Operational Research

LINMA2491

Project 1
Benders Decomposition of a Fixed Charge Transportation Problem

Do, Dinh Thanh Phong (MAP) - 13601700
Tran, Minh Phuong (MAP) - 51091600

18 April 2022

We consider the fixed charge transportation problem.

$$\begin{aligned}
&\min_{x,y} \sum_{i=1,j=1}^{n,m} c_{i,j}x_{i,j} + \sum_{i=1,j=1}^{n,m} f_{i,j}y_{i,j} && && (1)\\
&\text{s.t.} && && (2)\\
&\sum_j x_{i,j} = s_i && i = 1 \dots n && (3)\\
&\sum_i x_{i,j} = d_j && j = 1 \dots m && (4)\\
&x_{i,j} \leq M_{i,j}y_{i,j} && i = 1 \dots n, j = 1 \dots m && (5)\\
&x_{i,j} \geq 0 && i = 1 \dots n, j = 1 \dots m && (6)\\
&y_{i,j} \in \{0, 1\} && i = 1 \dots n, j = 1 \dots m && (7)
\end{aligned}$$

With n, the number of suppliers and m, the number of customers.

1 Condition on $M_{i,j}$

In our problem, the constraint given by the inequality (5) tells us that we can only have $x_{i,j} > 0$ when $y_{i,j} = 1$, i.e. we can transport a quantity $x_{i,j} > 0$ from supplier i to customer j only if we "open" the route from i to j.

Since each customer j needs a quantity d_j and each suppliers i produces a quantity s_i the constraint 5 involves there is a quantity $M_{i,j}$ which serves as an upper bound of what can be transported from a supplier i to a customer j. Moreover, we don't want this upper bound to actually constraint the quantity $x_{i,j}$. We want it to be large enough for $x_{i,j}$ to take any values possible when the route is "open", i.e. $y_{i,j} = 1$. We want to set the smallest possible value for $M_{i,j}$.

From the constraints 3 and 4, it is clear that :

$$\begin{aligned}
&x_{i,j} \leq s_i && i = 1 \dots n, j = 1 \dots m && (8)\\
&x_{i,j} \leq d_j && i = 1 \dots n, j = 1 \dots m && (9)
\end{aligned}$$

Then, it is natural to give $M_{i,j}$ the following value:

$$M_{i,j} = \min(s_i, d_j) \qquad i = 1, .., n \quad j = 1, ..., m \tag{10}$$

2 Conditions on parameters s_i, d_j

For our problem to be feasible we need to set a condition on s_i and d_j, the quantities supplied (from each supplier i) and demanded (from each customer j) . Indeed, if we want to both meet exactly all the demand requirements and sell everything that is supplied, the only constraint which makes sense is when the sum of the demands is equal to the sum of the supplies.

$$\sum_{i=1}^{n} s_i = \sum_{j=1}^{m} d_j \tag{11}$$

And of course $s_i, d_j \geq 0$.

3 Solving the optimization problem with the Benders decomposition

We are interested in solving this problem for 20 suppliers and 30 customers. For that, we generated a random instance and used the Benders decomposition to solve it.

As we will discuss it in the following subsections, the classical Benders decomposition might not be the most effective way to solve our problem in such high dimension. Which is why we will present first our results on smaller examples. And discuss its efficiency compared to a direct solve.

The fixed charge transportation given in (1) - (7) contains both positive variables x_{ij} and binary variables y_{ij}. The idea behind Benders decomposition is to decompose the problem in smaller parts: the first stage master problem 12 and the second stage subproblem 13. By solving those 2 problems recursively, we will obtain the objective value 1.

Master Problem The master problem is described by 12. The objective function of the master problem corresponds to the objective of our initial problem (1) for which θ corresponds to $\sum_{i=1,j=1}^{n,m} c_{ij}x_{ij}$. However, the main difference with the initial problem is that the constraint 4,5,6 are contained (partially) in the "Cuts" set which will be added if needed after solving the Benders sub-problem.

$$\begin{aligned} \min_{y} \quad & \theta + \sum_{i,j}^{n,m} f_{ij}y_{ij} \\ \text{st.} \quad & \text{Cuts} \\ & \theta \geq 0 \\ & y_{ij} \in \{0,1\} \end{aligned} \tag{12}$$

Benders sub-problem $S(y)$ in its primal form The idea of the Benders decomposition algorithm is to first solve the master problem and see if the solution $\hat{y}$ found permit a feasible solution for the sub-problem and if it is, optimality is reached. The sub-problem clearly depends on the value of $\hat{y}$, which is why we will denote it $S(y)$.

In the cases where the sub-problem is unfeasible (i.e. the $\hat{y}$ provided doesn't belong to the domain of $S(y)$), or that the optimality condition is not met, the sub-problem will provide new feasibility or optimality cuts to the master problem in order to have a better idea of the domain of $S(y)$.

$$\begin{aligned} \min_{x} & \sum_{i,j}^{n,m} c_{ij}x_{ij} \\ \text{st.} \quad & \sum_{j}^{m} x_{ij} = s_i && i = 1, ..., n \\ & \sum_{i}^{n} x_{ij} = d_j && j = 1, ..., m \\ & x_{ij} \leq M_{ij}\hat{y}_{ij} \\ & x_{ij} \geq 0 \end{aligned} \tag{13}$$

Benders sub-problem in its (classic) dual form The dual form of the Benders sub-problem given in 14 is found by dualizing the sub-problem given in 13. The benefit of the dual sub-problem is that it allows us to have an explicit expressions for the necessary optimality / feasibility cuts which will be then added to the master problem.

$$\begin{aligned} \max_{v,u,w} \quad & \sum_{i=1}^{n} v_i s_i + \sum_{j=1}^{m} u_j d_j - \sum_{i,j=1,1}^{n,m} w_{ij} M_{ij}\hat{y}_{ij} \\ \text{st.} \quad & v_i + u_j - w_{ij} \leq c_{ij} && i = 1, ..., n \quad j = 1, ..., m \\ & w_{ij} \geq 0 && i = 1, ..., n \quad j = 1, ..., m \\ & u_j, v_i \in \mathbb{R} && i = 1, ..., n \quad j = 1, ..., m \end{aligned} \tag{14}$$

3.a Comparison between the direct solve and the Benders decomposition

In our implementation of the Benders decomposition, we have used the Gurobi solver since it is faster than other solvers. We have set a few parameters for the solver to speed up the calculations (of course, they can be reset so that the solver do not use them):

- According to Gurobi, `presolve` allows the solver to map the problem to a smaller one which will be easier to compute. The solver will solve the smaller problem, obtain a solution, and then map back to the initial problem. We have used the presolve parameter for the master problem (MIP) since this problem possesses more and more cuts as the algorithm iterate. However, for the subproblem, no presolve has been used because it doesn't give the right attribute for `TerminationStatus` and `PrimalStatus`.
- We have also modified the MIPFocus (Most Important Parameter Focus). This parameter allows us to "modify our high level solution strategy depending on our goal". Since we want it to solve as fast as possible, we have set this parameter `MIPFocus` to 1.

	Benders decomposition	Direct solve
run-time	4.5456	0.0995
iterations	76	
optimality cuts	53	
feasability cuts	22	

Table 1: Comparison between Benders decomposition and the direct solve for a problem of size $(n, m) = (5, 10)$

	Benders decomposition	Direct solve
run-time	178.5001	0.1410
iterations	331	
optimality cuts	77	
feasability cuts	254	

Table 2: Comparison between Benders decomposition and the direct solve for a problem of size $(n, m) = (8, 10)$

	Benders decomposition	Direct solve
run-time	2110.9532	0.1658
iterations	679	
optimality cuts	554	
feasability cuts	124	

Table 3: Comparison between Benders decomposition and the direct solve for a problem of size $(n, m) = (10, 15)$

The solutions provided by the direct solve and Benders decomposition that we implemented are equal to a tolerance of 10^{-11} (we computed the 2-norm of the difference between the 2 solutions). This can be explained by the rounding errors by the computer.

We observe that the run-time of the Benders decomposition is much more important than the run-time of the direct solve. The Benders decomposition allows us to divide the problem into one smaller mixed integer optimization problem and a linear optimization problem, which should be easier to compute. However, after our implementations, we observe that in reality, it is not faster than the direct solve. Indeed, the convergence rate is not fast enough, and at each iteration, the master problem (MIP) grows with an additional cut which makes it more difficult to solve at each iteration.

We can observe this in the Figure 1. Indeed, while the computation time of the sub-problem remain fairly low and usually don't change much (since it is a LP problem with the same amount of variable and constraints at each iteration), the computation time of the master problem has a general tendency to increase at each iteration until a certain point.

One peculiar behaviour is that after a certain iteration (way before finding a feasible $\hat{y}$ for the sub-problem $S(y)$, which we do not fully understand), the computation time of the master problem drops at once and do not have a tendency to increase anymore. This means that the first feasibility cuts make the master problem more difficult than the later cuts do.

We can also wonder why the convergence rate is not good. To try to explain it, we can observe the number of feasibility cuts which is higher than the number of optimality cuts. It seems that it is very difficult to find a $\hat{y}$ such

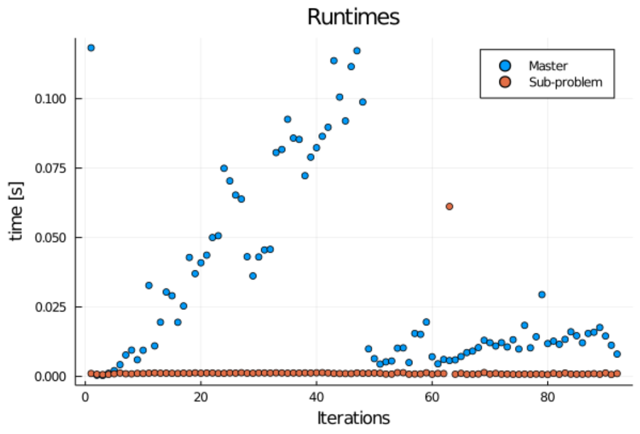

Figure 1: Evolution of run-times of Master problem and sub-problem at each iteration. The problem solved here had $(n, m) = (5, 8)$. The number of feasibility cuts is 67 and the number of optimality cuts is 24. We observe a drop in the run-time of the master problem at iteration 49.

that the sub-problem is feasible. Intuitively, we can see that describing the domain of $S(y)$ which is in very high dimension, with little number of hyperplanes can be difficult.

For the case with $(n, m) = (20, 30)$, the computation time with the Benders decomposition was way too important especially compared to direct solve. It is not reasonable to use the Benders decomposition for that problem.

The direct solve gave an optimal value of $1799815, 05$.

With the Benders decomposition, we could unfortunately not reach the end of the computations. Indeed, after 24 hours of running time and 410 iterations, we still didn't converge to the solution. The lower bound that we obtained at that iteration for the optimal value is 519.48 which is very far from what we obtained with the direct solve.

4 Conclusion

In conclusion, while the idea of the Benders decomposition makes sense, and seems at first suitable for our problem, it is in reality not as efficient as we could hope. Indeed, the convergence rate is too low and the master problems become increasingly (too) difficult to solve in a small amount of time. And it becomes even worse in higher dimensions.

To go further, it can be interesting to use the basis of this Benders decomposition algorithm and add it to other strategies to make it more efficient.