

## Algorithmic support for human rail shunting planners

Special session on "Human and Organisational Factors in Industrial Planning and Scheduling – HOPS"

W.M.C. van Wezel, J. Riezebos

Faculty of management and Organization

University of Groningen

Postbus 800, 9700 AV Groningen, The Netherlands

### Abstract

Traditionally, algorithms that generate schedules are either based on domain analyses or on task analyses. Both approaches have disadvantages. Algorithms based on domain analyses are often not used by human planners and schedulers. Algorithms based on task analyses, however, do not use the potential improvements in quality that domain based algorithms can attain. In this paper, the advantages of both approaches are combined by applying a mixed approach. Combinatorial optimization algorithms have been developed and implemented for two subtasks that were identified in a task analysis of shunting planners at The Netherlands Railways. Because of the approach taken the algorithms are usable in practice, which can improve both the quality and the speed of generating plans.

### Keywords:

Train shunting scheduling, K-shortest path algorithm, task analysis

## 1 INTRODUCTION

The field of operations research (OR) has a long-standing reputation in supporting planning decisions. Among the planning problems studied, transportation problems, routing problems, and assignment problems are classical examples to which the OR community has been contributing. The contribution often consists of specific algorithms that can be applied to find an optimal solution to a planning problem. The task of the planner is to provide the algorithm with the correct input and specify the weights in the objective function.

Reports on the actual usage of the algorithms that are developed for planning support are scarce. In the field of transportation scheduling and railway planning, planners often prefer manual planning to the use of algorithms. A recent survey of 153 papers on train routing and scheduling problems concludes: "even though most proposed models are tested on realistic data instances, very few are actually implemented and used in railway operations" [1].

Empirical research has provided insight into the applicability and usage of techniques that have been designed for production planning and scheduling [2][3]. Some areas and problems are more receptive than others, especially if the problem size is limited and model conditions can be specified with reasonable certainty. For detailed planning and scheduling, algorithms have been of limited value. Four reasons are: (1) inflexible models that are not able to cope with changing environments, (2) higher level priorities that are not correctly taken into account, (3) algorithms that are not able to cope with uncertainty regarding the actual state of the system, and (4) objectives of the algorithms that are only soft constraints and for the system as a whole only of limited value [2].

These conclusions seem also valid in the related field of transportation planning and railway scheduling. Research has shown that the reasons for not using optimization models in railway scheduling are (1) a lack of integration with software that fits the needs of the users, (2) a lack of co-operation between software developers and model builders, (3) a lack of extendibility of the models to real-life circumstances, and finally (4) no international applicability of the models [4].

We suggest that the underlying reason for not using these models is the design approach that is applied. Traditionally, model builders use a problem-oriented approach, focusing on the conceptualization of the planning problem as viewed by the experts. This approach ignores information on the strategies that are used by the planner in solving the problem, but focuses directly on the total problem that has to be solved by the planner. If the planner is not explicitly considered as an object of study in the design of the support system, two inherent dangers have to be faced. First, the planner may not accept the solutions generated by the support system, as he has no insight into the way these solutions are constructed. Second, the quality of the solutions may deteriorate over time due to the fact that the optimization model is not a perfect representation of the planning problem faced by the planner. Constraints that were neglected in the optimization model can at some moment in time become important for the planner to include in his solution approach. The life cycle of an optimization model is therefore often shorter than assumed by the original designer [5].

This paper proposes and applies a planner-oriented focus in designing OR-algorithms for a railway planning support system. This approach is compared with the traditional problem-oriented focus in OR design and a planner mimicking approach. The planner-oriented focus results in a different way of designing OR-algorithms. We show that a planner-oriented focus in designing OR-algorithms support the task of the planner without encountering the inherent dangers of a problem-oriented focus.

The paper is organized as follows. Section 2 describes the approach we use to analyze the task of the planner. Section 3 applies this approach to the problem of train shunting scheduling and describes the solution strategies of the planner. Section 4 describes the design of specific algorithms that support the task of the planner. Finally, section 5 presents conclusions and provides suggestions for further research.

## 2 BOTTOM-UP ALGORITHMIC PLANNING SUPPORT

Planning in organizations is a complex task that should be supported adequately. This notion is widely acknowledged and endorsed in both theory and practice. Furthermore, it is also generally acknowledged that planning support

should be more than a bunch of algorithms with a Gantt-chart. Notwithstanding the abundant research results on planning and scheduling that are published and the colorful scheduling support systems that are available, dedicated shunting planning support is not widely applied in practice [6]. Mostly, the use of the computer in the daily planning task in organizations is restricted to providing information (e.g., routing and processing data, order status and information). Actual support in the planning task is often limited to software for editing a plan. That software can be seen as a kind of word processor for planning. Plans can be copied, altered, printed, saved, and some basic calculations can be made. In many cases, the system that is used actually is a word processor or spreadsheet. Dedicated scheduling systems provide algorithms or heuristics that can generate solutions. Such algorithms are based on detailed analyses of the domain, or, in other words, analyses of the characteristics of the entities that are scheduled. In manufacturing, for example, such characteristics are machine capacity, job characteristics, routings, etc. This generative support, however, is used seldom in practice [4]. One of the reasons is that automated schedule generation often leaves little room for human control in the search process [7]. Researchers in the field of human factors argue that analytical models cannot deal adequately with uncertainty and instability of the real world [2][8][9][10].

The human emulation approach takes a different starting point than the domain-oriented approach. In a human emulation approach, algorithms emulate the decision process of a human planner. Such algorithms reduce the time necessary for performing the planning tasks. Other reasons to use this approach are: (1) algorithms embody knowledge of the planners on the problem, priorities, and data; (2) confidence of planners in solution methods is being build up by mimicking their behavior; and (3) planners can easier explain the outcomes of the planning process to other parties [7]. However, a disadvantage of such a human emulation approach is the lack of actual support of the planner during the planning process. Algorithms replace the planner instead of providing support. The inherent danger of such an approach is a lack of adaptability of the planning process to changes in the real world. Furthermore, mimicking the human planner will also mimic the bad habits of humans such as short term fire fighting, small working memory, terrible capacity to calculate uncertainties, and other limits imposed by our bounded rationality.

The aim of the shunting planning support project at the Netherlands Railways is to investigate if a way in between the analytical and human emulation approaches can be found. We want both to include algorithms in the planning support and let the human planner use them while still being in control.

The design of such algorithms requires a different model-building process. From the field of cognitive science it is known that a decision maker implicitly makes a trade-off between the (cognitive) cost of applying a decision aid (efforts to understand and employ the model and process the information) and the expected benefits (increased quality and speed of obtaining a solution). Decision aids (such as a planning tool or an algorithm) can be improved by applying the following three steps:

- Decompose the planning problem into sub problems and obtain estimates for the efforts (costs) to manually find solutions to these sub problems.
- Identify the sub problems with a high potential of effort (cost) reduction for the decision maker and identify a decision aid that reduces the total effort to find and use a solution for such a sub problem.

- Incorporate specific features for automating storage, retrieval, and computational tasks in the decision aids to manipulate the cognitive effort associated with using these decision aids [11].

In order to investigate whether this approach will also be worthwhile for scheduling and planning, we have developed a task-oriented scheduling system prototype for the shunting planners at the Netherlands Railways. This prototype implements the idea that algorithms should be created for subcomponents of the task strategy to support the problem solving process. The focus is on the level at which the system and the user communicate [12]. In other words, human and system should be problem solving in coincident problem spaces [13]. In our research, we try to find such coincident problem spaces by looking at the subcomponents of the task strategy of human planners.

In applying a task-oriented approach, we need to analyze the way in which a human planner makes a plan. First, we apply a task analysis. A task analysis describes the activities that constitute the task and the order in which the activities are carried out [14]. Analyses of planners have shown that planning tasks are performed in a hierarchy. Each activity, or subtask, is performed by a number of 'smaller' activities itself [15].

Next, we design algorithms bottom-up: first, analyze the planning subtasks; next, decide per subtask what kind of support is needed. In other words: algorithms are not created for the total planning problem, but for the planner's subtasks. As the resulting algorithms are closely related to the activities that a planner performs, we expect an increased chance in the actual usage of the system by the planner, without the risk of adopting all his non-optimal habits.

Applying algorithms to subtasks has the following advantages and possibilities [16]:

- The chance that the human planner will accept algorithms and their outcomes increases.
- Existing divisions of planning problems into sub-problems (which a human planner has learned by experience) can be reused in algorithmic design.
- Algorithms for subtasks can be used automatically in a sequence. If an algorithm is available for each subtask in a task, then they can be executed in one step (the 'push the button and get a plan'-approach).
- Algorithms for subtasks can be used interactively in a sequence. Instead of automatically executing the algorithms for a sequence of subtasks, they can be executed semi-automatically by providing the planner with a way to (manually) interfere after the execution of each algorithm.
- Algorithms can be applied under conditions chosen by the planner. For example, a production planner might want to let the computer plan production orders automatically, except when the capacity usage exceeds 90%.
- Designing algorithms for subtasks is less complex than for whole tasks.
- Different planners use different task strategies, i.e., they perform subtasks in different sequences [17]. Algorithms can be executed in various sequences and can therefore be used in different task strategies.

The next section describes the problem-solving task that a shunting planner at the Netherlands Railways faces each day.

### 3 PROBLEM: TRAIN SHUNTING SCHEDULING

In the Netherlands, most passenger trains stay at a station during the night. They arrive at the end of the day and depart the next day in a possibly different configuration of coaches and probably from a different track. During the night, they must be parked at one of the tracks in the shunting area; otherwise they would block the tracks that are needed for incoming and outgoing trains. Multiple trains can be put on a track in the shunting area simultaneously, but the total 'storage' capacity at a station is limited. Additionally, all trains must be cleaned both internally and externally during the night at a track that contains the cleaning equipment. The task of the shunting planner is to plan the movements of the trains and coaches, to decide on what tracks trains stay during the night, and to determine when and in what configuration they will be moved to the washing track. To plan the movements, the planner must also assign train drivers, train shunters (employees that amongst other tasks connect and disconnect coaches), and routes of the trains on the station.

The research project studied the planning task of planners that are responsible for making short-term adjustments (one week ahead) to already created plans (stage 2 of the planning process as described in [7]). Some of the constraints and goals that the planner takes into account are that at least two routes should be kept free for traffic going through the station, that only one movement may take place on a track in a three minute time window for safety reasons, that trains should be washed and cleaned each day, that the train length may never exceed the track length, etc.

The planning tasks are performed manually. Some computer programs are used to collect information, but the plan itself is made on paper before it is put in the computer. From the total group of 130 planners that performs the shunting planning in The Netherlands, about 60 are involved in planning these short-term adjustments. The short-term planners are geographically specialized, such that each planner performs these tasks for a limited number of stations.

In the research project, the task structure and task strategy of an experienced planner was extensively analyzed. The planner was requested to solve actual problems and to think aloud during his work. The thinking-aloud protocols were analyzed, after which the decisions made by the planner were actively discussed in a number of interactive sessions.

## 4 DESIGN OF ALGORITHMS

### 4.1 From subtask to algorithm

Each step or subtask that a human planner performs can recursively be divided further into more detailed subtasks. Identification of subtasks that are candidate for algorithmic support depends in a planner-oriented approach on the efforts associated with a specific subtask. The more frequently the subtask is repeated, or the more complex a subtask is, the more time that can be saved by using a decision aid. The remainder of this section describes two algorithms that were designed and implemented in the prototype of the planning system: finding a route to move the train from one track to another, and finding a number of tracks that a train can be parked on during the night. These two elementary subtasks are used in many of the higher level subtasks.

### 4.2 Design of routing algorithm

The repetition of the route-finding subtask is high; each time a train is moved, a valid route through the emplacement has to be found. The input for this task is the

infrastructure of the emplacement, the occupation of tracks during the time window of the movement, the originating track, and the target track. The output of the subtask is the shortest feasible route for shunting the train to the proposed track. As a cost function (or distance measure), the number of times a train changes its direction is used. At each direction change, the driver must get out of the cabin and walk to the other side of the train (all trains have a cabin on both sides) to continue driving in the opposite direction. This is very time consuming and therefore costly. A route is feasible if it consists of connected tracks that can be used by the specific train within a short interval of three minutes and if the direction changes occur at tracks that have enough space to hold the complete train. The planner may block several tracks if he wants them to be available for other purposes. If there are more solutions with the same number of direction changes, the alternatives should be ordered according to their total mileage.

This description contains several elements that point to a shortest path algorithm, which finds the shortest route from a source node to a sink node in a network of connected nodes. Several algorithms are available from operations research and computer science. Dijkstra was the first to present an algorithm that minimizes the distance from the source node to all other nodes if arc lengths are non-negative [18]. This algorithm was generalized to determine the shortest distance between all nodes in the network (graph) [19][20]. An efficient single pass algorithm to determine the K shortest paths in an undirected graph is available [21]. A review of different K-shortest path algorithms for directed and undirected graphs can be found in [22].

The question is which of these shortest path algorithms fits best with the requirements specified for the subtask and how this algorithm has to be tailored in order to solve the subtask. First, we note that the two objectives (primary objective: minimize the number of movements and secondary objective: minimize the traveling distance) can be transferred to a single objective by using a weight function with the weight per movement being larger than the largest traveling distance of any route. Next, we design our network. All tracks that we might use in the route are defined as nodes. The arc length between any pair of nodes is infinite unless the tracks of the two nodes are physically connected and free to use. Then, the arc length equals the distance between both tracks.

For this objective function and network, we modified the undirected K-shortest path algorithm [21] and determine the K shortest paths from the source track to the destination track. Modification of the algorithm was necessary in order to determine the occurrence of direction changes in a path. This modification will now be discussed in detail using the example in figure 1:

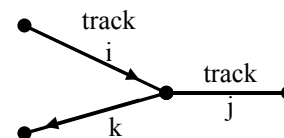


Figure 1: To go from track i to track k, a train needs track j

Direction changes are allowed at a limited number of tracks, because of the required track length and the position of safety signs. First, a node representing track i is included in the path. Next, it is not possible for the algorithm to select the node representing track k directly, because a train cannot make a turn like a car. So, in order to be able to go to k, it first has to select the node representing track j and change direction at j. Both tracks i

and  $j$  are connected, so this arc length will be finite. If the algorithm now selects the node representing track  $k$ , it will result in a direction change at track  $j$ . If it is not allowed to perform direction changes on track  $j$ , the arc length between  $j$  and  $k$  should be infinite, notwithstanding the actual distance being finite. If a direction change is possible at track  $j$ , the arc length between node  $j$  and node  $k$  should be increased with the cost of one direction change. The algorithm therefore takes the information on the preceding node into account when determining the arc length. It is possible to use an alternative directed graph that allows for a direct calculation of direction changes, but this increases the number of arcs and nodes with a factor larger than two.

### 4.3 Design of track-finding algorithm

The second algorithm that is developed aims at supporting the subtask in which a planner tries to find a number of consecutive tracks at which a train can stay during the night. Criteria that will affect the decision to what track the train should be moved are amongst others: the length of the time interval it can stay at this track, the routing distance to this track, the previous activities of driver and/or shunter, and the consequences for future actions with this train (i.e., internal cleaning, external cleaning, routing to the track from which it has to leave in the morning, etcetera).

Due to the large number of criteria, the algorithm for this subtask provides several alternative solutions (tracks), accompanied by relevant information on the criteria mentioned. The planner is often not able to overview the consequences of his decision for all criteria when planning manually. The algorithm takes these criteria into account when evaluating alternatives and presents these alternatives with their consequences to the planner.

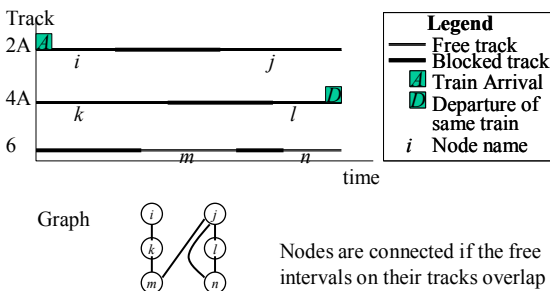


Figure 2: Graph construction from free intervals

The algorithm first determines the total set of available time windows (intervals) on tracks from the moment of the actual move. These intervals are train-specific, as the length or other characteristics of the train constrain the possibility of locating it at specific tracks. An interval has the characteristics of the track to which it is associated. The problem that the algorithm has to solve is defined as finding a sequence of partially overlapping time intervals from the moment of the actual move to the moment of departure. The intervals were considered to be the nodes in a network, as illustrated in Figure 2. Two nodes have a finite arc length if they partially overlap in time. Node  $i$  and  $j$  partially overlap if the end time of interval  $i$  is greater than or equal to the start of interval  $j$ , while the start of interval  $i$  is smaller than or equal to the end of interval  $j$ . The length of the arc between these nodes is defined as the cost of the shortest route between the two tracks with which the intervals are associated (see section 4.2). We assume this route will be available somewhere during the overlapping time window. The objective of the algorithm is to find the shortest sequence of intervals from start to destination. We use a weighted criterion function that weighs the total

number of track changes required in order to reach the destination with the total distance of the routings between the tracks involved. In general, the planner will try to avoid track changes, so the weight of a track change will dominate the routing distance.

An additional feature of the sequence is that it has to resemble the necessity of visiting the cleaning track somewhere over time. Therefore, the algorithm will have to include the possibility of stating a set of intermittent nodes (i.e., intervals on the cleaning track) from which at least one has to be included in the final sequence before the departure track is reached. Finally, it has to be possible for the planner to block several tracks that may not be included at all in the final sequence, as they have to be reserved for other purposes.

The problem can again be solved using a K-shortest path algorithm. We tailored the algorithm of section 4.2 to solve this problem efficiently. We first determine the shortest routings between all tracks at which this train may be located. The network for this first step therefore consists of only one node per track. We repeat the Dijkstra shortest path algorithm [18] for each track to determine the shortest distance between tracks, resulting in an  $O(1/2N^3)$  complexity (with  $N$  the number of tracks). As all arc lengths are non-negative, this is better than obtained by others [19][20]. Note that this step has only to be performed once. Next, the algorithm creates a new graph, consisting of intervals as nodes (see Figure 2). In general, this network will contain far more nodes, depending on the utilization of the tracks over time. We apply the extended K-shortest path algorithm of section 4.2 to this network, using weights for track changes and total traveling distance. The condition of including at least one of the intermittent nodes in the K shortest paths has been taken care of by applying a two-stage solution process. The first stage determines in a single pass of the algorithm the K shortest paths from source node to each intermittent node. The second stage applies the same algorithm for finding the K shortest paths from the destination node to each intermittent node. This approach is more efficient than applying a single pass all-K-shortest paths approach that determines the K shortest paths between each pair of nodes in the network. The two-stage solution approach has proven to be successful in the design of flow-shop scheduling algorithms and heuristics [23]. Finally, we present the best solutions that are found to the planner, accompanied by the information on the consequences of this choice for the moment of cleaning, the future track changes required for reaching the departure track, the total distance of the first move as well as the later moves in this sequence, and the consequences for the train drivers and shunters.

The design of this algorithm shows that it is possible to reuse the same algorithm in different subtasks. It shows the relations between tasks, subtasks and algorithms. Most of the time there is a singular relationship between subtask and algorithm. However, in some cases we can design OR algorithms such that they are able to support different subtasks.

## 5 CONCLUSIONS AND FUTURE RESEARCH

The current paper applies a bottom-up approach to algorithmic planning support. Thereby, methods and techniques from both problem-oriented approaches and mimicking approaches are used. The implementation of the algorithms in the prototype makes it possible to compare the bottom-up, planner-oriented approach to the other approaches. Three published examples of the different design approaches are compared in table 1. The comparison shows that the approaches result in different algorithms and computing times, as well as a different kind

of involvement of the planner in the decision making process.

The bottom-up design approach has several advantages. First, the algorithms are smaller and more robust, as they solve only a part of the whole task. Second, the same algorithms can be reused in different tasks, as identical subtasks will arise in different tasks. Third, the planner better understands the outcome of the algorithms, as the problem solved by an algorithm is rather small. Finally, maintenance of the algorithms becomes easier because of the modular design approach applied. If circumstances change, only a small number of algorithms have to be modified.

The hierarchical view on the planning task and generation algorithms that are applied in this paper provides stability and robustness for the design of planning support systems. For example, no matter what kind of strategy the planner uses, he will always need to perform small low level tasks such as finding routes, selecting train drivers, matching incoming trains to departing trains, etc.

After the OR algorithms were implemented and integrated in the prototype, the efficiency of the algorithms could be evaluated by comparing the time it takes to solve problems of several levels of complexity. However, the concept of planner-oriented design of algorithms is more difficult to evaluate. Whether the algorithms effectively support the task of the planner is currently being researched by empirically testing several kinds of support with a large number of shunting planners.

## 6 ACKNOWLEDGMENTS

This study has been supported by the Netherlands Railways. We gratefully acknowledge the management and planners of this company. Specifically, we want to thank Dr. L. Kroon, Bas Barten, and the planners of the Netherlands Railways, location Zwolle, for their willingness to co-operate.

## 7 REFERENCES

- [1] Cordeau, J. F., Toth, P., & Vigo, D. 1998, "A survey of optimization models for train routing and scheduling", *Transportation Science*, vol. 32, no. 4, pp. 380-404.
- [2] Buxey, G. 1989, "Production scheduling: practice and theory", *European Journal of Operational Research*, vol. 39, pp. 17-31.
- [3] Buxey, G. 1995, "A managerial perspective on aggregate planning", *International Journal of Production Economics*, vol. 41, no. 1-3, pp. 127-133.
- [4] Watson, R. 2000, "Prospects for computer aided railway scheduling: perspectives from users and parallels from mass transit", *Transportation Planning and Technology*, vol. 23, no. 4, pp. 303-321.
- [5] Ackoff, R. L. 1957, "The concept and exercise of control in operations research," in *Proceedings of the first international conference on operations research*, The English universities press Ltd, Oxford, pp. 26-43.
- [6] Allan, J. J., Mellitt, B., & Brebbia, C. A. 1996, *Computers in Railways V, railway systems and management Computational mechanics publications*, Southampton.
- [7] Carey, M. & Carville, S. 2003, "Scheduling and platforming trains at busy complex stations", *Transportation Research Part A*, vol. 37, pp. 195-224.
- [8] McKay, K. N., Safayeni, F. R., & Buzacott, J. A. 1988, "Job-shop scheduling theory: What is relevant?", *Interfaces*, vol. 18, no. 4, pp. 84-90.
- [9] McKay, K. N., Safayeni, F. R., & Buzacott, J. A. 1989, "The scheduler's knowledge of uncertainty: the missing link," in *Knowledge based production management systems*, J. Browne, ed., North-Holland: Elsevier Science Publishers.
- [10] Sanderson, P. M. 1989, "The human planning and scheduling role in advanced manufacturing systems: an emerging human factors domain", *Human Factors*, vol. 31, no. 6, pp. 635-666.

Table 1: Comparison of algorithmic planning support approaches

	Problem-oriented	Planner-oriented	Mimicking
Example approach	Freling, Lentink, Kroon & Huisman [24]	Current paper	Carey & Carville [7]
Source Data	Zwolle (The Netherlands)	Zwolle (The Netherlands)	Leeds (UK)
Type of OR algorithms	MIP and Column generation	Combinatorial optimization	Scheduling heuristics
Role planner in design process	Provides problem-related knowledge	Provides both approach- and problem-related knowledge	Provides approach-related knowledge
Role planner in solution process	Responsible for data input and weight factor setting	Responsible for solution process, decides on actual use of supporting algorithms	Planner is replaced by computer
Run time algorithm	20-40 minutes (imposed maximum)	2-5 seconds	Between seconds and a few hours, depending on the rules used
Number of solutions presented	1	5	1
Optimality gap	0-20%	0% for subtask, gap for total problem depends on planner	Heuristic, quality gap depends on planner that is mimicked
Completeness of solution	Incomplete (cleaning not considered)	Incomplete (planner supported, not replaced)	Complete

- [11] Benbasat, I. & Todd, P. 1996, "The effects of decision support and task contingencies on model formulation: A cognitive perspective", *Decision Support Systems*, vol. 17, pp. 241-252.
- [12] Newell, A. 1981, "The knowledge level", *AI Magazine*, vol. 2, no. 2, pp. 1-20.
- [13] Prietula, M. J., Hsu, W.-L., Ow, P. S., & Thompson, G. L. 1994, "MacMerl: Mixed-initiative scheduling with coincident problem spaces," in *Intelligent scheduling*, M. Zweben & M. S. Fox, eds., Morgan Kaufmann Publishers, San Francisco, California.
- [14] Schreiber, G., Akkermans, H., Anjewierden, A., De Hoog, R., Shadbolt, N., Van de Velde, W., & Wielinga, B. 2000, *Knowledge Engineering and Management: The Common-KADS Methodology* The MIT Press, Cambridge, MA.
- [15] Wezel, W. v. 2001, *Hierarchies, and Flexibility; Planning in Food Processing Industries*. University of Groningen, Capelle a/d IJssel.
- [16] Wezel, W. v. & Barten, B. 2002, "Hierarchical Mixed-Initiative Planning Support," in *Plansig 2002*, T. Grant & C. Witteveen, eds., Delft University of Technology, Delft.
- [17] Mietus, D. M. 1994, *Understanding planning for effective decision support*. University of Groningen, Groningen.
- [18] Dijkstra, E. W. 1959, "A note on two problems in connection with graphs", *Numer. Math*, vol. 1, pp. 269-271.
- [19] Floyd, R. W. 1962, "Algorithm 47, shortest path, communication", *ACM*, vol. 5, p. 345.
- [20] Dantzig, G. B. 1967, "All shortest routes in a graph," in *Theory of graphs, international symposium, Rome, 1966*, Gordon and Breach, New York, pp. 91-92.
- [21] Shier, D. R. 1976, "Iterative methods for determining the k shortest paths in a network", *Networks*, vol. 6, pp. 205-230.
- [22] Eppstein, D. 1998, "Finding the k shortest paths", *SIAM J.Computing*, vol. 28, no. 2, pp. 652-673.
- [23] Morton, T. E. & Pentico, D. W. 1993, *Heuristic scheduling systems; with applications to production systems and project management*. John Wiley, New York.
- [24] Freling, R., Lentink, R. M., Kroon, L. G., & Huisman, D. 2002, "Shunting of passenger train units in a railway station", *Econometric Institute Report Erasmus University Rotterdam no. EI2002-26*. Notes: Accepted for publication in: *Transportation Science*.