Optimal Infinite Runs in One-Clock Priced Timed Automata

Alexandre David[1], Daniel Ejsing-Duun[2], Lisa Fontani[2], Kim G. Larsen[1], Vasile Popescu[2], and Jacob Haubach Smedegård[2]

[1] {adavid,kgl}@cs.aau.dk
[2] {dejsin06,lfonta08,pvasil10,jhsm06}@student.aau.dk
Department of Computer Science, Aalborg University, Denmark

Abstract. We address the problem of finding an infinite run with the optimal cost-time ratio in a one-clock priced timed automaton and provide an algorithmic solution. Through refinements of the quotient graph obtained by strong time-abstracting bisimulation partitioning, we construct a graph with time and costs as weights and prove that the minimum ratio properties remain unaltered, allowing algorithms for finite weighted graphs to be applied. The resulting algorithm has an overall time complexity of $O((|Q| \cdot (|T| + |Q|))^6)$, $|Q|$ being the number of locations and $|T|$ the number of transitions of the one-clock priced timed automaton.

1 Introduction

Timed automata, introduced in 1990 by Alur and Dill [1, 2], and its extension, priced timed automata (PTA) [3, 4], are well established modeling formalisms for real time systems.

Consider as an example a PTA modeling a robotic vacuum cleaner (Fig.1). An interesting question for this model could be "Which is - energy-wise in the long run - the best cleaning strategy in a given situation for the robotic vacuum cleaner?". The cost-time ratios of diverse infinite runs of the same automaton can be widely different and it is important to be able to distinguish between these when measuring the efficiency of the robot. We can solve this problem by finding an optimal infinite run of the PTA to reduce energy consumption.

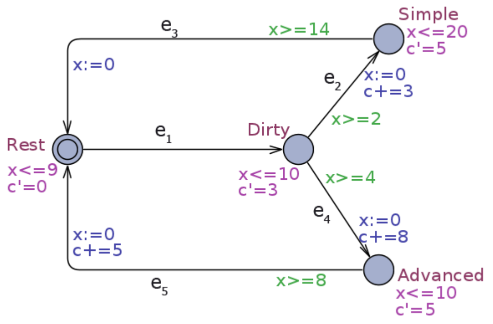

Fig. 1. PTA modeling the vacuum cleaner robot. Resets are denoted by "x:=0", costs, prefixed with "c+=", tell us the cost of taking a transition and "c'=" is the rate of growth for waiting in a location. The initial location of the model is *Rest*.

In this paper we will be concerned with finding such an optimal run for one-clock priced timed automata (1PTA).

The problem of optimal infinite scheduling can be defined differently depending on what is most relevant for the given situation. We define an optimal infinite run to be a run where the average cost per time unit of the entire run is as low as possible. When dealing with both costs and rewards an optimal strategy could imply the cost per reward to be minimized [5]. Instead for models where the present costs are more relevant than future costs, or where there is a growing probability that the object modeled will stop running (e.g. component failure), optimal infinite runs can be calculated using discount factors [6]. Both papers prove the respective problems to be PSPACE-complete, but lack a practically efficient implementation strategy (i.e. based on zones).

In this paper we provide an algorithmic implementation to solve this problem with polynomial complexity when restricting to the setting of priced timed automata with *one* clock.

2 Preliminaries

We will be concerned with priced timed automata with only one clock, x. The clock can be used to enforce time constraints on transitions and locations. The set of all clock constraints $\beta(x)$ is defined by the grammar "$g ::= x \sim k \mid g \wedge g$", where $k \in \mathbb{N}_{\geq 0}$ and $\sim \in \{\leq, <, =, >, \geq\}$. Also we have the opportunity of resetting the clock during a transition. This will be expressed by a boolean value.

Definition 1. *A one-clock priced timed automaton (1PTA) is a tuple $A = (Q, q_0, x, T, I, cost)$ where*

- *Q is a finite set of locations,*
- *$q_0 \in Q$ is the initial location,*
- *x is the model clock with valuations in the domain $\mathbb{R}_{\geq 0}$,*
- *$T \subseteq Q \times \beta(x) \times Bool \times Q$ is the set of transitions,*
- *$I : Q \to \beta(x)$ is an invariant function,*
- *cost: $Q \cup T \to \mathbb{N}_{\geq 0}$ is a price function.*

Like timed automata, the semantics of a 1PTA are given by a labelled transition system $(S, s_0, \to)$, where $S = \{(q, x, c) \mid x, c \in \mathbb{R}_{\geq 0},\ q \in Q,\ x \models I(q)\}$ is the set of states with accumulated cost c, $s_0 = (q_0, 0, 0)$ is the initial state, and the transition relation, $\to \subseteq S \times (T \cup \mathbb{R}_{\geq 0}) \times S$, is composed of delay moves and discrete moves. A *delay move* of δ time units, $(q, x, c) \xrightarrow{\delta} (q, x + \delta, c + \delta \cdot cost(q))$, is legal if for all $0 \leq \delta' \leq \delta$, $x + \delta' \models I(q)$. In a *discrete move* of the form $(q, x, c) \xrightarrow{e} (q', x', c + cost(e))$, the location is changed according to a transition $e = (q, g, r, q') \in T$ of the 1PTA. Also it must hold that $x \models g$, $x' \models I(q')$, and, if $r = True$, $x' = 0$, else $x' = x$.

A *finite run* γ of a 1PTA is a finite sequence of moves in the defined transition system, starting from the initial state: $\gamma = s_0 \to^1 s_1 \to^2 s_2 \to^3 \ldots \to^n s_n$. We can define the *cost* and *duration* of a run by:

$$cost(\gamma) = \sum_{1 \le i \le n} \begin{cases} \delta \cdot cost(q) & \text{if } \to^i \text{ is of the type } \xrightarrow{\delta}, \delta \in \mathbb{R}_{\ge 0}, s_{i-1} = (q, x, c) \\ cost(e) & \text{if } \to^i \text{ is of the type } \xrightarrow{e}, e \in T \end{cases}$$

$$dur(\gamma) = \sum_{1 \le i \le n} \begin{cases} \delta & \text{if } \to^i \text{ is of the type } \xrightarrow{\delta}, \delta \in \mathbb{R}_{\ge 0} \\ 0 & \text{if } \to^i \text{ is of the type } \xrightarrow{e}, e \in T \end{cases}$$

The *ratio* of a run expresses what the cost per time unit is in the run: $ratio(\gamma) = \frac{cost(\gamma)}{dur(\gamma)}$. Note that the ratio only exists if there is at least one delay transition, $\xrightarrow{\delta}$, where $\delta \neq 0$ in γ.

In this paper we will be concerned with infinite runs, viz. infinite sequences of moves: $\Gamma = s_0 \to^1 s_1 \to^2 s_2 \to^3 \dots \to^n s_n \to^{n+1} \dots$. Let Γ_n denote the prefix of length n of Γ. Now we can define the ratio of an infinite run as:

$$ratio(\Gamma) = \liminf_{n \to +\infty} ratio(\Gamma_n)$$

We will define as optimal a run where the ratio is minimal.

Definition 2. *Given a 1PTA A, we define μ^*_A to be the optimal ratio of A.*

$$\mu^*_A = \inf\{ratio(\Gamma) \mid \Gamma \; is\; an\; infinite\; run\; of\; A\}$$

A run Γ_{opt} of A is an *optimal infinite run* if $ratio\,(\Gamma_{opt})$ equals μ^*_A. Note that a ratio-optimal run may not exist. In this case, we will say that a family of runs is a ratio-optimal family if for any $\epsilon > 0$ there is a run of the family with ratio being ϵ-close to μ^*_A.

In this paper we will not consider models with zeno-behaviour, meaning that the 1PTAs we consider cannot model an infinite amount of discrete moves in a finite amount of time. Zeno-behaviours are not a realistic depiction of an actual system and can easily be detected through the use of existing automated tools.

3 Strong Time-Abstracting Bisimulation

For any 1PTA $A = (Q, q_0, x, T, I, cost)$ we can define equivalence classes between its states based on strong time-abstracting bisimulations (STAB)[7]. These classes permit us to abstract from the specific amount of time that passes and focus on the discrete actions that can be performed during specific intervals of time in each location.

Definition 3. *Let A be a timed automaton with transitions T and states S. A binary relation, $\Re$, on the states of A is a strong time-abstracting bisimulation (STAB) if for all tuples $(s_1, s_2) \in \Re$, where $s_1, s_2 \in S$ the following holds:*

- *if $s_1 \xrightarrow{e} s'_1$ for some $e \in T$ then there is a transition $s_2 \xrightarrow{e} s'_2$ such that $(s'_1, s'_2) \in \Re$*

- *if* $s_1 \xrightarrow{\delta_1} s'_1$ *for some delay* $\delta_1 \in \mathbb{R}_{\geq 0}$, *then there is a transition* $s_2 \xrightarrow{\delta_2} s'_2, \delta_2 \in \mathbb{R}_{\geq 0}$, *such that* $(s'_1, s'_2) \in \Re$
- *the same holds if the roles of* s_1 *and* s_2 *are reversed*

Known algorithms exist for computing the reachable classes produced by the greatest STAB [8]. The *quotient graph of A*, called A_Q, is a graph with the classes induced by the greatest STAB on A as nodes and two types of edges:

- $C_1 \xrightarrow{e} C_2$, for some $e \in T$, T being the set of transitions of A, if $\exists s \in C_1, s' \in C_2$ such that $s \xrightarrow{e} s'$
- $C_1 \xrightarrow{\epsilon} C_2$, where ϵ denotes some delay, if $\exists (q, x) \in C_1, \delta \in \mathbb{R}_{\geq 0}$, such that $(q, x + \delta) \in C_2$ and for all $0 \leq \delta' < \delta, (q, x + \delta') \in C_1 \cup C_2$.

This quotient graph is finite since it is proven that the region equivalence is a strong time-abstracting bisimulation [8]. Moreover A_Q is a forwards stable graph in which the classes obtained from the greatest STAB can be represented in terms of symbolic states.

A symbolic state is a tuple $S = (q, \langle x_1, x_2 \rangle)$, where $q \in Q$ and $x_1, x_2 \in \mathbb{R}_{\geq 0} \cup \{\infty\}$ such that for all $x' \in \langle x_1, x_2 \rangle, x' \models I(q)$, $\langle \in \{ \, (\, , \, [\, \}$ and $\rangle \in \{ \,) \, , \,] \, \}$.

A graph is forward stable if for any two symbolic states in the graph, S_1, S_2 s.t. $S_1 \rightarrow S_2$ it holds that for any state $s_1 \in S_1$ there exist a state $s_2 \in S_2$ s.t. $s_1 \rightarrow s_2$.

On the other hand backwards stability would require that for any two symbolic states, S_1, S_2, s.t. $S_1 \xrightarrow{e} S_2$, $e = (q, g, False, q')$, it must hold that for any state $s_2 \in S_2$ there exists a state $s_1 \in S_1$ s.t. $s_1 \rightarrow s_2$.

The quotient graph is not necessarily backwards stable. The quotient graph for the robotic vacuum cleaner is shown in Fig. 2(a). Note that the symbolic state $S = (Dirty, [4, 10])$ is not backwards stable, since the interval is not equal to the node before and the incoming transition does not require a reset.

For our purpose we need the graph to be both forwards and backwards stable. This will allow us to introduce precise weights on the edges of the graph representing cost and time, and hence calculate the best ratio and find the corresponding infinite run. We can split the classes obtained from the STAB to make them backwards stable and construct a refined quotient graph A_R. This can be viewed as a coarse region abstraction. The refined quotient graph for the robotic vacuum cleaner is shown in Fig. 2(b).

4 Minimizing the Ratio in Infinite Runs

We can now construct a graph that also considers the cost information of the 1PTA by making from each state in A_R two single nodes representing the states with the upper and lower extreme clock valuations of the interval.

Definition 4. *Let A be a 1PTA with cost function cost and* $A_R = (N, E)$ *be the refined quotient graph of A. The weighted graph of A,* $A_W = (N', E')$, *is a doubly weighted graph defined as follows:*

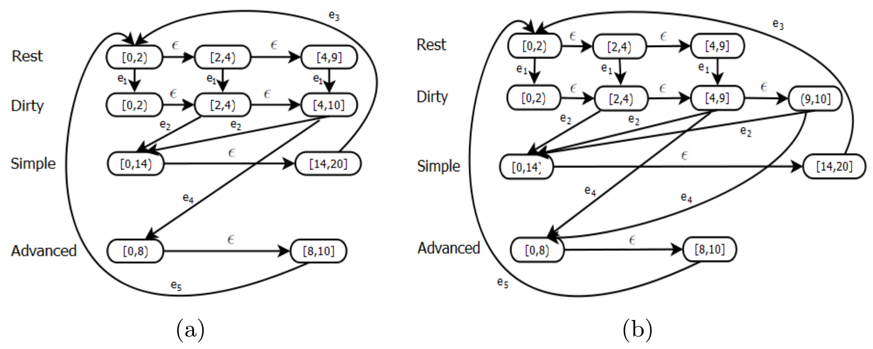

Fig. 2. Quotient graph (a) and refined quotient graph (b) for the robotic vacuum cleaner. The locations for the states are written on the left and the clock intervals for each location represent a node in the graph.

- *For all* $n = (q, \langle x_1, x_2\rangle) \in N$*, there is a node* $n_1 = (q, \langle x_1) \in N'$ *and if* $x_1 \neq x_2$ *also a node* $n_2 = (q, x_2\rangle) \in N'$ *and an edge from* n_1 *to* n_2 *with* $cost = cost(q) \cdot (x_2 - x_1)$ *and* $time = x_2 - x_1$*.*
- *For all* $n = (q, \langle x_1, x_2\rangle) \in N$ *if there exists an edge* $\xrightarrow{e} \in E$ *and some node* n' *such that* $n \xrightarrow{e} n'$*, then there is an edge* $e'_1 \in E'$ *from* n_1 *to* n'_1 *and an edge* $e'_2 \in E'$ *from* n_2 *to* n'_2 *(or* n'_1 *if* $\xrightarrow{e}$ *has a reset) , both with* $cost = cost(e)$ *and* $time = 0$*, where* n_1 *and* n_2 *have been made from splitting* n *and equally* n'_1 *and* n'_2 *have been made from splitting* n'*. If* $x_1 = x_2$*, then there will only be one edge from* n_1 *to* n'_1*.*
- *For all* $n = (q, \langle x_1, x_2\rangle) \in N$ *if there is a delay transition* $\xrightarrow{\epsilon} \in E$ *and some node* n' *such that* $n \xrightarrow{\epsilon} n', n' = (q, \langle x'_1, x'_2\rangle)$*, then there is an edge* $e' \in E'$ *from* n_2 *to* n'_1 *with* $cost = 0$ *and* $time = 0$*, where* n_1 *and* n_2 *have been made from splitting* n*. Similarly* n'_1 *and* n'_2 *have been made from splitting* n'*.*

A_W corresponds to a coarse corner point abstraction when dealing with one-clock timed automata [5]. The weighted graph for the robotic vacuum cleaner is shown in Fig. 3. An infinite path in A_W with the smallest ratio, $\mu^*_{A_W}$, has the same ratio as the optimal ratio of A. This leads to the following theorem:

Theorem 1. *Let* A *be a 1PTA. Then* $\mu^*_A = \mu^*_{A_W}$

Proof sketch: The full proof has been left out due to lack of space. However, we prove that:

1. For any infinite run Γ in A there exists an infinite path Γ_e in A_W s.t. $\mu(\Gamma_e) \leq \mu(\Gamma)$, and
2. For any infinite path Γ_e in A_W and any $\epsilon \geq 0$ there exists an infinite run Γ in A s.t. $\mu(\Gamma) \leq \mu(\Gamma_e) + \epsilon$
3. Q.E.D.

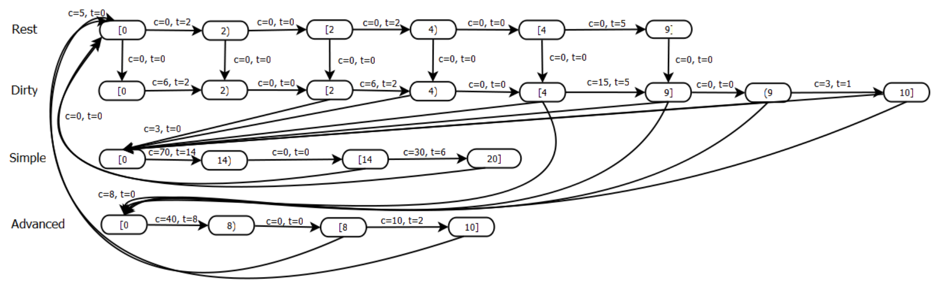

Fig. 3. The doubly weighted graph for the robotic vacuum cleaner. The time value of each edge is prefixed with "t=", while the cost value is prefixed with "c=".

Fig. 4 shows the comparison between a run in the original 1PTA and a run in A_W. As can be seen in A_W it is only possible to take discrete transitions at the endpoints of a time interval.

The proof will compare a run in the automaton A and an equivalent path in A_W and show that delaying only in the locations with the lowest possible cost within each time interval can make the path in A_W at least as good as the run in A. In case of transition resets we can determine if delaying before taking the transition affects our ratio or not. In both cases, we would get the optimal ratio from taking the transition at an interval endpoint.

The only runs that are represented in A_W which do not have a corresponding run in A are the ones that involve a transition constrained by strict guards or invariants. However such constraints do allow the run to take the transition at a time-point being arbitrarily close, thus leading to an optimal family of runs.

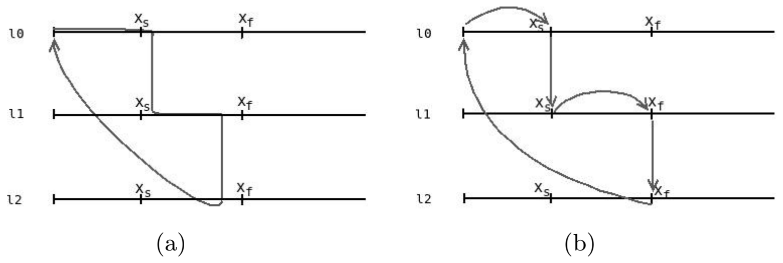

Fig. 4. An infinite run in original A (a) and the optimal infinite run found in the correspondent doubly weighted graph A_W (b).

The amount of nodes in A_W compared to A_R will increase, while the state space represented will dramatically decrease. Before all reachable states were represented, while now only the finite amount of states whose clock valuations correspond to the extremes of the time intervals of the previous states are left.

There is only a finite amount of nodes in A_W, which means that it is only possible to produce an infinite path in A_W by repeating some of these. This means that the infinite path will be composed of cycles and represent an infinite run in the original automaton. However, if we have a node representing infinite time in the graph, resulting from the split of an interval that has infinity as upper bound, we can have an infinite run by simply delaying infinitely in the respective location. We can remove these nodes and their adjacent edges from A_W and separately check for reachable locations where it is possible to wait an infinite amount of time when searching for the optimal infinite run. The problem of finding that run then reduces to finding the cycle with the lowest cost-time ratio in A_W and comparing it with the cost of the locations allowing infinite delay.

We can define a simple cycle in A_W as a cycle which does not visit any node twice, except the first one. In a similar way, a complex cycle shall be defined as a cycle that contains at least two nodes that are visited more than once or one node visited at least three times. Knowing these definitions, we can now state an important property for the cycle with the best ratio in A_W.

Theorem 2. *A complex cycle in a doubly weighted graph cannot have a better ratio than one of the simple cycles it is composed of.*

As with Theorem 1, the proof has been left out due to lack of space.

5 Minimal Cost-Time Ratio Cycle

Having a doubly weighted graph A_W with cost and time values on the edges and an initial node I, the cycle with the lowest cost per time unit, which is equal to the optimal ratio of the originating automaton, can be found using an algorithm presented by Lawler [9].

We can find an optimal infinite behaviour for the robotic vacuum cleaner by running this algorithm on its weighted graph, since no location allows for infinite delays. An optimal infinite run would hence wait 9 time units in *Rest*, go to *Dirty* and wait further 1 time unit. Afterwards it would start cleaning in advanced mode for 8 time units and go to *Rest* again, finishing the cycle. The ratio of this cycle is $\mu = \mu^* = 3.\overline{1}$ (periodic).

The total complexity of the bisimulation is bounded by $O(|Q|^2 \cdot (|Q| + |T|)^2)$ where $|Q|$ is the amount of locations and $|T|$ is the amount of transitions of the original 1PTA. The graph produced, A_W, has at most $2 \cdot |partitions|$ vertices, with $|partitions| = |Q| \cdot (|Q| + |T|) \cdot 2$ being the maximum number of symbolic states produced by the refined bisimulation. Lawler's algorithm makes use of the Moore-Bellman-Ford algorithm which has a complexity of $\mathrm{O}(|V|^3)$. This makes the algorithm for finding the optimal ratio $\mathrm{O}(|V|^6)$. With respect to the input 1PTA we then have an overall complexity of:

$$O((|partitions|)^6) = O((|Q| \cdot (|Q| + |T|))^6)$$

6 Experimental Evaluation

In this paper we have presented a simple example of a robotic vacuum cleaner cleaning a single room. In a more complex setting the robot could be faced with the job of cleaning a whole apartment. Let us consider the case where this apartment has more than one room and each room has one of four different sizes. The size of the rooms influences how much time the vacuum cleaner needs to clean in each mode. By using the algorithms presented, we can find an optimal infinite run for the vacuum cleaner in this larger model.

To test our solution, we have implemented it using the Python programming language and a series of rooms, designed on the same principle as the model in Fig. 1, extrapolated to different sizes to model the apartment. We can compare the running time of our solution to a more simple solution where splittings are made based on all constraints on all locations and edges in the model. All tests have been run on a 2.93 GHz Intel Core i7 using a single core and an insignificant amount of memory. A selection of results is shown in Table 1 together with the size of A_W in both cases.

Table 1. Performance of our solution compared to a complete splitting for the apartment cleaning problem.

Rooms	Our solution			Complete splitting		
	Time	Nodes	Edges	Time	Nodes	Edges
1	0.18 sec	22	38	0.43 sec	42	66
4	5.48 sec	64	111	3 min 3.91 sec	269	398
8	17.89 sec	120	185	38 min 34.04 sec	519	759
12	1 min 17.62 sec	176	269	1 h 36 min 52.05 sec	769	1120

We see here the strength of reducing the amount of symbolic states to the largest STAB that makes the equivalence classes forward and backward stable. In a model that makes use of resets in some transitions, our approach ensures a smaller size of the weighted graph and thus a lower running time. From the results we see that for the 12 room example, the running time of the complete splitting is more than 60 times higher than our approach. However, in the worst case, where no resets are used, our approach will also be forced to make a complete splitting of each interval and have a slightly higher running time than doing the complete splitting immediately.

7 Conclusion

In this paper we have shown a polynomial time algorithm for finding infinite runs with the optimal cost-time ratio in a one-clock priced timed automaton A. We show how the quotient graph A_Q obtained through strong time-abstracting bisimulation can be converted into a forwards and backwards stable quotient graph, A_R. We construct a doubly weighted graph A_W and prove that an optimal infinite run, Γ, in A has the same ratio as that of another optimal infinite run of A, Γ_e, represented in A_W. Γ_e either exhibits an infinite cyclic behaviour or is delaying infinitely in one location. Using Lawler's minimum ratio cycle algorithm on A_W we can then find an optimal infinite cyclic behaviour of A, if it exists. Finally, the ratio of the cycle can be compared to the cost of all locations allowing infinite delay to find the optimal infinite run of A.

A possible extension to the problem would be to consider more cost variables in the same one-clock priced timed automaton. In this situation the problem would not be reduced to finding an optimal infinite run for one cost variable, but instead the Pareto frontier for the cost variables of the automaton, since a run that is optimal with respect to one variable may not be optimal for the others.

References

1. Alur, R., Dill, D.L.: A theory of timed automata. Theoretical Computer Science **126**(2) (1994) 183 – 235
2. Alur, R., Dill, D.: Automata for modeling real-time systems. In Paterson, M., ed.: Automata, Languages and Programming. Volume 443 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg (1990) 322–335
3. Alur, R., La Torre, S., Pappas, G.: Optimal paths in weighted timed automata. In Di Benedetto, M., Sangiovanni-Vincentelli, A., eds.: Hybrid Systems: Computation and Control. Volume 2034 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg (2001) 49–62
4. Behrmann, G., Fehnker, A., Hune, T., Larsen, K., Pettersson, P., Romijn, J., Vaandrager, F.: Minimum-cost reachability for priced time automata. In Di Benedetto, M., Sangiovanni-Vincentelli, A., eds.: Hybrid Systems: Computation and Control. Volume 2034 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg (2001) 147–161
5. Bouyer, P., Brinksma, E., Larsen, K.: Optimal infinite scheduling for multi-priced timed automata. Formal Methods in System Design **32** (2008) 3–23
6. Fahrenberg, U., Larsen, K.G.: Discount-optimal infinite runs in priced timed automata. Electronic Notes in Theoretical Computer Science **239** (2009) 179 – 191 Joint Proceedings of the 8th, 9th, and 10th International Workshops on Verification of Infinite-State Systems (INFINITY 2006, 2007, 2008).
7. Larsen, K., Wang, Y.: Time-abstracted bisimulation: Implicit specifications and decidability. Information and Computation **134** (1997) 75–101
8. Tripakis, S., Yovine, S.: Analysis of timed systems using time-abstracting bisimulations. Formal Methods in System Design **18** (2001) 25–68
9. Lawler, E.L.: Chapter 13. In: Combinatorial Optimization: Networks and Matroids. Holt, Rinehart and Winston, New York, USA (1976) 94–97