The Problem of Scheduling for the Linear Section of a Single-Track Railway with Independent Edges Orientations

Elena N. Akimova[1,2], Damir N. Gainanov[1], Oleg A. Golubev[1], Ilya D. Kolmogortsev[1], and Anton V. Konygin[1,2⋆]

[1] Ural Federal University, Yekaterinburg, Russia,
[2] IMM UB RAS, Yekaterinburg, Russia
aen15@yandex.ru,damir@dc.ru,golubev.oa.66@gmail.com
ilia12r@gmail.com,konygin@dc.ru

Abstract. The paper is devoted to the problem of scheduling for the linear section of a single-track railway: how to organize the flow in both directions in the most efficient way. In this paper, the authors propose an algorithm for scheduling with independent edges orientations, examine the properties of this algorithm and perform the computational experiments.

Keywords: scheduling · track capacity · optimization · combinatorial graph algorithms

1 Introduction

In this paper, we continue a consideration of the problem of scheduling for the linear section of a single-track railway started in [1]. The schedule optimization allows to increase a track capacity of the section using the same physical resources and to simulate further modifications of the section.

This problem with different modifications is well-known (see [2], [3], [4]). For example, in [2] the simulation of trains on the railway based on the moving-block system and fixed-block system was presented. The previous work was devoted to the case, when all edges of a graph in every moment have similar orientations. Now we consider more general situation: edges can be orientated independently. For this case we get both theoretical results and experimental. Numeric experiments based on algorithm, which is implemented in C++ using the MPI+OpenMP hybrid technology.

In this work we use notations proposed in [1]. Let the linear section of a single-track railway be given. Thus, we have a graph with the set of vertices

$$V = \{v_i \mid 1 \le i \le n\}$$

⋆ This work was partly supported by the Center of Excellence "Quantum and Video Information Technology" of Ural Federal University Development Program. The work was supported by Act 211 Government of the Russian Federation, contract №02.A03.21.0006.

and the set of edges E (standard graph theory notations, see [5]). Furthermore, we assume that the vertices are indexed in such a way that the edges look like $\{v_i, v_{i+1}\}$.

For each station v_i we have a positive integer $m(v_i)$, being the number of auxiliary railway tracks at this station. These tracks are the only places where a train can stop. Each auxiliary track can hold only one train.

Therefore, if all auxiliary tracks of the station are occupied then a train cannot stop at this station.

By $l(e)$ we denote the length of the track corresponding to edge e from E. Suppose all trains have equal speeds. It should be a nonnegative time interval ϵ between two trains moving in same direction. Also, there is an isolated station $w \in V$ such that every train passing this station is obliged to stay at it for a nonnegative time δ due to technical reasons (the change of locomotive crew, train inspection, etc.). Stations v_1 and v_n are sources and receivers of trains. Thus, there are two directions of movement: from v_1 to v_n and from v_n to v_1 (these directions are denoted by $v_1 \mapsto v_n$ and $v_n \mapsto v_1$, respectively). We can assume that each train does not change the direction and does not visit any vertex twice. For passing each other when moving in different directions, one of the trains waits another one on an auxiliary track.

Problem statement

Let $\Gamma = (V, E)$, $m : V \to \mathbb{N}^0$ and T — the whole time period. Since adding new stations without auxiliary tracks, we get the same problem, we can suppose that, without loss of generality, all edges have the same length 1:

$$l(\{v_i, v_{i+1}\}) = 1.$$

Also, we assume that trains pass one edge per the unit of time.

The section corresponds to a single-track railway. So for every moment of time trains can move on every edge only in one direction. Thus, if we have R — a schedule of trains movement on Γ, then we can define map $s_R : T \times E \to \{-1, 1\}$, where $s_R(t, e) = 1$, if there is a movement with direction $v_1 \mapsto v_n$, and $s_R(t, e) = -1$ otherwise. Let $\mathcal{R}$ be a set of schedules such that there exist partition the whole time interval T onto disjoint half-intervals with similar length τ

$$T = \cup_{i=0}^{k} [t_i, t_i + \tau),$$

where $s_R(t_i, e) = s_R(t_i + \tau', e)$ for all $R \in \mathcal{R}$, $e \in E$ and $0 \leq \tau' < \tau$.

We need the following sets of schedules of trains movement.

1) Let $\mathcal{A}$ consists of all schedules R from $\mathcal{R}$, such that for all $t \in T$ we have $s_R(t, e) = s_R(t, e')$ for all $e, e' \in E$. Thus, for R in $\mathcal{A}$ there are no trains moving in opposite directions — in any moment of time trains move in fixed direction or stop on auxiliary tracks.

2) Let $\mathcal{B}$ consists of all schedules R from $\mathcal{R}$ with property, that there exist functions $m_1 : V \to \mathbb{N}^0$ and $m_2 : V \to \mathbb{N}^0$ such that $m_1(v_i) + m_2(v_i) = m(v_i)$ and, for each station v_i, there are $m_1(v_i)$ auxiliary tracks for the direction $v_1 \mapsto$

v_n and $m_2(v_i)$ auxiliary tracks for the direction $v_n \mapsto v_1$. Thus, we divide all the auxiliary tracks into two sets for both directions. Now, the trains can use auxiliary tracks corresponding to their directions and the whole task can be divided into two independent subtasks for each direction.

The problem is to construct and implement a scheduling algorithm to send as much as possible trains for period of time T in both directions (more specifically we are interested in the schedule at which the minimum number of trains in both directions over a specified period of time is maximal, this number per time τ we call track capacity of the section with given schedule).

Properties of schedules from $\mathcal{A}$ and $\mathcal{A} \cap \mathcal{B}$ were studied in [1]. In current paper our aim is to analyze properties of schedules from $\mathcal{B}$. We show, that with some additional assumptions, the maximal track capacity for schedules from $\mathcal{A} \cap \mathcal{B}$ equals to maximal track capacity for schedules from $\mathcal{B}$. In addition, we have numerical experiments, that confirm our conjecture, that this is true in common case (without additional assumptions).

2 Assessment of track capacity

In this section we suppose, that $\epsilon = 0$ and there are no isolated stations.

Proposition 1. *The maximal track capacity for schedules from $\mathcal{R}$ doesn't exceed doubled maximal track capacity for schedules from $\mathcal{A} \cap \mathcal{B}$.*

Proposition 2. *Let R is a schedule from $\mathcal{B}$ and for every edge of Γ total time, when the edge is orientated in direction $v_1 \mapsto v_n$ equals time, when the edge is orientated in direction $v_n \mapsto v_1$. Then there exist schedule R' from $\mathcal{A} \cap \mathcal{B}$ such that track capacity for schedule R' doesn't exceed track capacity for R.*

The following propositions were obtained earlier.

Proposition 3 ([1], Theorem 1). *Let R is a schedule from $\mathcal{A} \cap \mathcal{B}$. Then the track capacity for the section with schedule R doesn't exceed f, where*

$$f = \frac{1}{4} \min_i \sum_{i \le j \le i+\tau} m(v_i).$$

Proposition 4 ([1], Corollary). *Let R is a schedule from $\mathcal{A} \cap \mathcal{B}$. Then there exists schedule R' from $\mathcal{A} \cap \mathcal{B}$ such that the track capacity of the section with schedule R equals to track capacity of the section with schedule R' and for R' we can assume*

$$m_1(v_i) = m_2(v_i) = \frac{m(v_i)}{2}, \text{ for even } m(v_i),$$
$$|m_1(v_i) - m_2(v_i)| \le 1, \text{ for odd } m(v_i).$$

Thus propositions 2, 3 and 4 make possible to determine a maximal track capacity for schedules from $\mathcal{B}$ in case, when for every edge of Γ total time, when the edge is orientated in direction $v_1 \mapsto v_n$ equals time, when the edge

is orientated in direction $v_n \mapsto v_1$. In common case, rough estimations can be obtained from proposition 1.

We suppose that the following conjectures are correct.

Conjecture 1. Track capacity for the section with schedule from $\mathcal{B}$ doesn't exceed of track capacity of the section with some schedule from $\mathcal{A} \cap \mathcal{B}$.

Conjecture 2. Track capacity for the section with schedule from $\mathcal{R}$ doesn't exceed of track capacity of the section with some schedule from $\mathcal{A} \cap \mathcal{B}$.

3 Numerical experiments

Consider a mathematical model of linear section of a single-track railway with 65 stations and 2 of them are isolated with $\delta_1 = 80$ and $\delta_2 = 135$ (isolated station w with δ described in Introduction).

Algorithm

For τ from 60 to 720 minutes and for arbitrary independent edges orientation lets find the maximal track capacity. Below we give verbal description and pseudocode of the algorithm.

For $t \in T$

1. if there is no train at the first station v_1 or at the last station v_n, we create it there;
2. iterate stations from last to first
   (a) for each station, do the following;
   (b) if $s_R(t, \{v_i, v_{i-1}\}) = -1$ and train will be able to move to the next station v_{i+1} then we send it there;
   (c) if there is a train on v_i then go to (a);
3. as we reach the first station v_1, we create and send new trains as much as possible;
4. iterate stations from first to last
   (a) for each station, do the following;
   (b) if $s_R(t, \{v_i, v_{i+1}\}) = 1$ and train will be able to move to the next station v_{i-1} then we send it there;
   (c) if there is a train on v_i then go to (a);
5. as we reach the first station v_n, we create and send new trains as much as possible;
6. increase the time counter and go to 2.

Pseudocode of the algorithm

```
V_{i,t}:= []
V_{1,0}:= m_1(i,t)
V_{n,0}:= m_2(i,t)
for t in T
{
    for i in (n..2)
    {
        if s_r(t, e_{i,i-1}) == -1
        {
            while m_1(i,t) > 0
            {
                train <- minNumberTrainOnStation(V_{i,t})
                remove train from V_{i,t}
                add train to V_{i-1,t+1}
            }
         }
    }
    while m_1(1, t+1) == 0
    {
        create newTrain
        add newTrain to V_{1, t+1}
    }

    for i in (1..n-1)
    {
        if s_r(t, e_{i,i+1}) == 1
        {
            while m_2(i,t) > 0
            {
                train <- minNumberTrainOnStation(V{i,t})
                remove train from V_{i,t}
                add train to V_{i+1,t+1}
            }
        }
     }
     while m_2(n, t+1) == 0
     {
         create newTrain
         add newTrain to V_{n, t+1}
     }

  }
```

The algorithm was implemented in C++ (we used standard data structures, see [6], [7]) using Intel Xeon Phi with offload mode and MPI[8]). The data was distributed between nodes of supercomputers and calculated on Intel Xeon Phi.

Each node processes its own predefined set of time intervals. The communication between nodes is minimal, therefore we have obtained almost linear speedup.

For our experiments we use $|T| = 14400$ (equal to 10 days), cluster with 6 nodes. A node configuration one can see in Table 1.

Table 1. Configuration for one node of the cluster

CPU	2 x Intel Xeon Processor E5-2620 6C 2.0GHz 15MB Cache 1333MHz
RAM	4 x 8GB (PC3L-10600 CL9 ECC DDR3 1333MHz LP RDIMM)
Coprocessor	2 x Intel Xeon Phi 5110P
HDD	IBM 500GB 7.2K 6Gbps NL SATA 3.5" G2SS
NIC	Emulex Dual Port 10GbE SFP+ Embedded VFA III for IBM System x
Network	10 Gb Ethernet
Switch	IBM System Networking RackSwitch G8124E (Rear to Front)

The algorthim uses class *Generator*, which produces fixed distribution of movement direction for each edge with given time interval τ. All these distributions are processed independently, so we have linear growth of speedup. After the processing the algorithm chooses the best schedule.

Tasks for nodes are distributed with MPI in a way that all cores of the processor are used. Thus we create 24 threads with OpenMP for every node (2 processor with 12 cores per node).

We using Intel Xeon Phi with offload mode and create 240 threads with OpenMP.

Execution times, speedup and efficiency of the program for different configurations can be seen from Table 2 and Table 3. Table 2 describes speedup and efficiency of the program compared with 1 node with 12 threads. In this paper the speedup is the ratio of the execution time for 1 node with 12 threads to the value of the Table 2. The efficiency is a ratio of speedup to number of devices (number of nodes or number of Intel Xeon Phi).

Table 2. Execution time of the program

Number of nodes and number of thread per node	**Times (minutes)**
1 node x 12 threads	296
2 node x 12 threads	168
6 node x 12 threads	61
1 node x 1 mic	249
1 node x 2 mic	133
2 node x 1 mic	149
2 node x 2 mic	71

Table 3. Speedup and efficiency of the program

Number of node and number of thread per node	**speedup**	**efficiency**
2 node x 12 threads	1.761	0.88
6 node x 12 threads	4.852	0.8
1 node x 2 mic	1.87	0.93
2 node x 1 mic	1.67	0.83
2 node x 2 mic	3.5	0.82

4 Conclusion

Estimations of track capacity from $\mathcal{B}$ were obtained. The software, which implements the algorithm using MPI and Intel Xeon Phi coprocessor, was created. The numerical experiments were performed on the supercomputer with Intel Xeon Phi. Thus we have obtained a numerical confirmation of Conjecture 1. In future we are going to continue our research and check the correctness of Conjecture 2.

References

1. Akimova E.N., Gainanov D.N., Golubev O.A., Kolmogortsev I.D., Konygin A.V.: The Problem of Scheduling for the Linear Section of a Single-Track Railway. In: ICNAAM (2015)
2. KePing L., ZiYou G., Bin N.: Cellular automaton model for railway traffic. Journal of Computational Physics, vol. 209 (1), 179–192 (2005)
3. Harrod S.: Optimal Scheduling of Mixed Speed Trains on a Single Track Line, in http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.85.5597
4. Higgins A., Kozan E., Ferreira L.: Optimal Scheduling of Trains on a Single Line Track. In: Transpn. Res.-B., vol. 30 (2), 147–161 (1996)
5. Diestel R.: Graph Theory. Springer-Verlag, New York (2000)
6. Cormen Th., Leiserson Ch., Rivest R., Stein C.: Introduction to Algorithms, third edition. MIT Press (2009)
7. Sedgewick R., Wayne K.: Algorithms, fourth edition. Addison-Wesley (2011)
8. Gropp W., Lusk E., Thakur R.: Using MPI-2 Advanced Features of the Message-Passing Interface. MIT Press (1999)