I.J. Intelligent Systems and Applications, **2017, 10, 12-19**
Published Online October 2017 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijisa.2017.10.02

Scheduling Freight Trains in Rail-rail Transshipment Yards with Train Arrangements

Igor Grebennik
Kharkiv National University of Radio Electronics, Kharkiv, Ukraine
E-mail: igorgrebennik@gmail.com

Rémy Dupas
Univ. Bordeaux, CNRS, IMS, UMR 5218, 33405 Talence, France
E-mail: remy.dupas@gmail.com

Oleksandr Lytvynenko and Inna Urniaieva
Kharkiv National University of Radio Electronics, Kharkiv, Ukraine
E-mail: litvinenko1706@gmail.com, inna.urniaieva@nure.ua

Received: 21 June 2017; Accepted: 03 August 2017; Published: 08 October 2017

***Abstract*—**A problem of scheduling freight trains in rail-rail transshipment yards is considered. It is solved at a deeper level compared to original papers dedicated to this problem: besides scheduling service slots for trains, this article additionally solves a problem of assigning every train to a railway track. A mathematical model and a solving method for this problem are given. A key feature of the given mathematical model is that it doesn't use Boolean variables but rather operates with combinatorial objects (tuples of permutations). The solution method is also based on generation of combinatorial sets, which is quite an unusual approach for solving such problems.

***Index Terms*—**Scheduling, Freight Transportation, Transportation Logistics, Transshipment Yard, Combinatorial Set, Generation, Beam Search.

I. Introduction

Intermodal transportation [1] as well as train routing and scheduling [2] are important areas of operations' research nowadays. Particularly, a problem of Container Processing in Railway Yards has got lots of attention recently. A survey [3] describes the problem setting and its various extensions pretty well.

One of the main issues of the Container Processing in Railway Yards area is a problem of scheduling freight trains in rail-rail transshipment yards (TYSP) [4]. The original paper [4] describes five levels of depth for the overall train scheduling problem:

(i) to bundle each train to a service slot, i.e. to schedule trains;
(ii) to assign each train of a bundle to a railway track;
(iii) to make a decision on positions of trains' containers;
(iv) to assign container moves to portal cranes;
(v) to determine a sequence of moving containers for every gantry crane [4] (Fig.1).

The article [4] solves the level (i) of the problem (scheduling the service slots for trains). The article provides a mathematical model and two solution algorithms: an exact algorithm that uses dynamic programming and a heuristic algorithm that utilizes a beam search procedure. A complexity proof for given algorithms is also described there.

Later works bring further improvements to the initial mathematical model and improve solution algorithms. In [5], the original TYSP is extended by new real-world restrictions and a lot of new solution algorithms are given. In [6-9], a branch-and-bound algorithm (developed for the first time in [5]) is improved with a more effective Lagrangian lower bound.

However, [4] and all later works solve only the level (i) of TYSP (scheduling the service slots for trains).

In this article, a deeper problem of assigning every train to a railway track is considered. A mathematical model and a solving method for this problem are given here.

A distinctive feature of the given mathematical model is that Boolean variables are not used and the model mostly works with combinatorial objects (tuples of permutations). The proposed solution method is based on generation of combinatorial sets.

A problem of assigning trains to railway tracks in every service slot is considered. It could be an important task because a total cost of loading/unloading operations in real systems can depend on a distance between a source train and a target train. If a target train and a source one are served within the same time slot, it seems appropriate to place them on the closest railway tracks. If trains are served in different time slots, it is also important to place both trains in such a way that movements of the gantry crane are made as short as possible. In this case, the crane should firstly move containers from the source train to a storage area and then from a storage area to the target train. Thus, both trains should be as close to the storage area as possible.

The remainder of this paper is organized as follows. Section 2 describes a mathematical model of the proposed approach. Section 3 gives details to a solution algorithm. Section 4 presents computational results. Conclusions are given in the final section.

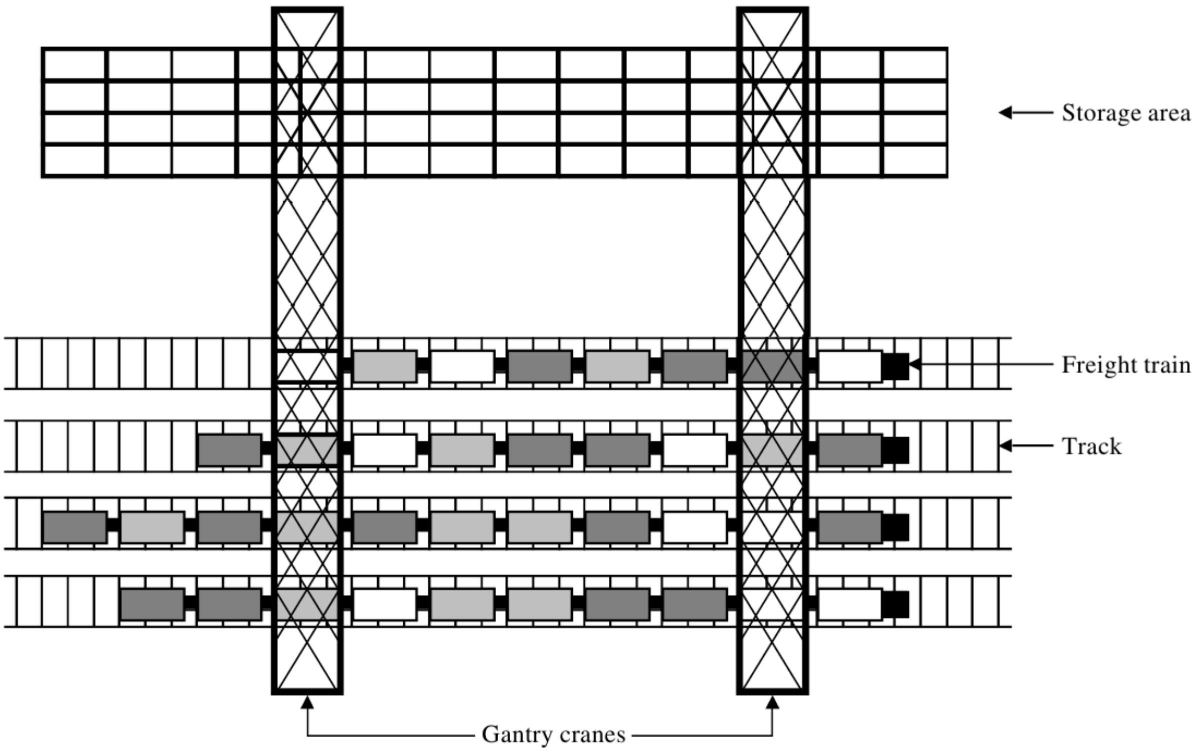


Fig.1. Representation of transshipment yard

II. The Mathematical Model

A mathematical model is a further extension of the one proposed in [4] as it describes a more detailed problem: the problem of assigning every train to a railway track (the level 2 according to [4]). Our model describes the problem using combinatorial structures instead of Boolean variables.

We should dwell on the problem [4] once again. There are G tracks and a given set I of trains, where each train has a predefined number of wagons and a certain load factor, defining a number of containers carried by this train. Each train is then assigned to a service slot $t = 1,\ldots,T$ of G simultaneously served trains. This assignment is restricted to the earliest available slot e_i of a train i (the earliest arrival time) and the latest available slot l_i (the latest departure time). The transshipment yard typically deals with distinct bundles of trains (also known as *service slots*). It means that G trains (one per a track) are simultaneously served and jointly leave the system after all container moves required for that bundle of trains have been accomplished. Then another bundle of G trains enters the yard [4]. Every iteration is a service slot.

Thus, some special situations are also considered in [4]:

(1) revisits, i.e. situations, when a train to have already been unloaded has to enter the transshipment yard again to be loaded with items to have been delivered after the first train's visit;

(2) split moves when a train i that carries a container dedicated to a train j and is served in a service slot t before a service slot t' of the train j.

A core decision of the transshipment yard scheduling problem (TYSP) involves assigning every train i of the given train set I to a service slot $t = 1,..., T$ [4]. In addition to [4], this article solves the problem of assigning every train to one of the tracks at each slot**.** At most G trains can be assigned to each slot t because G is a number of parallel railway tracks of the transshipment yard [4].

Let us construct the mathematical model of the problem in terms of combinatorial optimization.

Let's describe each time slot t using a tuple K^t containing an amount of all trains assigned to the slot t; we consider assigning every train to a certain railway track, their order is important, so $K^t = (k_1^t, k_2^t, ..., k_g^t, ..., k_G^t)$. Here $k_g^t \in I$ denotes an amount of trains assigned to the time slot t and located on a railway track $g \in G$.

It's worth noting that while forming a time slot, we are choosing G trains from $|I|$ possible ones, so that we are choosing K^t from a set of permutations $P_{|I|}^G$ (permutations of $|I|$ elements are taken from G at one time). So, choosing an optimal time slot K^t can be treated as the combinatorial optimization problem of choosing the optimal permutation from the set $K^t \in P_{|I|}^G$.

In this way, decision variables are time slots $K^t, t = 1, 2, ..., T$ to have been formed by the trains.

A decision result should meet three points:

1. A number of trains' revisits.
2. A cost of split moves for containers (which depends on assignment of trains to railway tracks) is to be minimized.
3. A cost of moves for containers between trains at the same time slot (which also depends on assignment of trains to railway tracks) is to be minimized.

The objective #1 is the same one as in [4]; the objective #2 is a more general case for the one described in [4] where an only number of split moves is considered; the objective #3 is a new one compared to [4] because trains' assignment to railway tracks is taken into account.

As described in [4], we have the multi-objective optimization problem (the objectives #1 and #2 exist here, but the objective #3 is new) and use linear scalarization to formulate the problem as the single-objective optimization one. Let's describe an objective function and constraints.

$$\alpha_1 \sum_{i \in I} y_i + \alpha_2 \sum_{t=1}^{T} \sum_{t'=t+1}^{T} \sum_{p=1}^{G} \sum_{q=1}^{G} z_{tt'} A_{k_p^t k_q^{t'}} C_{pq}^* + \\ + \alpha_3 \sum_{t=1}^{T} \sum_{p=1}^{G} \sum_{q=1}^{G} A_{k_p^t k_q^t} C_{pq} \to \min \tag{1}$$

$$z_{tt'} = \begin{cases} 1, if \ t < t' \\ 0, if \ t = t' \end{cases},$$

$$k_g^t \in I \quad \forall t = 1, 2, ..., T, \forall g = 1, 2, ..., G, \tag{2}$$

$$e_{k_g^t} \le t \le l_{k_g^t} \quad \forall t = 1, 2, ..., T, \forall g = 1, 2, ..., G, \tag{3}$$

$$k_p^t, k_q^{t'} : t \le t' + y_{k_q^{t'}} \cdot M, \tag{4}$$

$$y_i \in \{0,1\} \quad \forall i \in I \tag{5}$$

where $I = \{1, 2, \dots, N\}$ is a set of the trains (indices i and j) [4];
L_i is a set of the trains carrying containers dedicated to the train i [4];
T is a number of the time slots for trains' (un-)loading (an index t) [4];
G is a number of parallel tracks within the transshipment yard (an index g) [4];
$A = [A_{ij}]$, $i,j = 1,2,\dots, N$ is a number of containers the train i receives from the train j [4];
e_i is the earliest time slot the train i may be assigned to [4];
l_i is the latest time slot the train i may be assigned to [4];
M stands for a Big integer value (e.g., $M = T - 1$) [4];
$\alpha_1, \alpha_2, \alpha_3$ are given weights for the objectives #1-#3, $\alpha_1, \alpha_2, \alpha_3 \ge 0$;
y_i denotes a binary variable: 1, if the train i has to revisit the yard; 0, otherwise [4];
$K^t = (k_1^t, k_2^t ..., k_G^t) \in P_{|I|}^G$ is the time slot t formed by G trains assigned to the railway track;
C_{pq} determines a cost of picking a container from the source train on a track p and dropping it to the target train on a track q, if the trains are served within the *same time slot*;
C_{pq}^* is a cost of picking a container from the source train on a track p and dropping it to the target train on a track q, if the trains are served at *different time slots* (a cost of the split move).

Costs C_{pq} and C_{pq}^* are *predefined* values that can be calculated once (during an initialization procedure).

$$C_{pq} = c_{pq} \tag{6}$$

$$C_{pq}^* = c_{p0} + c_{0q} \tag{7}$$

Here c_{pq} is a constant cost of moving a container from the track p to the track q.

The storage area is denoted by a fictitious track 0. If the container is moved directly, the movement cost C_{pq} is just a cost of the direct container move c_{pq}. But if the trains are served at different time slots, then, first of all, a container should be moved from the source train to the storage area (c_{p0}) and later from the storage area to the target train (c_{0q}), so that $C_{pq}^* = c_{p0} + c_{0q}$.

All constants c_{pq} depend on the specific transshipment yard. In the simplest case, if the cost depends only on a distance between railway tracks, and neighboring tracks are located approximately at the same distance, c_{pq} can be calculated as $|p - q|$.

The condition (3) ensures that each train is served within the acceptable time slot [e_i; l_i].

Similar to the condition (4) in [4], our condition (4) ensures that the source train k_p^t arrives before the target train $k_q^{t'}$ (but if the target train revisits, then the condition is always satisfied due to a right summand).

III. A Solution Algorithm

We use the *beam search procedure* from [4] with an only distinction that we arrange the trains while forming every slot in order to minimize a sum of the move costs:

- between the trains within the current slot (a *direct move cost*);
- from the trains at the current slot to the staging area (if the trains have containers for other trains that are not at the current slot) – a *split move begin cost*;
- from the staging area to the trains at the current slot (if the staging area has containers for the trains at the current slot) – a *split move end cost.*

For now, minimization of the move costs is achieved by a simple exhaustive search over all permutations of the trains.

Let us briefly recall main steps of the algorithm [4]:

1. At the first step, t=0 and we have no schedules yet. It means that we have an empty node in an acyclic graph [4]. Let us call it a parent node.
2. Increase t and generate a set of possible time slots $K^t = (k_1^t, k_2^t ..., k_G^t) \in P_{|I|}^G$.
3. Calculate a value of (1) for each generated slot and select the best BW (a beam width and a predefined parameter) in terms of (1).
4. For each BW selected slot, build a new node in the acyclic graph [4] which is located under the parent node and recursively call the step 2 with setting this node as the parent one.

Example. Given G=2 tracks, 4 trains I ={1,2,3,4} and a beam width BW=2. The acyclic graph from [4] (in some different form: a node text indicates the current slot instead of the trains already scheduled up to the current slot), produced by the solution algorithm, is depicted in Fig.2. Nodes selected by the beam search are marked in a bold font; some nodes are also replaced by '…':

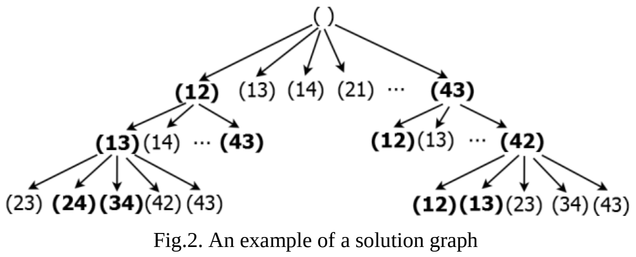


Fig.2. An example of a solution graph

The top of the graph is here the beginning t=0.

The next level t=1 represents a few possible variants of slots, i.e. (12), (13), …. (43). It should be noted that the slots (12) and (21) are different because train arrangement is important. For example, the train 2 has to unload containers to the staging area (which is considered as the track 0 which is closer to the track 1) and therefore it makes sense to place the train 2 to the track 1, which means that the slot (21) has a lower cost compared to (12).

Having generated all $|P_4^2| = 12$ possible slots K^1 , the beam search procedure selects BW=2 slots/nodes with the best value of (1) for expanding. Other slots/nodes are excluded from further consideration. Let's assume that we selected slots (12) and (43).

IV. Computational Results

We implemented the solution algorithm in Python 2.7. A developed application is available online at http://tsy-litvinenkoapps.rhcloud.com/. Please take a look at screenshots of an example of input data and its solution in Figs. 3, 8, 9, 10.

Computational experiments were carried out to show how train arrangement impacts a solution quality and time. We generated a bunch of test instances. Every instance is a combination of following input parameters:

- 9 options of a train count N=6,7, … ,14;
- 4 options of a track count G=2,3,4,5;
- 4 options of a track-track move cost:

$$c_{pq} = cost\_coef * |p - q|,\ cost\_coef = 1, 10, 50, 100;$$

- 5 various numbers of the target trains for each train: when each train carries containers for 1,2,3,4,5 other trains; in other words, a various sparseness of a matrix A: $\sum_{i=1}^{N} A_{ij} = 1, 2, 3, 4, 5,\quad j = 1, 2 ... N$. Let us denote this parameter as *cargos_per_train*.

Objective weights were taken equal for all instances $\alpha_1 = \alpha_2 = \alpha_3 = 1$.

We solved each of 9*4*4*5=720 instances twice: without train arrangement (solving only the level 1 issue described in [4]) and with train arrangement (solving also the level 2). For every instance, we calculated a relative time increase and a relative cost decrease of the solution with arrangements compared to the solution without arrangements:

$$\text{cost\_decrease} = (c_1 - c_2)/c_2, \tag{8}$$

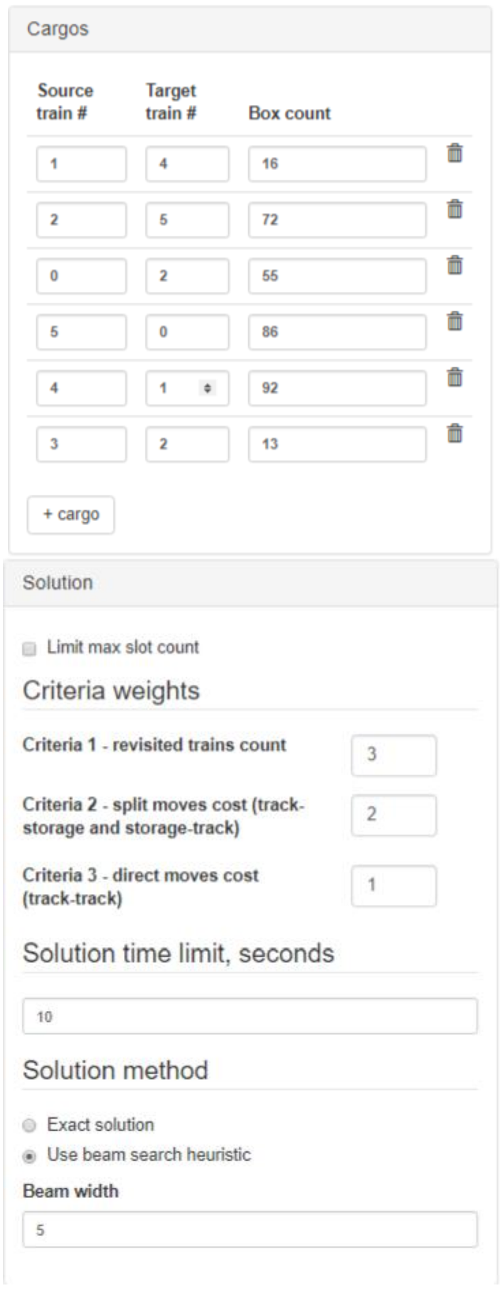
$$\text{time\_increase} = (t_2 - t_1)/t_1 \tag{9}$$

where c_1 and c_2 determine values of the expression (1) for the solutions with and without arrangements; t_1 and t_2 define the solution time for the solutions with and without arrangements respectively.

One can find a full set of the solution data at https://goo.gl/D36bf7.

We have analyzed in what way each described input parameter may impact both the time increase and the cost decrease. Results are depicted in Figs. 4-7. Y-axis in all diagrams contains logarithms of *cost_decrease* and *time_increase* (*y=lg cost_decrease*, *y=lg time_increase*), while X-axis contains values of a specific input parameter.

The diagrams show that *G* and *cargos_per_train* directly impact both the cost decrease and the time increase while *n* generally impacts the time increase and *cost_coef* does not seem to impact either the time increase or the cost decrease.



Cargos

Source train #	Target train #	Box count
1	4	16
2	5	72
0	2	55
5	0	86
4	1	92
3	2	13

+ cargo

Solution

☐ Limit max slot count

Criteria weights

Criteria 1 - revisited trains count	3
Criteria 2 - split moves cost (track-storage and storage-track)	2
Criteria 3 - direct moves cost (track-track)	1

Solution time limit, seconds

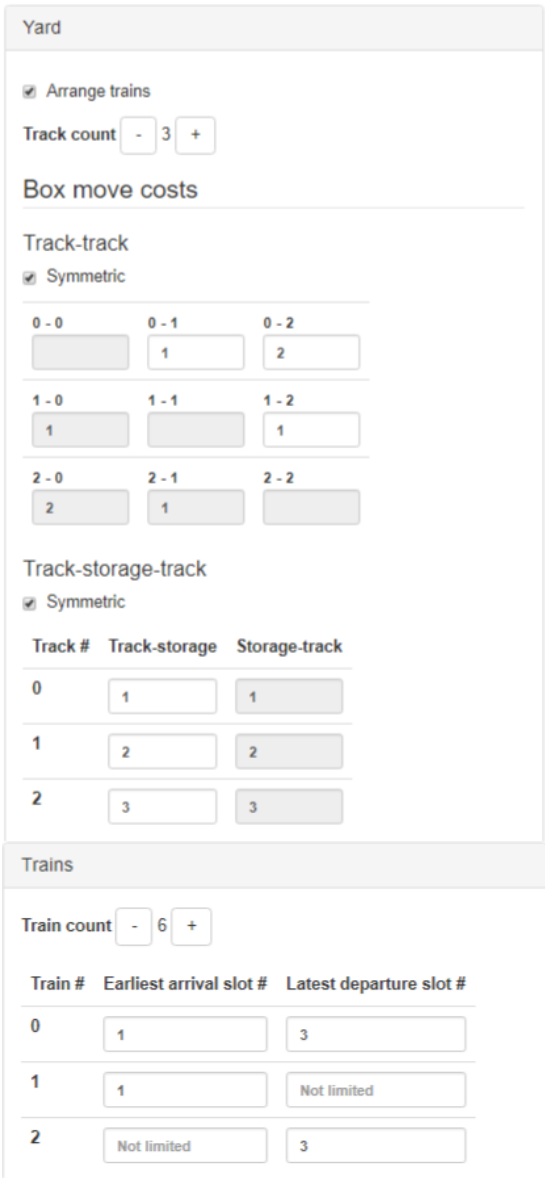
10

Solution method

○ Exact solution
● Use beam search heuristic

Beam width

5



Yard

☑ Arrange trains

Track count - 3 +

Box move costs

Track-track

☑ Symmetric

0 - 0	0 - 1	0 - 2
	1	2
1 - 0	**1 - 1**	**1 - 2**
1		1
2 - 0	**2 - 1**	**2 - 2**
2	1	

Track-storage-track

☑ Symmetric

Track #	Track-storage	Storage-track
0	1	1
1	2	2
2	3	3

Trains

Train count - 6 +

Train #	Earliest arrival slot #	Latest departure slot #
0	1	3
1	1	Not limited
2	Not limited	3

Fig.3. A screenshot of the developed application (input data)

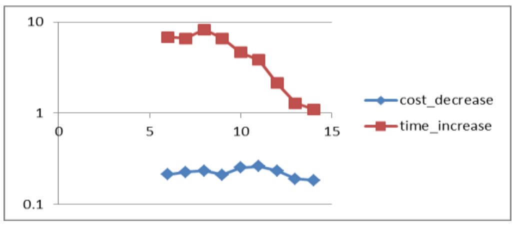


Fig.4. cost_decrease and time_increase (y axis) for various N (x axis)

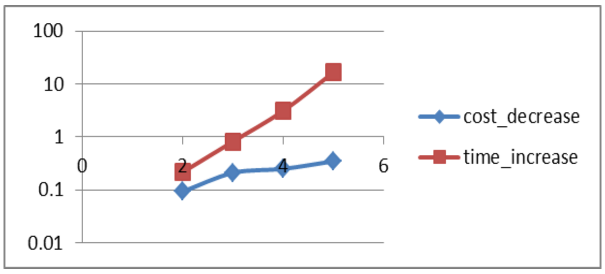


Fig.5. cost_decrease and time_increase (y axis) for various G (x axis)

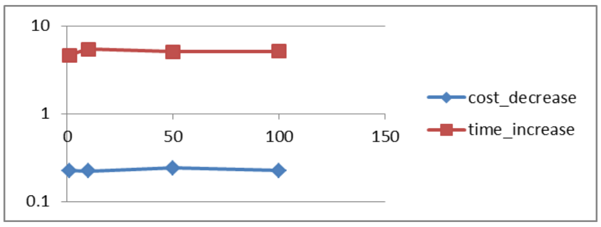


Fig.6. cost_decrease and time_increase (y axis) for various cost_coef (x axis)

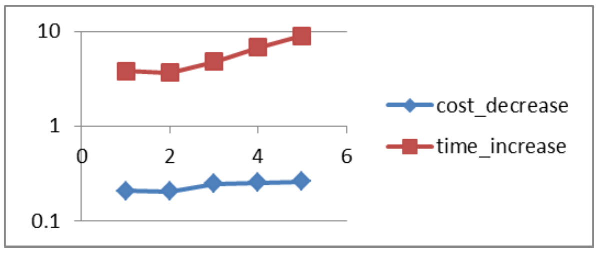


Fig.7. cost_decrease and time_increase (y axis) for various cargos_per_train (x axis)

V. Conclusion

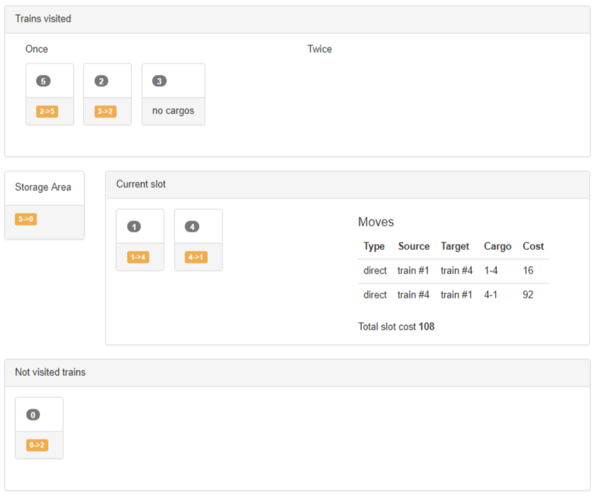
This article solves the second level of the transshipment yard scheduling problem, which includes assignment of the trains to the certain tracks.

The mathematical model and the solving method have been given for the described problem.

The article also describes TYSP in terms of the combinatorial optimization instead of using Boolean variables.

In computational experiments, we figured out that the solving problem for the second level allows decreasing the total schedule cost. However, more computational time is required to arrange the trains to the tracks.

The algorithm of the train arrangement could be improved in order to decrease computational time.


Trains visited

Once

Twice

5 2->5

2 3->2

3 no cargos

Storage Area

5->0

Current slot

1 1->4

4 4->1

Moves

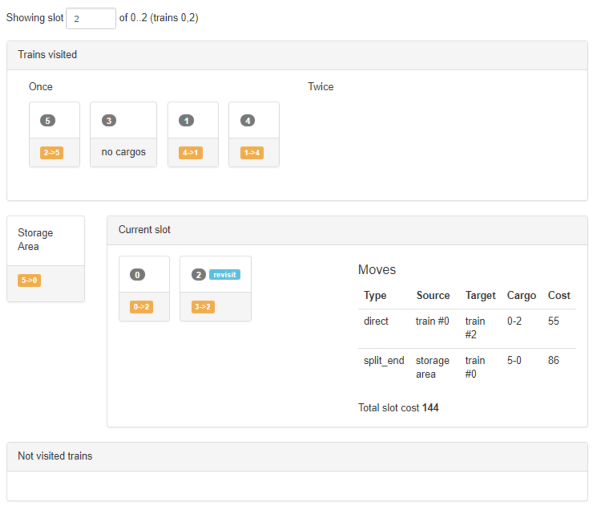
Type	Source	Target	Cargo	Cost
direct	train #1	train #4	1-4	16
direct	train #4	train #1	4-1	92

Total slot cost **108**

Not visited trains

0 0->2

Fig.8. A screenshot of the developed application (its solution for the second slot)


Showing slot 2 of 0..2 (trains 0,2)

Trains visited

Once

Twice

5 2->5

3 no cargos

1 4->1

4 1->4

Storage Area

5->0

Current slot

0 0->2

2 revisit 3->2

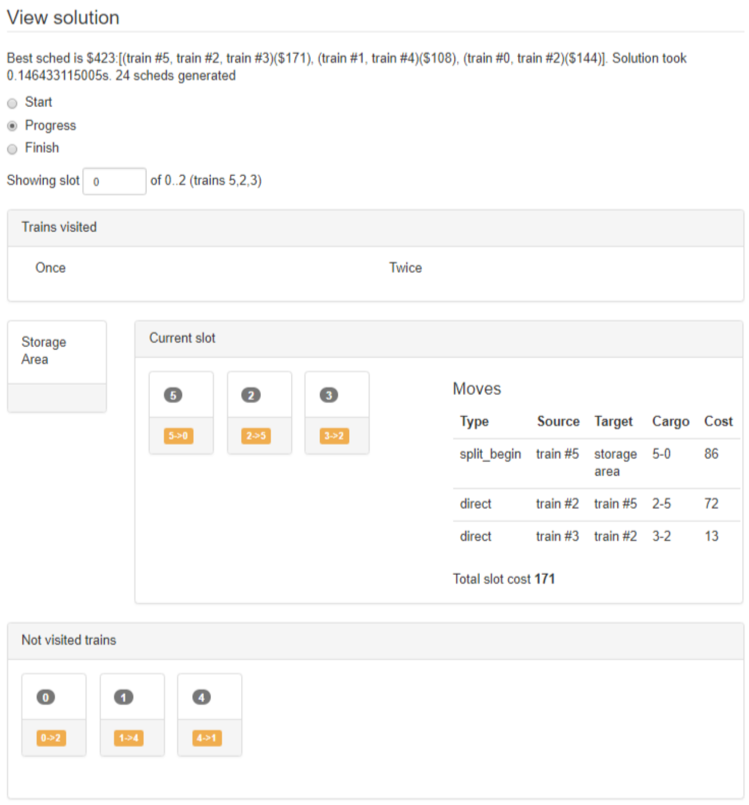
Moves

Type	Source	Target	Cargo	Cost
direct	train #0	train #2	0-2	55
split_end	storage area	train #0	5-0	86

Total slot cost **144**

Not visited trains

Fig.9. A screenshot of the developed application (its solution for the third slot)



View solution

Best sched is $423:[(train #5, train #2, train #3)($171), (train #1, train #4)($108), (train #0, train #2)($144)]. Solution took 0.146433115005s. 24 scheds generated

Start [✓ Progress] Finish

Showing slot 0 of 0..2 (trains 5,2,3)

Trains visited

Once | Twice

Storage Area

Current slot

Moves

Type	Source	Target	Cargo	Cost
split_begin	train #5	storage area	5-0	86
direct	train #2	train #5	2-5	72
direct	train #3	train #2	3-2	13

Total slot cost **171**

Not visited trains

Fig.10. A screenshot of the developed application (its solution for the first slot)

References

[1] T. Crainic and K. Kim, "Intermodal transportation", Handbooks Oper. Res. Manag. Sci., Vol.14, pp.467–537, 2007.

[2] J.-F. Cordeau, P. Toth, and D. Vigo, "A Survey of Optimization Models for Train Routing and Scheduling", Transp. Sci., Vol.32, pp.380–404, 1998.

[3] N. Boysen, M. Fliedner, F. Jaehn, et al., "A Survey on Container Processing in Railway Yards", Transp. Sci., Vol.47, pp.312–329, 2013.

[4] N. Boysen, F. Jaehn, and E. Pesch, "Scheduling Freight Trains in Rail-Rail Transshipment Yards", Transp. Sci., Vol.45, pp.199–211, 2011.

[5] N. Boysen, F. Jaehn, and E. Pesch, "New bounds and algorithms for the transshipment yard scheduling problem", J. Sched., Vol.15, pp.499–511, 2012.

[6] M. Barketau, H. Kopfer, and E. Pesch, "A Lagrangian lower bound for the container transshipment problem at a railway hub for a fast branch-and-bound algorithm", J. Oper. Res. Soc., Vol.64, pp.1614–1621, 2013.

[7] P.K. Agrawal, M. Pandit, H.M. Dubey,"Improved Krill Herd Algorithm with Neighborhood Distance Concept for Optimization", International Journal of Intelligent Systems and Applications(IJISA), Vol.8, No.11, pp.34-50, 2016.

[8] H.A.R. Akkar and F.R. Mahdi,"Grass Fibrous Root Optimization Algorithm", International Journal of Intelligent Systems and Applications(IJISA), Vol.9, No.6, pp.15-23, 2017.

[9] S.P. Singh and S.C. Sharma,"A Particle Swarm Optimization Approach for Energy Efficient Clustering in Wireless Sensor Networks", International Journal of Intelligent Systems and Applications(IJISA), Vol.9, No.6, pp.66-74, 2017.

Authors' Profiles

Igor Grebennik was born in 1966. He is D.Sc., professor, Chair of Systems Engineering Department at Kharkiv National University of Radio Electronics. I.Grebennik is an author of more than 180 publications and eight books.

Scientific interests: Combinatorics, Combinatorial Generation, Combinatorial Optimization, Combinatorial Optimization Problems of Placement of Objects, Mathematical Modeling, Vehicle routing problems.

Rémy Dupas was born in 1961. He is a professor at University of Bordeaux. Scientific interests: operational research, combinatorial optimization, industrial engineering, production and transportation problems (scheduling, supply chain planning and vehicle routing).

Oleksandr Lytvynenko was born in 1992. He graduated from Kharkiv National University of Radioelectronics (Automatics and Computer Technologies Faculty, specialization System Engineering) in 2013. He is currently a Ph.D. student and a web developer.

Inna Urniaieva was born in 1969. She is an assistant professor at Systems Engineering Department in Kharkiv National University of Radio Electronics.

Scientific interests: Combinatorial Optimization Problems of Placement of Objects, Mathematical Modeling, Vehicle routing problems.

How to cite this paper: Igor Grebennik, Rémy Dupas, Oleksandr Lytvynenko, Inna Urniaieva, "Scheduling Freight Trains in Rail-rail Transshipment Yards with Train Arrangements", International Journal of Intelligent Systems and Applications(IJISA), Vol.9, No.10, pp.12-19, 2017. DOI: 10.5815/ijisa.2017.10.02