

Computer Aided Scheduling of Switching Engines

Marco E. Lübbecke* and Uwe T. Zimmermann

Department of Mathematical Optimization, Braunschweig University of
Technology, Pockelsstraße 14, D-38106 Braunschweig, Germany

Abstract Scheduling the switching engines of an industrial railroad is a formidable and responsible task, closely related to the well-studied pickup and delivery problem with time windows. Aiming at an efficient usage of resources the need arose for a computer aided scheduling tool as support for the human dispatcher. We sketch a set partitioning formulation of this problem to be solved via column generation. The pricing subproblem is hard in the theoretical sense but can be solved by means of a combination of heuristics and exact algorithms. A trade-off between mathematical rigor and practicability becomes apparent and is extensively discussed. Our computational experience with an academic prototype implementation is encouraging. We succeed in obtaining practically acceptable solutions for instances of more than forty customers and six vehicles.

1 Motivation

Market deregulations in the German railroad sector some years ago forced private railroads to offer a better transportation quality and to decrease charges. The efficient use—and desirably a reduction—of available resources became indispensable. The research project *Umlaufplanung im Güterverkehr bei Werks- und Industriebahnen*¹ reported upon here deals with improving the productivity of the central resource, *viz.* the switching engines.

In applied mathematics, in particular in combinatorial optimization, we currently witness a change of focus towards making the large body of available methodology *utilizable* in practice. Optimal decision support for complex and large-scale real-world applications comes in reach due to innovative numerical and algorithmic techniques accompanied by an enormous progress made in modern computer technology: Large amounts of data now available for the description of a practical problem at a realistic level of detail can reveal the limitations of an algorithm probably not detected in a laboratory environment. Moreover, advanced implementations are the reason for the success of techniques known for a long time but used only recently. The ease of testing may be the key feature offered by fast computers as well as the ready availability of the most elaborated algorithms as commercial software.

* Corresponding author. Email: M.Luebbecke@tu-bs.de

¹ Funded by the German Federal Ministry of Education and Research (BMBF), grant no. 03-ZI7BR2-1

Our main purpose here is to sketch the necessary steps for scientists and practitioners jointly attacking an industrial problem, *viz.* problem identification, model building, algorithm design, implementation, interpretation of the solution, and installation in practice.

2 Engine Scheduling

Operating a private railroad system as a subsidiary is indicated for large companies in the chemical, automobile, and steel industry in order to maintain a timely around-the-clock production process. Some run more than 100 *switching engines*, cf. Fig. 1, which differ in their technical and personal equipment, to handle the flow of up to 6000 freight cars between various terminals called customers. No train schedules exist, except for trains connecting to German Rail. That is, each transportation is performed upon *request* only, defined by origin, destination, service times and time windows, tonnage, and a list of eligible engines. Most attributes are optional. This definition is an abstract mathematical device, distinct from the one used in practice. But it is able to also incorporate all occurring kinds of special events like engineer's breaks, fueling, maintenance, and repair, to list only a few. It is the dispatcher's task to allocate engines to requests such that all requested transportation is served to the customer's satisfaction, i.e., compliant to the above constraints. The tractive power of an engine must never be exceeded. It goes without saying that our presentation is highly simplified. In railroad traffic, requests are to be scheduled according to certain simple configurations, cf. Fig. 2, in order to avoid too much shunting; the topography of the tracks, peculiarities of single customers, weather conditions, service agreements, priorities, and many more complete the picture. For a much more detailed discussion we refer to [9,10].

Current information and disposition systems merely map reality onto the screen, offering an efficient administration e.g., by means of a graphical display of the track layout. The actual disposition, however, is done manually; the decision about how to schedule engines is completely up to the dispatcher and his or her experience and motivation. One would expect that in a more or less regular industrial operation an experienced dispatcher will anticipate most of the customer's requests and schedule the engines accordingly. While in normal operation this is actually true, during peak workloads looking ahead is practically impossible. Requests are served on a first-come first-serve basis, or, more realistically, in a manner depictedly termed *loudest-shout first-serve*. Clearly, a human judgment of dependencies between different decisions of such complexity is only local and incomplete. The present project aims at providing a computer tool which suggests a schedule, thus supporting and relieving the dispatcher. What is more, the tool should be able to pursue operational goals, e.g., the minimization of unproductive work. The following result [10] attests that *in principle* there is no better way of solving this problem other than evaluating each possible schedule and select the best.

Lemma 1. *The above Engine Scheduling Problem is \mathcal{NP} -hard.*



Figure1. A switching engine transports molten iron at a steel works

3 Set Partitioning Formulation

An obvious mathematical formulation does not necessarily yield a *good model*. This simple statement conceals that considerable mathematical experience and intuition are required in order to identify the non-obvious. A *natural* mixed-integer formulation of the Engine Scheduling Problem (ESP), following the lines of [5,12], with explicit decisions on engine assignments, request sequences, service times, and tonnage requirements, respectively, turned out to be highly *symmetric* [7]. That is, in a sense, each decision carries too few information; similar solutions are reproducible in many different ways. Such circumstance embodies the danger that a reasonably directed search for an optimum be fruitless. Bearing this knowledge in mind, we follow an alternative approach.

A complete schedule for a given planning horizon splits into proposals, one for each engine. Assume for the moment that we are able to derive from a particular subset of requests and a specific engine the *best possible* way

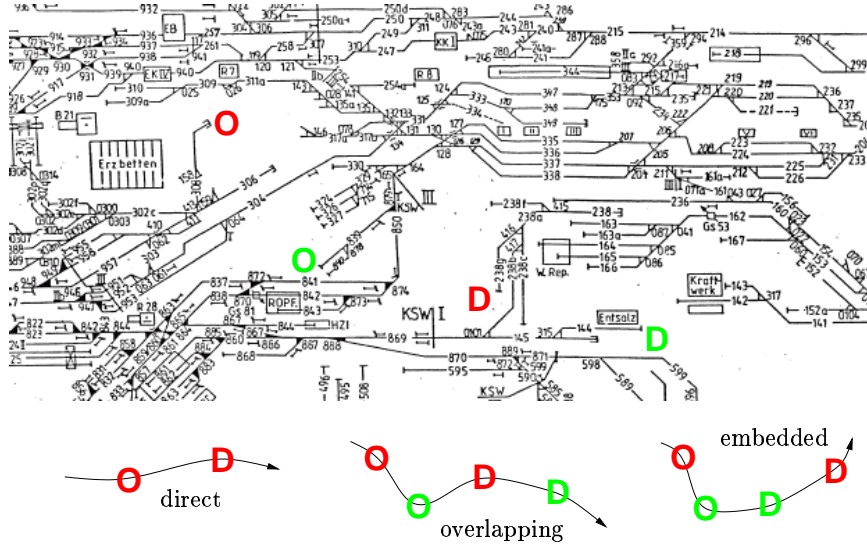


Figure 2. Detail of the track layout of an industrial in-plant railroad. Feasible tours are concatenations of simple configurations of the sketched three types. *Direct* transportation from the origin (O) to the destination (D) is the default in manual planning.

to serve the whole subset. Further assume that we only consider subsets which are feasible for the respective engine. Then the ESP is to select subsets of requests, at most one for each engine, such that the disjoint union of selected subsets contains each request precisely once, and the operational goal is minimized by the selection. To the adept this reads as follows.

$$\begin{aligned}
 & \min \sum_{e \in \mathcal{E}} \sum_{R \in \Omega_e} c_R^e x_R^e \\
 & \text{subject to } \sum_{e \in \mathcal{E}} \sum_{R \in \Omega_e} \delta_{rR} x_R^e = 1, \quad r \in \mathcal{R} \\
 & \quad \sum_{R \in \Omega_e} x_R^e \leq 1, \quad e \in \mathcal{E} \\
 & \quad x_R^e \in \{0, 1\}, \quad e \in \mathcal{E}, R \in \Omega_e
 \end{aligned} \tag{SP}$$

- | | | | |
|---------------|-----------------|---------------|--|
| \mathcal{R} | set of requests | Ω_e | request subsets feasible for engine e |
| \mathcal{E} | set of engines | c_R^e | minimal cost of serving $R \in \Omega_e$ on engine e |
| | | δ_{rR} | one if $r \in R$, zero otherwise |
| | | x_R^e | indicates whether e serves subset R |

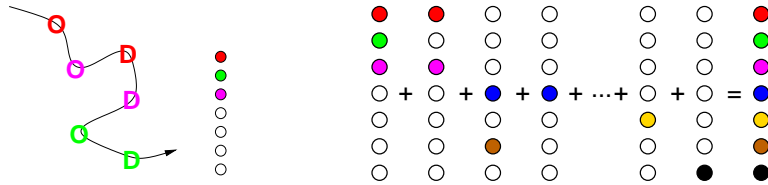


Figure 3. A tour, its incidence vector δ , and the set partitioning formulation

This seemingly compact, so called set partitioning formulation has been successfully applied to the generic problem, known as *Multiple-Vehicle Pickup and Delivery Problem with Time Windows* [5,12], but has some serious detriments. A minor difficulty is that determining the best possible cost of a particular subset itself is an \mathcal{NP} -hard combinatorial optimization problem since it involves the optimal solution of the ESP for a single engine. Secondly, although the model has only a few constraints, the number of variables—which is the number of *all* feasible subsets of requests—is so large that storing the model in core memory, let alone solving it, is definitively out of reach. What more directly meets the eye is the fact that all information about time windows, engine loads, and tour structures is implicitly hidden behind some obscure construction of *feasible subsets* of requests. It seems that we only complicated matters, but fortunately some other advantages are ahead.

4 Decomposition and Column Generation

Relaxing the integrality constraint to $0 \leq x_R^e \leq 1$ for all $e \in \mathcal{E}$, $R \in \Omega_e$, the set partitioning formulation becomes a linear program (LP) of prohibitive size. However, from the theory of the *simplex method* we know that in each iteration we only need access to as many variables as there are constraints present in the formulation, i.e., very few compared to the size of our model. It is thus an appealing idea to start with a very small portion of the model, and dynamically extend it in a smart way until it can be proved that all *relevant* information was considered, i.e., optimality is reached. The method is a two-level strategy; hence it is known as decomposition principle. A superior level determines what model extensions are locally most valuable, while the extension itself is calculated at an inferior level.

The theoretical principle underlying this idea can be best explained adopting an economic viewpoint, see e.g., [8]. Different branches of a large company independently take decisions on how much to use shared scarce resources in order to maximize the company’s overall profit. The branches inform a central coordinator about their locally optimal activities. The coordinator with a global view on the resource usage, but without direct influence on the subordinate decisions, fixes prices for each resource and returns them to the

branches. These revise their decisions according to the resource prices, again without knowledge about the other decisions. The iterative process stops as soon as the global prices are such adjusted that no branch can offer a revision which contributes to an overall improvement of resource usage.

A translation to the linear programming framework goes as follows. An arbitrary feasible solution to (SP) induces initial sets $\Omega'_e \subseteq \Omega_e, e \in \mathcal{E}$. Already determining such a feasible solution is \mathcal{NP} -complete. At the beginning, only the variables corresponding to the elements in $\Omega'_e, e \in \mathcal{E}$, are incorporated in the linear program, the so called *restricted master program* (RMP)—or the coordinator in the above wording. This latter is optimally solved based on the given tiny detail of the total model information, yielding an optimal dual solution (u^*, v^*) , partitioned according to the constraints of (LP). These prices—fixed by the coordinator—enable us to decide whether to enlarge the RMP by a certain, still unconsidered variable $x_{\bar{R}}^e$ or not by looking at the respective *reduced cost coefficient*

$$c_{\bar{R}}^e - \sum_{r \in \bar{R}} u_r^* - v_e^* . \quad (1)$$

In the case that this entity is negative, adding the feasible subset \bar{R} to Ω'_e is promising from a global point of view. This gives new dual prices and we iterate. Otherwise we know that the optimal primal solution to the RMP solves the huge (LP) as well. The crucial task in this scheme is that of the company's branches which have to propose favorable feasible subsets *without* explicitly considering them all. The strength of the described decomposition approach is that this is possible by solving an optimization subproblem, the so called *pricing problem*. Since variables and their associated matrix coefficients are successively adjoined to the RMP one also refers to column generation.

The subproblem is to construct a feasible subset of requests with (minimal) negative associated reduced cost (1), or to prove that none such exist. Omitting the details, one derives from the above that this amounts to finding a (shortest) feasible tour for an engine, respecting all the constraints mentioned in Sect. 2, with the additional cost or gain, respectively, of the dual price u_r incurred for each visited request $r \in \mathcal{R}$. By now we see where all the information vanished at the end of the last section. Sadly, the elegance of this algorithm is slightly afflicted by the following result [10].

Theorem 2. *The pricing problem is \mathcal{NP} -complete.*

Solution of the Pricing Problem

An explicit evaluation of (1) for all feasible tours and all engines would be justified by theory, cf. Theorem 2, but is out of the question from a practical point of view. Instead, an intelligent implicit enumeration is performed, in the hope—but without guarantee—that many solutions need not be considered.

The general strategy is, starting from empty tours, to successively append yet unvisited locations to partial tours, always preserving feasibility with respect to tonnage and time window constraints [3,4,5]. In our algorithm, we combine this technique with the fact that feasible tours are severely restricted in their combinatorial structure, cf. Fig. 2. Indeed, we proceed by iteratively concatenating entire request configurations as depicted in Fig. 4. This gives a quickly growing tree like structure. It happens in the course of this *dynamic programming* algorithm that two tours are constructed that visit the same set of requests and end in the same location. Then the more expensive one is disregarded, pruning the respective branch of the tree. Such dominance criteria help to reduce the number of solutions to be considered and are already stated in [4]. However, for the use of such a criterion it is necessary to have the tours to be compared already constructed, and it requires a good overview over the search tree. Rather, it would be desirable to anticipate unpromising tours *before* their construction.

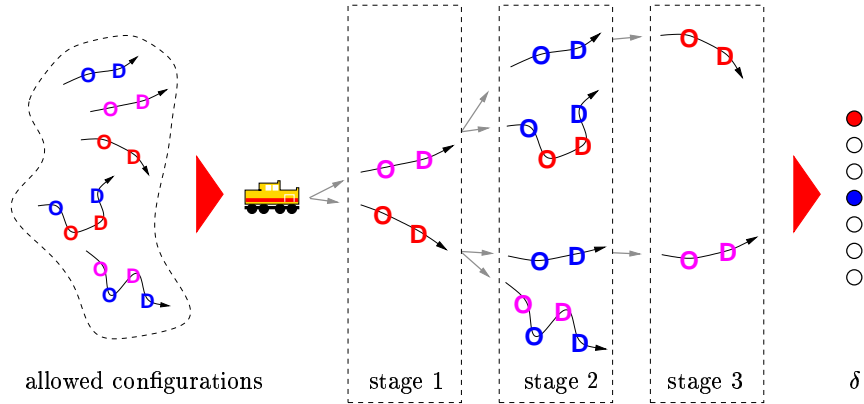


Figure 4. Iteratively extending tours by allowed configurations in a tree like manner. The best tour induces a column δ to be adjoined to the RMP.

To this end, we exploit the following observation. There is no use further considering a particular tour when we have evidence that the best possible extension of this tour cannot have cost smaller than the cost of the cheapest tour constructed so far. In view of (1), the most advantageous extension of a tour T is to visit exactly those yet unvisited requests $r \in \mathcal{R}$ with u_r positive. Whenever the so calculated *lower bound* LB is larger than the cost C of the currently best known tour, i.e.,

$$LB = \text{cost}(T) - \sum_{r \in \mathcal{R} \setminus R: u_r > 0} u_r \geq \min\{0, C\} \quad (2)$$

we disregard T —and need not generate any of its extensions in the tree. Since the pricing problem must return a column with negative reduced cost coefficient (1) we also check whether LB is non-negative. The use of this technique led to a considerable speedup of the pricing algorithm. In [9,10] we discuss possible refinements of LB . For an exhaustive presentation of the application of such bounds in implicit enumeration strategies we refer to [6].

Integer Solutions

It must be stated clearly that all our efforts so far only lead to an optimal solution of the linear relaxation of (SP) which may be fractional. Even worse, there is no guarantee that the final RMP is integer soluble at all; columns may be missing. That is, branch-and-bound, an implicit enumeration scheme to solve integer programs, must be enabled to generate additional columns if required. The resulting algorithm is known as branch-and-price [1].

The general proceeding is to iteratively fix the binary variables to zero or one in a binary tree manner. Again, evaluating each solution is impracticable, hence one strives to keep the search tree as small as possible. This is accomplished by virtue of lower bounds, similar in principle to our use in the dynamic programming algorithm for the pricing problem. Having fixed a set of variables, a lower bound on the integral objective function value is derived by solving the linear programming relaxation of the resulting (mixed) integer program. It goes without saying that the quality of the bound is of vital importance for the effectiveness of the procedure. Additionally, different model formulations may differ in the quality of their linear relaxations. This supplies a criterion for evaluating a formulation, and (SP) performs very well in this respect.

Our computational experiments indicate, under conditions to be explained in the next section, that the generated (“root node”) restricted master programs happen to be already integer feasible. Here again, we come across the phenomenon that reality is not as bad as suggested by computational complexity, or in other words, in practice the theoretically worst case rarely occurs. This was observed in many other practical settings as well where artificial data were compared to “real world” data. Although being no mathematically satisfactory explanation this is an encouragement to attack practical problems even in the presence of negative complexity results.

5 Implementation Issues

With a mature mathematical background the column generation principle is relatively easy to comprehend. However, when it comes to an implementation the generic textbook presentation, cf. Algorithm 1, is of little help. In our experience, every ingredient offers much degree of freedom, the impact of which

Algorithm 1 Generic column generation scheme to solve a linear program

```
Start with initial basis
while columns with negative reduced cost exist do
  Adjoin column to restricted master program
  Re-optimize
end while
```

on the overall performance cannot be overestimated, c.f. Fig. 5. Questions arise as how to provide an initial solution, heuristically or artificially (“first phase simplex”)? Solve the pricing problem exactly or approximately? What pricing rule to use? How many columns are adjoined at each iteration and how to make use of columns generated in earlier iterations (“column pool”)? What about the growing size of the RMP? How often to re-optimize and how to solve the RMP, and when to terminate the algorithm? This list is by far not complete and omits all problem specific issues.

Our bottleneck procedure is the pricing subproblem, which, as a general rule, should be solved exactly only when heuristics fail. We make extensive use of a column pool and add multiple columns per iteration. To our experience, this also increases the chance of obtaining integer solutions. There is no doubt that without the concerted interaction of heuristics and exact methods a solution even of smaller instances would not have been possible. The monographs [8,11] and the recent synopsis [2] on various acceleration techniques are valuable sources of information.

Computational Experience

As indicated earlier, our concept of request is more general than the one used in practice. Hence, all practical data had to be manually post-processed, and are only scarcely available. Thus we will only report on trends, but these are encouraging. The size of the instances we are able to solve within acceptable time, i.e., in a few seconds in order to guarantee interactivity, corresponds to a planning horizon of more than two hours. According to officers of in-plant railroads involved in our project it is seldom sensible to plan further ahead because of the increasing uncertainty of future requests. Still, if there is need for solving larger instances a decomposition of the planning horizon into overlapping, manageable time slices would be reasonable.

The objective function value cumulates the minutes during which engines wait or run without load. In our case the optimum usually ranges around half an hour. In view of practical relevance our optimality gap of less than ten minutes is definitively tolerable. On a today’s personal computer it currently takes a few minutes to obtain an integer solution for more than forty requests to be served by six engines. Depending on parameter settings, rarely more than 20,000 columns are generated, in less than 200 iterations. The ratio customers/vehicles of seven is notably large for our problem class.

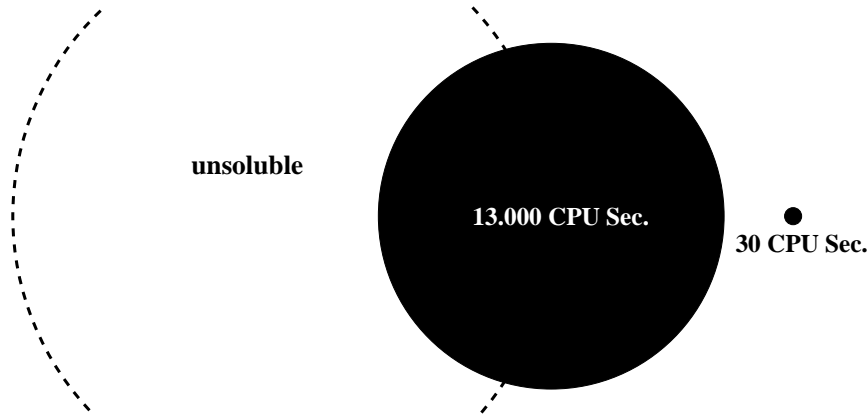


Figure 5. Impact of implementation details on the computation time for a particular instance. The generic algorithm even with usage of a column pool and a simple pricing heuristic terminated with a memory allocation failure. When the exact pricing algorithm is completed only when the pricing heuristics fail, at least the instance gets soluble. Using the lower bound (2) and a heuristic network reduction finally drastically speeds up the solution process.

6 Towards Computer Aided Scheduling

When we launched this project, we identified the necessity of three areas of contribution, for each of which different partners with their respective competence were involved. Railroad companies furnish the practical background and realistic data, mathematicians supply models, algorithms, and validation of the produced solutions as well as an academic prototype software. People from the IT business are responsible for the marketing process, including the production of the final tool. Here is some experience we have drawn from many discussions with these partners and we found useful for similar projects.

The support and feedback of a practitioner at any stage helps to identify and to incorporate additional or modified operational constraints not thought of in the initial problem formulation. As a starting point, these constraints are not forgotten but left out intentionally in order to quickly provide a first prototype implementation. The purpose of such a model that represents only a simple or simplified scenario is to supply the practitioner with an estimation of the capabilities and potency of a more elaborated model. People are usually not used to see a computer performing a task previously done by a human expert. Hence they are sceptical. We remark, however, that the planning process can be covered only partly by a model anyway. A reasonable goal is to depute routine tasks to the computer while the dispatcher concentrates on non standard work. This relieves the dispatcher especially during peak workloads.

It is by no means sufficient (but necessary!) to convince the management of the usefulness of an optimization based decision tool. The managerial decision maker is in general not identical to the person who will work with the tool in every day operation. It is therefore even more important to study and carefully evaluate the experience and demands of the end user.

A computer still is sometimes seen as an opponent. Even when an existing computer system is to be enhanced by optimization techniques in form of a planning or decision suggestion the fear of being replaced by the software is often seizeable. Objection against the new technology may be a consequence, especially when the underlying methodology is not common knowledge even for experts in the planning environment. The use of mathematical programming is in this sense a black box method and solutions are not immediately recognizable by the personnel. Note, that we do not refer to the mathematical solution, but to the solution in terms of the practical setting formulated in the language of the practitioner. The structure of such a solution may differ considerably from a manually generated one the planner is used to see. When it does not feel right, it is not right. Thus, it is helpful to accompany the introduction of the system not only by explaining how it works but also by showing *that* it works and what practical ideas motivate the differences in the appearance of the solution. A graphical simulation clarifying the system's suggestions may be of great help.

One point to particularly focus on is the quality of a solution. In mathematical terms, this notion is easily explained via bounds on the objective function value. In practice however, questions of stability and reliability arise. The former corresponds to the desired property that small changes in the input data shall result in small changes in the output, especially when the input changes frequently. Reliability means that the system is failsafe and provides a solution in any circumstance. A good strategy is to first compute a heuristic solution. Time permitting, an improvement of this solution based on new data is performed without giving notice and delivered only when needed.

Last, but not least, the notion of a computer suggestion should be taken literally. The proposal by the system serves as a default only, always subject to the final decision of the human dispatcher. Partial solutions fixed manually must never be overruled by the computer.

7 Discussion and Outlook

We have seen that the trade-off between mathematical rigor and practical needs is often decided in favor of the latter. One may have the impression that the benefits gained from the use of mathematical methods in industry are then compensated. Indeed, shortly after the introduction of a computer aided scheduling system, productivity may decrease significantly and the overall effect may even be negative. Of course, this is only one part of the truth. Once a planner judges the tool useful and reasonable, the default suggestion of the

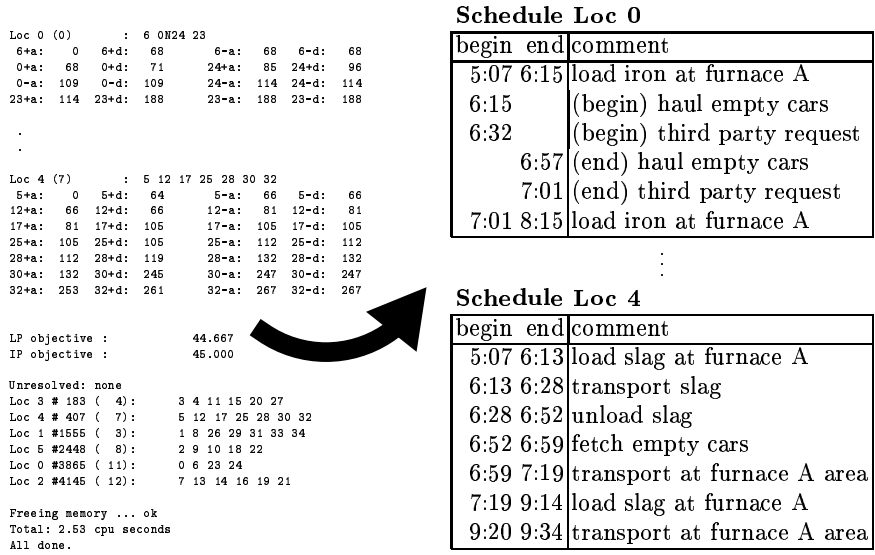


Figure 6. Similar output by our implementation leads to a readable schedule

system will be accepted in most cases, not least for the reason of convenience. Building on this basis further components can be added, making the introduction of new technologies a dynamic process. We believe that in the long run remarkable savings offered by mathematical optimization approaches are definitely worth enduring possibly occurring intermediate embarrassments.

Industrial in-plant railroads usually operate a surplus of engines in order to handle peaks in the demand for transportation. However, it is believed that a more regular and steady-going operation, also planning a little ahead, could help cutting back this overhead considerably. We have good reason to hope that in the end precisely this will happen due to the incorporation of the presented or revised algorithms in a dispatching software.

A shortcoming of our model is the assumption of independent engines, which needs not hold in general. In fact, individual transportation requests may be precedence related, a matter one can take care of if the respective requests are to be served by the same engine. If they are possibly incorporated in different tours, the presented set partitioning formulation is not suited to control feasibility of precedence constraints. Since this is a problem relevant to many practical vehicle routing settings lively research efforts are to be expected in this area.

Of course, we have seen a detail of the operational planning tasks of an industrial railroad only. When customers request for transportation capacity or a specified material, the flow of freight cars has to be managed. Only a small portion of the cars is owned by the railroad company; the majority is rent from

other railroads. Cars are not identical, and for a customer request there is a type that fits best. It is roughly known which cars will be needed in particular working shifts, but manually it is hardly possible to anticipate the exact stock. Lacking cars have to be ordered timely. Since paying the cars is on an hourly basis the dispatcher tries to return them as soon as possible. Emptied cars from one terminal may be assigned to requests of another terminal when appropriate. The goal is to minimize both the paid rent and the light running of cars while guaranteeing service without type mismatches. This problem of a distinct combinatorial nature also offers significant financial savings, and will be dealt with in an industrial project we are about to kick off.

Acknowledgments We are grateful to the in-plant railroads *Eisenbahn und Häfen GmbH*, *EKO Trans GmbH*, and *Verkehrsbetriebe Peine-Salzgitter GmbH* for many discussions and critical remarks, and for providing us with real world data.

References

1. C. Barnhart, E.L. Johnson, G.L. Nemhauser, M.W.P. Savelsbergh, and P.H. Vance. Branch-and-price: Column generation for solving huge integer programs. *Oper. Res.*, 46(3):316–329, 1998.
2. G. Desautniers, J. Desrosiers, and M.M. Solomon. Accelerating strategies in column generation methods for vehicle routing and crew scheduling problems. Les Cahiers du GERAD G-99-36, École des Hautes Études Commerciales, Montréal, Canada, August 1999.
3. J. Desrosiers and Y. Dumas. The shortest path problem for the construction of vehicle routes with pick-up, delivery and time constraints. In *Advances in Optimization and Control*, volume 302 of *Lecture Notes in Economics and Mathematical Systems*, pages 144–157, 1988.
4. J. Desrosiers, Y. Dumas, and F. Soumis. A dynamic programming solution of the large-scale single-vehicle dial-a-ride problem with time windows. *American Journal of Mathematical and Management Sciences*, 6:301–326, 1986.
5. Y. Dumas, J. Desrosiers, and F. Soumis. The pickup and delivery problem with time windows. *European J. Oper. Res.*, 54:7–22, 1991.
6. T. Ibaraki. *Enumerative Approaches to Combinatorial Optimization*, volume 10 and 11 of *Annals of Operations Research*. Baltzer, 1987.
7. E.L. Johnson. Modelling and strong linear programs for mixed integer programming. In S.W. Wallace, editor, *Algorithms and Model Formulations in Mathematical Programming*, pages 1–43, Berlin, 1989. Springer-Verlag.
8. L.S. Lasdon. *Optimization Theory for Large Systems*. Macmillan, London, 1970.
9. M.E. Lübbecke. *Optimal Engine Scheduling by Column Generation*. PhD thesis, Dept. Mathematical Optimization, Braunschweig University of Technology, 2000. In preparation.
10. M.E. Lübbecke and U.T. Zimmermann. Optimal engine routing and scheduling at industrial in-plant railroads. Technical report, Dept. Mathematical Optimization, Braunschweig University of Technology, 2000. In preparation.

11. J.L. Nazareth. *Computer Solution of Linear Programs*. Oxford University Press, Oxford, 1987.
12. M. Sol. *Column Generation Techniques for Pickup and Delivery Problems*. PhD thesis, Eindhoven University of Technology, 1994.