Solving Large Scale Crew Scheduling Problems by using Iterative Partitioning

Erwin Abbink[1], Joël van 't Wout[1] and Dennis Huisman[1,2]

[1] Department of Logistics, Netherlands Railways (NS), P.O. Box 2025, NL-3500 HA Utrecht, The Netherlands
[2] Erasmus Center for Optimization in Public Transport (ECOPT) & Econometric Institute, Erasmus University Rotterdam, P.O. Box 1738 NL-3000 DR Rotterdam, The Netherlands
Erwin.Abbink@ns.nl, Joel.vantWout@ns.nl, huisman@few.eur.nl

Abstract. This paper deals with large-scale crew scheduling problems arising at the Dutch railway operator, Netherlands Railways (NS). We discuss several methods to partition large instances into several smaller ones. These smaller instances are then solved with the commercially available crew scheduling algorithm TURNI. In this paper, we compare several partitioning methods with each other. Moreover, we report some results where we applied different partitioning methods after each other. With this approach, we were able to cut crew costs with 2% (about 6 million euro per year).

1 Introduction

In [13] it was shown that very large Crew Scheduling Problems can be solved using state of the art Operations Research (OR) techniques. At NS we use similar techniques to solve our Crew Scheduling Problem (CSP). We present several methods to handle even larger cases than presented in the referred paper.

NS is the main Dutch railway operator of passenger trains, employing in total 3,000+ drivers and 3,500+ conductors in 29 crew depots. A typical crew scheduling instance of NS related to a single duty type (driver or conductor) on each workday requires assigning about 14,000 timetabled trips to 1,000+ duties in 29 crew depots. Additionally, we would like to solve the problem for a complete week, which even gives a new dimension to the problem. This produces set-covering instances that are much larger than those addressed in the literature so far, and they have many additional nasty crew-depot constraints. Furthermore, these figures also imply that each duty covers about 14 trips on average, which is a higher number than airlines usually encounter. As described in [1], due to the complex set of labor rules, automated support in the crew scheduling process is absolutely necessary. Therefore, NS has been using the automated crew scheduling system TURNI since 2000. TURNI was developed by Double-Click, which has customized it several times to cope with the complex rules that govern NS crew schedules. For NS, the software is considered to be a black box where data are inserted and duties are returned. During the years of using

ATMOS 2007 (p.96-106)
7th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems
http://drops.dagstuhl.de/opus/volltexte/2007/1168

the software, we got the impression that although the system was capable of handling large instances, the results could be improved using the characteristics of our problem. E.g. experiments showed that re-optimizing a part of a solution resulted in better solutions. Next to that the developed working method handled the global (weekly) constraints in a rigid way. To explain this we present Figure 1. In this figure, a few possible duties are plotted. They are assigned to a certain base (A or B) and have a certain length. The vertical line indicates the moment in the night where no trains are operated.

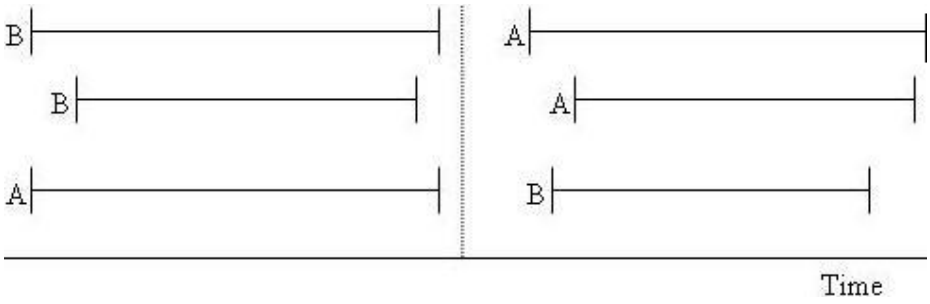

Fig. 1. Duties example

The problem is too large to solve it as a single instance. In the initial working method we created a CSP for each weekday and solved it. We could easily assign the trips to a single weekday because there are almost no trains running through the night, so there is a natural moment in time to split the problem. For each sub-problem we made a guess of the contribution to the global problem. This was fixed and there was no interaction between the sub-problems. E.g. the average duration of the duties for a crew depot is 8:00 hours. The sub-problem for the weekday was limited to an average of 7:40 hours and the problem for the weekend was limited to an average of 8:30. In this way the global average would be approximately 8:00 hours. We developed a method were we constructed and solved sub-problems iteratively. We iterated between large and small sub-problems and between day based and week based problems. The results of the solutions were passed to the consecutive problems. This method resulted in an improvement of about 2% on the solution costs. This paper will describe this method and the results in detail. The remainder of this paper is organized as follows. The concept of crew scheduling at NS is explained in more detail in Section 2. We will describe the characteristics of the problem which are used in the iterative approach. In Section 3, we briefly discuss some theory which is the basis for our method. Afterwards, in Section 4, we will present our method and we will analyze some examples of sub-problems that are constructed. The computational results of our method are presented in Section 5. Finally, we finish this paper with some concluding remarks.

2 Crew planning at NS

In Figure 2, we give a schematic overview of the crew planning process for drivers and conductors at NS. Other crew members (at ticketing offices, the call center, mechanics, etc.) fall outside the scope of this paper. The crew scheduling problem (CSP) is the problem of assigning tasks to anonymous duties. These tasks are given by the timetable and by the rolling stock schedule (see [11] for a discussion on all planning problems at NS). More formally, a *task* is the smallest amount of work that has to be assigned to one driver. A *duty* is the work for one crew member from a specific crew base on a certain day.

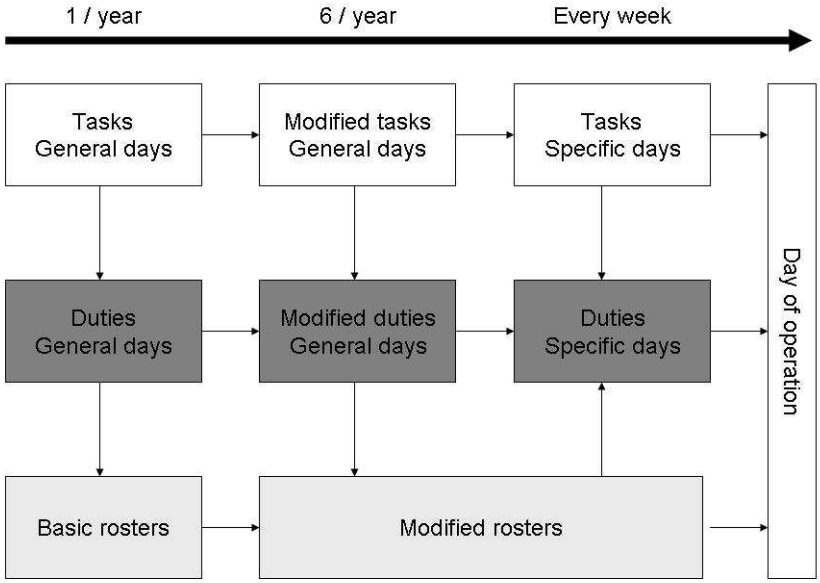

Fig. 2. Crew planning process

At NS, the crew scheduling process has been split in two stages. First, the crew schedules for the annual plan are constructed. Secondly, the crew rosters are created, where the crew members are assigned to operate the duties. This paper will focus on the first phase, the generation of the duties for the annual plan. This plan deals with a generic Monday, Tuesday and so on. This generic annual plan is modified about 6 times a year as a result of changes in timetable and rolling stock schedules. The other parts of the process fall outside the scope of this paper (for crew rostering, we refer to [8] and for crew re-scheduling to [10]). In the CSP that is solved for generating the generic annual plan, some rostering

aspects are also taken into account. For instance, the average duty length over all duties on a certain crew base should not exceed 8 hours. The reason is that, if this time is exceeded, then it is impossible to construct rosters where the average working time per week is less than 36 hours (in principle each full-time crew member works 9 days in two weeks). The number of night duties (duties with a working period between 1:00h and 5:00h) in a roster is also limited. This constraint should also be validated at a weekly basis. Moreover, it is important that to obtain a fair division of the work over the week for the different crew members, the work should be fairly spread over the different bases. The latter constraints are typical for the Dutch situation and are known as "Sharing Sweet & Sour" rules. They aim at allocating the popular and the unpopular work as fairly as possible among the different crew bases. For instance, some routes are more popular than others and intercity trains are preferred over regional trains. For a detailed description of these rules, we refer to [1]. One example is the percentage of work on intercity trains. Of the work assigned to a depot for a week, at least 25% should be on the intercity trains. Again, we could require every weekday to contain at least 25% of this work but it is better to check this constraint for a complete week.

3 Models and Algorithms for CSP

In this section, we give a short overview on models and algorithms that are used to solve the CSP. Moreover, we provide a mathematical formulation for a CSP containing 2 days without tasks overnight.

The airline industry has used OR models and techniques to solve crew scheduling problems for many years, see e.g. [2], [6] and [9]. However, in the railway industry the sizes of the crew scheduling instances are, in general, a magnitude larger than in the airline industry. The latter has made the application of these models in the railway industry prohibitive until recently. Developments in hardware and software enable the railway industry to use these models nowadays as well, see [4, 13, 14, 7], among others.

The CSP can be modeled as a set covering problem with additional constraints. If we consider the problem for a whole week where this is only a minor interaction between the different days, we get a special structure of the mathematical program. To show this, we give a mathematical formulation for the problem with two days. Let T^1 and T^2 be the set of tasks for day 1 and 2, respectively. Furthermore, D^1 and D^2 denote the set of duties for these days. The subset D_i^1 (D_i^2) of D^1 (D^2) consists of the set of duties containing task i. The binary decision variables x_j (and y_j) indicate whether duty $j \in D^1(D^2)$ is included in the solution or not. Every duty j has positive costs c_j. Furthermore, let S be the set of additional constraints and let l_s and u_s be the lower and upper bound for constraint $s \in S$. Finally, let v_j^s (and w_j^s) be the weight of duty $j \in D^1(D^2)$for constraint s. Then we can formulate this CSP as follows:

$$\min \sum_{j \in D^1} c_j x_j + \sum_{j \in D^2} c_j y_j \tag{1}$$

$$\sum_{j \in D_i^1} x_j \geq 1 \quad \forall i \in T^1, \tag{2}$$

$$\sum_{j \in D_i^2} y_j \geq 1 \quad \forall i \in T^2, \tag{3}$$

$$l_s \leq \sum_{j \in D^1} v_j^s x_j + \sum_{j \in D^2} w_j^s y_j \leq u_s \quad \forall s \in S, \tag{4}$$

$$x_j \in \{0,1\} \quad \forall j \in D^1, \tag{5}$$

$$y_j \in \{0,1\} \quad \forall j \in D^2. \tag{6}$$

Equation (1) is the objective function, which states that the sum of the duty cost is minimized. Constraints (2) and (3) guarantee that for each task i, at least one duty that contains this task is selected. Note that only duties of day 1 (2) can contain tasks of day 1 (2). It may sometimes be better to perform a task more than once. If, for example, the number of tasks going out of a crew base differs from the number of tasks going into the crew base on a day, overcovering is necessary. Moreover, even if overcovering is unnecessary, it may be cheaper to allow overcovering. By allowing overcovered tasks it can be that other tasks can be covered easier, resulting in a larger decrease in costs than the extra money for the overcovered task. Constraints (4) are additional constraints. Consider as an example of an additional constraint, a crew depot for which the total number of duties on both days is limited to 50. Then $l_s = 0$, $u_s = 50$ and $v_j^s(w_j^s) = 1$ for all duties belonging to this depot and $v_j^s(w_j^s) = 0$ for all other duties. For some additional constraints it is allowed to violate the constraint at the cost of a penalty. These constraints are moved to the objective function, along with the penalty. The last two sets of constraints (5,6) indicates that the decision variables are binary.

Often CSPs are solved day by day. Even then the resulting set covering problems are extremely large. Therefore, column generation techniques are often applied to tackle the large number of duties. We assume that the reader is familiar with the basic ideas of column generation (recent surveys on this topic are [2, 15, 5]).

TURNI uses column generation combined with Lagragian relaxation. In the remainder of this section, we give a short description on how TURNI works. TURNI is based on a heuristic presented in [3], which is designed for solving very large scale set covering instances. This Lagrangian-based heuristic, called CFT-heuristic, in which CFT stands for Caprara, Fischetti and Toth, forms the bases of TURNI. The main characteristics of the algorithm are a dynamic pricing scheme for the variables, coupled with subgradient optimization and greedy algorithms, and the systematic use of column fixing to obtain improved solutions. We will not discuss these characteristics in detail but we would like to

stress that, as a result of the algorithm, not only the final set of created duties is presented, but also a large number of "good" duties are available. We will use this additional information for constructing our sub-cases as described in Section 4.4. The process of creating duties from the tasks is called duty generation. The duties to be generated have to satisfy all constraints concerning a single duty, like rules about maximum length and rules for the breaks. When the graph is created, duties can be generated by finding a feasible path through the graph which starts and finishes in the same depot. A path is a feasible path when it satisfies all rules concerning the duty length and meal breaks. The costs of the arcs are defined in such a way that the total cost of a path is equal to the reduced cost of a duty. By finding the shortest, feasible path and checking whether its cost is negative or not, it is possible to check if there are still duties with negative reduced costs.

4 The partitioning method

Our method is based on two observations. First of all, the global constraints are to be validated on a weekly basis. The original method used a static partitioning of the complete problem into separate days of the week (Friday, Saturday and Sunday). Before a solution was computed an estimate was made on the effect the sub-problem would have on the complete problem. For example, the average duration of the duties must be below 8 hours per week per crew depot. For the sub-problem for the Friday this was set to a maximum average of 7:40 hours and for Saturday and Sunday this was set to a maximum of 8:30. Overall this will result in an average that is below 8:00 hours. These constraints were based on rules of thumb that were used by the planners for years during the manual planning. We observed that planning for a complete week and taking into account the real week constraint could lead to a better overall solution. The second observation was that in some cases the solution was improved if, for instance, the solution for one crew depot was re-scheduled. For this the duties and tasks for that depot were given to the TURNI software and the solution for this smaller problem was better than the original solution. This also indicated that solving the larger instance was becoming difficult for the current implementation. Combining the two observations, we reasoned that we could possibly improve the overall solution if we would take the solution for one or more depots for the separate days and combine them into a case for the complete week. Next to that, we reasoned that it would be good to have several iterative combinations of depots in order to minimize the effect of optimizing over a sub-problem. Next to that we are interested in the effect variation in size of the cases. The most important dimensions in scheduling are time and location of the activities. It seems natural to use this dimensions to partition the overall problem. We will now describe the different partitioning methods one by one.

4.1 Weekday partitioning

In this method we create a sub-problem per weekday. All trips belonging to the same weekday are combined in a sub-problem. Not all weekdays are included. Monday, Tuesday until Friday are very similar. Therefore we choose one (the Friday) of the weekdays as a pattern weekday. At the end, the solution for this weekday will be used as a solution for the other weekdays too. The differences between the weekdays will be handled manually. For the Saturday and the Sunday a separate solution is created. The advantage of this method is that it can be used without an initial solution. Because tasks of different weekdays cannot be scheduled together in a single duty at NS, this method is a good option to create an initial solution. In fact, this method was used as the only partitioning method during the first years of using the system

4.2 Geographical partitioning

The primary geographical partitioning is the depot to which a duty is created and assigned. After an initial solution is created we can combine all duties assigned to a depot for all weekdays. This results in 29 sub-problems. These sub-problems are very small and do not provide much room for improvement. Therefore we create some larger cases by clustering some depots based on the geographical location. We create small partitions where on average 3 depots are clustered and we create large partitions where on average 7 depots are clustered.

4.3 Line based partitioning

The railway product is defined by railway lines. Trains are operated along several railway lines at a certain frequency. The idea is to combine the depots into groups when there are many trains that connect these stations. We call this line base partitioning. A snapshot of the train services of the NS is given in Figure 3.

One can see that there are several series running between e.g. Amsterdam Centraal and Utrecht Centraal, which makes them good candidates to group into one depot cluster.

4.4 Partitioning based on column information

The last partitioning method we present is based on the information that is generated by the scheduling algorithm. As indicated in Section 3, TURNI uses a mechanism to rank duties according to their likelihood to be selected in an optimal solution. In this way good duties are created which have a high probability to be part of the optimal solution. Duties that have no contribution to a good solution are removed from the set, while new duties that have a positive contribution are added. Therefore the total set of duties is continually improving. TURNI not only returns the duties which are in the final solution, but it also returns these good duties which were generated during the solution process. These

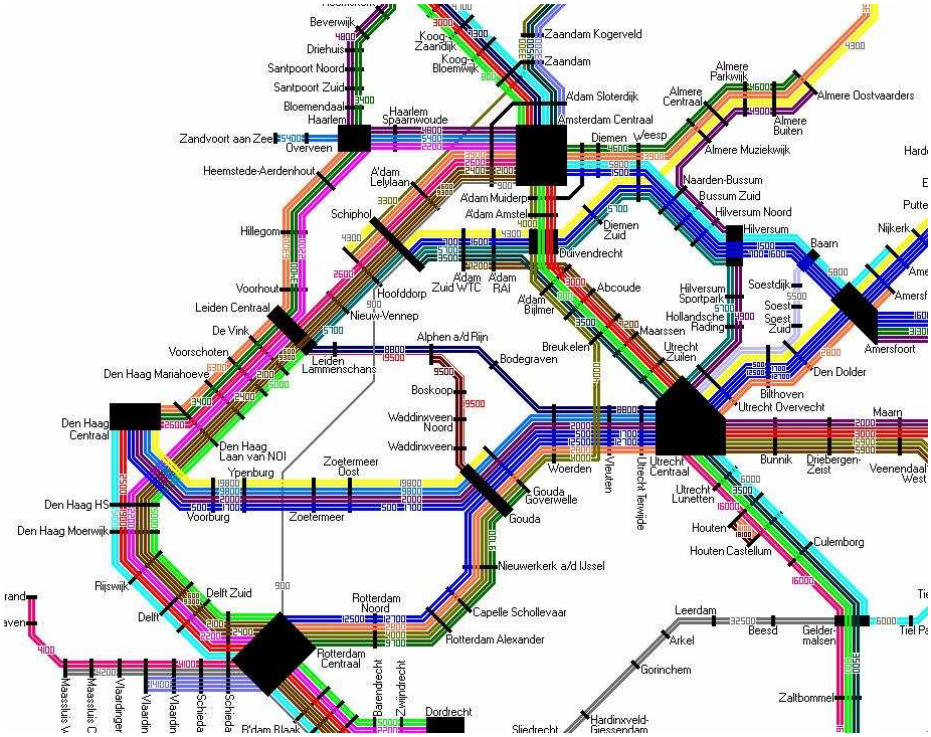

Fig. 3. Part of the Dutch railway network

duties can be used to give the information we are looking for. If two tasks appear together in many duties, it is likely that these two tasks will be assigned to the same duty in the optimal solution. If, on the other hand, two tasks (almost) never appear together in a duty, these tasks will probably be assigned to different duties in the optimal solution. Now, it is possible to give each pair of duties in the current solution a score which can be used as a measure for inserting a pair into a partition. This score is based on how often tasks from these two duties appear together in the set of all duties. We calculate the score for each pair as follows. First, we count for each combination of tasks in these duties, say t_1 and t_2, the number of duties in the whole set that covers task t_1 and t_2. Then, we add all these numbers. In this way, we can construct a graph $G = (V, E)$, where the duties are represented by the vertices, and the edges represent the fact that the score is positive. We define a weight $q(u, v)$ for each edge $(u, v) \in E$. This weight corresponds to the score calculated above. We want to find a partition of the vertices of G into k equal subsets $V_1, ..., V_k$, such that the total weight of the edges between different subsets is minimized, or more formally

$$\min \sum_{(u,v)\in E, u\in V_i, v\in V_j, i\neq j} q(u, v). \tag{7}$$

We use a generic algorithm for graph partitioning based on [12] to solve this problem. For the details, we refer to [16].

5 Results

5.1 Experimental Design

All experiments were carried out on the same hardware (Pentium IV, 3 GHz, 1Mb RAM). First we evaluated the different partitioning methods by running them after a base run in which we used the weekday partitioning. After that we made a final run in which all methods were applied sequentially. We used a maximum computation time of 6x24 hours in total. This means if a problem is partitioned into two sub-problems, both sub-problems have a maximum computation time of 3 days. In the final run, where all methods are used sequentially, parallel machines were used and the maximum computation time per sub-problem was set to 1 day.

5.2 Computational results

In Table 1 we present the results of the experiments. The numbers in the method columns indicate the orders of applying the method. An empty cell indicates that the method was not used in the experiment. In the last two columns, we report the number of duties and the relative improvement compared to the base case. We choose to report the number of duties instead of the objective function, because the value of the objective function is mainly determined by the number of duties.

Table 1. Results

	Weekday	Geo. Large	Geo. Small	Line	Column Info.	#Duties	ΔDuties
1	1					7432	-
2	1	2				7339	-1,3%
3	1		2			7318	-1,5%
4	1			2		7335	-1,3%
5	1				2	7331	-1,4%
6	1	2	3	4	5	7287	-2,0%

The results show that all partitioning methods (except the base one with only weekdays) perform more or less the same. For all of them, the number of duties is reduced by approximately 1.5%. An even larger improvement could be obtained by applying all methods sequentially after each other. In this case, several machines were used and the different sub-problems were run in parallel. In this way, an improvement of 2.0% could be obtained resulting in a saving of about 6 million euros. This final solution was implemented in practice for the crew schedules corresponding to the timetable of the year 2007.

6 Conclusions

In this paper we described a method that improved the usage of an advanced crew scheduling algorithm using iterative partitioning of the problem. The method is being used for creating the schedules of a large number of drivers. We have shown that applying some basic partitioning techniques can have a significant added value when combined with some advanced mathematical methods. Overall the efficiency was improved with about 2%. The method is automated which not only enables us to create an efficient production plan, but also gives us the possibility to use it for what-if scenario analyses. In the past the scenarios were only studied for a single weekday. With this method, the analyses are more reliable because the complete week is taken into account.

References

1. Abbink, E., Fischetti, M., Kroon, L., Timmer, G., Vromans, M.: Reinventing crew scheduling at Netherlands Railways. Interfaces **35** (2005) 393–401
2. Barnhart, C., Johnson, E., Nemhauser, G., Savelsbergh, M., Vance, P.: Branch-and-price: Column generation for solving huge integer programs. Operations Research **46** (1998) 316–329
3. Caprara, A., Fischetti, M., Toth, P.: A heuristic algorithm for the set covering problem. Operations Research **47** (1999) 730–743
4. Caprara, A., Fischetti, M., Guida, P., Toth, P., Vigo, D.: Solution of large-scale railway crew planning problems: the italian experience. In Wilson, N., ed.: Computer-Aided Transit Scheduling, Springer Verlag, Berlin (1999) 1–18
5. Desaulniers, G., Desrosiers, J., Solomon, M., eds.: Column Generation. Springer, New York (2005)

6. Desrosiers, J., Dumas, Y., Solomon, M., Soumis, F.: Time constrained routing and scheduling. In Ball, M., Magnanti, T., Monma, C., Nemhauser, G., eds.: Network Routing. Volume 8 of Handbooks in Operations Research and Management Science. North-Holland (1995) 35–139
7. Fores, S., Proll, L., Wren, A.: Experiences with a flexible driver scheduler. In Voß, S., Daduna, J., eds.: Computer-Aided Scheduling of Public Transport, Springer, Berlin (2001) 137–152
8. Hartog, A., Huisman, D., Abbink, E., Kroon, L.: Decision support for crew rostering at NS. Technical Report EI2006-04, Econometric Institute (2006)
9. Hoffman, K., Padberg, M.: Solving airline crew scheduling problems by branch-and-cut. Management Science **39** (1993) 657–682
10. Huisman, D.: A column generation approach to solve the crew re-scheduling problem. European Journal of Operational Research **180** (2007) 163–173
11. Huisman, D., Kroon, L., Lentink, R., Vromans, M.: Operations Research in passenger railway transportation. Statistica Neerlandica **59** (2005) 467–497
12. Kernighan, B., Lin, S.: An efficient heuristic procedure for partitioning graphs. Bell Systems Technical Journal **29** (1970) 291–307
13. Kohl, N.: Solving the world's largest crew scheduling problem. ORbit (2003) 8–12
14. Kroon, L., Fischetti, M.: Crew scheduling for netherlands railways "destination: Customer". In Voß, S., Daduna, J., eds.: Computer-Aided Scheduling of Public Transport, Springer, Berlin (2001) 181–201
15. Lübbecke, M., Desrosiers, J.: Selected topics in column generation. Operations Research **53** (2005) 1007–1023
16. Van 't Wout, J.: Crew scheduling at Netherlands Railways: using TURNI effectively. Master's thesis, Faculty of Economics, Erasmus University Rotterdam (2007)