On Preemptive Scheduling of Periodic, Real-Time Tasks on One Processor*

Sanjoy K. Baruah† Rodney R. Howell‡ Louis E. Rosier†

Abstract

We investigate the preemptive scheduling of periodic, real-time task systems on one processor. We present three major results. First, we show that the Simultaneous Congruences Problem is NP-complete in the strong sense. Although this result is included primarily as a lemma for showing our next major theorem, it is important in its own right, answering a question that has been open for ten years. Our second major result is perhaps the most important in the paper — that deciding whether a given task system is feasible on one processor is co-NP-complete in the strong sense. Our fourth major result is that for incomplete task systems, i.e., task systems in which the start times are not specified, the feasibility problem is Σ_2^P-complete. Several other results involving cases in which all tasks are initially released at the same time, or in which there are a fixed number of distinct types of tasks, can be derived from these three theorems.

1 Introduction

The study of scheduling theory has recently become a rapidly expanding area of research (see, e.g., [Bla87]). One specific area of scheduling theory which has been studied for a number of years involves preemptive scheduling of periodic, real-time task systems with deadlines [LL73, Lab74, LM80, LM81, LW82, Leu89]. A task system consists of a finite number of tasks, each of which is released at regular periodic intervals. In general, the initial release of individual tasks may occur at different times. Each time a task is released, it must be scheduled on a processor for some specified number of time units before its deadline is reached. The execution time requirement and the amount of time before the deadline is reached are both unchanged for each subsequent release of the task, but may be different for different tasks. We are interested in schedules in which a task may be preempted and resumed at a later time, or if we are considering a multiprocessor environment, resumed by a different processor. However, we require that no single task be simultaneously scheduled on more than one processor. There is no penalty associated with a preemption, but preemptions can only, occur at integer time values (we show in [BHR90] that this last restriction does not cause any loss of generality).

In this paper, we examine the computational complexities of various problems concerning the existence of a valid preemptive schedule on one processor. We examine two main types of task systems: *complete* task systems and *incomplete* task systems. In a complete task system, the start times, execution times, deadlines, and periods for each task are given as input; in an incomplete task system, the start times are omitted. A complete task system is said to be *feasible* on one processor if a valid schedule for one processor exists. An incomplete task system is said to be feasible on one processor if there exist start times such that the resulting complete task system is feasible. For each type of task system, we will examine the complexity of the *feasibility problem*; i.e., the problem of determining whether the given task system is feasible.

The feasibility problem for complete task systems on one processor was shown to be co-NP-hard by Leung and Merrill [LM80]. The proof consists of a reduction from the complement of the Simultaneous Congruences Problem (SCP), which had been shown to be NP-complete by Leung and Whitehead [LW82]. However, SCP was only shown to be NP-hard in the weak sense, and no pseudo-polynomial time algorithm was given. Whether this result could be sharpened has been presented as an open question [LM80, LW82]. For our first major result in Section 2, we answer this question by showing SCP to be NP-complete in the strong sense. From this result and the proof of Leung

*This work was supported in part by National Science Foundation Grant No. CCR-8711579.

†Dept. of Computer Sciences, The University of Texas at Austin, Austin, TX 78712, U.S.A.

‡Dept. of Computing and Information Sciences, Kansas State University, Manhattan, KS 66506, U.S.A.

and Merrill, it follows that the feasibility problem is co-NP-hard in the strong sense. We then show the feasibility problem on one processor to be co-NP-complete, thus improving upon the PSPACE upper bound given by Leung and Merrill [LM80]. Our co-NP upper bound is very important in that it leads to a number of additional results concerning synchronous systems (i.e., systems in which all start times are 0) and systems with a fixed number of distinct types of tasks.

In Section 3, we examine the feasibility problem for incomplete task systems. Our co-NP-completeness result for complete task systems implies that this problem is in Σ_2^P, the second level of the polynomial-time hierarchy (see [Sto77]). We then use a result from [BRTV89] to show the main result of this section — that the feasibility problem for incomplete task systems on one processor is Σ_2^P-complete. Finally, our co-NP-upper bound from Section 3 yields a pseudo-polynomial time algorithm for incomplete task systems with a fixed number of distinct types of tasks.

We conclude in Section 4 with a discussion of the possibility of further extensions. Due to space limitations, we have omitted some of the proofs in what follows; for more details, see [BHR90].

2 Complete Task Systems

In this section, we show that the feasibility problem for complete task systems on one processor is co-NP-complete in the strong sense. In order to show this fact, we first show the Simultaneous Congruences Problem (SCP) to be NP-complete in the strong sense. This result answers an open question posed by Leung and Merrill [LM80] and Leung and Whitehead [LW82]. Since Leung and Merrill [LM80] have given a reduction from SCP to the complement of the feasibility problem such that the values in the feasibility problem are bounded by a polynomial in the values in the given instance of SCP, it then follows that the feasibility problem is co-NP-hard in the strong sense. We then show the feasibility problem to be co-NP-complete. The best previous upper bound for this problem was PSPACE [LM80]. We construct a new set of conditions, verifiable in NP, that are necessary and sufficient for a complete task system to be infeasible on one processor. These conditions are very useful, leading to three additional results concerning synchronous systems and systems with a fixed number of distinct types of tasks.

We will now introduce the Simultaneous Congruences Problem. Let N denote the natural numbers, and let $N^+ = N - \{0\}$. Let $A = \{(a_1, b_1), ..., (a_n, b_n)\} \subseteq N \times N^+$ and $2 \leq k \leq n$ be given. The *Simultaneous Congruences Problem* is to determine whether there is a subset $A' \subseteq A$ of k pairs and a natural number x such that for every $(a_i, b_i) \in A'$, $x \equiv a_i (\text{mod } b_i)$. This problem was shown to be NP-complete by Leung and Whitehead [LW82], only in the weak sense. In what follows, we show the problem to be NP-complete in the strong sense. Our proof uses the Generalized Chinese Remainder Theorem (see, e.g., [Knu81]), which we now reproduce.

Lemma 2.1: (The Generalized Chinese Remainder Theorem.) Let $A = \{(a_1, b_1), ..., (a_k, b_k)\} \subseteq N \times N^+$. There is an x such that for all $(a_i, b_i) \in A$, $x \equiv a_i (\text{mod } b_i)$ iff for all $1 \leq i < j \leq k$, $a_i \equiv a_j (\text{mod } \gcd(b_i, b_j))$.

Thus, we can conclude that if each pair of distinct pairs (a_i, b_i) and (a_j, b_j) "collides" (i.e., there is an x such that $x \equiv a_i (\text{mod } b_i)$ and $x \equiv a_j (\text{mod } b_j)$), then there is a simultaneous collision of all pairs. This fact is very useful in the construction that follows.

The proof by Leung and Whitehead [LW82] that SCP is NP-hard (in the weak sense) consisted of a reduction from the CLIQUE problem [Kar72]. Given an undirected graph $G = (V, E)$ such that $V = \{v_1, ..., v_n\}$, they constructed a set of pairs $\{(a_1, b_1), ..., (a_n, b_n)\}$ such that $a_i \equiv a_j (\text{mod } \gcd(b_i, b_j))$ iff $(v_i, v_j) \in E$. Thus, there is a simultaneous collision of k items iff G has a clique of size k. However, since each a_i and b_i were the product of $O(n^2)$ distinct prime numbers, the values of a_i and b_i were not bounded by any polynomial in the size of the description of G. In order to overcome this problem, we give an entirely different reduction, from 3-SAT rather than CLIQUE.

Theorem 2.1: SCP is NP-complete in the strong sense.

Proof: Leung and Whitehead [LW82] have shown SCP to be in NP, so we need only show NP-hardness. The proof is via a polynomial-time reduction from 3-SAT [Coo71]. Let C be an instance of 3-SAT over variables $x_1, \ldots, x_n$; without loss of generality let $C = \bigwedge_{j=1}^{m} C_j$, $C_j = \bigvee_{k=1}^{3} c_{jk}$, where each c_{jk} is a distinct literal — i.e., either a variable or the negation of a variable. Also without loss of generality, we assume that no clause contains both a variable and