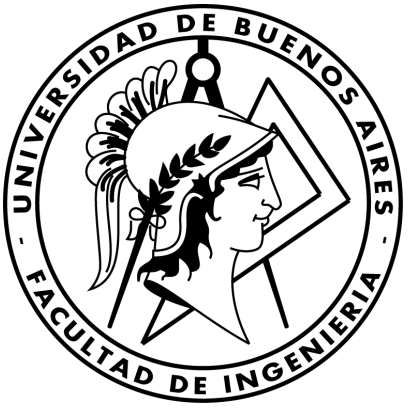
Debuggeando modelos incompatibles en GLPK



Modelos y Optimización 1

Redactado por Iván Terzano
Facultad de Ingeniería de la Universidad de Buenos Aires

Versión 1.0 - Mayo 2019

Índice

Repaso: Modelo Incompatible

Un modelo es incompatible cuando 1 o más restricciones no pueden ser satisfechas. Es decir, el problema no tiene solución factible.

Un modelo puede ser incompatible principalmente por dos razones:

1. Un error de modelado
2. La naturaleza del problema (ejemplo, la demanda no se puede satisfacer con los recursos que tenemos)

Encontrando el problema

Sea cual sea la fuente, es bueno entender por qué es incompatible el modelo.
En otros softwares de programación lineal existe la posibilidad de pedirle al mismo cuáles son las restricciones que están en conflicto. GLPK no permite esto, pero aún así podemos *debuggear* el modelo.

Lo que podemos hacer es agregar unas variables parecidas a las slack[1] a nuestras restricciones (para todas, para un subconjunto que consideremos más "peligrosas", o de alguna otra manera) agregándoles un costo muy alto en el funcional. De esta manera, el modelo va a tener solución, pero con alguna de estas variables tomando valor en el óptimo[2].

En concreto, si la restricción es de ≤. agregamos la variable slack restando. En cambio si la restricción es de ≥, agregamos la variable sumando. En ambos casos, en el funcional debemos darle un costo muy alto (sea lo que sea que eso signifique en nuestro funcional). Si la restricción es de =, debemos agregar dos variables slack, una sumando y la otra restando[3]. Nuevamente debemos darle un peso en el funcional que perjudique al mismo ampliamente.

De esta forma, vamos a haber relajado las restricciones de nuestro modelo. Como en el funcional les pusimos un costo muy alto[4], si alguna de estas variables toma valor, la restricción que la contiene es culpable de que el problema sea incompatible.

[1] No confundir con las variables slack. Son parecidas pero funcionan distinto.
[2] Algo parecido a lo que hacemos para resolver problemas con restricciones de mayor o igual en Simplex
[3] Ejercicio: Pensar por qué tenemos que agregar 2 variables en este caso.
[4] Ojo con los valores, un número extremadamente alto puede traer inestabilidades

Ejemplo aplicado a los ejercicios 2.1 y 2.5

2.1

```
#Problema 2.1

#Variables
var A >= 0; #Cantidad de pulóveres A que se fabrican
var B1 >= 0; #Cantidad de pulóveres B que se fabrican en la máquina 1
var B2 >= 0; #Cantidad de pulóveres B que se fabrican en la máquina 2
var C >= 0; #Cantidad de pulóveres C que se fabrican

#Funcional:
maximize z: 10*A + 15*(B1 + B2) + 18*C;

#Restricciones:
#Máquina 1
s.t. maq1: 5*A + 6*B1 <= 80;
#Máquina 2
s.t. maq2: 4*C + 4*B2 <= 80;
#Lana Mejorada
s.t. lanaMej: 1.6*A + 1.2*C <= 20;
#Lana Normal
s.t. lanaNorm: 1.8*B1 + 1.8*B2 <= 36;
#Demanda miníma
s.t. demMin: B1 + B2 >= 10;

```

```
#Problema 2.1 Adaptado

#Variables
var A >= 0; #Cantidad de pulóveres A que se fabrican
var B1 >= 0; #Cantidad de pulóveres B que se fabrican en la máquina 1
var B2 >= 0; #Cantidad de pulóveres B que se fabrican en la máquina 2
var C >= 0; #Cantidad de pulóveres C que se fabrican
var RMaq1 >= 0; # Variable para relajar la restricción maq1
var RMaq2 >= 0; # Variable para relajar la restricción maq2
var RLanaMej >= 0; # Variable para relajar la restricción lanamej
var RLanaNorm >= 0; # Variable para relajar la restricción lananorm
var RDemMin >= 0; # Variable para relajar la restricción demmin

#Funcional:
maximize z: 10*A + 15*(B1 + B2) + 18*C - 2000*(RMaq1 + RMaq2 + RLanaMej + RLanaNorm + RDemMin);

#Restricciones:
#Máquina 1
s.t. maq1: 5*A + 6*B1 - RMaq1 <= 80;
#Máquina 2
s.t. maq2: 4*C + 4*B2 - RMaq2 <= 80;
#Lana Mejorada
s.t. lanaMej: 1.6*A + 1.2*C - RLanaMej <= 20;
#Lana Normal
s.t. lanaNorm: 1.8*B1 + 1.8*B2 - RLanaNorm <= 36;
#Demanda miníma
s.t. demMin: B1 + B2 + RDemMin >= 10;

```

2.5

```
# Problema 2.5

# Variables
var ERSn >= 0; # Estampadoras rápidas usadas para tela Snoopy
var ERSc >= 0; # Estampadoras rápidas usadas para tela Scooby
var ELSn >= 0; # Estampadoras lentas usadas para tela Snoopy
var ELSc >= 0; # Estampadoras lentas usadas para tela Scooby

# Funcional
# Cómo la demanda es fija, no tiene sentido evaluar un funcional (aunque fabrique más,
# cosa que permitimos para hacer el modelo menos restrictivo, no lo va a vender)
maximize z: 1

# Restricciones
# Snoopy
s.t. snoopy: (10*ERSn + 2*ELSn)*8 >= 10000
# Scooby
s.t. scooby: (7*ERSc + 1*ELSc)*8 >= 9000
# Limite de estampadoras
s.t. rapidas: ERSn + ERSc <= 70
s.t. lentas: ELSn + ELSc <= 60

```

En este caso relajé las restricciones agregando la variable del otro lado de la restricción para que sea más claro su significado.

```
# Problema 2.5

# Variables
var ERSn >= 0; # Estampadoras rápidas usadas para tela Snoopy
var ERSc >= 0; # Estampadoras rápidas usadas para tela Scooby
var ELSn >= 0; # Estampadoras lentas usadas para tela Snoopy
var ELSc >= 0; # Estampadoras lentas usadas para tela Scooby
# "Slacks" par relajar
var Rsnoopy >= 0; # Variable para relajar la restricción: snoopy
var Rscooby >= 0; # Variable para relajar la restricción: scooby
var Rrapidas >= 0; # Variable para relajar la restricción: rapidas
var Rlentas >= 0; # Variable para relajar la restricción: lentas

# Funcional
maximize z: -1000 * (Rsnoopy + Rscooby + Rrapidas + Rlentas)

# Restricciones
# Snoopy
s.t. snoopy: (10*ERSn + 2*ELSn)*8 >= 10000 - Rsnoopy
# Scooby
s.t. scooby: (7*ERSc + 1*ELSc)*8 >= 9000 - Rscooby
# Limite de estampadoras
s.t. rapidas: ERSn + ERSc <= 70 + Rrapidas
s.t. lentas: ELSn + ELSc <= 60 + Rlentas

```

Casos sin resolver

Si bien este método puede ser útil, existen algunos casos en los cuales no se puede aplicar este método. Por ejemplo, si equivocadamente definimos una variable como Bivalente cuando debería ser Entera.

Bibliografía

1. http://lpsolve.sourceforge.net/5.5/Infeasible.htm