Rail Transport Control by Combinatorial Optimization Approach

Network Flow Algorithms Applied in Traction Equipment Management System

V.G. Matyukhin, A.B. Shabunin
Research & Design Institute for Information Technology,
Signaling and Telecommunications on Railway Transport
(JSC NIIAS)
Moscow, Russia
a.shabunin@vniias.ru

N.A. Kuznetsov
Moscow Institute of Physics and Technology (State University)
MIPT
Moscow, Russia

A.K. Takmazian
Research Institute of Mechanics
Lomonosov Moscow State University
Moscow, Russia
takmaz@imec.msu.ru

Abstract—**Railroad traction equipment (locomotives) matching to freight trains in absence-of-fixed-schedule situation is automated within Eastern Operational Domain (EOD) of Russian Railways. Graph (network flow) optimization methods are used to find globally optimal matching in time-space domain. Solution is being found in real time for real life data flow size.**

***Index Terms*— freight trains, locomotive-to-train assignment, scheduling problem, alternative graph, minimum cost flow, real-time calculations**

I. Introduction

Problem of traction resource management in railroad traffic has long history, perhaps as long as the rail transportation itself. With contemporary intensification of traffic this problem becomes more and more crucial in respect of cargo in-time shipment and optimal utilization of the traction resources themselves. The problem postulates as follows: at the given railroad net for given traffic size (i.e. set of trains) and fleet of locomotives find the best work schedule for each locomotive so every train will be hauled with minimum delay and total work time of the fleet will be maximal. Railroad net is a connected graph with stations as vertices and connecting road blocks as edges. Each train has its prescribed route (ordered list of stations to pass) and time when it's ready to depart from initial station.

The problem stated is being faced by every railway traffic operator in the situation when there's no pre-calculated schedule for any arbitrary train, and no traction resources (locomotives and foot-plate staff) pre-matched to the corresponding train route segments. The aim to develop an *automated* system for traction resource management was formulated within the course of works on the Intellectual Control System on Railway Transportations (ISUZhT) for the Freight Transportations on the Eastern Operation Domain (EOD) of Russian Railways [[1], [2], [3]]. The approximate number of entities involved is about 2000 freight trains every day and 2500 locomotives [2]; geographic scale of the EOD is more than 6000 kilometers from Krasnoyarsk to Vladivostok.

Technology specifics of the problem are that each locomotive has its confined operational domain so the train on its route has to be hauled by different locomotives in turns, changing loco at every boundary point between domains. Another feature is that every train has its specific weight and every loco has definite load-carrying capacity for each road block within its operational domain. Thus loco's capacity at each road block has to exceed train's weight for loco to haul the train.

Previous attempt [3] of solving the problem used the so-called Assignment Problem (AP) approach: each train route was consecutively projected onto locomotive domain set, obtaining a set of route segments ("loco slots") each wholly belonging to one loco domain. The problem was solved separately in each loco domain: one tried to match each loco slot to a real loco available at this domain, in some optimal sense using a real-valued utility function defined in train-loco pair space. To satisfy the AP formalism requirements one has to partition time range to intervals of some heuristic duration, and then solve the problem within each interval consequently. Interval duration was adjusted to average loco slot passing time, so that after having matched locos to slots in one time interval, one might expect the locos to disengage (*en masse*) and become available at the next time step. Auction method developed in [4] was invoked to efficiently solve the Assignment Problem. This method exploits inherent property of AP as a special case of Optimal Network Flow Problem (ONFP), and the well-known dualism between linear programming problems (a class to which ONFP pertains). Auctions method utilizes special relaxation condition on dual set of variables (*prices*) called *complementary slackness condition*. The method provides very fast convergence to globally suboptimal solution. Calculations showed good speed properties: 10 min. approx. for the real size of EOD freight traffic (to compare, if Hungarian algorithm for the simplex-method was used instead, calculations wouldn't finish in several hours' time).

Nevertheless, AP approach for the problem discussed possesses basic deficiency which totally dooms its applicability: though it finds global optimum at each time step, the algorithm is greedy in time dimension. Thus the best solution at first time step gradually (and quickly) deteriorates from step to step and at the end of time range becomes too much insufficient. So a different approach is needed. It has to find optimal solution not only for every time instant, but for the whole time span of the task. A formalism which not only allows find the total optimum for the task, but implements algorithms that solve the problem even much faster than in [3] is constructed in the present work.

The basic new principle of the present work is to solve the problem at the whole timespan at once, without partitioning it to time intervals. The key idea is to use the same graph optimization approach but in its more general form: instead of artificially cutting timespan in order to obtain set of APs, a general case of ONFP known as Minimum Cost Flow Problem (MCFP) is considered. Obviously, constructing such formalism one has to ensure that time dimension is essentially represented in the resulting graph structure. This provided, one may anticipate graph optimization methods to "do the work for him": i.e. globally optimize the task in the timespan as well. Detailed formulation of this new approach will be presented in the following section.

Assuming the MCFP is correctly and adequately formulated, one faces need of the efficient methods of solving it. Considering special case called Maximum Flow Problem (MFP) one may refer to classical works of Ford & Fulkerson, Edmonds & Karp, Dinic, Karzanov, Goldberg & Tarjan. Consistent review on the approach development done by each of them may be found in [5]. We mention briefly that they have had gradually improved its algorithmic simplicity and convergence speed. Whence first work was based on finding chains of мaximum flow; next one optimized its convergence by special order of chain search; third one endowed even more orderliness by grouping chains by their lengths, which was used in the fourth one (Karzanov's) to finally come to a notion of pre-flow, which is a relaxed form of flow, free from condition of zero divergence in vertices. Thus, it showed the best calculation speed amongst chain-search based methods. After the discovery of the pre-flow idea there was one step to convert the whole approach from chain-search to pre-flow-push method first discovered by Andrey Vladislavovich (Andrew) Goldberg (in his MIT PhD thesis cited in the reference above). Our present work problem is solved basing on this highly efficient method applied to MCFP.

II. Traction Resources Matching Problem

General task formulation is as follows: for a definite subdomain of railway network (EOD in the case) and a given train traffic, find the best locomotive-to-train matching that minimizes total train delay at stations and maximizes productive locomotive work time. Railway network subdomain is represented by a weighted bi-directed graph with stations as vertices and the average travel time between stations as weights of the edges. Trains are objects with attributes: weight, route, and departure readiness time. Train route is a sequence of railway stations from departure station to destination station. Railway graph is partitioned into connected sub-graphs, called *traction domains*, so that each locomotive may operate within definite traction domain(s). Locomotives are objects with attributes: allowed traction domain(s) and weight-carrying capacity.

Train routes are transformed into preliminary train schedules by assigning to each station time label when the train passes (or visits) the station. The pre-scheduling is performed in accordance with the train departure time and the average road block passing times. Then preliminary schedules are projected onto traction domains to form so-called *loco-slots* – atomic segments at which the train can be carried by one locomotive. In other words, loco-slots are objects to be matched by locomotives for time periods defined by the loco-slot start and finish time.

For the time horizons of interest (24–48 hours) each locomotive operation is confined by one traction domain. Therefore further analysis without much loss of generality can be performed in terms of one domain only. The problem reformulates: for given set of loco-slots and current locomotive positions we have to find locomotive schedules that fill maximum loco-slots with locomotives. The solution has to satisfy all weight limitations and minimize loco-to-train weight capacity excess. This excess includes empty loco runs (with no train at all) if needed.

III. Network Flow Approach

Loco-slots are just parts of train schedule in the time-space plane, confined by spatial limits of traction domain. Let's consider only *linear* traction domains – i.e. subgraphs which are *paths*. From the further reasoning it will be clear that without any complications it may be applied to any tree-like road subdomain as well. For linear domain we see two kinds of loco-slots, moving in opposite directions to each other. On the schedule plane those two kinds are represented by two sets of segments, one with positive – another with negative slope in respect to time axis (see Fig. 1.)

On Fig. 1. six different train schedules are shown. Vertical axis comprises four consecutive stations of the traction domain, and horizontal one represents time span in some abstract scale. Thus trains 1 to 3 go in upward (in respect to the Figure) direction and the other three go in downward direction. For the sake of simplicity all trains are shown with constant equal speed and no intermediate stops.

As you can see, schedule plane possesses features of a directed network structure: consider loco-slots as vectors in time-space plane directed from smaller time values towards greater ones. Let's think of those vectors as directed graph edges (and call them *slot edges*). Thus bounding vertices for the slot edges will be the *events* in the time-space plane: each edge starting from a *train departure event* point and finishing in the *train arrival event* point on the schedule plane. Slot edges are depicted as thick grey arrows at Fig. 2.

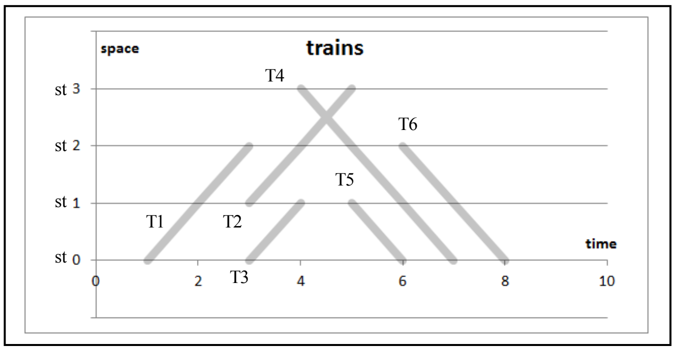

Fig. 1. Train schedule example

To transform a set of *slot edges* positioned on the schedule plane into a connected graph one principal constructive step must be performed. Its idea originates from the notion of alternative graph widely used in job-shop scheduling models applied for optimal train rescheduling problems [5]. Suppose that a locomotive finishes work with one loco-slot and then starts working with another one. This transition may be represented on the schedule plane as a special edge connecting end of the first slot edge to the beginning of the second one. Let's call this auxiliary edge a *transition edge*. Of course, the transition must be physically possible, i.e. there can be no edges directed back in time, or having slope greater than maximum locomotive speed, so as for loco to have time to get from the arrival station of the first slot to the departure station of the other (if they differ). The transition edges are shown by thin black curved arrows at Fig. 2.

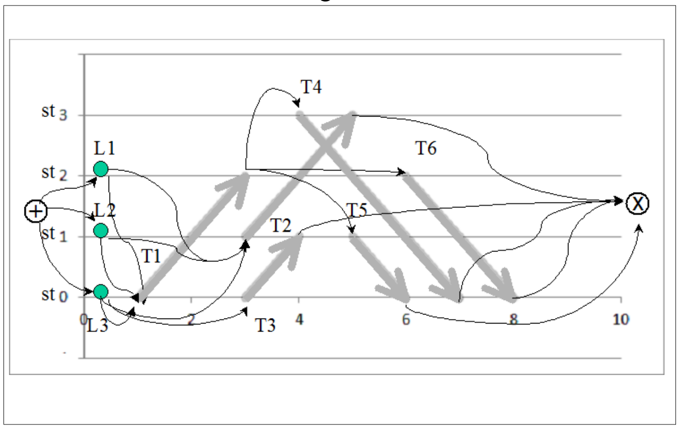

Fig. 2. Network graph for the loco assignment problem (some transition edges are omitted)

Our goal is to formulate a network flow problem, so having edges let's specify flow units, which are to pass through the network. Obviously they are locomotives. To complete the whole set into a network special vertices are added to graph. Those vertices are situated on the schedule plane at the points (or, to be precise, *events*), where locomotives become available at first time for the timespan (and this is not necessarily at the beginning). Let's call them *loco-vertices* (in contrast with *slot-vertices* confining slot edges). As was mentioned, slot-vertices are divided into *departure-vertices* and *arrival-vertices*, and functionally loco-vertices are special case of arrive-vertices, in a sense that loco *arrives* to (or appears at) this vertex to be used for the first time. Loco-vertices are shown as small filled circles labelled L1 … Ln at Fig. 2.

Finally, two obligatory vertices are to be added: the *source* and the *sink*. Surely, they do not lie in the physical time-space plane, though they are needed to complete the graph to a network. One may think of a source as lying somewhere before the timespan (big circle marked by plus-sign at Fig. 2.), and a sink – somewhere after it (circle marked by x-sign), but only virtually, just to simplify the representation.

Last three steps are to connect: a) the source to the loco vertices, b) the loco vertices to physically available departure-vertices, and c) all arrival-vertices (including loco-vertices) to the sink. Step (a) is absolutely formal and just includes the source to the network. Combined with formal initial push of flow units (locomotives) from the source to its neighbors (loco-vertices) it states the fact of initial presence of locomotives in loco-vertices, as was implied by the loco-vertices construction itself. Step (b) is necessary to put the locomotives into the game, i.e. it creates initial edges for locos to match to slots. And. vice versa, step (c) creates edges called *exit-edges* which give locos opportunity “to finish its workday”, either after some loco-slot matching or from the very beginning (as there may be locos unassigned to any job).

Thus the network graph topology is ready. Still, there’re steps to be done for MCFP to be complete – equip edges with costs and volume capacities, in such a way that the solution will represent the optimum for the original traction resource control problem. Obviously, costs must provide main property of the task – attraction of the resources to the slot-edges, so that maximum loco-slots are assigned with locomotives. Another condition is to minimize train and loco stops and loco empty runs. The former requirement is simply satisfied by setting slot-edge costs as small as it gets (e.g. zero, if all costs are to be nonnegative). The latter one is a bit more complicated and requires constructing special metrics in the physical time-space plane. These metrics must provide adequate charges for loco waiting and relocation to pick up a train at another station. Expectedly, maximum charges have to be put on the exit-edges, in order to prevent locomotive from becoming idle too early.

IV. Results

Highly efficient *preflow-push* algorithm of Goldberg [5] was used to solve the stated Minimum Cost Flow Problem (MFCP). Algorithm was implemented using Java programming language and tested on the very six-train example described in the previous section and shown at Fig. 2. The solution obtained is shown at Fig. 3. It demonstrates adequateness and optimality for the six-train case. Locomotive paths along graph are depicted by thin lines, while train schedules (loco slots) are shown by thick grey lines as before.

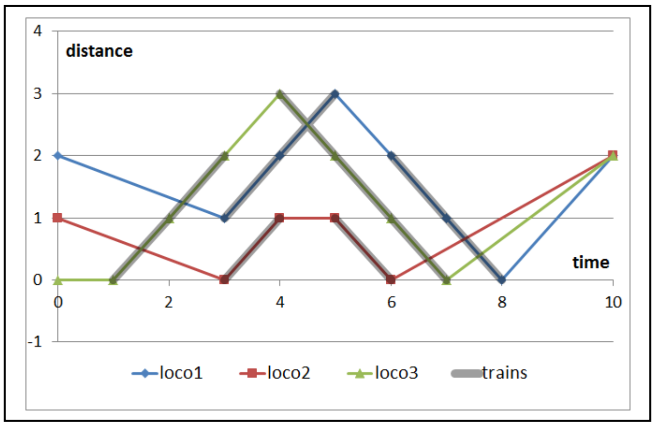

Fig. 3. Solution of the six-train example

On the next step the algorithm was applied to the model problem that simulated real life traffic along a separate segment of the Eastern Operational Domain (EOD). It comprised about 10% of all EOD traffic, and was free of some technology intricacies. There are 137 trains and 84 locomotives in that model example.

The algorithm showed good results: depending on maximum train delay allowance it matched from 62% (no delay permitted) to 100% (any delay permitted) of loco-slots to locomotives. Solution for the common tradeoff situation of maximum 4-5 hours of train delay is shown at Fig. 4. Hollow grey lines represent matched-to-loco trains and filled black lines represent trains unable to move due to lack of traction resources. This is not very good result but on the other hand test model example was constructed to testify algorithm in a hard situation of a resource deficit.

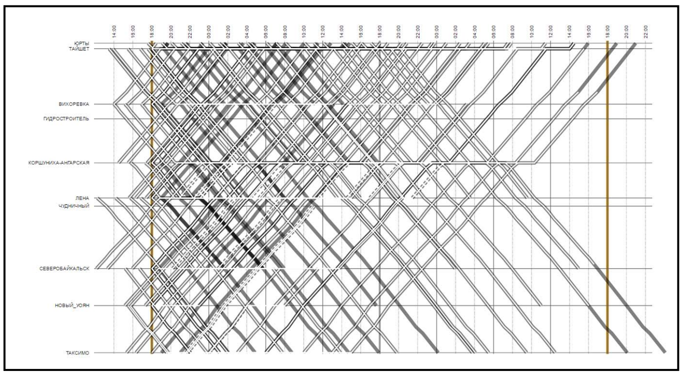

Fig. 4. Solution of the real life model example

Next the algorithm was applied to real traffic on the Eastern Operational Domain whose size was of about 1500 trains every day (3000 trains on the planning horizon of 48 hours), and 1800 locomotives available. Algorithm has finished calculations in 2 minutes and showed result of 85% of loco-slots matched to locos with common limitation of maximum train delay. This result greatly outperforms any automatic planning attempts done before. Along with high assignment percentage, other key indices of freight railway transportations are very high, too: loco daily productivity is over 2700 ton, and loco effective daily mileage is about 900 km. Nevertheless, these last two brilliant figures are to degrade after locomotive team assignment stage, since stops to change a loco team will be added to train schedules.

V. Conclusion

Thus, we demonstrated that network flow algorithms can be successfully applied to the railway traction equipment matching to trains optimization problem. As a matter of fact, the approach proposed in the present work proves to be much more effective than any other used up to the current moment. Push-relabel method of Goldberg was used to solve minimum-cost flow problem stated using technology restrictions formalization. Inherent rollbacks of Goldberg's algorithm allow for seeking time-global optimal solution, which is crucial in the case considered.

On a practical level, automatic planning of traction equipment (locomotives) attachment to freight trains was implemented for industrial use on the Eastern Operational Domain (EOD) of the Russian Railways. The planning is performed at 48-hours horizon for a real life traffic data. Efficiency of the algorithm and its implementation allows using it for real-time calculations for industrial data volumes for EOD at any time horizon of practical interest. The implementation of the algorithm is a part of a complex technology for EOD.

Acknowledgment

Programming implementation has been done in cooperation with Boris Smilga and Dmitry Ezhov. Real-life test model was prepared by Pavel Voytenko. Test model results visualization is automated by Boris Smilga.

The work was supported by Russian Foundation for Basic Research (projects no 17-20-01160 and 17-20-01180).

References

[1] V.G. Matyukhin, V.A. Sharov, and A.B. Shabunin "Online Control of Railway Transportation." (in Russian) Pult Upravleniya. 2012.

[2] V.G. Matyukhin, A.B. Shabunin, and E.F. Nemtsov. "Traction resources management on Eastern operating domain (of Russia), "Locomotiv" (in Russian), 2017, No 1 (721), pp. 8–9.

[3] F. F. Paschenko, I.K. Minashina, E.M. Zakharova, N.A. Kuznetsov, N.G. Ryabykh "Intelligent control system for the rail transportations", AICT2016 Conference printed proceedings. Baku: Curran associates, Inc., 2016. No 1. pp. 383–387. 2016

[4] D.P. Bertsekas, D. Castanon. "A forward/reverse auction algorithm for asymmetric assignment problems", Computational Optimization and Applications», 1992, 277 – 297.

[5] A.V. Goldberg and R. E. Tarjan, "Efficient Maximum Flow Algorithms," Communications of the ACM. 2014. vol. 57, No 8, pp. 82–89.

[6] A. Mascis, D. Pacciarelli, "Job-shop scheduling with blocking and no-wait constraints", European Journal of Operations Research, 2002, vol. 143, No 3, pp. 498–517.