SCHEDULING LACQUER PRODUCTION BY REACHABILITY ANALYSIS - A CASE STUDY [1]

Gerd Behrmann * **Ed Brinksma** **
Martijn Hendriks *** **Angelika Mader** **

* *Aalborg University, Denmark*
** *University of Twente, The Netherlands*
*** *University of Nijmegen, The Netherlands*

Abstract: In this paper we describe a case study on lacquer production scheduling that was performed in the European IST-project AMETIST and was provided by one of the industrial partners. The approach is to derive schedules by means of reachability analysis: with this technique the search mechanism of model checkers, in our case here UPPAAL, is used to find feasible or optimal schedules. The advantage of this approach is that the expressiveness of timed automata allows to model scheduling problems of different kinds, unlike many classical approaches, and the problem class is robust against changes in the parameter setting. To fight the typical state space explosion problem a number of standard heuristics have to be used. We discuss the difficulties when modelling an industrial case of this kind, describe the experiments we performed, the heuristics used, and the techniques applied to allow to optimize costs (storage costs, delay costs, etc.) while searching for schedules.

Keywords: scheduling, model checking, cost optimization, industrial case study

1. INTRODUCTION

Scheduling algorithms play an important role in many embedded systems with real-time characteristics. They occur both as part of the embedded computational system, where limited resources (processor capacity, memory, i/o ports) must be shared by different processes that must meet real-time response requirements of the environment, and as part of the embedding environment, where external resources are managed by the embedded software (e.g. process control, production planning, manufacturing, etc.).

Scheduling theory is a well-established branch of operations research, and has produced a wealth of theory and techniques that can be used to solve many practical problems, such as real-time problems in operating systems, distributed systems, process control, etc. (Pinedo and Chao, 1999).

In spite of this success, in the last few years alternative approaches to scheduling synthesis have been proposed based on the application of reachability analysis. The basic idea is to use the search mechanisms of model checkers for finding schedules. The main advantage of this approach is that automata form a rich class of models and scheduling problems with variations in parameter settings can easily be modelled.

The case study of this paper is one of the four industrial case studies of the European IST project AMETIST, that focusses of the application of advanced formal methods for the modelling and analysis of complex, distributed real-time sys-

[1] Work supported by IST-2001-35304 project AMETIST (Advanced Methods for Timed Systems) `ametist.cs.utwente.nl`

tems, with dynamic resource allocation as one of its special topics. The application of timed reachability analysis to this problem is of the main approaches of the project. Technical material related to this case study, and different approaches to its solution can be retrieved from the AMETIST website (AMETIST, n.d.).

The contents of the remainder of this paper is organized as follows. The principles of the erivation of schedules by reachability analysis is sketched in section 2. Section 3 contains a description of the case study. Modelling issues and the use of heuristics are discussed in section 4 and 5. An extension of the case study deals with a cost-optimization problem rather than a feasibility problem and is presented in section 6. The results of our model checking experiments are collected and discussed in section 7. Section 8 evaluates the model checking approach to the case study and concludes the paper.

2. SCHEDULING SYNTHESIS BASED ON TIMED AUTOMATA MODEL CHECKING.

The synthesis of schedules can be seen as a special case of control synthesis (Maler *et al.*, 1995). It was first introduced by (Fehnker, 1999), and by (Abdeddaïm and Maler, 2001).

In general, a model class suitable for real-time control synthesis must provide the possibility to represent system events as well as timing information. The underlying framework used in this paper is the one of timed automata as introduced in (Alur and Dill, 1994) and implemented in the model checker Uppaal. Timed automata extend the traditional model of automata with clock variables whose values increase at the rate of the progress of time. Clocks can be reset and used as guards for transitions, as well as in state invariants. In general, timed automata models have an infinite state space. The region automaton construction (Alur and Dill, 1994), however, shows that this infinite state space can be mapped to an automaton with a finite number of equivalence classes (regions) as states. Finite-state model checking techniques can be applied to the reduced, finite region automaton.

Scheduling synthesis based on timed automata makes use of the search strategies of model checking. First, a model of the overall, uncontrolled system behaviour has to be constructed, which in our case consists of all possible production steps (of all orders) possible at every moment. Feasibility is formulated as a real-time property ("The production is finished by Friday evening"). The model checker searches the reachable state space for a state where this property holds. If it has found one, it provides a diagnostic trace. The diagnostic trace contains a sequence of actions and delays from the initial state to the state found. The start of a processing step is encoded as an action and can be found in the diagnostic trace together with the timing information. This suffices to extract a feasible schedule from a diagnostic trace.

The advantage of this approach is its robustness against changes in the setting of parameters, as timed automata provide a very general model class. The disadvantage lies in the well-known state-space explosion problem. For interesting cases the model checking approach as described above does not terminate. The way out is to add heuristics, or features of schedules, that reduce the search space to a size that cen be traversed more easily. In section 5 we discuss the heuristics used for the case study presented here.

3. PROBLEM DESCRIPTION.

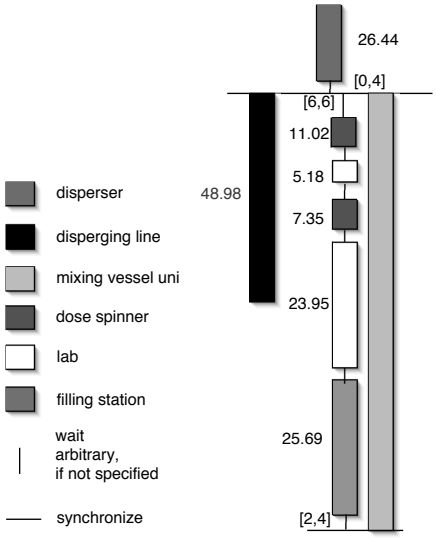
The case is about lacquer production scheduling. Lacquers come in three types and each type is produced following a different recipe. In the recipes the order of production steps is specified, with different conditions relating start and/or end times of subsequent production steps. Furthermore, for each processing step the necessary resources and the processing times are given.

Examples of resources are dose spinners, mixing vessels, predispersers, filling lines, etc. The resources have to be shared by different processes and this is the source of the scheduling problem. There is a set of orders, each order specifying the type, amount, earliest start date and due date. The original version of the case study deals with 29 orders for a planning period of 2 months (later extended to a case involving 73 orders).

The processing times come in two versions. The "pure" version indicates how long a processing step takes using a resource. Possible machine breakdown is encoded in an performance factor and the processing times are extended accordingly, i.e. if a machine works 85 percent of the time the original processing time p extends to $p*1/0.85$. The same holds for an availability factor indicating the restriction of working hours. We will deal with both, pure and extended processing times. The basic scheduling problem is to find schedules such that each order is finished before its due date.

In a further extension we consider also different costs, such as storage costs (for orders finishing too early), delay costs (for delivery after the due date), and costs that are caused by colour changes on resources that have to be cleaned. There,

Fig. 1. A graphical representation of a recipe for uni lacquers



the problem is to find cost-optimal (or close to optimal) solutions.

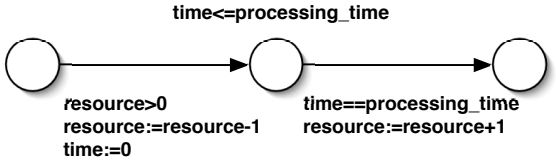
4. MODELLING METHOD.

A substantial amount of the time spent on the case study went into the modelling activities. The most difficult part here was the information transfer from the industrial partner to the academic partners. In the first place, there was a language problem regarding the domain specific interpretation of terminology. For this purpose we compiled an initial dictionary in which relevant terms used are explained in natural language. This dictionary served as an agreement with the industrial partner on the main, basic facts. In the second place, there was a documentation problem, regarding the (implicit) knowledge that always exists beyond any written specification. Explanations from the dictionary were used to put the documentation, mainly consisting of tables, in the intended context.

Another difficulty was caused by the format that the industrial partner used for the recipies, which was neither standard, nor intuitive. A better (from the computer science perspective, at least) representation had to be divised (see figure 1). This new notation also helped to detect other gaps in the case description.

The lacquer production case is very similar to the job shop scheduling problem, involving just a few additional timing constraints, and the basic modelling by timed automata roughly follows (Abdeddaïm and Maler, 2001). Each processing step can be mapped to a sequence of three locations in a timed automaton (fragment), see figure 2, where the transition between the first two locations claims the resource, the second location represents the processing period, and the transition to the last location frees the resource. The sequential and interleaved composition of the

Fig. 2. A single processing step modelled as timed automaton fragment



automaton fragments follows the descriptions and timing restrictions in the recipes.

To reduce the search space a number of heuristics were used to improve the model. In a later version also costs were added for storage, delay or colour change.

For each recipe there is a timed automaton (template) with free parameters for earliest start date and due date. Resources are modelled as counters, and only for the version where costs are considered, the filling stations are represented by their own timed automata. Colour change causes costs at filling stations, which makes it necessary to distinguish the filling stations and equip them with a memory for the last colour processed. There are altogether 29 (resp. 73) instantiations of the recipe automata with the example data for orders. The instantiated automata (and the filling station automata) in parallel composition form the system model.

When looking for feasible schedules we checked the reachability property "all orders (automata representing an order) reach their final state", where a guard in the model only allowed to enter the final state if the due date has not passed already. For the cost-optimal schedules we checked the same property without restricting the accessability of the final states, imposing a penalty for late delivery instead.

5. HEURISTICS.

The heuristics we used are more or less standard in operations research. They are not specific for this case study. The modelling of these heuristics can be seen as standard patterns that can be reused for similar cases.

Each heuristics reduces the search space. We distinguish two kinds: "nice" heuristics, where we know that for each good schedule that was pruned away (by search space reduction) there is a schedule in the remaining search space that is at least

as good. The other heuristics follow a "cut-and-pray" strategy: the search space is reduced and we hope that we find good schedules in the remaining search tree.

Below we discuss each of the heuristics we used.

Non-overtaking. This heuristic is applied within each group of orders following the same recipe. It says, that an order started earlier also will get critical resources earlier than an order started later. As the orders follow identical recipies this obviously is also a "nice" heuristic.

Non-laziness. In operations research non-lazy schedules are called active. The following behaviour is excluded: a process needs a resource that is available, but it does not take the resource. Instead, the resource *remains unused*, no other process takes it. Then, after a period of waiting the process decides to take the resource. (And we regard this waiting time as wasted, which is only true if there are no timing requirements for starting moments of subsequent processes.) This is a "nice" heuristic.

Greediness. This is a "cut-and-pray" heuristic. If there is a process step that needs a resource that is available, then the process step claims this resource immediately. By this it excludes possibly better schedules where some other (more important, because closer to deadline) process would claim the same resource shortly later. Note, that greediness is stronger than non-laziness, i.e. every greedy schedule is also a non-lazy one.

Reducing active orders. When not restricting the number of active orders (i.e. the orders that are processed at a certain moment), it often happens that many processes fight for the same resources, and block other resources while they wait. In our example the dose spinners (2 instances of these available) have to be used by each process twice, which makes them the most critical resource. Restricting the overall number of active orders avoids analysis of behaviour that is likely to be ineffective. This heuristic was very powerful, but belongs to the "cut-and-pray" type.

6. ADDING COSTS AND OTHER CONSTRAINTS.

Using standard Uppaal we had initially approximated some constraints to simplify the problem. In this section we discuss the extension of the model to cope with the full constraints. We begin with an informal explanation of these constraints.

Setup times and costs. The filling lines must be cleaned between two consecutive orders if those orders are not of the same type. Thus, additional cleaning time (5 – 20 hours) is needed and there is a certain cost involved with cleaning.

Delay and storage costs. The happiness of a customer decreases linearly with the lateness of his order. Thus, each order has a delay cost, which is a "penalty" measured in euros per minute. Similarly, if an order is finished too early, then it has to be stored and this also costs a certain amount of euros per minute. In the initial problem, the costs are approximated by requiring that every order must be finished before its deadline. A more refined cost model enables us to prefer an order that is five minutes late above an order that is weeks early.

Working hours. The lacquer production is goverened by personnel that works in two or three shifts, depending on the machine they operate. Furthermore, the production is interrupted in weekends. Note that this constrained is approximated in the initial problem by the *availability factor* of machines. Another complicating factor is that some production steps may only be interrupted for 12 hours.

The above constrainst were addressed in the following ways:

- Setup times and costs pose no problems. Instead of modelling the filling lines by an integer variable, they are now each modeled by an automaton that keeps track of the type of the order that has last been processed by it.
- There is a Uppaal version for *linearly priced timed automata* that enables us to model delay and storage costs in a natural way (Larsen *et al.*, 2001). It allows the represntation of costs as affine functions of the cloch variables.
- Modelling the working hours proved to be more involved. A separate automaton was added that computes the *effective* processing time e, given the current time and the net processing time c. For instance, if the current time and c are such that the processing must be interrupted, then $e = c + B$, where B equals the length of the interruption. The additional automaton is rather big and laborious to produce, but quite logical in structure.

7. MODEL CHECKING EXPERIMENTS.

In table 1 we collected models and model checking experiments for the feasability analysis and schedule synthesis. The results were obtained using Uppaal 3.4.6 on a laptop with an AMD Duron processor of 1GHz and with 512MB memory, running under Linux Red Hat 9.0. In table 1 the "-" for termination time expresses that the search was stopped after 1 minute. In the first case (29

jobs, no heuristics) we stopped the search after 10 minutes. The results nn table 1 show that for

Table 1. Characteristics of models and experiments

number of jobs	extended processing times	heuristics	number of active orders	termination time
29	no	-	-	-
29	no	nl	-	1s
29	no	nl no	-	1s
29	no	g	-	<1s
29	no	g no	-	<1s
29	yes	nl	-	2s
29	yes	nl no	-	1.1s
29	yes	g	-	<1s
29	yes	g no	-	1s
73	no	-	-	-
73	no	nl	-	-
73	no	nl no	-	-
73	no	nl no	3	7s
73	no	nl no	4	27s
73	no	g	-	-
73	no	g no	-	53s
73	no	g no	4	7s
73	no	g no	3	5s
73	yes	nl	-	-
73	yes	nl no	-	-
73	yes	nl no	3	-
73	yes	nl no	4	3s
73	yes	g	-	-
73	yes	g no	-	-
73	yes	g no	3	-
73	yes	g no	4	3s
73	yes	g no	5	5s

heuristics: g:greedy, nl:nonlazy, no:non-overtaking

the case of 29 jobs non-laziness is sufficient as only strategy, for extended processing times non-overtaking has some small additional effect.

The picture changes when we go to 73 orders. Greediness as the more aggressive heuristic gives no (fast) results, but together with non-overtaking the search terminates within a minute. With additional restriction of the active orders the results come much faster. For non-laziness we get only (fast) results if non-overtaking and restriction of the active jobs is added. The experiments also show that the good upper bound for the number of active jobs can vary in different settings and can only determined during experimentation.

Experiments have been performed also for the extended version of the case study for 29 orders, where costs were introduced, using the UPPAAL version for *linearly priced timed automata*. Again, schedules such that each order is finished before its deadline can be found easily. The schedules found have a cost that is equivalent to the cost of every order being finished 30 hours too early. Due to the enormous size of the state space, however, we are not able to tell whether this is the best schedule.

Introducing working hours makes the problem significantly more complex. Still, we derived schedules, but with very high costs, roughly equivalent to a scenario where each order is finished 10 days after its due date. As the heuristics above are not applicable in this case we are looking for suitable heuristics, which is ongoing work.

8. STOCHASTIC ANALYSIS.

As explained earlier, so-called performance and availability factors are used to indicate the percentage of time that a resource is unavailable. The way in which the industrial partner deals with this information is that the processing time on each resource is extended by the corresponding factor. E.g. if a machine only is available half of the time, the processing time for each processing step using this resource is doubled. Schedules are derived assuming that the process durations are extended in this way. This raised the question on the interpretation of the schedules derived with the extended processing times. Stochastic analysis (Bohnenkamp *et al.*, 2004) showed that the schedules derived in this way have less chance to reach the due dates than schedules without extended times. The interpretation roughly is as follows: if we reserve time for break-down when a resource is actually available, this time is simply wasted. Later, when the resource really breaks down, there will be too little time left to reach the due date. A conclusion is that extending processing times may give a useful indication how many orders can probably be done within a long time interval, say a few months, but it does not help for daily fine-tuned scheduling.

9. EVALUATION AND CONCLUSION.

We showed that feasible schedules for a lacquer production case can be derived doing real-time reachability analysis with the timed automata model checker UPPAAL. We could treat 29 orders, and an extension to 73 orders did not significantly increase the computation times. In both cases it took about 1-2 seconds on a PC. To deal with the full set of constraints we had to introduce costs into the model, that came as setup-costs for filling stations, storage costs (for too early finished orders), and delay costs (for orders that are finished too late). In doing so, the problem

was transformed into a cost-optimization problem. This was treated by a cost-optimal version of UPPAAL. A further extension of the model consisted in adding working-hours constraints, which increased the size complexity of the model significantly. Yet, also for this case schedules could be derived using the cost-optimal version of UPPAAL.

On the one hand, it is clear that this application of model checking techniques is not (yet) push-button technology: to obtain results models have to be constructed with care, and the right heuristics have to be identified. On the other hand, it is to be expected that many production scheduling problems will have similar ingredients and that modelling techniques and patterns for typical plant processes and heuristics can be reused. Further experiments have to be carried out to identify a useful collections of these.

Of course, there still are a number of open issues. One important question is to which extend the approach scales up. We treated 29 orders in the first experiments, and 73 in further experiments, which did not require significantly more time. We believe that this is due to the fact that the heuristic that limits the number of active orders gives a decomposition of the problem which limits complexity for larger problems. Currently, we are working on cases involving several thousands of orders. One of the problems here is even to construct a representative case of this size.

This case study also raised a number of pragmatic questions concerning the modelling. It turned out to be quite difficult to obtain al the relevant information from the industrial partner. In spite of all our efforts in creating a dictionary and better graphical representations, the models had to be changed substantially after several months, as it turned out that the initially provided information was inaccurate (one requirement was, in fact, over-specified). This experience suggests that beyond a dictionary, there should have been some joint activity of both parties to certify the informal explanations. Altogether, the information transfer was time consuming and inefficient, and guidelines to support this process would have been helpful.

Another, related aspect is that the problem description of the case study provider was strongly influenced by the capabilities of their own planning tool. For example, very high delay costs are a specific way to make it treat due dates as hard deadlines. This raises the question to what extent we were modelling the original problem, or remodelling the model of the industrial partner.

The use of the performance and availability factors also leads to questions of interpretation. Extending the processing times by these factors can be used to analyse how many orders could be treated at all over longer periods of time. Given such an objective it does not seem very useful to include penalties for such things as changing colours that are insignificant compared to the costs of missing deadlines, as is done in the model.

The stochastic analysis showed that using performance and availability factors for concrete scheduling for short periods increases the probability to miss deadlines. It is unclear what modelling assumptions are best suitable for the derivation of such schedules, where storage costs have to be minimized and delay costs be avoided.

Summarizing, we can say the the application of model checking techniques for production scheduling is promising, but that further work on modelling methods, reusability of modelling patterns, identification and evaluation of heuristics, all in the context of case studies of greater orders of magnitude, is needed.

REFERENCES

Abdeddaïm, Y. and O. Maler (2001). Job-Shop Scheduling using Timed Automata. In: *13th Conference on Computer Aided Verification (CAV'01)*.

Alur, R. and D.L. Dill (1994). A theory of timed automata. *Theoretical Computer Science* (138), 183–335.

AMETIST (n.d.). (Advanced Methods for Timed Systems), IST project IST-2001-35304. `http://ametist.cs.utwente.nl/`.

Bohnenkamp, H. C., H. Hermanns, R. Klaren, A. Mader and Y. S. Usenko (2004). Synthesis and stochastic assessment of schedules for lacquer production. In: *1st Int. Conf. on Quantitative Evaluation of Systems (QEST)*. IEEE Computer Society Press, Los Alamitos, California. Enschede, The netherlands. p. to appear.

Fehnker, Ansgar (1999). Scheduling a Steel Plant with Timed Automata. In: *Sixth International Conference on Real-Time Computing Systems and Applications (RTCSA'99)*. IEEE Computer Society Press.

Larsen, K. G., G. Behrmann, E. Brinksma, A. Fehnker, T. Hune, P. Pettersson and J. Romijn (2001). As cheap as possible: Efficient cost-optimal reachability for priced timed automata. In: *CAV 2001* (G. Berry, H. Comon and A. Finkel, Eds.). number 2102 In: *LNCS*. Springer–Verlag. pp. 493–505.

Maler, Oded, Amir Pnueli and Joseph Sifakis (1995). On the synthesis of discrete controllers for timed systems. In: *Proceedings of STACS'95*. Vol. 900 of *LNCS*. Springer.

Pinedo, Michael and Xiuli Chao (1999). *Operations Scheduling with Applications in Manufacturing Systems*. McGraw-Hill.