Tutorial
Discrete Optimization - Centrale Supélec
Solution

F. Rocaries & H. Talbot

April 18, 2018

Here is a collection of problems in integer and binary programming. The course is online at `https://hugues-talbot.github.io/teaching/2018_Discrete_Optimisation_CS`.

1 A young couple

1.1 formulation

A youg couple, Alice et Benoît, constantly argue with respect to domestic chores (and they don't even have children yet!). In order to solve the problem, as good mathematicians, they try to find the most equitable way to share the chores. After a lot of research, they draw the following tableau:

	Groceries	Cooking	Dishwashing	Laundry
Alice	4.5h	7.8	3.6	2.9
Benoît	4.9h	7.2	4.3	3.1

They decide to each only take care of two tasks, which cannot be shared. In order to both minimize their working hours.

1. Formulate this problem as an IP.
2. Solve the LP relaxation, for example with our crude Simplex solver in Python. Don't forget to initialize the problem.
3. Comment on the solution as much as possible:
   - Number of non-degeenerate basis variables
   - Relationship to the expected size of the basis
   - Nature of the solution
   - Etc.
4. What if they had decided that they should have the most *equitable* workload irrespective of efficiency, how would you formulate and solve the problem ?

1.2 Solution

This is a classical assignment transport problem, however we can formulate it as an IP.

formulate the problem We call x_{ij} a binary variable meaning that person i took on job j, with $i = 1$ being Alice, $i = 2$ being Benoît. Here $j = 1 \dots 4$ are respectively groceries, cooking, dishwashing and laundry.

The w_{ij} are the corresponding weights in the matrix. The cost is $z = \sum_{ij} w_{ij} x_{ij}$. The constrainsts are:

$$\forall i, \sum_j x_{ij} = 2 \text{ (each people take on 2 jobs).}$$

$$\forall j, \sum_i x_{ij} = 1 \text{ (each job must be taken exactly once)}$$

There are 2+4 = 6 constraints and 8 variables. The A matrix of the problem is therefore 6×8.

solve the LP relaxation Here is the code for solving the problem with the hand-written Simplex solver in Python (available from the web page of the course)

```
## Young couple
def youngcouple(debug=False):
    A = np.array([[ 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0],
                  [ 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0],
                  [ 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0],
                  [ 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0],
                  [ 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0],
                  [ 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1]
                 ])
    b = np.array([2, 2, 1, 1, 1, 1])
    C = np.array([4.5, 7.8, 3.6, 2.9, 4.9, 7.2, 4.3, 3.1, 100, 100, 100, 100, 100, 100])
    IBV = np.array([8,9,10,11,12,13])
    sol = simplexit(A,b,C,IBV,debug)
    z= C[sol[0]].dot(sol[1])
    return(z)
```

Note: Variables are ordered in row-major order: x_{11} is variable 0, x_{12} is variable 1, x_{21} is variable 4. The 6 extra variables at the end are for initializing the solver (the students know about this). They have a high cost so that they disappear from the solution.

One can run the code this way:

```
print(youngcouple())
```

And the solution should be:

```
 optimum found!
IBV =  [ 2  5  4 11  6  3]
Bbar=  [1. 1. 1. 0. 0. 1.]
18.6
```

1.3 Comment the solution

The variables with values 1 are 2, 3, 4 and 5. With our formulation they correspond to $x_{13}, x_{14}, x_{21}, x_{22}$. The other two variables in the basis are degenerate and don't count.

This means Alice does the dishwashing and laundry while Benoît does the groceries and cooking. This makes sense given the weights but it is nice to find out we do get the right solution.

The solution is already binary so no need to use an IP solver.

Even though the simplex rank is 6, the solution only has 4 non-xero variables and so the real problem is rank 4. This can be explained because some of the constraints are redundant.

the students could try removing some of the constraints. For example the first and the last one. They need to retain one of the row constraint.

1.4 Equitable problem

The problem is equitable if Alice and Bob work similar number of hours. The objective becomes minimizing the difference that Alice and Benoît work. The cost can be written

$$z = \min . |\sum_j x_{1j} w_{1j} - \sum_j x_{2j} w_{2j}|$$

This is a convex cost but no longer linear due to the absolute value. However we can deal with that with extra variables. We write

$$z = w \tag{1}$$
$$a = \sum_j x_{1j} w_{1j} \tag{2}$$
$$b = \sum_j x_{2j} w_{2j} \tag{3}$$
$$\tag{4}$$

with the constraints

$$a - b \leq w \tag{5}$$
$$b - a \leq w \tag{6}$$

All the inequalities are linear, and so is the cost, so this is a linear program. You might have a question that w is in the objective function and so is not a constant, so obviously the solution is to write the constraints like this:

$$a - b - w \leq 0 \tag{7}$$
$$b - a - w \leq 0 \tag{8}$$

2 Processors

2.1 Formulation

A processor manufacturer wants to reinvent their product line. To help with their product line decision, the following tableau is given:

	Product 1	Product 2	Product 3	Product 4
Starting costs	€50 000	€40 000	€70 000	€60 000
Revenue/unit	€70	€60	€90	€80

Denoting x_i is the production of i, we want to maximize the profit (i.e. revenues - costs), knowing that:

- At most two products can be fabricated.
- Product 3 or 4 require the production of one of product 1 or 2.
- There exist production limits, expressed thus:

$$\begin{array}{rcrcrcrcll} 5x_1 & + & 3x_2 & + & 6x_3 & + & 4x_4 & \leq & 6000 & \text{or} \\ 4x_1 & + & 6x_2 & + & 3x_3 & + & 5x_4 & \leq & 6000 & \end{array}$$

1. Model the problem as a mixed linear-integer program.
2. Solve the problem with a spreadsheet program

2.2 Solution

That problem is fairly classical, it is a "fixed load" problem with non-linear starting costs. This means that making product 1 costs 0 if none is made, but 50k€is at least one is made. Of course revenue is 0 in the former and 70€per item in the latter. I've shown how to formulate that in the course so you can refer them to lecture 5. The fixed load formulation is described on transparencies 12 and 13.

The other constraints are classical, they need to use indicator variables for them. At any rate you should not help them too much for this exercise, there is no particular trap.

They know how to use a spreadsheed program with the associated Solver to find solutions of LP programs. This was explained in the second tutorial. In the dialog box that pops up when filling the constraints, the students can indicate which variables are binary or integer.

3 The textbooks

3.1 Formulation

Company WSP sells textbooks. This company has two sale representatives to best cover a region split into zones. The number of students in each zone is given in figure 1:

Each representative must be associated to two adjacent zones, for instance one rep can be affected to zones A and B but not A and D.

a Propose a formulation allowing WSP to maximize the number of students reached by the two representatives.

b Solve the problem with a spreadsheet program.

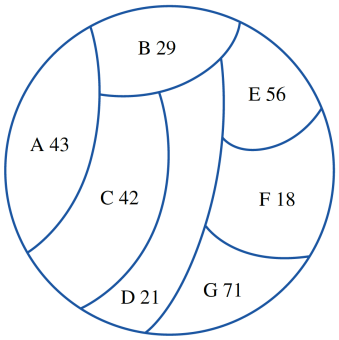

Figure 1: Régions

3.2 Solution

This is a set covering problem, and we have seen how to formulate these in lecture 5 (same as above). This is the last example in the slides.

One difficulty here is to formulate the fact that there should be two representatives that cover effectively 4 zones, and to make sure the covered zone do not overlap. This is doable by writing the constraints explicitely.

However the problem becomes much simpler and can be solved by hand if the students enumerate all the possible adjacent pairs and associate the objective function to the pairs. There are not many possible pairs (AB, AC, BC, etc up to FG). This then becomes an assignment problem with the constraint that exactly two pairs must be chosen, i.e. $x_{AB} + x_{BC} + \ldots + x_{FG} = 2$. The objective is to maximize $w_{AB} + w_{BC} + \ldots + w_{FG}$. The cost w_{AB} is of course the sum of the students in both A and B, i.e. 72, and the x_{yz} are binary.

Then the question might arise how to make sure the pairs don't have any element in common. They could use XOR constraints (i.e only one of AC and AB can be chosen, which is written $x_{AB} + x_{AC} = 1$, (they must do the same with all the pairs with a node in common).

but as it turns out, the optimal solution does not have any overlapping elements, so this set of constraint is in fact unnecessary. Formally they should propose a way to do it though.

The two best pairs are AC and DG. Of course they need to compute this solution and not just guess it!