Special algorithm for Three-Stations Railway problem

Alexander Lazarev[1], **Elena Musatova**[2], **Nail Husnullin**[3]

[1] *Institute of Control Sciences RAS, Moscow State University, National Research University – Higher School of Economics, Moscow, Russia;* jobmath@mail.ru

[2] *Institute of Control Sciences RAS, Moscow, Russia;* nekolyap@mail.ru

[3] *Institute of Control Sciences RAS, Moscow, Russia;* nhusnullin@gmail.com

Problem statement

The paper is devoted to the problem of railway transportation. The railway network consists of stations between which freight cars are transported. Let us S be a set of stations. Every station s has a set $N^s = \{J_1, ...J_{n_s}\}$ of orders to deliver. $N = \cup_{s\in S} N_s$ is a common set of orders. Each order represents one freight car. If an order consists of k cars, we will consider it as k different orders. We know realise time r_j^s and due date d_j^s of delivery for each car $J_j \in N^s$. Let us p_j^s be traversing time for the car $J_j \in N^s$ and w_j^s be its weight (importance). Our goal is to design freight trains and work out their schedule. Objective functions can be the following:

- minimizing the weight total tardiness
$$\min \sum_{s\in S} \sum_{j\in N_s} w_j^s \max\{0, C_j^s - d_j^s\};$$
- minimizing the total completion time
$$\min \sum_{s\in S} \sum_{j\in N_s} C_j^s;$$
- minimizing the maximal lateness
$$\min \max_{j\in N_s, s\in S} \{C_j^s - d_j^s\};$$
- in the set N find subset $\overline{N} \subseteq N$ of cars that can be delivered on time:
$$\max \sum_{j\in \overline{N}} w_j^s - \sum_{j\in N\setminus \overline{N}} z_j^s.$$

These problems are large-scale and difficult to solve. Therefore, we proposed to divide them into subproblems which are easier to solve and consider special cases which would help to find important structural properties which are hard to recognize in the general case. We have suggested a number of railway basic models (with two stations, with chain of stations and so on) that gives us an opportunity to develop special exact algorithms which can be used in general railway problems. Some algorithms for two-stations railway problems can be found in [1]. In this paper we propose an algorithm for the special case of three stations.

Three-Stations Railway problem

Consider the problem with 3 stations that are connected by a rail road and one locomotive. A valid arrangement is shown on fig. 1. Arrows indicate a possible route of the locomotive from one station to another.

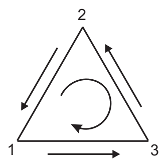

Fig. 1 Possible arrangement of stations

We have to implement a set of orders. N^{ij} is a set of orders that should be delivered from the station i to the station j. So $N^1 = N^{12} \cup N^{13}$ etc. Let us assume that p is a traversing time from one station to another, q is a capacity of a train, r_i^s is a release time of $J_i^s \in N^s, s \in \{1,2,3\}$, n_s is a common number of orders at the station s, n_{si} is a number of orders at station S that are should be delivered to the station i.

Objective function of the problem is following:

$$\min \sum_{J_i^1 \in N^1} C_i^1 + \sum_{J_i^2 \in N^2} C_i^2 + \sum_{J_i^3 \in N^3} C_i^3. \qquad (1)$$

It is easy to see the locomotive has the following strategies when he arrives to a station:

1. staying at the station and waiting a new order;

2. idling to the next station;
3. idling to the previous station;
4. moving to the next station with the largest possible number of cars;
5. moving to the previous station with the largest possible number of cars.

Only for this objective function (1) we can reduce the number of possible solutions if we consider the following:

- idling to the next or previous station is the same as moving without the wagons;
- if we can move at the second station with q wagons it is more preferable for us than stay at the station;
- idling to another station is preferable, if all orders have been delivered from station s.

It is obviously that in an optimal schedule the train begins his movement from a station s only at the moments of its arrival to this station or at the moment of appearance of a new order, i.e. at the moment r_i^s. So times points at which the train begins and ends movement between stations belong to $T = \{t : \exists r_j^s, \exists l \in \{1, \ldots, (n_1+n_2+n_3)\},\ t = r_j^s + lp\}$.

Let us denote by

$$S(s, t, k_{12}, k_{13}, k_{21}, k_{23}, k_{31}, k_{32}) \tag{2}$$

the state at the moment $t \in T$, where s is the number of the station where the locomotive is, k_{12} is the number of delivered orders from the first to the second station, k_{23} the number of delivered orders from the second station to the third one, etc. Let us assume that $P(s, t, k_{12}, k_{13}, k_{21}, k_{23}, k_{31}, k_{32})$ is the smallest total delivery time in the scheduling which leads to state $S(s, t, k_{12}, k_{13}, k_{21}, k_{23}, k_{31}, k_{32})$. For the objective function (1) the optimal solution of the problem is

$$\min_{s,t} P(s, t, n_{12}, n_{13}, n_{21}, n_{23}, n_{31}, n_{32}). \tag{3}$$

E x a m p l e 1. Consider the following problem. Let us assume that $r^{12} = (1, 2, 3)$, $r^{23} = (2, 3, 4)$, $r^{13} = (3, 4, 5)$, $r^{21} = (2, 3)$, $r^{32} = (1, 2)$,

$r^{31} = (2,3)$, $p = 2$, $q = 2$. One of the possible scheduling solutions is shown on fig. 2.

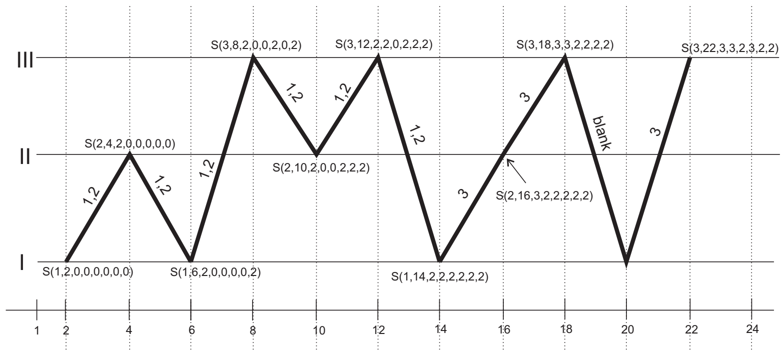

Fig. 2 One of the possible scheduling solutions

Algorithm 1 describes the calculation of the number of cars that the train can take to the next station in (2).

Algorithm 1

```
1: function GetPossibleOrders(s,$n_{s,futureS}$)
2:     $futureQ \leftarrow 0$
3:     while $r_i^s < t$ do
4:         if $i > n$ & $i - n <= q$ then $futureQ \leftarrow futureQ + 1$
5:         end if
6:     end whilereturn $futureQ$
7: end function
```

Let us introduce the following denotations:
t — current time;
s — station number, where the locomotive is located at the time t;
j — number of wagons, which the locomotive can take at the current time;
n — number of delivered orders from station s;
q — max number of wagons that the locomotive can carry at a time;
$N[s][futureS]$ — array, which contain number of delivered orders from station s to $futureS$;
Runner — entry point, which execution of the program begins with;
$existsCarsOnStay$ — function, which returns false, if all orders have been delivered from station s.

Algorithm 2 creates nodes of the tree and allows to move from one station to another.

```
Algorithm 2
function RUNNER
    int[,] N ← new int[s,s]
    $N \leftarrow 0$
    $newS \leftarrow S(1, 0, N)$
    $BuildTree(newS)$
end function

function BUILDTREE(prevS)
    $j \leftarrow 1$
    $q \leftarrow 0$
    $s \leftarrow prevS.s$
    $t \leftarrow prevS.t$
    $N \leftarrow prevS.N$
    while $j..3$ do
        if $j = 1$ then $futureS \leftarrow s \oplus_3 1$
        end if
        if $j = 2$ then $futureS \leftarrow s \ominus_3 1$
        end if
        if $j = 3$ then $futureS \leftarrow s$
        end if
        $q \leftarrow GetPossibleOrders(s, N[s, futureS])$
        $N[s, futureS] \leftarrow N[s, futureS] + q$
        if $j <> 3$ then
            $t \leftarrow t + p$
        else
            $t \leftarrow t + 1$
        end if
        $newS \leftarrow S(futureS, t, N)$
        if $existsCarsOnStay(newS)$ then
            $BuildTree(newS)$
        end if
    end while
end function
```

As we have $O((n_1 + n_2 + n_3)^2)$ possible time moments running time of the proposed algorithm is $O((n_1 + n_2 + n_3)^2 n_{12} n_{13} n_{21} n_{23} n_{31} n_{32})$.

The authors were supported by the Russian Foundation for Basic Research (project no. 11-08-13121).

REFERENCES

1. *A.A. Lazarev, E.G. Musatova, E.R. Gafarov, A.G. Kvaratskhelia.* Schedule theory. Problems of railway planning, ICS RAS, Moscow (2012).