Konrad-Zuse-Zentrum
für Informationstechnik Berlin

Takustraße 7
D-14195 Berlin-Dahlem
Germany

RALF BORNDÖRFER ANDREAS LANGENHAN
ANDREAS LÖBEL CHRISTOF SCHULZ
STEFFEN WEIDER

Duty Scheduling Templates

ZIB-Report 12-09 (February 2012)

Herausgegeben vom
Konrad-Zuse-Zentrum für Informationstechnik Berlin
Takustraße 7
D-14195 Berlin-Dahlem

Telefon: 030-84185-0
Telefax: 030-84185-125

e-mail: bibliothek@zib.de
URL: http://www.zib.de

ZIB-Report (Print) ISSN 1438-0064
ZIB-Report (Internet) ISSN 2192-7782

Duty Scheduling Templates

Ralf Borndörfer* Andreas Langenhan** Andreas Löbel*
Christof Schulz* Steffen Weider*

February 17, 2012

Abstract

We propose duty templates as a novel concept to produce similar duty schedules for similar days of operation in public transit. Duty templates can conveniently handle various types of similarity requirements, and they can be implemented with ease using standard algorithmic techniques. They have produced good results in practice.

1 Introduction

Duty scheduling in public transit deals with the construction of the daily shifts of work for bus, tram, or subway drivers. It is well known that this task can be modeled as a set partitioning problem, which can be solved efficiently using column generation methods, see, e.g., Desaulniers et al. [1998] for an overview. This approach typically produces high quality solutions, but it is also highly sensitive, i.e., already small changes in the input data can lead to completely different solutions. As public transit companies typically operate slightly different timetables on the individual days of the week, on vacations and holidays, at special events etc., the associated optimal duty schedules may vary widely. This is often considered as undesirable for various reasons including operational stability and mnemonic ease; it also complicates the subsequent construction of a periodic duty roster. What is wanted is rather some kind of "uniformity" or "regularity" of the duty schedules, i.e., similar duties covering similar parts of the timetable.

Giving a precise meaning to term *regularity* is not easy. Simple approaches, however, such as rewarding the reuse of duties or changeovers in day-to-day or regular-to-irregular methods often do not produce the desired results because

*Zuse Institute Berlin, Takustr. 7, 14195 Berlin, Email mailto:[surname]@zib.de, URL http://www.zib.de, and Dres. Löbel, Borndörfer & Weider GbR, Churer Zeile 15, 12205 Berlin, URL http://www.lbw-berlin.de

**IVU Traffic Technologies AG, Bundesallee 88, 12161 Berlin, Email mailto:al@ivu.de, URL http://www.ivu.de

of a lack of degrees of freedom, i.e., the schedules do not "look similar", or the costs are much too high.

We propose in this article a concept that we call *duty templates* as a means to achieve a well-balanced compromise between regularity and efficiency. A duty template specifies the beginning and the end of a duty, leaving the intermediate part open. In this way, essential characteristics of a duty, in particular, those that are relevant for duty rostering, can be recovered, while on the other hand plenty of degrees of freedoms are available for an efficient duty construction. Duty templates can be implemented in terms of ordinary duty types, and handled by standard algorithmic techniques – a particular advantage of this approach.

This paper introduces the concept of duty templates, discusses their algorithmics, and their use to produce both regular duty schedules per se as well as their use to initialize the construction of periodic duty rosters. Computational results for a number of case studies are reported.

2 Regularity of Duty Schedules

A *day of operation* in the planning process of a public transit company is characterized by a timetable of vehicle trips and by a set of planning parameters, such as the available resources, scheduling rules, etc.. Typically, several weekdays featuring the same characteristics are amalgamated into a single day of operation; "Monday–Friday" is typical day of operation. These basic days of operation are further differentiated, because not only do working days and weekends have their own timetables; variations of the timetable are also due to holidays, special events (like trade fairs), road works, and extra bus tours, e.g., to bring school classes to their swimming lessons, giving rise to days of operation such as "Monday–Friday (holidays)". However, the differences between the timetables for working days and those for Saturdays and Sundays are much larger than the differences between the timetables for individual working days. On working days, home-to-work or, in rural areas, home-to-school commuter traffic dominates, while on weekends shopping and recreational traffic play are more important role. In addition, there are typically less trips scheduled on weekends and therefore also less drivers needed than on working days. In fact, the duty schedules for working days and weekends may be very different on purpose, because some drivers are not available on weekends and/or some duty types are only valid on working days.

In general, public transit companies strive for *similar duty schedules for similar days of operation*. We call this in the following *regular duty scheduling*. Regular duty schedules make operations easier, and, what is especially important in Germany, it reduces the workload of the workers council, which must approve all duty schedules. It also simplifies the subsequent step of duty roster

planning.

Regular duty scheduling is, of course, as old as public transit itself. For a long time, practitioners have tackled the problem using the copy function of their scheduling systems. One first produces a duty schedule for a reference day of operation. This *reference schedule* is copied to another, similar day of operation. Most scheduling systems will recover the duties (fully or in part) as far as possible. This produces a partial schedule plus a remainder of unscheduled tasks. These tasks are then scheduled by putting them into the copied duties (for new tasks), if possible, or by creating new duties, possibly also modifying some of the copied duties.

The scientific literature has taken up the subject only recently. The article Steinzen et al. [2007] gives a short overview on regular duty scheduling in the airline and railway context and discusses, as far as we know for the first time, the public transit case. The authors propose a classification into regularity and rescheduling approaches. In *rescheduling* approaches a reference schedule is calculated first, and then similar schedules are computed for daily variations of the underlying problem. In *regularity* approaches several similar duty scheduling problems are considered, and simultaneously solved together. Choosing a suitable objective, a duty schedule is computed for each problem, and these duty schedules are similar to each other. The article also describes a way to implement this procedure implicitly using special branching rules in a MIP-framework for integrated vehicle and duty scheduling in public transport. Building on this work, Amberg et al. [2011] formulates MIP-models for two such approaches, namely, so-called day-to-day similarity and simultaneous similarity. The latter is based on a concept of regular patterns, i.e., (not necessarily contiguous) chains of regular tasks, which are supposed to appear simultaneous in several duty schedules. Computational results for three vehicle scheduling instances with up to about 800 trips a day are given.

In this terminology, our template approach can be classified as a rescheduling method.

3 Duty Templates

We introduce in this section our basic concept, the *duty template*, and discuss its use in regular duty scheduling with respect to different types of similarity.

3.1 Duty Scheduling Problem

The duty scheduling problem can be modeled as a *set partitioning problem* with additional *(duty) mix constraints* as follows:

$$
\begin{array}{lll}
(\mathsf{DSP}) \quad \min & \displaystyle\sum_{d\in\mathcal{D}} c_d x_d, & \\
\text{s.t.} & \displaystyle\sum_{d\in\mathcal{D}(t)} x_d = 1, & \forall t \in T, \\
& \displaystyle\sum_{d\in\mathcal{D}} M_{bd} x_d \leq r_b, & \forall b \in \mathcal{M}, \\
& x_d \in \{0,1\}, & \forall d \in \mathcal{D}.
\end{array}
$$

In this model T denotes the set of all tasks which have to be scheduled; a *task* is a minimal part of a trip which arises from cutting the trip at its relief points. $\mathcal{D}$ denotes the set of all feasible *duties*, $\mathcal{D}(t)$ the set of all duties that cover a task $t \in T$, and $T(d)$ the set of tasks covered by duty $d \in \mathcal{D}$. The objective function sums up the costs of the duties. The cost of a duty itself is a linear combination of fixed costs, costs for paid time, and various penalties for over- or under-running certain key indicators of duties. Finally, $\mathcal{M}$ is a set of knapsack-type duty mix constraints; $M \in \mathbb{R}^{\mathcal{M}\times\mathcal{D}}$ is the coefficient matrix associated with the base constraints. The mix constraints can be used to control the number of duties of certain duty types or the consumption of resources like paid time of sets of duties in the solution.

In our approach we associate with every duty a *duty type* and a *home depot*. A duty type can equivalently be described either explicitly as a set of duties or implicitly as a set of rules which has to be fulfilled by every duty of the respective type. Typically duty types can be differentiated by their starting- and ending-times, such as "early duties" or "night duties", by their durations, such as "short duties", or by the number of duty parts, such as "split duties" or "straight duties".

We solve (DSP) by *column generation*, because the number of duties in $\mathcal{D}$, and thus the number of variables of (DSP) is in general too large to solve it directly. The pricing problem is solved separately for each combination of duty type and home area. It is modeled as a shortest path problem with additional resource constraints on a graph that depends on the duty type and the home depot. More details on this algorithm can be found in the publications Borndörfer et al. [2003]; Weider [2007].

3.2 Template Concept

A *duty template* is a tuple consisting of a set of duty types, a set of home depots, and a set of additional *template restrictions* for the feasibility of duties, such as a time window for the start of a duty or a set of tasks which have to be

covered by a duty. The rules for the construction of templates are such that a template can again be modeled as an (artificial) duty type, namely, a so-called *template duty type*. We say that a duty is a *specialization* of a template if its duty type and home depot match the template, and if it satisfies the template restrictions, i.e., if it is feasible for the template duty type.

Given a set of templates, the idea is to construct a duty schedule that consists to a certain percentage of duties that are specializations of templates. To this purpose, we simply add the template duty types to the original duty types, and control their use by adding suitable mix constraints, i.e., a regular duty scheduling problem is just a somewhat larger ordinary duty scheduling problem. For example, the number of template specializations can be controlled by adding a single duty mix constraint. Alternatively, we are can also give a bonus to specializations, such that specializations will be preferred in a solution.

The usefulness of this concept depends, of course, on the construction of the templates and on the construction of the mix constraints associated with them. The following two simple approaches have produced good results in practice.

Task Similarity. The idea is to produce duties that are similar in the sense that they contain a maximum amount of identical tasks, possibly adding new tasks. We also consider the case of exact task similarity, where no new tasks must be added (i.e., duties are reproduced as far as possible). This approach can be implemented in terms of *task similarity templates* as follows. At first a duty schedule S for some *reference scenario* is computed. This reference scenario may consist of a typical day of operation or, alternatively, of all regular trips, i.e., all trips that are common to all considered days of operation. Then duty templates for a *similarity scenario*, i.e., a similar day of operation, are generated automatically using the duties of the duty schedule S. To this purpose, we consider for each individual duty d in S the set $T(d)$ of tasks it covers in the reference scenario; let $T(d)' \subseteq T(d)$ be the set of tasks that also appear in the similar scenario. If $T(d)'$ is empty, it makes no sense to copy duty d and no template is generated. Otherwise, we generate a task similarity template that stipulates to cover all tasks in $T(d)'$, the duty type of d, and the home depot of duty d. This resulting similarity template is, in fact, a restriction of the duty type associated with d. As the duties in such a template must cover at least one task of $T(d)'$, the associated task covering constraints guarantee that at most one duty can be constructed from this template, such that there is no need for further mix constraints associated with *individual* similarity templates; in the extreme case of exact task similarity, a single duty can be generated, namely, the one that covers exactly the tasks in $T(d)'$. We finally add one overall *similarity mix constraint* to control the minimum number of specializations of similarity templates in a solution of the similarity scenario.

Time Window Similarity. Another application for templates is to generate duties that fit into an existing duty roster. The feasibility of a roster depends to a large extent on the rest time between duties and on maximum working and driving times per week or month. In this case, it is not of much interest, which tasks a duty covers; what really matters is only that the duty starts and ends within the right time interval or time window. We therefore use *time window templates* to generate duties which can be put into the slots of an existing roster. These templates specify a starting time window (in our experiments of 4 hours) and a maximal shift duration (here 9 hours and 50 minutes). This is also the maximum shift duration of the continuous duty types in our scenarios; it is, however, also possible to build split duties as long as they do not exceed this shift duration. In our experiments we classify the duties of the reference solution whose shift times do not exceed 9 hours 50 minutes by their starting time window, and count the number of such duties in each time window. Then we generate a template for each time window, and add a mix constraint to reproduce in each time window at least the number of duties that were present in the reference solution.

4 Computational Results

Templates have been implemented in autumn 2009 in version 3.99 of our `sched-opt` optimization suite for public transit, and are available in the commercial planning suite `ivu.plan` since release 10. The following computations have been done with `sched-opt` version 4.30, which is integrated in all branches and service packs of the latest `ivu.plan` release 11.

All following computations were done on an Intel(R) Xeon(R) CPU E31280 with 3.50GHz. We used only one core. Our code was compiled as 32bit code, that is, the memory consumption is always below 4GB.

4.1 Urban Duty Scheduling

We have tested two real scenarios from a large German urban public transit company. Table 1 lists some statistics: the number of tasks (# tasks), the number of duties of the reference scenario that can be used unmodified in a (partial) start solution for the new scenario (#duties in start solution), and the number of tasks in the duties of the previous row (#tasks in start solution). The scenarios feature two types of straight duties and one type of split duties. The duty types allow for various break rules, such as block breaks with one, two, or three breaks, and the 1/6-quotient rule. The straight duty types have a maximum shift time of 9 hours, 50 minutes, the split duty type of 14 hours. The maximum duty time for the split duty is also 9 hours, 50 minutes.

Table 1: Urban duty scheduling scenarios.

	urban medium	urban large
#tasks	504	2320
#duties in start solution	44	194
#tasks in start solution	471	1749

Table 2: urban medium: Regular duty scheduling w.r.t. exact task similarity.

#reference duties	–	32	38	40
#duties	42	42	43	44
paid time [h:m]	317:43	316:34	317:15	318:51
objective value	77.37	77.98	79.14	80.15
running time [sec]	1199	787	904	452

4.1.1 Task Similarity

Table 2 shows results of an optimization using exact task similarity for the small scenario. That is, we try to reproduce a given number of duties of a reference duty schedule (not listed in the table). The number of duties to be reproduced, i.e., the rhs of the similarity mix constraint, is given in row 1 (#reference duties). The first column of this table shows the results of an optimization run without similarity requirements. The second row shows the total number of duties in the schedule (# duties) and third row shows the total time paid to the drivers (paid time [h:m]), the fourth the objective value of the mathematical model (objective value), the last the running time (running time [sec]). As expected, the objective value increases as the number of duties to be reproduced grows. However, a duty schedule with 32 out of 42 identical duties is only marginally more expensive than the best duty schedule without similarity. At 40 identical duties, the optimizer only finds a solution with 2 additional duties and 1 hour 8 minutes more paid time. It was not possible to find a feasible solution with more than 40 identical duties.

As exact similarity already produces extremely similar schedules at almost no

Table 3: urban medium: Regular duty scheduling w.r.t. task similarity.

#reference duties	–	32	38	40
#identical duties	–	29	32	34
#similar duties		5	6	6
#duties	42	42	42	43
paid time [h:m]	317:43	316:34	317:05	318:31
objective value	77.37	77.97	78.28	79.06
running time [sec]	1199	698	675	495

Table 4: urban large: Regular duty scheduling w.r.t. exact task similarity.

#reference duties	–	170	180	187	188 (190)
#duties	227	229	230	233	233
paid time [h:sec]	1758:13	1767:52	1773:56	1775:34	1773:42
objective value	421.97	426.02	428.03	431.06	461.06
running time [sec]	6848	1299	2013	1161	1142

Table 5: urban large: Regular duty scheduling w.r.t. task similarity.

#reference duties	–	170	180	187	190
#identical duties	–	150	167	180	181
#similar duties	–	20	13	8	9
#duties	227	228	229	232	234
paid time [h:sec]	1758:13	1767:44	1770:51	1776:08	1777:20
objective value	421.97	425.9	426.22	430.51	432.62
running time [sec]	6848	2969	1373	1759	902

cost, adding more degrees of freedom can improve the schedules only slightly. Table 3 shows the results a task similarity optimization for the same scenario, where where we allow to add tasks to reference duties. That is, tasks that appear in the similarity scenario, but not in the reference scenario, may be inserted into the reference duties, and we still count them as similar. Table table 3 lists the results. The new row 2 shows the number of completely identical duties in the similarity solution (#identical duties), row 3 the number of reference duties into which additional tasks have been inserted (#similar duties). Indeed, the results are very close to the exact similarity case, albeit the solution quality is slightly better (as expected).

The running times of the similarity scenarios with an active similarity mix constraint are shorter than those for the unconstrained scenario. The reason for this behavior is that our algorithm is able to utilize the reference solution as a starting solution, and this shortens the column generation phase of our algorithm.

Tables 4 and 5 shows results for the same experiment for our bigger scenario, using the same duty types and rules. In this case, a higher similarity can be achieved (190 similar duties instead of 187), if additional tasks can be inserted into the reference duties. The last column in table 4 was produced stipulating 190 identical duties, the optimizer could, however, only find a solution with 188 identical duties (the high objective value is due to a penalty for the violated similarity mix constraint).

Table 6: Time window mix constraints.

Scenario	0–4	4–8	8–12	12–16	16–20	20–24
urban medium	–	19	6	12	5	–
urban large	2	92	13	70	20	–

Table 7: Regular duty scheduling w.r.t. time window similarity.

	urban medium	urban large
#duties	44	227
paid time [h:sec]	319:42	1757:53
objective value	79,24	422,05
running time [sec]	136	56334

4.1.2 Time Window Similarity

In our second experiment, we computed for the reference schedules of the scenarios in table 1 the numbers of duties having length at most 9 hours, 50 minutes, that begin in the four-hour time windows from 0–4, 4–8, and so on, see table 6, in order to reproduce a similar solution with the same characteristics.

Table 7 shows the results of the associated time window similarity optimization. The solution quality is nearly identical to the unconstrained case. The objective value raised from 77.37 (table 2, column 1) to 79.24 for the smaller scenario and for the larger from 421,97 (table 5, column 1) to 422.05. The running time for the large scenario went significantly up.

4.2 Regional Integrated Vehicle and Duty Scheduling

As duty templates fit within the standard duty scheduling modeling frame, they can also be used in an integrated vehicle and duty scheduling context, which is important in regional scenarios, see Weider [2007].

We consider here a scenario of a German regional public transit carrier. This scenario features 526 timetabled trips, 1620 tasks on these timetabled trips, 5 depots, 4 vehicle types, 4 duty types (short duties, normal duties, split duties, and duties for contractors), and 11.308 potential deadhead trips. The reference solution is the solution of a “Thursday” operation. It uses 45 duties, 35 vehicles, has an objective value of 131.01, and a paid time of 338h 47s. We try to construct a similar solution for a “Friday” of operation. The input differs by 19 trips which are only operated at Thursdays and by 15 trips that are only operated at Fridays. These differing trips are all short (up to 25 minutes driving time, with an average of 10 minutes). They stem from school- and swimming-trips. On “Friday” 23 duties from “Thursday” are still valid. 15

Table 8: Regular integrated vehicle and duty scheduling w.r.t. task similarity.

	thu	fri				tw
#reference duties	–	–	30	35	40	45
#identical duties	–	5	12	13	22	–
#similar duties	–	12	19	22	19	45
#duties	45	45	45	45	45	45
#vehicles	36	36	35	35	36	35
paid time [h:sec]	338:47	332:20	333:18	332:04	340:19	330:19
objective value	131.01	129.48	129.32	128.99	130.97	129.50
running time [sec]	16891	23654	15646	17359	24169	52472

additional duties from "Thursday" can be built on "Friday" by only adjusting deadhead trips.

4.2.1 Task Similarity

Table 8 lists the results of our task similarity computations. Column "thu" shows the characteristics of the solution for the "Thursday" of operation. The next 4 "fri" columns show results for "Friday". The last "tw" column is explained in the next section. Row "#reference duties" gives the number of "Thursday" duties that we want to reproduce on "Friday" ("–" means that no similarity is required). Row "#identical duties" is the number of identical duties in comparison with the reference solution, i.e. these duties have the same tasks and the same deadheads. Row "#similar duties" shows the number of similar duties; these duties contain all tasks associated with timetabled trips of the reference duties that are valid on Thursday as well as on Friday. Additional tasks that are only valid Friday are allowed, also adaptations of deadhead trips are allowed.

It can be seen that 30 or 35 reference duties can be reproduced at essentially no cost (the decreases of the objective function values are due to some heuristic decisions of our algorithm, see Weider [2007]. If we want to reproduce 40 similar duties (last row), the objective value increases significantly.

4.2.2 Time Window Similarity

In our last experiment we tried to construct a solution for "Friday" having similar duty start- and end-times and durations as the "Thursday"-duties. to this purpose, we manually group the 45 duties into the following sets: 4 duties starting between 4am and 8am with unlimited duration, 26 duties starting between 4am and 8am with a duration of at most 9h 50min, 1 duty starting between 8am and 12 am with a duration of at most 9h 50min, and 9 duties

starting between 12am and 4pm with a duration of at most 9h. Then we try to find a solution for "Friday" containing the same numbers of duties with these properties. The last column ("tw") of table 8 shows the results. The run took significantly longer, but it finds a solution fulfilling the requirements. The objective value is only slightly worse than the unconstrained one.

5 Conclusion

Duty templates are a versatile tool to construct similar duty schedules for similar days of operation. Various types of similarity, including task similarity (reconstruction of duties) and time windows similarity (duty distribution) can be handled in a convenient way, and in both stand-alone as well as integrated scheduling approaches, using standard algorithmic techniques. The method produces good results for real-world instances.

References

Amberg, Amberg & Kliewer (2011). Approaches for Increasing The Similarity of Resource Schedules in Public Transport. *Procedia - Social and Behavioral Sciences 20(0), 836 – 845.*

Borndörfer, Grötschel & Löbel (2003). Duty Scheduling in Public Transit. In W. Jäger & H.-J. Krebs (Eds.), *MATHEMATICS – Key Technology for the Future* pp. 653–674. Berlin: Springer Verlag. ZIB Report 01-02.

Crainic & Laporte (Eds.) (1998). *Fleet Management and Logistics.* Kluwer Academic Publishers.

Desaulniers, Desrosiers, Ioachim, Solomon, Soumis & Villeneuve (1998). A unified framework for deterministic time constrained vehicle routing and crew scheduling problems. In Crainic & Laporte [1998], chapter 3.

Steinzen, Suhl & Kliewer (2007). Branching Strategies to Improve Regularity of Crew Schedules in Ex-Urban Public Transit. In Liebchen, Ahuja & Mesa (Eds.), *ATMOS*, volume 7 of *OASICS*. Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany.

Weider (2007). *Integration of Vehicle and Duty Scheduling in Public Transport.* PhD thesis, TU Berlin.