Practical Session 3 Report: Cutting Stock Problem

YAHIAOUI Borhen, MOHAMED Farouk

April 2024

Contents

1 Context

A textile company produces rolls of fabric 1 meter wide which are then cut into rolls of 21 smaller widths. Customer requests for rolls of each width are aggregated to obtain a total demand that indicates the total number of rolls to be produced in each width. The problem of the company is to minimize the number of rolls of 1 meter cut in order to satisfy the demand. The total demand for different roll sizes are as shown in the following table:

2 Part 1: Pattern Generation + Linear model

2.1 Pattern Generation

To address this problem, we can list all the possible patterns for cutting a roll of 1 meter. For example, one pattern is to cut 1 roll of 45cm, one of 36cm, and one of 14cm. We start by randomly generating a big number of patterns ($= NbPatGen$) which will later be used in the optimization model. To do this, we define an array of a (21 , $NbPatGen$) shape in which a row represents a certain size-roll and a column

Roll size	Number of rolls to cut
72	23
68	80
65	102
63	23
60	57
56	45
52	83
49	125
47	240
45	97
41	435
36	610
31	395
27	350
24	303
22	321
20	160
17	266
16	253
14	211
11	175

Figure 1: Demands for different roll sizes

represents a pattern. So going through the values for each row at a given column gives us the number of rolls to cut for each corresponding size-roll for the considered pattern.
The code for the data generation is as shown below:

```
execute
{
  var pat;
  for(pat in patterns){
    var width; // one of the 21 widths
    var CurrentWitdth = 0; // Total width used
    for (var j=1 ; j<= 21 ; j++)
    {
    width= Opl.rand(21) + 1; // select one width
    if (Pattern[width][pat]==0) { // check if this width has no yet been selected
        var nbRoll = 1+Opl.rand(9); //number of rolls of this width
        //the max. number of rolls is 9
        if ((CurrentWitdth + nbRoll*SizeRoll[width])<=100) {
        //the current pattern is feasible
        Pattern[width][pat] =nbRoll;
        CurrentWitdth = CurrentWitdth + nbRoll*SizeRoll[width];
        }
     }
    }

 }

}
```

Figure 2: Plan for Year 1 provided by Efes

2.2 Linear Model

The model enabling the optimization of the number of rolls used is as follow:
Minimize:

$$Z = \sum_{i \in [1..NbPatGen]} X[i]$$

Subject to:

$$\forall i \quad in \quad 1..21; \quad \forall j \quad in \quad 1..NbPatGen$$
$$Pattern[i,j] \cdot X[j] \quad >= Demand[i]$$

Where $X[i]$ is the number of 1 meter rolls used for pattern i, Pattern in the matrix previously generated and Demand is a column matrix containing the demand for each size-roll j in 1..21.
The corresponding code in CPLEX is as follows:

```
dvar float+ X[i in patt] ;
minimize sum(i in patt)(loss[i]*X[i]);
subject to{
  forall(i in 1..21) C: sum (j in patt)(Pattern[i][j]*X[j]) >=const[i];
}
```

Figure 3: Linear model 1

As we can see in the model, we start by preparing a non integer linear model. We will now proceed with column generation.

3 Part 2: Column Generation

3.1 Model without loss taken into account

In the previous linear model we presented, we included all 500 generated patterns to find those that optimize our consumption of 1meter long rolls in order for us to reach the predicted demand. However, a better way of solving this problem(where a huge number of variables is taken into account) would be column generation. So we proceed as follows:

1. Choose an initial number of patterns significantly less than the total number of patterns(NbPat=50 for example)
2. Solve the linear model for these patterns
3. Search for a variable that we didn't include in our model that has a negative reduced cost and add it to our model
4. Repeat steps 2 and 3 until no variable with a negative reduced cost is found

Since the software of OPL CPLEX doesn't give access to the reduced costs of the variables that were nont included in our model for column generation, we need to add an **execute** part in our code calculating these reduced costs, storing them in a matrix and then displaying the variables that have a negative reduced cost.
The formula we use to calculate the reduced costs is:

$$RC[j] = C_N[j] - y^T \cdot N_j \quad , \quad \forall j \quad in \quad NbPat + 1..NbPatGen$$

Where :

- C_N is the column matrix containing the values for the coefficients of each non basic variable in the objective function. This means that, for the variables X[j] whose reduced costs we want to determine, $C_N[j] = 1$.
- y is the dual column matrix. It is determined directly with CPLEX.
- N_j is the j-th column in the matrix N containing the coefficients of the non-basic variables in the models constraints. In our case, $N_j[i]$ is the number of rolls of j-th size(in the SizeRolls list) to cut in the i-th pattern, $N_j[i] = Pattern[j, i]$

In order for us to select these NbPat patterns from NbPatGen patterns, we introduce a list called *Selected* Composed of 1s if the corresponding patterns is included and 0s if the patterns are not considered at the time being. The model is:
Minimize:

$$Z = \sum_{i \in [1..NbPatGen]} selected[i] \cdot X[i]$$

Subject to:

$$\forall i \quad in \quad 1..21; \quad \forall j \quad in \quad 1..NbPatGen$$
$$selected[i] \cdot Pattern[i, j] \cdot X[j] \quad >= Demand[i]$$

The implementation of this is as follows:

```
dvar int+ X[i in patt] ;
int selected[i in patt];
execute
{
  for(i in index ){
    selected[i]=1;
  }
  for(i in index1){
    selected[i]=0;
  }
  //selected[359]=1;
  //selected[360]=1;
}


minimize sum(i in patt)(selected[i]*X[i]);
subject to{
  forall(i in 1..21) C: sum (j in patt)(selected[j]*Pattern[i][j]*X[j]) >=const[i];
}

execute{
  var i;
  for(i =1;i<=NbPatGen-NbPat;i++){
    c[i]=1
    for(var j in r){
      c[i]=c[i]-C[j].dual*Pattern[j][i];
    }
}
}
```

Figure 4: Model with column generation

We execute the algorithm described above for multiple iterations until we witness a convergence of our objective function's value. The convergence curve is as shown below:

We started off with 50 patterns, and after 23 iterations(23 patterns added) we solved the model to optimality.

3.2 Integer linear model

Keeping into account the fact that the variables need to be integer(the variables are in fact numbers of 1 meter long rolls to use), we need to add the integer constraint in our model. Since Cplex solves this type of problems using the Branch and Bound method, we just need to define our decision variables ad follow **dvar int+ X[i in 1..NbPatGen]** and execute the model to find the solution for the ILM.

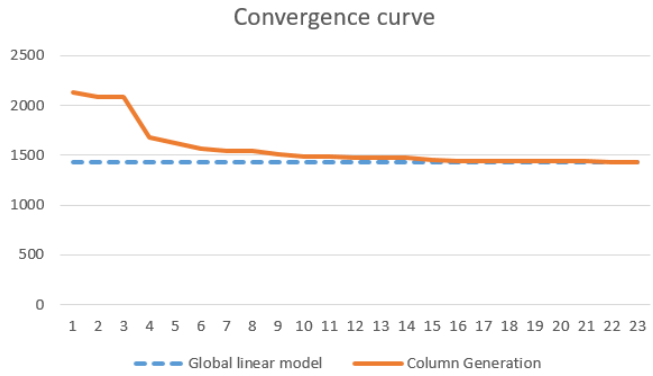

Figure 5: Model with column generation

3.3 Linear model with loss taken into account

Since for almost all the patterns we cut the 1 meter long rolls and lose a certain length that wouldn't be used later on, we need to modify our model to take this into account and optimize the loss. That's why we modify the expression of our objective function by multiplying each decision variable by the amount of leftover length after the patterns is applied to the 1 meter long roll. This way we minimize both the number of rolls used and at the same time the total loss. the model becomes as follows:
Minimize:

$$Z = \sum_{i \in [1..NbPatGen]} loss[i] \cdot X[i]$$

Subject to:

$$\forall i \quad in \quad 1..21; \quad \forall j \quad in \quad 1..NbPatGen$$
$$Pattern[i,j] \cdot X[j] \quad >= Demand[i]$$

where loss is a list containing the amount of roll length leftover after using the i-th pattern.