Case Study 4:

Value Chain Optimization

–Final Report–

UniDo

June 3, 2005

AMETIST DELIVERABLE 3.4.4

Project acronym: AMETIST
Project full title: Advanced Methods for Timed Systems
Project no.: IST-2001-35304 Project Co-ordinator: Frits Vaandrager
Project Start Date: 1 April 02
Duration: 36 months
Project home page: `http://ametist.cs.utwente.nl/`

1 Introduction

The industrial partner AXXOM provided a value chain optimization problem which represents a typical planning and scheduling task as it occurs in the lacquer producing industries. Throughout the duration of AMETIST, this case study served as a case study to (i) develop and compare different modeling formalisms for scheduling problems, (ii) to evaluate the solution performance of the various approaches to scheduling problems investigated within AMETIST, (iii) determine which type of constraints are difficult or relatively easy to handle, and (iv) to further develop the methods based on the bottlenecks found in the previous steps. This report first contains the problem description including different versions of the case study, and then summarizes the modeling formalisms, solution approaches, and results for six different tools applied to the problem.

2 Problem Statement

2.1 Original Formulation

The benchmark is derived from an existing pipeless batch plant in which three different lacquers are produced according to the following scheme [10]: The materials required to obtain a specific lacquer are first prepared in a pre-dispersion and a dispersion unit, and are subsequently filled into mobile mixing vessels. After completing the filling procedure through a set of dose spinners, the mixing is carried out for a product-dependent duration. A quality check determines if the product meets given quality requirements. If so, the mixing vessels are emptied into filling stations and the product is delivered to the customers – if not the quality is improved by returning the lacquers to the dosing operation. The plant comprises 8 production resources and 6 mobile vessels, where some resources and one vessel are only used for one type of recipe while the others can be assigned freely. The operations cannot be interrupted (non-preemptive scheduling) and the material remains in the mixing vessels during some of the steps, as the quality check in the laboratory, or possible additional mixing. This leads to a situation where the operation times of the vessels are variable because they result from the scheduling of the operations for this batch. For some operations, more than one piece of equipment can be used. Each production order has a release date (earliest starting time) and a due date (deadline). The main interest is in meeting the deadlines.

The problem of assigning the jobs (consisting of the aforementioned sequence of operations to obtain a lacquer) to the 14 resources belongs to the class of job shop problems, with the additional complication that mixing vessels are required as a second resource to perform the operations on some machines. Another property of the problem, which is not standard in job-shop problems, but often found in scheduling problems of the chemical industries, is the presence of constraints on the storage times. These constraints are expressed as bounds on the differences between the starting and the ending times of two operations of a job. Three types of such constraints occur: start-start, end-start, and end-end constraints. The time horizon of the entire problem defined by the earliest release date and the latest deadline of all jobs comprises approximately 7 weeks, and the processing times on the machines range from few hours up to three days for laboratory testing. The objective function of the scheduling problem combines costs for the delayed finishing of jobs, operational costs per amount of product, and storage costs (i.e. early termination of jobs is penalized). When minimizing this cost function, it has to be considered that the three different lacquers incur different production cost.

2.2 Extended Version

In an extended version of the case study (set up during the second year of the project), a set of new problem instances has been defined, which either involve new constraints or additional production orders. The first extension introduces two new types of constraints: sequence-dependent changeover procedures and the interruption of operations during weekends, holidays and at nights.

Changeover procedures are defined to prepare the filling stations for new operations, i.e. to change the configuration of the resource. Such procedures often represent cleaning of the machines. The resource is allocated in an exclusive fashion for 5 to 20 hours and the desired operation is started immediately afterwards. Changeover procedures must be invoked when the filling station has to process a different type of lacquer, i.e. when it has to enter a new configuration. The possible configurations correspond to the three basic recipes for lacquers. The effect on the schedule is two-fold: the start of the operation is delayed by the changeover and it causes costs.

Since some machines are operated in a 2-shift mode on working days only, appropriate modeling of interruptions during night shifts and weekends is required. Interruptions mean that the operations have to be stopped at the beginning of the given interruption period, and to be resumed when the next working day starts. During the period, the resource remains allocated and the current operation cannot be preempted by other operations. Some operations may be interrupted for at most 12 hours. Both extensions reflect the industry's need for solutions to realistic and challenging problems.

In the other problem instances, the number of production orders has been increased to 73, 219, 500 and 1000. As a new optimization criterion, the minimization of total costs of the schedule is the goal. For this purpose, different costs have been assigned to the storage of the final products, to the delay of finishing jobs, and to changeover procedures. These extended settings involve fixed costs as well as cost rates to be integrated over the duration of operations or quantities of the orders.

3 Modeling and Requirement Specification

It was observed in the initial phase of the project that the academic partners interpreted the various constraints and cost contributions contained in the problem formulation not identically. It was deemed necessary to produce a representation scheme that formulates the requirements in an intuitive and unique manner. A systematic modeling scheme comprising the following steps was developed [12, 13]:

(a) create a dictionary to explain the domain-specific vocabulary used by the industrial problem providers,

(b) resolve semantic ambiguities in the specification of the production sequence and constraints,

(c) define adequate levels of abstraction with an explicit representation of the design decisions,

(d) supplement the so-called *product flow diagrams* (provided by AXXOM) with recipe-like representations, and

(e) systematically transform the latter representations into timed automata.

With respect to step (d), the product flow diagrams provided by AXXOM, which represent the production steps in a graphical arrow-node representation, were first enriched by the processing and offset times for the operations and by the information on the resources on which an operation can be carried out. (The latter data were originally provided in separate tables.) An example for such an extended product flow diagram is shown in Fig. 1 – a drawback of these diagrams is, however, that parallel and alternative operations cannot be distinguished in all cases, and that some timing restrictions have different meanings in different situations. It has thus been proposed in [13] to supplement the product flow diagrams by the recipe-based representation shown exemplarily in Fig. 2. This representation illustrates in which sequence the different types of resources have to be allocated for a specific lacquer, and which timing requirements are relevant.

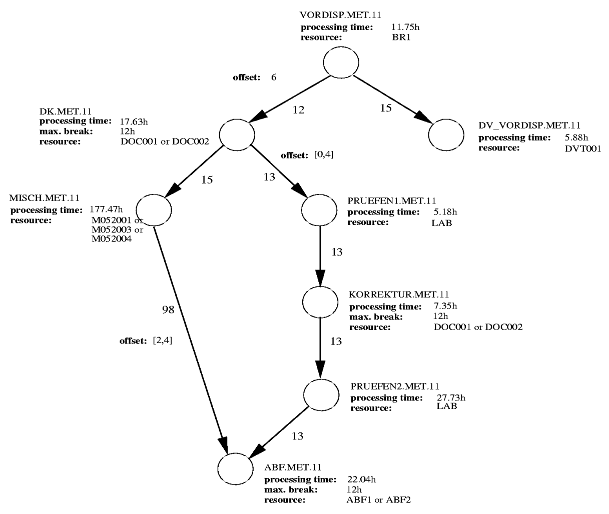

Figure 1: Example of a product flow diagram.

Based on these representations, the systematic transformation into Timed Automata is achieved relatively easily by:

1. translating each production step from the recipe representation into an automaton fragment (i.e. a sequence of states and transitions),
2. composing the fragments,
3. converting the timing constraints (e.g., the processing times) into corresponding guard and invariant conditions, and
4. modeling the resources by variables.

A different modeling approach, common in the industrial scheduling community, is the state-task-network (STN) formulation with subsequent solution as algebraic program by the use of mixed-integer optimization techniques. A standard representation of STN is one with discrete time representation where changes of the states of the model can only occur at equally spaced time instants. As any possible event can occur at any of these instants (modeled by a binary variable), these models quickly become very large if the number of events is large, i.e. the scheduling horizon is large compared to the maximal resolution required to meet the constraints. For the given case study, this ratio becomes rather large and hence such models are not suitable here.

The alternative approach for algebraic modeling followed in this project is the use of continuous *event points* at which an event (e.g. the start or the end of an operation) can occur [14, 1]. The number of such events can be computed from the production orders and the recipes (plus eventually a worst-case estimate for sequence dependent operations such as cleaning operations). In the example considered here, there are typically not more than a few hundred events whereas a discrete time STN formulation leads to several hundred thousand binary variables. On the other hand, though, the relative positions of the events must be represented by binary variables in order to express constraints of the type that two operations cannot be performed on the same machine at the same time (but either operation A must finish before B starts or B must finish before A starts). It turned out that a standard formulation of a continuous time MILP model also gives rise to rather large models with thousands of binary variables for problems with more than 10 jobs. Therefore the model is reduced by additional heuristics, which exclude alternatives that cannot lead to an improvement of the schedule. The heuristics, which are described in more detail in the next section, eliminate a considerable number of binary variables, such that the total number of binary variables increases only linearly with the problem size.

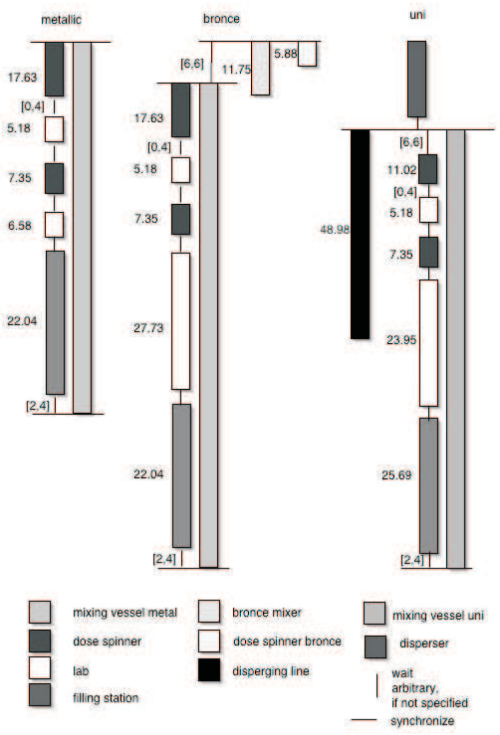

Figure 2: Recipe-based representation of the production sequence.

4 Tool Application and Solution

4.1 Mixed-Integer Linear Programming with Cplex

In order to solve the algebraic program sketched before, the following heuristics were employed [19]: (i) pre-ordering by earliest due dates (EDD), (ii) non-overtaking of operations belonging to jobs of the same type, and (iii) non-overtaking of operations whose start and finishing dates do not overlap. These heuristics were applied in a two-step-procedure as follows: First, the heuristics (i) and (iii) are applied by fixing binary precedence variables, and the problem is solved. In the second step, the variables fixed before are relaxed, and the heuristics (ii) and (iii) are applied by fixing the corresponding binary variables. This problem is then solved by using the result of step one as initialization. This procedure was combined with a rolling-horizon solution procedure which decomposes the problem into a series of smaller sub-problems. These are first solved separately, and the solution of the complete problem is finally composed of the solutions of the sub-problems.

The optimization was performed with the commercial package GAMS/Cplex. Parameter studies of various Cplex parameters led to the result that the option $dpriind = 1$ and default values for all other parameters are the preferable option for the given problem. Table 1 shows results obtained for the original formulation of the case study as described in Sec. 2.1, but with a varying number of jobs using GAMS/Cplex, version 21.7. The optimality gap as termination criterion was set to zero, meaning that the search is continued until the optimal solution is found. In all tests, an optimal solution with zero accumulated tardiness was found. Further results are reported in [9, 19, 15]. In the last year of AMETIST, the MILP model was extended by constraints for changeover procedures (see Sec. 2.2). It involves additional real variables to determine direct predecessors of operations on the filling stations. If a changeover procedure is necessary for a pair of two successive operations, the duration of the second operation becomes extended by the duration of the changeover procedure. The results in Tab. 2 show the computational effort required to solve the extended model for a

Table 1: Optimization of the original problem formulation with a variable number of jobs. *time 1* and *time 2* refer to the two stages of the solution procedure. Times are in seconds.

number of jobs	time 1	time 2
10	3.95	2.87
14	15.86	0.61
16	23.83	0.92
18	46.72	1.20
20	58.71	1.53
22	100.23	1.85
29	412.93	2.61

varying number of jobs. Two versions of GAMS/Cplex were used to document the impact of the software version on the results. The optimization objective here was to minimize the aggregated tardiness of all jobs. In all test the optimality gap was set to zero and optimal schedules with zero accumulated tardiness were found. The experimental environment was a 2.4 GHz Pentium 4 machine with 1 GB of memory and the Linux operating system. The following conclusions can be drawn from the experiments:

1. The version of the software has no considerable influence on the solution performance and, thus, the results are comparable to those published in [19].
2. The fact that new variables and inequalities had to be defined for the modeling of changeover procedures only causes a moderate increase of the computational effort.

Table 2: Results for a MILP model extended to changeover procedures. Solution times are given for problem instances with varying numbers of jobs and two versions of GAMS/Cplex. The pairs of numbers refer to the two stages of the solution procedure. Times are in seconds.

number of jobs	GAMS 21.3	GAMS 21.7
10	0.90/2.85	3.95/2.87
20	232.48/5.64	245.55/5.78
25	779.10/7.56	321.38/7.76
29	848.60/5.88	830.26/6.42

4.2 ORION-PI

Axxom applied its own software for specifying and solving scheduling problems, ORION-PI. It uses a particular modeling concept based on so-called *quants*, which represent the smallest logistic units required to model the problem. ORION-PI performs a quant-based combinatorial optimization algorithm that employs the principle of branch-and-bound. The exact solution algorithm is, however, not public domain.

Axxom performed two series of tests, one for the original version, and one for the extended version that takes additional constraints for the operating hours, maximum break times, and production rates into account. The cost function is given as the sum of delay costs (delay time times input costs times the quantity of the order) and storage costs (storage time times input costs times quantity of the order). Different scenarios for 29, 500, and 1000 orders are considered, and for 29 orders, cases with and without requiring a delay-free solution are included.

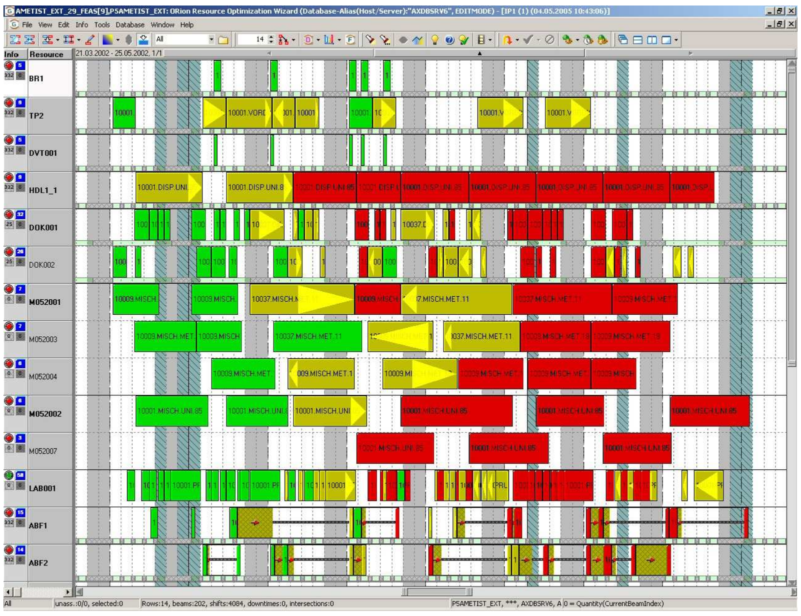

Figure 3: Gantt chart for the case: extended problem version, 29 orders, red (dark) operations are delayed.

Table 3: Results for the original version of the case study. The column *delay costs* answers the question whether a delay-free schedule could be found or not (this was the only optimization criterion required for this comparison). For the first scenario, 'O' refers to results generated with ORION-PI, 'U' to Uppaal, and 'G' to GAMS/Cplex – if not marked the results are obtained with ORION-PI. The tests with ORION-PI were carried out on a PC with 2.66 GHz processor and 1 GB RAM, and the last two on a PC with 3.06 GHz processor and 2 GB RAM.

Number of jobs	storage costs	delay costs	total costs	jobs in time	calc. time
29	O: 2627.32	O: 0	O: 2627.32	29	11 sec
	U: 2840.18	U: 0	U: 2840.18	29	0.2 sec
	G: 2113.83	G: 0	G: 2113.83	29	413 sec
29	1459.62	146.08	1605.71	22	13 sec
500	10648.16	34102.61	44750.77	200	32 min
1000	21750.51	66645.25	88395.77	403	97 min

Table 3 shows results for the original version of the case study and for four different scenarios. For the first scenario (29 orders, delay-free solution), results obtained with Uppaal and GAMS/Cplex are included for comparison. Results for the extended version of the case study are shown in Tab. 4, and Fig. 3 contains exemplarily the Gantt chart obtained with ORION-PI for the first scenario listed in Tab. 4. The visualization shows the single quants assigned to the resources with the following meaning: red - delayed, yellow - just in time, green - too early. The generated schedules meet the given constraints, as e.g. the requirement that the resources can only be used within the given operating hours.

Table 4: Results for the extended version of the case study. The accumulated delays of the schedules are shown in column *delay costs*. 'O' / 'U' refer to results obtained with ORION-PI, or Uppaal respectively. All ORION-PI test were performed a PC with 3.06MHz processor and 2 GB RAM.

Number of jobs	storage costs	delay costs	total costs	calc. time
29	O: 2662.68	O: 9070.04	O: 11732.73	11 sec
	U: 5706.78	U: 64.02	U: 5770.80	-
29	409.46	17496.61	17906.07	13 sec
500	33363.29	335754.56	369117.86	65 min
1000	138085.68	301544.73	439630.42	145 min

4.3 Synthesis of Schedules with Uppaal

Using the procedure to obtain timed automaton models of the cases study, sketched in Sec. 3, the tool Uppaal [6, 2] was used to derive feasible and optimal schedules[3, 4]. The scheduling problems were composed of the following components:

1. A set of recipe templates representing individual types of lacquers; these templates were instantiated with parameters (release and due dates, quantities, costs) to model concrete production orders as job automata;
2. Additional automata to model resources which have their own clocks (for changeovers) as well as for interruptions and synchronization purposes;
3. Extensions of the automata to model various heuristics and cost criteria;
4. A property of the composed TA to be satisfied (minimizing a given cost criterion).

The following set of heuristics was considered: no overtaking of jobs (jobs started earlier receive a resource earlier), non-laziness (a required and available resource must not remain unused), greediness (a job allocates a resource as soon as possible), reduction of active orders (i.e. the number of jobs currently processed), increase of release dates (to reduce the possible storage costs). How these different heuristics and the different types of costs are established in Uppaal (or Uppaal-Cora for cost-optimal reachability) is described in detail in [5]. Overall, 33 Uppaal models were created for a varying number of jobs (29, 73, 219), the availability of resources (always available, or use of availability factors [as the fraction of time in which a resource can be used because operators are available], or available at explicit working times), optimization criteria (with and without costs), heuristics (non-laziness, greedy, non-overtaking, increment of release dates, limitation of active jobs).

The groups of problem instances are numbered from 1 to 7, and the corresponding problem properties can be summarized as follows:

1. 29 orders, hard release and due dates, feasibility problem (i.e, deadlines must be satisfied, no cost optimization);
2. 73 orders, hard release and due dates, feasibility problem;
3. $n \times 73$ orders as in 2), hard release and due dates, n vertical concatenations of the job table from 2) with proportional extensions of release and due dates, feasibility problem;
4. $n \times 73$ orders as in 2), hard release and due dates, n horizontal concatenations of the job table from 2) with the release and due dates, and the n replicates of all resources from case 2); feasibility problem;

5. 29 orders, no hard deadlines, cost minimization of storage and delay costs for final products and setup costs for filling lines;
6. 29 orders, no hard deadlines, working time constraints, cost minimization of storage and delay costs for final products and setup costs for filling lines;
7. 29 orders, no hard deadlines, working time constraints, cost minimization of storage costs for intermediate and final products, delay and setup costs for filling lines.

If the generation of delay-free schedules is the objective, the reachability algorithm of UPPAAL searches for a reachable state representing that all jobs are completed before their due dates are passed. The trace into such a state (produced by UPPAAL) represents directly the schedule. The analysis is performed by choosing depth-first or random depth-first search, where the latter was more successful. Table 5 shows results for the feasibility analysis as obtained with Uppaal 3.5.6 on a 2.6 GHz Intel-P4-Xeon processor and 2.5 GB of memory running Linux kernel 2.6.8. The model number refers to the problem instances listed above.

Initial experiments revealed scalability problems, partly caused by the large number of clocks contained in the models. The heuristic limit of the number of active jobs also provides a limit on the number of clocks needed (one per active job instead of one per job), and the non-overtaking heuristics provides an easy way of uniquely assigning shared clocks to jobs, since the starting order of jobs of a particular type is fixed. This change reduced the number of clocks to $3 \cdot A + 3$, where A is the maximum number of active jobs. The results show that, even for the case of 29 jobs, the use of the heuristics is essential. The non-overtaking heuristics does not make much difference for a case without availability factors, whereas in the case with availability factors the performance gets worse. For 73 jobs, however, non-overtaking decreases the computation time considerably. Limiting the number of active jobs increases the speed by several orders of magnitude (partly due to the possible reuse of clocks).

model version	**number of jobs**	**heuristics**	**pf / av / ex**	**max. active orders**	**search time**	**termination rate**	**min. costs**	**average costs**
5	29	g	yes/yes/no	5	10 min.	10/10	$11 \cdot 10^6$	$13 \cdot 10^6$
6	29	-	yes/no/yes	5	10 min.	10/10	$2.1 \cdot 10^6$	$2.7 \cdot 10^6$
7	29	-	yes/no/yes	5	10 s.	3/10	$80 \cdot 10^6$	$81 \cdot 10^6$

Figure 4: Table of experiments for the versions including costs with performance factors (pf) for all models, availability factors (av) in model 5, and explicit working hours (ex) in the models 6 and 7. A random-best-depth-first search of 10 minutes was used for the experiments with models 5 and 6. The experiments with model 7 (which used the same search order) had to be limited to 10 seconds due to problems with running UPPAAL CORA. The number of successful runs (termination rate), the lowest cost of any run and the average cost of each run is shown.

In addition, experiments were performed for the extended version of the case study (Sec. 2.2) using Uppaal-Cora, see Tab. 4. Although the constraints of explicit working hours (used in the models 6 and 7) makes the model much more complex, the results show that the schedules have much smaller cost than the schedules for model 5. A possible explanation is that the availability factors distribute the availability uniformly over time. In reality, however, a dose spinner is completely

model version	**number of jobs**	**heuristics**	**max. active orders**	**no av, no pf**	**av, no pf**	**av, pf**
1	29	-	-	78.0	-	-
1	29	nl	-	0.3	1.5	3.5
1	29	nl no	-	0.3	2.5	3.6
1	29	g	-	0.2	2.8	-
1	29	g no	-	0.2	2.9	-
2	73	-	-	-	-	-
2	73	nl	-	-	-	-
2	73	nl no	-	24.6	-	-
2	73	nl no	3	0.4	-	-
2	73	nl no	4	1.1	0.4	0.3
2	73	nl no	5	4.4	0.6	0.9
2	73	g	-	15.0	-	-
2	73	g no	-	9.2	43.3	30.5
2	73	g no	3	0.3	165.1	-
2	73	g no	4	0.4	0.3	0.3
2	73	g no	5	2.2	0.4	6.8
3	146	g no	4	1.3	0.9	0.8
3	219	g no	4	3.5	2.1	1.9
3	292	g no	4	7.7	4.4	4.0
3	365	g no	4	13.3	7.5	6.7
3	438	g no	4	20.0	11.2	10.0
3	511	g no	4	28.2	15.8	14.0
3	584	g no	4	37.7	21.2	18.6
3	876	g no	4	89.7	49.7	43.8
3	1168	g no	4	166.4	92.1	80.7
3	1460	g no	4	270.9	149.4	131.0
3	1752	g no	4	401.3	222.3	194.0
3	2044	g no	4	565.4	311.6	271.9
4	146	g no	5	9.7	-	-
4	146	g no	6	167.8	-	-
4	146	g no	7	-	21.0	-
4	146	g no	8	-	18.2	12.6
4	219	g no	8	-	-	-
4	219	g no	9	-	-	347.9
4	219	g no	10	-	-	-
4	219	g no	11	-	-	-
4	219	g no	12	-	-	-

Table 5: Experiments for the generation of delay-free schedules. Abbreviations for the heuristics are: em g - greedy, *nl* - not lazy, and *no* - no overtaking. Each experiment was repeated with a model without working hours (*no av*, *no pf*), a model with availability factors and no performance factor (*av*, *no pf*) and a model with both availability factors and performance factors (*av* & *pf*). For each experiment the run time in seconds is provided. All measurements were done using depth-first search. A run was terminated after 10 minutes or when memory consumption reached 2GB (indicated by a '-').

available during the weekdays and totally unavailable in the weekends. Therefore, the availability factors give in some cases a large over-approximation of the processing times, with the result that scheduling becomes much harder. The meaning of the costs is as follows: in model 6, a schedule with cost of approximately 2 million (the best schedule of the 10 runs) is a schedule in which 2 orders are a bit late (1 day and 45 minutes, respectively), and the others are much too early (since intermediate products do not incur storage costs). In model 7, this effect is countered by storage costs for intermediate products, what makes the schedules more expensive. The investigation of these effects is still ongoing work. It should be noted that the non-laziness heuristics is not applicable to the extended case, since storage costs make it advantageous to be lazy. Also the non-overtaking heuristics cannot be adapted, since it can be advantageous for a job with low storage costs to overtake a job with higher storage costs. The greediness heuristics cannot be used either in the models 6 and 7, due a limitation in Uppaal. Current work includes to adapt all heuristics to the extended version.

The results show that feasible schedules can be derived with Uppaal such that the case with 29 orders is solved within 1 second, and the extension to 73 orders does not significantly increase the computation times (if suitable heuristics are used). To further scale up the model size, a vertical multiplication of two models with 73 orders was considered, and solutions could be obtained for up to 2044 orders. For horizontal composition of the case with 73 orders (multiplying also the resources) was successful up to 146 orders. To deal with the full set of constraints of the original problem (incurring setup-costs for filling stations, storage costs, and delay costs) the problem was transformed into a cost-optimization problem, and the latter was solved by Uppaal-Cora. A further extension was required to deal with the constraints for working-hours, which increased the size and complexity of the model significantly. But also for this case, feasible schedules could be derived with Uppaal-Cora.

4.4 Stochastic Analysis with Moebius

For the feasible schedules obtained with Uppaal, our objective was to investigate the robustness with respect to breakdowns of the resources, and to rank alternative schedules accordingly. The problem formulation provided by Axxom contains performance factors (formulating the fraction of time in which a resource is not operational) and availability factors (as the fraction of time at which a resource can be used because operators are available). Axxom proposed to extend the durations of operations by considering these factors. In order to check whether this modification is advantageous and to accomplish the schedule assessment, the following approach was taken [7]: The scheduling task was modeled in the language *MoDeST*, which combines modeling features from stochastic process algebra and from timed and probabilistic automata with light-weight notations such as exception handling. It is supported by the *Motor* tool, which facilitates the execution and evaluation of *MoDeST* specifications by means of the discrete event simulation engine of the *Moebius* tool. The performance and availability were modeled in detail by considering the mean time between failures (MTBF), the mean time to repair (MTTR), and a pace as the frequency of failure occurrence.

The performance analysis with *Moebius* included 80 experiments overall, where 20 different feasible schedules were investigated for two different paces and two different deadline policies (1: give up a job once it is sure it misses its deadline, 2: process all jobs to the end). One half of the schedules included the modeling of availability and performance, while the other half did not. Each experiment was executed 20,000 times and took around 4 to 5 hours. Figure 5 summarizes the results from this investigation (left part: availability and performance not considered; right part: both factors are considered). It can be concluded that: (a) a higher pace is advantageous to successfully complete a job in time, (b) the success rate of the investigated schedules does not differ significantly, and (c) it is not advantageous to simply extend the duration of operations by the availability and performance factors since jobs are started later as necessary if an equipment malfunction does not occur. Hence, the availability and performance factors should be used only for sequencing and the prediction of delivery dates but not for the timing of operations.

The investigation does furthermore allow to quantify the success rate for each individual job. The obtained rates confirm the expectation that jobs which are scheduled to start late finish less likely on time and that two jobs which are roughly started at the same time have a lower probability to be finished timely.

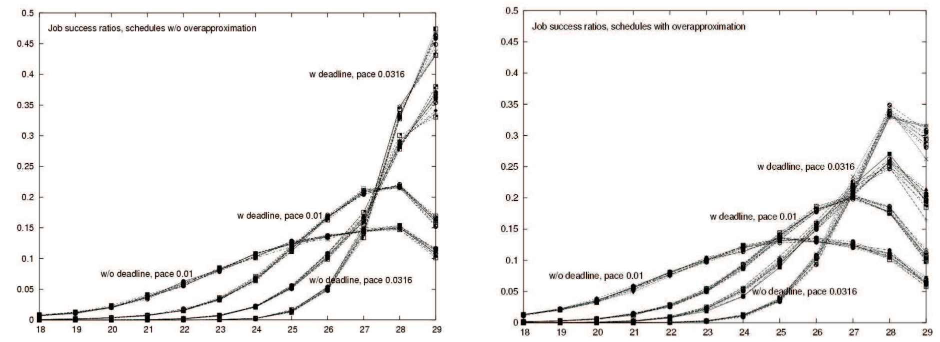

Figure 5: Assessment of schedules.

4.5 Generation of Schedules by IF

Schedules for the original formulation of the case study were also derived with the tool IF [8]. Using the restrictions of fixed processing times on resources and exactly two quality checks per job, a timed automaton model was built such that (i) the production sequence of each lacquer was modeled by an acyclic timed automaton, and (ii) the availability of resources was represented by shared variables. For the composition of all resulting automata, the scheduling task means to find a minimum cost path leading to a (global) final state in which all lacquers are finished. Since brute-force exploration of the state-space obtained for the problem with 29 jobs was impossible, three heuristics were employed: (a) only non-lazy runs are explored, (b) overtaking of jobs is forbidden, and (c) minimal separation times between the starting times of lacquers of the same type are chosen.

Modeling these heuristics in the IF language and solving the original version of the problem with the associated IF tools (in particular the module for computing paths with minimal costs), the following results were obtained: a feasible schedule without delay is obtained in 15 seconds, and the schedule can be extracted from the corresponding execution path of the model (with a depth of 750) almost instantaneously by random execution. The schedule is shown as Gantt chart in Fig. 6.

4.6 The Solution by TAopt

TAopt combines principles known from mixed-integer programming with reachability analysis of timed automata [17, 16], in order to establish efficient pruning criteria for the graph search. The main idea is to use linear programming to solve tailored relaxed subproblems in order to compute lower cost bounds for pruning. The lower cost bounds are also used as a heuristics to steer the search. The embedded linear programs are updated iteratively and then solved by Cplex to compute the lower bound of the cost-to-go for the current node of the reachability tree.

In order to model the case study in TAopt, the scheduling problems was first formulated as a resource task network (RTN) extended by additional information. RTN are a common and illus-

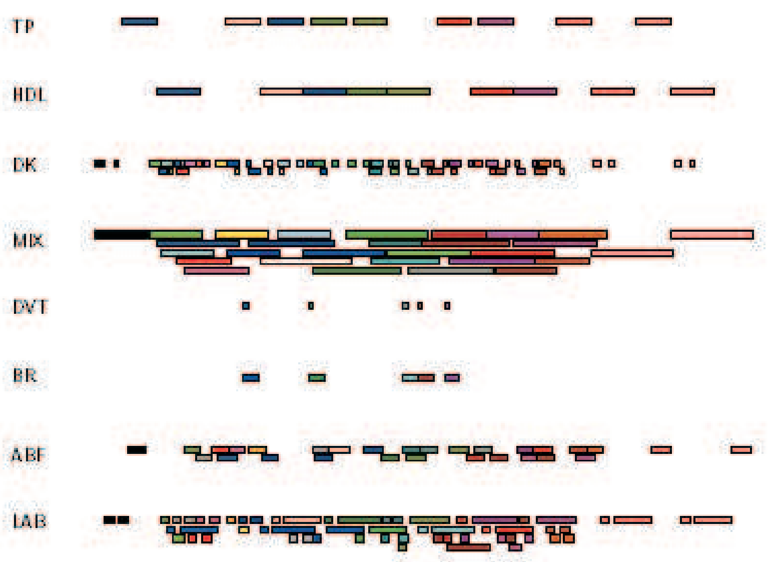

Figure 6: Schedule obtained with the tool IF.

trative description method for recipes in the operations research community and are widely used to describe scheduling problems. A problem specification in TAopt consists of:

1. A description of the plant, including all resources with capacity constraints, resource configurations and transitions between configurations (which represent changeover procedures);

2. Recipes formulated as RTN consisting of tasks, products, resource and configuration references, and arcs between the individual elements; additional timing and sequence constraints can be expressed using *places* labeled by a *marking* (similar to time Petri nets);

3. Production orders, each of which references a recipe and produces a new job as an instance of the recipe.

The instances can be modified with respect to release dates, due dates, and production quantities according to requirements of the individual production orders. In addition, tasks and places of the RTN can be decorated by costs and cost rates, respectively. An example, of an RTN for one type of lacquer is shown in Fig. 7.

The meaning of the individual elements of this RTN is as follows: Tasks are represented by rectangles and decorated with durations. Places represent timing and sequencing constraints of tasks and are depicted by thin circles. Initial markings are shown as black bullets. The attached intervals contain the minimal and maximal time in which a marking may be in the place. The following constant values are used in the recipe to instantiate a job: the release date *rd*, the due date *dd* of the job, and the time horizon *H*, i.e. an upper bound of the makespan. Thick circles represent resources which are linked to tasks by dashed undirected lines. Some of the links are additionally decorated with *MET*, which is the name of the configuration required by the corresponding task. Tasks without resources are dummy tasks used for synchronization purposes only. The lab resource and the corresponding tasks have been removed here.

Given such an extended RTN as input, TAopt automatically generates a timed automaton (TA) model and a corresponding MILP model (with partly relaxed integrality constraints). The TA model is extended by costs on transitions and cost rates on locations according to costs and cost rates obtained from the RTN. Both models are then used in TAopt to perform a cost-optimal

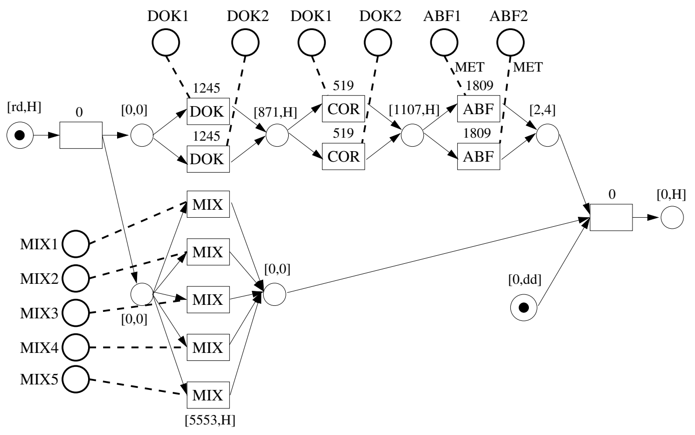

Figure 7: A RTN of the recipe for metallic lacquers.

reachability analysis (of the TA model), and to compute lower bounds of the cost-to-go using the LP model.

Table 6 shows results obtained with TAopt for the original version of the case study with a reduced number of jobs. We considered the minimization of the makespan for a varying number of jobs, where the first five problems do not include hard deadlines for the jobs, while the last two do. The search was terminated when a specified number of nodes (*node limit*) was explored. The table lists the number of solutions found, the best makespan among all solutions required to process all jobs, and the computation time (in seconds) to obtain the best result. In all tests no heuristics were used, thus these results reflect the performance of a pure depth-first search with cost minimization. Both criteria, maximal depth and minimal costs, were used as decision rule for removing nodes from the waiting list. For the last instance, no feasible solution could be found before the search was terminated after exploring $1 \cdot 10^6$ nodes.

As TAopt is still a prototype tool at this stage, embedded linear programs can so far be used only for job-shop problems. Another limitation comes from the fact that the traces potentially explored by TAopt are limited to *immediate traces* in which one dominating vector of clock valuations is identified in each zone to be stored in the internal data structures. Such representation allows for efficient representation and computation of symbolic states, but limits the possible optimization criteria. The chosen representation is sufficient to compute optimal schedules for simple optimization criteria as makespan, tardiness costs and storage costs with hard deadlines. Other more complicated cost criteria can be defined as well, but in some cases no optimal schedules can be guaranteed, e.g. for the combination of storage and tardiness costs. The solution of larger problem instances of the Axxom case study with embedded LPs is a subject of current work.

5 Assessment of the Achievements and Open Issues

Overall, different modeling techniques to suitably represent the value chain challenge problem were developed and tested within AMETIST, and the considered set of tools has been found to be able (after introduction of suitable heuristics in some cases) to solve different problem instances efficiently.

With respect to the approaches based on timed automata and (extended) model-checking techniques (Uppaal, IF), the experiences can be summarized as follows: A considerable effort had to be put into producing an appropriate TA representation of the problem specification provided by AXXOM. One part of this effort was to clarify and exactly interpret the constraints and relevant

Table 6: Optimization of the original problem formulation with TAopt for a varying number of jobs using the makespan as the optimization criterion.

No. jobs	deadlines	node limit	No. of solutions	best makespan	calc. time (sec)
10	no	$2 \cdot 10^5$	9	30920	54.9
12	no	$2 \cdot 10^5$	3	53216	75.9
14	no	$1 \cdot 10^6$	2	55371	525.5
16	no	$1 \cdot 10^6$	5	53792	661.9
18	no	$1 \cdot 10^6$	5	53692	839.5
10	yes	$2 \cdot 10^5$	6	30920	65.9
12	yes	$1 \cdot 10^6$	0	-	-

costs. In addition, specific schemes had to be developed to include the various constraints into the model. This shows, on the one hand, that the application of model checking techniques to this kind of production scheduling problems is not yet a push-button technology, since the models have to be constructed with care, and suitable heuristics have to be identified. On the other hand, one can realistically assume that many production scheduling problems have similar structures, such that the modeling patterns developed in AMETIST can be reused for similar scheduling problems. Further investigations have to identify a core collection of such patterns. Once the TA model is obtained, the application of model checking techniques for production scheduling is promising, as feasible schedules could be obtained in short computation times for not too large problems.

In direct comparison to the TA-based approaches, the effort for solving algebraic programs with the MILP approach was found to be higher in many instances. Even when heuristics were implemented, the solution procedure (which includes the time-consuming startup of GAMS/Cplex) often takes more time than the entire optimization performed by Uppaal. On the other hand, the solution quality is expected to be better because a true optimization problem is solved. Creating efficient MILP models is usually a laborious task and requires not only experience but also the investigation of suitable solver parameters to accelerate the solver performance. The advantage of the MILP approach is the possibility to assess the quality of the solution by evaluating lower bounds of the costs. Another advantage arises from the fact that the optimization criterion can be an arbitrary linear function, and its choice hardly affects the rest of the model. Thus, it seems easier to combine different models, heuristics and objective functions. However, the MILP approach is limited to medium-sized problems.

First test results of the scheduling algorithm implemented in TAopt[18] have shown that embedding the solution of LPs into the reachability algorithm for timed automata can efficiently prune the reachability tree, and thus reduces the memory consumption considerably. This advantage dominates the effort to solve embedded LPs when large models are optimized. In comparison to GAMS/Cplex it was shown in [18] that feasible schedules for benchmark job-shop problems could be computed quickly and the quality was better than for GAMS/Cplex, when appropriate state space reduction methods and heuristics are activated. However, the applicability of the implementation of TAopt is currently limited to models that can be expressed as RTNs with timing constraints, i.e. only a simplified version of the case study could be solved.

The approach of the industrial partner Axxom is competitive to the other approaches with respect to the solution effort. Similar to the MILP approach, it is based on a branching algorithm, extended by different heuristics to efficiently prune the state space and to provide feasible solutions quickly. Its advantage is the flexibility of using different (also non-linear) cost functions and constraints. The modeling is made comfortable by an appropriate GUI.

Based upon the results reported above, the performance and the range of application of the tools that were applied to the benchmark problem can be summarized as follows:

- For problem instances of moderate size, with possibly complex (but linear) cost functions and simple timing constraints (only conditions for the relative time-shift between operations), the MILP-approach is able to compute better solutions than a TA-based approach or ORION-PI in reasonable computing times. However, the modeling effort is comparatively large.
- For scheduling problems of moderate size but with complex timing constraints (e.g. limited working hours) and simple cost functions, the TA-based approach, e.g. Cora-Uppaal, is able to provide better solutions than ORION-PI if suitable reductions and heuristics are used. The modeling by MILPs in these cases gives rise to very large models in a continuous as well as a discrete time formulation that cannot be solved without further simplification.
- For large-scale problems with complex timing constraints (e.g. the consideration of night shifts) and cost minimization, ORION-PI at present is the only tool that can provide solutions with reasonable computational effort.
- Timed Automata are not yet a push-button technology to be applied without problem specific modeling and solution strategies. But the generation of libraries of templates for typical configurations seems promising and appears as a path towards to more widespread and easier application for non-TA-specialist users.

As for the stochastic assessment of schedules (with Moebius/Modest/Motor), the use of the performance and availability factors led to the question of proper interpretation. Extending the processing times by these factors can be used to analyze how many jobs can be handled on a longer time horizon. However, the stochastic analysis has shown that using performance and availability factors to obtain concrete schedules, increases the probability to miss deadlines if not only the sequencing but also the timing of such schedules is implemented. It is unclear at this stage what modeling assumptions are best suitable for the derivation of concrete short-term schedules, where storage costs have to be minimized, and where delay costs should to be avoided. A suitable idea may be to use a form of schedule refinement, taking rough long-term schedules as a basis for obtaining precise schedules for a shorter term – such a refinement approach was successfully used in the Cybernetix case study [11]. Another idea to be investigated in the future is that of searching for schedules in reverse time, starting from the due dates of orders – valid schedules obtained this way avoid storage and delay costs by construction.

References

[1] AMETIST, *First year report on case study 4: Value chain optimization*, may 2003, Deliverable 3.4.2 from the IST project AMETIST.

[2] G. Behrmann, *Guiding and cost optimizing uppaal*, Web-page, 2002.

[3] G. Behrmann, E. Brinksma, M. Hendriks, and A. Mader, *Scheduling lacquer production by reachability analysis – a case study*, Proceedings of the 16th IFAC World Congress, 2005, to appear.

[4] ______, *Scheduling lacquer production by reachability analysis – a case study*, Workshop on Parallel and Distributed Real-Time Systems, IEEE Computer Society Press, 2005, To appear.

[5] G. Behrmann, M. Hendriks, A. Mader, and E. Brinksma, *Axxom scheduling with uppaal. Technical Report, AMETIST*, 2005.

[6] Gerd Behrmann, Johan Bengtsson, Alexandre David, Kim G. Larsen, Paul Pettersson, and Wang Yi, *UPPAAL implementation secrets*, Proc. of 7th International Symposium on Formal Techniques in Rea-Time and Fault Tolerant Systems, 2002.

[7] H.C. Bohnenkamp, H. Hermanns, R. Klaren, A. Mader, and Y.S. Usenko, *Synthesis and stochastic assessment of schedules for lacquer production*, Proc. QEST'04, LNCS, sep 2004, To appear.

[8] M. Bozga and O. Maler, *Timed automata approach for the Axxom case study*, Tech. report, Verimag, 2003.

[9] Sebastian Engell and Sebastian Panek, *Mathematical model formulation for the Axxom case study*, Technical Report AMETIST, May 2003.

[10] S. Loeschmann and D. Ludewig, *Case study 4: Detailed description of the model of a lacquer production*, 2002, Deliverable 3.4.1 from the IST project AMETIST.

[11] A. Mader, *Deriving schedules for the Cybernetix case study*, 2003.

[12] ———, *Towards modelling a value chain management example with Uppaal - Ametist case study 4*, 2003.

[13] A. Mader, G. Behrmann, and M. Hendriks, *Axxom case study: Modelling and schedule synthesis. Technical Report, AMETIST*, 2004.

[14] Alan S. Manne, *On the job-shop scheduling problem*, Operations Research **8** (1960), no. 2, 219–223.

[15] S. Panek, S. Engell, and C. Lessner, *Scheduling of a pipeless multi-product batch plant using mixed-integer programming combined with heuristics*, Proc. European Symposium on Computer Aided Process Engineering, ESCAPE 15, 2005, pp. 1033–1038.

[16] S. Panek, O. Stursberg, and S. Engell, *Job-shop scheduling by combining reachability analysis with linear programming*, Proc. 7th Int. Workshop on Discrete Event Systems, 2004, pp. 199–204.

[17] ———, *Optimization of timed automata models using mixed-integer programming*, Formal Modeling And Analysis of Timed Systems, LNCS, vol. 2791, Springer, 2004, pp. 73–87.

[18] S. Panek, O. Stursberg, and S. Engell, *Efficient synthesis of production schedules by optimization of timed automata*, Control Engineering Practice (2005), submitted.

[19] Sebastian Panek and Sebastian Engell, *Value chain optimisation / improvements in the solution by mathematical programming*, Technical Report AMETIST, University of Dortmund, May 2004, Case study 4, deliverable 3.4.3.