The Prominence of Stabilization Techniques in Column Generation: the case of Freight Transportation

R. Sadykov[1,2], A. A. Lazarev[3,4,5,6], A. Pessoa[7], E. Uchoa[7] and F. Vanderbeck[2,1]

[1] INRIA Bordeaux, team RealOpt, 200 avenue de la Vielle Tour, Talence, France
Ruslan.Sadykov@inria.fr
[2] Université Bordeaux, Institut de Mathématique de Bordeaux
351 cours de la Libération, 33405 Talence, France
fv@math.u-bordeaux1.fr
[3] Institute of Control Sciences, 65 Profsoyuznaya street, 117997 Moscow, Russia
lazarev@ipu.ru
[4] National Research University Higher School of Economics, Moscow, Russia
[5] Lomonosov Moscow State University, Moscow, Russia
[6] Institute of Physics and Technology State University, Moscow, Russia
[7] LOGIS, Universidade Federal Fluminense, Niteroi, Brazil
{artur.a.pessoa, eduardo.uchoa}@gmail.com

Keywords: *Column Generation, Stabilization, Freight Transportation*

1 Introduction

Routing and logistics applications are often viewed as intractable for exact optimization tools. Although such problems are naturally suited for a decomposition approach, branch-and-price-and-cut algorithms of the literature typically do not scale to the size of real-life instances. Some recent progress in stabilization techniques amongst other advances (such as diving heuristics, strong branching, and the combination with cutting plane approaches) generate new ambitions for column generation approach in solving approximately very large scale instances. Let us for instance point to the new benchmarks for the Capacitated Vehicle Routing Problem (CVRP) in [2]. This paper illustrates this trend, showing exact results for freight transportation instances of a scale never considered before. Our column generation algorithm yields dual bounds and serves as the core procedure for a primal heuristic. The overal procedure is quite competitive in great part due to the convergence speed-ups resulting from efficient stabilization schemes. It typically provides optimal solutions as primal and dual bounds tend to be equal. The very large scale freight transportation instances (with up to 1,025 stations, 5,300 demands, and 12,651 rail cars) were submitted to us by our Russian partner Freight-One.

2 The freight transportation application

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 can 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.

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 executing transportation demands and the costs paid to the state company for exploiting the railroad network.

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 for each "origin-destination" pair of stations are also determined and applied by the state company. Times are measured in days and are rounded up. The number of cars, as well as their initial locations and availability dates are known. Cars are divided into types. The type $c \in C$ of a car determines the types of products which can be loaded on this car. The route of a car consists of a sequence alternating loaded and empty movements at prescribed stations. Cars can wait at stations before and after executing transportation demands. In this case, a charge is applied at some daily rate.

Each transportation demand $q \in Q$ is defined by a number n_q of cars and an origin-destination pair of stations. Let C_q be the set of car types which can fulfil demand q. 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 for partially meeting the demand $q \in Q$ depends on the number of cars provided (at most equal to n_q) and the dates of transportation. We emphasize that the transportation contract is concluded on a "per car" basis; thus, the profit for delivering cars with the product of a given demand at a certain date depends linearly on the number of cars. The profit function already takes into account the charges paid for using the railroad network. Further details on the application can be found in [4].

The problem can be modeled as an integer multi-commodity flow, each commodity being associated with a type $c \in C$ flowing in a large directed acyclic time-space graph $G_c = (V_c, A_c)$. Set $A_{cq} \subset A_c$ of arcs corresponds to performing a transportation demand $q \in Q$ by a car of type c ($A_{cq} = \emptyset$ if $c \notin C_q$). Arcs in $A_c \setminus \bigcap_{q:c \in C_q} A_{cq}$ correspond to empty car movements and waiting of cars of type c. The flow balance $b(v)$ is negative in vertices $v \in V_c$ that define initial positions of cars of type c; while the flow balance $b(v_t^c)$ is positive in one artificially introduced terminal vertex v_t^c and equal to the number of cars of type c; and the flow balance of all remaining arcs in V_c is zero. An integer variable x_a represents the flow on arc $a \in \bigcup_{c \in C} A_c$. Let g_a be the profit of unitary flow on arc a. This profit can be negative. The formulation is then

$$\max \sum_{c \in C} \sum_{a \in A_c} g_a x_a \tag{1}$$

$$\sum_{c \in C_q} \sum_{a \in A_{cq}} x_a \leq n_q \qquad \forall q \in Q \tag{2}$$

$$\sum_{a \in \delta^-(v)} x_a - \sum_{a \in \delta^+(v)} x_a = b(v) \quad \forall c \in C, v \in V_c \tag{3}$$

$$x_a \in \mathbb{Z}_+ \qquad \forall c \in C, a \in A_c. \tag{4}$$

Constraints (2) specify that the number of cars assigned to demand q should not exceed n_q. Constraints (3) are flow conservation constraints for each commodity.

A classic approach for solving the multi-commodity flow problem is to reformulate it using path variables. In our case, each path variable corresponds to a route taken by a car. The linear relaxation of this reformulation with path variables is solved using a column generation algorithm. However, such method for solving the problem suffers from slow convergence.

3 Stabilization Techniques & Column Generation Strategy

The column generation procedure does suffer several drawbacks: *dual oscillations*, *tailing-off effect*, *primal degeneracy*. Stabilization techniques are essential to address these drawbacks. The most standard stabilization are linear programming based approaches such as piece-wise penalty function approaches and so-called dual price smoothing techniques. In [3], we revisited those techniques to adjust their parameters dynamically for a greater efficiency. The resulting speed-ups allows one to handle much larger instance using column generation.

In this paper, we combine these two stabilization techniques: smoothing of dual values and adding a piece-wise function in the dual space which penalizes a deviation from the current best dual solution. These generic stabilization schemes to reduce the number of iterations are combined with a mutli-column generation strategy attempting to generate complementary columns at each iteration. In the specific context of this application, this scheme is implemented in the subproblem: we do not search for a shortest path for each car individually, but for a tree of shortest paths for all cars of the same type. More importantly, the search for a shortest tree is repeated several times removing the demand already covered at previous stages (as explained in more details in [4]).

4 Diving Heuristic

To obtain a feasible solution for the problem, we combine stabilized column generation with a diving heuristic as proposed in [1]. In short, after solving the linear relaxation, the variable with the maximum fractional value is rounded to the next integer, and the residual problem is resolved again by column generation. This rounding continues until an integer solution is found or until the problem become infeasible.

5 Benchmark results

We applied our approach on six real-life freight car flow problem instances provided to us by our partner company. These instances have different sizes depending on the length of the planning horizon (from 40 to 140 days). These instances contain 1,025 stations, up to 5,300 demands, 11 car types (or commodities), and 12,651 cars. The multi-commodity time-space graph, $\bigcup_{c \in C} G_c$, contains up to about 230,000 nodes and 7,500,000 arcs. Using our diving heuristic, we were able to obtain optimal solutions to all instances. Indeed, for all instances, the optimal solution value is equal to the lower bound value provided by the column generation.

In Table 1, we compare solution times (in seconds) of the diving heuristic with the time needed by Cplex 12.6 to solve IP formulation (1)–(4). There, we give, the instance size (length of the planning horizon), solution time taken by non-stabilized column generation (T_{CG}), column generation with smoothing stabilization (T_{smooth}), column generation with penalty function stabilization (T_{pen}), column generation with combined stabilization (T_{comb}), total solution time of our algorithm including the diving heuristic, and the time taken by Cplex to solve the IP. Note that we excluded time needed for reading the data and for the generation of the formulation (which is larger for the arc formulation submitted to Cplex).

size	T_{CG}	T_{smooth}	T_{pen}	T_{comb}	T_{total}	T_{Cplex}
40	23	19	32	21	22	51
60	117	73	155	95	100	111
80	570	234	178	114	145	245
100	2481	607	278	152	211	408
120	8947	1465	410	213	344	633
140	28884	3069	756	338	377	1127

Table 1: Comparision of solution times

As it can be seen from Table 1, non-stabilized column generation does not scaled up to instances with longer time horizon. Each stabilization technique used separately accelerates the convergence. But the best acceleration is achieved when both techniques are used together. This “double” stabilization coupled with the diving heuristic allows us to solve to optimality all the instances, doing so considerably faster than the Cplex MIP solver.

References

[1] 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.

[2] Diego Pecin, Artur Pessoa, Marcus Poggi, and Eduardo Uchoa. Improved branch-cut-and-price for capacitated vehicle routing. In Jon Lee and Jens Vygen, editors, *Integer Programming and Combinatorial Optimization*, volume 8494 of *Lecture Notes in Computer Science*, pages 393–403. Springer International Publishing, 2014.

[3] Artur Pessoa, Ruslan Sadykov, Eduardo Uchoa, and François Vanderbeck. Automation and combination of linear-programming based stabilization techniques in column generation. Technical Report hal-01077984, HAL Inria, https://hal.inria.fr/hal-01077984, 2014.

[4] Ruslan Sadykov, Alexander A. Lazarev, Vitaliy Shiryaev, and Alexey Stratonnikov. Solving a freight railcar flow problem arising in russia. In Daniele Frigioni and Sebastian Stiller, editors, *13th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems*, volume 33 of *OpenAccess Series in Informatics (OASIcs)*, pages 55–67, Dagstuhl, Germany, 2013. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.