Analysis and Tools: Scheduling and Planning Algorithms
A revised Version

Oded Maler

VERIMAG

May 20, 2006

AMETIST DELIVERABLE 2.2.2

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/`

Introduction: Planning and Scheduling in AMETIST

Scheduling and planning are not considered to be typical application domains of formal verification and fall more into the realm of disciplines such as AI and operations research as well as to specific application domains. The more general term, planning, is viewed within the AI literature as the process of finding a sequence of actions that brings a system from a given state to a desired goal state. In the AI tradition, unfortunately, the dynamics of the system is defined in a more implicit fashion using logical methods that sometime does not give a clear view of the dynamics of the system, as does automaton-based methods. A state of the system is viewed as a collection of "facts" in a data base that describe the current state of affairs. Actions are characterized by their preconditions and by their side effects, that is, the facts that they add to or remove from from the data base. This is the classical STRIPS approach whose origin is in solving puzzles in the "blocks world", a virtual world with facts like "the red block is on the blue block" and actions that put blocks on each other or remove top blocks. The AI planning community is centered around the annual conference ICAPS and holds planning competitions on problems coming from different domains. Space application are a popular source for planning problems as the distance between the earthly operator and the satellites, telescopes, spaceships and robots to be controlled is sometimes too large for tele-operations and autonomous solutions are sought.

In the OR tradition, planning refers more to some kind of "strategic" and "tactical" decision making, for example, choosing a machine renewal policy that optimizes amortized cost or an inventory handling policy. Scheduling can be viewed as a special case of planning where the action repertoire is more restricted and the decisions of the planner are about allocating bounded resources to tasks that may compete for them. The OR planning and scheduling problems tend to be more *uniform* in structure and hence can be described and solved more naturally using traditional tools of applied mathematics.[1]

When we restrict ourselves to a finite horizon, various planning and scheduling problem can be transformed into *constrained optimization* problems over a finite number of decision variables. The simpler planning and scheduling problems coming from OR translate into a more stylized format of constraints that can be solved sometimes even using simple linear programming. However most problems have a discrete decision component (for example priority between two jobs on a machine in job-shop scheduling) that renders the solution space highly non convex. Such problems are solved sometime using methods such as MILP which are more oriented toward numerical rather than logical problems. More complex planning and scheduling problems, including those treated by the AI community, have constraints with a more dominant logical part and are typically solved using constraint propagation techniques originating from constraint logic programming, or more recently by enriched SAT solvers. The alternative approach, which is the main approach pursued within the AMETIST project, consists of searching the space of paths in the dynamic transition graph implied by the system description.

Both approaches need to be upgraded in order to treat situations where the future outcome does not depend exclusively on the actions of the planner/scheduler but also on some external uncontrolled actions (the "environment" in the verification jargon. "disturbances" in the control sense). In the AI planning literature this is called *reactive planning*, and has been, in fact, subject to intense research in the context of game-playing programs. Problem with uncertainty ("adversary") are solved using algorithms that search the game graph to find a "winning" strategy. Since typically the space of possible executions grows exponentially with the depth of the game tree, clever heuristic methods are

[1] In fact, it is a chicken and egg problem, as the availability of mathematical tools often influences our choice of models and problems.

needed to solve large problems. In the case of chess, special parallel hardware was needed to beat the human world champion. On the OR side, where problem are reduced to static optimization, it is not always easy to deal with alternating min-max non-determinism and the uncertainty is often associated with probabilities, sometimes for historical reasons.

This report summarizes the progress in understanding, development and implementation of planning and scheduling *algorithms*, as well as additional efforts to make connections with other disciplines and communities that occupy themselves with these issues. To avoid redundancy we will *not* elaborate here much on the work reported under the case studies as well as most of the work reported in the 2nd-year deliverable 2.2.1 "control synthesis algorithms" because within the AMETIST approach controller synthesis and planning/scheduling are almost synonyms. Note that this report is primarily about new *algorithms* and not about the application of old algorithms to new problems.

The highlights of the AMETIST work on planning and scheduling algorithms can be grouped into the following sub topics:

1. *Priced timed automata*: An extension of the timed automaton models to express cost which turns out to be decidable, including its integration in UppAaal.

2. *Scheduling under uncertainty*: New models and algorithms for synthesizing scheduling strategies under uncertainty.

3. *Timed automata and constraints*: Combination of idea from optimization and satisfiability with timed automata.

4. *Additional work*: including stochastic scheduling of batch plants and synthesizing schedulers for embedded systems.

5. *Bridge building*: dissemination and reverse-dissemination toward relevant communities, mostly the AI planning community.

Priced Timed Automata

One of the major achievements of AMETIST is the development of the theoretical results, algorithms and a tool for reasoning about *priced timed automata* [13, 9, 15, 16, 14, 10, 11]. Before AMETIST the only cost that could be associated with a run of the automaton was the elapsed time and impulse costs. Priced timed automata extend timed automata by allowing rate costs, i.e. accumulated cost as a function of time. Rate costs are defined for each location of the automaton and may vary between locations. Decidability follows from the fact that, albeit being a hybrid automaton, transition guards are not influenced by the cost of a run and a finite quotient property still holds (although more complex than for plain timed automata).

During the project, important improvements have been made to priced zones, a critical data structures for the analytical analysis of priced timed automata. A zone is a convex set of valuations of the clocks of the timed automaton. A priced zone adds an affine function, which for each clock valuation in the set gives the cost of a run ending in the valuation. The state space of a priced timed automaton can conveniently be represented by a number of priced zones (together with information about other state variables). One particularly time consuming operation is to find the infimum cost of a priced zone. The discovery of a reduction from the problem of finding the infimum cost of a priced zone to the min-cost flow problem led to a performance improvement of several orders for magnitude for computing the infimum [34]. This change alone has improved the overall speed of the prototype implementation of

UPPAAL CORA by one order of magnitude. Without these and other less fundamental (engineering type) improvements to the prototype analysis tool UPPAAL CORA, handling the case studies of the AMETIST project would not have been possible.

Within the project, a graphical front end for priced timed automata has been developed based on the UPPAAL graphical front end. This front end has eased the development of a model for the extended Axxom case studies and has enabled the partners to evaluate various modelling techniques for the case study. Furthermore, the tool has matured within the lifetime of the project to a state were it was releasable to the public. Obviously, UPPAAL CORA does not get close to the popularity of UPPAAL, but approximately 10 to 15 downloads a month is still respectable for an academic prototype.

On the theoretical side, the results were extended to the problem of finding the optimal conditional run for multi-priced timed automata. More precisely, the problem of determining the minimal cost of reaching a given target state, with respect to some primary cost variable, while respecting upper bound constraints on the remaining (secondary) cost variables [28]. However, prototype implementations could not be me completed before the project was concluded.

Algorithms for Scheduling under Uncertainty

Temporal Uncertainty

Zone manipulation penalizes the performance of timed automata significantly and this is not going to improve with the priced zones used for priced timed automata. The approach pursued in [4, 6, 1, 3, 2] attempts to avoid as much as possible computation with zones by restricting the search to a finite subset of automaton runs, namely the "non lazy" runs. These runs correspond to notions such as active and semi-active schedules in the scheduling literature and they are guaranteed to include the optimum in certain deterministic problems (job-shop and task-graph included). Manipulation of n-dimensional vectors instead of $n \times n$ matrices and all the overhead associated with their manipulation has led to an efficient implementation.

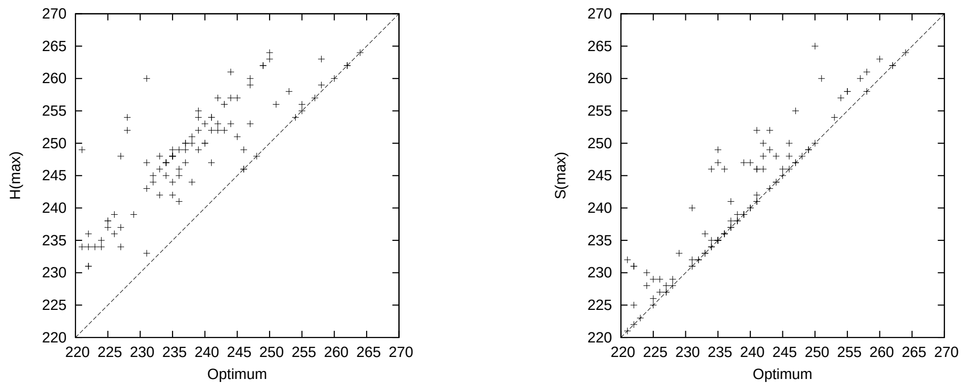
The major justification for this restriction of the search space is due to fact that in the modeling of deterministic scheduling problems by timed automata, all the non-determinism is related to the the decisions of the scheduler. However when it comes to scheduling problems admitting uncertainty, the situation is different. The work in [1, 2] explored the first type of such uncertainty, namely in task durations which was considered to bounded within an interval.[2] For this type of problems, focusing on a discrete subset of the feasible schedules does not make immediate sense because these "decisions" are not taken by the scheduler. Hence, the simplest way to "cover" all choices of the adversary was to use a backward dynamic programming algorithm on zones, developed and implemented within AMETIST [1, 2]. The algorithm, implemented using the zone library of IF, is proved to produce *offline* a scheduling strategies which are equivalent to solving a "nominal" deterministic problem and then re-scheduling *online* each time a deviation from the nominal behavior is observed.

To illustrate the quality of the strategies algorithm, we have compared it with two alternatives:

1. Static worst-case solution: solve the deterministic problem that corresponds to the worst-case, i.e. taking the upper bound for each interval;

2. Hole filling: compute the worst-case schedule but allow shifting tasks forward when their predecessors terminate earlier, without modifying inter-task priorities.

[2]It is worth mentioning that we are talking here about "closed" uncertainty in the terminology of Deliverable 2.2, and not about open-ended uncertainty.



Hole filling   Dynamic programming on TA

Figure 1: The quality of schedules produced by the hole-filling (H) strategy and the optimal strategy computed using dynamic programming on zones (S). Each instance is drawn as a point (x, y) on the plane with x indicating the length of the optimal clairvoyant schedule and y — the length of the schedule produced by the corresponding strategy.

Consider the following uncertain job-shop problem:

$$\begin{array}{ll} J^1: & (m_2, 34), (m_4, [21, 54]), (m_3, 74), (m_5, [6, 26]), (m_1, 5)(m_6, 43) \\ J^2: & (m_2, 24), (m_5, [13, 28]), (m_1, 53), (m_3, 8), (m_6, [16, 23]), (m_4, 45) \\ J^3: & (m_6, [35, 75]), (m_5, 14), (m_3, [\,8, 15]), (m_1, 31), (m_2, 24), (m_4, 6) \\ J^4: & (m_1, [12, 42]), (m_3, [25, 32]), (m_6, 15), (m_4, 42), (m_5, 62), (m_2, 18) \end{array}$$

The static worst-case optimal schedule for this problem is 268. We generated random examples with durations drawn uniformly from the intervals and compared both strategies with the static worst-case scheduler and with an optimal clairvoyant scheduler that knows the durations in in advance. It turns out that the static schedule is, in the average, longer than the optimum by 12.54%. The hole filling strategy deviates from the optimum by 4.44% while the strategy computed by the new algorithm produces schedules that are longer than the (clairvoyant) optimum by 1.14%! (see Figure 1). Albeit these impressive results let us note the question of scaling-up the results to larger problems remains. Currently the computation of a strategy for the 4×6 example takes around 10 minutes and there is not much hope to go significantly beyond this size using exhaustive backward reachability on zones.

An alternative forward algorithm has been proposed in [8] based on a minimized finite-state abstraction of the automaton. Recently this idea has been improved in [20] to allow some of the minimization to be done on-the-fly, without necessarily generating the whole state space.

Conditional Uncertainty

The next type of uncertainty that has been approached was that of conditional (discrete) uncertainty. The first step was to extend the notion of a precedence graph to include conditional dependencies between tasks, where the the outcome of one task may determine whether another task should be

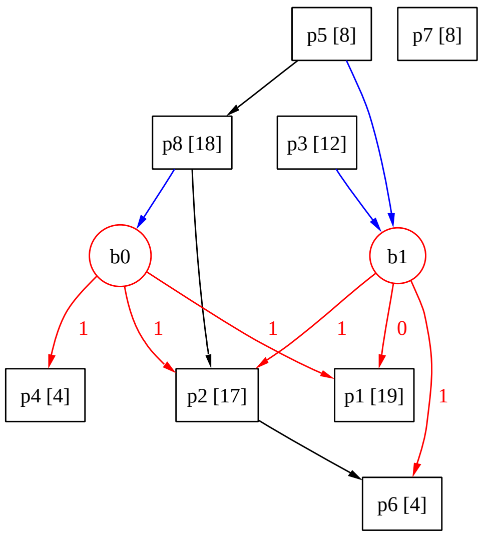


Figure 2: A conditional precedence graph. Numbers in square brackets indicate durations. Edge on labels from Boolean to ordinary tasks denote positive or negative influence, for example the activation condition for p_1 is $b_0 \wedge \neg b_1$.

executed. This is a natural model for scheduling programs with *if-then-else* branches on parallel machines, or for modeling faults. The *conditional precedence graph* (CPG) introduced in [17], uses a second type of tasks (Boolean tasks) which are preceded by the task that determine their value, and which participate in the *activation conditions* of other tasks. A CPG with 8 tasks and 2 Boolean tasks is depicted in Figure 2. In this example, the Boolean task b_1 is evaluated after the termination of both p_3 and p_5 (for example a comparison of values computed by these tasks) and participates in the activation conditions for p_2, p_1 and p_6.

The conditional precedence graph is transformed into a *timed game automaton* where the adversary chooses the task outcomes, that is, the values for certain Boolean variable that may appear in the activation condition of further tasks. As in the deterministic case, focusing on non-lazy runs, the problem is then reduced to a shortest path problem in discrete weighted *game graphs*. This problem is, in turn, solved using some variants of heuristic depth-first min-max search with non-chronological backtracking. In terms of [performance and the results are rather promising. It is possible to find optimal solutions for problems with 20 tasks and few conditions, and sub-optimal (but good) solutions for problems with hundreds of tasks and up to 20 conditions. The optimal solution, in terms of a branching Gantt chart (produced automatically from the solution) is depicted in Figure 3.

Encouraged by the performance of depth-first search variants we tried to apply it to the problem of dense temporal uncertainty previously mentioned. We are exploring an approach based on discretizing the duration uncertainty intervals into finitely many possible durations for each task. This approach facilitates forward heuristic search but it suffers from two drawbacks: 1) The estimation of the cost of a strategy is not based on all possible adversary choices; 2) Likewise, since not all these choices were taken into account, the system may reach states for which a strategy has not been computed at all. However using the notion of an *approximate strategy* which takes the same decision as in the nearest point where a strategy is defined, we can produce sub-optimal strategies, with a significant increase in performance compared to zone-based methods. Preliminary experiments of this idea have

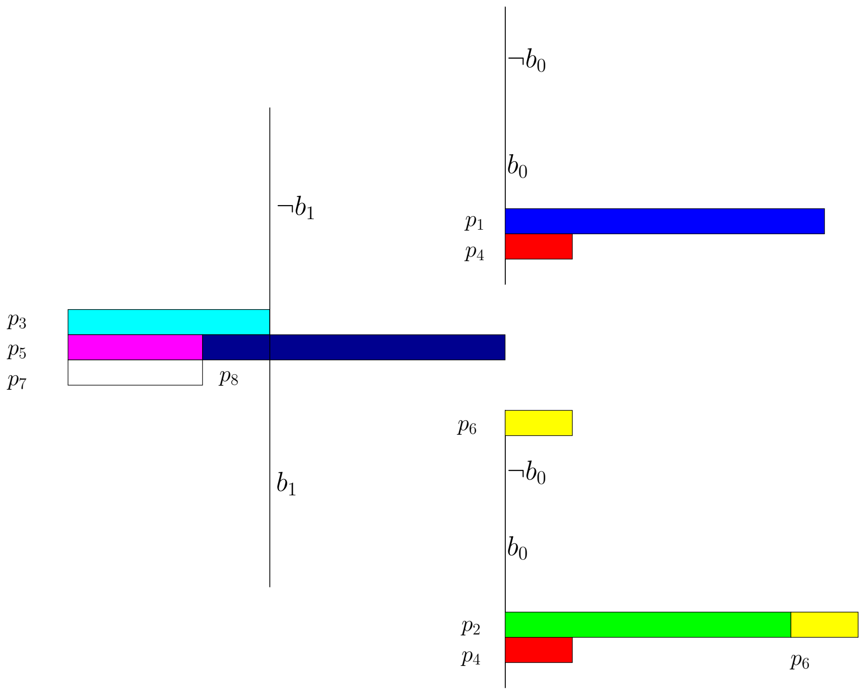


Figure 3: An optimal worst-case schedule for the conditional precedence graph of Figure 2. Vertical lines indicate moments in time when the value of Boolean tasks becomes known.

been conducted but no conclusive results have been reported to date. Another approach for restricting the space of searched trajectories using partial-order methods has been reported in [5].

Timed Automata and Constraints

Although optimization techniques differ from timed automata reachability algorithms, they share the same type of constraints needed for computation, namely difference constraints of the form $x - y < c$ that specify bounds on distance between events. Such constraints are prominent in the classical formulation of scheduling, and are also popular in the framework of *satisfaction modulo theories* (SMT) for combining SAT solvers with richer constraints. The advantage of such constraints is that checking whether there exist an assignment satisfying the a conjunction of such constraints can be resolved by detecting negative cycles in graphs[3] rather than full-fledged linear programming.

The work on constraints within AMETIST has focused on two directions. The work using the tool TAOPT [32, 31, 33], was based on using the off-the-shelf tool CPLEX for mixed integer-linear programming (MILP), and integrate its functionality in timed-automaton based scheduling. The relaxations used in MILP were used to obtain bounds on schedule durations, which were used as an estimation function to steer depth-first search. This work also took care of the upstream problem specification in the form of resource-task networks (RTN). Some of the experimental results confirm the opinion that relaxation-based methods, which work quite nicely for problems were the integer variables have a numerical significance (e.g. renewal problems) are less successful for scheduling problems where one has to deal with “real” Boolean variables. In terms of expressivity, one advantage of this approach

[3]The DBM representation is, in fact, an encoding of such a graph.

is in the ability to go beyond difference constraints.

The second constraint-related direction was to build dedicated solvers for difference constraints. The first solver that has been developed also within AMETIST was MX-Solver [29] who served as an entry point to the domain and its problems (interaction between the Boolean part and the constraints, handling of huge DBMs). Although its performance for solving the motivating problem, namely *bounded horizon* verification of arbitrary timed automata was deceiving (much behind standard TA reachability), it already was the best among several competing tools in solving job-shop problems. Two subsequent solvers DL-SAT [21] and `jat` [22] achieved an impressive performance with the latter being able to find the optimum for 20×10 problems taken from the job-shop benchmarks, and beat the best existing tools on these problems.[4]

Additional Work

We mention briefly other scheduling-related work, already mentioned in previous deliverables. The work described in [35, 24, 23, 36], considers the mid-term scheduling of a multi-product batch plant, where due to the long horizon, demands are not known but can be probabilistically represented. A two-stage stochastic integer programming model for the problem is introduced and solved by a scenario-decomposition method based on Lagrangian relaxation. Several heuristics and preprocessing for both the single- and the multi-scenario models were developed and tested.

The work described in [7, 37] examines with the role of timed automata schedulability in the context of embedded software. Optimal scheduling is only one part of the problem in these applications, and not always the most important one. Due to the component-based methodology that everybody attempts to achieve for the development of such systems, one has to face there traditional verification problems such as the absence of deadlocks and timelocks. As mentioned in deliverable 1.1, timeliness is more important for these applications than global optimality and, moreover, the resources available for the scheduler both in terms of computing the strategy and observing the state of the execution, hence the working solutions for some instances of embedded systems scheduling proposed in [26, 25, 19, 18] are more pragmatic and do not use the more costly algorithms and strategies of [1, 3]. The application of the behavioral formalism of *live sequence charts* (LSCs) to the scheduling problem is explored in [27].

Bridge-building Activities

Members of AMETIST made efforts encourage diffusion of ideas between the timed automaton community and other communities which are relevant to these problems, which include the AI planning community, operations research, constraint solving, control and embedded systems. The opening toward the planning community started with the translation from the planning specification language PDDL to UPPAAL, and continued by the presentation of AMETIST results by O. Maler and Y. Abdeddaim in affiliated workshops of ICAPS'02 in Toulouse, and a year later by the one-day tutorial in ICAPS'03 in Trento about timed automata and scheduling by O. Maler who also participated in a Dagstuhl seminar on planning and control synthesis which, unfortunately took place during the review and K. Larsen gave an invited talk at ICAPS'05 on real-time planning using timed automata. The work on the NASA K9 Rover case-study [12] has demonstrated the applicability of timed automata for space application, and led to a discussion with labs such as LAAS, Toulouse about closer

[4]The problem of bounded-horizon verification for TA, remains unimpressed by this tool, and this is considered a general feature of *asynchronous* systems.

collaboration between the AI planning crowd and the timed automaton community.[5]

References

[1] Y. Abdeddaïm, E. Asarin, and O. Maler. On optimal scheduling under uncertainty. In H. Gargamel and J. Hatcliff, editors, *Proc. TACAS*, volume 2619 of *LNCS*. Springer, 2003.

[2] Y. Abdeddaïm, E. Asarin, and O. Maler. Scheduling with timed automata. *Theoretical Computer Science*, 2005. to appear.

[3] Y. Abdeddaïm, A. Kerbaa, and O. Maler. Task graph scheduling using timed automata. In *FMPPTA*, 2003.

[4] Y. Abdeddaïm and O. Maler. Preemptive job-shop scheduling using stopwatchautomata. In J.-P. Katoen and P. Stevens, editors, *TACAS*, 2002.

[5] Y. Abdeddaïm and P. Niebert. On the use of partial order methods in scheduling. In *Ninth International Conference on Project Management and Scheduling (PMS 04)*, 2004. to be published as online abstract.

[6] Yasmina Abdeddaïm. *Scheduling with Timed Automata*. PhD thesis, INPG Grenoble, November 2002.

[7] K. Altisen, G. Goessler, and J. Sifakis. Scheduler modeling based on the controller synthesis paradigm. *Journal of Real-Time Systems, special issue on Control Approaches to Real-Time Computing*, 23:55–84, 2002.

[8] K. Altisen and S. Tripakis. Tools for controller synthesis of timed systems. In *RT-TOOLS*, 2002.

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

[10] G. Behrmann, K.G. Larsen, and J.I. Rasmussen. Optimal scheduling using priced timed automata. *Performance Evaluation Review, ACM Sigmetric*, 2005. To appear.

[11] G. Behrmann, K.G. Larsen, and J.I. Rasmussen. Priced timed automata: Algorithms and applications. In *Proceedings of FMCO'04*, Lecture Notes in Computer Science. Springer Verlag, 2005. To appear.

[12] S. Bensalem, M. Bozga, M. Krichen, and S. Tripakis. Testing conformance of real-time applications by automatic generation of observers. In *Runtime Verification (RV'04)*, 2004.

[13] P. Bouyer, F. Cassez, E. Fleury, and K. G. Larsen. Optimal strategies in priced timed game automata. BRICS Report Series RS-04-4, Basic Research In Computer Science, 2004.

[14] Patricia Bouyer, Ed Brinksma, and Kim G. Larsen. Staying alive as cheaply as possible. In Rajeev Alur and George J. Pappas, editors, *Proceedings of the 7th International Conference on Hybrid Systems: Computation and Control (HSCC'04)*, volume 2993 of *Lecture Notes in Computer Science*, pages 203–218, Philadelphia, Pennsylvania, USA, March 2004. Springer-Verlag.

[5]This is the subject of a new French project between VERIMAGand LAAS based on their IxTeT environment for plan generation.

[15] Patricia Bouyer, Franck Cassez, Emmanuel Fleury, and Kim G. Larsen. Optimal strategies in priced timed game automata. In Kamal Lodaya and Meena Mahajan, editors, *Proceedings of the 24th Conference on Fundations of Software Technology and Theoretical Computer Science (FSTTCS'04)*, volume 3328 of *Lecture Notes in Computer Science*, pages 148–160, Chennai, India, December 2004. Springer.

[16] Patricia Bouyer, Franck Cassez, Emmanuel Fleury, and Kim G. Larsen. Synthesis of optimal strategies using HyTech. In Luca De Alfaro, editor, *Proceedings of the Workshop on Games in Design and Verification (GDV'04)*, volume 119 of *Electronic Notes in Theoretical Computer Science*, pages 11–31, Boston, Massachusetts, USA, February 2005. Elsevier Science Publishers.

[17] M. Bozga, A. Kerbaa, and O. Maler. Optimal scheduling of acyclic branching programs on parallel machines. In *Real-Time Systems Symposium (RTSS)*, pages 208–217. IEEE Press, 2004.

[18] P. Caspi, A. Curic, A. Maignan, C. Sofronis, and S. Tripakis. Translating discrete-time Simulink to Lustre. In *Embedded Software (EMSOFT'03)*, volume 2855 of *LNCS*. Springer, 2003.

[19] P. Caspi, A. Curic, A. Maignan, C. Sofronis, S. Tripakis, and P. Niebert. From Simulink to SCADE/Lustre to TTA: a layered approach for distributed embedded applications. In *Languages, Compilers, and Tools for Embedded Systems (LCTES'03)*. ACM, 2003.

[20] F. Cassez, A. David, E. Fleury, K.G. Larsen, and D. Lime. Efficient on-the-fly algorithms for the analysis of timed games. In Luca de Alfaro Martin Abadi, editor, *To appear in Proceedings of CONCUR 2005*, Lecture Notes in Computer Science. Springer Verlag, 2005.

[21] S. Cotton, E. Asarin, O. Maler, and P. Niebert. Some progress in satisfiabilty checking for difference logic. In *FORMATS/FTRTFT'04*, number 3253 in LNCS, pages 263–276. Springer, 2004.

[22] Scott Cotton. DPLL and difference constraints. Master's thesis, Max-Planck Doctoral School Saarbrucken, June 2005. Verimag.

[23] S. Engell, A. Maerkert, G. Sand, and R. Schultz. Aggregated scheduling of a multiproduct batch plant by two-stage stochastic integer programming. *Optimization and Engineering*, 5:335–359, 2004.

[24] S. Engell and G. Sand. A two-stage stochastic integer programming approach to real-time scheduling. In I. E. Grossmann and C. M. McDonald, editors, *4th Int. Conf. on Foundations of Computer-Aided Process Operations*, pages 347–350, Austin, 2003.

[25] Ch. Kloukinas and S. Yovine. Synthesis of safe, qos extendible, application specific schedulers for heterogeneous real-time systems. In *Proceedings of '5th Euromicro Conference on Real-Time Systems (ECRTS'03)'*, Porto, Portugal, July 2003.

[26] Christos Kloukinas, Chaker Nakhli, and Sergio Yovine. A methodology and tool support for generating scheduled native code for real-time Java applications. In Rajeev Alur and Insup Lee, editors, *EMSOFT 2003*, volume 2855 of *Lecture Notes in Computer Science*, pages 274–289, Philadelphia, Pennsylvania, USA, October 2003. Springer-Verlag.

[27] Hillel Kugler and Gera Weiss. Planning a production line with LSCs. Research report, Weizmann, 2004.

[28] Kim G. Larsen and Jacob I. Rasmussen. Optimal conditional reachability for multi-priced timed automata. In *In Proceedings of FoSSACS 2005*, number 3441 in Lecture Notes in Computer Science, pages 234–249, 2005.

[29] Moez Mahfoudh. *On Satisfaiblity Checking for Difference Logic*. PhD thesis, UJF Grenoble, May 2003.

[30] O. Maler. On optimal and sub-optimal control in the presence of adversaries. In *Workshop on Discrete Event Systems (WODES)*, pages 1–12. IFAC, 2004. Invited talk.

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

[32] S. Panek, O. Stursberg, and S. Engell. Optimization of timed automata models using mixed-integer programming. In *Formal Modeling And Analysis of Timed Systems*, volume 2791 of *LNCS*, pages 73–87. Springer, 2004.

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

[34] J. Rasmussen, K. G. Larsen, and K. Subramani. Resource-optimal scheduling using priced timed automata. In *Proc. 10th Int. Conf. of Tools and Algorithms for the Construction and Analysis of Systems (TACAS'2004)*, volume 2988 of *Lecture Notes in Computer Science*, pages 220–235. Springer-Verlag, 2004.

[35] G. Sand and S. Engell. Aggregated batch scheduling in a feedback structure. In J. v. Schijndel and J. Grievink, editors, *European Symp. on Computer Aided Process Engineering-12*, volume 10 of *Computer-Aided Chemical Engineering*, pages 775–780. Elsevier Science, 2002.

[36] G. Sand and S. Engell. Risk conscious scheduling of batch processes. In *Proc. Computer-Aided Chemical Engineering*, pages 588–593, 2004.

[37] J. Sifakis, S. Tripakis, and S. Yovine. Building models of real-time systems from application software. In *Proceedings of the IEEE Special issue on modeling and design of embedded*, pages 91(1):100–111, January 2003.