2012 IEEE International Conference on Control Applications (CCA)
Part of 2012 IEEE Multi-Conference on Systems and Control
October 3-5, 2012. Dubrovnik, Croatia

Timed Automata Based Scheduling for a Miniature Pipeless Plant with Mobile Robots[*]

Christian Schoppmeyer, Martin Hüfner, Subanatarajan Subbiah, and Sebastian Engell

Abstract— **In this contribution we present a conceptual idea on modeling a scheduling problem in a miniature pipeless plant with mobile robots by timed automata (TA) and solving it using reachability analysis. Two TA-based tools, TAOpt and UPPAAL, are evaluated on the miniature pipeless plant, and on job shop benchmark instances from the OR literature. The comparative study shows that the proposed modeling approach handles the crucial constraints of the miniature pipeless plant in an effective and straightforward way.**

I. Introduction

Planning and scheduling of modern, multi-product batch plants are complex processes and require the use of intelligent decision-support systems. The efficiency of such flexible manufacturing environments can be increased by computing an optimized schedule for the job operations such that the scarce resources are utilized in an optimal fashion. In contrast to traditional chemical plants, where the equipment is connected by pipes, pipeless plants use mobile production vessels to transport the materials between the different processing stations. Hence, in addition to the problem of planning the operations the underlying problem of routing the mobile robots in an efficient way should be addressed.

The state-of-the-art approach in handling scheduling problems in manufacturing industries is to model the problem as MILP or MINLP and to solve them using commercial solvers implementing branch-and-bound and/or cutting-plane methods. For scheduling of pipeless plants the authors in [1] propose a simulation based optimization approach where a combination of evolutionary algorithm is used to compute and optimize the schedules and the feasibility of the schedule with the routing is validated using a simulation. The authors in [2] proposed a constraint satisfaction technique to solve the integrated scheduling and routing problem in pipeless plants with several layouts.

An alternative approach is to model the problem as timed automata (TA) and to solve it using the technique of cost-optimal reachability analysis, see [3] and [4]. The advantages of the TA-based approach are the intuitive graphical formalism to specify the problem, the modular modeling of the recipes and of the resources individually as sets of automata, and the use of efficient reachability algorithms.

Several TA-based scheduling tools have been developed and used to solve scheduling problems in the past: TAOpt, developed by the Process Dynamics and Operations Group at TU Dortmund, and UPPAAL, developed by Uppsala University and Aalborg University for simulation and model checking of discrete systems [5]. Although UPPAAL, in comparison to TAOpt, uses a more generic approach of modeling and model checking using reachability analysis, searching for time-optimal schedules using the built-in model checking engine is feasible which makes it a suitable candidate for comparative studies with TAOpt.

In this contribution, we discuss modeling and solving of job shop scheduling problems using timed automata and reachability analysis and propose a TA model for the scheduling problem in the miniature pipeless plant. Subsequently, the performances of UPPAAL and TAOpt for benchmark instances from OR literature and the presented case study are compared and discussed.

II. Timed automata based scheduling

Timed automata (TA) are finite state automata extended by the notion of clocks to model discrete event systems with timed behavior. The clocks are initialized with a value of zero at the start of the system and their valuations increase at the constant rate of $\dot{c}_i = 1$. The clocks in a TA cannot be stopped but can be reset to zero. After resetting a clock c_i to zero ($c_i := 0$) the valuation starts increasing again.

A timed automaton is defined by a tuple, $TA = (L, l_0, F, C, \Theta, inv)$, in which L represents the finite set of discrete locations, l_0 represents the initial location, and F represents the set of final locations. C represents the set of clocks assigned to the TA. The relation $\Theta = L \times \varphi(C) \times Act \times U(C) \times L$ represents the set of transitions between the locations. $\varphi(C)$ is a set of guards specified as conjunctions of constraints of the form $c_i \bullet n$ or $c_i - c_j \bullet n$, where $c_i, c_j \in C$, $\bullet \in \{<, \leq, ==, \neq, >, \geq\}$ and $n \in R \geq 0$. The guard conditions on the clock valuations have to be satisfied in order to enable a transition but their satisfaction does not force the transition to take place. Act denotes a set of actions (e.g. changing the value of a variable). $U(C)$ represents the set of clocks that are reset to zero after taking the transition. inv represents a set of invariants which assign conditions for staying in locations. The invariant conditions on the clock valuations must evaluate to true for the corresponding location to be active. The automaton is forced to leave the location when the invariant evaluates to false. A transition between a source location l and target location l' with a guard $g \in \varphi(C)$,

[*]This work has been partly performed as part of the MULTIFORM project, supported by the Seventh Research Framework Programme of the European Commission, grant agreement number: INFSO-ICT-224249.

C. Schoppmeyer, M. Hüfner, and S. Subbiah are junior researchers at the Process Dynamics and Operations Group, Department of Biochemical and Chemical Engineering, Technische Universität Dortmund, Germany (phone: +49 231 755 6080; fax: +49 231 755 5129; e-mail: christian.schoppmeyer@bci.tu-dortmund.de).

Sebastian Engell is Professor of the Process Dynamics and Operations Group, Department of Biochemical and Chemical Engineering, Technische Universität Dortmund, Germany (e-mail: s.engell@bci.tu-dortmund.de).

performing an action $a \in Act$ and resetting the clocks $r \in U(C)$ is denoted by (l, g, a, r, l'). Such a transition can occur only when the transition guard $g \in \varphi(C)$ is satisfied.

A. *Modeling approach for job shop problems*

The conceptual idea of modeling a job shop scheduling problem as TA is explained by a simple example problem. Two products (jobs) *A* and *B* have to be produced using machines M_1 and M_2. The task sequence and the durations for job *A* are $(t_1, M_1, 7) \rightarrow (t_2, M_2, 3)$. Similarly, for job *B* they are $(t_3, M_2, 5) \rightarrow (t_4, M_1, 2)$. The objective is to minimize the makespan.

The modular TA model for the example problem is shown in Fig 1. For each job $j \in \{A, B\}$ a separate *job automaton* is created and for each machine $k \in \{M_1, M_2\}$ a separate *machine automaton* is created.

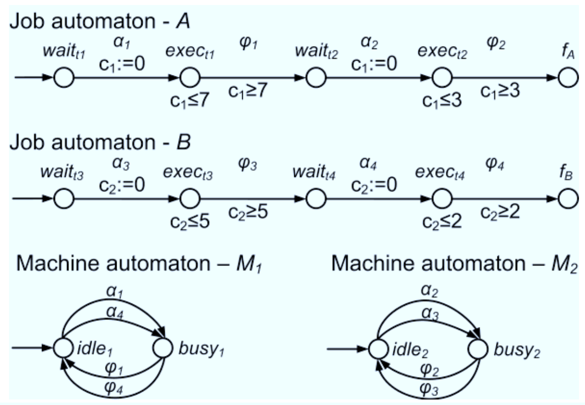

Figure 1. Modular TA model for the example problem.

Each machine automaton *k* consists of an $idle_k$ location representing that the machine is not executing a task and a $busy_k$ location representing that it is executing a task. The allocation of the machine to perform a task is represented by a transition from the $idle_k$ to the $busy_k$ location and the release of the machine after executing the task is represented by a transition from $busy_k$ to $idle_k$.

In the job automaton of a product *j* each task *i* is represented by two locations namely $wait_{ti}$ - indicating that the operation is waiting to be executed in the corresponding machine and $exec_{ti}$ - indicating that the operation is currently being executed in the machine. An additional location f_j is defined to indicate the termination of job *j*. Starting the execution of a task *i* by occupying a machine is represented by a transition labeled α_i and finishing an operation by releasing a machine is represented by a transition labeled ϕ_i. A clock c_j is introduced in the job automaton to model the elapse of time during the execution of tasks. The invariants in the $exec_{ti}$ locations of the corresponding tasks force the automaton to leave the locations once the specified durations have expired. The guard conditions on the clock on the transitions labeled with ϕ_i ensure that the tasks are executed at least for the corresponding durations. The invariants and the guards together ensure that a task is executed exactly for the defined duration.

The communications between the machine automata and the job automata are realized by the synchronization labels (α_i and ϕ_i) in the transitions. A transition in the job automaton can be taken only when a transition in the machine automaton with the same synchronization label can take place. This ensures that a machine does not perform more than one task at a time.

B. *Solution approach using reachability analysis*

The interacting automata are composed to form the complete model of the scheduling problem using a technique known as *parallel composition*. This technique is performed on-the-fly during the reachability analysis to reduce the state space complexity. The composed automaton is a directed graph, called the reachability graph, with nodes representing the state of the system and arcs representing the transitions from one state to another.

A cost-optimal reachability analysis is performed on the composed automaton, searching for a cost minimal path among all the paths starting from the initial location l_{init}, which represents that all tasks of all jobs are waiting to be executed and all machines are idle, to a final target location l_{target}, which represents that all jobs are finished and all machines are back to the idle state. This enumerative technique explores the state space step by step by evaluating the transition system on the fly. A detailed description of the basic reachability algorithm is given in [6]. Any path from the initial location l_{init} to a target location l_{target} represents a feasible schedule. The cost of a path is measured by a *global clock (GC)* which is introduced in the composed automaton. This global clock is started at l_{init}, never reset to zero, and stopped when l_{target} is reached. The time elapsed on the global clock at a node in the search tree indicates the cost incurred to reach this particular node, and the elapsed time on the global clock at a target node l_{target} represents the makespan of the corresponding schedule. The path with the minimal cost represents the optimal schedule.

For a given TA model, computing all paths from l_{init} to l_{target} to find the path with minimal cost among them is computationally expensive due to the combinatorial explosion. This feature demands for efficient techniques to reduce the search space to compute optimal or at least good solutions within reasonable computation times. Various state space reduction techniques, such as the *sleep-set* method which prunes redundant paths in the search tree and the *non-laziness* reduction scheme which prunes suboptimal paths are introduced to reduce the search effort. Different guiding techniques adapted from graph theory to guide the search to promising regions of the search tree are implemented in the reachability algorithm to guide the search to promising regions of the search space [6]. In [7], a linear program based (tailor made from the MILP formulation presented in [8]) bounding procedure embedded into the reachability algorithm to compute lower bound was introduced. Embedding lower bound computation enhanced the efficiency of the approach by pruning suboptimal paths in the search tree. In [9], a new *minimum remaining processing time (MRPT)* routine for lower bound computation was introduced and discussed which generates tighter lower bounds and is proven to be computationally more efficient than the LP-based bounding procedure.

III. Miniature pipeless plant

As a case study, a lab-scale pipeless plant (see Fig. 2), which is developed in the scope of the EU-project MULTIFORM, that provides numerous design and control challenges on different levels of the control hierarchies that are often confronted in industrial design processes, is considered [10].

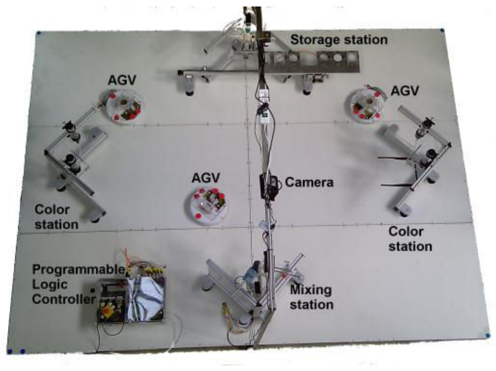

Figure 2. Picture of the miniature pipeless plant.

In the miniature pipeless plant, individually colored pieces of multi-layered *plaster art* are produced. The production of the pieces is done in mobile vessels which are transported between the different processing stations (machines) by a fleet of Automated Guided Vehicles (AGVs). The plant consists of filling stations which feed colored water to the mobile vessels, mixing stations which add plaster and mix the compound, and storage stations which can store up to six vessels each either filled with material to get hardened or empty.

The process steps (tasks) for producing a piece of *plaster art* are as follows: an empty vessel from a storage station is transported to one or more filling station(s) and filled with colored water. The vessel filled with the colored water is transported to a mixing station where plaster is added and mixed to form a compound. The compound is then transported to a storage station and let to harden for a few minutes to obtain a layer of the final product. Several cycles of this process can be executed to obtain multi-layered individually colored pieces of *plaster art*. The mobile vessels remain on the AGVs when the filling task is performed in the filling stations, but they need to be lifted up for processing in the mixing stations and in the storage stations. For each task performed in a station, the AGV is docked at the respective station. Hence, the AGV blocks the station from the time it reaches the station, during execution of a task, until it transports the vessel to the next station.

A. Case study data of the miniature pipeless plant

The layout of the miniature pipeless plant considered as case study in this contribution consists of two AGVs (*A1* and *A2*), two filling stations (*F1* and *F2*), one mixing station (*M*), one storage station (*S*), and six empty vessels. The objective is to produce six different pieces of *plaster art* (jobs) with three layers each with minimal makespan. For each layer, three different colors exist: $color_{F1}$, $color_{F2}$, and $color_{F3}$ where $color_{F1}$ is filled at filling station *F1*, $color_{F2}$ is filled at filling station *F2*, and $color_{F3}$ needs both colors of *F1* and *F2*, respectively. The complete color data for the six pieces of *plaster art* is given in Table I.

TABLE I. Color data for the case study

***Plaster art* piece**	**Colors per layer**		
	Layer 1	***Layer 2***	***Layer 3***
Plaster art 1	$color_{F1}$	$color_{F3}$	$color_{F2}$
Plaster art 2	$color_{F3}$	$color_{F1}$	$color_{F2}$
Plaster art 3	$color_{F2}$	$color_{F2}$	$color_{F1}$
Plaster art 4	$color_{F1}$	$color_{F1}$	$color_{F2}$
Plaster art 5	$color_{F3}$	$color_{F2}$	$color_{F3}$
Plaster art 6	$color_{F1}$	$color_{F3}$	$color_{F1}$

To reduce the complexity, it is assumed that if an AGV is assigned to a job then for the complete production of a layer it is attached to the vessel and can only be released after processing the layer completely. The durations of the tasks to produce one layer are given in Table II. To reduce the number of tasks, the time taken to transport the vessel to the station responsible for the next task are included in the respective task durations, e.g. in the first column for $color_{F1}$ two alternative tasks are listed which have to be executed in the storage station. The task with duration of 33 time units in the table represents the case where an idle AGV moves from an initial position to the station, docks it, and placing the vessel on the AGV. The task with duration of 13 time units represents the case where an idle AGV is docked in the storage station and the vessel can directly be placed in the AGV. The hardening tasks (see last column of Table II) can be executed independently without occupying any of the stations. Hence, *H* is a dummy machine.

TABLE II. Task durations of the case study

Layer color	**Processing stations and durations per task**					
$Color_{F1}$	*S*,33	*F1*,22		*M*,83	*S*,26	*H*,300
	S,13					
$Color_{F2}$	*S*,33	*F2*,32		*M*,76	*S*,26	*H*,300
	S,13					
$Color_{F3}$	*S*,33	*F1*,22	*F2*,23	*M*,83	*S*,26	*H*,300
	S,13	*F2*,32	*F1*,23	*M*,76		

B. Modeling the case study

The case study is modeled as a flexible job shop problem. For each job (piece of *plaster art*) a separate job automaton is created which consists of three basic parts, each representing the production of one layer.

For each AGV $l \in \{A1, A2\}$, a global shared variable S_l of integer type is introduced to model the assignment of the AGV to the production of a layer of a job. The value of the shared variable represents the current assignment and also the current status of the AGV: a value of *1* indicates that the

AGV is idle and docked in the storage station, a value of *2* indicates that the AGV is idle but not docked in the storage station (which corresponds to a collision-free initial position for each of the AGVs), and a value in the range of *[11, 16]* indicates that the AGV is currently assigned to a job (*plaster art* 1-6) and transports the corresponding vessel of the job.

For each station $k \in \{S, M, F1, F2\}$, a global shared variable S_k of type binary is introduced to model the blocking of the AGVs after executing a task in the corresponding station. The value of the shared variable represents the current state of the station: a value of *0* indicates that the docking position of the station is currently blocked by an AGV, and a value of *1* indicates that the docking position is free.

Fig. 3 shows the complete job automaton for the production of a layer. Each basic part of a job automaton consists of a *storage selection* part, a *filling* part (either *F1*, or *F2*, or both), a *mixing* part, a *storage* part, and a *hardening* part. The ending location of one part and the starting location of the next part are merged to single locations thus connecting the three basic parts of the job automaton.

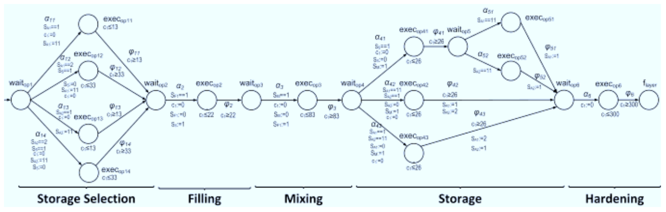

Figure 3. A complete recipe for one layer.

The *storage selection* part of a job automaton (see Fig. 4) consist of one common $wait_{op1}$ location, indicating that the combined task of selecting an AGV for the job and putting the production vessel on the AGV is waiting to be executed, and four different *exec* locations, indicating that the combined task is currently being executed. The guards on the transitions from the common $wait_{op1}$ location to the individual *exec* locations ensure that either a free AGV is currently docked at the storage station ($S_{A1} == 1$ or $S_{A2} == 1$), or that a free AGV is at its initial position and the docking position of the storage station is currently free for the AGV to dock ($S_{A1} == 2$ *and* $S_s == 1$ or $S_{A2} == 2$ *and* $S_s == 1$). The actions on the transitions update the respective shared variables of the selected AGV and of the occupancy of the docking position of the storage station.

The modeling of the *filling* parts and the *mixing* parts of a job automaton for a layer producing $color_{F1}$ or $color_{F2}$ are shown in Fig. 5 and Fig. 6, respectively. The guards on the shared variables on the transitions ensure the availability of the next station for docking the AGV and the actions update the values of the shared variables according to the movement of the AGVs.

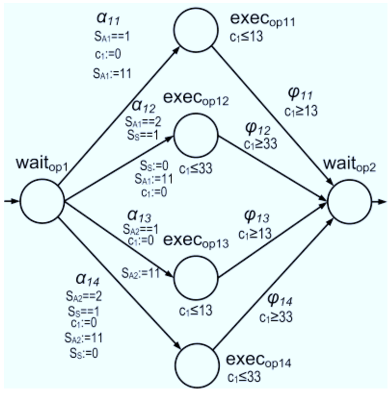

Figure 4. Storage selection part.

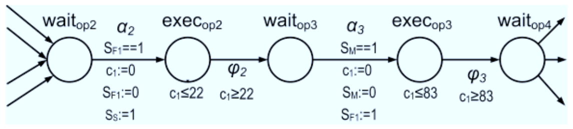

Figure 5. Filling at *F1* and mixing parts.

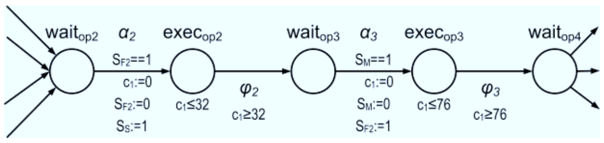

Figure 6. Filling at *F2* and mixing parts.

Fig. 7 shows the modeling of the *filling* part and the *mixing* part of a job automaton for the case of a layer producing $color_{F3}$. The sequencing of the tasks is split up into two possible combinations: $F1 \rightarrow F2$ or $F2 \rightarrow F1$. Since the duration of the mixing task includes the transportation time from the preceding station (*F1* or *F2*) to the mixing station, the *mixing* part has to be included in each of the possible combinations with the corresponding duration (see Table II).

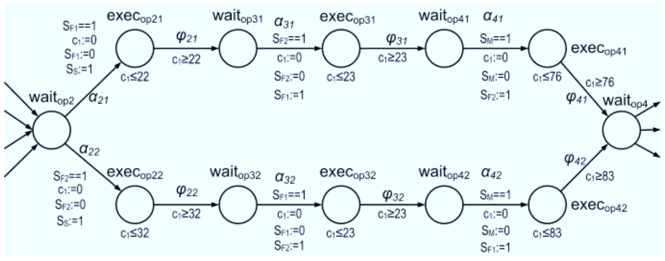

Figure 7. Filling *F1*&*F2* or *F2*&*F1* and mixing parts.

The *storage* part of a job automaton is shown in Fig. 8. The state that the job is waiting to start the combined tasks of transporting the vessel to the storage station, docking at the storage station and storing the vessel in the storage station is represented by the common location $wait_{op4}$. The execution of the combined tasks when the docking position of the storage station is free is represented in the top part. The update of the AGV availability and position after executing the combined task is also represented in the top part by the succeeding locations $wait_{op5}$, the $exec_{op51}$, and the $exec_{op52}$. The execution of the combined task when the docking position is occupied by AGV 1, or AGV 2, is represented in the middle and the lower part, respectively. The state of the job after finishing the combined storage task waiting to perform the hardening task is represented by the location *wait*$_{op6}$.

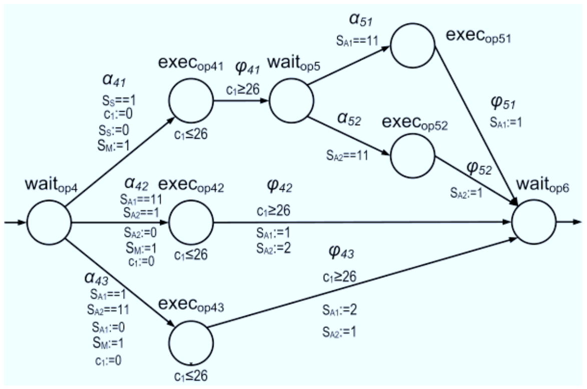

Figure 8. Storage part.

The *hardening* part of a job automaton (see Fig. 9) consists of *wait*$_{op6}$ location, representing the state where the job is waiting to start the execution of the hardening task, an *exec*$_{op6}$ location, indicating that the hardening task is currently being executed in a dummy machine, and a *f*$_{layer}$ location, indicating the finishing of a layer of the job. If the current layer is the third layer, the *f*$_{layer}$ location represents the finish location of the complete job automaton, otherwise the *f*$_{layer}$ location is removed and replaced by the *wait*$_{op1}$ location of the storage selection part of the next layer.

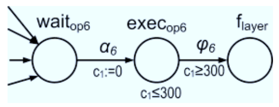

Figure 9. Hardening part.

IV. RESULTS

Two different TA-based tools have been considered to compare the effectiveness of the proposed modeling approach, TAOpt and UPPAAL 4.1.7. TAOpt is a TA-based scheduling tool developed in our group in which the *sleep-set* method, the *non-laziness* reduction scheme, various guiding techniques (see [6]), and the MRPT lower bound procedure (see [9]) are implemented. UPPAAL is a TA-based tool for model checking and simulation of discrete systems which offers various guiding techniques and a concept of urgent broadcast channels which prohibits unnecessary waiting in locations.

In the tests conducted using TAOpt, a combination of a depth-first followed by a best-first search strategy is chosen as the search algorithm. The MRPT lower bound computations are used for all the tests. In the *safe* settings where the optimal solution is not pruned the *safe sleep-set* method (*SSM*) and the *safe non-laziness* reduction are used and in the *unsafe* settings where the optimal solution might be pruned the *greedy sleep-set* and the *strict non-laziness* reduction are used. In the tests conducted using UPPAAL, the search algorithm chosen is the depth-first and fastest-trace for all test cases. The concept of urgent broadcast channels was used in the *asap* settings. The computation equipment used is a 64-bit Linux compute server with 2×2.4 GHz speed and 16 GB memory.

In order to investigate the performance of the TA-based tools considered, first a series of job shop problems were generated using the benchmark instance generator from [11]. The number of jobs is varied from 2 to 7 and the number of resources is varied from 2 to 7. The generator creates as many operations as there are resources for each job. The durations for the job operations are distributed uniformly within the range of [20-60]. The main reason for considering these test cases is to ensure that the complete search space for all problems can be enumerated by both tools.

In the tests conducted using TAOpt, the reachability analysis is terminated if: (a) the search space is completely explored or (b) the computation limit of 3600s is reached or (c) the number of nodes explored has reached 10 million. In the tests conducted using UPPAAL due to the lack of a possibility to provide a termination criterion, the reachability analysis is manually interrupted after the computation time limit of 3600s is reached.

Table III shows the number of nodes explored and the computation times required to prove the optimality of the best solution obtained in the reachability analysis (*safe* settings), or to find the best solution (*unsafe*, *asap* settings). The results show that both tools can enumerate the complete search space for small scale instances. For the larger instances among the test cases UPPAAL with the *safe* setting was not able to find a solution within the given time limit whereas TAOpt was able to find the optimal solution for all the tested instances with the *safe* settings.

UPPAAL and TAOpt perform equally for the smaller instances when the *unsafe* (*asap*) settings are employed. For the larger instances the combination of reduction techniques with MRPT lower bound computations reduces the space complexity in terms of numbers of nodes explored and the time complexity in terms of computation time required to find the best schedule. For an increase in the number of jobs and with constant number of resources, the search space increases substantially due to an increase in the number of degrees of freedom. This can be seen in Table III from the number of nodes explored in the reachability analysis. Since the search space increases, the computation time also increases as can be seen in Table III.

TABLE III. RESULTS ON THE GENERATED JOB SHOP INSTANCES I

Jobs / Res	**TAOpt *safe***		**TAOpt *unsafe***		**UPPAAL *safe***		**UPPAAL *asap***	
	EN	**T_{CPU}**	**EN**	**T_{CPU}**	**EN**	**T_{CPU}**	**EN**	**T_{CPU}**
2/2	12	0.002	9	0.008	11	0.022	11	0.022
2/3	16	0.010	13	0.006	120	0.021	15	0.023
2/4	25	0.012	17	0.002	329	0.024	18	0.023
2/5	30	0.008	21	0.010	681	0.026	22	0.022
2/6	44	0.012	26	0.004	1452	0.032	53	0.027
2/7	46	0.016	29	0.010	2135	0.036	33	0.026
3/2	33	0.002	25	0.010	278	0.025	33	0.021
3/3	42	0.008	19	0.010	2532	0.038	24	0.026
3/4	67	0.016	33	0.014	10122	0.100	60	0.023
3/5	228	0.020	77	0.012	35588	0.315	99	0.027
3/6	328	0.026	72	0.012	97608	0.933	78	0.029
3/7	232	0.014	85	0.012	287640	2.972	89	0.031
4/2	30	0.012	21	0.008	2348	0.045	98	0.025
4/3	170	0.008	76	0.014	36349	0.365	254	0.027
4/4	356	0.030	128	0.016	307203	4.143	345	0.029
4/5	464	0.038	104	0.016	2480210	50.957	137	0.032
4/6	783	0.064	97	0.022	11548496	330.455	106	0.037
4/7	1718	0.152	88	0.020	39567900	1464.471	141	0.037

Jobs / Res	TAOpt *safe*		TAOpt *unsafe*		UPPAAL *safe*		UPPAAL *asap*	
	EN	T_{CPU}	EN	T_{CPU}	EN	T_{CPU}	EN	T_{CPU}
5/2	37	0.012	30	0.010	13985	0.136	461	0.034
5/3	191	0.012	92	0.012	507424	6.054	836	0.033
5/4	362	0.030	99	0.008	13877012	401.756	555	0.036
5/5	2376	0.158	125	0.022	182847431	16734.371	192	0.035
5/6	1825	0.178	124	0.026	n/a	n/a	686	0.045
5/7	19235	1.958	334	0.046	n/a	n/a	2520	0.081
6/2	49	0.014	37	0.008	89816	0.897	2278	0.045
6/3	89	0.018	62	0.014	7467408	121.883	3867	0.060
6/4	3484	0.198	693	0.042	208822392	9935.047	11384	0.152
6/5	10322	0.782	132	0.028	n/a	n/a	2186	0.063
6/6	164674	15.148	3741	0.238	n/a	n/a	54581	0.707
6/7	107445	14.490	958	0.108	n/a	n/a	8452	0.152
7/2	61	0.012	58	0.012	352097	3.766	7282	0.089
7/3	449	0.036	216	0.026	83286276	1642.607	39144	0.511
7/4	19508	1.188	1192	0.076	n/a	n/a	70840	1.292
7/5	169219	13.968	3824	0.226	n/a	n/a	413	1.906
7/6	1260416	132.368	6742	0.446	n/a	n/a	51323	0.751
7/7	1371565	215.970	1351	0.148	n/a	n/a	35229	0.612

n/a – no solution could be obtained – computation time limit of 3600s reached

Table IV shows the best solutions (makespan value) obtained in the reachability analysis for the instances tested. When the *unsafe* (*asap*) settings are employed both tools might prune the optimal solution and should only be considered as a heuristic.

TABLE IV. RESULTS ON THE GENERATED JOB SHOP INSTANCES II

Jobs / Res	TAOpt		UPPAAL		Jobs / Res	TAOpt		UPPAAL	
	safe	unsafe	safe	asap		safe	unsafe	safe	asap
2/2	97	97	97	97	**5/2**	175	175	175	175
2/3	173	173	173	173	**5/3**	219	219	219	219
2/4	202	202	202	202	**5/4**	267	267	267	267
2/5	221	221	221	221	**5/5**	291	305*	291	305*
2/6	296	296	296	296	**5/6**	379	379	n/a	379
2/7	336	336	336	336	**5/7**	435	443*	n/a	443*
3/2	114	114	114	114	**6/2**	258	258	258	258
3/3	167	167	167	167	**6/3**	274	274	274	274
3/4	246	267*	246	267*	**6/4**	339	350*	339	350*
3/5	293	298*	293	298*	**6/5**	367	367	n/a	367
3/6	299	299	299	299	**6/6**	433	433	n/a	433
3/7	338	346*	338	346*	**6/7**	407	407	n/a	407
4/2	169	169	169	169	**7/2**	303	303	303	303
4/3	232	232	232	232	**7/3**	318	318	318	318
4/4	264	268*	264	268*	**7/4**	336	336	n/a	336
4/5	302	304*	302	304*	**7/5**	407	413*	n/a	413*
4/6	307	307	307	307	**7/6**	412	419*	n/a	419*
4/7	344	372*	344	372*	**7/7**	426	437*	n/a	437*

* - suboptimal solution – optimal solution pruned by unsafe reduction techniques
n/a – no solution could be obtained – computation time limit of 3600s reached

The second series of test instances considered is from the presented miniature pipeless plant case study. The number of jobs is varied from 1 to 6. Table V shows the number of nodes explored, the computation times required to prove the optimality of the best solution obtained in the reachability analysis (*safe* settings), or to find the best solution (*unsafe*, *asap* settings), and the makespan value of the best solution obtained in the reachability analysis by both tools.

TABLE V. RESULTS ON THE MINIATURE PIPELESS PLANT CASE STUDY

Jobs / Layers / Stations / AGVs	TAOpt *safe*		TAOpt *unsafe*		UPPAAL *safe*		UPPAAL *asap*		Best found solution
	EN	T_{CPU}	EN	T_{CPU}	EN	T_{CPU}	EN	T_{CPU}	
1/3/4/2	64	0.010	64	0.020	123	0.033	123	0.036	1371
2/3/4/2	538	0.030	300	0.030	226471	1.080	817	0.051	1474
3/3/4/2	8046	0.220	1979	0.130	161740845	2160.208	3358	0.079	1547
4/3/4/2	114217	3.790	8942	0.330	n/a	n/a	17572	0.183	1623
5/3/4/2	2424814	109.440	61061	2.260	n/a	n/a	293522	3.205	1694
6/3/4/2	10000000*	620.600*	772763	33.670	n/a	n/a	2408840	34.458	1802*

* - suboptimal solution – node limit of 10 million reached
n/a – no solution could be obtained – computation time limit of 3600s reached

In all cases except for the case of 6 jobs the optimality of the best solution could be proved by TAOpt using the *safe* settings. UPPAAL was not able to find a solution for the larger instances using *safe* settings. For the *unsafe*, *asap* settings both tools perform equally in terms of the computation time required to find the best solution, but TAOpt reduces the space complexity more efficiently in terms of numbers of nodes explored.

V. CONCLUSIONS

Two different TA-based tools (TAOpt vs. UPPAAL) were tested in a comparative study on job shop scheduling problems to show the general applicability of the approach to solve job shop problems and to enumerate the complete search space. A scheduling problem in a complex miniature pipeless plant case study was introduced and a timed automata model was presented. The proposed approach to model it as TA and to solve them using reachability analysis can be performed by both tools.

For obtaining fast, feasible, and good solutions, the TA-based scheduling tool TAOpt performs substantially better compared to UPPAAL. The combination of different reduction schemes with efficient guiding techniques from graph search and the MRPT-based lower bound computations implemented in the tool TAOpt are the salient features that make it an efficient tool in solving complex scheduling problems modeled as timed automata.

Current work includes building a reactive scheduling framework using timed automata models and a moving horizon technique for the presented case study with non-deterministic job arrival times and layer configuration.

REFERENCES

[1] S. Piana, S. Engell, "Hybrid Evolutionary Optimization of the Operation of Pipeless Plants", *Journal of Heuristics – Special Issue on Advances in Metaheuristics*, vol. 16, pp. 1381-1231, 2010.

[2] W. Huang, P. W. H. Chung, "Integrating routing and scheduling for pipeless plants in different layouts", *Computers and Chemical Engineering*, vol. 29, pp. 1069-1081, 2005.

[3] G. Behrmann, K. G. Larsen, J. I. Rasmussen, "Optimal scheduling using priced timed automata", *ACM Sigmetrics, Performance Evaluation*, vol. 32, pp. 34-40, 2005.

[4] Y. Abdeddaiem, E. Asarin, O. Maler, "Scheduling with timed automata", *Theoretical Comp. Science*, vol. 354, pp. 272-300, 2006.

[5] K. G. Larsen, P. Pettersson, W. Yi, "UPPAAL in a Nutshell", *Journal on Software Tools for Techn. Transfer*, vol. 1, pp. 134-152, 1998.

[6] S. Panek, S. Engell, S. Subbiah, O. Stursberg, "Scheduling of multi-product batch plants based upon timed automata models", *Computers and Chemical Engineering*, vol. 32, pp. 275-291, 2008.

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

[8] A. S. Manne, "On the job shop scheduling problem", *Operations Research*, vol. 8, pp. 219-223, 1960.

[9] S. Subbiah, C. Schoppmeyer, S. Engell, "Efficient scheduling of batch plants using reachability tree search for timed automata with lower bound computations", in *Proc. 21st European Symposium on Computer-Aided Process Engineering (ESCAPE-21)*, Chalkidiki, 2011, pp. 930-934.

[10] M. Reke, S. Grobosch, R. Hamberg, M. Hüfner, V. Kamin, M. Komareji, C. Sonntag, "D6.1.2 Final description of the case studies", Technical Report, *MULTIFORM*, 2010.

[11] E. Demirkol, S. Mehta, R. Uzsoy, "Benchmarks for shop scheduling problems", *European Journal of Operations Research*, vol. 8, pp. 219-223, 1998.