpMinimize)

 #Usamos pulp para crear todas las variables con las combinaci nes posibles dadas nuestros horarios.
 variables = pulp.LpVariable.dicts("X_",(t_l,m_l,h_l,s_l),0, None, pulp.const.LpBinary)

 #FUNCI N OBJETIVO.
 #Fo es una variable que sirve para acumular los diferentes costos datos por los profesores, materias, horarios y salones
 fo = pulp.lpSum([])

 #Esta parte agrega el costo por profesor. El proceso es que
 # va por cada profesor y checa cada costo para luego multiplicarlo por la variable correspondiente.
 if costo_profe != None:
  for t in t_l:
   for i in costo_profe[t]:
    aux =  pulp.lpSum([i*variables[t][m][h][s] for m in m_l for h in h_l for s in s_l])
   fo = fo+aux
   aux  = pulp.lpSum([])
  del i

 #Luego, agregamos el costo por salon(el proceso es el mismo)
 if costo_salon != None:
  for s in s_l:
   for i in costo_salon[s]:
    aux =  pulp.lpSum([i*variables[t][m][h][s] for m in m_l for h in h_l for t in t_l])
   fo = fo+aux
   aux = pulp.lpSum([])
  del i

 #Agregamos el costo por horario
 if costo_horario != None:
  for h in h_l:
   for i in costo_horario[h]:
    aux =  pulp.lpSum([i*variables[t][m][h][s] for m in m_l for t in t_l for s in s_l])
   fo = fo+aux
   aux  = pulp.lpSum([])
  del i
```

```
 #Agregamos el costo por materia
if costo_materia != None:
 for m in m_l:
  for i in costo_materia[m]:
   aux =  pulp.lpSum([i*variables[t][m][h][s] for h in h_l for t in t_l for s in s_l])
  fo = fo+aux
  aux  = pulp.lpSum([])
 del i

#Y el resultado es una funci n que toma en cuenta todos los costos
prob+= fo

#En caso de que ningun costo es dado, se asegura de que aun funcione
if (costo_profe == None) and (costo_materia == None) and (costo_horario == None) and (costo_salon == None):
 prob += pulp.lpSum([variables[t][m][h][s] for h in h_l for t in t_l for s in s_l for m in m_l])

#Restricci n 1 _ Una materia {m} solo puede tener un sal n , un horario y un profesor
#Para lograr lo anterior, nos aseguramos por medio del primer for que cada m(de manera que la m por restricci n es constante)
#  se separe.
# Por lo que cada materia m solo puede tener un sal n , horario y profesor.
#El proceso es el mismo para el resto de restricciones. Por tanto, solo se mencionar  que variable
# se mantiene constante en cada restricci n cuando se explique el funcionamiento.
for m in m_l:
 prob += (pulp.lpSum([variables[t][m][h][s] for t in t_l
                     for h in h_l
                     for s in s_l]) == 1,
                     f" Restricci n 1: Una materia {m} solo puede tener un sal n , un horario y un profesor ")

#Restricci n 2. "Restricci n 2: El profesor {t} est  capacitado para dar la materia {m}
# Para esto, se mantiene la t constante. Y por cada diferente t(profesor) vericamos
#que materia pueda dar observando el diccionario r_profe_materia atraves del segundo for.
for t in t_l:
 for m in r_profe_materia[t]:
  prob += (pulp.lpSum([variables[t][m][h][s] for h in h_l for s in s_l])) <= 1,
  f"Restricci n 2: Un profesor {t} est  capacitado para la materia {m} "

#Restricci n 3: El profesor {t} NO puede dar {m}"
# Aqu  igual mantenemos t constante. Ahora pasamos por tadas las materias y verificamos que el profesor no las de.
# Si es as  , agregamos una restricci n para que ese profesor no de esa materia
for t in t_l:
 for m in m_l:
  if (m in r_profe_materia[t]) == 0:
   prob += (pulp.lpSum([variables[t][m][h][s]
                       for h in h_l
                       for s in s_l]) == 0,
                       f"Restricci n 3: El profesor {t} NO puede dar {m}")

#Restricci n4. Un profesor {t} est  disponible en el horario {h}
# Para esto, se mantiene la t constante. Y por cada diferente t(profesor) vericamos
#que horario puede dar observando el diccionario r_profe_horario atraves del segundo for.
for t in t_l:
 for h in r_profe_horario[t]:
  prob += pulp.lpSum([variables[t][m][h][s] for m in m_l for s in s_l]) <= 1,
  f"Restricci n 4: Un profesor {t} est  disponible en el horario {h}"

# Restricci n 5: El profesor {t} no puede dar clase en el horario {h}
# Mantenemos t constante. Ahora pasamos por tadas los horarios y verificamos que el profesor no este disponible.
# Si es as  , agregamos una restricci n que se asegure que nunca de en ese horario a traves del segundo for

for t in t_l:
 for h in h_l:
  if (h in r_profe_horario[t]) == 0:
   prob += (pulp.lpSum([variables[t][m][h][s] for m in m_l for s in s_l])) == 0,
   f"Restricci n 5: El profesor {t} no puede dar clase en el horario {h}"

#Restriccion 6 Restricci n 6: El sal n {s} solo puede ser ocupado una vez en el horario {h}"
#s,h constante
for s in s_l:
 for h in h_l:
  prob += (pulp.lpSum([variables[t][m][h][s] for m in m_l for t in t_l]) <= 1,
  f"Restricci n 6: El sal n {s} solo puede ser ocupado una vez en el horario {h}")

#Restriccion 7. Restricci n 7: Un profesor {t} solo pueda dar una materia en horario {h}
# t,h constante
for t in t_l:
 for h in h_l:
  prob +=  pulp.lpSum([variables[t][m][h][s] for m in m_l for s in s_l  ]) <= 1,
  f"Restricci n 7: Un profesor {t} solo pueda dar una materia en horario {h}"

#En base a las restricciones y la funci n objetivo, resuelve el problema
status = prob.solve()
#Adjunta los combinaciones que dan el costo minimo en una lista.
l_resultado = []
for t in t_l:
 for m in m_l:
  for h in h_l:
   for s in s_l:
    if variables[t][m][h][s].value() == 1 :
     l_resultado.append([t,m,h,s])

print(f"Entonces el minimo costo posible es: {pulp.value(prob.objective)}")

#Regresa una lista con los datos de que minimizan los costos.
return l_resultado, (pulp.value(prob.objective))
```

```
# Mandamos a llamar el excel donde ponemos los datos de los profesores, materias, horarios y salones con sus respectivas restricciones
wb = xl.Book(r'Datos_profesores.xlsx')
c=wb.sheets[0] # cuaderno donde se encuentran nuestros datos

# Datos recolectados desde excel
c.range('B17:E30').value = '' # reiniciamos los resultados
c.range('F19').value = ''

# Listas de profesores, materias, horarios y salones
prof = qn(list((c.range('A2:A10').value)))
mat = qn(list((c.range('B2:B10').value)))
hor1 = list(map(str,list(map(int,qn(list((c.range('C2:C10').value)))))))
hor = qn(list((c.range('C2:C10').value)))
sal = qn(list((c.range('D2:D10').value)))

# Restricciones de las materias que puede dar cada profesor
rpm = dict()
rpm = aresm(rpm,prof)

# Restricciones de los horarios que puede impartir cada profesor
rph = dict()
rph = aresh(rph,prof)

# Costos de profesores, materias, horarios y salones
cp = dict()
cp = acos(cp,prof,'p')
cm = dict()
cm = acos(cm,mat, 'm')
ch = dict()
ch = acos(ch,hor, 'h')
ch_temp = list(map(str,list(map(int,ch))))
for val in ch_temp:
 ch[val] = ch.pop(float(val))
del ch_temp
cs = dict()
cs = acos(cs,sal,'s')

resultado, costo_fin = asignaciones(prof,mat,hor1,sal,rpm,rph,cp,cs,ch,cm)
print(resultado)

# Vamos a pegar nuestro resultado al Excel
c.range('B17').value = resultado
c.range('F19').value = costo_fin

del c,ch,cm,cp,cs,hor,hor1,mat,prof,sal,resultado,rph,rpm,val,wb,costo_fin
```

Ahora anexaremos la Liga de Collab en donde está nuestro código por si el usuario quiere probar el funcionamiento de este. https://colab.research.google.com/drive/1IwbLHtGjKp7C8ywc65BLD1HYjUc2rZOM?usp=sharing

Tambien anexaremos el libro de excel que es nuestra API, la cual el usuario va a poder descargar y ponerla en su entorno de ejecución del programa para que pueda hacer uso correcto del programa. https://drive.google.com/drive/folders/1LvVzJ5EchF5GUDVrUR62cKIFVatFkuaO?usp=sharing