ECOLE POLYTECHNIQUE FÉDÉRAL DE LAUSANNE

SEMESTER PROJECT

A solution approach for the Multicommodity Flow Problem within rail freight transportation

Author:
Nicolas PRADIGNAC

Supervisor:
Mr. Stefano BORTOLOMIOL
Mr. Nikola OBRENOVIC

A semester project in fulfillment of the requirements

in the

TRANSP-OR Laboratory

May 2, 2018

ECOLE POLYTECHNIQUE FÉDÉRAL DE LAUSANNE

Abstract

TRANSP-OR Laboratory

A solution approach for the Multicommodity Flow Problem within rail freight transportation

by Nicolas PRADIGNAC

The objective of this project is to study the Muticommodity Flow Problem within the transport industry. In that field, companies aim to minimize the cost of routing the goods given that some resources, such as railways, have limited capacities. The report is divided into two stages. First the two main variants of the MCF : the path-based formulation and the arc-node model are coded and experimented using CPLEX on small datasets. Afterward, another approach called Column-Generation is implemented in Python to solve the MCF for larger instances.

Contents

Chapter 1

Literature Review

The multicommodity flow problem (MCFP) is a well-known network flow problem which has been studied since 1958 by William S. Jewell. As the MCFP comes up in many areas, the problem can be visualized in many ways. For instance, it can be described as follows :

Some producers want to ship their products, through railways, to some consumers. Both producers and consumers are modeled as nodes. Each product, called commodity, is identified by 3 components : its source (the producer of the product), its destination (the consumer of the product) and its quantity. Moreover, each railway, modeled by an arc in the network, is characterized by its cost per quantity unit and its maximal quantity capacity. In some cases, the nodes may be restricted by a maximal capacity as well. Then, the aim is to assign a path to each commodity from its source to its destination, while minimizing the total cost function and without exceeding the maximal capacity constraints.

The MCFP can be expressed as a linear-program and consequently solved in polynomial time using common LP techniques. However, these methods are not very efficient to solve large instances of the problem. That is why, today, heuristic or alternative methods are mainly employed to solve the MCFP.

The MCFP can model a wide variety of real-world problem [1], and a lot of case-specific variants exist. We can split them in three main categories depending on the domain of the decision variables, either they are integers, then the problem is called integer MCFP, approximation algorithm are mainly used in this case. Second, the decision variables take on real values and the problem can be modeled as a LP. The last category is mixed integer MCFP, where some decision variables are integers and others real values. It has been proved that in integer and mixed integer MCFP, the problem becomes NP-hard.

Today most of the methods employed to solve the MCFP use "decomposition" techniques and can be classified in three categories [2] :

1. **Price-directive decomposition** :
   The idea is to replace complicated constraints by adding "prices" variables which penalize bad solutions. This leads to solving easier LP and so gain efficiency. Some widely use price-directive methods are Lagrangian relaxation, Dantzig-Wolfe decomposition or Column-Generation.

2. **Resource-directive decomposition** :
   These methods aim for reducing the MCFP to multiple single commodity flow

problem which can then be solved very efficiently. To do so, a new constraint is introduced which allocate a certain amount of capacity on each arc for each commodity, without exceeding the total original arc capacity. However these methods does not ensure convergence.

3. **Partitioning methods** :
   Most of the well-known LP solver methods can be adjusted to the MCFP specifically, for instance the "network simplex" is a speed-up modified version of simplex method.

Besides, it has been shown that the resource-directive algorithms converge rapidly for small problems but are outperformed by the price-directive techniques for large instances.
The above description of the MCFP make us think that this is a mainly transportation problem. However this problem comes up in a lot of other fields ranging from scheduling and telecommunication to image processing.

First, Niluka Rodrigo1 and Lashika Rjapaksha [3], showed that the MCFP can be used to solve the facility location problem (FLP). In this problem the goal is to ship several goods from plants to customers while passing through a distributed center. The goal is thus to determine which distribution center to assign to which plant, while minimizing the transportation cost and satisfying the capacity constraints. Murat Oguz, Tolga Bektas, Julia A. Bennell [4] describe in their paper a restricted version of the FLP based on the interger MCFP. Moreover, they use a Benders decomposition algorithm to optimally solve the problem, and show that the performance of their algorithm has a better running time than the previous best-known algorithm. Unlike the Column-Generation approach, the Benders decomposition method adds new constraints as it converges to the optimal solution.

Besides, Jean-François Cordeau, François Soumis and Jacques Desrosiers [9] propose a method to solve the assignement of locomotives and cars to trains problem. This minimization problem is subjected to compatibility equipment type constraints, making the problem harder. To solve it, they extent their previous approach(2000) based on the MCFP, and use a combination of branch-and-bound, followed by a Bender decomposition to solve a mixed-integer program at each node, in which the subproblems, corresponding to LP-relaxations of the MCFP are optimized by a Dantzig-Wolfe decomposition.

Another major problem in public transport, involving the MCFP, is the line-planning problem. Given a transportation demand, it consists in defining suitable line routes and their frequencies, in a public transport network, made of several modes (bus, tram, subway...). Most of the time, the objective is to minimize the operating costs of lines and the total passenger traveling time. Borndörfer, Ralf, Martin Grötschel, and Marc E. Pfetsch [10] propose a new multicommodity flow model for this problem where the commodities correspond to the demand between two stations. First, the algorithm discussed solve the LP-relaxation of the problem using a Column-Generation approach and then uses a greedy algorithm to construct a good integer solution. Their experiments show that their algorithm has a significant optimization potential and might be efficient to solve large instances.

Moreover, the MCPF comes up in many air transportation systems. Indeed, Dai, Weibin, et al.[11] propose an new heuristic algorithm to assign passenger demands, seen as commodities, to feasible airlines. Most of the time, the cost vector is based on the distance between the nodes. Their algorithm is based on column generation approach, in the sense that, first, they generate an initial set containing short paths. The shortness of a path is first measured in terms of pure distance value, then they adapt a new measure, using the number of nodes that a path contains to define their set of relevant paths. Based on their experiment, it turns out that the node-based distance measure is much more efficient. Even if their path-finding algorithms are time-consuming, a similar approach will be employed in the second part of the report.

As previously said, the MCFP can be found is scheduling problems too. Indeed Arton P. Dornelesa, Olinto C. B. de Araújob and Luciana S. Buriola [5] published a paper where they propose a mixed integer MCF model for the high-school timetabling problem. The goal is to build weekly schedule where a set of classes must be assigned to timeslots. Basically, the arcs represent transition of time periods and each teacher is represented by a commodity. To solve the problem, they propose an algorithm based on Dantzig-Wolfe decomposition which is a price-directive decomposition, and use a column generation approach to solve their Master problem. Moreover, two strategies are proposed to speed-up the algorithm. The analysis shows that the running time of their method is faster than the previous existing algorithm.

Finally, in most of the real-time applications, the commodities are unsplittable. Indeed in the maritime surveillance case, Hela Masri1, Saoussen Krichen, and Adel Guitouni[6] proposed to model the problem as an integer MCFP. Some messages, modeled as commodities, need to be shared among several nodes, surveillance employees, for instance. Then, the goal is to minimize the most congested arc traffic value. An effective heuristic method is proposed, called Hybrid Genetic Algorithm (HGA), and is compared to another method called Ant Colony System (ACS). It turns out that HGA produces slightly better output value than ACS and most importantly, the running time of HGA is extremely smaller than ACS for large instances.

The MCFP can even be found in, a priori, not related field, as image processing. Indeed Horesh Ben Shitrit, Jerome Berclaz, Francois Fleuret, and Pascal Fua [7] formulated a multi-persons tracking problem as a MCFP. In result, they obtained one of the best tracking algorithm so far.

All those examples show that a wide variety of methods exist to solve MCFP. Weibin Dai, Jun Zhang, Xiaoqian Sun [8] discussed the performance of two widely used methods, namely, Column-Generation and Lagrangian relaxation across various implementations and datasets. The results indicate that, in general, column generation approach obtains better output values and running times than Lagrangian relaxation, except in large network with high number of commodities where the running time of Lagrangian relaxation is slightly lower.

Chapter 2

Arc-Node and Path-based models

From all these papers, which present approaches to real-life problems based on the MCFP, we observe that it can be formulated in many ways. In this report we will focus on the two mains formulations used today, to solve the MCFP. Also, as this report is related to the rail freight transportation, the objective is to minimize the total cost to transport all the goods from their source to their recipient. The nodes represent the stations, the arcs correspond to railways and commodities are goods. Throughout this report, each commodity is sent from exactly one node and is shipped to exactly one node. Moreover all the commodities are splittable.

2.1 Arc-Node model

Given a directed graph $G := (N,A)$, the arcs are defined by their start and end nodes (i, j), and their cost per unit c_{ij}. Each node $n \in N$ and arc $(i, j) \in A$ have a maximum capacity u_n and v_{ij} respectively. Moreover, a commodities $k \in K$ is characterized by its start and end nodes, as well as a quantity q_k. Finally, to simplify the expression of the flow conservation constraints, we define a variable δ_n^k which equals 1 if the node n is the start node of the commodity k, equals -1 if n is the end node, otherwise it equals 0. The decision variable x_{ij}^k correspond to the fraction of the total quantity of the commodity k passing through the arc (i, j).
The node-arc formulation is defined as follow :

$$\text{minimize} \quad \sum_{k\in K} \sum_{(i,j)\in A} c_{ij} x_{ij}^k q_k \tag{1.1}$$

$$\text{subject to} \quad \sum_{(i,j)\in A} x_{ij}^k - \sum_{(j,i)\in A} x_{ji}^k = \delta_i^k \quad \forall i \in N, \forall k \in K \tag{1.2}$$

$$\sum_{k\in K} \sum_{(j,i)\in A} x_{ji}^k q_k \leq u_i \quad \forall i \in N \tag{1.3}$$

$$\sum_{k\in K} x_{ij}^k q_k \leq v_{ij} \quad \forall (i,j) \in A \tag{1.4}$$

$$x_{ij}^k \geq 0 \quad \forall (i,j) \in A, \forall k \in K \tag{1.5}$$

This formulation aims to minimize the total cost of shipping all the commodities (1.1), while conserving, for each commodity, the entire flow between the start and end nodes (1.2). Moreover, the total quantity of the commodities at each node and arcs is limited (1.3),(1.4), and the commodities are allowed to split during the shipping (1.5).
The arc-node model represents the MCFP in a intuitive and stepwise way. Indeed, it

can be seen as follow, for each arc (*i,j*), how valuable is it to assign a commodity k to the arc.

2.2 Path-based model

Another widely used MCF formulation is called path-based model. Given a graph $G := (N, A)$ similar to the one defined in the arc-node model, nodes and arcs have a capacity u_n and v_{ij} respectively, the cost per unit of an arc is c_{ij}, and the quantity of the commodity k is q_k. Also, we define the sets $P(k)$ such that it contains all the possible paths from the start and end node of the commodity k. The cost per unit of a path p is denoted c_p, it corresponds to the sum of c_{ij}, for all arcs (*i,j*) contained in the path p. Unlike the arc-node variant, the decisions variables are x_p^k which is the fraction of the total quantity of the commodity k passing through the path $p \in P(k)$. Besides, to simplify the formulation of the capacity constraints, we define α_{ij}^p to be equals to 1 if the arc (*i,j*) is a part of the path p. As well as β_n^p to be equal to 1 if the node n is contained in the path p.
The path-based formulation is described as follow :

$$\text{minimize} \quad \sum_{k \in K} \sum_{p \in P(k)} c_p x_p^k q_k \tag{2.1}$$

$$\text{subject to} \quad \sum_{p \in P(k)} x_p^k = 1 \qquad \forall k \in K \tag{2.2}$$

$$\sum_{k \in K} \sum_{p \in P(k)} x_p^k q_k \beta_i^p \leq u_i \qquad \forall i \in N \tag{2.3}$$

$$\sum_{k \in K} \sum_{p \in P(k)} x_p^k q_k \alpha_{ij}^p \leq v_{ij} \qquad \forall (i,j) \in A \tag{2.4}$$

$$x_p^k \geq 0 \qquad \forall p \in P(k), \forall k \in K \tag{2.5}$$

Similarly, the objective is the minimize the total cost of transporting all the commodities(2.1), while conserving, for each commodity all the flow from the start node to the end node (2.2). Besides the nodes and arcs have maximal capacities (2.3),(2.4) and the commodities can split during the transportation(2.5).

This formulation allow us to visualize the MCFP in another way, instead of focusing on the arcs, we look at the bigger picture and compute how good it is to assign a certain amount of a commodity to a path.

2.3 Analysis and Experiments

Thus, we defined the MCFP in two different ways. It would be interesting to know in which cases a formulation is more efficient than the other one. Table 1 show the number of variables and constraints in both formulations. Let $P = \bigcup_{k \in K} P(k)$. First, observe that in general, $\|P\|$ is extremely larger than $\|A\|$, so the number of variables in the path-based formulation is highly bigger than in the arc-node model. Besides the number of variable grows exponentially with the number of arcs in the path-based model. However, notice that the number of flow conservation constraints is highly

lower in the path-based model. Note that the positivity constraints (1.5),(2.5) are easy to very, and thus not taken into account during the analysis. Besides, the number of nodes and arcs capacities constraints are the same. Thus, as the number of commodities grows, the number of constraints in the arc-node model grows greatly faster than in the path-based model.

	#Variables	#Constraints	
Arc-node formulation	$\|K\|\|A\|$	$\|K\|\|N\|$	(1.2) Flow conservation
		$+ \|N\|$	(1.3) Node capacity
		$+ \|A\|$	(1.4) Arc capacity
		$+ \|K\|\|A\|$	(1.5) Positivity
Path-based formulation	$\|K\|\|P\|$	$\|K\|$	(2.2) Flow conservation
		$+ \|N\|$	(2.3) Node capacity
		$+ \|A\|$	(2.4) Arc capacity
		$+ \|K\|\|P\|$	(2.5) Positivity

TABLE 2.1: Number of variables and constraints in the arc-node and path based formulations

The performance of these two different formulations are tested using data sets of small, medium and large size, while varying the node and arc capacities. All the experiments were run on a PC with a 1,6 GHz Intel Core i5 processor and 8Go of RAM, using CPLEX v12.8.0.0. The tables 2.2, 2.3 and 2.4 present the results obtained on the datasets.

	Constraints values	Objective func.	Running time
	$u_n = \infty, v_{ij} = \infty$	1 623 760	0.63sec
Arc − node	$u_n = \{2800, 5600, 8400\}, v_{ij} = \{560, 980, 1400, 1820\}$	1 628 400	1.01s
	$u_n = \{2600, 5200, 7800\}, v_{ij} = \{520, 910, 1300, 1690\}$	1 657 820	0.91s
Memory peak	$u_n = \{2400, 4800, 7200\}, v_{ij} = \{480, 840, 1200, 1560\}$	1 690 260	0.78s
$\simeq$ 24M	$u_n = \{2200, 4400, 6600\}, v_{ij} = \{440, 770, 1100, 1430\}$	1 724 660	0.81s
	$u_n = \{2100, 4200, 6300\}, v_{ij} = \{420, 735, 1050, 1365\}$	unfeasible	0.58s
	$u_n = \infty, v_{ij} = \infty$	1 623 760	65.86s(30.8 + 1.2 + 8.7)
Path − based	$u_n = \{2800, 5600, 8400\}, v_{ij} = \{560, 980, 1400, 1820\}$	1 628 400	70.98s(30.2 + 1.3 + 11.8)
	$u_n = \{2600, 5200, 7800\}, v_{ij} = \{520, 910, 1300, 1690\}$	1 657 820	66.2s(30.6 + 1.2 + 8.8)
Memory peak	$u_n = \{2400, 4800, 7200\}, v_{ij} = \{480, 840, 1200, 1560\}$	1 690 260	74.9s(32.3 + 1.4 + 9.75)
$\simeq$ 1.5G	$u_n = \{2200, 4400, 6600\}, v_{ij} = \{440, 770, 1100, 1430\}$	1 724 660	69.32s(31.2 + 1.1 + 8.7)
	$u_n = \{2100, 4200, 6300\}, v_{ij} = \{420, 735, 1050, 1365\}$	unfeasible	39.70s(30.1 + 1.68)

TABLE 2.2: Small dataset : 20 nodes, 48 arcs, 202 commodities.
The nodes and arcs constraints are split in 3 and 4 categories respectively. The running time is contains 3 main stages : (loading and precompute the data + solving time + postprocessing time)

	Constraints values	Objective function	Running time
Arc − node	$u_n = \{40, 80, 100, 150\}$, $v_{ij} = \infty$	unfeasible	
Memory peak	$u_n = \{45, 80, 150, 200\}$, $v_{ij} = \infty$	43 598 821,5	1h15
$\simeq$	$u_n = \{60, 90, 130, 180\}$, $v_{ij} = \infty$	42 909 708	1h19
	$u_n = \infty$, $v_{ij} = \infty$	42 469 841	1h21
Path − based		Require too much pre-compuation	

TABLE 2.3: Medium dataset : 2172 nodes, 4546 arcs, 242 commodities. The quantity of each commodity is lower than in the small dataset, which explains the low capacity constraints values. For the moment the arcs capacity constraints are not included.

	Constraints values	Objective function	Running time
Arc − node Memory peak $\simeq$	$u_n = \infty$, $v_{ij} = \infty$	536 675 196 (to be confirmed)	12h31 (to be confirmed)
Path − based		Require too much pre-compuation	

TABLE 2.4: Large dataset : 2172 nodes, 4546 arcs, 1173 commodities. For the moment the arcs capacity constraints are not included.

First, notice that CPLEX solve to optimality the problem, which is why the objective function values are the same in both formulation. Moreover, for each dataset, the number of arcs is equals to a bit more than twice the number of nodes. The sparse structure of the network explains why the objective function grow relatively slowly from the unconstraint to unfeasible instance.

For the small dataset the running time of the arc-node formulation is extremely faster than the path-based model. Indeed, the later require a large amount of pre-computational time, to generate the set $P(k)$ for each commodity and the number of variables is larger in the path-based formulation. Besides, the path based model requires much more memory than the arc-node formulation, again it can be explained by the large difference of number of variables generated between the two models.

Chapter 3

Column-Generation approach

The experiments performed in last section show that the arc-node formulation is extremely faster, and consumes less memory than the path-based approach. The later tend to be even computationally unfeasible as the size of the network grows. From the running time analysis and the table 2.1, the main reason explaining the huge gap between these models is the fact that the path-based approach contains an enormous number of variable and grows very fast as the size of the network increases, resulting in a large amount of pre-computational time to generate all of them.

However from Table 2.1, we notice that the main benefit of the path-based approach is the low number of constraints. Moreover, most of the time the optimal solution contains only a small amount of variables among all the ones generated, meaning that we do not need all variables to obtain the optimal solution. One way to leverage these observations would be to employ a method which does not require all the variables, but only the relevant ones. From the literature review we have seen that a wide variety of approaches exist, with the aim of cutting down the amount of variables considered. One of them is the column generation approach, which is today extensively used. Besides, it is an exact method which is well suited for our datasets given their sparse structures. Finally, the comparative analysis performed in Weibin Dai, Jun Zhang, Xiaoqian Sun paper [8], tend to show that column generation approach outperform Lagrangian decomposition for large instances.

Hence, the column generation approach appears to be a very efficient method in our case, we are going to apply it to our network.

3.1 How does it work ?

First, we call the problem that we desire to solve, the Master problem (MP). In our case, it is the path-based model described from (2.1) to (2.5), without constraints (2.3) for the moment. As stated earlier, the goal is to narrow down the number of variables generated, and solve the problem to optimality. This problem is called the Restricted Master problem (RMP). The difference between the MP and the RMP is only that the RMP contains a subset of variables instead of all of them.

The optimal solution will be obtained by solving the RMP repeatedly. However, how can we be certain that there is not a better solution if we do not even generate all variables ? To resolve this issue, we will make use of the Primal-Dual Optimality conditions.

Theorem 1. *Given a primal linear problem P, let us denote its corresponding dual problem D :*

$x*$ and $y*$ are respectively the optimal solution of the primal and dual problem if and only if the three following conditions hold :

- *Primal feasibility : x* satisfies each constraint of P*
- *Objective function : The value of the primal objective function for x* is the same as the dual objective function value for y*.*
- *Dual feasibility : y* satisfies each constraint of D*

Thus, given the optimal solution to the Restricted Master problem called $\widetilde{x*}$ and its dual variables $\widetilde{y*}$, we can confirm that $\widetilde{x*}$ satisfies the Primal feasibility condition as the RMP contains the same constraints as the MP. Besides the Objective function condition is also satisfied thanks to the strong duality theorem. The only condition left that we must check, to ensure that the optimal solution of the RMP is also the optimal solution to the MP, is the dual feasibility. That is, verify that $\widetilde{y*}$ satisfies all the constraints of the dual of the MP.

Let y_k with $(k \in K)$ and $y_{(i,j)}$ such that $(i,j) \in A$ denote the dual variables corresponding to the k^{th} constraint of (2.2) and to the $(i,j)^{th}$ constraint of (2.4). The dual problem of the MP is formulated as follows : (TO DO : INCLUDE NODE CAPACITY CONSTRAINTS + INCLUDE SMALL FIGURE TO SHOW ALL THE DIFFERENT VARIANTS OF THE MCFP)

$$\begin{aligned}
\text{maximize} \quad & \sum_{k \in K} y_k + \sum_{(i,j) \in A} y_{(i,j)} v_{ij} && && (3.1)\\
\text{subject to} \quad & y_k + \sum_{(i,j) \in A} y_{(i,j)} q_k \alpha_{ij}^p \leq c_p^k q_k && \forall k \in K, \forall p \in P(k) && (3.2)\\
& y_k \in \mathbb{R} && \forall k \in K && (3.3)\\
& y_{(i,j)} \leq 0 && \forall (i,j) \in A && (3.4)
\end{aligned}$$

From the Theorem 1, we know that the aim now is to verify whether $\widetilde{y*}$ satisfies all the (3.2) constraints, this problem is called the Pricing problem. Nonetheless, the amount of (3.2) constraints is equal to the number of variables in the MP, which is extremely large. This huge quantity of variables was the reason why we needed to find a new efficient approach to exploit the path-based model. Thus the key is to identify an efficient technique to verify all those constraints without explicitly enumerate all of them.

We are going to formulate the (3.2) constraints in another way to tackle the pricing problem from a new point of view :

$$\begin{aligned}
\forall k \in K, \forall p \in P(k) : \quad & y_k + \sum_{(i,j) \in A} y_{(i,j)} q_k \alpha_{ij}^p \leq c_p^k q_k \\
\equiv \quad & -y_k - \sum_{(i,j) \in A} y_{(i,j)} q_k \alpha_{ij}^p + c_p^k q_k \geq 0
\end{aligned}$$

$$\begin{aligned} &\equiv \quad -y_k - \sum_{(i,j)\in A} y_{(i,j)} q_k \alpha_{ij}^p + q_k \sum_{(i,j)\in A} c_{ij}^k \alpha_{ij}^p \geq 0 \\ &\equiv \quad -y_k + q_k \sum_{(i,j)\in A} \alpha_{ij}^p (c_{ij}^k - y_{(i,j)}) \geq 0 \end{aligned}$$

Consequently, for each commodity $k \in K$ all of the following inequalities need to be satisfied :

$$-y_k + q_k \sum_{(i,j)\in A} \alpha_{ij}^p (c_{ij}^k - y_{(i,j)}) \geq 0 \qquad \text{subject to : } p \in P(k)$$

Which is equivalent, for all commodities, to :

$$\begin{aligned} & -y_k + q_k \sum_{(i,j)\in A} \alpha_{ij}^p (c_{ij}^k - y_{(i,j)}) \geq 0 \\ \text{subject to :} \quad & \sum_{(i,j)\in A} \alpha_{ij}^p - \sum_{(j,i)\in A} \alpha_{ji}^p = \gamma_i^k && \forall i \in N \\ & \alpha_{ij}^p \in \{0,1\} && \forall p \in P, \forall (i,j) \in A \end{aligned}$$

where γ_i^k equals 1 if i is the starting node of the commodity k, equals -1 if i is the destination of k and 0 otherwise. And P denotes all paths in the network.

Finally, we can rewrite the pricing problem as Integer Program for each commodity :

$$\begin{aligned} \text{minimize} \quad & -y_k + q_k \sum_{(i,j)\in A} \alpha_{ij} (c_{ij}^k - y_{(i,j)}) && && (4.1) \\ \text{subject to :} \quad & \sum_{(i,j)\in A} \alpha_{ij} - \sum_{(j,i)\in A} \alpha_{ji} = \gamma_i^k && \forall i \in N && (4.2) \\ & \alpha_{ij} \in \{0,1\} && \forall (i,j) \in A && (4.3) \end{aligned}$$

Indeed if the objective function value of the optimal solution is negative, then it implies that the path p described by the arcs α_{ij} violates a constraint in the dual of the MP. Thus the corresponding variable in the MP need to be taken into account to satisfy the constraint in the future. Then if, for each commodity, the objective function value of the optimal solution is positive of equals to 0, thanks to Theorem 1 it implies that we corresponding RMP solution is also optimal to the MP.

The constraints (4.3) and (4.2) aim to produce a set of arcs such that it contains exactly one arc starting from the origin of the commodity k, and one arc ending at the destination of commodity k, moreover the set of arcs must form a path. Thus α_{ij}'s represent all path from the origin of commodity k to its destination. On the other hand, the objective function contains one constant y_k, and the other term is a cost vector modified by the dual variables $y_{(ij)}$. Therefore, this integer program is in fact a shortest path problem with penalized arcs costs. Plenty of polynomial times algorithms exist to solve this problem efficiently. Thus, the pricing problem boils down to a set of $\|K\|$ shortest path problems.

Thus, we obtained a efficient method to verify if the optimal solution of the RMP is also optimal to the MP. Moreover, by adding the variables in the RMP, corresponding to the violated constraints in the pricing problem, we ensure that the constraints will be satisfied in the next iteration of the pricing problem.

However, on the first hand, we need to find a feasible set of variables to the RMP to guarantee the primal feasibility condition of theorem 1. This task might be challenging for large instance. As we only request feasibility, a fast heuristic algorithm can be applied.

Finally the column generation approach can be summarized in Figure 1 :

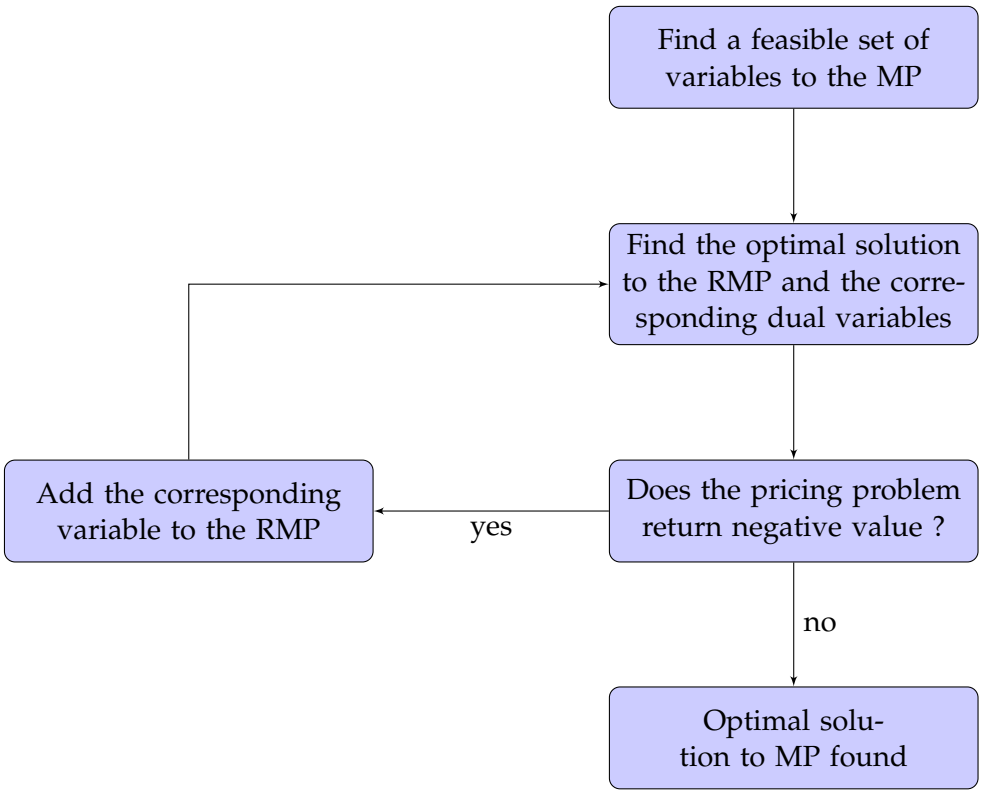

FIGURE 3.1: Sketch of the column generation method

Throughout the experiment part, several variant will be tested, for instance : Do we obtain a better running time if we add more than one variable to the RMP at each step. Moreover many different algorithms can be employed to solve the pricing problem, as well as, to find a feasible promising set of variables in the initialization step. The aim is to find the more efficient column generation variant to solve the MCFP in our network.

Bibliography

[1] C. Barnhart N. Krishnan and P. Vance. *Multicommodity Flow Problems*. In C. A. Floudas and P. M. Pardalos editors Encyclopedia of Optimization pages 2354-2362. Springer US 2009.

[2] J. Lindstrom. *Multicommodity Network Flow - Methods and Application* in Proc. Information Resources Management Association International Conference (IRMA04) Business Process Management Tools and Technologies Track New Orleans Louisiana May 2004 pp. 1-8.

[3] Rodrigo, Niluka, and Lashika Rjapaksha. *Mathematical Model and a Case Study for Multi-Commodity Transportation Problem.* International Journal of Theoretical and Applied Mathematics 4.1 (2018)

[4] Oğuz, Murat, Tolga Bektaş, and Julia A. Bennell. *Multicommodity flows and Benders decomposition for restricted continuous location problems.* European Journal of Operational Research 266.3 (2018): 851-863.

[5] Dorneles, Árton P., Olinto CB de Araújo, and Luciana S. Buriol. *A column generation approach to high school timetabling modeled as a multicommodity flow problem.* European Journal of Operational Research 256.3 (2017): 685-695.

[6] Masri, Hela, Saoussen Krichen, and Adel Guitouni. *A Hybrid Genetic Algorithm for Solving the Unsplittable Multicommodity Flow Problem: The Maritime Surveillance Case.* International Conference on Algorithmic Applications in Management. Springer, Cham, 2014.

[7] Shitrit, Horesh Ben, et al. *Multi-commodity network flow for tracking multiple people.* IEEE transactions on pattern analysis and machine intelligence 36.8 (2014): 1614-1627.

[8] Weibin, D. A. I., Jun Zhang, and S. U. N. Xiaoqian. *On solving multi-commodity flow problems: An experimental evaluation.* Chinese Journal of Aeronautics 30.4 (2017): 1481-1492.

[9] Jean-François Cordeau, François Soumis and Jacques Desrosiers. *Simultaneous assignment of locomotives and cars to passenger trains.* Operations research 49.4 (2001): 531-548.

[10] Borndörfer, Ralf, Martin Grötschel, and Marc E. Pfetsch. *A column-generation approach to line planning in public transport.* Transportation Science 41.1 (2007): 123-132.

[11] Weibin Dai, Jun Zhang, Xiaoqian Sun, and Sebastian Wandelt *Node Dependency in Multi-Commodity Flow Problem with Applications to Transportation Networks.* CICTP 2016. 2016. 1989-2001.