Technical Report

Exact Rate-Monotonic Schedulability Analysis for Implicit-Deadline Systems with $U < 1$ has Polynomial Time-Complexity

Björn Andersson

Eduardo Tovar

TR-060501

Version: 1.0

Date: May 2006

Exact Rate-Monotonic Schedulability Analysis for Implicit-Deadline Systems with $U < 1$ has Polynomial Time-Complexity

Björn ANDERSSON and Eduardo TOVAR

IPP-HURRAY!

Polytechnic Institute of Porto (ISEP-IPP)

Rua Dr. António Bernardino de Almeida, 431

4200-072 Porto

Portugal

Tel.: +351.22.8340509, Fax: +351.22.8340509

E-mail: {bandersson, emt}@dei.isep.ipp.pt

http://www.hurray.isep.ipp.pt

Abstract

Consider a set of periodically arriving real-time tasks scheduled on a single processor using rate-monotonic. We address the problem of deciding if and only if these tasks meet their deadlines; this is called exact schedulability analysis. We propose an algorithm for exact schedulability analysis of static-priority scheduled task sets and analyse its time-complexity. We prove that the time-complexity of the proposed algorithm can be expressed as a function of the utilisation (U) of the task set and that it is polynomial for any fixed $U < 1$. This is a major contribution to the state-of-the-art since, to the best of our knowledge, no other previously proposed schedulability analysis of static-priority scheduled task sets could claim to be exact while presenting polynomial time-complexity.

Exact Rate-Monotonic Schedulability Analysis for Implicit-Deadline Systems with U < 1 has Polynomial Time-Complexity

Björn Andersson and Eduardo Tovar
IPP Hurray Research Group
Polytechnic Institute of Porto, Portugal
{bandersson,emt}@dei.isep.ipp.pt

Abstract

Consider a set of periodically arriving real-time tasks scheduled on a single processor using rate-monotonic. We address the problem of deciding if and only if these tasks meet their deadlines; this is called exact schedulability analysis. We propose an algorithm for exact schedulability analysis of static-priority scheduled task sets and analyse its time-complexity. We prove that the time-complexity of the proposed algorithm can be expressed as a function of the utilisation (U) of the task set and that it is polynomial for any fixed U < 1. This is a major contribution to the state-of-the-art since, to the best of our knowledge, no other previously proposed schedulability analysis of static-priority scheduled task sets could claim to be exact while presenting polynomial time-complexity.

1. Introduction

A fundamental problem in the design of real-time systems is to schedule a set of tasks to meet its deadlines. Accordingly, a plethora of scheduling algorithms has been developed. One important class of such algorithms uses *static-priority scheduling*. A static priority is assigned to each task, and whenever a task requests to execute it competes with the other tasks that have requested to execute on the processor. The task with the highest priority is selected for execution. The tasks' priorities are usually assigned upon some relevant timing characteristics of the task.

A common approach is to use the *implicit-deadline model*. In this model, a task is assumed to arrive periodically or sporadically [1] and the relative deadline is equal to its period or to its minimum inter-arrival time [1]. Under this task model, it is known that assigning priorities according to *rate-monotonic* (RM) [2, 3] is optimal, in the sense that if any priority assignment causes deadlines to be met then RM will do so as well. With RM, tasks are assigned priorities monotonically increasing with their arrival rate [2, 3].

Rate-monotonic is one of the earliest developed priority-assignment schemes, and it has since become quite popular [4, 5]. It is simple, it is supported by various operating systems and, importantly, quite a few schedulability analysis techniques are available for it.

Schedulability analysis techniques are crucial for the design of hard real-time computing systems. Their purpose is to predict whether all tasks will meet their deadlines at run-time, assuming that the characteristics of the tasks are known before run-time. Naturally, it is desirable that a schedulability analysis is fast, especially when it is used for online admission control. It is also important that it is fast when used for offline optimisation that repeatedly tests different task sets, such as selecting periods for control tasks to maximize overall control performance [6], or for selecting the processor speed to minimise energy-consumption [7]. The schedulability analysis should be safe, while at the same time not imposing overestimation of required resources.

A schedulability analysis is said to be *sufficient* if every task set that is guaranteed by the analysis to meet deadlines will do so during run-time. Conversely, a schedulability analysis is said to be *necessary* if every task set that is not guaranteed by the analysis to meet deadlines will miss at least one deadline during run-time. Therefore, a schedulability analysis is said to be *exact* if it is both sufficient and necessary. Obviously, a schedulability analysis which is sufficient but not exact is said to be pessimistic.

A number of sufficient schedulability analysis techniques [2, 3, 8-15] with polynomial time-complexity have been previously proposed for RM with the implicit-deadline assumption. Unfortunately, none of them happen to be necessary, thus not being exact as well. One of those approaches is called

approximate schedulability analysis [13-15], and has recently received a great deal of interest. Indeed, it offers polynomial time-complexity. Additionally, albeit not providing a necessary schedulability test, it considers a parameter ε "tunable" by the system's designer that governs the pessimism. If the approximate schedulability analysis does not give a guarantee that deadlines are met, then the designer does not know whether the task set will meet all deadlines or not. However, the approach provides the designer with the perception that if the speed of the processor is reduced by an amount ε then the task set would miss a deadline.

Therefore, and unfortunately, the approximate schedulability analysis only provides a relative guarantee (what would happen if a slower processor was used); it can happen that it does not state whether the tasks under consideration meet their deadlines or not.

On the other hand, several approaches providing exact schedulability analysis for rate-monotonic with the implicit-deadline assumption have been proposed in the literature (e.g., [16-20]). It happens however that none presents polynomial time-complexity. The time complexity of the processor-demand analysis [16] is clearly pseudo-polynomial [21]. The complexity of the response-time analysis [17] is unknown, but it is pseudo-polynomial [21], and actually is generally believed to be not polynomial.

Improvements to these schemes have been developed, but they do not change the basic fact: no exact schedulability analysis for RM with implicit-deadline is known with polynomial time-complexity. Actually, designing such an algorithm (or proving its non-existence) has been regarded as one of the major outstanding questions in real-time computing [22].

In this paper we study *exact* schedulability analysis for RM for the case where the task set utilisation is strictly less than 100%. We propose an algorithm for this, and analyze its computational complexity. We prove that its computational complexity is $O(n^4/(1-U))$, where n is the number of tasks and U is the utilisation of all tasks in the task set. Hence, for any fixed $U < 1$, its computational complexity is polynomial.

We believe that this result is significant since such an approach is superior to the approximate schedulability analysis (e.g., [13-15]) in that it gives *absolute* guarantees rather than *relative* guarantees.

The remainder of this paper is structured as follows. Section 2 states the system model and terminology. It also presents results from previous research upon which our approach builds. Section 3 provides the needed intuition to the new exact schedulability analysis. It presents formally the novel approach and then also formally proves its computational complexity. In Section 4 we discuss the ability of previously proposed approaches to achieve exact schedulability analysis, and a comparison with our new algorithm is reasoned out. We also discuss known results about the computational complexity of exact schedulability analysis [23]. Finally, in Section 5, conclusions are drawn.

2. Rate-Monotonic Scheduling

Consider a task set $\tau = \{\tau_1, \tau_2, \ldots, \tau_n\}$ that is scheduled on a single processor. Each task τ_i is characterized by T_i and C_i with the following interpretation. A task τ_i makes requests to execute and the time between two consecutive requests from τ_i is at least T_i. Every time a task requests to execute it needs to execute C_i time units no later than T_i time units after it has requested to execute.

Each task is assigned a priority according to RM; that is, if $T_j < T_i$ then τ_j is assigned a higher priority than τ_i. At every moment the task selected for execution on the processor is the highest-priority task among the tasks that have arrived and have remaining execution. We assume that tasks can be preempted and later resumed. Preemptions and resumptions are allowed at every time, and it is assumed that no overhead is associated to these actions. We also assume that tasks require no other resources than the processor. We address the following problem.

> *Design an algorithm which decides whether a task set meets deadlines. The algorithm takes a task set as input and produces an output: SUCCESS or FAILURE. If the algorithm declares SUCCESS for a task set then it must hold that all deadlines are met when the task set is scheduled using RM. If the algorithm declares FAILURE for a task set then it must hold that it is possible to assign arrival times subject to the constraint given by T_i such that there is at least one task that misses at least one of its deadlines when scheduled by RM.*

With no loss of generality, it is assumed that $T_1 \leq T_2 \leq \ldots \leq T_n$. For historical reasons, in the rest of this manuscript T_i will be referred to as the period of τ_i.

We will now discuss some previous relevant results that will serve as a starting point for the new proposed algorithm. It is known from previous research [2] that all tasks meet their deadlines if and only if they meet their

deadlines when all tasks synchronously request execution at a time instant known as the *critical instant* [2]. Given that, a sufficient schedulability test has been proposed. Let *ln* 2 denote the natural logarithm of 2, which is approximately 0.69.

A sufficient schedulability analysis can then be formulated as stated in Theorem 1, which follows.

Theorem 1 (taken from [2])**.** *Given the periodic tasks* $\tau_1, \tau_2, \ldots, \tau_n$*, if it is true that*

$$\sum_{j=1}^{n} \frac{C_j}{T_j} \leq \ln 2 \qquad (1)$$

then all deadlines of all tasks are met. □

An exact schedulability test was also proposed [16]. Let us follow the reasoning used in [16] by defining the cumulative demands as

$$W_i(t) = \sum_{j=1}^{i} \left(\left\lceil \frac{t}{T_j} \right\rceil \times C_j \right) \qquad (2)$$

and the scheduling points S_i as

$$S_i = \left\{ k \times T_j \mid j = 1,..,i;\ k = 1,..,\left\lfloor \frac{T_i}{T_j} \right\rfloor \right\} \qquad (3)$$

An exact analysis can then be phrased as follows.

Theorem 2 (taken from [16])**.** *Given the periodic tasks* $\tau_1, \tau_2, \ldots, \tau_n$*, then* τ_i *can be scheduled for all task phasings using the rate-monotonic algorithm if and only if*

$$L_i(t) = \min_{\{t \in S_i\}} \frac{W_i(t)}{t} \leq 1 \qquad (4)$$

and therefore the entire task set is schedulable for all task phasings using the rate-monotonic algorithm if and only if

$$L = \max_{\{1 \leq i \leq n\}} L_i \leq 1 \qquad (5)$$

□

The computational complexity of the test in (5) is $O(n^2 \times \max\{T_1, T_2, \ldots, T_n\})$ since all scheduling points in S_i must be evaluated. Hence, the algorithmic approach inherent to the Theorem 2 has pseudo-polynomial time-complexity [21], and not polynomial time-complexity.

3. The New Exact Schedulability Analysis

We will now reason and propose a new schedulability analysis. We will prove that the proposed test is exact and that it has polynomial time-complexity.

In order to provide some intuition on the proposed approach, consider the task set (n = 2) example described in Table 1.

Table 1. Task set example.

Task	C_i	T_i
τ_1	0.2	1
τ_2	79	100

Theorem 2 can be used to decide whether this task set meets all its deadlines. It is easily obtained (using (3) that $S_1 = \{1\}$. Therefore, it results (using (2) that $W_1(1) = 0.2$, and hence $L_1 = 0.2$.

The schedulability analysis for τ_2 requires more computations since S_2 has many more elements: $S_2 = \{1, 2, 3, 4, \ldots, 100\}$. Values for W_2 need to be computed for all those values in S_2. The results are plotted in the graph provided by Figure 1. It results that $L_2 = 0.99$, meaning that (by (5)) $L = \max\{L_1, L_2\} = 0.99$. Therefore, and since $L \leq 1$, all deadlines are met.

In this example, the largest t in S_2 corresponds to the smallest value for $W_2(t)/t$ (refer to Figure 1), but this does not hold for other task sets. The crucial point in this small example is that $W_2(t)/t$ needs to be evaluated for a large number of scheduling points, therefore causing the time-complexity of the scheme to be quite high. This happens in task set examples where one task has a period which is much larger than the other tasks' periods.

Observe that the utilisation of the task set in Table 1 is strictly less than 100%. Let us create a new task set by multiplying T_2 and C_2 with a large constant (say 1000). This does not change the utilisation of the task set. The larger the constant is, the larger the number of scheduling points that needs to be explored is and hence the larger the running time of the algorithm in Theorem 2 is. However, the larger the constant is, (2) becomes more and more similar to what (2) would have been if the ceiling functions in (2) were dropped out. Hence, with a sufficiently large constant, τ_2 will meet its deadlines. The schedulability analysis for this case is trivial. Consequently, in order for the schedulability analysis to have a large computational complexity, T_2/T_1 should be large (so that there are many scheduling points to explore) but

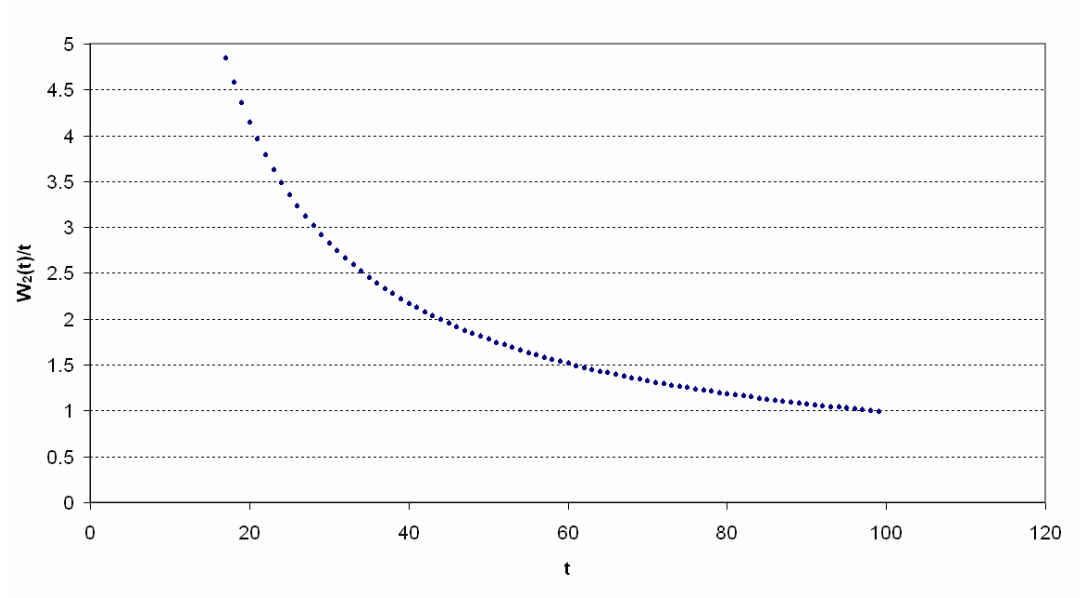

Figure 1. $W_2(t)/t$ as a function of t for the task set in Table 1.

not too large (because if it would be too large then the schedulability analysis would be trivial).

The higher the utilisation is, the larger T_2/T_1 can be while still keeping the schedulability analysis problem non-trivial. Hence in order for the schedulability analysis problem to have a high computational complexity it is necessary that the utilisation of the task set is high. For this reason, we will study exact schedulability analysis for the case of low utilisation (in Section 3.1) and for the case of high utilisation (in Section 3.2). These results will then be generalised in Section 3.3.

3.1. Case of low utilisation task sets

From the example described earlier, one could realise that if the period of the task under analysis is sufficiently large compared to the period of a higher priority task then all deadlines are met even if the utilisation was close to 100%. Based on this intuition we can prove a new schedulability test, which is formally phrased in Theorem 3 below.

Theorem 3. *Given a set of periodic tasks $\tau_1, \tau_2, \ldots, \tau_n$, if the following condition is verified*

$$\forall i \in \{1,\ldots,n\} : \sum_{j=1}^{i} \frac{C_j}{T_j} \le 1 - \sum_{j=1}^{i-1} \frac{C_j}{T_i} \qquad (6)$$

then all deadlines are met.

Proof: We will prove this theorem with induction on i.

Base case. The theorem is true for $i = 1$.

Proof: This is trivially true.

The induction step. Consider a number $k \ge 1$. We claim that if the theorem is true for $i = k$ then the theorem is true for $i = k + 1$ as well.

Proof. From the assumptions of the induction step it results that:

$$\sum_{j=1}^{k+1} \frac{C_j}{T_j} \le 1 - \sum_{j=1}^{k} \frac{C_j}{T_{k+1}}$$

which can be re-written as follows:

$$\frac{C_{k+1}}{T_{k+1}} + \sum_{j=1}^{k} \left(\frac{C_j}{T_j} + \frac{C_j}{T_{k+1}} \right) \le 1$$

Multiplying both sides of the previous inequality by T_{k+1} and simplifying yields that:

$$\frac{C_{k+1} + \sum_{j=1}^{k} \left(\frac{C_j}{T_j} \times T_{k+1} + C_j \right)}{T_{k+1}} \le 1$$

which can be re-arranged as follows:

$$\frac{C_{k+1} + \sum_{j=1}^{k} \left(\left(\frac{T_{k+1}}{T_j} + 1 \right) \times C_j \right)}{T_{k+1}} \le 1$$

Observing that $\lceil x \rceil < x + 1$ yields that:

$$\frac{C_{k+1} + \sum_{j=1}^{k} \left(\left\lceil \frac{T_{k+1}}{T_j} \right\rceil \times C_j \right)}{T_{k+1}} \leq 1$$

which is equivalent to:

$$\frac{\sum_{j=1}^{k+1} \left(\left\lceil \frac{T_{k+1}}{T_j} \right\rceil \times C_j \right)}{T_{k+1}} \leq 1$$

Taking into account (2) it results that:

$$\frac{W_{k+1}(T_{k+1})}{T_{k+1}} \leq 1$$

Thus, considering (3) and (4), it turns out that:

$$L_{k+1}(T_{k+1}) \leq 1$$

Hence τ_{k+1} meets its deadlines. It was already known that $\tau_1, \tau_2, \ldots, \tau_k$ all meet their deadlines. Consequently the induction step is true.

Combining the base case and the induction step proves the theorem. □

It can be seen that the exact schedulability test of task sets with low utilisation has also low computational complexity; it is only a matter of computing the utilisation as is done by Inequality 6.

3.2. Case of high utilisation task sets

The algorithm in Theorem 2 will now be applied to analyze task τ_i under the following assumption (high utilisation task set)

$$\sum_{j=1}^{i} \frac{C_j}{T_j} > 1 - \sum_{j=1}^{i-1} \frac{C_j}{T_i} \quad (7)$$

and a study on the complexity of the schedulability analysis will be performed. From (3) one can state that the number of scheduling points in S_i is smaller than or equal to $\sum_{j=1..i} \lfloor T_i / T_j \rfloor$.

With a simple re-arrangement, one can state that the number of scheduling points in S_i is thus smaller than or equal to:

$$\left(\sum_{j=1}^{i-1} \frac{T_i}{T_j} \right) + 1 \quad (8)$$

In the rest of this section, we will consider the case where:

$$\sum_{j=1}^{i} \frac{C_j}{T_j} < 1 \quad (9)$$

Let *UBU* (Upper Bound on the Utilisation) be used to denote a number such that:

$$\sum_{j=1}^{i} \frac{C_j}{T_j} \leq UBU \quad (10)$$

Clearly, *UBU* is a constant which satisfies the following condition:

$$UBU < 1 \quad (11)$$

If (9) is true then there is an *UBU* which satisfies both (10) and (11).

From Theorem 1, if the utilisation is smaller than $\ln 2$ then all deadlines are met. Hence, it can be assumed that:

$$\ln 2 \leq \sum_{j=1}^{i} \frac{C_j}{T_j} \quad (12)$$

Lemma 1 will now be introduced.

Lemma 1. *Given the periodic tasks $\tau_1, \tau_2, \ldots, \tau_i$, and that (12) holds, and that an UBU can be found such that (10) and (11) hold as well, then if the algorithm in Theorem 2 is used on task sets satisfying (7) it will hold that:*

$$|S_i| \leq \frac{(i-1)^2}{1 - UBU} + 1 \quad (13)$$

Proof: The lemma will be proved by considering the following optimisation problem: maximise the left hand side (LHS) of (13) subject to the constraint of (7) and subject to other obvious constraints that will be stated later on in this proof.

A sequence of optimisation problems will be presented such that the objective function of every one of them provides a bound on the objective function to all previous ones. It will be shown that the last optimisation problem in the sequence will have an objective function that does not exceed the right hand

side (RHS) of (13), and this implies that the lemma is true.

Let hp(i) denote the set of positive integers that are strictly less than i, that is, hp(i)={1,2,3,…,i-1}.

We will start by formulating the following optimisation problem (OP1).

OP1: Maximise $|S_i|$ subject to
Inequality (7)
$\forall j \in hp(i)$: $C_j \leq T_j$
$0 \leq T_1 \leq T_2 \leq \ldots \leq T_i$
Inequality (12)
Inequality (10)

An upper bound on the objective function of OP1 can be obtained by replacing its objective function by formulation (8). Additionally, the first constraint in OP1 can be relaxed by allowing equality. We can also apply $\forall j \in hp(i)$: $C_j \leq T_j$ on the RHS of the 1st constraint. After that one can increase T_i such that the relaxed version of 1st constraint becomes the equality

$$\sum_{j=1}^{i} \frac{C_j}{T_j} = 1 - \sum_{j=1}^{i-1} \frac{T_j}{T_i} \tag{14}$$

rather than an inequality. (This maintains feasibility and increases the objective function.) We can also relax the constraints on the order of periods.

Therefore, an upper bound on the objective function of OP1 can be obtained by solving the following optimisation problem (OP2).

OP2: Maximise $(1+\sum_{j=1}^{i-1} T_i/T_j)$ subject to
Equation (14)
$\forall j \in hp(i)$: $0 \leq T_j \leq T_i$
Inequality (12)
Inequality (10)

For convenience, let OP2 be re-written into a minimization problem (OP3).

OP3: Minimise $(-1-\sum_{j=1}^{i-1} T_i/T_j)$ subject to
Equation (14)
$\forall j \in hp(i)$: $0 \leq T_j \leq T_i$
Inequality (12)
Inequality (10)

Now, let us denote the quantity (T_i/T_j) as r_j and the quantity $\sum_{j=1..i} C_j/T_j$ as UI. In this way, OP3 can be re-written as formulated next (OP4).

OP4: Minimise $(-1-\sum_{j=1}^{i-1} r_j)$ subject to

$$-UI + 1 - \sum_{j=1}^{i-1} 1/r_j = 0$$

$\forall j \in hp(i)$: $1 - r_j \leq 0$
$ln\,2 - UI \leq 0$
$UI - UBU \leq 0$

In OP4, r_j and UI are variables, while UBU is a constant. Now, the solution for this optimisation problem will be addressed.

It is known that a global minimiser is also a local minimiser, and that a necessary condition for a local minimiser is given by the Karush-Kuhn-Tucker (KKT) condition [24]. Using this result on both the objective and four constraints of OP4 yields a necessary condition for a globally optimal solution (assuming that μ_j, λ_1, λ_2 and λ_3 are the multipliers for constraints 2nd, 1st, 3rd and 4th constraint in OP4, respectively).

From the KKT condition, the gradient of the objective function plus the sum of the gradients of the constraints weighted by their multiplier is zero. This expresses a vector, where each element is zero. Let us look at each element in the vector. There are i-1 elements related to r_j, each of those expressed as follows (j ranging from 1 to i-1):

$$\forall j \in hp(i): -1 - \mu_j + \lambda_1 \times \frac{1}{r_j^{\,2}} = 0 \tag{15}$$

For UI it holds that:

$$-\lambda_1 - \lambda_2 + \lambda_3 = 0 \tag{16}$$

From the KKT condition, in each inequality constraint the multiplier will be zero and/or the constraint will be zero. Therefore, for the 2nd constraints in OP4 it results that (for each r_j, with j ranging from 1 to i-1):

$$\forall j \in hp(i): \mu_j \times (1 - r_j) = 0 \tag{17}$$

For the 3rd constraint in OP4 it results that:

$$\lambda_2 \times (\ln 2 - UI) = 0 \tag{18}$$

Finally, for the 4th constraint in OP4, it results that:

$$\lambda_3 \times (UI - UBU) = 0 \tag{19}$$

For the remaining reasoning, Lemma 2 is useful.

Lemma 2. *Consider OP4. For all feasible points that are local minimisers it holds that:* $\forall j \in$ hp(i): $\mu_j = 0$.

Proof: This lemma can be proved by contradiction. If this lemma was false, then there must be a j such that $\mu_j \neq 0$. Since multipliers must be non-negative for minimisers, there must a j such that $\mu_j > 0$. It follows from $\mu_j > 0$ that there must be a j such that $r_j = 1$ in order to satisfy the constraints (17). With this j such that $r_j = 1$ it turns out that the LHS of the 1st constraint in OP4 is negative, and thus the constraint cannot be satisfied. Hence the conclusion is that $\forall j \in \text{hp}(i)$: $\mu_j = 0$. □

Therefore, applying $\forall j \in \text{hp}(i)$: $\mu_j = 0$ on (15) gives:

$$r_1^2 = r_2^2 = \ldots = r_{i-1}^2 \tag{20}$$

Combining the 2nd constraint in OP4 with (20) yields:

$$r_1 = r_2 = \ldots = r_{i-1} \tag{21}$$

Applying (21) to the 1st constraint in OP4 yields:

$$\left(1 - UI\right) - \frac{i-1}{r_j} = 0 \tag{22}$$

Combining (21) with (22) yields:

$$r_1 = r_2 = r_3 = \ldots = r_{i-1} = \frac{i-1}{1-UI} \tag{23}$$

Applying (23) on OP4 yields the following objective function:

$$-\frac{(i-1)^2}{1-UI} - 1 \tag{24}$$

Let obj_{OPx} denote the value of the objective function for the optimisation problem x for the optimal solution. Reviewing the reasoning up to now, it yields that OP2 is a relaxation of OP1. OP3 is just a re-formulation of OP2 into a minimization problem. OP4 is a re-formulation of OP3 with a replacement of variables.

Therefore, we can state the following:

$$\left|S_i\right| \leq obj_{OP1} \leq obj_{OP2} = -obj_{OP3} = -obj_{OP4} \tag{25}$$

Thus, combining (24) and (25), changing variables between OP3 and OP4 and using (10) yields:

$$\left|S_i\right| \leq \frac{(i-1)^2}{1-UBU} + 1$$

and this states Lemma 1. □

If (9) holds then we can rewrite Lemma 1 into Theorem 4.

Theorem 4. *Given periodic tasks $\tau_1, \tau_2, \ldots, \tau_i$ and that (9) holds and (12) holds, if the algorithm in Theorem 2 is used then on task sets with*

$$\sum_{j=1}^{i} \frac{C_j}{T_j} > 1 - \sum_{j=1}^{i-1} \frac{C_j}{T_i}$$

then it holds that

$$\left|S_i\right| \leq \frac{(i-1)^2}{1 - \sum_{j=1}^{i} \frac{C_j}{T_j}} + 1$$

Proof: Select *UBU* such as

$$\sum_{j=1}^{n} \frac{C_j}{T_j} = UBU$$

and then apply Lemma 1. This gives the theorem. □

3.3. The general case

The schedulability analysis of tasks with low utilisation has been analysed in Section 3.1. The number of scheduling points that needs to be explored was analyzed in Section 3.2. Based on these results, the new algorithm performing an exact schedulability test of RM scheduling on a single processor can be presented. Figure 2 outlines such an algorithm (Algorithm 1).

The properties of this algorithm will be proved next.

Property 1. *Algorithm 1 terminates: it declares success, failure or undecided.*

Proof: Follows from the observation that Algorithm 1 consists of a for-loop. □

Property 2. *If $\sum_{j=1..n} C_j/T_j < 1$ then Algorithm 1 declares success or failure.*

Proof: Follows directly from the algorithm description in Figure 2 and from the fact that it terminates. □

Property 3. *If Algorithm 1 declares success then all deadlines are met.*

```
1. if ∑j=1..n Cj/Tj >1 then declare FAILURE endif
2. if ∑j=1..n Cj/Tj =1 then declare UNDECIDED endif
3. if ∑j=1..n Cj/Tj <1 then
4.   for i=1 to n loop
5.     if (∑j=1..i Cj/Tj >ln2) and (∑j=1..i Cj/Tj >1-∑j=1..i-1 Cj/Ti) then
6.       run the exact schedulability analysis expressed in Theorem 2
7.       if Theorem 2 could not guarantee that deadlines are met then
8.         declare FAILURE
9.       endif
10.    endif
11.  endfor
12.  declare SUCCESS
13.endif
```

Figure 2. Exact schedulability analysis for RM on a single processor (Algorithm 1).

Proof: It follows from Theorems 1-3. □

Property 4. *If Algorithm 1 declares failure then at least one deadline will be missed.*

Proof: Follows from the observation that Theorem 2, used on line 6, is a necessary schedulability analysis. □

Lemma 3 (Property 5). *The execution of lines 5-10 of Algorithm 1 has a time complexity of* $O(i^3/(1-\sum_{j=1..i} C_j/T_j))$.

Proof: Follows from the observation that among lines 5-10, all lines of code requires at most O(i), except line 6. From Theorem 4 it results that line 6 explores at most $O(i^2/(1-\sum_{j=1..i} C_j/T_j))$ scheduling points. For each scheduling point (2) is applied, and (2) has a time complexity of $O(i)$. Hence the execution of lines 5-10 has a time complexity of $O(i^3/(1-\sum_{j=1..i} C_j/T_j))$. □

Theorem 5 (Property 6). *Algorithm 1 has a time complexity* $O(n^4/(1-\sum_{j=1..n} C_j/T_j))$

Proof: Since $i \le n$ it results that $i^3/(1-\sum_{j=1..i} C_j/T_j)) \le n^3/(1-\sum_{j=1..n} C_j/T_j)$. Hence, from Lemma 3 it follows that the execution of lines 5-10 of Algorithm 1 has a time-complexity of $O(n^3/(1-\sum_{j=1..n} C_j/T_j))$. Since lines 5-10 are executed n times, it results that Algorithm 1 has a time complexity $O(n^4/(1-\sum_{j=1..n} C_j/T_j))$. □

It can be seen from these Properties 1-4 and 6 that Algorithm 1 is an exact schedulability test and has polynomial time-complexity.

4. Discussion and Previous work

An algorithm that achieves exact schedulability analysis for RM for task sets with utilisation strictly less than 100% has been proposed in this paper. We will now review previous work and discuss their ability to achieve this.

The rate-monotonic scheduling algorithm for a single processor was introduced in the 1960:ies [25]. It was shown that if the utilisation is 69% or less, then all deadlines are met [2, 3]. In addition, there are task sets with marginally higher utilisation which misses a deadline when scheduled by RM. The result of the 69% bound has been found correct by several generations of researchers but the proof of it has been found incorrect [26] and it has been re-proved [27-29].

Later, exact schedulability analyses were presented. One of such analyses corresponds to Theorem 2 in this paper. The approach evaluates the work performed for every scheduling point. It clearly has pseudo-polynomial time-complexity, since the number of scheduling points grows as a function of the maximum period [16]. Another exact approach [17] is often called *response-time analysis* (RTA). It calculates the response time of each individual task. The response time of a task is the maximum time that elapses from when a task requests to execute until it finishes the execution of that request. The RTA starts with a lower bound on the response time and uses fixed-point iteration to finally converge so that the lower bound is the response-time. This corresponds to an exact schedulability analysis since the response time can be compared to the deadline. The number of iterations that are needed to perform RTA is unknown, but current research provide an upper bound on the number of iterations; which is pseudo-polynomial.

The test expressed by Theorem 2 [16] has been improved [18] through the enumeration of the scheduling points by starting with large numbers. This improves performance in simulation experiments. Using the same idea it was shown that the computational complexity can be bounded as $O(2^n)$ [19]. (The idea of exploring scheduling points in backward order was also used [29] to obtain an alternative proof of the 69% bound or RM.). The response-time calculations [17]

have been improved as well [20]. This was achieved by using tighter lower bounds on the response-time as an initial lower bound in the calculations, and this improved the performance in simulation experiments. However, none of these techniques have been proven to have a polynomial time-complexity.

Several schedulability analyses that are sufficient but not necessary have also been proposed in the literature. Some of these derive a utilisation bound as a function of the periods [8] or as a set of predefined possible periods [30]. With knowledge of the execution times, the utilisation of the task set can be computed and compared to the specialized utilisation bound; if the utilisation is less then all deadlines are met. Other techniques compute similar utilisation bounds which are not tight [9] but they can be computed faster. Yet another technique computes a hyperbolic function [10] (also on page 153 in [11]) which expresses a subset of the region of schedulable task sets. Originally this technique was proposed for use in partitioned multiprocessor scheduling [12].

One can easily perceive from Theorem 2 that if periods in a task set are harmonic then the ceilings can be dropped out in Theorem 2. As a result, all task sets with a utilisation of 100% or less will meet their deadlines. However, even if only some tasks are harmonic, it still improves performance [31, 32]. One technique [33] attempts to transform a task set to a harmonic task set where periods are shorter. If this transformation succeeds and if the utilisation does not exceed 100% then it results that the original task set meets all deadlines as well.

The schedulability of a task set can be assessed by a family of techniques called *approximate schedulability analysis* [13-15]. The one which is most relevant for comparison with our algorithm is the one which is *pessimistic*. Therefore it is a sufficient schedulability analysis. It is not necessary but if the schedulability analysis cannot guarantee that deadlines are met then it is known that if the speed of the processor is decreased by ε then the task set will miss a deadline. A small value of ε makes the analysis more similar to an exact analysis but it also increases the computational complexity. This technique was applied to analyze static-priority scheduling on a single processor [13]. This is similar to our technique in that both have a polynomial time-complexity. It differs however in that our technique gives absolute guarantees (the task set meets all deadlines or the task set misses at least one deadline), whereas the approximate schedulability analysis gives relative guarantees (the task set misses a deadline if it is executed on a slower processor). It is interesting to see that if the "1" in lines 1-3 in Algorithm 1 (Figure 2) is replaced with 1-ε then our scheme offers the same performance guarantee as the scheme in [13] for the special case where the relative deadline is equal to the period (the work in [13] studied the general case where the relative deadline does not have to be equal to the period).

The computational complexity of exact schedulability analysis is a well-studied problem (see [23] for an excellent coverage). It is known that the problem of deciding whether a periodic task set with known arrival times and arbitrary relative deadlines can be scheduled with EDF is co-NP-hard [34] (see page 8 in [23] for a more accessible proof). A similar technique was used to obtain the same complexity for static-priority scheduling [35] (see page 23 in [23] for a more accessible proof). However, these results assume that a task arrives periodically and has a pre-specified time when a task arrives for the first time (often called *offset*). For this reason, the computational complexity of our problem (static-priority scheduling with implicit deadline and all tasks can arrive at the same time) has (as pointed out on page 24 in [23]) pseudo-polynomial time-complexity or lower complexity. In this paper, we have proposed an algorithm that performs an exact schedulability analysis for every task set with $U < 1$, and the algorithm has provably a polynomial time-complexity.

5. Conclusions

We addressed the problem of exact schedulability analysis of RM on a single processor for the case where U is fixed such that $U<1$. We proposed an algorithm for this and we saw that its computational complexity is $O(n^4/(1-U))$, where n is the number of tasks and U is the utilisation of all tasks in the task set. An important implication is that for any fixed $U < 1$, the computational computational complexity of the analysis is polynomial. We consider for future work finding techniques that allow analyzing task sets with $U=1$. Checking if the period of the task under consideration is harmonic with its higher-priority tasks might be such a technique.

References

[1] A. Mok, "Fundamental Design Problems of Distributed Systems for the Hard Real-Time Environment," in *Electrical Engineering and Computer Science*. Cambridge, Mass.: Massachusetts Institute of Technology, 1983.

[2] C. L. Liu and J. W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment," *Journal of the ACM (JACM)*, vol. 20, pp. 46-61, 1973.

[3] O. Serlin, "Scheduling of Time Critical Processes," presented at Proceedings of the Spring Joint Computer Conference, Atlantic City, NJ, 1972.

[4] K. Tindell, A. Burns, and A. J. Wellings, "An Extendible Approach for Analyzing Fixed Priority Hard Real-Time Tasks," *Real-Time Systems*, vol. 6, pp. 133-151, 1994.

[5] L. Sha, R. Rajkumar, and S. S. Sathaye, "Generalized Rate Monotonic Scheduling Theory: A Framework for Developing Real-time Systems," *Proceedings of the IEEE*, vol. 82, pp. 68-82, 1994.

[6] D. Seto, J. P. Lehoczky, L. Sha, and K. G. Shin, "On task schedulability in real-time control systems," presented at 17th IEEE Real-Time Systems Symposium, Washington, D.C., USA, 1996.

[7] E. Bini, G. C. Buttazzo, and G. Lipari, "Speed Modulation in Energy-Aware Real-Time Systems," presented at 17th Euromicro Conference on Real-Time Systems, Palma de Mallorca,. Palma de Mallorca, Balearic Islands, Spain, 2005.

[8] D.-W. Park, S. Natarajan, A. Kanevsky, and K. Myung Jun, "A generalized utilization bound test for fixed-priority real-time scheduling," presented at Proceedings Second International Workshop on Real-Time Computing Systems and Applications, 1995.

[9] S. Lauzac, R. Melhem, and D. Mossé, "An improved rate-monotonic admission control and its applications," *IEEE Transaction on Computers*, vol. 52, pp. 337- 350, 2003.

[10] E. Bini, G. C. Buttazzo, and G. M. Buttazzo, "Rate monotonic analysis: the hyperbolic bound," *IEEE Transaction on Computers*, vol. 52, pp. 933-942, 2003.

[11] J. W. S. Liu, *Real-Time systems*. Upper Saddle River, NJ: Prentice Hall, 2000.

[12] O. Yingfeng and S. H. Son, "Allocating fixed-priority periodic tasks on multiprocessor systems," *Real-Time Systems*, vol. 9, pp. 207 - 239, 1995.

[13] N. Fisher and S. K. Baruah, "A Fully Polynomial-Time Approximation Scheme for Feasibility Analysis om Static-Priority Systems with Arbitrary Relative Deadlines," presented at Euromicro Conference on Real-time Systems, Palma de Mallorca, Balearic Islands, Spain, 2005.

[14] S. Chakraborty, S. Künzli, and L. Thiele, "Approximate Schedulability Analysis," presented at IEEE Real-Time Systems Symposium, Austin, TX, USA, 2002.

[15] K. Albers and F. Slomka, "An Event Stream Driven Approximation for the Analysis of Real-Time Systems," presented at Euromicro Conference on Real-time Systems, Catania, Sicily, Italy, 2004.

[16] J. Lehoczky, L. Sha, and Y. Ding, "The Rate monotonic scheduling algorithm: exact characterization and average case behavior," presented at Proceedings of the IEEE Real-Time System Symposium, 1989.

[17] M. Joseph and P. Pandya, "Finding Response Times in a Real-Time System," *The Computer Journal, British Computer Society*, vol. 29, pp. 390-395, 1986.

[18] E. Bini and G. C. Buttazzo, "Schedulability analysis of periodic fixed priority systems," *IEEE Transaction on Computers*, vol. 53, pp. 1462- 1473, 2004.

[19] Y. Manabe and S. Aoyagi, "A Feasibility Decision Algorithm for Rate Monotonic and Deadline Monotonic Scheduling," *Real-Time Systems*, vol. 14, pp. 171-181, 1998.

[20] M. Sjödin and H. Hansson, "Improved Response-Time Analysis Calculations," presented at Real-Time Systems Symposium, Madrid, Spain., 1998.

[21] M. R. Garey and D. S. Johnson, *Computers and Intractability A guide to the Theory of NP-Completeness* New York: W. H. Freeman and Company, 1979.

[22] S. Baruah and J. Anderson, "Where is Real-Time and Embedded Systems Research Going?," vol. 1. IEEE Distributed Systems Online, 2000.

[23] S. Baruah and J. Goossens, "Scheduling Real-time Tasks: Algorithms and Complexity," in *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*, J. Y.-T. Leung, Ed.: Chapman Hall/ CRC Press, 2004.

[24] S. G. Nash and A. Sofer, *Linear and Nonlinear optimization*: McGraw-Hill, 1996.

[25] C. L. Liu, "Scheduling algorithms for multiprocessors in a hard real-time environment," in *JPL Space Programs Summary 37-60*, vol. 2, 1969, pp. 28-31.

[26] R. Devillers and J. Goossens, "Liu and Layland's schedulability test revisited," *Information-Processing-Letters*, vol. 73, pp. 157-161, 2000.

[27] D. Shuzhen, X. Qiwen, and Z. Naijun, "A formal proof of the rate monotonic scheduler," presented at Real-Time Computing Systems and Applications, Hong Kong, China, 1999.

[28] D. Chen, "Real-time data management in the distributed environment," The University of Texas at Austin, 1999.

[29] N. Nissanke, *Real-time systems*. London ; New York: Prentice Hall, 1997.

[30] C.-G. Lee, L. Sha, and A. Peddi, "Enhanced utilization bounds for QoS management," *IEEE Transaction on Computers*, vol. 53, pp. 187- 200, 2004.

[31] T.-W. Kuo and A. K. Mok, "Load adjustment in adaptive real-time systems," presented at Real-Time Systems Symposium, San Antonio, TX, USA, 1991.

[32] C. Jiongxiong, "Extension to Fixed Priority with Preemption Threshold and Reservation-Based Scheduling." Ontario: University of Waterloo, 2005.

[33] C.-C. Han and H.-Y. Tyan, "A better polynomial-time schedulability test for real-time fixed-priority scheduling algorithms," presented at IEEE Real-Time Systems Symposium, San Francisco, CA, USA, 1997.

[34] J. Leung and M. Merrill, "A note on the preemptive scheduling of periodic, real-time tasks," *Information Processing Letters*, vol. 11, pp. 115-118, 1980.

[35] J. Leung and J. Whitehead, "On the Complexity of Fixed-priority Scheduling of Periodic Real-Time Tasks," *Performance Evaluation, Elsevier Science*, vol. 22, pp. 237-250, 1982.