

Mathematical Programming: Modeling and Applications

Giacomo Nannicini

`giacomon@lix.polytechnique.fr`

The Inheritance Problem

- A rich aristocrat passes away, leaving a large legacy which must be shared between his two sons
- We assume that the value for all items in the legacy is known
- The legacy must be split between the two sons (i.e. we must find a bipartition) minimizing the difference between the value of the two parts
- This problem is also known as the Subset-Sum problem

The Inheritance Problem

- Given a set A with n elements, we must find a bipartition: A_1, A_2 such that $A_1 \cap A_2 = \emptyset$, $A_1 \cup A_2 = A$
- Suppose each item $a \in A$ has a value $v(a)$
- Objective: $\min \left| \sum_{a \in A_1} v(a) - \sum_{a \in A_2} v(a) \right|$
- Determine the decision variables, write the objective function and the constraints
- In AMPL, the absolute value is the function $abs()$
- Write the model and the data file

Run File

- Mixed Integer NonLinear Problem solver: boncouenne
- Display the assignment of each item to one of two sons
- Display the total value of the legacy received by each one of the two sons
- Note: you can define and use parameters in the run file
`param value1;`
`let value1 := sum {i in 1..n} ...`

Data

● Object values:

1. 25000
2. 5000
3. 20000
4. 40000
5. 12000
6. 12000
7. 12000
8. 3000
9. 6000
10. 10000
11. 15000
12. 10000
13. 13000

Solution: Model

```
# model file for the inheritance problem
```

```
param n;
```

```
set Objects := 1..n;
```

```
param v{Objects};
```

```
var x{Objects} binary;
```

```
var y{Objects} binary;
```

```
minimize cost:
```

```
abs(sum{i in Objects}(v[i]*x[i]) - sum{i in Objects}(v[i]*y[i]));
```

```
subject to split {i in Objects}: x[i] + y[i] = 1;
```

Solution: Data

```
# data file for the inheritance problem
```

```
param n := 13;
```

```
param: v :=
```

```
1 25000
```

```
2 5000
```

```
3 20000
```

```
4 40000
```

```
5 12000
```

```
6 12000
```

```
7 12000
```

```
8 3000
```

```
9 6000
```

```
10 10000
```

```
11 15000
```

```
12 10000
```

```
13 13000;
```

Solution: Run File

```
# run file for inheritance problem

model inherit.mod;
data inherit.dat;
option solver boncouenne;
solve;
display x,y;
param value1;
param value2;
let value1 := sum {i in Objects} v[i]*x[i];
let value2 := sum {i in Objects} v[i]*y[i];
display value1, value2;
```

Solution: Output

```
:      x      y      :=  
1      0      1  
2      1      0  
3      1      0  
4      1      0  
5      1      0  
6      0      1  
7      0      1  
8      0      1  
9      0      1  
10     0      1  
11     1      0  
12     0      1  
13     0      1  
;
```

```
value1 = 92000
```

```
value2 = 91000
```

A Harder Instance

- Try with this data:

```
# data file for the inheritance problem
```

```
param n := 25;
```

```
param: v :=
```

```
1 25000      13 13000
2 5000       14 9000
3 20000      15 7000
4 40000      16 14000
5 12000      17 21000
6 12000      18 13000
7 12000      19 4000
8 3000       20 6000
9 6000       21 18000
10 10000     22 59000
11 15000     23 25000
12 10000     24 19000
              25 7000;
```

Improving The Formulation

- Solving MINLPs is hard!
- However, there is a linear formulation for this problem
- Let $V = \sum_{a \in A} v(a)$
- Obviously, both sons cannot receive more than $V/2$
- Objective: one son should receive as close to $V/2$ as possible (the other one gets the remaining part of the legacy)

Improving The Formulation

- Write a model file
- Write a run file (use the same data as before), displaying the item assignment and the value of the legacy received by each son

- Useful commands:

```
for {i in 1..n} {...}  
if (condition) then {...} else {...}  
printf "%d \n", i;
```

Solution: Model

```
# model file for the inheritance problem (better formulation)

param n;
set Objects := 1..n;
param v {Objects};
param total := sum {i in Objects} v[i];
var x {Objects} binary;
minimize cost: sum {i in Objects} v[i] * x[i];
subject to limit: sum {i in Objects} v [i]* x[i] >= 0.5 * total;
```

Solution: Run File

```
# run file for the inheritance problem (better formulation)
model inheritance.mod;
data inheritance.dat;
option solver cplex;
solve;
printf " 1 2\n";
for {i in Objects}{
    if (x[i] = 1) then {
        printf "%d 1 0\n", i;
    }
    else {
        printf "%d 0 1\n", i;
    }
}
param value1;
param value2;
let value1 := sum {i in Objects} v[i]*x[i];
let value2 := sum {i in Objects : x[i] = 0} v[i];
display value1, value2;
```

Solution: Output

```
1 2
1 1 0      19 0 1
2 1 0      20 0 1
3 0 1      21 0 1
4 1 0      22 1 0
5 1 0      23 0 1
6 0 1      24 1 0
7 1 0      25 0 1
8 0 1      value1 = 193000
9 1 0      value2 = 192000
10 0 1
11 1 0
12 0 1
13 0 1
14 0 1
15 0 1
16 0 1
17 0 1
18 0 1
```

The Production Line Problem



- Production of items on different machines, which must be used in order (production line)
- Each item requires a different time on each machine
- We want to minimize the completion time for the whole production (i.e. the time at which the last item is completed)

The Production Line Problem

- Five items, three machines, item i requires time s_{ij} on machine j , where s_{ij} is given by the matrix:

$$\begin{pmatrix} 3 & 1 & 1 \\ 2 & 1.5 & 1 \\ 3 & 1.2 & 1.3 \\ 2 & 2 & 2 \\ 2.1 & 2 & 3 \end{pmatrix}$$

- Define model, data and run file to display the production order

Solution: Model

- Indices: i, j on the set L of items, k on the set V of machines
- Parameters: s_{ik} is the time required by item i on machine k , M is an upper bound on the total completion time
- Variables: $t_{ik} \geq 0$ is the starting time of item i on machine k , $T \geq 0$ is the completion time for the last item, $y_{ijk} = 1$ if item i uses machine k before item j , 0 otherwise
- Objective: $\min T$

Solution: Model

- Constraints:
 - Sequential: $\forall i \in L, k \in V \setminus \{v\} (t_{ik} + s_{ik} \leq t_{i(k+1)})$
 - Last machine: $\forall i \in L (t_{ik} + s_{ik} \leq T)$
 - Non overlapping:
 $\forall i, j \in L, k \in V, i \neq j (t_{ik} + s_{ik} \leq t_{jk} + M(1 - y_{ijk}))$
 - Disjunction: $\forall i, j \in L, k \in V, i \neq j (y_{ijk} + y_{jik} = 1)$
- Mixed Integer Linear Problem, can be solved with Cplex

Solution: Model

```
# production.mod
param l >= 1;
param v >= 1;
set L := 1..l;
set V := 1..v;
set V0 := 1..v-1;
param s{L,V} >= 0;
param M default sum{i in L, k in V} s[i,k] ;
var t{L,V} >= 0;
var T >= 0;
var y{L,L,V} binary;

minimize makespan : T;

subject to sequential{i in L, k in V0} : t[i,k] + s[i,k] <= t[i,k+1];
subject to lastmachine{i in L} : t[i,v] + s[i,v] <= T;
subject to nonoverlap{i in L, j in L, k in V : i != j} :
    t[i,k] + s[i,k] <= t[j,k] + M * (1 - y[i,j,k]);
subject to disjunction{i in L, j in L, k in V : i != j} :
    y[i,j,k] + y[j,i,k] = 1;
```

Solution: Data

```
# production.dat

param l := 5;
param v := 3;
param s : 1 2 3 :=
1 3.0 1.0 1.0
2 2.0 1.5 1.0
3 3.0 1.2 1.3
4 2.0 2.0 2.0
5 2.1 2.0 3.0 ;
```

Solution: Run File

```
# production.run
model production.mod;
data production.dat;
option solver cplex;
solve;
display makespan;
for {i in L} {
    printf "batch %d : ", i;
    for {k in V} {
        printf "[%f] ", t[i,k];
    }
    printf "\n";
}
```

Solution: Output

```
ILOG AMPL 10.100, licensed to "ecolepolytechnique-palaiseau".  
AMPL Version 20060626 (Linux 2.6.9-5.ELsmp)  
ILOG CPLEX 10.100, licensed to "ecolepolytechnique-palaiseau",  
options: e m b q use=8  
CPLEX 10.1.0: optimal integer solution; objective 14.1  
1175 MIP simplex iterations  
235 branch-and-bound nodes  
makespan = 14.1
```

```
batch 1 : [9.100000] [12.100000] [13.100000]  
batch 2 : [0.000000] [2.000000] [3.500000]  
batch 3 : [6.100000] [10.600000] [11.800000]  
batch 4 : [2.000000] [4.000000] [6.000000]  
batch 5 : [4.000000] [6.100000] [8.100000]
```