Angewandte Mathematik und
Informatik
Universität zu Köln

Report No. 97-262

CATS–Computer Aided Tram Scheduling

by

Peter Heusch
Frank Meisgen
Ewald Speckenmeyer

1997

To appear in: Proceedings of SOR'97

Peter Heusch
Frank Meisgen
Ewald Speckenmeyer
Institut für Informatik
Universität zu Köln
Pohligstraße 1
D-50969 Köln

1991 Mathematics Subject Classification: 65C99, 68U20, 90C27
Keywords: traffic simulation, combinatorial optimization, scheduling

1 Introduction

If public transport systems circulate periodically (e.g. 4 times per hour), their timetable is completely determined by the timetable for a single period, the so-called initial timetable. The initial timetable can then be used to calculate the other schedules, mainly those for vehicles and those for crews. While construction of the latter schedules is supported by a number of commercially available products[LS96], the initial timetable is mostly calculated by hand. One of the reasons for this is the lack of suitable tools that fulfill the needs of a simple user interface as well as the support to check whether the constructed timetables obey some operational constraints. Some effort to construct such a tool for railway networks has recently led to LOP[ZBKW97].

In our project CATS we deal with the computer aided construction and optimization of initial timetables. We currently develop a tool that allows for a simple user interface, very similar to the normally used paper-based user interface, which eases the task of construction while at the same time the developed plan is checked against the set of constraints. The data is stored in a database to facilitate the communication between timetable construction and the following steps.

2 Optimization

Let a network $\mathcal{N}(S, C, t)$ be given with stations $S = \{s_1, \ldots, s_n\}$, connections $C = \{c_1, \ldots, c_v\}$ and a weight function $t : (S \cup C) \to [0 \ldots T-1]$ determining the running time between two adjacent stations and the stopping time at stations. All arithmetic is performed modulo some time period $T \in \mathbb{N}$. Every line $L = \{l_1, \ldots, l_m\}$ corresponds to a simple route in $\mathcal{N}$ with $l_j = (s_0^j c_1^j s_1^j c_2^j \ldots c_k^j s_k^j)$. Departure times at the first station are denoted by λ_j, the vector of the m different departure times is denoted by λ while Λ denotes the T^m element set of those vectors.

For the line l_j the arrival time at station s_i^j can be computed from λ_j and the sum of all waiting and driving times upto but not including s_i^j, its departure time is given by the sum of arrival time plus waiting time $t(s_i^j)$. Fixing the vector of the departure times establishes the complete schedule, which then can be evaluated. A measure that has often been used to check the quality of a given schedule is the sum of waiting times for all passengers who want to change the routes[Voß92]; to compute this some additional information

about the passengers must also be given, e.g. the origination/destination matrix. Different algorithmic paradigms have been employed to optimize schedules according to this criterion, for example Branch&Bound[Dom89] and heuristics like rigid regret, simulated annealing[DFV92] or tabu search[JV95].

We want to optimize the initial timetable for the KVB (Kölner Verkehrsbetriebe) such that the constructed timetable was primary robust, i.e. if some vehicles fall behind their schedule they should only need a comparatively small amount of time to get back into it. For this reason we use the safe distance as a criterion, which we try to maximize. The safe distance $\delta(s_i, l_j, l_k, \lambda)$ at station s_i between the lines l_j and l_k is defined by the difference of the arrival time of l_j and the departure time of l_k. The minimum of the pairwise safe distances of all lines crossing station s_i is called the safe distance $\delta(s_i, \lambda)$ at station s_i. We want to compute a departure time vector $\lambda \in \Lambda$ such that

$$\max_{\lambda \in \Lambda} \min_{1 \leq i \leq n} \delta(s_i, \lambda)$$

holds. This objective function favors solutions where the smallest and the largest waiting times that occur when passengers change lines are not to far from the mean waiting time; small waiting times are increased, but large waiting times are reduced. This is due to the cyclic nature of periodic schedules: if the time to change from line A to line B is k, then the time to change from B to A is $T-k$, hence an increase in the first waiting time automatically reduces the second.

Maximizing the safe distance mainly optimizes the timetable at downtown stations where many lines circulate, safe distances at less often serviced stations are considered to be optimal, if they are greater than those at the downtown stations. This can lead to schedules that are very bad for rarely serviced stations, e.g. a safe distance of 2 minutes for two lines departing every 30 minutes would be considered to be optimal just because a safe distance for some downtown station with 5 lines departing every 10 minutes is two minutes, too. Therefore it seems to make sense to use a two-stage optimization problem. At first we optimize with the objective function given above and compute its optimum value. In the second stage we restrict the set of feasible solutions by converting the objective function to an additional constraint. This additional constraint expresses the requirement that for any feasible solution the former objective function must yield its optimum value. We then introduce another criterion (e.g. $\max \sum_{i=1}^{n} \delta(s_i, \lambda)$) as a new objective function in order to select the final solution from the set of all schedules.

Determining an optimal departure vector is NP-hard already for a single station if the lines touching this station have different cycle times[Gul80, Bur86, BBH90]. In the case that all lines have the same frequency the problem (for more than one station) can be shown to be NP-hard by reduction to graph coloring[GJ79]. The following figure sketches how this can be done:

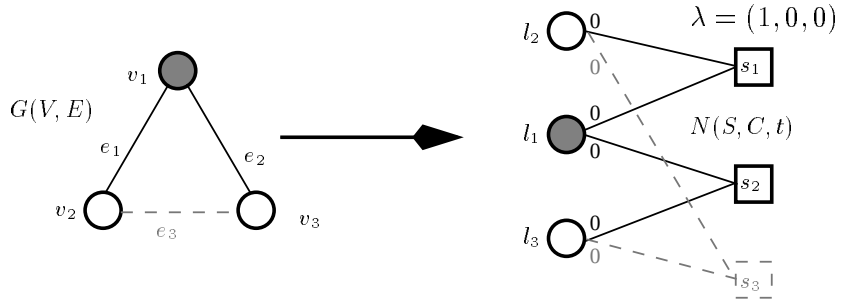


Figure 1: *Reduction to graph coloring*

The graph $G = (V, E)$ is transformed to a (reduced) network $\mathcal{N}(S, C, t)$ by replacing any node from V with a line and every edge with a station. Station s_j and line l_i are connected via an edge in $\mathcal{N}$, if v_i is incident to e_j in G. We choose the arrival time at all stations to be equal to the period of the timetable. Then G can be coloured with k colors iff a departure time vector λ exists for $\mathcal{N}$ such that all lines can circulate with a period of k and the safe distance between any two lines at every station is at least one. In our example G is 2-colorable, because for $\lambda = (1, 0, 0)$ the safe distance is one. If otherwise the edge e_3 is added then G is no longer 2-colorable and no permissible departure vector does exists: Now line l_2 and l_3 arrive at the same time at station s_3.

In this way the problem of constructing initial timetables can be seen as a generalization of graph coloring. A backtracking algorithm based on this technique was able to calculate a timetable for the 14 lines and 19 critical crossing stations of the Cologne public transportation network in less than 1 minute. The algorithm examines the lines in a predetermined order, starting with a fixed departure time for the first line. When the next line is inserted, the safe distances for all T possible departure times are calculated. The backtracking strategy chooses now the departure time with the highest safe distance. The backtracking step is executed, if all lines are inserted or the safe distance of the departure time is to small.

3 Consistency check

As for every discrete optimization problem both establishing the set of constraints as well as testing whether some solution meets all of them is a major part of the problem. The constraints express requirements that must be fulfilled by the schedules in order to make them usable, for example:

- If two lines meet at some crossing they shall arrive at the same time to allow passengers to change from one line to the other and vice versa.
- If a network contains single track parts two trains must not use these parts at the same time in opposite directions
- If two lines serve the same stations and both lines depart every d minutes, then their departure times should be interleaved in such a way that for the "combined" line trains depart nearly every $d/2$ minutes.
- To achieve robustness of the schedule, waiting times at line endpoints should not be too short in order to compensate for possible delays.

These constraints are easily constructed from templates, such that a small number of templates suffices to formulate a great number of constraints. Our tool constructs the constraints given only the templates as inputs by filling the variable parts with suitable values from the timetable, and testing whether the constraint is fulfilled.

Since the input into our program is not the constraint itself but the template, we use a non-standard syntax to formalize these templates.

Any template is a Boolean expression consisting of variables, literals and functions in the style of PASCAL. Variables (lowercase identifier) or literals (uppercase identifiers or numbers) are used for lines, stations, platforms or time values, their types are computed from their occurrence in the expression. Functions have a double purpose, somewhat comparable to predicates in PROLOG. If a variable is used for the first time, its type is bound to the expected type of the argument in the template. Moreover for every possible value a constraint is constructed from the template. The variable is bound to the inserted value, if the same variable is used again in the template, this value is simply inserted. By using $Arr(x, y, z)$ and $Dep(x, y, z)$ as the arrival and departure times of a line x at station y, platform z and $Dist(v, w)$ to compute the time for traveling from v to w, we can express a template for a

constraint that checks whether a single track between two stations A and B is not used in both directions at the same time. We assume that platform 1 is used by northbound trains in both stations, while southbound trains use platform 2 in both stations.

$$Dep(x, A, 1) + Dist(A, B) < Dep(y, B, 2) + 1 \qquad \text{AND}$$
$$Dep(v, B, 2) + Dist(B, A) < Dep(w, A, 1) + 1$$

To facilitate the construction of consistency templates and to limit the number of possible combinations to be tested, we only use nondecreasing and adjacent departure and arrival times, whilst evaluating functions. In the example given above, if trains leave from station A, platform 1 at times $5, 15, 25, \ldots$ and trains leave from station B, platform 2 at times $7, 12, 17, \ldots$, we would only check the constraints for pairs $Dep(x, A, 1) = 5$ and $Dep(y, B, 2) = 7$. The next combination of departure times would then be $Dep(x, A, 1) = 15$ and $Dep(y, B, 2) = 17$, not $Dep(x, A, 1) = 5$ and $Dep(y, B, 2) = 12$. However, since it is possible to assign the departure times to variables, it is possible to perform nearly any calculation. To achieve this, we write e.g.

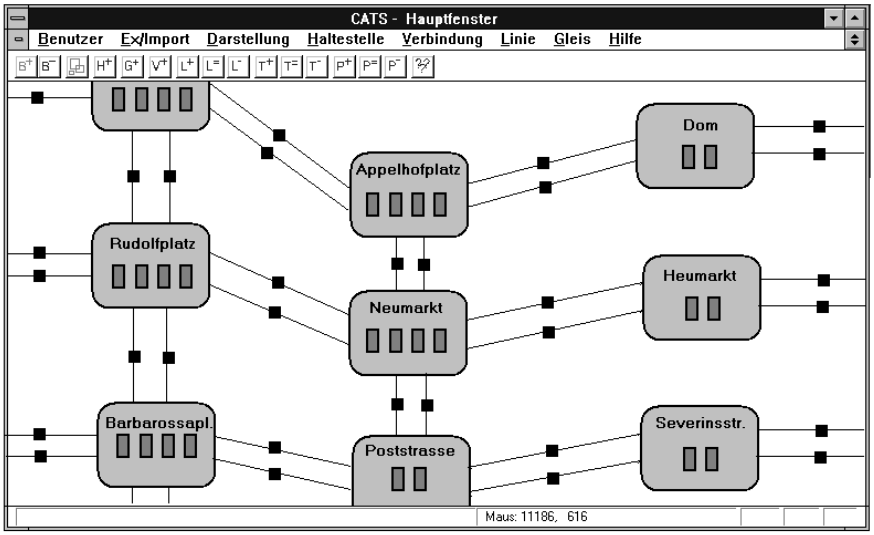
$$u = Dep(x, A, 1) \text{ AND } v = Dep(y, B, 2)) \text{ AND } w = Arr(z, C, 3)) \text{ AND } \ldots$$

and replace ... by an arithmetic expression. Since by convention an assignment is "fulfilled" if any possible values are found, we get the desired behaviour.

To evaluate the constraints the expressions are parsed and translated into an internal byte code, this code is then interpreted to check the constructed timetable against the constraints.

4 Graphical User Interface

To ease the usage of our tools we have implemented a graphical user interface that provides an intuitive support for most of the basic tasks. An example from the construction of a timetable for the Kölner Verkehrsbetriebe is shown below:
The user interface allows treating the plan in variable scales in order to support the different steps of plan construction. During the construction of the network itself the details of the connections are visible, they are hidden in



later phases when the schedule itself is built. In the future we plan to extend this program by simulation tools that allow further testing, especially measuring how fast randomly inserted delays can be eliminated. This permits measuring the robustness of constructed plans under more realistic environment conditions.

References

[BBH90] P. Brucker, R.E. Burkard, and J. Hurink. Cyclic schedules for r irregularity occurring events. *J. Comput. Appl. Mat.*, 30(2):173–189, 1990.

[Bur86] R.E. Burkard. Optimal schedules for periodically recurring events. *Discrete Appl. Math.*, 15:167–180, 1986.

[DFV92] W. Domschke, P. Forst, and S. Voß. Tabu search techniques for the quadratic semi-assignment problem. In G. Fandel, T. Gullegde, and A. Jones, editors, *New Directions for Operations Research in Manufacturing*, pages 389–405. Springer, 1992.

[Dom89] W. Domschke. Schedule synchronization for public transit networks. *OR Spektrum*, 11:17–24, 1989.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Freeman, New York, 1979.

[Gul80] F. Guldan. Maximization of distances of regular polygons on a circle. *Apl. Mat.*, 25:182–195, 1980.

[JV95] J.R. Daduna and S. Voß. Practical experiences in schedule synchronization. In J.R. Daduna, I. Branco, and J.M. Pinto Paixão, editors, *Computer-Aided Transit Scheduling*, volume 430 of *Lecture Notes in Economics and Mathematical Systems*, pages 39–55. Springer, 1995.

[LS96] A. Löbel and U. Strubbe. Wagenumlaufoptimierung - Methodischer Ansatz und praktische Anwendung. In H. Keller, editor, *Heureka '96: Optimierung in Verkehr und Transport*, pages 341–355, Köln, März 1996. Forschungsgesellschaft für Strassen und Verkehrswesen.

[Voß92] S. Voß. Network design formulations in schedule synchronization. In M. Desrochers and J.M. Rousseau, editors, *Computer-Aided Transit Scheduling*, volume 386 of *Lecture Notes in Economics and Mathematical Systems*, pages 137–152. Springer, 1992.

[ZBKW97] U.T. Zimmermann, M.R. Bussieck, M. Krista, and K.-D. Wiegand. Linienoptimierung – Modellierung und praktischer Einsatz. (Line optimization – modelling and practical service). In Karl-Heinz Hoffmann, editor, *Mathematik: Schlüsseltechnologie für die Zukunft. Verbundprojekte zwischen Universität und Industrie*, pages 595–607. Springer, 1997.