Algorithm Engineering and Industrial Applications

Article *in* it - Information Technology · December 2011

DOI: 10.1524/itit.2011.0657 · Source: DBLP

1 author:

Rolf H. Möhring
Technische Universität Berlin
193 PUBLICATIONS **8,317** CITATIONS

Algorithm Engineering and Industrial Applications

Algorithm Engineering und Industrielle Anwendungen

Rolf H. Möhring, TU Berlin

Summary Algorithm engineering consists of the design, analysis, implementation and experimental evaluation of practicable algorithms. Research in this direction can be done both in a lab environment and in a real world application. Experience shows that there are significant differences in methodology between these two settings. This paper will illustrate them on two selected applications and highlight some of the main differences from the viewpoint of algorithm engineering.

►►► **Zusammenfassung** Das Algorithm Engineering behandelt Entwurf, Analyse, Implementierung und experimentelle Bewertung von praktikablen Algorithmen. Dies kann sowohl in einer Laborumgebung wie in einer industriellen Anwendung geschehen. Die Erfahrung zeigt jedoch, dass zwischen diesen beiden Szenarien große Unterschiede im methodischen Vorgehen bestehen. Dieser Artikel illustriert diese an zwei ausgewählten Anwendungen und arbeitet einige wesentliche Unterschiede aus der Sicht des Algorithm Engineering heraus.

Keywords G.2.3 [Mathematics of Computing: Discrete Mathematics: Applications]; F.2.2 [Theory of Computation: Analysis of Algorithms and Problem Complexity: Nonnumerical Algorithms and Problems], routing and layout, sequencing and scheduling, industrial applications, automated guided vehicles, coil coating, logistics in steel manufacturing, transport logistics ►►►
Schlagwörter Industrielle Anwendungen, fahrerlose Transportfahrzeuge, Oberflächenbeschichtung, Logistik in Stahlwerken, Transportlogistik

1 Features of Industrial Projects

Algorithm engineering as a scientific discipline entails the design, analysis, implementation and experimental evaluation of practicable algorithms. This can be done in a lab environment under controlled experimental conditions as in a physics lab, but there is the natural challenge to prove that the results are successful in the real world. Industrial applications provide a great opportunity to demonstrate this, and the chances to do so are much greater if the whole algorithm engineering cycle is done in close cooperation with an industrial partner on a real world project.

However, be aware that you then leave the secure and controlled ivory tower that the lab provides! You will enter a world with different motives, different goals, a different language and different time scales. Still, the direct interaction with consumers of your algorithms may be very rewarding, and not just because you will get additional funding. No, they will pose new challenges in the form of unthought of algorithmic problems that you need to solve, they will question your initial models and are ready to change it anytime when they think that a feature not yet considered is valuable to them. It is like a game between an algorithm engineer who knows the tricks of the trade and an opponent who has a quite different utility function and changes goals and/or the model during the game.

This game may lead to great and unexpected scientific results if you play it well. I will illustrate this on two examples below, a project on improving the routing of automated guided vehicles in a container terminal and a project on improving the coating of coils in a steel

company. Before going into the details of these projects, I will summarize some "rules of the game" that I have learned from more than twenty years of experience with industrial partners, and which I regard as typical features of industrial projects in traffic, logistics and production.

1.1 Designing the Cooperation

Never loose sight of your own goals. The industrial partner usually wants a software tool to save cost in a production process, or an algorithmic solution of a complex logistic problem, whereas you as algorithm engineer in academia want publications, new algorithmic challenges that match your expertise, and real-world data. So think about your motivation for doing the project (it's not the money) and weigh the time and resources that you and your team will spend against the desired outcome.

Make sure that you understand the task. The problem is usually presented in an unfamiliar and confusing terminology with an overwhelming wealth of details, an absence of mathematical precision and no clear objective.

Figure 1 Data delivery. This is not a joke!.

Keep asking questions: What are the key decisions to be made? What is it that makes the task hard? What is the benefit expected from a practicable algorithm? Make sure you understand exactly and get all relevant details.

Understand your partner's (different) goals. Why this project and why with you, who are the stakeholders? Argue for the importance of your own goals. Discuss software design issues including the type of implementation (prototyping or integration, language, operating system) and the supply of real data.

Agree on timing and avoid pressure. Clarify what you do and do not deliver and state clearly when you need what from your partner. All timing depends on it. Define milestones at which you will present your work and which may also serve as decision point for both partners on the next steps and as a reevaluation of the cooperation. Should the project be stopped because expectations are too unrealistic? Does the approach taken work? Do the next steps require more resources and time?

Finally, name a realistic price and renegotiate at milestones if necessary. After all, the project is research and outcomes are uncertain.

1.2 Exchanging Data

Emphasize the importance of data from the start and quantify the need for test instances. Agree on a data format and insist on data delivery in that format. Include it in the contract. Remember, all timing depends on data delivery. Do not accept chaotic data as in Fig. 1, you will regret it! Plan on multiple iterations until your partner gets the data right.

Discussing solutions is the final model verification. Show (even preliminary) solutions to your partner early and communicate them in a good form. If possible, use visualization as in Fig. 2 or a simulation model as in Fig. 9. A good representation of solutions is the key for acceptance of your results and a very helpful means for model

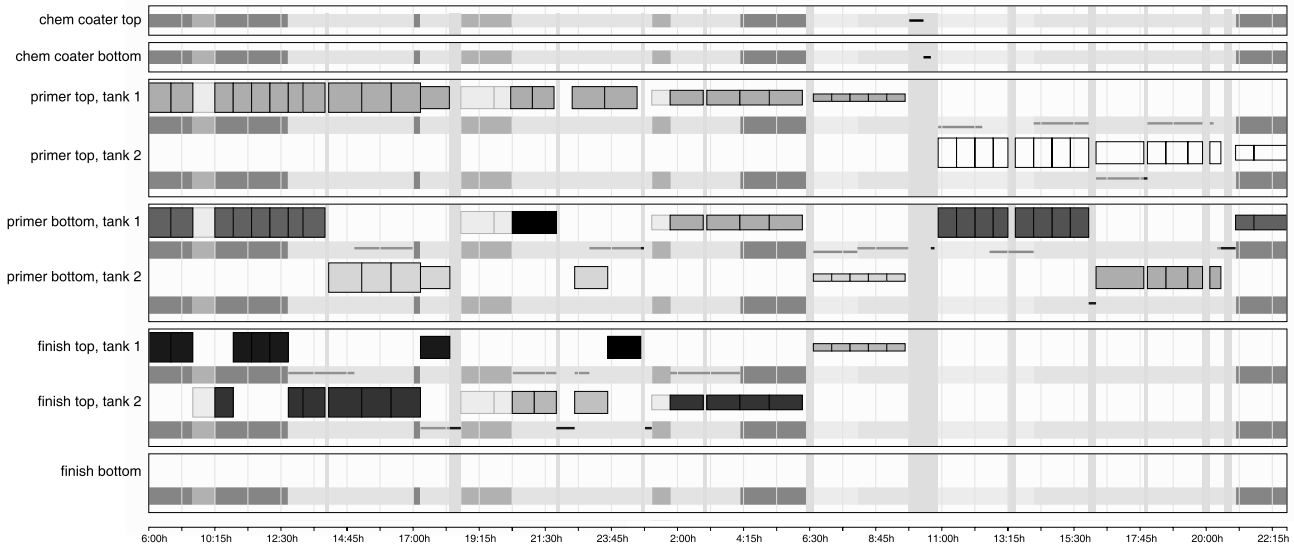

Figure 2 Representation of a solution of the coil coating algorithm (see Sect. 3).

verification. The nicest model or algorithm is completely useless when not accepted. Therefore, take concerns seriously and build confidence in the "black box" that your approach usually is to your partner. Explain your solutions and why they "look different" from what planners expect to see.

1.3 Communication

Learn the lingo. Learn correctly and use application terminology. Use mathematical terminology sparingly and consciously. Say "task", not "problem".

Use good means of communication, from personal meetings (most effective but time-consuming) to email (precise questions and written answers, but slow and prone to misunderstandings) and phone calls (fast and direct, but without documentation for later reference). Phone calls with email summary are a good combination.

Be professional. Remember names, titles and roles. Prepare presentations, make plans, set goals, deliver on time. Propose next steps, next meetings, and stay in touch.

Respect cultural differences. Be aware of skepticism towards mathematics and algorithmics. Aim at a goal-oriented communication and try to not discuss math or proofs with partners. Keep roles and (different) goals in mind and help everybody to achieve theirs.

1.4 Summary

Industry cooperations can be both fun and rewarding, but you must make sure that you know your own goals, that you do understand the task and plan it well, and that you leave no questions open. Data is of key importance but hard to obtain. Acting strategically and communicating professionally helps.

2 Routing Automated Guided Vehicles in a Container Terminal

Automated Guided Vehicles (AGVs) are state-of-the-art technology for optimizing large scale production systems and are used in a wide range of application areas. A standard task in this context is to find efficient routing schemes, i. e., algorithms that route these vehicles through the particular environment. The productivity of the AGVs is highly dependent on the used routing scheme.

In an application with the Hamburger Hafen und Logistik AG (HHLA) we studied how to compute good routes and improve the container throughput for the Container Terminal Altenwerder (CTA) [1]. Figure 3 displays the layout of this terminal. About 70 AGVs transport containers between the quay and the storage in the routing area shown at the top of Fig. 3. Each AGV is connected by radio to a host computer (the router). Through a system of transponders that are embedded in the floor of the routing area, the AGVs know their position, report them back repeatedly to the router and get back instructions for their driving behavior. AGVs are symmetric, i. e., they can travel in both of the two driving directions equally well and can also change directions during a route. They do not have any sensors and depend completely on the driving instructions from the router, which they get in several segments along their route.

2.1 Designing the Cooperation

We first made ourselves familiar with the implemented routing algorithm and why we were considered for that project.

The routing algorithm was static with a collision avoidance mechanism at runtime. A virtual routing graph was laid out on the routing area (see Fig. 4) and used to compute routes one by one for the online arriving routing

Figure 3 Layout of Container Terminal Altenwerder. © HHLA.

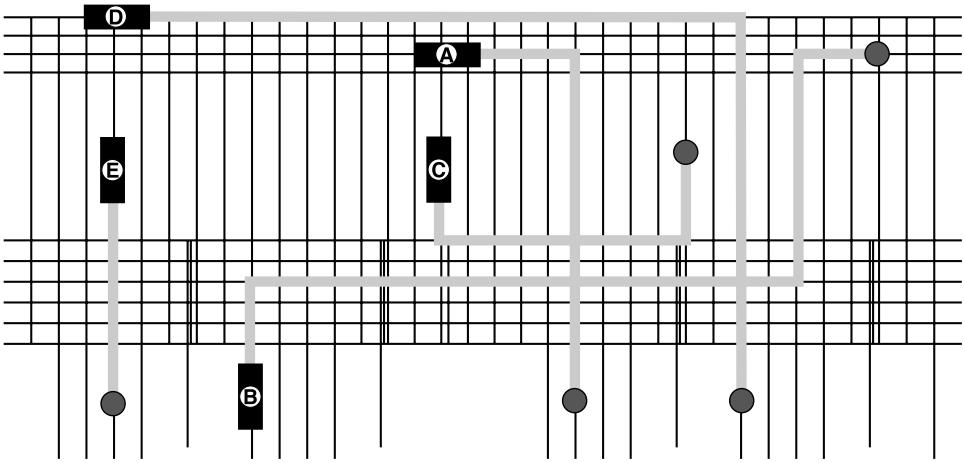

Figure 4 Routing graph with precomputed routes.

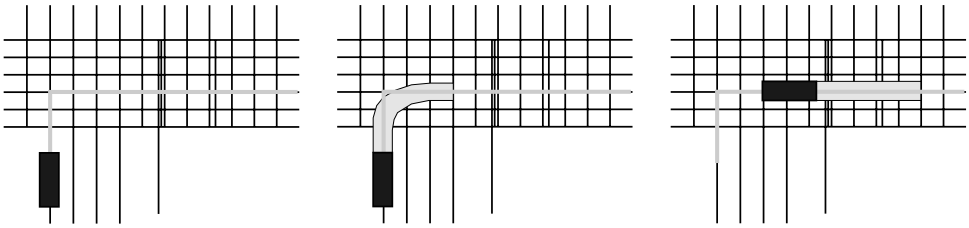

Figure 5 Claims (light gray polygons) along a route.

requests with a shortest path algorithm, in which edges already used for earlier routes were made more expensive to discourage the use of the same edge in many routes.

The computed routes are, of course, not collision-free. Hence, one needs an additional conflict avoidance system that prevents collisions at execution time of the routes. This was done by iteratively allocating to an AGV the next segment of its route (the "claim") and blocking it for all other AGVs, see Fig. 5.

The advantage of the static approach is clear. It is easy to implement and allows very fast route computation. However, the collision avoidance mechanism is quite complex and leads to an unpredictable routing behavior and even occasional deadlocks, in which various AGVs block each other in continuing their planned route. These and other side effects had an unwanted deteriorating effect on the system performance.

HHLA considered us because of our experience with traffic planning and flows over time [3], and so a cooperation was set up with the aim to improve the throughput of container transports through dynamic routing methods.

2.2 Initial Model and Ideas

We started with a simple model in which we took the given virtual routing graph with constant travel times on the edges and routing requests that arrived online. Our initial algorithm used the paradigm of time-expansion. It routed requests sequentially and blocked previously computed routes for the currently considered AGV in the time-expanded graph, as depicted in Fig. 6. Collision-free routes then correspond to disjoint paths in the time-expanded graph.

Instead of an explicit time-expanded graph, which would computationally be much too inefficient, we modeled the time expansion implicitly through blocked intervals on the edges of the routing graph. These "blockings" keep track of the times at which an edge is used by previously routed AGVs. Computing the next route then amounts to computing a quickest path in a graph with time-depending blocking on the edges.

Although many routing problems with time windows are known to be NP-hard, we were lucky in our choice of model. It turned out that computing a quickest path is polynomial if we permit waiting along a route. One then minimizes elapsed time, i. e., the sum of the (constant) travel time on the edges and the time spent waiting on edges. The algorithm is a generalization of Dijkstra's algorithm and can also incorporate changes of orientations along a route. Figure 7 illustrates the idea.

2.3 Towards a Practicable Routing Algorithm

The first milestone was passed successfully. The proof of concept provided by our first model and algorithm led to a continuation of the project. The next challenge was to make the routing algorithm practicable, and this is where the engineering part of the project started to dominate. We had to cope with three aspects. First, we had to consider the physical dimensions of the AGVs, which do not at all behave like points in a graph. Second we had to

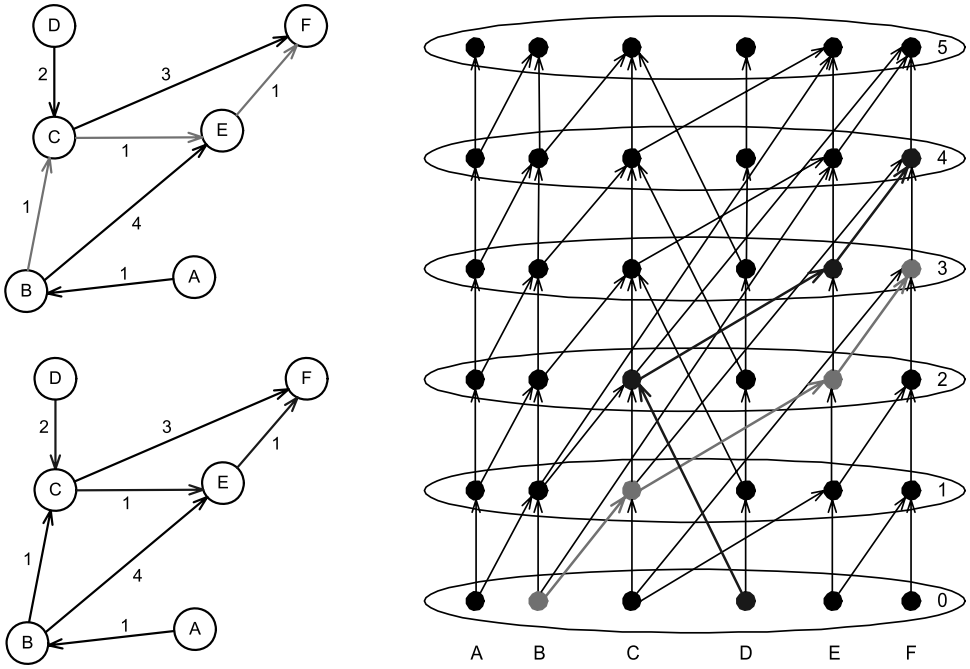

Figure 6 Two static routes B-C-E-F (light gray, top left) and B-C-E-F (dark gray, bottom left) with common edge (C,E) that are collision-free in the time-expanded graph (right).

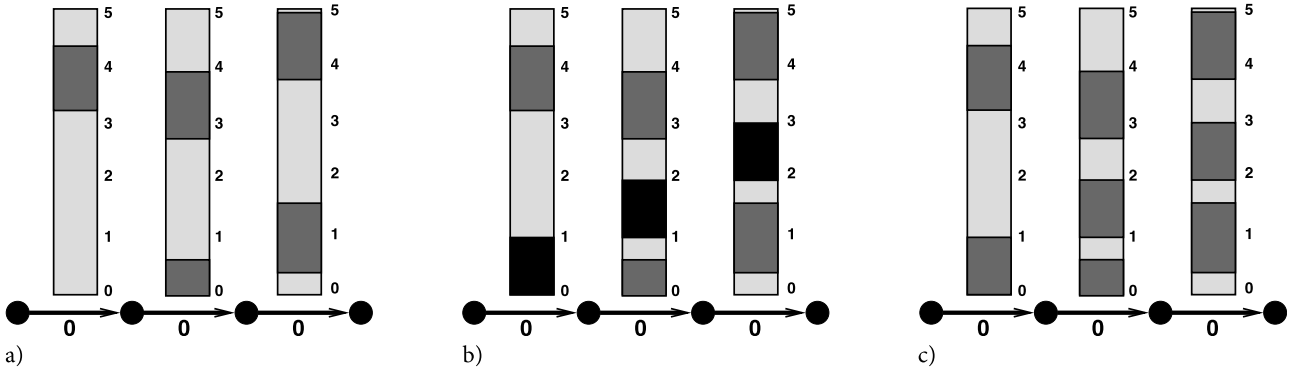

Figure 7 Illustration of the real-time computation on three consecutive arcs with transit time 1. (**a**) shows the situation before the new request arrives. Edges have blockings (dark gray) and free intervals (light gray) on the time axis (*y* axis). The task is to compute a quickest path that respects the time-windows. This is illustrated in (**b**) by black intervals. An accepted path is blocked afterwards as in (**c**).

integrate the driving dynamics, i. e., acceleration, deceleration and very slow movement for exact positioning under automated cranes. And third, the resulting algorithm had to be at least as quick as the static router.

For the dynamics, we agreed on a piecewise linear approximation and a policy to speed up to maximum speed whenever possible, which was the case in practice anyway. The physical dimension played a role when an AGV turns, which leads to blockings of graph edges near the AGV's current position, see Fig. 8. This was modeled by adding "turn" arcs into the routing graph at appropriate positions and by maintaining additional blockings when an AGV is on a turn arc, see again Fig. 8.

All this took several discussions, also with the manufacturer of the AGVs in order to agree on a realistic dynamic model and to get the correct dimensions of the turning polygon. The size of the routing graph was thus increased to roughly 5,500 vertices and 43,000 edges. Still, carefully engineering the routing algorithm made it fast enough to compute a quickest route and update the blocking in less than 50 milliseconds on average. This was fast enough for the second milestone, and we entered the final phase of the project.

2.4 Coping with Disturbances

To use a router in daily operation, it must be robust under small disturbances. This was a principle problem for our routing paradigm. It was based on disjoint paths in an (implicit) time expanded graph. However, the time expansion changes when AGVs do not drive as specified by the router. Starting the engine could take more time than expected, a slightly ill-positioned container would automatically trigger a lower driving speed, and AGVs might halt unexpectedly because of a technical problem.

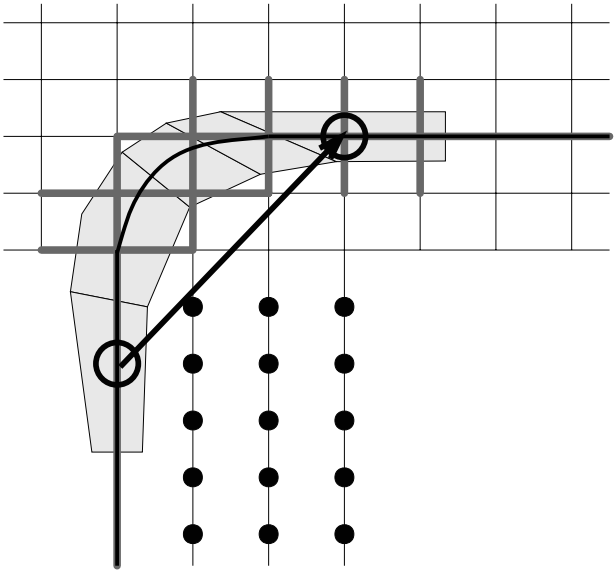

Figure 8 A turn arc (black diagonal arc) that causes blockings (light gray) on arcs in the routing graph due to polygon that circumscribes the area used by an AGV when it turns. The black points denote start points of turn arcs.

Also, the radio communication with the router might suffer from small delays and instructions on the next part of their route might not get to some AGVs in time.

The final phase of the project was therefore devoted to rerouting techniques. The router was fast enough so that it could react to unforeseen events in real time and compute changes to routes or even new routes. However, to implement this without creating deadlocks again required some care.

To this end, routes were now considered as "timed" routes. They were divided into segments as in the static router, but now with temporal information computed by the router (arrival time, speed, etc.). AGVs had to come to a stop at the end of their current segment unless they get a prolongation of their route in time. AGVs would report their status to the router so that it could detect deviations from their "timed" behavior within their current segment. The rerouting algorithm would then identify the affected timed routes of other AGVs and use several graph structures (dependency graph, route interaction graph) to determine a good order in which to change these routes. In the worst case, AGVs would get no prolongation of their current segment and part of their current segment would be blocked for other AGVs until they are considered again.

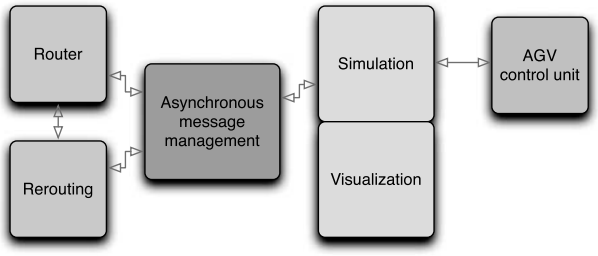

Figure 9 Architecture of the whole system.

These rerouting techniques were successfully tested on specially designed scenarios and with an asynchronous message management between the router and an AGV control unit that triggered the disturbances. Figure 9 shows the essential components of our implementation. HHLA acquired the source code and the copyright in 2009.

2.5 The Impact on Theory

This project has stimulated several research questions that were not part of the project with HHLA, but are typical examples of how industrial applications trigger research.

One type of questions concerned our sequential approach. Why route AGVs sequentially one after the other? Might it not be better to first gather some routing requests and then compute routes for them simultaneously? What is the "sequential gap", i. e., the optimality gap between these two approaches?

Another type of questions deals with the difference between the static router and the dynamic router developed in the project. Is it possible to improve the static router by better deadlock avoidance? Is it then competitive with the dynamic router?

In both cases, these questions were answered by a combination of new theoretical results and again experiments, thus creating their own algorithm engineering problems.

Questions about the sequential approach were treated in a simplified model of "dynamic grid flows" [4], in which the routing graph is a subgraph of a grid graph, travel times are constant along an edge, and simultaneous bends and crossings at vertices and simultaneous use of edges are forbidden. We could show that deviations from our sequential approach lead to NP-hard problems in this model. More precisely, if waiting along a route is not permitted (except at the start), then it is weakly NP-hard to compute a quickest route. The simultaneous routing of several AGVs even leads to a strongly NP-hard multi-commodity flow problem. So, unless $P = NP$, a polynomial routing algorithm must permit waiting along a route and compute the routes sequentially. This nicely justifies our approach from a theoretical viewpoint.

The sequential gap was determined experimentally by comparing the sequential router in the simplified model with an exact integer multi-commodity flow algorithm in the time expanded version of the routing graph, see Fig. 9. It was significant for a small number of horizontal lanes (11% for $m = 2$ lanes), but dropped quickly to less than 1% on average for 6 horizontal lanes. In our application, the routing graph had 6 horizontal lanes at the bottom (see Fig. 4), showing that the sequential gap can be neglected in practice.

Possible improvements of the static router were also first considered in a theoretical model which led to a two-stage algorithm [4]. In the first step, we computed short

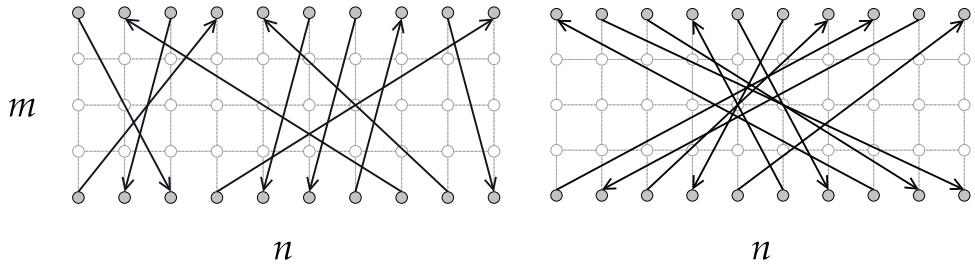

Figure 10 Routing graph (light gray) and routing requests (directed edges) for the experimental evaluation of the sequential gap as a function of m for large n. The gap was on average less than 1% for $m \geq 6$.

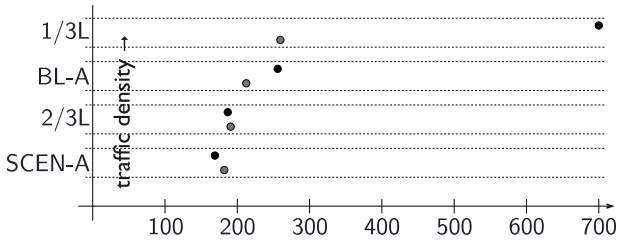

Figure 11 Comparing average travel times (x-axis) of the improved static router (dark gray points) and the dynamic router (light gray points) on 4 scenarios (y-axis) with increasing traffic densities. The dynamic router is clearly superior on high traffic scenarios.

static routes in an online fashion that load balancing with length bounds on the paths to create few collisions at run time. In the second step, we then computed a deadlock-free schedule for these routes. This second problem turned out to be NP-hard again, but can be solved by an algorithm that is exponential only in the number of AGVs.

An experimental evaluation (see Fig. 11) showed that the improved deadlock-free router is slightly superior to the dynamic router (bottom scenarios) but clearly loses against it in high traffic scenarios (top scenario).

Summing up the findings from the project and the additional, partially ex post theoretical analysis, the complexity results justify the sequential routing algorithm with waiting along a route. It is polynomial in theory and fast in practice and can handle disturbances. The resulting sequential gap is negligible for a harbor grid layout. The static approach can be improved by load balancing and proper deadlock avoidance but the average travel times increase rapidly with traffic density.

3 Integrated Sequencing and Scheduling in Coil Coating

The coating of coils plays an essential role in shaping steel producers' extremely diverse product portfolio. Coils are rolls of sheet metal (see Fig. 12) that need to be color coated in consecutive stages. They are thus prepared for specific uses in different companies. For instance, sheet metals for home appliances already have their typical white coating when bought from the steel supplier, and the sheet metal used for car bodies already has an anticorrosion coating before it arrives at the automotive plant for pressing.

Figure 12 Coils in a steel factory.

In an application with PSI Metals and Salzgitter Flachstahl GmbH we studied how to compute good production plans for the coil coating process [2]. A sequence of coils of sheet metal needs to be color coated in consecutive stages. Different coil geometries and changes of colors require time-consuming setup work. In most coating stages one can choose between two parallel color tanks. This can either reduce the number of setups needed or enable setups concurrent with production. A production plan comprises the sequencing of coils, and the scheduling of color tanks and setup work. The aim is to minimize the makespan for a given set of coils.

3.1 Designing the Cooperation

Communication was one of the key issues in this project. We learned the problem initially only from PSI Metals who wanted an optimization module for their planning software suite in use at Salzgitter Flachstahl GmbH. Only later we got into direct communication with Salzgitter Flachstahl GmbH. This led to several surprises in designing the right model and choosing the right algorithmic tools.

At the outset, the problem presented itself as a sequencing problem (sequence the coils) with complex setup conditions between coils that may be sequence dependent (details below). We were assured that the main task lies in finding a good sequence. Finding

the best schedule observing the setup conditions for a fixed sequence would be easy. Only after delivering our first schedules (see Fig. 2) to Salzgitter Flachstahl and discussing the results we learned that there were yet unknown subtle conditions that made this scheduling problem for a fixed sequence NP-hard. As a consequence, we had to change large parts of our model, which required a new theoretical analysis (which we enjoyed), but we busted our deadline and had to redesign our code and renegotiate the cooperation.

3.2 Finding the Right Model

Leaving out many details, a basic model of coil coating is as follows. Coils usually have a length of 1–5 km, and their central attributes are the coatings they receive in the four coating stages, chosen from a palette of several hundreds, and their width, usually 1–1.8 m. Before entering the coating process, each coil is unrolled and stapled to the end of its predecessor. To bridge non-productive time during setups, transition coils are inserted in between actual coils, so essentially a never-ending strip of sheet metal is continuously running through the coil coating line. After undergoing some chemical conditioning of their surface, the coils run through a top and bottom primer coater, an oven, a top and bottom finish coater, and through a second oven. In the ovens, the respective coating layers are fixed. After the coating process, the coils are rolled up again, now ready for shipping. A schematic view of a typical coil coating line is depicted in Fig. 13.

Some coaters have two tanks, each with its own roller, which can be used alternatingly to apply color. Setups comprise color or roller changes on a coater if consecutive coils use the same tank. A color change refers to cleaning a tank and filling it with a new color. A roller change needs to be performed when a coil with smaller width is preceding a coil with larger width on the same tank. Figure 13 displays some typical setups. Note that setups may depend on a long subsequence since tanks may be idle for coils between start and end of the subsequence. The optimization goal is to minimize the makespan for coating the given set of coils, i. e., the completion time of the last coil in the sequence.

One of the modeling surprises was that setups can be scheduled in two ways for a given sequence. Either between coils causing non-productive time, or concurrently to production on an idle tank. While the former results in an increase in makespan, the latter does not. Consequently, there are two possibilities to save cost by good scheduling. On the one hand, the tank assignment can preserve used colors and/or rollers on an idle tank for later coils, thereby reducing the number of setups to be performed. On the other, setups can be performed concurrently to coating on idle tanks, thereby keeping setups from increasing the makespan. Figure 15 illustrates this.

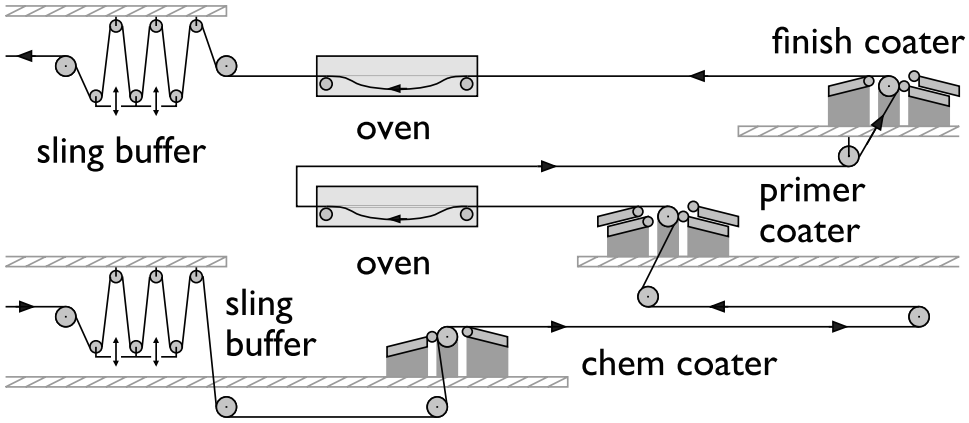

Figure 13 A coating line.

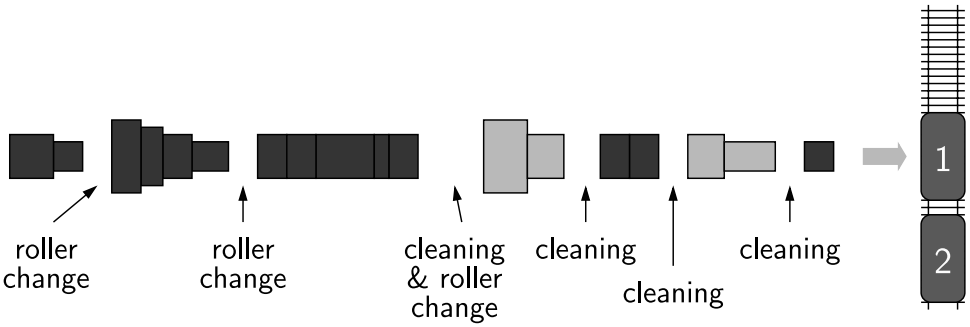

Figure 14 Setups between coils (white space). A cleaning includes a color change. At the right is a coater with two color tanks.

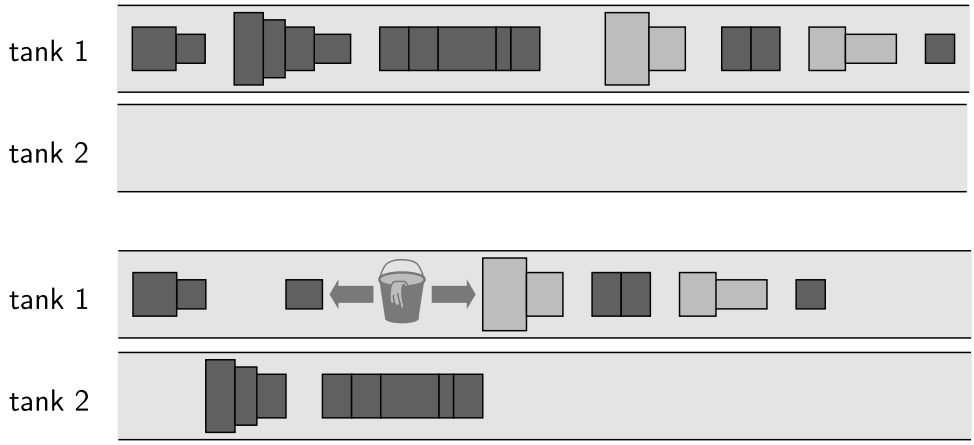

Figure 15 Concurrent setup scheduling. The schedule on the top uses only one tank, while the schedule below uses both and can schedule a cleaning setup on tank 1 concurrently with coating on tank 2.

Moreover, and this was an additional surprise, there is a safety constraint. While there may be several work teams to perform setups, at most one team can work on the same coater concurrently to production to ensure safety. Contrarily, during non-productive time, the teams work together to finish work as fast as possible. In addition, concurrent setups are not allowed while transition coils are run, as transitions require careful monitoring. Finally, the execution of one concurrent setup must not be preempted by another.

These "surprises" required a new theoretical analysis of the problem and a redesign of our algorithm.

3.3 Coping with Concurrent Scheduling

We developed a representation of schedules for a given sequence as a family of weighted 2-dimensional intervals, where the first dimension is related to a tank assignment and the second to performing concurrent setup work. We call the axis-parallel rectangle induced by two such intervals a "saving rectangle" as it represents savings through concurrent scheduling. Two saving rectangles are called independent, if their projections onto neither of the axes intersect. Due to the safety constraint, these rectangles share a common axis representing concurrent setup work which may only be done by one team, while the tank assignment axes are independent for different coaters. Altogether, this gives a "cylinder" as depicted in Fig. 16 with the concurrent setup axis common to all rectangles.

An optimal schedule S then corresponds exactly to a maximum weight subset of pairwise independent 2-dimensional saving intervals, cf. Figure 16. All this is done in a proper graph theoretic framework provided by so-called 2-union graphs, which generalize interval graphs.

This model essentially laid the groundwork for a number of results relevant to solving the concurrent scheduling problem.

First, the concurrent scheduling problem is strongly NP-hard when the number of coaters is not fixed. Second,

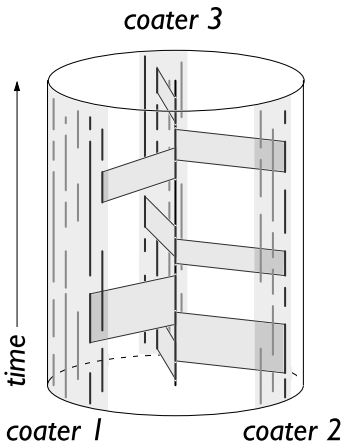

Figure 16 Representation of saving rectangles.

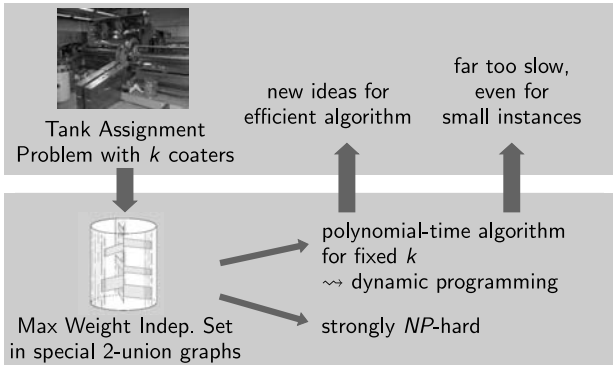

Figure 17 Interplay between theory (bottom row) and practice (top row).

for a fixed number of coaters, it leads to both an exact polynomial dynamic programming algorithm and a fast heuristic derived from it for our application, the former yielding optimal coil coating plans for fixed sequences of coils in polynomial time for any constant number of coaters. Figure 17 illustrates this interplay between theory and practice.

3.4 Combining Sequencing and Scheduling

We were now ready for the final design of our algorithm. Because of the sequence-dependent nature of the setups, we used a genetic algorithm for the generation of sequences and fast heuristics for computing the schedule for a fixed sequence. Algorithm engineering experiments evaluated different features for the genetic algorithm, different heuristics for computing good schedules, and different lower bounding strategies for evaluating the quality of our solutions.

A key ingredient of the genetic algorithm was the construction of its initial population, which was best when it was highly diverse with respect to different beneficial aspects. Recall that the cost of a solution essentially computes as the sum of local and global cost, where the former only depends on the sequence, while the latter is greatly affected by the scheduling. We generated a broad variety of completely different individuals, each composed of sections which are attractive for a special part of the objective. In addition we optimized their local cost, which is an asymmetric traveling salesman problem, with the Lin–Kernighan–Helsgaun heuristic.

For the scheduling part, we compared the independent set heuristic derived from the dynamic program with a simple on-line first-in first-out heuristic. Whenever subsequent coils have different colors, switch the tank. If the new tank does not contain the required color, a color change on that tank becomes necessary. So whenever a third color besides the two in the shuttle tanks is required, the color which was in use earlier is discarded. As expected, our independent set heuristic for setup scheduling proves superior to the simpler FIFO online rule. We were unable, however, to find a uniform choice of best parameters of the independent set heuristics suitable for all instances alike. As a result we used the independent set heuristics for long-term planning (coils for one week) and the FIFO online rule for short-term planning (coils for one day).

Finally, we developed and implemented an integer programming model of a combinatorial relaxation of the problem, which we were able to solve by branch-and-price for short-term planning instances. The solutions yield lower bounds on the optimal makespan of our short-term instances when using an online tank assignment rule, proving that our heuristic solutions have an optimality gap of no more than 10%. Figure 18 displays the relative performance of our algorithm for short-term planning.

Our optimization module has been acquired by PSI Metals and is in use for day-to-day planning at Salzgitter Flachstahl GmbH since December 2009.

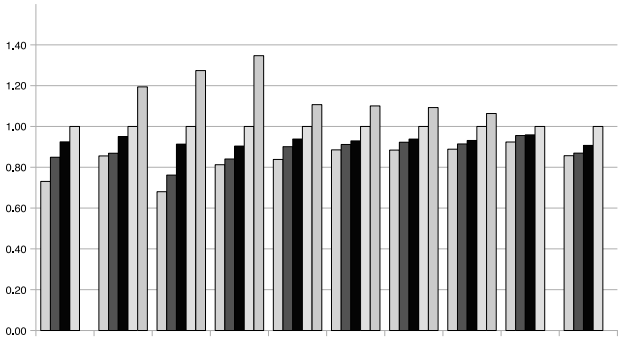

Figure 18 Comparison of normalized bounds and makespan for representative short-term instances when restricted to FIFO tank assignment. Bars from left to right are (1) the trivial lower bound given by the sum of coil processing times, (2) the sum of processing times plus local cost in an optimal, local cost based TSP solution, (3) our IP bound, (4) the normalized makespan obtained using the FIFO online rule for scheduling (value equal to 1), and (5) a reference solution devised by an expert human planner where it was available.

Acknowledgements

I thank Felix König for many of the pointed statements about the "rules of the game" in Sect. 1.

References

[1] E. Gawrilow, E. Köhler, R. H. Möhring, and B. Stenzel. Dynamic routing of automated guided vehicles in real-time. In: *Mathematics – Key Technology for the Future*. Joint Projects between Universities and Industry 2004–2007, W. Jäger and H.-J. Krebs (eds), pages 165–178, Springer, 2008.

[2] W. Höhn, F. G. König, M. E. Lübbecke, and R. H. Möhring. Sequencing and scheduling in coil coating with shuttles. In: *Management Science*, **57**(4):647–666, 2011.

[3] O. Jahn, R. H. Möhring, A. S. Schulz, and N. E. Stier Moses. System-optimal routing of traffic flows with user constraints in networks with congestion. In: *Operations Research*, **53**(4):600–616, 2005.

[4] B. Stenzel. Online Disjoint Vehicle Routing with Application to AGV Routing. PhD thesis, Technische Universität Berlin, 2008.

Received: March 4, 2011

Prof. Dr. Rolf H. Möhring is Professor at TU Berlin, where he heads the research group "Combinatorial Optimization and Graph Algorithms" (COGA). His research interests center on combinatorial optimization, graph algorithms, and industrial applications. Part of his research is done in DFG Research Center Matheon, where he is Scientist in Charge of Application Area "Logistics, traffic, and telecommunication networks".

Address: Technische Universität Berlin, Institut für Mathematik, MA 5-1, Straße des 17. Juni 136, 10623 Berlin, Germany,
e-mail: rolf.moehring@tu-berlin.de