Periodic Timetabling in public transport

Alperovich Galina
Wednesday, 16:15
Open Informatics - Artificial Intelligence
shchegal@fel.cvut.cz

Abstract

Mathematical optimization in public transport area is becoming a very popular task for the several last years. Periodic timetables are widely used for local, regional, and long-distance traffic because periodic timetable has the property that certain departure times occure in fixed intervals which ensures a very regular service. Timetable is a very important factor in public transport organization: it helps to optimize city resources, provide mobility and reasonable travel times for many people. In current work we have considered step-by-step model construction, formulation as integer linear programming and two ways of solving the task: classical Branch and Bound and genetic algorithm.

I. Assignment

A. Problem Statement

The problem of finding a periodic timetable in a timetable structure has been described in the literature as the *Periodic Event Scheduling Problem* (PESP). This task is NP-complete, that is, no polynomial-time algorithm has yet been discovered, and a superpolynomial-time lower bound has not been proven. PESP application is a public transport area, so we will consider some vehicles (in our case metro trains) which are moving between stations on the transportation lines. In order to set the formulation of the task, let's define several terms we are going to use in a current work:

Line is a transportation route between two terminal stations. **Demand period** is a fixid period of time, for example from 8 am to 9 am. We will use this term in a context that passengers want to travel from an origin point to a destination point within a demand period. **Frequence for a line** is number of times the line is served depending on the demand period: the more rush-hour traffic is - the higher frequency for demand period.

Periodic Timetabling task is to schedule the trips of all lines which yields the timetable. A trip is the movement of a vehicle between the terminal stations of a line. Scheduling the trips includes **to determine the arrival and departure times at the stations in the transportation network**.

Periodic Timetabling is preceded by several sophisticated tasks such Travel Demand Estimation, Network Planning and Line Planning which are out of this work.

Requirements: While solving PESP task we assume that we have:

1) transport lines network (name of all stations for all lines, stations of line intersection)
2) frequencies for each line for different demand periods; in our work we consider one demand period and one frequency which is common for all lines
3) time constraints for all types of activities: time for vehicle movement between stations, time for vehicle to stay on one station, time for people to change the line.
4) *period time* T - fixed interval between the trips of a line.

TABLE I
An example of a periodic timetable for one station

h	Mondays to Fridays
...	
05	07 17 27 37 47 57
06	02 07 12 17 22 27 32 37 42 47 52 57
07	02 07 12 17 22 27 32 37 42 47 52 57
08	02 07 12 17 27 37 47 57
...	

The period time T usually depends on the frequency of the line which may vary over the day. An example of a periodic timetable is shown in Table I-A. In demand period of rush-hour traffic, from 6:00 am to 8:20 am, the line is operated 28 times. This corresponds to a frequency of 28/140 min. The period time is given as the reciprocal of the frequency, i. e. T = 5 min.

Our task is computing a periodic timetable for a single demand period (for example from 6:00 to 8:20), during which the period time of all lines are assumed be fixed and equals T. The full-day timetable is then made by gluing together the timetables of all demand periods.

B. Mathematical formulation of the Periodic Event Scheduling Problem (PESP)

As we said before our task is determine the arrival and departure times at the stations in the transportation network. An arrival or departure of a directed line at a station is called an *event*. We use the notation *arr(L, S)* for the arrival of line L at station S, and *dep(L, S)* for the departure of line L from station S. Let V denotes the set of all possible events in the transportation network. Let's vector $t \in \{0, \dots, T-1\}^V$ defines an arrival and departure times in the timetable. Also t_v is the *time* of the event v.

A pair of events $(v, w) \in V \times V$ is called an *activity* and we write variable A for the set of all activities. Each activity $a \in A$ is associated with a minimum and maximum allowable time duration, denoted by l_a and u_a, respectively. The interval $[l_a, u_a]$ is referred to as the *time window* of activity a. Some examples of elements from set A (activities):

- *(dep(L, S), arr(L, S'))* is a trip activity, where S and S' denote two consecutive stations on the route of line L.
- *(arr(L,S), dep(L,S))* is a dwell activity, where S denotes a station on the route of line L
- *(arr(L, S), dep(L', S))* is a transfer activity, where S denotes a station where passengers can change between the lines L and L'.

PESP formulation: **With given** requirements, we construct directed *event-activity graph* $D = (V, A)$ with node set $V = \{t_1, ..., t_n\}$ and arc set A. Events represented by nodes, and activities represented by arcs in the graph. Every arc $a \in A$ is associated with the time window $[l_a, u_a]$ of its belonging activity (we have these constraints in requirements). So vectors $l, u \in \mathbb{Q}^A$, and $T \in \mathbb{N}$.

The task is to find for each vertex $t_i \in V$ a number (also denoted by t_i) such that

$$t_j = (t_i + x) \bmod T, \ x \in [l_{ij}, u_{ij}]$$

for all arcs $(i, j) \in E$ is satisfied OR to establish that this is not possible.
The periodicity in this equation can as well be expressed if we introduce a new integer variable p_{ij} for each constraint:

$$t_i + l_{ij} \le t_j + p_{ij}T \le t_i + u_{ij}$$

$$\Leftrightarrow$$

$$l_{ij} \le t_j - t_i + p_{ij}T \le u_{ij}$$

The event-activity graph $D = (V, A)$, the vectors $l, u \in \mathbb{Q}^A$, and the period time T together define a PESP instance. A solution $t \in \{0, \dots, T-1\}^V$ to this instance yields the periodic timetable.

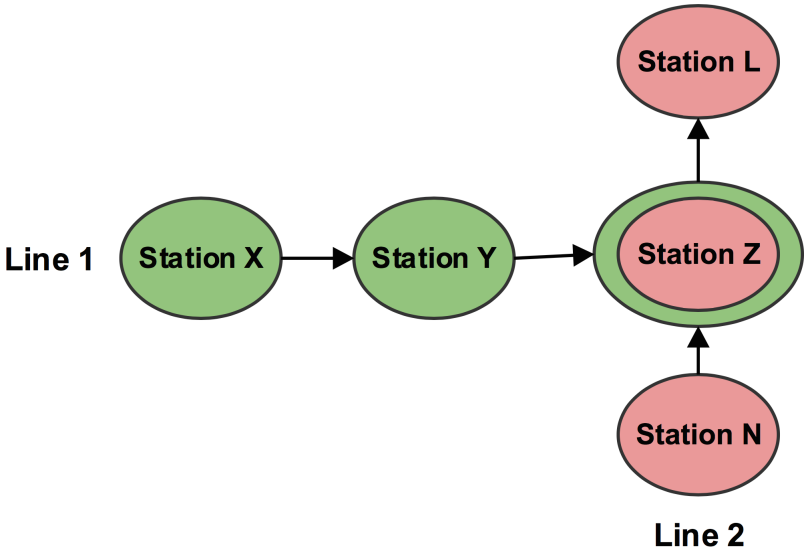
An instance of the PESP with such objective function is called PESP optimization instance and denoted by (D, l, u, w, T). The PESP is NP-complete, since it generalizes Vertex Coloring (Odijk (1997)).

C. Small illustrative example on PESP instance

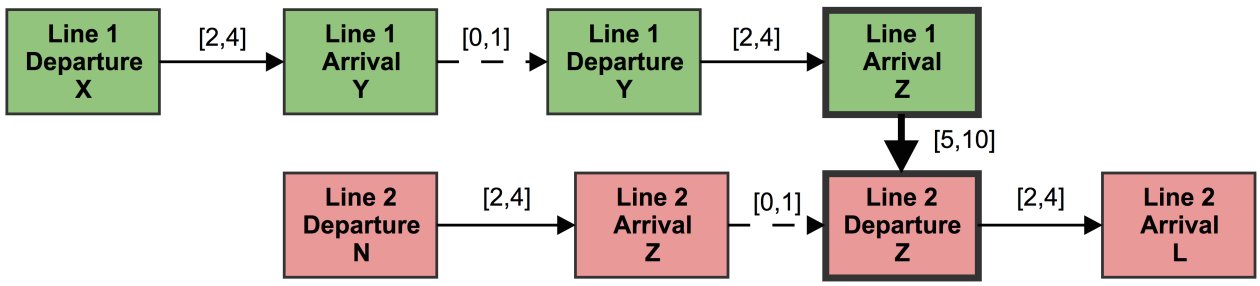
Let's consider small illustrative example in order to understand the problem statement and definition of PESP instance better.

For this task we assume to have a ready line network, frequencies for each line (how often vehicle come to the station), lower and upper bound for all time constraints. For example on the picture below you see simple line structure which consists of Line 1 (stations X, Y, Z) and Line 2 (stations L, Z, N), both lines are operating in only one direction, lines have shared station Z. Let's assume our time period equals 60 (T = 60), that means vehicle visit each station every 60 minutes. Also let's set our time constrains (components of vectors $l, u \in \mathbb{Q}^A$):

- time for vehicle movement between stations: from 2 to 4 minutes
- time for people in order to go out and in to vehicle: from 0 to 1 minutes
- time for people in order to switch the line: from 2 to 4 minutes



As we have mentioned in a previous section, the event-activity graph $D = (V, A)$, the vectors $l, u \in \mathbb{Q}^A$, and the period time T together define a PESP instance. So now we need to construct event-activity graph where each vertex has the following structure: **Line number, Activity type (arrival or departure), Station name**. Such a graph for our line structure you see on the picture below.



Thus for a 2 lines structure with 5 different stations and 1 shared station we have an event-activity graph with 8 vertices and 7 arcs. Green vertexes are related to Line 1, red ones are related to Line 2. Normal arcs show movement between stations, dashed arcs show the fact that the vehicle is staying on the station, bold arc show the fact of switching the line. Above each arc you see corresponding time constraints given before. Such a graph is an event-activity graph and together with vectors $l, u \in \mathbb{Q}^A$, constant T we constructed such called PESP instance. Now the task is to assigne a starting time for each vertex of a graph (you can think about it like this because actually we create one new variable for each such vertex). If we formulate PESP problem with ILP, we will have in general 15 variables (8 for each vertex and one additional variable p for each arc) and 14 constraints (for each arc left and right inequality).

II. Related works

The first applications of the PESP were not concerned with periodic timetabling at all. Introduced by Serafini and Ukovich (1989a) [10] as a model for a periodic job shop, it has also been used for traffic light scheduling (Serafini and Ukovich, 1989b [11]) and airline scheduling (Gertsbakh and Serafini, 1991 [12]). To simplify our work, we investigate the basic model which requires identical line frequencies, i. e., we construct the timetable for a uniform period time $T \in \mathbb{N}$. This condition is not necessary for the Extended Periodic Event Scheduling Problem for which details can be found in Serafini and Ukovich (1989a)

[10] and Nachtigall (1996b) [13].

There are numerous authors that developed different techniques to address scheduling problems and specifically the train timetabling problem. Odijk and Ukovich [10] proposes the PCG (PESP Cut Generation) algorithm to generate timetables for train operations using a mathematical model consisting of periodic time window constraints as input.

Kroon et al. [2] separate the problem in two parts: the generation of the departure/arrival times and the selection of the routes through the stations.

Liebchens [9] work is also based on the PESP, however two optimization variants are presented and discussed: The Max-T-Pesp, and a heuristic method called Cut-Heuristic. Then these two techniques are applied to solve an optimization problem on a real scenario: the subway system of Berlin.

Semets et al. [7] approach focus on the reconstruction of a timetable following a small perturbation. They try to minimize the total accumulated delay by adapting the departure and arrival times for each train and allocation of resources. They use a permutation-based evolutionary algorithm to gradually reconstruct the schedule.

An example of complete use of genetic algorithms to address the problem, Nachtigallt and Voget: [13] use an evolutionary algorithm for the automatic generation train timetables. They presented algorithm in a weak sense *gen* and more comprehensive algorithm *localgen*.

In 2005, the first optimized periodic timetable was successfully put into daily operation for the Berlin subway (see [9]).

III. Problem solution

A. Design

In our work we have solve PESP task with two different approaches. First approach is a classic one and based on ILP formulation and Branch and Bound (BB) algorithm. The simplicity of this task is that there are many well known solvers which solve ILP task with BB method. The complexity of this algorithm is a necessary creation of PESP instance which includes construction of the event-activity graph. Number of vertexes and time constraints are growing dramatically and ILP task is becoming more and more complicated with growth of number of stations.

Second approach is based on genetic algorithm which use several specific parameters as well as different implementation of genetic steps as mutation, crossover and objective function itself. The simplicity of this approach is that we don't need to construct the event-activity graph and formulate all constraints in a ILP manner. Complexity of this approach is experimenting with different algorithms parameters and implementation of genetic operators. Both with genetic algorithm and BB algorithm we reach optimal solution in our task.

As an input data we consider Prague subway map which includes 3 different lines with pairwise intersections.

Algorithm 1: Pseudocode for the first algorithm (ILP task solver with BB)

```
1 function ILPModel (Requirements);
  Input  : List of items from Requirements section: line structure, period time T, time constraints
  Output: vector of timings t ∈ {0,...,T−1}^V for each action
  // From all input data build event-activity graph as it was shown in Illustrative
     example section
2 G = BuildEventActuvityGraph(Input data);
  // By event-activity graph create ILP model: needed variables, constraints and
     objective function
3 ILPModel = CreateILPModel(G);
  // Solve ILPModel with BB method
4 OptimalSolution = SolveILPModel(ILPModel);
5 return OptimalSolution;
```

Algorithm 2: Pseudocode for the second algorithm (genetic algorithm)

function GeneticModel $(Requirements)$;
Input : List of items from Requirements section: line structure, period time T, time constraints
Output: vector of timings $t \in \{0, \ldots, T-1\}^V$ for each action
// Initialize first population
P_{start} = InitializePopulation(Input data)
$P_0 = P_{start}$
t = 0
while *number of iteration is exceeded or objective function is close to zero* **do**
    $P_{offspring}$ = []
    while *needed number of offsprings* **do**
        $P_{parents}$ = SelectParents(P_{start})
        $P_{offspring} = P_{offspring} \cup$ Mutate(Crossover($P_{parents}$))
    end
    $P_{t+1} = P_{parent} \cup P_{offspring}$
    P_{t+1} = Reduction(P_{t+1}) t = t+1
end
return argmin(ObjectiveFunction(P_t))

B. Implementation

ILP Model

To solve the ILP task we need to construct PESP instance from input data. After we constructed event-activity graph, we can walk through this graph and add needed constraints, variables and build objective function expression.

In the following we consider such optimization instances of the PESP. Recall that such instances are given by a directed graph $D = (V, A)$, vectors $l, u \in \mathbb{Q}^A$ and integer T.

t_v timing of event $v \in V$
l_a minimum allowable time duration of activity $a \in A$
u_a maximum allowable time duration of activity $a \in A$
p_a periodical offset of activity $a \in A$
T period time

Every activity in the transportation network has a minimum time duration. Exceeding this time duration affects the quality of the timetable. Thus, we want to penalize the *slack time* $(t_w - t_v - l_a) \bmod T$ for every activity $a = (v, w) \in A$. The objective of the Periodic Event Scheduling Problem (PESP) task is to minimize the weighted sum of the slack times.
We can model the problem of finding an optimal timetable for the PESP instance (D, l, u, T) as the mixed integer program:

$$
\begin{aligned}
\text{minimize} \quad & \sum_{a=(v,w)\in A} (t_w - t_v + Tp_a - l_a) \mod T & \\
\text{subject to} \quad & t_w - t_v + Tp_a \geq l_a, & \text{for all } a = (v,w) \in A \\
& t_w - t_v + Tp_a \leq u_a, & \text{for all } a = (v,w) \in A \\
& 0 \leq t_v \leq T-1, & \text{for all } v \in V \\
& t_v \in \mathbb{Z}, & \text{for all } v \in V \\
& p_a \in \mathbb{Z}, & \text{for all } a \in A
\end{aligned}
$$

Thus we do not directly minimize the total travel time in the transportation network, but we can influence factors like transfer waiting time.

We can build such a model by our event-activity graph: for each vertex we add variable t_i, for each edge we add constraints and create variables p_j.
In order to simplify the process we can construct special matrix *edge - vertex* M for the graph in such way:

$$M_{(t_i,t_j),t_k} = \begin{cases} 1, & \text{if j = k} \\ -1, & \text{i = k} \\ 0, & \text{otherwise} \end{cases}$$

That means for example if we have two vertexes t_1 = (Line 1, Departure, X) and t_2 = (Line 2, Arrival, Y) with time constraint [2,4] and edge from t_1 to t_2, then we will have $M_{(t_1,t_2),t_1}$ = -1 , $M_{(t_1,t_2),t_2}$ = 1 and 0 for all other vertexes. Then multiplying such matrix by vector $(t_1, \ldots, t_n)$ we will get expression $(t_2 - t_1)$ which is needed for constraints as well as for objective function.

After walking through the event-activity graph in such way we have constructed ILP model with all needed variables t_i, p_j and constraints. Such model can be solved by any BB solvers.

Genetic Model

Algorithm for a genetic model was described above in Algorithms section. We operate with individual string where each ith bit is a value of variable t_i. In genetic algorithm we can play with many parameters and implementations of genetic operations. In the list above we have stated those ones which were implemented in current work:

- Number of individuals in parent population and offspring population
- Implementation of initial population
  - Individuals from random bits constrained by main constraint $0 \leq t_v \leq T - 1,$ for all $v \in V$
  - Improved initialization: choose first variable from the line randomly and set others with lowest possible interval.
- Implementation of crossover of two parents for creating a new offspring:
  - Uniform crossover for each bit of parents
  - Uniform crossover for each line part of parents (in order to keep consistency in one line)
  - One point crossover
- Implementation of mutation algorithm under each offspring
  - Mutation in each bit with different probability
  - Mutation in a variables with correspond to beginning of the lines and fixed others.
- Implementation of selection algorithm

IV. Experiments

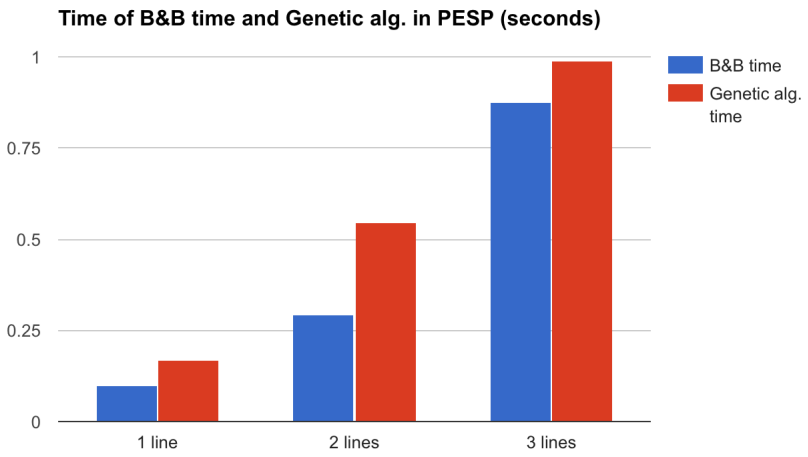
A. Benchmark Settings

Input instances is a Prague subway which consists of 3 intersecting lines and 61 station. With bigger number of stations there were a problem with Gurobi Academical license.
Computer details: 2.5 GHz Intel Core i5, 8 GB 1600 MHz DDR3, OSX Yosemite 10.10.5.
Everything was implemented with Python programming language in PyCharm IDE, with using of Gurobi solver for ILP part of the task, self-implemented functions for genetic algorithm part and NumPy library for matrix computation.

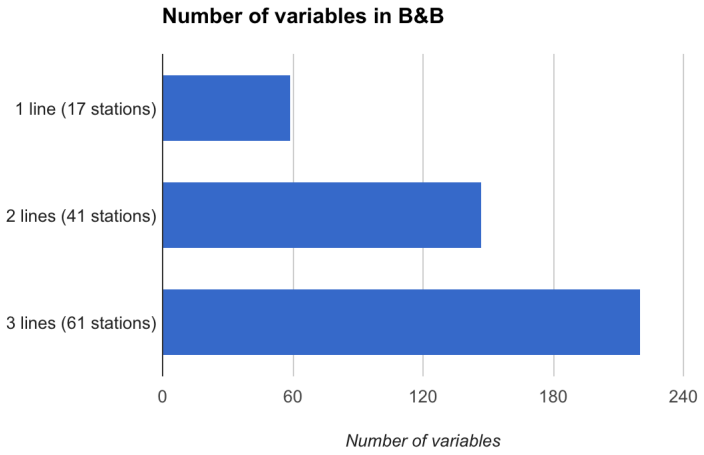
Data instance were taken from Wikipedia article about Prague metro stations https://en.wikipedia.org/wiki/Prague_Metro.

B. Results

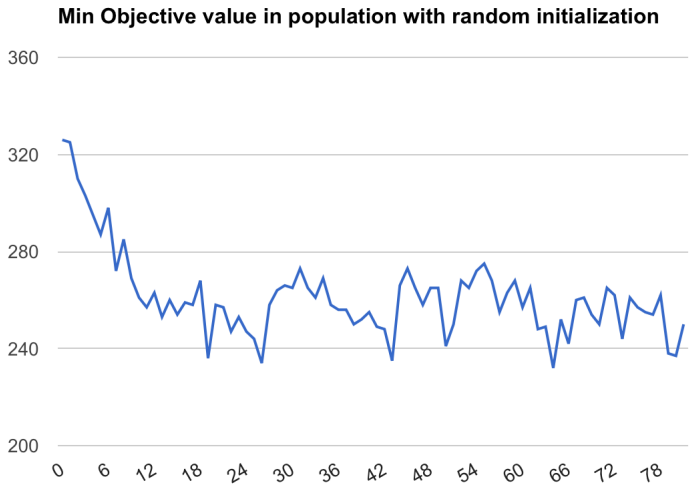
We have considered different number of lines in order to compare these two approaches in different data input instances. You can see that BB algorithm is faster than genetic algorithm. In previous work [5] the same result was shows: for small instances BB algorithm performs better. In our case we have a small transportational network, that is way BB performs better too. We could not check this approach for bigger instances because of limitations of Gurobi Academical license.



Here is on the picture you can see the number of variables in ILP part when you add new transportational lines:



While experimenting with parameters and implementation of genetic operators in genetic algorithm part, we have faced with several difficulties also described in previous work with genetic algorithm approach [13]. With random initialization genetic algorithm can be unconvertible to the optimal solution. See the picture above:



We have improved the algorithm with better initialization population with allows the objective value converge. This method

of initialization is based on random first variable in a line and the rest fixed intervals for this line. The value of the interval is a lowest possible value for current edge.

C. Discussion

We have implemented two different methods for solving PESP task. All results which we have correspond with previous research results (on small instances). It was expected that BB algorithm works better on small instances due to very fast implementation in Gurobi solver. Genetic algorithm is fully implemented by ourself and written in Python, so probably that is one one the reason why it takes longer time.

As we were mentioning above these two approaches have their own advantages and disadvantages.
Classic ILP approach with BB algorithm is simple to solve with ready made solvers, but it is complicated to create PESP instance which includes construction of the event-activity graph. ILP task is becoming more and more complicated with growth of number of stations and lines.

Genetic algorithm is simple to implement but difficult to set all parameters and choose right implementations. Also it is important that all our variables are related with each other depending on line structure and number of line. That's why we need to implement special crossover operator which recognize the number of line in a string of bits. Advantage of this algorithm is that the PESP instance can be scaled a lot and genetic algorithm should still solve everything in a reasonable time. It was shown in previous research that genetic algorithm handle big instances easier than BB approach.

V. Conclusion

In current work we have considered very interesting optimization problem PESP and implemented two algorithms for solving the task: BB method which require event-activity graph construction and genetic algorithm with different modifications. In current work detailed PESP construction was described. All results are agreed with previous works. Also several modifications were made in implementations of genetic algorithm, these modifications helped to converge much faster. Everything was implementing on Python and open-sourced on GitHub (there were not PESP-related work there yet), so the code can be used for further experimentations.

References

[1] Malte Helmert *Models for Periodic Timetabling*
[2] Karl Nachtigallt and Stefan Voget *A Genetic algorithm approach to periodic railway synchronization*
[3] Carole Giesemann *PERIODIC TIMETABLE GENERATION: Seminar on Algorithms and Models for Railway Optimization*
[4] Mathias Kinder *Discrete optimization in public rail transport*
[5] Christian Liebchen, Mark Proksch, and Frank H. Wagner *Performance of Algorithms for Periodic Timetable Optimization*
[6] Christian Liebchen and Rolf H. Mhring *The Modeling Power of the Periodic Event Scheduling Problem: Railway Timetables and Beyond*
[7] Thomas Lindner *Train Schedule Optimization in Public Rail Transport*
[8] Laura Galli and Sebastian Stiller *Strong formulations for the Multi-module PESP and a quadratic algorithm for Graphical Diophantine Equation Systems*
[9] Liebchen, C. *Periodic Timetable Optimization in Public Transport.*Ph.D. thesis, Technische Universita t Berlin.
[10] Serafini, P. and Ukovich, W. (1989a). *A mathematical model for periodic scheduling problems.* SIAM Journal on Discrete Mathematics, 2(4), 550581.
[11] Serafini, P. and Ukovich, W. (1989b). *A mathematical model for the fixed-time traffic control problem.* European Journal of Operational Research, 42(2), 152165.
[12] Gertsbakh, I. and Serafini, P. (1991). *Periodic transportation schedules with flexible departure times : An interactive approach based on the periodic event scheduling problem and the deficit function approach.* European Journal of Operational Research, 50(3), 298309.
[13] Nachtigall, K. (1996b). *Periodic network optimization with different arc frequencies.* Discrete Applied Mathematics, 69(1-2), 117.