Proyecto 1: Programación Restringida

Mauricio Trejo Andrés Ángeles

Octubre 2019

1 Planteamiento del problema

El dueño de una fabrica de dulces ha detectado caramelo sobrante durante el proceso de fabricación, existen diferentes tipos de caramelo en la cantidad de sobrante. El objetivo del empresario es obtener más ganancias al fabricar un tipo de caramelo de calidad de exportación con el sobrante obtenido y asi garantizar que su producto llegue al mercado.

Si arma paquetes que pesen a lo más 3kg no pagara ningún tipo de arancel, el empaque tiene una capacidad de 1 dm^3.

Para que el negocio sea redondo, la producción del paquete debe de costar a lo más $200 y además garantizar que el caramelo sea lo más dulce posible. El objetivo es indicar el número óptimo de cada tipo de caramelo en cada paquete a partir de la siguiente información:

Caramelo	Peso (gramos)	Dimensiones (cm)	Dulzura	Valor ($)
A	100	8 x 2.5 x 0.5	20	8
B	45	7 x 2 x 0.5	16	6.8
C	10	3 x 2 x 0.5	9	4
D	25	3 x 3 x 0.5	7	3

2 Modelado y solución del problema mediante el uso de Programación de restricciones

Definimos x_i como el número de caramelos del tipo i que utilizamos en cada empaque donde $i = 1, 2, 3, 4$.

El objetivo es que el caramelo del empaque sea lo más dulce posible, por lo que la función objetivo queda dada por

$$z = f(\bar{x}) = 20x_1 + 16x_2 + 9x_3 + 7x_4 \tag{1}$$

Además, sabemos que cada decímetro cúbico equivale a mil centímetros cúbicos, por lo que la restricción del volumen del empaque es

$$(8\times2.5\times0.5)x_1+(7\times2\times0.5)x_2+(3\times2\times0.5)x_3+(3\times3\times0.5)x_4 = 10x_1+7x_2+3x_3+3x_4 \leq 1000 \tag{2}$$

y el modelo del problema es

$$
\begin{aligned}
\text{maximizar} \quad & z = 20x_1 + 16x_2 + 9x_3 + 7x_4 \\
\text{sujeto a} \quad & 8x_1 + 6.8x_2 + 4x_3 + 3x_4 \leq 200 \\
& 10x_1 + 7x_2 + 3x_3 + 3x_4 \leq 1000 \\
& 100x_1 + 45x_2 + 10x_3 + 25x_4 \leq 3000 \\
& x_1, x_2, x_3, x_4 \in \mathbb{Z}^+ \cup \{0\}
\end{aligned} \tag{3}
$$

2.1 Búsqueda de los dominios restringidos de cada variable

Consideremos las restricciones del modelo:

$$8x_1 + 6.8x_2 + 4x_3 + 3x_4 \leq 200 \tag{4}$$
$$10x_1 + 7x_2 + 3x_3 + 3x_4 \leq 1000 \tag{5}$$
$$100x_1 + 45x_2 + 10x_3 + 25x_4 \leq 3000 \tag{6}$$
$$x_1, x_2, x_3, x_4 \in \mathbb{Z}^+ \cup \{0\} \tag{7}$$

Si igualamos x_2, x_3 y x_4 a cero, tenemos que $8x_1 \leq 200$, $10x_1 \leq 1000$ y $100x_1 \leq 3000$, por lo que concluimos que

$$x_1 \in \{0, 1, 2, \ldots, 25\} \tag{8}$$

De la misma forma, deducimos que

$$x_2 \in \{0, 1, 2, \ldots, 29\} \tag{9}$$
$$x_3 \in \{0, 1, 2, \ldots, 50\} \tag{10}$$
$$x_4 \in \{0, 1, 2, \ldots, 66\} \tag{11}$$

y planteamos la primera versión de nuestro problema restringido:

[21]:
```
from constraint import *
import time

# Creando el problema y el rango de las variables:
problem = Problem()
problem.addVariable("x1", range(26))
problem.addVariable("x2", range(30))
problem.addVariable("x3", range(51))
problem.addVariable("x4", range(67))

# Creando las restricciones
problem.addConstraint(lambda x1, x2, x3, x4: 8*x1 + 6.8*x2 + 4*x3 + 3*x4 <=␣
 ↪200, ("x1", "x2", "x3", "x4"))
problem.addConstraint(lambda x1, x2, x3, x4: 100*x1 + 45*x2 + 10*x3 + 25*x4 <=␣
 ↪3000, ("x1", "x2", "x3", "x4"))
problem.addConstraint(lambda x1, x2, x3, x4: 10*x1 + 7*x2 + 3*x3 + 4.5*x4 <=␣
 ↪1000, ("x1", "x2", "x3", "x4"))

# Obteniendo las soluciones factibles
solutions = problem.getSolutions()
```

```
# Creando la función objetivo
def fncn_obj(x1, x2, x3, x4):
    return 20*x1 + 16*x2 + 9*x3 + 7*x4

auxProblem = problem

len(solutions)
```

[21]: 126423

Existen un total de 126,423 soluciones factibles para este modelo

2.2 Acotamiento de la función objetivo para encontrar la solución óptima

De acuerdo con Frederick Hillier, la especialidad de la programación restringida o la programación de restricciones son los problemas con muchas restricciones y sin una función objetivo. Sin embargo, también concede que la presencia de una función objetivo no es impedimento y, en muchos casos, ayuda para restringir el conjunto factible del modelo aún más (Hillier, 478).

Antes vimos que las variables x_1, x_2, x_3 y x_4 son tales que

$$x_1 \in \{0, 1, 2, \ldots, 25\} \tag{12}$$
$$x_2 \in \{0, 1, 2, \ldots, 29\} \tag{13}$$
$$x_3 \in \{0, 1, 2, \ldots, 50\} \tag{14}$$
$$x_4 \in \{0, 1, 2, \ldots, 66\} \tag{15}$$

En particular, notemos que $\bar{x}_1 = (0, 0, 50, 0)$ y $\bar{x}_2 = (0, 29, 0, 0)$ son soluciones factibles y que

$$z_1 = f(0, 0, 50, 0) = 450 < 464 = f(0, 29, 0, 0) = z_2 \tag{16}$$

lo cual es un enorme avance porque no solo podemos aseverar que el valor óptimo de z es positivo, sino que debe satisfacer la restricción

$$z \geq 450 \tag{17}$$

la cual nos permite descartar más de 100,000 soluciones factibles adicionales, tal y como mostramos a continuación:

[22]:
```
auxProblem.addConstraint(lambda x1, x2, x3, x4: 20*x1 + 16*x2 + 9*x3 + 7*x4 >=␣
↪450, ("x1", "x2", "x3", "x4"))
solutions = auxProblem.getSolutions()
len(solutions)
```

[22]: 20949

En realidad no hay razón para haberse conformado con probar un valor al azar de alguna componente de $\bar{x}$. Podemos ver que el mayor coeficiente en la función objetivo corresponde a x_1, por lo que definimos la restricción

$$z \geq f(25, 0, 0, 0) = 500 \tag{18}$$

y tenemos que el conjunto factibles se ha reducido tanto que únicamente nos queda una única solución

[18]:
```
problem.addConstraint(lambda x1, x2, x3, x4: 20*x1 + 16*x2 + 9*x3 + 7*x4 >=␣
↪500, ("x1", "x2", "x3", "x4"))
solutions = problem.getSolutions()
len(solutions)
```

[18]:
```
1
```

y que, por supuesto, podemos corroborar es la solución óptima del modelo que planteamos originalmente:

[19]:
```
print("El valor máximo es: {} y la solución es {}".
↪format(z,(solucion['x1'],solucion['x2'],solucion['x3'],solucion['x4'])))
```

```
El valor máximo es: 500 y la solución es (25, 0, 0, 0)
```

[28]:
```
# Creando el problema y el rango de las variables:
original = Problem()
original.addVariables(["x1", "x2", "x3", "x4"], [x1 >= 0, x2 >= 0, x3 >= 0, x4␣
↪>= 0])
#problem.addVariables()
# Creando las restricciones
problem.addConstraint(lambda x1, x2, x3, x4: 8*x1 + 6.8*x2 + 4*x3 + 3*x4 <=␣
↪200, ("x1", "x2", "x3", "x4"))
problem.addConstraint(lambda x1, x2, x3, x4: 100*x1 + 45*x2 + 10*x3 + 25*x4 <=␣
↪3000, ("x1", "x2", "x3", "x4"))
problem.addConstraint(lambda x1, x2, x3, x4: 10*x1 + 7*x2 + 3*x3 + 4.5*x4 <=␣
↪1000, ("x1", "x2", "x3", "x4"))
problem.addConstraint(lambda x1, x2, x3, x4: 20*x1 + 16*x2 + 9*x3 + 7*x4 >= 0,␣
↪("x1", "x2", "x3", "x4"))

# Obteniendo las soluciones factibles
solutions = original.getSolution()

# Creando la función objetivo
def fncn_obj(x1, x2, x3, x4):
    return 20*x1 + 16*x2 + 9*x3 + 7*x4

print("El valor máximo del modelo de programación lineal entera es: {} y la␣
↪solución es {}".
↪format(z,(solucion['x1'],solucion['x2'],solucion['x3'],solucion['x4'])))
```

```
El valor máximo del modelo de programación lineal entera es: 500 y la solución
es (25, 0, 0, 0)
```

3 Bibliografía

1. Hillier, Frederick S., and Gerald J. Lieberman. *Introducción a La investigación De Operaciones*. 9th ed. México D.F.: McGraw-Hill, 2010.