Schedulling the delivery of orders by a freight train

A.Lazarev
V.A.Trapeznikov Institute
of Control Sciences
of Russian Academy of Sciences,
M.V.Lomonosov Moscow State University,
National Research University
Higher School of Economics
Email: jobmath@mail.ru

E.Musatova
V.A.Trapeznikov Institute
of Control Sciences
of Russian Academy of Sciences,
Email: nekolyap@mail.ru

N.Khusnullin
V.A.Trapeznikov Institute
of Control Sciences
of Russian Academy of Sciences,
Email: nhusnullin@gmail.com

***Abstract*—In this research it was considered the particular case of a railway problem, specifically, the construction of orders delivery schedule for one locomotive plying among three railway stations. In this paper it was suggested a polynomial algorithm and were shown the results of a computing experiment.**

I. Introduction

Nowadays, problems of the rail planning are attracting attention of specialists due to the fact that they are challenging, tough, nontrivial and, what is more important, are of practical significance.

In this research we consider the problem of making up a freight train and the routes on the railway. It is necessary from the set of orders available at the stations to determine time-scheduling and destination routing by railways in order to minimize the total completion time.

In this paper it was studied the particular case of the problem, specifically, the construction of orders delivery schedules among 3 railway stations by one locomotive (Fig. 1). Application of dynamic programming is very effective for the solution of this problem. In this paper it was suggested a polynomial algorithm and shown the results of the computing experiment.

II. Problem statement

At each station there is a set of orders available for delivery. Each order is characterized by a release date and a destination station. If the order consists of a few cars $k > 1$ then for each car there will be created a separate order.

Let us introduce the following notations:

- q – the maximal number of the cars (wagons);
- O – set of all orders;
- n – total number of orders;
- n_{ij} – set of orders available for delivery between stations i and j;
- J_{ijk} – k-th order for delivery from station i to destination station j;
- r_{ijk} – release time of the orders;
- p_{ij} – travelling time.

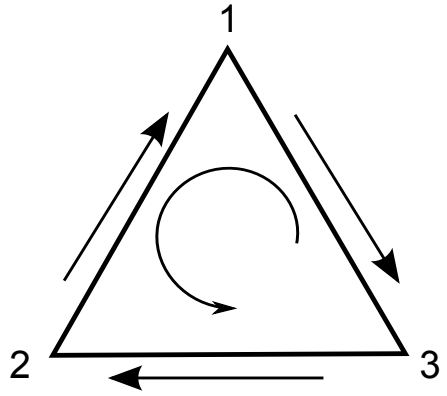

Fig. 1. The railway station location

To simplify the description of our algorithm we will assume that $p_{ij} = p \; \forall i \neq j$.

The objective function which tries to minimize total completion time is the following:

$$\min F = \sum_{J_{ijk} \in O} C_{ijk}, \tag{1}$$

where C_{ijk} is the completion time to destination station. Also, this function describes the average time of order delivery so it can be rewritten in the form:

$$F = \sum_{J_{ijk} \in O} \frac{C_{ijk} - r_{ijk}}{n}.$$

This problem is the generalization of the two stations problem for which polynomial algorithms are known.

It is not difficult to notice that the locomotive can have the following strategies of its route management.

1. Moving. If the locomotive stays at any station, moving is possible in one from two directions with maximum of orders available but not more than q.

2. Waiting. This point is possible if the total number of orders available for delivery is less than q (cars capacity of

a train). And this mode is impossible if the flow of the new orders is not expected.

3. Idle. This mode is necessary if the number of orders is not available for delivery. Obviously, the "idle" is impossible to use twice in succession. Also, the locomotive can idle only after departure to the station or at the starting time.

It is easy to show that using of these strategies does not cut optimal schedule that minimize the objective function. Therefore, let us assume that the locomotive movement satisfies these rules.

Definition 1. *Let us suppose, that the locomotive is in the state* $S(s,t,k_{12},k_{23},k_{31},k_{13},k_{32},k_{21})$ *if at the time moment* $t \in T$*, it is at the station* s *and by the current time moment has been delivered* k_{12} *orders from the first to the second station,* k_{23} *orders from the second to the third station and etc.*

Let the objective function value of the state $S(s,t,k_{12},k_{23},k_{31},k_{13},k_{32},k_{21})$ be denoted by $C(s,t,k_{12},k_{23},k_{31},k_{13},k_{32},k_{21})$

The transition from one state to another occurs according to the strategies mentioned above. In this case, if the locomotive can move from the state S^1 to S^2 directly, then the objective value of the state can be calculated with the help of the following formula:

$$C^2 = C^1 + (t' + p) * k,$$

where t' is the time moment from the state S^1 and k is the number of the orders delivery when transforming into the new state.

The objective function value does not change if the locomotive moves to another station in idle or waiting mode.

In the case of different travelling times $p_{12},\ p_{23},\ p_{13}$ the set of possible moments of the locomotive departure equals to

$$T = \{t = r_{ijk} + m_1 p_{12} + m_2 p_{23} + m_3 p_{13}\} \cup$$

$$\cup \{t = m_1 p_{12} + m_2 p_{23} + m_3 p_{13}\},$$

where $i,j \in \{1,2,3\}$, $k \in \{1,\dots,n_{ij}\}$, $m_1 + m_2 + m_3 \in \{0,\dots,2n-1\}$. It means that the power of the set T is $O(n^5)$.

III. Concepts of the algorithm

The main idea of the algorithm is the following: first of all, the graph of states in ascending order of t is built. The states are generated by the strategies mentioned above. From the same two states in the tree remains the only one that has a lower value of objective function. The solution to the problem is to reach the state which has the lowest value of the objective function from the set of the states completed:

$$\min_{s,t} C(s,t,n_{12},n_{23},n_{31},n_{13},n_{32},n_{21}).$$

Complexity of the algorithm is estimated by the total number of states in the graph. Since the number of time moments t from the set T is $O(n^2)$, the total number of states can be estimated as $O(n^2 \prod_{i \neq j}(n_{ij}+1))$ or $O(m^8)$, where $m = \max_{ij} n_{ij}$.

One of the key moments of our approach is the merge of the same nodes. The states are considered equal if at the time moment t both locomotives are on the same station and the numbers of the orders delivered to each station are also equal. Obviously, from two states in the tree remains the only one that has a lower value of the objective function. If the state was added to the tree before, the algorithm will replace it, otherwise we choose just added one.

This situation is represented in the Fig. 2. As you can see the value of the state S(1,7,2,0,0,0,0,2) equals to 22, if its parents were enclosed in the quadrilateral and equals to 24, if its parents were enclosed in the pentagon. This condition can be an important factor in choosing between them (parent' branch). Thereby, on each step it is necessary a full tree survey.

In the simpliest implementation of the algorithm the solution tree can be stored in the memory. But this approach it not optimal. For minimization of the memory used and increasing the performance this work suggests the other tree representations in the memory and also creats a garbage collector. In the RAM arestored only the states which belong to $[t_i - p; t_k]$, where t_i is the current time moment, p is the traveling time and t_k is the maximum value of the time of the set T. States that do not satisfy this condition should be relocated to the hard disk. They will be needed later when it is necessary to build and show a full branch of the tree.

During the tree creation process, as well as for the branch and bound scheme, one of the important factors is a cutting off an "unpromising" branch. We obtain the upper bound $\overline{C}$ when the first complete state (all orders were delivered) is received.

After that, the algorithm tries to check the execution of the inequation for all of the following states in order to cut off the nodes that have the worst value of the objective function:

$$C' + \sum_{J_{ijk}} [max\{t, r_{ijk} + p\}] > \overline{C},$$

where C' is the value of the current state, t is the current time moment. The left side of the inequation is the lower bound for the current state (all unfulfilled orders delivered to the destination after they are received immediately).

In order to illustrate our approach, let us the following example and set n=6, $r_{1,2}$=$r_{2,3}$=$r_{3,1}$={1,3}, q=2, p=2. The locomotive at the initial time t = 0 is at the station 1 and has the following options:

- to stay at the station s = 1 until the time of order receipt t = 1, thus go to state $S(1,1,0,0,0,0,0,0)$;
- to move to the station s = 2 by the idle, $S(2,2,0,0,0,0,0,0)$;
- to move to the station s = 3 by the idle, $S(3,2,0,0,0,0,0,0)$.

If at the initial time t=0, the locomotive stays at the station s=1 until the time of order receipt then it is possible to deliver the first order available either to the station s=2 at the next time moment, $S(2,3,1,0,0,0,0,0)$, or to stay at the station s=1 until the time of order receipt, $S(1,3,0,0,0,0,0,0)$. If at the initial time t=0 the locomotive moves to the station

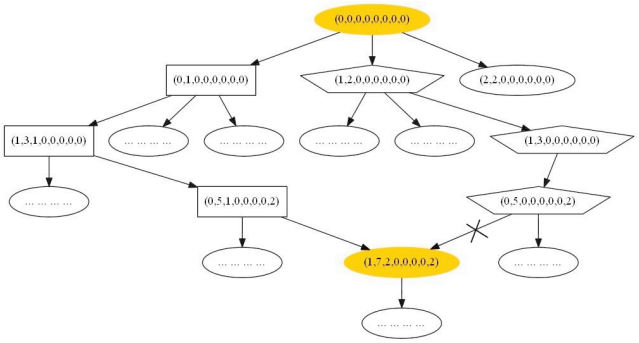

Fig. 2. The same states merging process

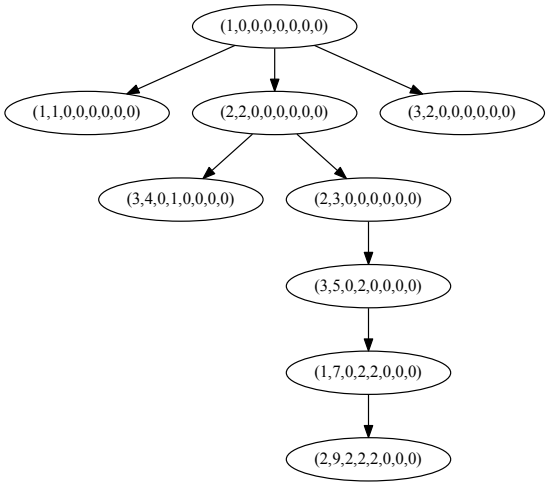

Fig. 3. The part of states graph

s=2 by the idle, then at the next time moment the locomotive can transport all orders available to the station s=3 or stay at the station s=2 until the time of the order receipt. In the latter case the locomotive has the only one choice: to carry all orders available at this time moment to the station s=3, $S(3,5,0,2,0,0,0,0)$. It should be noted that for the locomotive there are no any other options for the transition from the previous state. When the locomotive stays at the station s=3, he has the only one possible way: to carry all orders available to the station s=1, $S(1,7,0,2,2,0,0,0)$. After that the locomotive can ship remaining orders to the station s=2, $S(2,9,2,2,2,0,0,0)$ and in this state the locomotive delivers all orders available. The part of states graph is shown in the Fig. 3

IV. Computing experiment

Table I shows the results of a computing experiment. The first column contains input parameters – time moments, the second column contains the total number of orders, the third – the number of the nodes in the tree if the problem was solved through the blind search, the fourth – the number of theoretical nodes, in the last one - the number of the nodes which were obtained in practice. In all examples set $p = 2$, $q = 2$. Also,

TABLE I. Results of computing experiment

input values	cars count	blind search	theoretic dynamic prgrm.	practic dynamic prgrm.
$r_{1,2} = r_{2,3} = r_{3,1} = \{1,3\}$	6	3^{27}	648	38
$r_{1,2} = r_{2,3} = r_{3,1} = r_{1,3} = r_{3,2} = r_{2,1} = \{1,3\}$	12	3^{51}	753	387
$r_{1,2} = r_{2,3} = r_{3,1} = r_{1,3} = r_{3,2} = r_{2,1} = \{1,3,5\}$	18	3^{77}	166 212	2 260
$r_{1,2} = r_{2,3} = r_{3,1} = r_{1,3} = r_{3,2} = r_{2,1} = \{1,3,5,7\}$	240	3^{725}	1 154 289 852	1 268 585

from this table it may be seen that the practical complexity is much lower than it is theoretical estimation.

V. Conclusion

In this research it was analysed the problem of making up a freight train and its routes on the railway. Also, it was proposed a polynomial algorithm for the construction of orders delivery schedules for one locomotive plying among 3 railway stations. As an example, were represented the steps of making up a freight train and destination routing in order to minimize the total completion time. Also, there were shown the results of the computing experiments, the upper bound of the complexity and the total number of nodes while solving the problem by different approaches. The complexity of this algorithm is $O(n^8)$ operations.

Future research

- Creation of a fast and accurate technique to determine a lower bound for cutting off an unpromising branch;
- Consideration of more complex arrangement of the stations in the limits of which a locomotive will have an opportunity to deliver orders;
- Investigation of the case when orders are delivered by means of several locomotives;
- Improvement of the algorithm performance and decreasing the RAM usage;
- Parallelizing the algorithm.

References

[Lazarev et al. "Theory of Scheduling. The tasks of railway planning"(2012)] Lazarev A.A., Musatova E.G., Gafarov E.R., Kvaratskheliya A.G. Theory of Scheduling. The tasks of railway planning. – M.: ICS RAS, 2012. – p.92

[Lazarev et al. "Theory of Scheduling. The tasks of transport systems management"(2012] Lazarev A.A., Musatova E.G., Gafarov E.R., Kvaratskheliya A.G. Theory of Scheduling. The tasks of transport systems management. – M.: Physics Department of M.V.Lomonosov Moscow State University, 2012. – p.160

[Caprara(2011)] A. Caprara, L. Galli, P. Toth. Solution of the Train Platforming Problem Transportation Science. 2011. - 45 (2), P. 246-257.

[Zwaneveld(2001)] Zwaneveld P. J., Kroon L. G., van Hoesel S.P.M. Routing trains through a railway station based on a node packing model. European Journal of Operational Research. 2001. - No. 128 P.14-33.

[Lazarev et al. ”The integral formulation the tasks of making up a trains and their movement schedules”(2012)] Lazarev A.A. Musatova E.G. The integral formulation the tasks of making up a trains and their movement schedules. The managing a large systems. The issue 38. M.: ICS RSA, 2012. – p.161-169.

[Liu(2012)] Liu S.-Q., Kozan E. Scheduling trains as a blocking parallel-machine job shop scheduling problem. Computers and Operations Research. - 36(10) P. 2840-2852.

[Baptiste ”Batching identical jobs”(2000)] Baptiste Ph. Batching identical jobs. Math. Meth. Oper. Res. 2000. - No. 52 P.355-367.

[Hagai Ilani et al. ”A General Two-directional Two-campus Transport Problem”(2012)] Hagai Ilani, Elad Shufan, Tal Grinshpoun. A General Two-directional Two-campus Transport Problem. Proceedings of the 25th European Conference on Operational Research, Vilnius, 8-11 July, 2012. - P.200.