Erasmus University Rotterdam

FEM21040

Mathematical Programming Assignment 2 - Dantzig-Wolfe Algorithm

Author:
Wessel Boosman 589273

29 september 2021

Graded assignment 2 - Dantzig-Wolfe Algorithm

In this assignment a multi-commodity flow problem is considered. The first part of the assignment will be dedicated to some interpretation of the Dantzig-Wolfe reformulation of the problem. Thereafter two multi-commodity flow problems will be solved.

A. A Dantzig-Wolfe reformulation of the multi-commodity flow problem can be constructed. The reformulation can be decomposed into multiple activities and a set of linking constraints. Each commodity flow is an activity in the reformulation. The linking constraints are the capacity constraints on the arcs. This makes sense since all commodities are sent over the same network, and are therefore *'linked'* by these commodity independent capacity constraints.

B. Yes, it is possible to get an indication upfront whether the subproblem is unbounded. When solving the subproblem of an activity (commodity) you are looking for an extreme point of the polyhedron corresponding to this activity. This extreme point actually corresponds to a shortest path (or in our case lowest cost) in the network over which you can send this commodity. If there is a cycle on the network with negative cost and positive capacity, the commodity can be send over this cycle over and over again and thus reducing the cost to minus infinite. When this is the case, the subproblem is unbounded.

C. The Dantzig-Wolfe decomposition algorithm can be initialized by including only slack variables for the capacity constraints and only dummy variables for the activity constraint in the 'Restricted Master Problem' (RMP) such that all constraints are satisfied. The dummy variables corresponding to each activity constraint are multiplied with a very large number and then included in the objective function. The dummy variables in the objective function should actually correspond to an extreme point, and thus a path in the network. So the dummy variables can be seen as extremely expensive paths in the network. Therefore, these dummy variables will never occur in the optimal solution since we solve a minimization problem. In other words, we will find paths in the network for each commodity that have costs far less then the arbitrarily chosen large cost in the initialization.

D. At each iteration, the algorithm adds an extreme point to the RMP for one of the activities. In other words, at each iteration a shortest path for a certain commodity is added to the RMP.

E. The solution of the small commodity flow problem can be found in table 1. I made use of an algorithm with a greedy approach, that means that the extreme point corresponding to the first subproblem the algorithms finds with a reduced negative cost will be added to the RMP. The final solution is given in equation 2 to 4.

$$\vec{x}_k = (f_{12,k}, f_{13,k}, f_{24,k}, f_{25,k}, f_{32,k}, f_{34,k}, f_{35,k}, f_{45,k}, f_{46,k}, f_{56,k}) \tag{1}$$

$$\vec{x}_1 = (1.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0) \tag{2}$$

$$\vec{x}_2 = (3.0, 0.0, 0.0, 3.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0) \tag{3}$$

$$\vec{x}_3 = (0.0, 2.0, 0.0, 0.0, 0.0, 2.0, 0.0, 0.0, 2.0, 0.0) \tag{4}$$

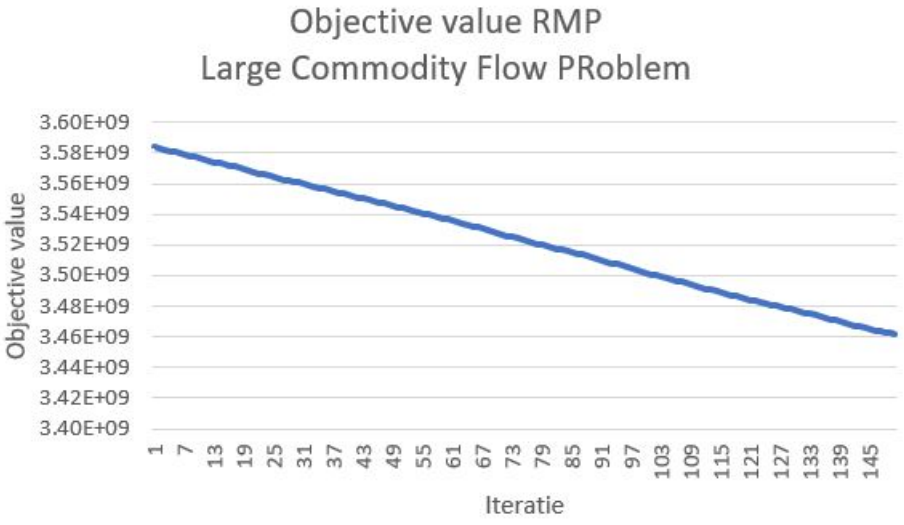
Tabel 1: Table with solution information of the Dantzig-Wolfe Algorithm performed on a small commodity flow problem.

Iteration	Objective value RMP	Extreme point	Reduced Cost	Commodity	Optimal?
0	3000	[1.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]	-998.0	1	No
1	2002	[0.0, 1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0]	-985.0	1	No
2	2002	[3.0, 0.0, 0.0, 3.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]	-991.0	2	No
3	1011	[3.0, 0.0, 3.0, 0.0, 0.0, 0.0, 0.0, 3.0, 0.0, 0.0]	-964.0	2	No
4	1011.0	[2.0, 0.0, 0.0, 2.0, 0.0, 0.0, 0.0, 0.0, 0.0, 2.0]	-990.0	3	No
5	373.33	[0.0, 3.0, 0.0, 0.0, 0.0, 0.0, 3.0, 0.0, 0.0, 0.0]	-934.0	2	No
6	59.0	[0.0, 2.0, 0.0, 0.0, 0.0, 2.0, 0.0, 0.0, 2.0, 0.0]	-968.0	3	No
7	43				Yes

F. The solution values for the RMP in the case of a large multi-commodity flow problem are plotted in figure 1. These objective values do not represent any kind of useful solution values. This is the case for the following reason, since there are 3480 commodities, there are also 3480 activities and 3480 dummy variables which need to be replaced by paths in the network with less costs then the arbitrarily chosen value as discussed in question 'C'. This means that at least 3480 iterations of the Dantzig-Wolfe algorithm have to be performed, in the case that at each iteration a subproblem is solved with a negative reduced cost. These iterations take too much time in my algorithm and therefore after ten minutes the objective value is still very high. I tried the following things to speed up my algorithm:

- Solve the first subproblem with a negative reduced cost, this has the disadvantage of risking optimizing the flow a commodity, while a much greater reduction of the objective value can be obtained by solving the subproblem of a different commodity.
- Add an extreme point corresponding to the subproblem with the lowest reduced cost. This has the disadvantage of solving 3480 subproblems in each iteration.
- Solve a random subproblem. This has the advantage that in the beginning you will probably solve a subproblem in which you obtain a great reduction in objective value and you only have to solve one subproblem. The disadvantage is that you need a large number of iterations, since ideally you want to solve each subproblem at least a few times, and the subproblem you solve each iteration is random.

In hindsight, the best thing to do, was probably solve all subproblems and add in one iteration all the extreme points to the RMP corresponding to the subproblems with a negative reduced cost. Still this would require at least multiple iterations, and one iteration in which I solved all subproblems already took over five minutes.



Figuur 1: Objective value of the RMP plotted against iterations