FI MU

Faculty of Informatics
Masaryk University Brno

Dual-Priced Modal Transition Systems with Time Duration

by

Nikola Beneš
Jan Křetínský
Kim G. Larsen
Mikael H. Møller
Jiří Srba

FI MU Report Series **FIMU-RS-2012-01**

 January 2012

Publications in the FI MU Report Series are in general accessible via WWW:

http://www.fi.muni.cz/reports/

Further information can be obtained by contacting:

Faculty of Informatics
Masaryk University
Botanická 68a
602 00 Brno
Czech Republic

Dual-Priced Modal Transition Systems with Time Durations

Nikola Beneš
Masaryk University
xbenes3@fi.muni.cz

Jan Křetínský
Masaryk University
Technische Universität München
jan.kretinsky@fi.muni.cz

Kim G. Larsen
Aalborg Universitet
kgl@cs.aau.dk

Mikael H. Møller
Aalborg Universitet
mikaelhm@cs.aau.dk

Jiří Srba
Aalborg Universitet
srba@cs.aau.dk

January 16, 2012

Abstract

Modal transition systems are a well-established specification formalism for a high-level modelling of component-based software systems. We present a novel extension of the formalism called modal transition systems with durations where time durations are modelled as controllable or uncontrollable intervals. We further equip the model with two kinds of quantitative aspects: each action has its own running cost per time unit, and actions may require several hardware components of different costs. We ask the question, given a fixed budget for the hardware components, what is the implementation with the cheapest long-run average reward. We give an algorithm for computing such optimal implementations via a reduction to a new extension of mean payoff games with time durations and analyse the complexity of the algorithm.

1 Introduction and Motivating Example

Modal Transition Systems (MTS) is a specification formalism [15, 2] that aims at providing a flexible and easy-to-use compositional development methodology for reactive systems. The formalism can be viewed as a fragment of a temporal logic [1, 8] that at the same time offers a behavioural compositional semantics with an intuitive notion of process refinement. The formalism of MTS is essentially a labelled transition system that distinguishes two types of labelled transitions: *must* transitions which are required in any refinement of the system, and *may* transitions that are allowed to appear in a refined system but are not required. The refinement of an MTS now essentially consists of iteratively resolving the presence or absence of may transitions in the refined process.

In a recent line of work [14, 3], the MTS framework has been extended to allow for the specification of additional constraints on quantitative aspects (e.g. time, power or memory), which are highly relevant in the area of embedded systems. In this paper we continue the pursuit of quantitative extensions of MTS by presenting a novel extension of MTS with time durations being modelled as controllable or uncontrollable intervals. We further equip the model with two kinds of quantitative aspects: each action has its own running cost per time unit, and actions may require several hardware components of different costs. Thus, we ask the question, given a fixed budget for the investment into the hardware components, what is the implementation with the cheapest long-run average reward.

Before we give a formal definition of modal transition systems with durations (MTSD) and the dual-price scheme, and provide algorithms for computing optimal implementations, we present a small motivating example.

Consider the specification $\mathcal{S}$ in Figure 1a describing the work of a shuttle bus driver. He drives a bus between a hotel and the airport. First, the driver has to `Wait` for the passengers at the hotel. This can take one to five minutes. Since this behaviour is required to be present in all the implementations of this specification, it is drawn as a solid arrow and called a *must* transition. Then the driver has to `Drive` the bus to the airport (this takes six to ten minutes) where he has to do a `SmallCleanup`, then `Wait` before he can `Drive` the bus back to the hotel. When he returns he can do either a `SmallCleanup`, `BigCleanup` or `SkipCleanup` of the bus before he continues. Here we do not require a particular option to be realised in the implementations, hence we only draw the transitions as dashed arrows. As these transitions may or may not be present in the implementations, they are called *may* transitions. However, here the intention is to require at least one option be realised. Hence, we specify this using a propositional formula Φ assigned to the state $\mathtt{t}$ over its outgoing transitions as described in [5, 6]. After performing one of the actions, the driver starts over again. Note that next time the choice in $\mathtt{t}$ may differ.

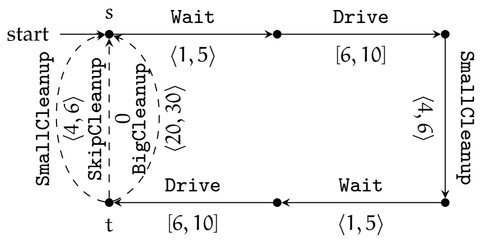
Observe that there are three types of durations on the transitions. First, there are *controllable* intervals, written in angle brackets. The meaning of e.g. $\langle 1, 5 \rangle$ is that in the implementation we can instruct the driver to wait for a fixed number of minutes in the range. Second, there are *uncontrollable* intervals, written in square brackets. The interval $[6, 10]$ on the `Drive` transition means that in the implementation we cannot fix any particular time and the time can vary, say, depending on the traffic and it is chosen nondeterministically by the environment. Third, the degenerated case of a single number, e.g. 0, denotes that the time taken is always constant and given by this number. In particular, a zero duration means that the transition happens instantaneously.

The system $\mathcal{S}_1$ is another specification, a *refinement* of $\mathcal{S}$, where we additionally specify that the driver must do a `SmallCleanup` after each `Drive`. Note that the `Wait` interval has been narrowed. The system $\mathcal{I}_1$ is an implementation of $\mathcal{S}_1$ (and actually also of $\mathcal{S}$) where all controllable time intervals have already been fully resolved to their final single values: the driver must `Wait` for 5 minutes and do the `SmallCleanup` for 6 minutes. Note that uncontrollable intervals remain unresolved in the implementations and the time is chosen by the environment each time the action is performed. This reflects the inherent uncontrollable uncertainty of the timing, e.g. of a traffic.

The system $\mathcal{S}_2$ is yet another specification and again a refinement of $\mathcal{S}$, where the driver can always do a `BigCleanup` in `t` and possibly there is also an alternative allowed here of a `SmallCleanup`. Notice that both `SmallCleanup` intervals have been restricted and changed to uncontrollable. This means that we give up the control over the duration of this action and if this transition is implemented, its duration will be every time chosen nondeterministically in that range. Finally, $\mathcal{I}_2$ is then an implementation of $\mathcal{S}_2$ and $\mathcal{S}$.

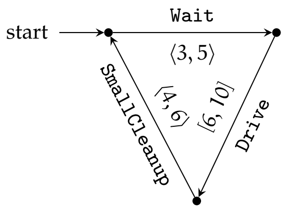
Furthermore, we develop a way to model cost of resources. Each action is assigned a *running price* it costs per time unit, e.g. `Drive` costs 10 each time unit it is being performed as it can be seen in the left table of Figure 1f. In addition, in order to perform an action, some hardware may be needed, e.g. a `VacuumCleaner` for the `BigCleanup` and its price is 100 as can be seen on the right. This *investment price* is paid once only.

Let us now consider the problem of finding an optimal implementation, so that we spend the least possible amount of money (e.g. the pay to the driver) per time unit while conforming to the specification $\mathcal{S}$. We call this problem *the cheapest implementation problem*. The optimal implementation is to buy a vacuum cleaner if one can afford an investment of 100 and do the `BigCleanup` every time as long as possible and `Wait` as shortly as possible. (Note that `BigCleanup` is more costly per time unit than `SmallCleanup` but

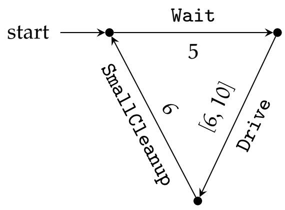


$\Phi(t) = \quad (\texttt{BigCleanup}, s) \vee$
$(\texttt{SkipCleanup}, s) \vee$
$(\texttt{SmallCleanup}, s)$

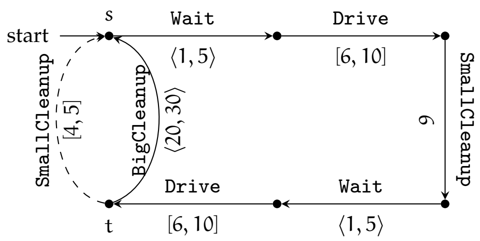
(a) Specification $\mathcal{S}$



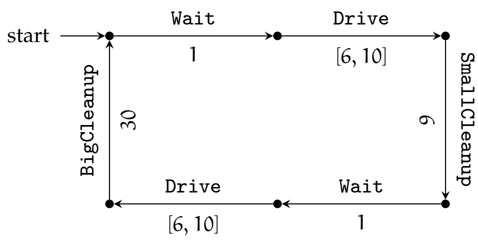
(b) Specification $\mathcal{S}_1$



(c) Implementation $\mathcal{I}_1$



(d) Specification $\mathcal{S}_2$



(e) Implementation $\mathcal{I}_2$

$H = \{\texttt{VacuumCleaner}, \texttt{Sponge}\}$

$a \in \Sigma$	$r(a)$
Wait	8
Drive	10
SmallCleanup	6
BigCleanup	7
SkipCleanup	0

$$\Psi(a) = \begin{cases} \texttt{VacuumCleaner} & \text{if } a = \texttt{BigCleanup} \\ \texttt{Sponge} \vee \texttt{VacuumCleaner} & \text{if } a = \texttt{SmallCleanup} \\ \mathit{true} & \text{otherwise} \end{cases}$$

$h \in H$	$i(h)$
VacuumCleaner	100
Sponge	5

(f) Price Scheme

Figure 1: Example of Dual-Priced Modal Transition Systems with Time Durations

lasts longer.) This is precisely implemented in $\mathcal{I}_2$ and the (worst-case) average cost per time unit is ≈ 7.97. If one cannot afford the vacuum cleaner but only a sponge, the optimal worst case long run average is then a bit worse and is implemented by doing the `SmallCleanup` as long as possible and `Wait` now as *long* as possible. This is depicted in $\mathcal{I}_1$ and the respective average cost per time unit is ≈ 8.10.

The most related work is [11] where prices are introduced into a class of interface theories and long-run average objectives are discussed. Our work omits the issue of distinguishing input and output actions. Nevertheless, compared to [11], this paper deals with the time durations, the one-shot hardware investment and, most importantly, refinement of specifications. Further, timed automata have also been extended with prices [4] and the long-run average reward has been computed in [9]. However, priced timed automata lack the hardware and any notion of refinement, too.

The paper is organized as follows. We introduce the MTS with the time durations in Section 2 and the dual-price scheme in Section 3. Section 4 presents the main results on the complexity of the cheapest implementation problem. First, we state the complexity of this problem in general and in an important special case and prove the hardness part. The algorithms proving the complexity upper bounds are presented only after introducing an extension of mean payoff games with time durations. These are needed to establish the results but are also interesting on their own as discussed in Section 4.1. We conclude and give some more account on related and future work in Section 5.

2 Modal Transition Systems with Durations

In order to define MTS with durations, we first introduce the notion of *controllable* and *uncontrollable* duration intervals. A controllable interval is a pair $\langle m, n\rangle$ where $m, n \in \mathbb{N}_0$ and $m \le n$. Similarly, an uncontrollable interval is a pair $[m, n]$ where $m, n \in \mathbb{N}_0$ and $m \le n$. We denote the set of all controllable intervals by $\mathfrak{I}_c$, the set of all uncontrollable intervals by $\mathfrak{I}_u$, and the set of all intervals by $\mathfrak{I} = \mathfrak{I}_c \cup \mathfrak{I}_u$. We also write only m to denote the singleton interval $[m, m]$. Singleton controllable intervals need not be handled separately as there is no semantic difference to the uncontrollable counterpart.

We can now formally define modal transition systems with durations. In what follows, $\mathcal{B}(X)$ denotes the set of propositional logic formulae over the set X of atomic propositions, where we assume the standard connectives $\wedge, \vee, \neg$.

Definition 2.1 (MTSD). *A* Modal Transition System with Durations *(MTSD) is a tuple* $\mathcal{S} = (S, T, D, \Phi, s_0)$ *where* S *is a set of* states *with the* initial state s_0*,* $T \subseteq S \times \Sigma \times S$ *is a set of* transitions*,* $D : T \to \mathfrak{I}$ *is a* duration interval function*, and* $\Phi : S \to \mathcal{B}(\Sigma \times S)$ *is an* obligation function*. We assume that whenever the atomic proposition* (a, t) *occurs in the Boolean formula* $\Phi(s)$ *then also* $(s, a, t) \in T$*.*

We moreover require that there is no cycle of transitions that allows for zero accumulated duration, i.e. there is no path $s_1 a_1 s_2 a_2 \cdots s_n$ *where* $(s_i, a_i, s_{i+1}) \in T$ *and* $s_n = s_1$ *such that for all* i*, the interval* $D((s_i, a_i, s_{i+1}))$ *is of the form either* $\langle 0, m\rangle$ *or* $[0, m]$ *for some* m*.*

Note that instead of the basic may and must modalities known from the classical modal transition systems (see e.g. [2]), we use arbitrary boolean formulae over the outgoing transitions of each state in the system as introduced in [6]. This provides a higher generality as the formalism is capable to describe, apart from standard modal transition systems, also more expressive formalisms like disjunctive modal transition systems [16] and transition systems with obligations [5]. See [6] for a more thorough discussion of this formalism.

In the rest of the paper, we adapt the following convention when drawing MTSDs. Whenever a state s is connected with a solid arrow labelled by a to a state s', this means that in any satisfying assignment of the Boolean formula $\Phi(s)$, the atomic proposition (a, s') is always set to true (the transition *must* be present in any refinement of the system). Should this not be the case, we use a dashed arrow instead (meaning that the corresponding transition *may* be present in a refinement of the system but it can be also left out). For example the solid edges in Figure 1a correspond to an implicitly assumed $\Phi(s) = (a, s')$ where (s, a, s') is the (only) outgoing edge from s; in this case we do not explicitly write the obligation function. The three dashed transitions in the figure are optional, though at least one of them has to be preserved during any refinement (a feature that can be modelled for example in disjunctive MTS [16]).

Remark 2.2. *The standard notion of modal transition systems (see e.g. [2]) is obtained under the restriction that the formulae* $\Phi(s)$ *in any state* $s \in S$ *have the form* $(a_1, s_1) \wedge \ldots \wedge (a_n, s_n)$ *where* $(s, a_1, s_1), \ldots, (s, a_n, s_n) \in T$*. The edges mentioned in such formulae are exactly all must transitions; may transitions are not listed in the formula and hence can be arbitrarily set to true or false.*

Let by $T(s) = \{(a, t) \mid (s, a, t) \in T\}$ denote the set of all outgoing transitions from the state $s \in S$. A modal transition system with durations is called an *implementation* if

$\Phi(s) = \bigwedge T(s)$ for all $s \in S$ (every allowed transition is also required), and $D(s, a, s') \in \mathcal{I}_u$ for all $(s, a, s') \in T$, i.e. all intervals are uncontrollable, often singletons. Figure 1c shows an example of an implementation, while Figure 1b is not yet an implementation as it still contains the controllable intervals $\langle 3, 5\rangle$ and $\langle 4, 6\rangle$.

We now define a notion of modal refinement. In order to do that, we first need to define refinement of intervals as a binary relation $\leq \subseteq \mathfrak{I} \times \mathfrak{I}$ such that

- $\langle m', n'\rangle \leq \langle m, n\rangle$ whenever $m' \geq m$ and $n' \leq n$, and
- $[m', n'] \leq \langle m, n\rangle$ whenever $m' \geq m$ and $n' \leq n$.

Thus controllable intervals can be refined by narrowing them, at most until they become singleton intervals, or until they are changed to uncontrollable intervals. Let us denote the collection of all possible sets of outgoing transitions from a state s by $\mathrm{Tran}(s) := \{E \subseteq T(s) \mid E \models \Phi(s)\}$ where $\models$ is the classical satisfaction relation on propositional formulae assuming that E lists all true propositions.

Definition 2.3 (Modal Refinement). *Let* $\mathcal{S}_1 = (S_1, T_1, D_1, \Phi_1, s_1)$ *and* $\mathcal{S}_2 = (S_2, T_2, D_2, \Phi_2, s_2)$ *be two MTSDs. A binary relation* $R \subseteq S_1 \times S_2$ *is a* modal refinement *if for every* $(s, t) \in R$ *the following holds:*

$$\begin{aligned}
&\forall M \in \mathrm{Tran}(s) : \exists N \in \mathrm{Tran}(t) : \\
&\qquad \forall (a, s') \in M : \exists (a, t') \in N : D_1(s, a, s') \leq D_2(t, a, t') \ \wedge\ (s', t') \in R \textit{ and} \\
&\qquad \forall (a, t') \in N : \exists (a, s') \in M : D_1(s, a, s') \leq D_2(t, a, t') \ \wedge\ (s', t') \in R\,.
\end{aligned}$$

We say that $s \in S_1$ *modally refines* $s' \in S_2$*, denoted by* $s \leq_m s'$*, if there exists a modal refinement* R *such that* $(s, s') \in R$*. We also write* $\mathcal{S}_1 \leq_m \mathcal{S}_2$ *if* $s_1 \leq_m s_2$*.*

Intuitively, the pair (s, t) can be in the relation R if for any satisfiable instantiation of outgoing edges from s there is a satisfiable instantiation of outgoing edges from t so that they can be mutually matched, possibly with s having more refined intervals, and the resulting states are again in the relation R.

Observe that in our running example the following systems are in modal refinement: $\mathcal{I}_1 \leq_m \mathcal{S}_1 \leq_m \mathcal{S}$ and thus also $\mathcal{I}_1 \leq_m \mathcal{S}$, and similarly $\mathcal{I}_2 \leq_m \mathcal{S}_2 \leq_m \mathcal{S}$ and thus also $\mathcal{I}_2 \leq_m \mathcal{S}$.

The reader can verify that on the standard modal transition systems (see Remark 2.2) the modal refinement relation corresponds to the classical modal refinement as introduced in [15].

3 Dual-Price Scheme

In this section, we formally introduce a dual-price scheme on top of MTSD in order to model the *investment cost* (cost of hardware necessary to perform the implemented actions) and the *running cost* (weighted long-run average of running costs of actions). We therefore consider only deadlock-free implementations (every state has at least one outgoing transition) so that the long-run average reward is well defined.

Definition 3.1 (Dual-Price Scheme). *A* dual-price scheme *over an alphabet* Σ *is a tuple* $\mathcal{P} = (r, H, \Psi, i)$ *where*

- $r : \Sigma \to \mathbb{Z}$ *is a* running cost *function of actions per time unit,*
- H *is a finite set of available* hardware,
- $\Psi : \Sigma \to \mathcal{B}(H)$ *is a* hardware requirement *function, and*
- $i : H \to \mathbb{N}_0$ *is a hardware* investment cost *function.*

Hence every action is assigned its unit cost and every action can have different hardware requirements (specified as a Boolean combination of hardware components) on which it can be executed. This allows for much more variability than a possible alternative of a simple investment cost $\Sigma \to \mathbb{N}_0$. Further, observe that the running cost may be negative, meaning that execution of such an action actually gains rather than spends resources.

Let $\mathcal{I}$ be an implementation with an initial state s_0. A set $G \subseteq H$ of hardware is *sufficient* for an implementation $\mathcal{I}$, written $G \models \mathcal{I}$, if $G \models \Psi(a)$ for every action a reachable from s_0. The *investment cost* of $\mathcal{I}$ is then defined as

$$\mathsf{ic}(\mathcal{I}) = \min_{G \models \mathcal{I}} \sum_{g \in G} i(g) \ .$$

Further, a *run* of $\mathcal{I}$ is an infinite sequence $s_0 a_0 t_0 s_1 a_1 t_1 \cdots$ with $(s_i, a_i, s_{i+1}) \in T$ and $t_i \in D(s_i, a_i, s_{i+1})$. Hence, in such a run, a concrete time duration in each uncontrollable interval is selected. We denote the set of all runs of $\mathcal{I}$ by $\mathcal{R}(\mathcal{I})$. The *running cost* of an implementation $\mathcal{I}$ is the worst-case long-run average

$$\mathsf{rc}(\mathcal{I}) = \sup_{s_0 a_0 t_0 s_1 a_1 t_1 \cdots \in \mathcal{R}(\mathcal{I})} \limsup_{n \to \infty} \frac{\sum_{i=0}^{n} r(a_i) \cdot t_i}{\sum_{i=0}^{n} t_i} \ .$$

Our *cheapest-implementation problem* is now defined as follows: given an MTSD specification $\mathcal{S}$ together with a dual-price scheme over the same alphabet, and given an upper-bound max_{ic} for the investment cost, find an implementation $\mathcal{I}$ of $\mathcal{S}$ (i.e. $\mathcal{I} \leq_{\mathrm{m}} \mathcal{S}$) such that $\mathsf{ic}(\mathcal{I}) \leq max_{\mathsf{ic}}$ and for every implementation $\mathcal{I}'$ of $\mathcal{S}$ with $\mathsf{ic}(\mathcal{I}') \leq max_{\mathsf{ic}}$, we have $\mathsf{rc}(\mathcal{I}) \leq \mathsf{rc}(\mathcal{I}')$.

Further, we introduce the respective decision problem, the *implementation problem*, as follows: given an MTSD specification $\mathcal{S}$ together with a dual-price scheme, and given an upper-bound max_{ic} for the investment cost and an upper bound max_{rc} on the running cost, decide whether there is an implementation $\mathcal{I}$ of $\mathcal{S}$ such that both $\mathsf{ic}(\mathcal{I}) \leq max_{\mathsf{ic}}$ and $\mathsf{rc}(\mathcal{I}) \leq max_{\mathsf{rc}}$.

Example 3.2. *Figure 1f depicts a dual-price scheme over the same alphabet* $\Sigma = \{\mathtt{Wait}, \mathtt{Drive}, \mathtt{SmallCleanup}, \mathtt{BigCleanup}, \mathtt{SkipCleanup}\}$ *as of our motivating specification* $\mathcal{S}$. *The running cost of the implementation* $\mathcal{I}_2$ *is* $(1 \cdot 8 + 10 \cdot 10 + 6 \cdot 6 + 1 \cdot 8 + 10 \cdot 10 + 30 \cdot 7)/(1 + 10 + 6 + 1 + 10 + 30) \approx 7.97$ *as the maximum value is achieved when* $\mathtt{Drive}$ *(with running cost 10) takes 10 minutes. On the one hand, this is optimal for* $\mathcal{S}$ *and a maximum investment cost at least 100. On the other hand, if the maximum investment cost is 99 or less then the optimal implementation is depicted in* $\mathcal{I}_1$ *and its cost is* $(5 \cdot 8 + 10 \cdot 10 + 6 \cdot 5)/(5 + 10 + 6) \approx 8.10$.

Remark 3.3. *Note that the definition of the dual-price scheme only relies on having durations on the labelled transition systems. Hence, one could easily apply this in various other settings like in the special case of traditional MTS (with may and must transitions instead of the obligation function) or in the more general case of parametric MTS (see [6]) when equipped with durations as described above.*

4 Complexity Results

In this section, we give an overview of the complexity of our problem both in general and in an important special case. We start with establishing the hardness results. The matching upper bounds and the outline of their proofs follow. When referring to the size of MTSDs and the dual-price scheme, we implicitly assume binary encoding of numbers. We start by observing that the implementation problem is NP-hard even if no hardware is involved.

Proposition 4.1. *The implementation problem is NP-hard even for the hardware requirement function* Ψ *that is constantly true for all actions.*

Proof. We shall reduce the satisfiability problem of Boolean formulae (SAT) to our problem. Let φ be a Boolean formula over the variables $x_1, \ldots, x_n$. We define a MTSD $\mathcal{S}$

over the set of actions $\Sigma = \{x_1, \ldots, x_n, *\}$ such that the running cost is $r(x_j) = 1$ for all $1 \leq j \leq n$ and $r(*) = 2$ and the duration of all actions is 1. The specification $\mathcal{S}$ has one state s and a self-loop under all elements of Σ with the obligation function $\Phi(s) = \varphi \vee (*, s)$. The reason for adding the action $*$ is to make sure that in case φ is not satisfiable then we can still have a deadlock-free, but more running-cost-expensive implementation. Now we set the hardware to $H = \emptyset$ and the hardware requirement function $\Psi(a)$ constantly true for all $a \in \Sigma$. It is easy to observe that the formula φ is satisfiable iff $\mathcal{S}$ has an implementation $\mathcal{I}$ with $\mathsf{rc}(\mathcal{I}) \leq 1$ (and $\mathsf{ic}(\mathcal{I}) = 0$). $\square$

Note that in the proof we required Φ to be a general Boolean formula. If, for instance, we considered Φ in *positive form* (i.e. only containing $\wedge$ and $\vee$ operators and not $\neg$), the hardness would not hold. Thus on the one hand, one source of hardness is the complexity of Φ. On the other hand, even if Φ corresponds to the simplest case of an implementation (Φ is a conjunction of atomic propositions), the problem remains hard due to the hardware.

Proposition 4.2. *The implementation problem is NP-hard even for specifications that are already implementations.*

Proof. We reduce the NP-complete problem of vertex cover to our problem. Let (V, E) where $E \subseteq V \times V$ be a graph and $k \in \mathbb{N}$ be an integer. We ask whether there is a subset of vertices $V_k \subseteq V$ of cardinality k such that for every $(v_1, v_2) \in E$ at least $v_1 \in V_k$ or $v_2 \in V_k$. Let us construct an MTSD specification $\mathcal{S}$ with hardware $H = V$ and the investment function $i(v) = 1$ for all $v \in H$, such that $\mathcal{S}$ has only one state s and a self-loop under a single action a that is required ($\Phi(s) = (a, s)$) and where the hardware requirement function is $\Psi(a) = \bigwedge_{(u,v) \in E}(u \vee v)$. There is now a vertex cover in (V, E) of size k iff $\mathcal{S}$ has an implementation $\mathcal{I}$ with $\mathsf{ic}(\mathcal{I}) \leq k$. Setting e.g. $D(s, a, s) = 1$ and the running cost $r(a) = 0$ establishes NP-hardness of the implementation problem where we ask for the existence of an implementation of $\mathcal{S}$ with maximum running cost 0 and maximum investment cost k.

Alternatively, we may introduce a self-loop with a new action name $a_{(u,v)}$ for every edge (u, v) in the graph such that $\Psi(a_{(u,v)}) = u \vee v$, showing NP-hardness even for the case where the hardware requirement function is a simple disjunction of hardware components. $\square$

In the subsequent sections, we obtain the following matching upper bound which yields the following theorem.

Theorem 4.3. *The implementation problem is NP-complete.*

By analysing the proof of Proposition 4.2, it is clear that we have to restrict the hardware requirement function before we can obtain a more efficient algorithm for the implementation problem. We do so by assuming a constant number of hardware components (not part of the input). If we at the same time require the obligation function in positive form, we obtain a simpler problem as stated in the following theorem.

Theorem 4.4. *The implementation problem with positive obligation function and a constant number of hardware components is polynomially equivalent to mean payoff games and thus it is in NP $\cap$ coNP and solvable in pseudo-polynomial time.*

The subsequent sections are devoted to proving Theorems 4.3 and 4.4. The algorithm to solve the implementation problem first reduces the dual-priced MTSD into a mean payoff game extended with time durations and then solves this game. This new extension of mean payoff games and an algorithm to solve them is presented in Section 4.1. The translation follows in Section 4.2. Since this translation is exponential in general, Section 4.3 then shows how to translate in polynomial time with only local exponential blow-ups where negations occur. Section 4.4 then concludes and establishes the complexity bounds.

4.1 Weighted Mean Payoff Games

We extend the standard model of mean payoff games (MPG) [13] with time durations. Not only is this extension needed for our algorithm, but it is also useful for modelling by itself. Consider, for instance, energy consumption of 2kW for 10 hours and 10kW for 2 hour, both followed by 10 hours of inactivity. Obviously, although both consumptions are 20kWh per cycle, the average consumption differs: 1kW in the former case and 20/12kW in the latter one. We also allow zero durations in order to model e.g. discrete changes of states, an essential part of our algorithm. Another extension of MPGs with dual-cost was studied in [7].

Definition 4.5. *A* weighted mean payoff game *is* $G = (V, V_{min}, V_{max}, E, r, d)$ *where* V *is a set of vertices partitioned into* V_{min} *and* V_{max}*,* $E \subseteq V \times V$ *is a set of edges,* $r : E \to \mathbb{Z}$ *is a rate function,* $d : E \to \mathbb{N}_0$ *is a duration function.*

It is assumed that there are no deadlocks (vertices with out-degree 0) and that there are no zero-duration cycles. The game is played by two players, min and max. The play is an infinite path such that each player picks successors in his/her vertices. The value of a play $v_0v_1v_2\cdots$ is defined as:

$$\nu(v_0v_1v_2\cdots) = \limsup_{n\to\infty} \frac{\sum_{i=0}^{n} r(v_i, v_{i+1}) \cdot d(v_i, v_{i+1})}{\sum_{i=0}^{n} d(v_i, v_{i+1})} . \qquad (*)$$

Player min tries to minimize this value, while max aims at the opposite. Let $\nu(s)$ denote the infimum of the values min can guarantee if the play begins in the vertex s, no matter what the player max does.

Note that the standard MPGs where edges are assigned only integer weights can be seen as weighted MPGs with rates equal to weights and durations equal to 1 on all edges.

We now show how to solve weighted MPGs by reduction to standard MPGs. We first focus on the problem whether $\nu(s) \geq 0$ for a given vertex s. As the durations are nonnegative and there are no zero-duration cycles, the denominator of the fraction in $(*)$ will be positive starting from some n. Therefore, the following holds for every play $v_0v_1v_2\ldots$ and every (large enough) n:

$$\frac{\sum_{i=0}^{n} r(v_i, v_{i+1}) \cdot d(v_i, v_{i+1})}{\sum_{i=0}^{n} d(v_i, v_{i+1})} \geq 0 \iff \frac{1}{n}\sum_{i=0}^{n} r(v_i, v_{i+1}) \cdot d(v_i, v_{i+1}) \geq 0 .$$

We may thus solve the question whether $\nu(s) \geq 0$ by transforming the weighted MPG into a standard MPG, leaving the set of vertices and edges the same and taking $w(u, v) = r(u, v) \cdot d(u, v)$ as the edge weight function. Although the value $\nu(s)$ may change in this reduction, its (non)negativeness does not.

Further, we may transform any problem of the form $\nu(s) \geq \lambda$ for any fixed constant λ into the above problem. Let us modify the weighted MPG as follows. Let $r'(u, v) = r(u, v) - \lambda$ and leave everything else the same. The value of a play $v_0v_1v_2\cdots$ is thus changed as follows.

$$\nu'(v_0v_1v_2\cdots) = \limsup_{n\to\infty} \frac{\sum_{i=0}^{n}(r(v_i, v_{i+1}) - \lambda) \cdot d(v_i, v_{i+1})}{\sum_{i=0}^{n} d(v_i, v_{i+1})} = \nu(v_0v_1v_2\cdots) - \lambda$$

It is now clear that $\nu(s) \geq \lambda$ in the original game if and only if $\nu'(s) \geq 0$ in the modified game.

Furthermore, there is a one-to-one correspondence between the strategies in the original weighted MPG and the constructed MPG. Due to the two equivalences above, this correspondence preserves optimality. Therefore, there are optimal positional strategies in weighted MPGs since the same holds for standard MPGs [13]. (A strategy is positional if its decision does not depend on the current history of the play but only on the current vertex, i.e. can be described as a function $V \to V$.)

4.2 Translating Dual-priced MTSD into Weighted MPG

We first focus on the implementation problem without considering the hardware ($H = \emptyset$). We show how the implementation problem can be solved by reduction to the weighted MPGs. The first translation we present is exponential, however, we provide methods for making it smaller in the subsequent section.

We are given an MTSD $\mathcal{S} = (S, T, D, \Phi, s_0)$ and a dual-price scheme (r, H, Ψ, i) and assume that there is no state s with $\emptyset \in \mathrm{Tran}(s)$. Let us define the following auxiliary vertices that will be used to simulate the more complicated transitions of MTSD in the simpler setting of weighted MPG (by convention all singleton intervals are treated as uncontrollable).

$$T_u = \{(s, a, t) \mid (s, a, t) \in T;\ D(s, a, t) \in \mathcal{I}_u\}$$
$$T_c = \{(s, a, t) \mid (s, a, t) \in T;\ D(s, a, t) \in \mathcal{I}_c\}$$
$$T_* = \{(s, a, j, t) \mid (s, a, t) \in T;\ j \in D(s, a, t)\}$$

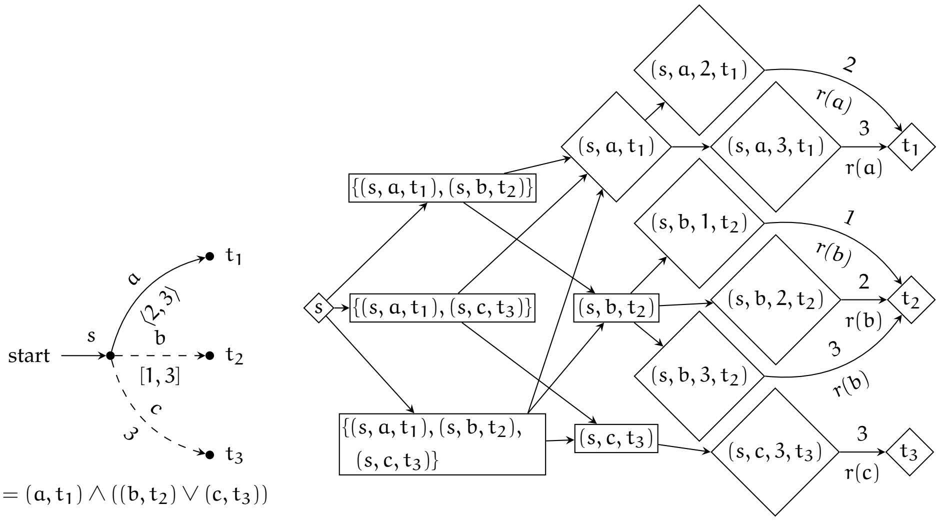
We construct the weighted mean-payoff game with $V_{min} = S \cup T_c \cup T_*$, $V_{max} = 2^T \cup T_u$ and E defined as follows:

$$\begin{aligned}
(s, X) \in E \quad &\Longleftrightarrow \quad \exists V \in \mathrm{Tran}(s) : X = \{(s, a, t) \mid (a, t) \in V\} \\
(X, (s, a, t)) \in E \quad &\Longleftrightarrow \quad (s, a, t) \in X \\
((s, a, t), (s, a, j, t)) \in E \quad &\Longleftrightarrow \quad j \in D(s, a, t) \\
((s, a, j, t), t) \in E \quad &\quad \text{(always)}
\end{aligned}$$

Further, $r((s, a, j, t), t) = r(a)$, $d((s, a, j, t), t) = j$ and $r(-, -) = d(-, -) = 0$ otherwise.

Example 4.6. *In Figure 2 we show an example of how this translation to weighted MPG works. For simplicity we only translate a part of the MTSD $\mathcal{S}$ shown in Figure 2a. The resulting weighted MPG is shown in Figure 2b. The diamond shaped states belong to min and the squared states belong to max. In the vertex s, min chooses which outgoing transition are implemented. Only the choices satisfying $\Phi(s)$ are present in the game. Afterwards, max decides which transition to take. The chosen transition is then assigned by one of the players a time that it is going to take.*

Notice that (s, a, t_1) is the only transition controlled by min, because it has a controllable interval $\langle 2, 3\rangle$. The remaining transitions with uncontrollable intervals are operated by max who chooses the time from these intervals. All the "auxiliary" transitions are displayed without



(a) A part of an MTSD S (b) Resulting weighted MPG from translation of S

Figure 2: Translating MTSD to weighted MPG

any labels meaning their duration (and rate) is zero. Thus, only the transitions corresponding to "real" transitions in MTSDs are taken into account in the value of every play.

We now show how a strategy for player *min* in the constructed weighted MPG may be translated into an implementation of the original MTSD. In Section 4.1, we have shown that there are optimal *positional* strategies. Hence, we may safely restrict this translation to positional strategies. Let σ be such a positional strategy. We build the implementation as $\mathcal{I} = (S', T', D', \Phi', s_{0I})$ where

- $S' = \{s_I \mid s \in S\}$
- $(s_I, a, t_I) \in T'$ if $(s, a, t) \in X$ where $\sigma(s) = X$
- $D'(s_I, a, t_I) = D(s, a, t)$ if $D(s, a, t) \in \mathcal{I}_u$
- $D'(s_I, a, t_I) = j$ if $D(s, a, t) \in \mathcal{I}_c$ and $\sigma((s, a, t)) = (s, a, j, t)$
- $\Phi'(s_I) = \bigwedge_{(s_I, a, t_I) \in T'} (a, t_I)$

The set of states remains the same as in the original MTSD; we change every state s into s_I to be able to reason about the states of the original MTSD and of the implementation separately. The following two lemmas state that the construction is sound.

Lemma 4.7. *The constructed implementation $\mathcal{I}$ is an implementation of the original MTSD.*

Proof. We show that $R = \{(s_I, s) \mid s \in S\}$ is a modal refinement relation. Let $(s_I, s) \in R$ and let $M \in \mathrm{Tran}(s_I)$. Clearly $M = \{(a, t_I) \mid (s_I, a, t_I) \in T'\}$. We take $N = \{(a, t) \mid (a, t_I) \in M\}$. The fact that $N \in \mathrm{Tran}(s)$ is clear from the construction. We now need to show that $D'(s_I, a, t_I) \leq D(s, a, t)$. If $D(s, a, t) \in \mathcal{I}_u$ then $D'(s_I, a, t_I) = D(s, a, t)$ and the statement holds. If $D(s, a, t) \in \mathcal{I}_c$ then $D'(s_I, a, t_I) = j$ where $j \in D(s, a, t)$ due to the construction. Thus R is a modal refinement relation where $s_I \leq_m s$ for all $s \in S$ and therefore $\mathcal{I} \leq_m \mathcal{S}$. □

Lemma 4.8. *We have* $\mathsf{rc}(\mathcal{I}) \leq \lambda$ *if and only if the strategy* σ *ensures a value of at most* λ.

Proof. Every run of the implementation corresponds to a play (where player *min* plays according to strategy σ) and vice versa. Hence, the worst case of the long run average over all runs in $\mathcal{I}$ is the same as over all plays according to σ. □

We conclude the proof of correctness of the reduction by showing its completeness.

Lemma 4.9. *For every implementation* $\mathcal{I}$ *of* $\mathcal{S}$*, there exists a strategy* σ *for player* min *such that* σ *ensures value of at most* $\mathsf{rc}(\mathcal{I})$.

Proof. We show how to transform an arbitrary implementation $\mathcal{I}$ into a strategy $\sigma : V^* \to V$ (depending on the whole prefix of a play where we currently are) for player *min* that guarantees the same or smaller value. (Although this is not a positional strategy, we know there is also a positional strategy ensuring the same or smaller value.) The idea of the construction is that for each history σ mimics the behaviour of the implementation at its respective state. In other words, the decision of σ in some history is based on mapping the history to the implementation and doing what the implementation does. However, there is a small catch: the implementation may implement more possible decisions (its branching may be even uncountable). Nonetheless, we may take any of the decisions, as the resulting strategy then corresponds to a pruning of the original implementation and the strategy's worst case long run average is thus either the same or even smaller.

Let i_0 be the initial state of $\mathcal{I}$ and s_0 the initial state of $\mathcal{S}$; we have $i_0 \leq_m s_0$. The corresponding vertex of the constructed weighted MPG also bears the name s_0. We first define a mapping $\mu : V^* \to I^*$ from paths in the weighted MPG of the form $s_0\,(s_0, N_0)\,(s_0, a_0, s_1)\,(s_0, a_0, j_0, s_1)\,s_1 \cdots s_n$ to sequences of states of $\mathcal{I}$ inductively as follows.

$$\mu(s_0) = i_0$$
$$\mu(s_0 \cdots s_n\,(s_n, N_n)\,(s_n, a_n, s_{n+1})\,(s_n, a_n, j_n, s_{n+1})\,s_{n+1}) = \mu(s_0 \cdots s_n) i_{n+1}$$

where i_{n+1} is an arbitrary state satisfying $(i_n, a_n, i_{n+1}) \in T$ and $i_{n+1} \leq_m s_{n+1}$. The image of μ now defines the desired pruning of $\mathcal{I}$ that is still an implementation of $\mathcal{S}$, has at most the same worst case long run average, and can now be canonically mapped to a deterministic strategy σ as follows.

Let $\pi = s_0\,(s_0, N_0)\,(s_0, a_0, s_1)\,(s_0, a_0, j_0, s_1)\,s_1 \cdots x \in V^*$ be a prefix of a play with $x \in V_{min}$. Denote s_n the last element of $\mathcal{S}$ in π and $i_0 \cdots i_n = \mu(s_0 \cdots s_n)$. We define $\sigma(\pi)$ as follows.

- If $x \in S$, then $\pi = s_0 \cdots s_n$ and due to $i_n \leq_m s_n$ there exists N from the definition of modal refinement. We set $\sigma_s(\pi) = (s_n, N)$.
- If $x \in T_c$, then $\pi = s_0 \cdots s_n\,(s_n, N_n)\,(s_n, a_n, s_{n+1})$ and there exists a unique i_{n+1} such that $i_0 \cdots i_n i_{n+1} = \mu(s_0 \cdots s_n s_{n+1})$. Here we use the fact that μ does not depend on which vertices of the form (s, a, j, t) were visited. We set $\sigma(\pi) = (s_n, a_n, D(i_n, a_n, i_{n+1}), s_{n+1})$.
- If $x \in T_*$, then $x = (s_n, a_n, j_n, s_{n+1})$ and there exists only one outgoing edge. We set $\sigma(\pi) = s_{n+1}$.

Now for each play that player *min* plays according to σ, there is a run in the original implementation that has the same long-run average. Hence the supremum over all plays is at most the supremum over all runs. □

4.3 Optimizations

We now simplify the construction. The first simplification is summarized by the observation that the strategies of both players only need to choose the extremal points of the interval in vertices of the form (s, a, t).

Lemma 4.10. *There are optimal positional strategies σ', ρ' for* min *and* max*, respectively, such that the choice in vertices of the form (s, a, t) is always one of the two extremal points of the interval $D(s, a, t)$.*

Proof. Let σ and ρ be optimal positional strategies for *min* and *max*. We transform them into σ' and π'. The proof is done by induction on the number of vertices of the form (s, a, t) from which one of the strategies chooses a non-extremal point of the interval $D(s, a, t)$.

If there are no such vertices, we are obviously done. Suppose further that there is at least one such vertex, say (s, a, t). Without loss of generalization, let this vertex be player *min*'s vertex. (The other case is handled similarly.) We thus have $D(s, a, t) = \langle m, n\rangle$ and $\sigma((s, a, t)) = (s, a, j, t)$ with $m < j < n$.

We investigate three cases, depending on the relationship of the rate $r(a)$ and the value $v(s)$.

- $r(a) = v(s)$. Then the duration of the transition (s, a, t) does not matter and we may freely change σ into σ' by defining $\sigma'((s, a, t)) = (s, a, n, t)$ and $v(s)$ remains the same.

- $r(a) > v(s)$. Clearly, changing σ into σ' by defining $\sigma'((s, a, t)) = (s, a, m, t)$ may only decrease $v(s)$ or not change it at all. (As σ is optimal strategy, $v(s)$ remains the same using σ'.)

- $r(a) < v(s)$. Using similar argument as in the previous case, we change σ into σ' by defining $\sigma'(s, a, t) = (s, a, n, t)$ and σ' remains optimal.

Repeated use of this argument concludes the proof. $\square$

We may thus simplify the construction according to the previous lemma so that there are at most two outgoing edges for each state of the form (s, a, t) are as follows: $((s, a, t),(s, a, j, t)) \in E$ iff j is an extremal point of $D(s, a, t)$.

We can also optimize the expansion of $\mathrm{Tran}(s)$. So far, we have built an exponentially larger weighted MPG graph as the size of $\mathrm{Tran}(s)$ is exponential in the out-degree of s. However, we can do better if we restrict ourselves to the class of MTSD where all $\Phi(s)$ are positive boolean formulae, i.e. the only connectives are $\wedge$ and $\vee$. Instead of enumerating all valuations, we can use the syntactic tree of the formula to build a weighted MPG of polynomial size.

Let $sf(\varphi)$ denote the set of all sub-formulae of φ (including φ). Let further $S_* = \{(s, \varphi) \mid s \in S;\ \varphi \in sf(\Phi(s))\}$. The weighted MPG is constructed with

- $V_{min} = \{(s, \varphi) \in S_* \mid \varphi = \varphi_1 \vee \varphi_2 \text{ or } (\varphi = (a, t) \text{ and } D(s, a, t) \in \mathcal{I}_c)\} \cup T_*$

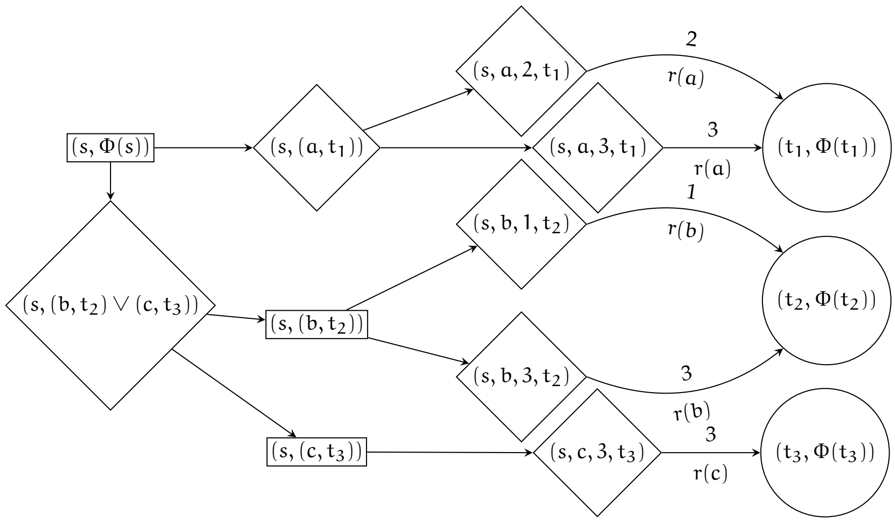


Figure 3: Result of the improved translation of S in Figure 2a

- $V_{max} = \{(s, \varphi) \in S_* \mid \varphi = \varphi_1 \wedge \varphi_2 \text{ or } (\varphi = (a, t) \text{ and } D(s, a, t) \in \mathcal{I}_u)\}$
- E is defined as follows:

$$\begin{aligned}
((s, \varphi_1 \wedge \varphi_2), (s, \varphi_i)) \in E \qquad & i \in \{1, 2\} \\
((s, \varphi_1 \vee \varphi_2), (s, \varphi_i)) \in E \qquad & i \in \{1, 2\} \\
((s, (a, t)), (s, a, j, t)) \in E \qquad & \iff j \text{ is an extremal point of } D(s, a, t) \\
((s, a, j, t), (t, \Phi(t))) \in E \qquad & \text{(always)}
\end{aligned}$$

- $r((s, a, j, t), (t, \Phi(t))) = r(a)$ and $r(-, -) = 0$ otherwise
- $d((s, a, j, t), (t, \Phi(t))) = j$ and $d(-, -) = 0$ otherwise.

Example 4.11. *In Figure 3 we show the result of translating the part of an MTSD from Figure 2a. This weighted MPG is similar to the one in Figure 2b, but instead of having a vertex for each satisfying set of outgoing transitions, we now have the syntactic tree of the obligation formula for each state. Further the vertex* $(s, b, 2, t_2)$ *is left out, due to Lemma 4.10. Note that the vertices* $(t_1, \Phi(t_1))$, $(t_2, \Phi(t_2))$ *and* $(t_3, \Phi(t_3))$ *are drawn as circles, because the player of these states depends on the obligation formula and the outgoing transitions.*

Remark 4.12. *Observe that one can perform this optimization even in the general case. Indeed, for those* s *where* $\Phi(s)$ *is positive we locally perform this transformation; for* s *with* $\Phi(s)$ *containing negations we stick to the original expansion. Thus, the exponential (in out-degree) blow-up occurs only locally.*

Lemma 4.13. *Both optimized translations are correct and on MTSDs where the obligation function is positive they run in polynomial time.*

Proof. A strategy for the player min of a weighted MPG from a translation utilizing the optimizations of Section 4.3 can be translated into an implementation of the original MTSD as follows. Let σ be the strategy. We first define an auxiliary function on the vertices of the MPG as follows:

$$\begin{aligned} f(s,(a,t)) &= \{(a,t)\} \\ f(s,\varphi \vee \psi) &= f(\sigma(s,\varphi \vee \psi)) \\ f(s,\varphi \wedge \psi) &= f(s,\varphi) \cup f(s,\psi) \end{aligned}$$

We then build the implementation as (S', T', D', Φ') where

- $S' = \{s_I \mid s \in S\}$
- $(s_I, a, t_I) \in T'$ if $(a,t) \in f(s, \Phi(s))$
- $D'(s_I, a, t_I) = D(s,a,t)$ if $D(s,a,t) \in \mathfrak{I}_u$
- $D'(s_I, a, t_I) = j$ if $D(s,a,t) \in \mathfrak{I}_c$ and $\sigma((s,(a,t))) = (s,a,j,t)$
- $\Phi'(s) = \bigwedge_{(s_I,a,t_I)\in T'} (a, t_I)$.

The fact that the constructed implementation is indeed an implementation of the given MTSD and that the running cost of the implementation starting from state s_I is the same as $\nu((s,\Phi(s)))$ is straightforward and can be proved as in the previous case.

Similarly, one can derive a strategy from an implementation so that its value does not get worse. We may safely assume that the implementation only implements durations as the extremal points of the controllable intervals. (Indeed, due to the original translation, Lemma 4.10, and the transformation back of Lemma 4.9, one can obtain an implementation with only extremal points that have the same or smaller cost.) The new transformation (strategy σ and mapping μ) is the same as the previous one of Lemma 4.9, the difference is that for history π ending in $(s, \varphi_1 \vee \varphi_2)$ we set $\sigma(\pi) = (s, \varphi_j)$ for an arbitrary $j \in \{1,2\}$ such that $T(i_n) \models \varphi_j$ where $\mu(\pi') = i_0 \cdots i_n$ and π' is the longest prefix of π ending with the vertex $(s, \Phi(s))$.

The polynomial running time is clear from the construction. □

4.4 The Algorithm and its Complexity

The algorithm for our problem, given a specification $\mathcal{S}$, works as follows.

1. Nondeterministically choose hardware with the total price at most max_{ic}.
2. Create the weighted MPG out of $\mathcal{S}$.
3. Solve the weighted MPG using the reduction to MPG and any standard algorithm for MPG that finds an optimal strategy for player *min* and computes the value $\nu(s_0)$.
4. Transform the strategy to an implementation $\mathcal{I}$.
5. In the case of the cheapest-implementation problem return $\mathcal{I}$;
   in the case of the implementation (decision) problem return $\nu(s_0) \leq max_{\mathsf{rc}}$.

We can now prove the following result, finishing the proof of Theorem 4.3.

Proposition 4.14. *The implementation problem is in NP.*

Proof. We first nondeterministically guess the hardware assignment. Due to Section 4.2, we know that the desired implementation has the same states as the original MTSD and its transitions are a subset of the transitions of the original MTSD as the corresponding optimal strategies are positional. The first optimization (Section 4.3) guarantees that durations can be chosen as the extremal points of the intervals. Thus we can nondeterministically guess an optimal implementation and its durations, and verify that it satisfies the price inequality. □

Proposition 4.15. *The implementation problem for MTSD with positive obligation function and a constant number of hardware components is in NP ∩ coNP and solvable in pseudo-polynomial time.*

Proof. With the constant number of hardware components, we get a constant number of possible hardware configurations and we can check each configuration separately one by one. Further, by the first and the second optimization in Section 4.3, the MPG graph is of size $\mathcal{O}(|\mathsf{T}| + |\Phi|)$. Therefore, we polynomially reduce the implementation problem to the problem of solving constantly many mean payoff games. The result follows by the existence of pseudo-polynomial algorithms for MPGs [17]. □

Further, our problem is at least as hard as solving MPGs that are clearly a special case of our problem. Hence, Theorem 4.4 follows.

5 Conclusion and Future Work

We have introduced a new extension of modal transition systems. The extension consists in introducing (1) variable time durations of actions and (2) pricing of actions, where we combine one-shot investment price for the hardware and cost for running it per each time unit it is active. We believe that this formalism is appropriate to modelling many types of embedded systems, where safety comes along with economical requirements.

We have solved the problem of finding the cheapest implementation w.r.t. the running cost given a maximum hardware investment we can afford, and we established the complexity of the decision problem in the general setting and in a practically relevant subcase revealing a close connection with mean payoff games.

As for the future work, apart from implementing the algorithm, one may consider two types of extensions. First, one can extend the formalism to cover the distinction between input, output and internal actions as it is usual in interface theories [11], and include even more time features, such as clocks in priced timed automata [4, 9]. Second, one may extend the criteria for synthesis of the cheapest implementation by an additional requirement that the partial sums stay within given bounds as done in [10], or requiring the satisfaction of a temporal property as suggested in [11, 12].

References

[1] Aceto, L., Fábregas, I., de Frutos-Escrig, D., Ingólfsdóttir, A., Palomino, M.: Graphical representation of covariant-contravariant modal formulae. In: EXPRESS. EPTCS, vol. 64, pp. 1–15 (2011)

[2] Antonik, A., Huth, M., Larsen, K.G., Nyman, U., Wasowski, A.: 20 years of modal and mixed specifications. Bulletin of the EATCS no. 95 pp. 94–129 (2008)

[3] Bauer, S.S., Fahrenberg, U., Juhl, L., Larsen, K.G., Legay, A., Thrane, C.R.: Quantitative refinement for weighted modal transition systems. In: MFCS. LNCS, vol. 6907, pp. 60–71. Springer (2011)

[4] Behrmann, G., Larsen, K.G., Rasmussen, J.I.: Priced timed automata: Algorithms and applications. In: FMCO. LNCS, vol. 3657, pp. 162–182. Springer (2004)

[5] Beneš, N., Křetínský, J.: Process algebra for modal transition systemses. In: MEMICS. OASICS, vol. 16, pp. 9–18. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany (2010)

[6] Beneš, N., Křetínský, J., Larsen, K., Møller, M., Srba, J.: Parametric modal transition systems. In: Proceedings of ATVA'11. LNCS, vol. 6996, pp. 275–289. Springer-Verlag (2011)

[7] Bloem, R., Greimel, K., Henzinger, T.A., Jobstmann, B.: Synthesizing robust systems. In: Proc. of FMCAD09. pp. 85–92. IEEE (2009)

[8] Boudol, G., Larsen, K.G.: Graphical versus logical specifications. Theor. Comput. Sci. 106(1), 3–20 (1992)

[9] Bouyer, P., Brinksma, E., Larsen, K.G.: Optimal infinite scheduling for multi-priced timed automata. Formal Methods in System Design 32(1), 3–23 (2008)

[10] Bouyer, P., Fahrenberg, U., Larsen, K.G., Markey, N., Srba, J.: Infinite runs in weighted timed automata with energy constraints. In: FORMATS. LNCS, vol. 5215, pp. 33–47. Springer (2008)

[11] Chakrabarti, A., de Alfaro, L., Henzinger, T.A., Stoelinga, M.: Resource interfaces. In: Alur, R., Lee, I. (eds.) EMSOFT. Lecture Notes in Computer Science, vol. 2855, pp. 117–133. Springer (2003)

[12] Chatterjee, K., Doyen, L.: Energy parity games. In: Abramsky, S., Gavoille, C., Kirchner, C., auf der Heide, F.M., Spirakis, P.G. (eds.) ICALP (2). Lecture Notes in Computer Science, vol. 6199, pp. 599–610. Springer (2010)

[13] Ehrenfeucht, A., Mycielski, J.: Positional strategies for mean payoff games. International Journal of Game Theory 8, 109–113 (1979), 10.1007/BF01768705

[14] Juhl, L., Larsen, K.G., Srba, J.: Introducing modal transition systems with weight intervals. Journal of Logic and Algebraic programming (2011), to appear

[15] Larsen, K.G., Thomsen, B.: A modal process logic. In: LICS. pp. 203–210. IEEE Computer Society (1988)

[16] Larsen, K.G., Xinxin, L.: Equation solving using modal transition systems. In: LICS. pp. 108–117. IEEE Computer Society (1990)

[17] Zwick, U., Paterson, M.: The complexity of mean payoff games on graphs. Theoretical Computer Science 158, 343–359 (1996)