https://doi.org/10.26552/com.C.2015.2.23-29

Matyas Koniorczyk - Borbala Talas - Ferenc Gedeon *

PRECONDITIONING IN THE BACKTRACKING DUTY GENERATION OF PASSENGER RAIL CREW SCHEDULING: A CASE STUDY

We describe briefly the crew scheduling and rostering approach implemented in Railm@n, the system used by MAV START, the passenger railway transport company of Hungary, to organize the work of passenger train crews that is, conductors. Then we discuss the scheduling (duty generation) phase of the algorithm in detail. When treated in full generality, the problem already scales to an untractable size. We describe our successful experience with the use of preconditioning to keep the problem tractable. The approach may be useful in timetable planning and depot planning, too.

Keywords: *Passenger crew rostering, duty generation.*

1. Introduction

Passenger railway crew scheduling and rostering, that is, the planning of the work of employees that form the crew of passenger trains, is a central problem in passenger railway operations. It has an extensive literature in operations research. Reference [1] contains an exhaustive collection of the problem's scientific background. As passenger railway transportation is of utmost importance both as fundamental requirements of mobility, and railway companies provide significant opportunities for employment, the problem has a high societal impact.

There are two main challenges in the topic of crew scheduling and rostering. On the one hand, the topology of the railway network, the structure of the passenger timetables and the applied technologies, as well as the legal context (employment regulations) may differ radically in the case of each railway company, giving rise to different models. For instance, in countries where the duration of train trips can last for even multiple days (see e.g. Ref. [2]) the problem is very different from the typical Central European situations where trips can range from less than one hour to few hours. Also, there is a variety of possible optimization objectives. In a recent paper (Ref. [3]) the aim is to design fair rosters with a judicious distribution of popular and unpopular duties, while in this study the final goal is to save cost and reduce overtime shifts. On the other hand, the algorithmic solution of the arising models and even their subproblems generate computationally hard issues, like large-scale set covering problems [4] or the hereby studied graph search.

Algorithmic optimization is also a tool in long-term planning such as setting up or abandoning crew depots, and planning their recruitment schedules. The preparation of passenger timetable plans also require an analysis of feasibility with respect to feasibility subject to the given set of human resources.

In the present paper we discuss our approach to a subproblem arising in Hungarian passenger railway transportation. MAV START, the Hungarian passenger railway company runs around 3000 trains daily. Besides the train drivers, whose work is organized by the traction division of the company, almost all trains have traveling crews, that is, different types of conductors. The company has around 3000 employees for this task. We consider the planning of their work.

The planning process has three different scopes: long-term (yearly) planning, monthly planning and operational planning. In the yearly planning phase the plan of the timetable of the period, that is, the set of planned train paths and a version of the rolling stock circulation plan is given. (This latter, however, is not always available at the time when the planning is made.) Based on this information, rosters (sequences of daily duties) are designed for each crew depot. In the monthly planning phase, a group of suitable employees is assigned to the roster, and periodic rostering is made. This means that if the roster is n days long, it is assigned to n people so that the pattern is shifted one day for each employee. Shifting is done modulo the number of days in the month. By this *Latin square* construction, each daily duty of the roster is realized once. The number of the people assigned to an actual roster is set so that n is the 2/3 of the actual value,

* [1,2]**Matyas Koniorczyk,** [1]**Borbala Talas,** [3]**Ferenc Gedeon**
[1]Institute of Mathematics and Informatics, Faculty of Natural Sciences, University of Pecs, Hungary
[2]Rail Navigator Kft., Budapest, Hungary
[3]MAV-START Zrt., Budapest, Hungary
E-mail: kmatyas@gamma.ttk.pte.hu

thus there are people to whom no duty is given initially. During the planning phase this basic monthly schedule is edited in order to take into account illnesses, holidays, special duties such as education, etc. During this phase all the crew members are given valid duties for the month. Finally the operative planning consists of handling and logging situations arising within the month, and generating payroll data at the end of it.

In the year 2010 MAV START introduced a newly developed IT system, named „Railm@n" to support the described process, and also the organization of station crews (cashier staff, technical personnel, etc.) The main intention was to handle the processes in a system based on a central database. However, the need for an algorithmic optimization at certain points was also apparent. The present results are due to this development.

Besides the already mentioned extensive theoretical background, there are pieces of software on the market which have advanced capabilities for crew rostering problems. In spite of this, since the problem depends very much on the legal circumstances concerning the employees as well as the characteristics of the topology of the network and the timetable, there is no out-of-the-box solution which is suitable for the needs of a particular railway company as it is. It is also an important point that the algorithmic optimization tools should be an integrated part of crew rostering systems, and should support certain steps of manual design operations due to subjective or non-modelable issues.

Our results presented here relate only a subproblem, namely the duty generation for rostering. The idea of implementing a duty generator came from Reference [5]. First we describe the structure of input data and the role of the algorithm. We also outline the means of use of the generator's output. Then we present the details of the algorithm and the experience with its scaling under certain circumstances. Finally the results are summarized and conclusions are drawn.

2. Input data for crew scheduling and rostering

The main input of the algorithm is the timetable, which determines in time and space the tasks the crews have to perform. The timetable contains train paths and regularity prescriptions which define the days on which a train path is realized. The timetable is valid for a period of one year, but it is a subject of various changes during its validity period. (These are due to seasonal traffic in certain regions, track maintenance, etc.) The days on which a particular train runs depend on the day of week, and non-periodic conditions (e.g. a train does not run on particular dates) may also be imposed.

Instead of going into the details on the regularity prescriptions we restrict ourselves to the design of a periodic roster of an average workday. In practice the handling of e.g. the traffic on the weekend is handled by designing parallel versions of daily duties for a given day of the roster that belong to a particular weekday. These are always considered as alternatives for the basic version of the given duty, which is designed for an average workday (that is, not the first or the last working day of the week). Our task is to define the sequence of the basic daily duties that constitute the base roster. We remark here that in practice the periodicity of a train path does not necessarily coincide with the periodicity of the related tasks. It may occur that though the given train runs everyday, the crew is provided by different depots on different days, or an additional crew member is needed during workday when the number of passengers is higher. These issues can be handled in the second step of roster design as additional conditions of the set covering problem.

Having imposed the restriction that we generate the basic duties from which a basic roster can be selected, we are given particular train paths, defined in time and space, for an average day to be served. The required number and type of crew members on a train can be determined according to the rolling stock circulation plan, which, on the other hand, is designed according to the available rolling stock and expectable passenger flow. If this information is not available at the time of roster design, the number and type of crew members can be estimated manually.

Another issue is that a given crew member does not necessarily serve the train during its whole trip. Hence, the train paths maybe broken into parts which may be served by different personnel. This may either be treated algorithmically or manually. For the calculations in the present paper we assume that each train requires a single crew member of the same kind, and the crew travels from terminal to terminal. It is quite easy, however, to include the general scenario into our approach: essentially we need a set of trips with given starting and ending points in time and space, with a specification of the skills of the required crew member. In the simplified model studied in this paper, all the crew members have the same skills and they travel from terminal to terminal. Importantly, this input may also be generated from the public timetable for experimental purposes, which makes our caclulations reproducible without using company confidental data.

As supplementary inputs for our algorithm, we have information about the type of the train, and the division of the railway network to lines. There are several kinds of the trains such as local slow trains, inter-regional fast trains, intercity trains, suburban commuter trains, etc. The division of the network to lines is mainly defined by historical and technical considerations, however, it is typical that there is a set of trains commuting between the ends of lines. There are trains whose trip includes, in whole or in part, multiple lines. Our main intention in the present paper is to demonstrate that a filtering of the search space according to permitted or not permitted train types or lines can limit the number of possible daily duties of a depot to a tractable magnitude.

A specialty of the Hungarian railway network is its single-centered structure. Almost all main lines start from Budapest, and the possible transitions between lines that avoid Budapest is

limited. This star topology, illustrated in Fig. 1 is highly reflected in the behavior of the number of possible rosters.

Based on all the input described above, our task is to find preferably all possible daily duties for a depot. We do this by a backtracking algorithm which follows the ideas described in Reference [5]. Since the number of the possible duties can be very high, we should discard some of them by preselection. The output of the algorithm will be the basis of the next phase of the roster design, in which we choose duties to be realized. This is done by solving a set covering problem in which the objective function might be, e.g. the overall worktime, which we need to minimize. As conditions we may prescribe that given trains should be served by given depots, or we may give lower and upper bounds for the selected rosters. All this can be done either with 0-1 programming or specialized algorithms for set covering. This part of the planning is, however, not the subject of the present paper. In spite of that we should keep in mind that the designed duties will become variables of a 0-1 program, so their number should be kept at a reasonable level.

Our main result here will be a demonstration of the fact that *preconditioning*, that is, making reasonable restrictive constraints already in the duty generation phase can indeed control the algorithm to produce reasonable results, keeping it just at the borderline of the explosion due to its otherwise bad scaling. So let us turn our attention to the details.

3. Details of the duty generation algorithm

From a mathematical point of view, the trips to be performed by the crew members can be considered as the vertices of the directed graph, the edges of which represent the fact that a trip at the target vertex of the given edge can be performed after the one at its source vertex. Apart from the trivial constraint that the previous trip should end at the station where the next one starts from, one has to keep in mind the additional technological times to be elapsed between two adjacent trips (such as e.g. the preparation of the train by the crew, time which depends on the kind of train, etc.). In addition, there is an ordering of the edges emanating from an edge according to the starting time of the trip corresponding to the other end of the edge: they should be considered one after another ordered according to the starting time of the following trip.

Having set up this search graph, one can implement a backtracking algorithm very much like the eight-queens problem known from basic textbooks. Given home depot and the earliest starting time of the duty, we start from a node representing the earliest feasible trip. The fundamental condition that describes a complete duty is that the ending point of the actual edge should represent a trip ending at the home depot.

There are, however, certain additional considerations to be taken into account in the algorithm. The main issues are:

- Additional tasks such as duty starting activities and short breaks prescribed by the legislation have to be considered. This can be done after producing a sequence of trips. If, for instance, the breaks cannot be included according to the rules, the duty is invalid.
- The time duration of the duties have a minimum and maximum duration. Legally it is between 6 and 16 hours, but we shall see that in certain cases it is beneficial to consider a shorter interval. This constraint gives rise to a trick in the algorithm leading to significant speed-up: for each search, edges that trivially lead to too long duties are a priori omitted.
- For stability reasons (that is, the robustness of plans with respect to train delays), *pairs of trains*, that is, trips to and fro the home depot after each other, are preferred. In some cases, however, it is allowed to include, e.g. a trip to and fro a third station from the first touched station after the depot. This is modeled by a parameter d, so that after at most d+1 trips from the depot, the depot should be reached. This will be termed as the *maximum distance from the depot*. For d=1, for instance, trips to and fro the depot will only be considered.

The so designed algorithm generates daily duties for each depot. These can be either used as the basis of a set covering problem ensuring that we chose sequences of them covering given trips. Or they can be used in the manual design of the rosters: the designer can pick them as templates form a „palette" and insert them into the designed rosters. Of course, parameters of the generated duties, such as their total duration and *efficiency*: the ratio of productive tasks to unproductive ones (e.g. waiting for the next trip) are evaluated as they are present in the objective function and can be used by the designer for filtering.

The algorithm, as we shall see, may explode. Therefore we use certain *preconditionings* to keep the run time and the number of generated duties reasonable. Before going to the detailed examples, let us devote a few sentences to the implementation of the algorithm.

4. Implementation

The automated design part of the software system is implemented in the Python programming language (ver. 2.7.6). It can get its input from either the client of the „Railm@n" system which has 3 tiers, an Oracle (TM) database at the bottom and a thick client (Windows executable) on the other hand. The latter can invoke the Python program. It also supports, however, the operation from saved local files containing serialized data structures and objects. These can also be obtained from a network server forming a middleware between the Python client and the database. In this layer, certain preprocessings can be done.

As a principle we find that even though it is tempting to prefilter some data already in the database layer, relying on the power of SQL queries, it appears that the raw collection of

data and the filtering done by Python's tools (list filtering and functional programming solutions) is more effective. The total amount of input data requires 1 gigabytes of memory, so it is indeed viable and effective to keep all the data in the memory of the client. The collection of the data takes a few minutes only, while the effective algorithm is fast.

A specific issue is the possibility of filtering tasks related to railway lines. A line is labeled by a number and it is a sequence of adjacent stations of the network which form an operational and/or technological unit. (E.g. line 40 is the Budapest-Pecs line). While many trains run within a line, some of them run on a part of a line or on (parts of) several different lines. We found that for an efficient prefiltering, we have to be able to decide the percentage of the stops of the trip the task is related to, which are located along the given line. In the particular case it means at least around 300.000 records storing (train, railway line, percentage) triplets: for about 3.000 trips and about 100 numbered railway lines, we have to evaluate the percentage of the stops of the given trip lying on the given railway line, the evaluation of which is rather slow both in database and in Python. We may, however, solve the problem by evaluating the data only once and storing the result. The evaluation takes a couple of tens of minutes, but the time required for using them is practically negligible. The generated data are sparse: a typical trip stops only on 1-3 lines, and the zero percentages need not be stored. (We remark, however, that in a realistic situation there are more duties than trains, as the crew of a train may have several members and they can change underways. Also, the exact percentage may depend on the version of the train path valid on the given date.)

5. Results

As the input of the algorithm we use data of the currently valid 2013-2014 passenger timetable of MAV START. The calculations were performed on an up-to-date, but low-end computer with an Intel® Pentium® CPU N3520 @ 2.16GHz, having two cores, two-threaded each. The computer has 4 gigabytes of memory, of which about 1.3 gigabytes were used by the process performing the calculation. The computer runs Linux Mint 17 so it can be considered as an average workstation. On more powerful machines the calculation times mightbe shorter.

As for the search space, we consider a single kind of conductor (in practice there are four types of them with a given hierarchy, and multiple members can form the crew of the given train). We assume that a conductor travels between the two terminals of the train. In practice the trip might be shared between depots, which means that the actual trips should be divided to multiple trips at certain stations in between. We consider a single chosen version of each train path, and do not consider different regulations of the traffic depending on the day of the week. These simplifications provide a more-or-less realistic problem showing the main characteristics, and many of the practically used duties are indeed generated. As a further simplification we do not consider night duties. These can be taken into account by adding a set of trips for the second day to the set of trips which we omit here. This explains why the generated duties start typically before the early afternoon.

The above simplifications can be easily overcome in the framework, and it is indeed done so in the practical implementation of the algorithm, integrated to the actually used software system. For demonstrational purposes, however, the simplified data are easier to understand. In addition, this makes the present calculations reproducible entirely from publicly available data.

As for the possible prefilterings, we consider the following:

- We may limit the tasks to certain railway lines. This is a traditional attitude in the everyday practice of manual design. To illustrate the lines and locations of the depots we will use in our examples, we have included a map of the Hungarian railway network in Fig. 1. Here we shall consider a train to belong to a railway line if at least 70% of its stops are on the given line.
- We may limit ourselves to given types of trains, e.g. suburban or InterCity (IC) trains only.
- We may limit the maximum distance from the depot. This is considered to be 4 by default, as this results in practically accepted and stable trip patterns (pairs, etc.).
- We may limit the minimum and maximum duration of the duties to a more stringent one than the one the legislation allows for.

The introduction of any of these preconditionings can have good reasons behind, and it is likely that they do not exclude duties which would benefit highly the roster design process. Let us now turn our attention to the practical cases we have studied.

Let us first consider a depot relatively far from Budapest. Pecs is located in south Hungary, the IC trains run around 3 hours to Budapest. The Pecs-Budapest line is No. 40, and trains of lines 65 (Pecs-Villany-Mohacs) and 60 (Pecs-Nagykanizsa) are also served partly from this depot. Figure 2 shows the timing of the calculation. In both subplots, the x axis displays the computational time in seconds. It is the system time we register each time when a valid daily duty is found and stored. On the primary (left) y axis there is the number of daily duties already generated, while the secondary (right) axis corresponds to the curves showing the starting time of the first trip of the given duty, measured in minutes from 0:00 a.m. of the given day. (The backtrack produces duties whose starting time does not decrease.) These figures reflect the progress of the duty generation. In Figs. 3 and 4 we shall use the same representation.

In each case we consider duties of length between 6 and 16 hours, which is the allowed interval according to the legal regulations.

On subplot "a" of Fig. 2 we can see all the possible daily duties of the depot, without any preconditioning. A total number

Fig. 1 The Hungarian railway network. (Based on Ref. [6]); lines in argument are colored differently from the others, the studied depots have red circles around them

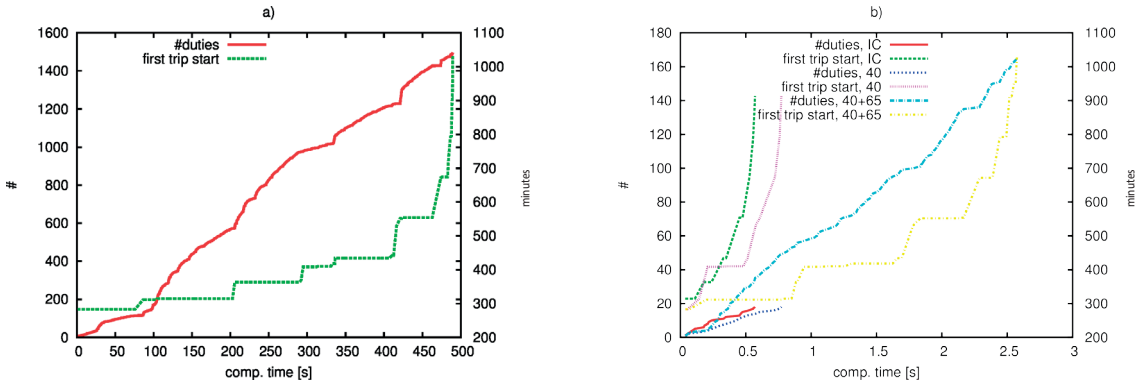

Fig. 2 The results for Pecs depot

of 1496 duties were generated in 8 minutes and 9 seconds. Remember that the set of duties to be realized are chosen either manually or by a 0-1 program in a next step (not discussed in the present paper), in which the number of generated duties will be the binary decision variables. This running time and number of variables is perfectly acceptable if we are planning for this single depot. If, however, a network-wide covering problem is addressed, this may result in too many variables. Also, for a single depot with manual planning, this is a relatively large number of duties to be considered. Hence we make certain prefilterings. Three possibilities were checked, the details of their characteristics are included in Table 1. In subplot "b" of Fig. 2 we can see the characteristics of three kinds of prefilterings: InterCity trains only, all passenger trains of lines 40 and 65, all passenger trains of line 40 only. We can see that the number of generated duties as well as the calculation time is radically decreased. However, for InterCity-only duties the number is probably too small, if we let the staff handle lower-rank trains, too, there are definitely more options for a duty. Since InterCities run on line 40, the calculation time is similar to that of all trains running on line 40, and the number of the generated duties is (accidentally) the same. Serving lines 40 and 65 together is frequently used in practice, and indeed, it generates a number of duties which is satisfactory for all purposes. So for this depot we find that though we have

Summary of the results

Table 1

Depot (and abbrev.)	Restriction	Max. dist. from depot	Min. duration [hours]	Max. duration [hours]	# possible trips	Calc. duration [min:sec]	# of duties generated
Pecs (PS)	none	4	6	16	3212	08:09	1492
Pecs (PS)	InterCity only	4	6	16	150	<00:01	18
Pecs (PS)	Lines 40 and 65 only	4	6	16	134	00:02	165
Pecs (PS)	Line 40 only	4	6	16	100	<00:01	18
Szolnok (SL)	none	4	6	16	3212	> 20:00	> 30 000
Szolnok (SL)	line 120a only	4	6	16	117	06:54	25 176
Szolnok (SL)	line 120 and 120a only	4	6	16	187	08:00	26 322
Szolnok (SL)	suburban trains only	1	6	16	1232	01:02	2 932
Cegled (CV)	lines 100a and 140	4	6	16	182	08:50	28 106
Cegled (CV)	lines 100a	4	6	16	129	08:14	27 716
Szekesfehervar (SV)	lines 30a, 42 and 44 only, suburban trains only	4	8	10	108	09:51	637
Szekesfehervar (SV)	lines 29, 30, 30a, 42, and 44 only	4	8	10	288	18:20	895

results which are usable even without preconditioning, a proper prefiltering makes the algorithm even more effective.

Our second example is the Szolnok depot. As it appears from the fifth line of Table 1, without any preconditioning, the run time and the number of generated duties is unacceptable. The reason is, that from Szolnok there are frequent suburban trains of short trip time to two Budapest head terminals (Keleti and Nyugati). As the topology of the network is highly Budapest-centered, for Keleti and Nyugati there is an enormous number of possible duties. These come into play for Szolnok as well.

If, however, we restrict ourselves to the trains of line No. 120a, which is the Budapest-Szolnok suburban line, we obtain a tractable, though still too large (25 176) number of generated duties in an acceptable calculation time. If we do not restrict ourselves to the suburban part of the line, but also allow trains on the whole line (120, Budapest-Lokoshaza), the behavior is very similar. For this behavior we evaluated another example: Cegled, with respect to the suburban line 100a, versus 100a and the long-distance line 140, produces a very similar behavior (see lines 9 and 10 of Table 1). Returning to the case of Szolnok, as illustrated in line 8 of Table 1, if we do not restrict to lines at all, but we consider suburban trains running from and back to Szolnok only, we obtain a very reasonable number of 2 932 tasks within around a minute. As these trains are typically served by a given group of employees, this is a very reasonable assumption. The time behavior of the calculations is illustrated in Fig. 3. The diagram should be read in the same way as described in the case of those for Pecs (Fig. 2).

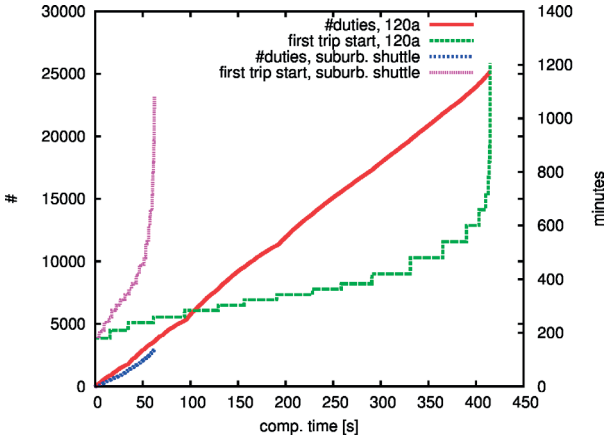

Fig. 3 Time behavior of calculations for Szolnok depot

Finally let us discuss the case of Szekesfehervar depot. Due to the frequent and fast availability of Budapest-Deli and the large amount of available trains, all the already discussed strategies to keep computation time and the generated number of duties reasonable fail in this case. If, however, instead of searching for duties between the duration between 6 and 16 hours, we consider a minimum of 8 and a maximum of 10 hours of duty duration, the number of the generated duties will become very reasonable. The running time will not decrease as much as in the previously described case, but the achievable time of about 10 minutes for suburban trains only, and 20 minutes for a broader set of train types and lines (including long-distance trains) is still acceptable. This is illustrated in the last two lines of Table 1, as well as in

Fig. 4, in the same way as previously. It is interesting to observe that while in the previous examples the number of generated duties grows mainly linearly with the computational time, in this case there is a fast increase in the valid duties found at the end of the computational period.

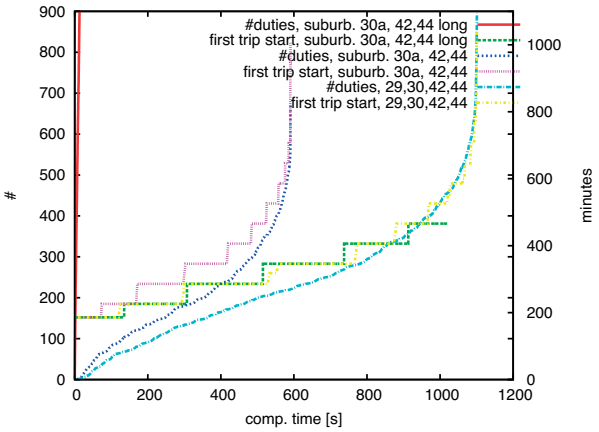

Fig. 4 Time behavior of calculations for Szekesfehervar depot

From the presented cases it is apparent that the means of preconditioning are proper ways of making the automated duty generation useful. However, the expertise of the staff and some experimentation is still needed in order to find the proper conditions, as well as to evaluate the generated duties with respect to their quality in order to assess whether the conditions are effective and not too restrictive.

6. Conclusions

In conclusion, we have presented our experience with preconditioning in backtracking duty generation for the crew scheduling problem of conductors at MAV-START, Hungary. The considered algorithm is used in practice, and it provides duties to aid manual planning as well as the input of an automated optimization of rosters by solving a set covering problem. The introduced examples illustrate that certain reasonable means to restrict the scope of trips served from certain depots, which are also used in manual planning, indeed makes the otherwise badly scaling computation tractable and useful.

The present results are side-effects of the development of the currently used crew scheduling and rostering software systems of MAV-START and they are being continuously integrated to this solution. By integration we mean that they are accessible for the designers both in system's manual and automated planning functionality.

Besides the planning of rosters for actual depots, the automated algorithm can be used in several ways. For a new timetable plan, the required number of employees can be estimated, and the compatibility of the new timetable plan with the present number of employees can be analyzed. It may be checked, for instance, if there can be efficient duties defined for given depots according to the plan (currently the timetable is planned well before the planning of crews and there is no feedback at all.).

Given a set of valid tasks, the duty generation for a given station can reveal properties of the duties that can be generated for the station. Examining their efficiency and other characteristics, or organizing them into possible rosters may be helpful in really long-term planning tasks such as opening new depots, planning the recruitment policy of new personnel, or eliminating certain depots.

Acknowledgments

We thank T. Kardos, director of Rail Navigator Kft. to support this work, and MAV-START Zrt. for their interest and continuous feedback.

References

[1] ERNST, A. T., JIANG, H., KRISHNAMOORTHY, M., OWENS, B., SIER, D.: An Annotated Bibliography of Personnel Scheduling and Rostering, *Annals of Operations Research,*. vol. 127, 2004, pp. 21-144.

[2] ERNST, A. T., JIANG, H., KRISHNAMOORTHY, M., NOTT, H., SIER, D.: An Integrated Optimization Model for Train Crew Management, *Annals of Operations Research* vol. 108, 2001, pp. 211-224.

[3] HARTOG, A., HUISMAN, D., ABBINK, E. J. B., KROON, L. G.: Decision Support for Crew Rostering at NS, *Public Transport*, vol. 1, 2009, pp. 121-133.

[4] CAPRARA, A., FISCHETTI, M., TOTH, P., VIGO, D., GUIDA, P. L.: Algorithms for Railway Crew Management, *Mathematical Programming,* vol. 79, 1997, pp. 125-141.

[5] GOUMOPOULOS, C., HOUSOS, E.: Efficient Trip Generation with a Rule Modeling System for Crew Scheduling Problems, *J. of Systems and Software*, vol. 69, 2004, pp. 43-56.

[6] http://en.wikipedia.org/wiki/, Hungarian_State_Railways\mediaviewer\File: Magyarorszag_Vasuti_Terkepe.svg, November 26, 2014.