Solving a Freight Railcar Flow Problem Arising in Russia

Ruslan Sadykov, Alexander Lazarev, Vitaliy Shiryaev, Alexey Stratonnikov

Solving a Freight Railcar Flow Problem Arising in Russia

Ruslan Sadykov*[1], Alexander A. Lazarev[2,3], Vitaliy Shiryaev[4], and Alexey Stratonnikov[4]

1 **INRIA Bordeaux – Sud-Ouest,**
351, cours de la Liberation, 33405 Talence, France
Ruslan.Sadykov@inria.fr
2 **Institute of Control Sciences,**
65 Profsoyuznaya street, 117997 Moscow, Russia
lazarev@ipu.ru
3 **National Research University Higher School of Economics,**
20 Myasnitskaya street, 101000 Moscow, Russia
4 **JSC Freight One**
Staraya Basmannaya st., 12 bld. 1, 105064 Moscow, Russia
{ShiryaevVV,StratonnikovAA}@pgkweb.ru

Abstract

We consider a variant of the freight railcar flow problem. In this problem, we need 1) to choose a set of transportation demands between stations in a railroad network, and 2) to fulfill these demands by appropriately routing the set of available railcars, while maximizing the total profit. We formulate this problem as a multi-commodity flow problem in a large space-time graph. Three approaches are proposed to solve the Linear Programming relaxation of this formulation: direct solution by an LP solver, a column generation approach based on the path reformulation, and a "column generation for extended formulations" approach. In the latter, the multi-commodity flow formulation is solved iteratively by dynamic generation of arc flow variables. Three approaches have been tested on a set of real-life instances provided by one of the largest freight rail transportation companies in Russia. Instances with up to 10 millions of arc flow variables were solved within minutes of computational time.

1998 ACM Subject Classification G.1.6 Optimization

Keywords and phrases Freight routing, multi-commodity flow, column generation

Digital Object Identifier 10.4230/OASIcs.ATMOS.2013.55

1 Introduction

In Russia, the activity of forming and scheduling freight trains is separated by a regulation from the activity of managing the fleet of freight railcars. A state company is in charge of the first activity. Freight railcars are owned by several independent companies. Every such company is quite limited in transportation decisions due to the separation of activities. A company which owns a fleet of railcars can only accept or refuse a transportation demand. Then it must assign railcars to accepted demands. In some cases, the company has a possibility to slightly modify the execution date of a demand, which gives more flexibility to the decision process but makes it more complicated.

* Corresponding author

13th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS'13).
Editors: Daniele Frigioni, Sebastian Stiller; pp. 55–67

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Thus, an operational plan of such a company is determined by 1) a set of accepted transportation demands, 2) for each demand, its execution date and the set of cars assigned to it, and 3) empty cars movements to supply each demand. As the company is commercial, a reasonable criterion for the quality of an operational plan is the profit generated by it. The profit is determined by the difference between the price collected for fulfilling transportation demands and the costs paid to the state company for exploiting the railroad network.

In this paper, we study the problem of finding a most profitable operational plan for a company which owns and manages a fleet of railcars. This problem was formulated by the mathematical modeling department of one of the largest such companies in Russia.

For this problem, we are given a railroad network, a set of transportation demands, an initial location of cars of different types. The network data consists of a set of stations, travel times and costs between them. As it was mentioned above, the company does not schedule trains. Thus, actual transportation of loaded and empty railcars is performed by the state company, who charges predetermined costs per trip. Estimated travel times are also determined and applied by the state company. Note that the transfer cost for an empty car depends on the type of product type loaded previously to this car. This way, the state increases attractiveness of the transportation of "socially important" products (for example, coil). This custom is being vanished now, but it is still practiced for some car types.

The objective is, for a given period of time, to choose a set of demands to be met, totally or partially, and, for each car, find a route which includes both loaded and empty transfers.

To solve the problem, we start from an integer multi-commodity flow model, which has been proposed by Stratonnikov and Shiryaev [10]. The time horizon in this model is discretized in periods of one day. This discretization choice is reasonable for Russia, as distances are measured in thousands of kilometers, and the average speed of freight trains is relatively low: about 300 kilometers per day (it tends to decrease further with a saturation of the network).

This model has a very large size, and even solving its Linear Programming (LP) relaxation using modern commercial solvers can take hours of computation time for real-life instances. However, a solution of this LP relaxation allows one to obtain a very tight dual bound for the objective function value. This fact has been also noticed for similar models considered in [4, 7]. Therefore, we concentrate on solving the LP relaxation of this formulation, leaving the problem of obtaining an integer solution out of the scope of the paper. In practice, rounding a fractional solution in a straightforward way allows one to obtain an integer solution with very small gap.

To solve the LP relaxation faster, we devise two variants of the column generation procedure, where columns represent railcar routes or flows of the railcars of the same type. The first variant we tried is the classic Dantzig-Wolfe approach. In the second variant, called "column generation for extended formulations" in [9], columns are disaggregated into individual arc flow variables when added to the master. Thus, the master problem is equivalent to the original multi-commodity flow model, but its variables are generated dynamically. On almost all real-life instances provided by the company, either the first or the second variant of the column generation approach significantly outperformed the solution by a solver of the LP relaxation of the original multi-commodity model, preprocessed by a problem-specific procedure.

To our knowledge, the closest model considered in the literature is the freight car flow problem faced by a Brazilian logistics operator and described by Fukasawa et al. [4]. In this paper, authors proposed a similar integer multi-commodity flow model and solved it using a simple preprocessing and an Mixed Integer Programming (MIP) solver. The main

difference with our model is the availability of a fixed train schedule. In their model cars must be assigned to trains to be transported. In our model, we cannot rely on the train schedule information, as it is very approximative and rarely respected in practice. Instead, we use the normative travel times given by the state company which is in charge of forming and scheduling trains. Additionally, Fukasawa et al. considered loading and unloading times, which we neglect here because they are much smaller than the length of time periods.

Another similar car flow model has been considered by Holmberg et al. [5]. In this model, one searches only for a flow of empty cars, the flow of loaded cars being fixed. Thus, a heuristic iterative procedure is applied to optimize the total flow of cars.

A paper which is related to our research in terms of the solution approach applied is due to Löbel [7], who considered a vehicle scheduling problem arising in public mass transit. This problem is modeled by a multi-commodity flow model formulation, the LP relaxation of which is solved by dynamically generating arc variables, as we do. With this approach, Löbel was able to solve LP relaxations with millions of variables as we do for the freight car flow problem.

2 Problem description

We give a detailed description of the variant of the freight car flow model considered here. The problem is to find a feasible flow of railcars (i.e. a feasible route for each car) that maximizes the profit by meeting a subset of the transportation demands.

The railroad network consists of a set of stations. Travel times and costs are known for each "origin-destination" pair of stations. Times are measured in days and rounded up. The cost for an empty car transfer depend on the type of the latest product this car has transported, as explained in the introduction.

Number of cars, their initial locations and availability dates are known. Cars are divided into types. The type of a car determines types of products which can be loaded on this car. The route of a car consists of a sequence of alternating loaded and empty movements between stations. Cars can wait at stations before and after fulfilling transportation demands. In this case, a charge is applied. Daily rate of this charge depends on the demand before (or after) the waiting period.

Each transportation demand is defined by a (maximum) number of cars compatible with the product that should be taken from an origin station to a destination station. Some demands can be fulfilled partially. In this case, the client communicates the minimum number of cars which should be delivered. Thus, the total number of transported cars for every accepted demand should be between the minimum and maximum number.

The client specifies the availability date of the product and the delivery due date which cannot be exceeded. The demand transportation time is known. This allows us to determine the latest date at which the transportation must start. The profit we gain for meeting the demand depends on the date the transportation of a loaded car starts. In practice, the contract is concluded for transportation of each car separately. Thus the profit we gain for delivering cars with the product of a same demand at a certain date depends linearly on the number of cars. Note that the profit function already takes into account the charges paid for using the railroad network.

We now specify notations for the data of the problem. Following sets are given.

- I — set of stations.
- C — set of car types.
- K — set of product types

- Q — set of demands
- S — set of "sources" which specify initial state of cars.
- T — set of periods (planning horizon).

For each station, $i \in I$ we know sets W_i^1 and W_i^2 of standing daily rates for cars waiting to be loaded and waiting after unloading.

For each demand $q \in Q$ we know:

- $i_q \in I$ — origin station
- $j_q \in I$ — destination station
- $k_q \in K$ — type of product to be transported
- $C_q \subseteq C$ — set of car types, which can be used for this demand
- $n_q^{\max}$ — number of cars needed to fullfil the demand
- $n_q^{\min}$ — minimum number of car needed to partially fullfil the demand
- $r_q \in T$ — demand availability, i.e. the period starting from which the transportation of the product can start
- Δ_q — maximum delay for starting the transportation
- ρ_{qt} — profit from delivery of one car with the product, transportation of which started at period t, $t \in [r_q, r_q + \Delta_q]$
- $d_q \in \mathbb{Z}_+$ — transportation time of the demand
- $w_q^1 \in W_{i_q^1}^1$ — daily standing rate charged for one car waiting before loading the product at origin station
- $w_q^2 \in W_{i_q^2}^2$ — daily standing rate charged for one car waiting after unloading the product at destination station

For each car type $c \in C$, we can obtain set Q_c of demands, which a car of type c can fulfill.

For each source $s \in S$, we are given:

- $\vec{i}_s \in I$ — station where cars are located
- $\vec{c}_s \in C$ — type of cars
- $\vec{r}_s \in T$ — period, starting from which cars can be used
- $\vec{w}_s \in W_{i_s}^2$ — daily standing rate charged for cars
- $\vec{k}_s \in K$ — type of the latest delivered product
- $\vec{n}_s \in \mathbb{N}$ — number of cars in the source

For each car type $c \in C$, we can obtain set of sources $S_c = \{s \in S : \vec{c}_s = c\}$.

Additionally, functions $M(c, i, j, k)$ and $D(c, i, j)$ are given which specify cost and duration of transportation of one empty car of type $c \in C$ from station $i \in I$ to station $j \in I$ under condition, that the type of the latest delivered product is $k \in K$ (for the cost).

3 Mathematical model

We represent movements of cars of each type $c \in C$ by commodity c. For each commodity $c \in C$, we introduce a directed graph $G_c = (V_c, A_c)$. Set V_c of vertices is divided into two subsets V_c^1 and V_c^2 which represent respectively states in which cars stand at a station before being loaded and after being unloaded. A vertex $v_{cit}^{1w} \in V_c^1$ represents stay of cars of type c waiting to be loaded at station $i \in I$ at daily rate $w \in W_i^1$ at period $t \in T$. Flow balance $b(v_{cit}^{1w})$ of this vertex is zero. A vertex $v_{cit}^{2wk} \in V_c^2$ represents stay of cars of type c after being unloaded at station $i \in I$ at daily rate $w \in W_i^2$ at period $t \in T$. Here $k \in K$ is the type of unloaded product. Flow balance $b(v_{cit}^{2wk})$ of this vertex is determined as follows:

$$b(v_{cit}^{2wk}) = \begin{cases} \vec{n}_s, & \exists s \in S_c : \vec{i}_s = i, \vec{r}_s = t, \vec{w}_s = w, \vec{k}_s = k, \\ 0, & \text{otherwise.} \end{cases}$$

Additionally, there is a single terminal vertex with flow balance equal to $-\sum_{s \in S_c} \vec{n}_s$.

There are three types of arcs in A_c: waiting, empty transfer, and loaded transfer arcs.

- A waiting arc $a_{cit}^{\alpha w k}$ represents waiting of cars of type c from period $t \in T$ to $t+1$ at station $i \in I$ at daily rate $w \in W_i^\alpha$ before being loaded ($\alpha = 1$) or after being unloaded ($\alpha = 2$). $k \in K$ is the type of unloaded product in case $\alpha = 2$. This arc goes from vertex $v_{cit}^{\alpha w k}$ to vertex $v_{c,i,t+1}^{\alpha w k}$, or to the terminal vertex if $t+1 \notin T$. Cost of this arc is w.
- An empty transfer arc $a_{cijt}^{w' w'' k}$ represents a transfer of empty cars of type c waiting at station $i \in I$ at daily rate $w' \in W_i^2$ to station $j \in I$ where they will wait at daily rate $w'' \in W_j^1$, such that the type of latest unloaded product is $k \in K$, and transfer starts at period $t \in T$. This arc goes from vertex $v_{cit}^{2w'k}$ to vertex $v_{cjt'}^{1w''}$, or to the terminal vertex if $t' \notin T$, where $t' = t + D(c,i,j)$. Cost of this arc is $M(c,i,j,k)$.
- A loaded transfer arc a_{cqt} represents transportation of the product of demand $q \in Q$ by cars of type c starting at period $t \in T \cap [r_q, r_q + \Delta_q]$. This arc goes from vertex $v_{ci_qt}^{1w_q^1}$ to vertex $v_{c,j_q,t+d_q}^{2w_q^2 k_q}$, or to the terminal vertex if $\{t + d_q\} \notin T$. The cost of this arc is $-\rho_{qt}$.

A small example of graph G_c is depicted in Figure 1. In this example, there is only one "before" vertex and one "after" vertex for each time period and each station. In real-life examples, there are several rows of "before" and "after" vertices for each station.

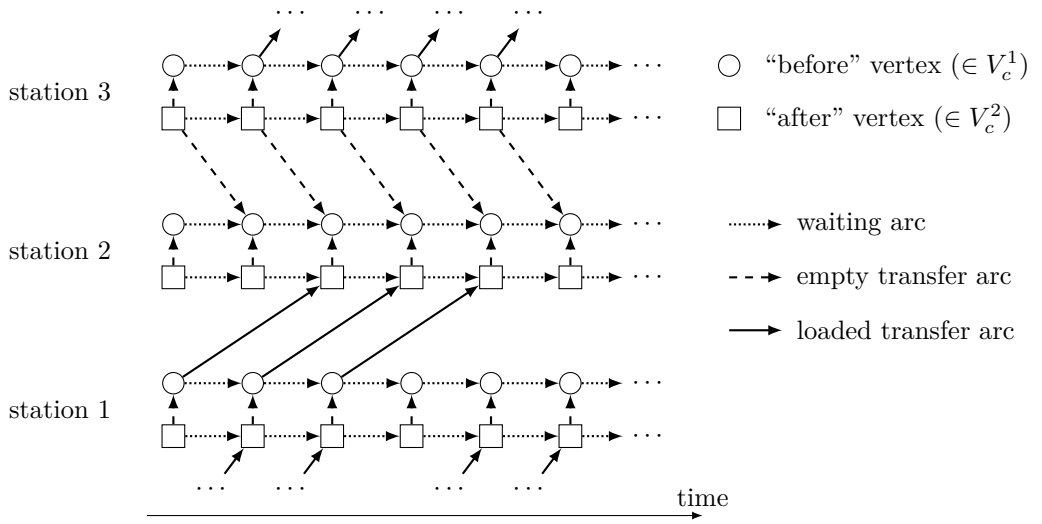


Figure 1 An example of graph G_c

We denote as A_{cq} the set of all loaded transfer acs related to demand $q \in Q_c$: $A_{cq} = \{a_{cq't} \in A_c : q' = q\}$. Also we denote as $\delta^+(v)$ and $\delta^-(v)$ the sets of incoming and outgoing arcs for vertex v_c.

From now on, graph G_c is assumed to be trivially preprocessed: we remove vertices with degree two (replacing appropriately incident arcs), and remove every vertex (together with incident arcs) such that there is no path from any source to it or there is no path from it to the terminal vertex.

For each commodity $c \in C$ and for each arc $a \in A_c$, we define an integer variable x_a which represents the flow size of commodity c along arc a. Cost of arc a is denoted as $g(a)$. Additionally, for each demand $q \in Q$, we define a binary variable y_q which indicates whether demand q is accepted or not.

Now we are able to present a multi-commodity flow formulation (MCF) for the problem.

$$\min \sum_{c\in C}\sum_{a\in A_c} g(a)x_a \tag{1}$$

$$\sum_{c\in C_q}\sum_{a\in A_{cq}} x_a \le n_q^{\max} y_q \quad \forall q \in Q \tag{2}$$

$$\sum_{c\in C_q}\sum_{a\in A_{cq}} x_a \ge n_q^{\min} y_q \quad \forall q \in Q \tag{3}$$

$$\sum_{a\in\delta^-(v)} x_a - \sum_{a\in\delta^+(v)} x_a = b(v) \quad \forall c \in C, v \in V_c \tag{4}$$

$$x_a \in \mathbb{Z}_+ \quad \forall c \in C, a \in V_c \tag{5}$$

$$y_q \in \{0,1\} \quad \forall q \in Q \tag{6}$$

Constraints (2) and (3) specify that the number of cars assigned to accepted demand q should be between $n_q^{\min}$ and $n_q^{\max}$. Constraints (4) are flow conservation constraints for each commodity. As formulation (MCF) generalizes the standard multi-commodity flow problem (where variables y are fixed to one), our problem is NP-hard in the strong sense.

The formulation (MCF) was tested in [10]. The main difficulty was to solve the LP relaxation of the problem, which we denote as $(MCF)_{LP}$. Even after non-trivial problem-specific preprocessing, solution time of $(MCF)_{LP}$ for typical real-life instances by a modern LP solver on a modern computer is more than one hour. Therefore, our research is concentrated on accelerating the resolution of $(MCF)_{LP}$.

4 A column generation approach

A classic approach to solve multi-commodity flow formulations is to apply the column generation procedure. Instead of working with arc variables, one uses variables of one the following types.

- A "path variable" determines the flow size of a commodity along a path from one of the source nodes to a sink node of this commodity.
- A "tree variable" specifies whether the flow of a certain size from a single source of a commodity goes along a fixed (directed) tree with fixed flow sizes along its arcs. Leaves of this tree are sink nodes of this commodity.
- A "flow enumeration variable" specifies whether the flow of a single commodity is equal to a fixed flow.

We now reformulate (MCF) using path variables, and then using flow enumeration variables. The reformulation which uses the tree variables was not tried, as there is only one sink per commodity.

4.1 Path reformulation

For each commodity $c \in C$ and each source $s \in S_c$, we denote as P_s the set of paths going from the corresponding source vertex in V_c to the terminal vertex of graph G_c. Each such path represents a route for cars originating at source s. For a path $p \in P_s$, we introduce a variable λ_p which determines the flow size along path p (or number of cars taking this route). Let A_p^{path} be the set of arcs taken by a path $p \in P_s$ and g_p^{path} be the cost of the path: $g_p = \sum_{a\in A_p^{path}} g(a)$. Let also Q_p^{path} be the set of demands "covered" by path p. The path reformulation (PTH) of (MCF) is the following.

$$\min \sum_{c\in C}\sum_{s\in S_c}\sum_{p\in P_s} g_p^{path}\lambda_p \tag{7}$$

$$\sum_{c\in C_q}\sum_{s\in S_c}\sum_{p\in P_s:\ q\in Q_p^{path}} \lambda_a \le n_q^{\max} y_q \quad \forall q\in Q \tag{8}$$

$$\sum_{c\in C_q}\sum_{s\in S_c}\sum_{p\in P_s:\ q\in Q_p^{path}} \lambda_a \ge n_q^{\min} y_q \quad \forall q\in Q \tag{9}$$

$$\sum_{p\in P_s} \lambda_p = \vec{n}_s \qquad \forall c\in C, s\in S_c \tag{10}$$

$$\lambda_p \in \mathbb{Z}_+ \qquad \forall c\in C, s\in S_c, p\in P_s$$

$$y_q \in \{0,1\} \qquad \forall q\in Q$$

Constraints (8) and (9) are rewritten constraints (2) and (3). Constraints (10) guarantee that a route is assigned to every car in each source.

In order to solve the LP relaxation $(PTH)_{LP}$ of formulation (PTH), we apply the column generation procedure. On each iteration of it, the formulation $(PTH)_{LP}$ with a restricted number of variables λ (which we will call the restricted master) is solved, and optimal primal and dual solutions are obtained. Let $\pi^{\max}$, $\pi^{\min}$, and μ be the vectors of optimal dual solution values corresponding to constraints (8), (9), and (10). Then the pricing problem is solved which determines whether there exists a variable with a negative reduced cost absent from the restricted master. The reduced cost $\bar{g}_p^{path}$ of a variable λ_p, $p\in P_s$, $s\in S_c$, $c\in C$ is computed as

$$\bar{g}_p^{path} = \sum_{a\in A_p^{path}} g(a) + \sum_{q\in Q_p^{path}} (\pi_q^{\max} - \pi_q^{\min}) - \mu_s. \tag{11}$$

The problem of finding a variable λ with the minimum reduced cost can be solved by a sequence of shortest path problems between each source $s\in S_c$ and the terminal vertex for every commodity $c\in C$. To accelerate the solution of the pricing problem, instead of searching the shortest path separately for each source, in each graph G_c, we can find a minimum cost in-tree to the terminal vertex from every source in S_c. As directed graphs G_c are acyclic (each arc except those from V_c^2 to V_c^1 induces a time increase), the complexity of this procedure is linear in the number of arcs for each graph G_c.

This procedure is quite fast, but its disadvantage consists in significant demand "overcovering". This means that many generated paths contain arcs corresponding to same demands, i.e. much more cars are assigned to these demands than needed. This has a bad impact on the convergence of column generation.

Therefore, we developed an iterative procedure which heuristically constructs a solution to the original problem with demand profits modified by the current dual solution values. Then all paths which constitute this solution are added to the master. On each iteration, we search for a shortest path tree and then remove covered demands and cars assigned to them for the next iteration. The heuristic stops when either all demands are covered, or all cars are assigned, or maximum number of iterations is reached. The latter is a parameter which we denote as `nbPricIter`. This procedure for commodity $c\in C$ is formally presented in Algorithm 1. This procedure can be viewed as a heuristic for generating additional paths to be added to the master for the convergence acceleration.

Algorithm 1: Path generation iterative pricing procedure for graph G_c

foreach *demand* $q \in Q_c$ **do** $uncovCars_q \leftarrow n_q^{\max}$;
foreach *source* $s \in S_c$ **do** $remCars_s \leftarrow \vec{n}_s$;
$iter \leftarrow 0$;
repeat
  Find an in-tree to the terminal from sources $s \in S_c$, $remCars_s > 0$;
  Sort paths p in this tree by non-decreasing of their reduced cost $\bar{g}_p^{path}$;
  foreach *path* p *in this order* **do**
    $minCars \leftarrow \min\{uncovCars_q \mid q \in Q_p^{path}\}$;
    if $\bar{g}_p < 0$ **and** $minCars > 0$ **then**
      Add variable λ_p to the restricted master;
      $s \leftarrow$ the source of p;
      $remCars_s \leftarrow remCars_s - \min\{remCars_s, minCars\}$;
      foreach $q \in Q_p^{path}$ **do**
        $uncovCars_q \leftarrow uncovCars_q - \min\{remCars_s, minCars\}$;
  $iter \leftarrow iter + 1$;
until $uncovCars_q = 0$, $\forall q \in Q_c$, **or** $remCars_s = 0$, $\forall s \in S_c$, **or** $iter =$ `nbPricIter`;

4.2 Flow enumeration reformulation

Each car type defines a commodity $c \in C$. We define by F_c the set of all fixed solutions (fixed flows) for commodity c. For a flow $f \in F_c$, we introduce a binary variable ω_f which specifies whether cars of type c are routed according to flow f or not. Let f_a be the size of flow f along arc $a \in A_c$ and g_f^{flow} be the cost of the flow: $g_f^{flow} = \sum_{a \in A_c} f_a \cdot g(a)$. The commodity reformulation (FEN) of (MCF) is the following

$$\min \sum_{c \in C} \sum_{f \in F_s} g_f^{flow} \omega_f \tag{12}$$

$$\sum_{c \in C_q} \sum_{f \in F_c} \sum_{a \in A_{cq}} f_a \omega_f \le n_q^{\max} y_q \quad \forall q \in Q \tag{13}$$

$$\sum_{c \in C_q} \sum_{f \in F_c} \sum_{a \in A_{cq}} f_a \omega_f \ge n_q^{\min} y_q \quad \forall q \in Q \tag{14}$$

$$\sum_{f \in F_c} \omega_f = 1 \qquad \forall c \in C \tag{15}$$

$$\omega_f \in \{0,1\} \qquad \forall c \in C, f \in F_c$$

$$y_q \in \{0,1\} \qquad \forall q \in Q$$

Constraints (13) and (14) are rewritten constraints (2) and (3). Constraints (15) guarantee that exactly one flow is assigned to commodity $c \in C$.

LP relaxation $(FEN)_{LP}$ of formulation (FEN) can also be solved by the column generation procedure. The pricing problem here decomposes into the minimum cost flow problems for each commodity $c \in C$.

Our computational results showed that, solving $(FEN)_{LP}$ by column generation is not practical due to convergence problems. However, in the next section, we present a modification of this approach, which is computationally much more efficient.

5 A "column generation for extended formulations" approach

We adapt here the hybrid approach, reviewed under the name "column generation for extended formulations" (CGEF) in [9]. The idea is to solve formulation $(MCF)_{LP}$ iteratively by generating arc flow variables dynamically. On each iteration, we generate columns (single commodity flows) for formulation $(FEN)_{LP}$, and translate them into arc flow variables which are added to formulation $(MCF)_{LP}$.

In the CGEF approach, on each iteration, we first solve the formulation $(MCF)_{LP}$ with a restricted number of variables x. We will also call this formulation the restricted master. Then, we verify whether there are variables x with a negative reduced cost absent from the restricted master. However, we do not do it by enumeration, but by using the same pricing problem as in classic column generation for solving formulation $(FEN)_{LP}$. If a pricing problem solution with a negative reduced cost is found, we add to the restricted master variables x which are positive in this solution (some of them can be already in the restricted master).

As a consequence of the theorem proved in [9], we know that, if an arc flow variable x is absent from the restricted master and has a negative reduced cost in the current solution of the restricted master, there exists a pricing problem solution with a negative reduced cost where this variable is positive. Therefore, if there are no pricing problem solutions with a negative reduced cost, the current solution of the restricted master is optimal for $(MCF)_{LP}$.

When solving formulation $(MCF)_{LP}$ by the CGEF approach, the pricing problem is decomposed to a sequence of min-reduced-cost flow problems for each commodity $c \in C$, as in the column generation approach for solving the commodity reformulation $(FEN)_{LP}$. Let $\pi_q^{\max}$ and $\pi_q^{\min}$ be the vectors of optimal dual solution values corresponding to constraints (2), (3). Then, the reduced cost $\bar{g}_f^{flow}$ of a flow $f \in F_c$ is computed as

$$\bar{g}_f^{flow} = \sum_{a \in A_c} f_a \cdot g(a) + \sum_{q \in Q_c} \sum_{a \in A_{cq}} f_a \cdot (\pi_q^{\max} - \pi_q^{\min}). \tag{16}$$

Note that dual values corresponding to flow conservation constraints (4) are not taken into account when solving the pricing problem, as explained in [9]: this follows from the fact that these constraints are satisfied by the pricing problem solution.

On each iteration, for each commodity $c \in C$, the pricing problem generates a flow $f \in F_c$ with the minimum reduced cost. If this cost is negative, variables x corresponding to arcs on which flow f is positive, are added to the restricted master. Otherwise, the current solution of the restricted master is optimal for $(MCF)_{LP}$, and we stop.

6 Numerical results

The test instances were provided to us by the mathematical modeling departement of *JSC Freight One*, which is one of the largest freight rail transportation companies in Russia.

We have numerically tested the following three approaches for solving formulation $(MCF)_{LP}$ on these real-life instances.

1. Direct solution of $(MCF)_{LP}$ by the *Clp* LP solver [1]. Before applying the LP solver the formulation is preprocessed by a non-trivial problem specific procedure. This procedure is not public and it was not available to us. Moreover, the open-source solver *Clp* was specifically modified to better tackle formulation $(MCF)_{LP}$. Thus, this approach was applied inside the company. We tried to solve $(MCF)_{LP}$ with only trivial preprocessing by the default version of both LP solvers *Clp* and *Cplex* [2], but our solution times on a

comparable computer were significantly larger. Therefore, for the comparison, we use the solution times communicated to us by the company. We denote this approach as DIRECT.

2. Solution of the path reformulation $(PTH)_{LP}$ by column generation. The pricing problem here is solved by the iterative shortest path tree procedure presented in Algorithm 1. The effect of applying the iterative pricing procedure was significant. After preliminary tests, the parameter `nbPricIter` was set to 5. For better convergence of the column generation procedure, the following improvements are applied.
   - The restricted master is initialized with paths according to which cars stay at their initial locations during all the planning horizon.
   - Stabilization by dual prices smoothing [8] is applied.
   - The restricted master is cleaned up every 10 iterations by deleting all columns with a positive reduced cost.

   The column generation approach was implemented in C++ programming language using the *BaPCod* library [11] and *Cplex* as LP solver. We denote this approach as COLGEN.
3. The solution of $(MCF)_{LP}$ by the CGEF approach. The pricing problem here is solved using the minimum cost flow solver *Lemon* [3]. To improve convergence of the algorithm, the master is initialized with the full set of waiting arcs. Note that in distinction to DIRECT only a trivial procedure was applied to preprocess the formulation. This approach was also implemented in the same manner as the previous one. We denote this approach as COLGENEF.

The approach DIRECT was run on a computer with a processor Intel Xeon X5677 3.47 GHz, the approaches COLGEN and COLGENEF were run on a computer with a processor Intel Xeon X5460 3.16 GHz in a single thread mode.

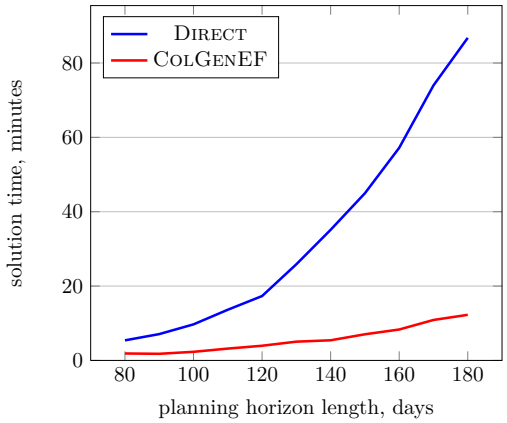
The first test set consists of 3 instances. Characteristics of these instances and results for 3 tested approaches for these instances are presented in Table 1.

The difference in performance of the approaches DIRECT and COLGENEF on instances `x3` and `x3double` can be explained by the problem specific preprocessing. Although we are not aware of preprocessing details, we know that it is based on similarities between car types. For instance `5k0711q` in which there is only one car type, difference between two approaches is much smaller. Note that this instance has been artificially created from the real-life one by merging car types into one.

The second test set consists of instances with larger planning horizon length. These instances contain 1'025 stations, up to 6'800 demands, 11 car types, 12'651 cars, and 8'232

Table 1 The first set of instances: characteristics and numerical results.

Instance name	`x3`	`x3double`	`5k0711q`
Number of stations	371	371	1'900
Number of demands	1'684	3'368	7'424
Number of car types	17	17	1
Number of cars	1'013	1'013	15'008
Number of sources	791	791	11'215
Time horizon, days	37	74	35
Total number of vertices, thousands	62	152	22
Total number of arcs, thousands	794	2'846	1'843
Solution time for DIRECT	20s	1h34m	55s
Solution time for COLGEN	22s	7m53s	8m59s
Solution time for COLGENEF	3m55s	>2h	43s



Horizon	Direct	ColGenEF
80	5m24s	1m52s
90	7m05s	1m47s
100	9m42s	2m19s
110	13m38s	3m11s
120	17m19s	3m57s
130	25m52s	5m03s
140	35m08s	5m25s
150	44m58s	7m02s
160	57m11s	8m19s
170	1h13m58s	10m53s
180	1h26m46s	12m16s

Figure 2 Solution times for test instances with larger planning horizon length.

sources. The planning horizon length is from 80 to 180 days. The graph $\cup_{c \in C} G_c$ for the largest instance contains about 300 thousands nodes and 10 millions arcs. For these instances, the two best approaches are Direct and ColGenEF. The comparison of their solution times is presented in Figure 2. The approach ColGen is two to three times slower than Direct.

An important observation is that the algorithm ColGenEF generally converges in less than 10 iterations (and always in less than 15 iterations). The restricted master on the final iteration contains only about 3% of the arc flow variables of formulation (MCF).

7 Conclusions and perspectives

We have formulated a freight car flow problem variant as a multi-commodity flow problem in a large space-time graph. Three approaches for solving the LP relaxation of this formulation has been tested on a set of real-life instances provided by one of the largest freight rail transportation companies in Russia.

Computational results show that the classic column generation approach is the best for instances with relatively small number of sources (different initial locations of cars). For other instances, approaches based on the multi-commodity formulation produce better results. Problem-specific preprocessing based on similarities between car types is an important ingredient, which allows a modern LP solver to tackle quite efficiently instances with a relatively small time horizon length. The best approach for instances with larger time horizon length is solving the multi-commodity formulation by dynamically generating arc flow variables (the "column generation for extended formulations" approach). Even without applying problem-specific preprocessing, it outperforms the direct resolution approach, and this advantage increases with the increase of the time horizon length. It is likely that the combination of the CGEF approach with problem-specific preprocessing will produce even better results. Such a combination could be used to produce better solutions by shortening the time period length.

The most important research direction for the future is to obtain integer solutions for the problem either by a branch-and-bound (or branch-and-price) methods or by heuristics based on the fractional solution obtained by the approaches proposed here. Simple heuristics can be based on rounding. Experiments conducted inside the company show that this approach already produces good results, as the LP relaxation solutions are almost integer. Column

generation based heuristics of the "diving" type [6] are likely to produce better results.

Note that the problem studied in this paper does not incorporate some practical considerations. Some of them can be easily modeled by enlarging the space-time graph used in the current approaches. These considerations are the following.

- Waiting rates for cars are generally not linear but progressive. When a car arrives to a station, the owner should pay an initial rate for every standing day. After a certain period of time this rate increases. This increase can happen several times. In other words, the daily waiting rate in every station is a non-decreasing function of the current stay duration.
- There is a set of special stations where cars can stay for a lower rate (although there is a fix rate for putting cars there). A car can go to one of this stations between fulfilling two demands in order to pay less for waiting. Note that it is advantageous to use these stations only if a sufficiently long planning horizon is considered.
- There is a compatibility function between two consecutive types of loaded products. This means that even if a car type is suitable for a demand, a car of this type may not be able to fulfill it because the type of previously loaded product is incompatible with the product type of the demand. For example, the petrol cannot be loaded to a car after the oil, but the oil can be put after the petrol. Also, there are special stations where cars can be washed for a fee. After washing a car, the type of product previously loaded to it is "nullified".
- When a demand is not selected, a penalty payment may be due.

There exists however a problem extension which cannot be solved by the approaches presented. As mentioned in the introduction, transportation times and costs between each pair of stations are communicated by the state company which is in charge of forming and scheduling trains. In this paper, we considered that they depend only on the origin and destination stations. However, in practice usually they also depend on the size of the group of cars sent together. The larger this group is, the faster and with smaller unitary cost it will be delivered to the destination. It seems that exact solution of real-life instances of this extension of the problem is out of reach of modern optimization tools.

References

1 Clp – COIN-OR Linear Programming Solver. `https://projects.coin-or.org/Clp`.

2 IBM ILOG CPLEX Optimizer. `http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/`.

3 LEMON Graph Library. `https://lemon.cs.elte.hu/trac/lemon`.

4 Ricardo Fukasawa, Marcus Poggi de Aragão, Oscar Porto, and Eduardo Uchoa. Solving the freight car flow problem to optimality. *Electronic Notes in Theoretical Computer Science*, 66(6):42–52, 2002. ATMOS 2002.

5 Kaj Holmberg, Martin Joborn, and Jan T. Lundgren. Improved empty freight car distribution. *Transportation Science*, 32(2):163–173, 1998.

6 Cédric Joncour, Sophie Michel, Ruslan Sadykov, Dmitry Sverdlov, and François Vanderbeck. Column generation based primal heuristics. *Electronic Notes in Discrete Mathematics*, 36:695–702, 2010.

7 Andreas Löbel. Vehicle scheduling in public transit and lagrangean pricing. *Management Science*, 44(12):1637–1649, 1998.

8 Artur Pessoa, Ruslan Sadykov, Eduardo Uchoa, and Francois Vanderbeck. In-out separation and column generation stabilization by dual price smoothing. In *12th International Symposium on Experimental Algorithms*, volume 7933 of *Lecture Notes in Computer Science*, pages 354–365. 2013.

9 Ruslan Sadykov and François Vanderbeck. Column generation for extended formulations. *EURO Journal on Computational Optimization*, 1(1-2):81–115, 2013.

10 Alexey Stratonnikov and Vitaly Shiryaev. A large-scale linear programming formulation for railcars flow management (in Russian). In *Fifth Russian conference on Optimization Problems and Economic Applications*, Omsk, Russia, July 2012.

11 François Vanderbeck. BaPCod — a generic Branch-And-Price Code. `https://wiki.bordeaux.inria.fr/realopt/pmwiki.php/Project/BaPCod`.