Sayfa 364 Soru 8 Çözüm

Soru: Bir müzik şirketi, yıldızı yeni parlayan bir şarkıcıyla sekiz şarkılık bir anlaşma yapmıştır. Şarkılar sırasıyla, 8, 3, 5, 5, 9, 6, 7 ve 12 dakikalık farklı sürelere sahiptir. Şirket bu şarkıları bir kasetin iki yüzüne kaydet- meyi planlamaktadır. Kasetin her yüzü 30 dakika olup, şirket şarkıları her iki yüze mümkün olduğunca eşit dağıtmak arzusundadır. Problemi TDP olarak formüle edin ve optimum çözümü bulun.

Çözüm:Şirket şarkıları her iki yüze eşit dağıtmak istemektedir.Kasetin yüzlerine A VE B Diyelim.

|A yüzündeki toplam süre -B yüzündeki toplam süre| =y1

|B yüzündeki süre -A yüzündeki süre|=y2 olsun

Min: y1+y2 yi arıyoruz.

Şarkı süreleri üstteki gibi olduğu için şarkının A yüzünde olmasına 1 B yüzünde olmasına 0 değerini verirsek kısıtlamaları aşağıdaki gibi yazabiliriz.

Kısıtlar:

(x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 == 4)

(8*x1 + 3*x2 + 5*x3 + 5*x4 + 9*x5 + 6*x6 + 7*x7 + 12*x8 - y1 <= 30)

(x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 <= 4)

(-x1 - x2 - x3 - x4 - x5 - x6 - x7 - x8 + 4 - y2 <= 4)

(-8*x1 - 3*x2 - 5*x3 - 5*x4 - 9*x5 - 6*x6 - 7*x7 - 12*x8 <= 30)

(-x1 - x2 - x3 - x4 - x5 - x6 - x7 - x8 <= 4)

X1,x2,…x8= binary (0|1)

Dallanma ve Sınırlandırma Algoritması

Dallanma ve sınırlandırma (branch and bounce) algoritmasını Jupiter Notebook içinde Python kodumuza gurobipy as gp kütüphanesini import ederek çözüme ulaşabiliriz.Kodu aşağıdaki şekilde yazdığımızda ,1 çıktısını verenler A yüzünde 0 çıktısını verenler B yüzünde olacaktır.

```
import gurobipy as gp

# Create a Gurobi model
m = gp.Model()

# Add variables to the model
x1 = m.addVar(vtype=gp.GRB.BINARY)
x2 = m.addVar(vtype=gp.GRB.BINARY)
x3 = m.addVar(vtype=gp.GRB.BINARY)
x4 = m.addVar(vtype=gp.GRB.BINARY)
x5 = m.addVar(vtype=gp.GRB.BINARY)
x6 = m.addVar(vtype=gp.GRB.BINARY)
x7 = m.addVar(vtype=gp.GRB.BINARY)
x8 = m.addVar(vtype=gp.GRB.BINARY)
y1 = m.addVar(vtype=gp.GRB.CONTINUOUS, lb=0)
y2 = m.addVar(vtype=gp.GRB.CONTINUOUS, lb=0)

# Set up constraints
m.addConstr(x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 == 4)
m.addConstr(8*x1 + 3*x2 + 5*x3 + 5*x4 + 9*x5 + 6*x6 + 7*x7 + 12*x8 - y1 <= 30)
m.addConstr(x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 <= 4)
m.addConstr(-x1 - x2 - x3 - x4 - x5 - x6 - x7 - x8 + 4 - y2 <= 4)
m.addConstr(-8*x1 - 3*x2 - 5*x3 - 5*x4 - 9*x5 - 6*x6 - 7*x7 - 12*x8 <= 30)
m.addConstr(-x1 - x2 - x3 - x4 - x5 - x6 - x7 - x8 <= 4)

# Set up objective function
m.setObjective(y1 + y2, sense=gp.GRB.MINIMIZE)

# Optimize the model
m.optimize()

# Print the optimal solution
print("Optimal solution:")
print("x1 =", x1.x)
print("x2 =", x2.x)
print("x3 =", x3.x)
print("x4 =", x4.x)
print("x5 =", x5.x)
print("x6 =", x6.x)
print("x7 =", x7.x)
print("x8 =", x8.x)
print("y1 =", y1.x)
print("y2 =", y2.x)
```

```
Restricted license - for non-production use only - expires 2024-10-28
Gurobi Optimizer version 10.0.0 build v10.0.0rc2 (win64)

CPU model: Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz, instruction set [SSE2|AVX|AVX2
Thread count: 4 physical cores, 8 logical processors, using up to 8 threads

Optimize a model with 6 rows, 10 columns and 50 nonzeros
Model fingerprint: 0xb9811f8e
Variable types: 2 continuous, 8 integer (8 binary)
Coefficient statistics:
  Matrix range     [1e+00, 1e+01]
  Objective range  [1e+00, 1e+00]
  Bounds range     [1e+00, 1e+00]
  RHS range        [4e+00, 3e+01]
Found heuristic solution: objective 0.0000000

Explored 0 nodes (0 simplex iterations) in 0.01 seconds (0.00 work units)
Thread count was 1 (of 8 available processors)

Solution count 1: 0

Optimal solution found (tolerance 1.00e-04)
Best objective 0.000000000000e+00, best bound 0.000000000000e+00, gap 0.0000%
Optimal solution:
x1 = 1.0
x2 = 1.0
x3 = 1.0
x4 = 1.0
x5 = -0.0
x6 = -0.0
x7 = -0.0
x8 = -0.0
y1 = 0.0
y2 = 0.0
```

1,2,3,4 .şarkılar 1.yüzde 5,6,7,8 inci şarkılar 2.yüzde olmalıdır