

6. Durante la construcción de una casa, se deben recortar seis viguetas de 24 pies cada una a la longitud correcta de 23 pies. La operación de recortar una vigueta implica la siguiente secuencia:

Operación	Tiempo (segundos)
Colocar la vigueta en caballetes de aserrar	15
Medir la longitud correcta (23 pies)	5
Marcar la línea de corte para la sierra circular	5
Recortar la vigueta a la longitud correcta	20
Apilar las viguetas recortadas en un área designada	20

Intervienen tres personas: Dos deben realizar al mismo tiempo las operaciones 1, 2 y 5, y un cortador se ocupa de las operaciones 3 y 4. Hay dos pares de caballetes de aserrar donde se colocan las viguetas sin recortar, y cada par puede manejar tres viguetas. Sugiera un buen plan para recortar las seis viguetas.

Parámetros

$T_C = 15$: *Tiempo para Operacion de Colocado*

$T_M = 5$: *Tiempo para Operacion de Medicion*

$T_L = 5$: *Tiempo para Operacion de Marcado*

$T_R = 20$: *Tiempo para Operacion de Recortado*

$T_A = 20$: *Tiempo para Operacion de Apilado*

Variables de decisión

El proceso de mecanizar las 6 vigas será seccionado en bloques b en las cuales cada persona A, B, C realizarán ciertas operaciones sobre las vigas:

$t_A(b)$: Tiempo que necesita la persona A en el bloque b

$t_B(b)$: Tiempo que necesita la persona B en el bloque b

$t_C(b)$: Tiempo que necesita la persona C en el bloque b

Función Objetivo

El objetivo es minimizar el tiempo total al mecanizar las 6 vigas:

Se identificaron 8 bloques

- **Ronda 1**
A y B colocan, miden las viguetas 1 y 2
- **Ronda 2-7**
A y B colocan, miden y apilan según se requiera
C marca y corta viguetas en orden
- **Ronda 8**
B apila la última vigueta

Por lo tanto, el tiempo total estará dado por la suma de los tiempos máximos de cada bloque:

$$\min T = \sum_{r=1}^8 \max (t_A(b), t_B(b), t_C(b))$$

Restricciones

Orden de acciones

Se debe seguir un orden lógico de acciones. Es decir, no puedes cortar sin antes haber marcado.

Ejemplo factible:

$b = 1$

- A: Coloca y mide la vigueta 1
- B: Coloca y mide la vigueta 2
- C: Espera

$b = 2$

- A: Coloca y mide la vigueta 3
- B: Coloca y mide la vigueta 4
- C: Marca y corta la vigueta 1

$b = 3$

- A: Apila la vigueta 1
- B: Coloca y mide la vigueta 6
- C: Marca y corta la vigueta 2

$b = 4$

- A: Coloca y mide la vigueta 5
- B: Apila la vigueta 2
- C: Marca y corta la vigueta 3

$b = 5$

- A: Apila la vigueta 3
- B: Espera
- C: Marca y corta la vigueta 4

$b = 6$

- A: Espera
- B: Apila la vigueta 4
- C: Marca y corta la vigueta 5

$b = 7$

- A: Apila la vigueta 5
- B: Espera
- C: Marca y corta la vigueta 6

$b = 8$

- A: Termina sus tareas
- B: Apila la vigueta 6

- C: Termina sus tareas

Después de pensar, la implementación del modelo a un código de Python no fue posible.

7. Se construye una pirámide (bidimensional) en cuatro capas. La capa inferior se compone de los puntos (equidistantes) 1, 2, 3 y 4; la siguiente incluye los puntos 5, 6 y 7; la tercera comprende los puntos 8 y 9, y la superior el punto 10. Lo que se quiere es invertir la pirámide (que la capa inferior incluya un punto y la superior cuatro) cambiando de lugar los puntos.
- Identifique dos soluciones factibles.
 - Determine el número mínimo de movimientos necesarios para invertir la pirámide.

Parámetros

En este problema, los parámetros será cantidad de puntos:

$$i = 10$$

La cantidad de capas:

$$j = 4$$

Información sobre la posición inicial de los puntos por capa:

$$C_1 = \{1,2,3,4\}$$

$$C_2 = \{5,6,7\}$$

$$C_3 = \{8,9\}$$

$$C_4 = \{10\}$$

Variables de decisión

Podemos ver las 4 capas como una matriz de 4x4:

Posición Inicial

10			
8	9		
5	6	7	
1	2	3	4

Posición Final

10	2	3	4
8	9	7	
5	6		
1			

Aquí, con solo 4 movimientos logramos el objetivo.

Tenemos 16 posiciones en donde pueden moverse 10 diferentes puntos, pero los puntos solamente pueden moverse de una capa a otra, por lo que, cada punto realmente solo tiene 4 posiciones: en la capa 1,2,3 o 4, es decir, para nuestro modelo, estas tablas son equivalentes:

10			
8	9		
5	6	7	
1	2	3	4

10			
8	9		
7	6	5	
4	2	3	1

Por lo tanto: para el punto i se tienen 4 variables de decisión, por ejemplo:

M_{11} : Mover el punto 1 a la capa 1

M_{12} : Mover el punto 1 a la capa 2

M_{13} : Mover el punto 1 a la capa 3

M_{14} : Mover el punto 1 a la capa 4

Y así sucesivamente, por lo que se tendrán 40 variables de decisión...

De la misma forma, nos hace falta una variable que nos permita registrar la posición de los puntos:

$$x_{ij}$$

Donde i representa el punto, y j la capa

De la misma forma, tendremos otras 40 variables de decisión

Función Objetivo

Nuestro objetivo es minimizar el número de movimientos, como M_{ij} representa cada movimiento:

Minimizar

$$\sum_{j=1}^4 \sum_{i=1}^{10} M_{ij}$$

Restricciones

Se tienen que cumplir las siguientes restricciones:

Cantidad de puntos en las capas finales

La variable x_{ij} nos ayudará a cumplir con la cantidad de puntos que debe tener cada capa al final:

$$x_{14} + x_{24} + x_{34} + x_{44} + x_{54} + x_{64} + x_{74} + x_{84} + x_{94} + x_{104} = 4$$

$$x_{13} + x_{23} + x_{33} + x_{43} + x_{53} + x_{63} + x_{73} + x_{83} + x_{93} + x_{103} = 3$$

$$x_{12} + x_{22} + x_{32} + x_{42} + x_{52} + x_{62} + x_{72} + x_{82} + x_{92} + x_{102} = 2$$

$$x_{11} + x_{21} + x_{31} + x_{41} + x_{51} + x_{61} + x_{71} + x_{81} + x_{91} + x_{101} = 1$$

Movimientos aceptados

Cada capa debe de contener la cantidad de movimientos necesarios para completar la cantidad de puntos requeridos, esto se puede ver por medio de las restricciones:

$$M_{14} + M_{24} + M_{34} + M_{44} + M_{54} + M_{64} + M_{74} + M_{84} + M_{94} + M_{104} = 3$$

$$M_{13} + M_{23} + M_{33} + M_{43} + M_{53} + M_{63} + M_{73} + M_{83} + M_{93} + M_{103} = 2$$

$$M_{12} + M_{22} + M_{32} + M_{42} + M_{52} + M_{62} + M_{72} + M_{82} + M_{92} + M_{102} = 0$$

$$M_{11} + M_{21} + M_{31} + M_{41} + M_{51} + M_{61} + M_{71} + M_{81} + M_{91} + M_{101} = 0$$

Esto se puede leer a que la capa 4 necesita recibir 4 puntos

La capa 3, 2 puntos

Y las capas 1 y 2, no ocupan recibir puntos.

Al hacer un script en Python con estas restricciones se tiene:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  SQL CONSOLE

Objective value:          4.00000000
Enumerated nodes:         0
Total iterations:         0
Time (CPU seconds):       0.01
Time (Wallclock seconds): 0.01

Option for printingOptions changed from normal to all
Total time (CPU seconds):  0.02  (Wallclock seconds):  0.02

Solución:
Movimientos: 4.0
  2 a 4
  3 a 4
  4 a 4
  5 a 3
PS E:\Documentos\UANL\6\Investigacion de Operaciones\AMPL\Py> 
```

Las variables son:

$$M_{24} = 1$$

$$M_{34} = 1$$

$$M_{44} = 1$$

$$M_{53} = 1$$

El modelo final es:

Minimizar

$$\sum_{j=1}^4 \sum_{i=1}^{10} M_{ij}$$

Sujeto a

$$x_{14} + x_{24} + x_{34} + x_{44} + x_{54} + x_{64} + x_{74} + x_{84} + x_{94} + x_{104} = 4$$

$$x_{13} + x_{23} + x_{33} + x_{43} + x_{53} + x_{63} + x_{73} + x_{83} + x_{93} + x_{103} = 3$$

$$x_{12} + x_{22} + x_{32} + x_{42} + x_{52} + x_{62} + x_{72} + x_{82} + x_{92} + x_{102} = 2$$

$$x_{11} + x_{21} + x_{31} + x_{41} + x_{51} + x_{61} + x_{71} + x_{81} + x_{91} + x_{101} = 1$$

$$M_{14} + M_{24} + M_{34} + M_{44} + M_{54} + M_{64} + M_{74} + M_{84} + M_{94} + M_{104} = 3$$

$$M_{13} + M_{23} + M_{33} + M_{43} + M_{53} + M_{63} + M_{73} + M_{83} + M_{93} + M_{103} = 2$$

$$M_{12} + M_{22} + M_{32} + M_{42} + M_{52} + M_{62} + M_{72} + M_{82} + M_{92} + M_{102} = 0$$

$$M_{11} + M_{21} + M_{31} + M_{41} + M_{51} + M_{61} + M_{71} + M_{81} + M_{91} + M_{101} = 0$$

$$M_{i,j}, x_{i,j} \geq 0$$

8. Cuenta con cuatro cadenas y cada una consta de tres eslabones sólidos. Tiene que hacer un brazalete conectando las cuatro cadenas; romper un eslabón cuesta 2 centavos, y volverlo a soldar 3 centavos.
 - Identifique dos soluciones factibles y evalúelas.
 - Determine el costo mínimo para hacer el brazalete.

Parámetros

$$\text{Brazaletes: } n = 4$$

$$\text{Eslabones: } m = 3$$

$$C_R = 2$$

$$C_S = 3$$

Aquí se puede hacer una simplificación, ya que, para cada corte, se va a requerir hacer un soldado, por lo que existirá la misma cantidad de C_R que C_S , podemos agruparlo en una variable global

$$C = 2 + 3 = 5$$

Variables de decisión

Aquí solamente se va a especificar los cortes y soldadas de eslabones:

$$x_{i,j}$$

Donde la variable x representa el corte y la soldada del eslabón de la posición i de la cadena j

Aquí se tiene que hacer la siguiente suposición para cada operación de corte-soldado

El eslabón de la cadena j se va a conectar al eslabón primer eslabón de la cadena $j + 1$

Por ejemplo, esta variable nos da la siguiente información:

Si $x_{3,1} = 1$. Quiere decir que se cortó el eslabón 3 de la cadena uno, y se soldó al eslabón 1 de la cadena 2

Solamente para el caso de $j = 4$ se supone que se conectará a la primera cadena

Función objetivo

El objetivo es minimizar la cantidad de operaciones de Corte-Soldado necesarias para hacer el brazalete:

Minimizar

$$C_B = 5 \sum_{j=1}^4 \sum_{i=1}^3 x_{i,j}$$

Restricciones

Para poder formar el brazalete, es necesario cortar el eslabón final de cada una de las cadenas, por lo que la siguiente restricción debe de cumplirse:

$$x_{3,1} + x_{3,2} + x_{3,3} + x_{3,4} = 4$$

Así como también, cada cadena debe de tener por lo menos 1 corte:

$$x_{1,1} + x_{2,1} + x_{3,1} \geq 1$$

$$x_{1,2} + x_{2,2} + x_{3,2} \geq 1$$

$$x_{1,3} + x_{2,3} + x_{3,3} \geq 1$$

$$x_{1,4} + x_{2,4} + x_{3,4} \geq 1$$

Por lo que el modelo matemático final queda de la forma:

Minimizar

$$C_B = 5 \sum_{j=1}^4 \sum_{i=1}^3 x_{i,j}$$

$$x_{3,1} + x_{3,2} + x_{3,3} + x_{3,4} = 4$$

$$x_{i,j} \geq 0$$

Al realizar un pequeño script en Python de este modelo se tiene:

```

35 siguiente_cadena = (j + 1) if j < 4 else 1
36 print(f"Se cortó el eslabón {i} de la cadena {j} y se soldó al eslabón 1 de la cadena {siguiente_cadena}.")
37

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SQL CONSOLE

Matriz de resultado (x_(i,j)):

```

[0.0, 0.0, 0.0, 0.0]
[0.0, 0.0, 0.0, 0.0]
[1.0, 1.0, 1.0, 1.0]

```

Interpretación de los cortes y soldaduras:

```

Se cortó el eslabón 3 de la cadena 1 y se soldó al eslabón 1 de la cadena 2.
Se cortó el eslabón 3 de la cadena 2 y se soldó al eslabón 1 de la cadena 3.
Se cortó el eslabón 3 de la cadena 3 y se soldó al eslabón 1 de la cadena 4.
Se cortó el eslabón 3 de la cadena 4 y se soldó al eslabón 1 de la cadena 1.

```

PS E:\Documentos\UANL\6\Investigacion de Operaciones\AMPL\Py>

9. Los cuadros de una tabla rectangular de 11 filas y 9 columnas están numerados en secuencia del 1 al 99 con una recompensa monetaria oculta de entre 0 y 20 dólares, asignada a cada cuadro. El juego consiste en que un jugador elige un cuadrado seleccionando cualquier número de dos dígitos y luego restando al número seleccionado la suma de sus dos dígitos. El jugador recibe entonces la recompensa asignada al cuadro seleccionado. Sin importar cuántas veces se repita el juego, ¿qué valores monetarios deben asignarse a los 99 cuadros para minimizar la recompensa de los jugadores? Para hacer el juego interesante, asignar \$0 a todos los cuadros no es una opción.

Análisis del problema

Podemos ver este problema como un gran tablero de “rascadito” con 99 posiciones de premios de entre 0 y 20 dólares.

0	1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16	17
18	19	20	21	22	23	24	25	26
27	28	29	30	31	32	33	34	35
36	37	38	39	40	41	42	43	44
45	46	47	48	49	50	51	52	53
54	55	56	57	58	59	60	61	62
63	64	65	66	67	68	69	70	71
72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89
90	91	92	93	94	95	96	97	98

La acción de elegir un número n y posteriormente restarle sus dígitos se puede ver de la forma:

$$P = n - (\text{suma de los dígitos de } n)$$

Donde P es la posición de la recompensa a rascar

Patrón recurrente

Sea A un número de $n - 1$ cifras el cual se puede representar como una suma de potencias base 10:

$$A = a_n 10^n + a_{n-1} 10^{n-1} + a_{n-2} 10^{n-2} \dots + a_1 10^1 + a_0 10^0$$

Donde a_n es la cifra que se encuentra en la posición n

Por ejemplo, si $A = 85$

$$A = (8)10^1 + (5)10^0$$

La suma de las cifras de A es:

$$a = a_n + a_{n-1} + a_{n-2} \dots + a_1 + a_0$$

Si restamos realizamos la resta del número menos sus cifras se tiene:

$$A - a = (a_n 10^n + a_{n-1} 10^{n-1} \dots + a_1 10^1 + a_0 10^0) - (a_n + a_{n-1} + a_{n-2} \dots + a_1 + a_0)$$

Al agrupar y factorizar términos:

$$A - a = a_n(10^n - 1) + a_{n-1}(10^{n-1} - 1) + \dots + a_1(10^1 - 1) + a_0(10^0 - 1)$$

Si ponemos atención al termino $(10^n - 1)$ y empezamos a darle valores a n mayores a 0 tenemos:

$$(10^3 - 1) = 999$$

$$(10^2 - 1) = 99$$

$$(10^1 - 1) = 9$$

$$(10^0 - 1) = 0$$

Por lo que este termino nos asegura que cada término de la expresión de $A - a$ es múltiplo de 9.

Con esto establecido, podemos establecer que los recuadros que nos interesan son solamente los que tienen las posiciones:

9,18,27,36,45,54,63,72,81,90

Los demás no nos interesan, ya que jamás saldrán al realizar el proceso de selección.

Parámetros

La información que nos da el problema es el monto mínimo y máximo que pueden tener las recompensas

El cual es 0 y 20 respectivamente

Así como también las reglas del juego

Y nos dan la única restricción que no se pueden poner 0s en todo el tablero

Variables de decisión

Solo centraremos en la recompensa que se tiene que establecer en las casillas múltiplos de 9

R_1 : Recompensa para el primer multiplo de 9

R_2 : Recompensa para el segundo multiplo de 9

R_3 : Recompensa para el tercer multiplo de 9

R_4 : Recompensa para el cuarto multiplo de 9

R_5 : Recompensa para el quinto multiplo de 9

R_6 : Recompensa para el sexto multiplo de 9

R_7 : Recompensa para el septimo multiplo de 9

R_8 : Recompensa para el octavo multiplo de 9

R_9 : Recompensa para el noveno multiplo de 9

R_{10} : Recompensa para el decimo multiplo de 9

Función Objetivo

El objetivo es minimizar las ganancias que pueden obtener los participantes

Minimizar

$$\sum_{i=1}^{10} R_i$$

Restricciones

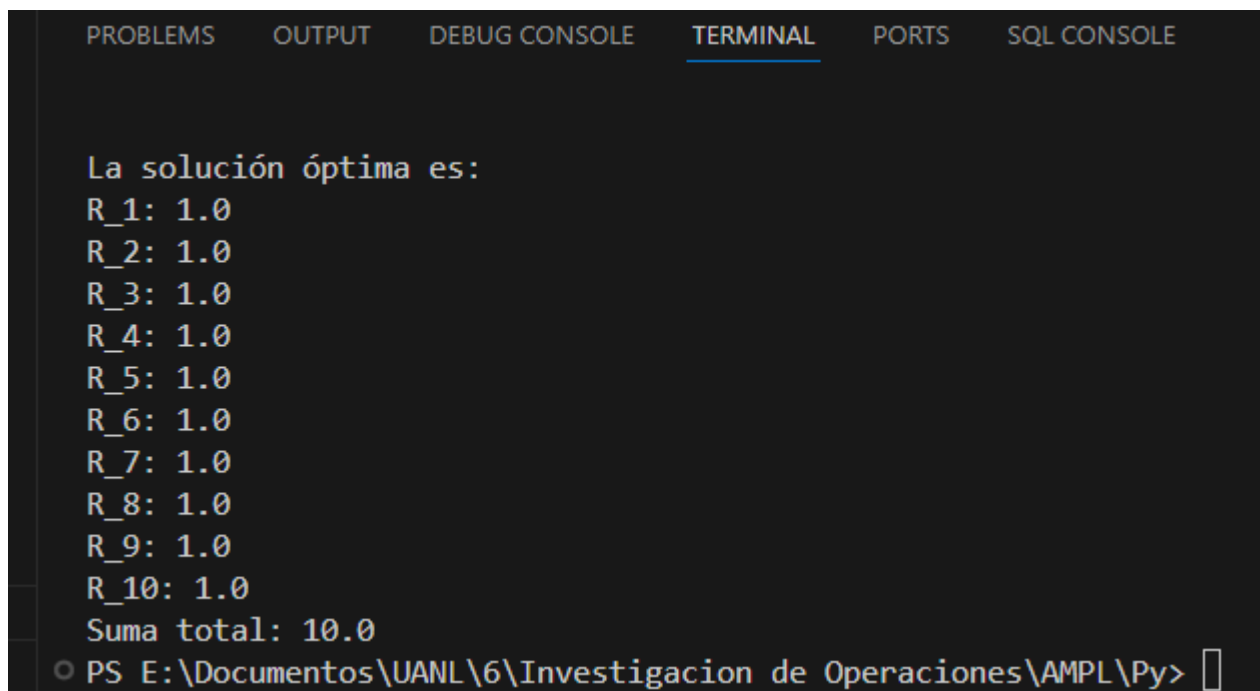
No se pueden llenar todos los recuadros con 0s, en este caso, se va a suponer que ninguno de los múltiplos de 9 se llenará con 0s, ya que no será algo atractivo para el participante

$$R_i > 0$$

Valores permitidos para la recompensa:

$$0 < R_i \leq 20$$

Al modelarlo por medio de Python, nos da el resultado esperado, al hacer todas las variables de decisión 1:



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  SQL CONSOLE

La solución óptima es:
R_1: 1.0
R_2: 1.0
R_3: 1.0
R_4: 1.0
R_5: 1.0
R_6: 1.0
R_7: 1.0
R_8: 1.0
R_9: 1.0
R_10: 1.0
Suma total: 10.0
PS E:\Documentos\UANL\6\Investigacion de Operaciones\AMPL\Py>
```

Las demás posiciones del tablero las llenaremos de valores al azar, no afectaran en nuestra función objetivo, y harán atractivo el tablero.