The Periodic Polytope and its applications to a Scheduling Problem – A Static Perspective

K. Subramani *　　　Ashok Agrawala †

Abstract

Parameter variability and the existence of complex constraints between tasks are assured features of real-time scheduling. *Periodicity* of task sets is an additional feature that needs to be accomodated. Traditional scheduling models ignore the complexities involved in real-time scheduling by making simplistic assumptions about task interactions. In this paper, we present a model that captures the issues that we deem central to real-time scheduling in periodic task sets and demonstrate the existence of efficient and easily implementable algorithms for addressing schedulability queries in this model. Our model is very general and applicable to diverse areas ranging from real-time process scheduling in operating systems and avionics to manufacturing and traffic control.

1 Introduction

In real-time scheduling, we are often confronted with `process parameter variability` [SA00b, SA00d, Sta88]. *Process execution time* is a prime example of a variable parameter. Secondly, the execution of real-time processes is constrained through relationships that exist between their start times (and execution times) [SA00a, Das85, BFR71, GH93]. Traditional models do not accomodate either feature completely; instead they make simplifying assumptions about execution time behaviour that do not hold at run-time. In scheduling litreature, variable execution times are modeled through *worst-case* times and inter-process constraints are restricted to those relationships that can be represented by precedence graphs [Cof76, Pin95]. These simplifying assumptions are not only unrealistic, but as shown in [SA00b] lead to to catastrophic consequences, when the *inter-process constraints* are sufficiently complex and the system is "hard" [LTCA89].

Real-time design is thus a two-step process:(a) Proposing a model that captures the intricacies involved in real-time scheduling, and (b) Designing efficient algorithms that answer queries of interest posed in the proposed model. In this paper, we propose a model that adequately captures the issues involved in the scheduling of periodic task sets in real-time systems. Our scheduling model finds application in diverse areas such as assembly production [Y.K80], real-time databases [BFW97], traffic control [ABNM93] and real-time operating systems [LTCA89]. To the best of our knowledge, our model and the accompanying analysis are the first of their kind in the litreature.

Our investigations are concerned with the following issues:

- Do periodic task sets with inter-period constraints have finite length *cyclic schedules* ? (to be defined in §2)
- Can we determine such schedules efficiently, if they exist ?

The rest of the paper is organized as follows: Section §2 describes the periodic scheduling model; we also pose the *periodic schedulability query* there i.e. what it means for a periodic task set to be schedulable, from a static perspective. In this paper, our focus is on static perspectives only. In the succeeding section, viz. §3, we motivate the necessity for this model and discuss related approaches to the `periodic scheduling problem.` Our analysis commences in §4, where we study a certain class of infinite linear programs, called *periodic linear programs.* We show that the feasibility of periodic linear programs with `inter-period` constraints can be determined in

*Department of Computer Science, University of Maryland, College Park, ksmani@cs.umd.edu

†Department of Computer Science, University of Maryland, College Park, agrawala@cs.umd.edu

polynomial time. The class of periodic linear programs captures the essential complexity of the periodic scheduling model and this fact is brought out in §5. The discussion in §5 is formalized into the *periodic scheduling algorithm* in §6, which answers the query posed in §2. §6 also analyzes the complexity of the scheduling algorithm and discusses a special case of interest. We conclude in §7 with a summary of our results and some open problems for future research.

2 The Periodic Scheduling Model (Static Perspective)

Assume that the time axis is divided into intervals of length L starting at time $t = 0$ and extending infinitely. These intervals are called *scheduling windows* i.e. the first scheduling window is $[0, L]$, the second scheduling window is $[L, 2L]$ and in general the i^{th} scheduling window is $[(i-1).L, i.L]$. We are given a set of ordered non-preemptive jobs $\{J_1, J_2, \ldots J_n\}$, which have instances in each scheduling window. The set $\Gamma^1 = \{J_1^1, J_2^1, \ldots, J_n^1\}$ corresponds to the instance of the job set in the first scheduling window; the sets $\Gamma^i, i = 2, \ldots, \infty$ are defined similarly. The instances of a job within a scheduling window are called tasks. Associated with each job J_i is an execution time e_i, which is *not constant*, but can vary in a restricted manner, as discussed below. All task instances of a job have the same execution time i.e. e_i is the execution time for $J_i^j, j = 1, 2, \ldots$.

In scheduling window $[(i-1).L, i.L]$ linear constraints exist between the start times $\{s_1^i, s_2^i, \ldots, s_n^i\}$ of the tasks and their execution times $\{e_1, e_2, \ldots, e_n\}$. The constraint system is expressed in matrix form as :

$$A.[\vec{\mathbf{s^i}}, \vec{\mathbf{e}}] \leq \vec{\mathbf{b}}, \tag{1}$$

where,

- $\vec{\mathbf{s^i}} = [s_1^i, s_2^i, \ldots, s_n^i]$ is an $n-$vector of the start times of the tasks in the i^{th} scheduling window $[(i-1).L, i.L]$,
- $\vec{\mathbf{e}} = [e_1, e_2, \ldots, e_n] \in \mathbf{E}$ is an $n-$vector of the execution times of the tasks and $\mathbf{E}$ is a convex set capturing their interdependence.
- A is a $m \times 2.n$ matrix of rational numbers, m denoting the number of constraints,
- $\vec{\mathbf{b}} = [b_1 + u_1.(i-1).L, b_2 + u_2.(i-1).L, \ldots, b_m + u_m.(i-1).L]$ is an $m-$vector. The b_i are rational numbers, while the u_i are $\{0, 1\}$ integer constants, which are part of the input.

The vector elements $b_k + u_k.(i-1).L$ model the fact that in some constraints the constants will have to shift with each period, while in other constraints, the constants remain unaltered. Consider a constraint that requires the start time of the first task to be no less than 5 units from the start of the period. Clearly $s_1^1 \geq 5$, will work only in the first period. In the second period, the correct constraint is: $s_1^2 \geq 5 + L$ and in the i^{th} scheduling window, the constraint is:$s_1^i \geq 5 + (i-1).L$. On the other hand, some constraints are strictly relative i.e. they capture relationships between relative positioning of jobs. Consider the requirement that the second task should start at least 3 units after the first task commences. This is expressed as: $s_2^i \geq s_1^i + 3$. Observe that this constraint should not be modified in any way as we go from one scheduling window to the next. This is the rationale behind the u_i constants; they are chosen as 0 or 1 for each constraint depending on whether the constraint is strictly relative or not. Note that the u_i are fixed and part of the input.

There is a second constraint system which captures relationships between tasks in successive periods. This system is expressed as:

$$C.[\vec{\mathbf{s^i}}, \vec{\mathbf{s^{i+1}}}, \vec{\mathbf{e}}] \leq \vec{\mathbf{d}}, \tag{2}$$

where,

- $\vec{\mathbf{s^{i+1}}}$ is the start time vector of tasks in the $(i+1)^{th}$ period.
- C is a $m' \times 3.n$ matrix of rational numbers, m' denoting the number of constraints,
- $\vec{\mathbf{d}} = [d_1 + v_1.(i-1).L, d_2 + v_2.(i-1).L, \ldots, d_{m'} + v_{m'}.(i-1).L]$ is an $m'-$ vector; the d_i are rational numbers while the v_i are $[0, 1]$ constants similar to the u_i constants in System (1) and having the same rationale.

System (1) is a convex polyhedron in the $2.n$ dimensional space, spanned by the start time axes $\{\vec{\mathbf{s_1^i}}, \vec{\mathbf{s_2^i}}, \ldots, \vec{\mathbf{s_n^i}}\}$ and the execution time axes $\{\vec{\mathbf{e_1}}, \vec{\mathbf{e_2}}, \ldots, \vec{\mathbf{e_n}}\}$. It represents the constraints on the start-times of tasks within a scheduling window i.e. it captures the set of `intra-period` constraints. Likewise, sytem (2) is a convex polyhedron in the $3.n$ dimensional space, spanned by the start time axes $\{\vec{\mathbf{s_1^i}}, \vec{\mathbf{s_2^i}}, \ldots, \vec{\mathbf{s_n^i}}\}$, $\{\vec{\mathbf{s_1^{i+1}}}, \vec{\mathbf{s_2^{i+1}}}, \ldots, \vec{\mathbf{s_n^{i+1}}}\}$, and the execution time axes $\{\vec{\mathbf{e_1}}, \vec{\mathbf{e_2}}, \ldots, \vec{\mathbf{e_n}}\}$. It represents the set of `inter-period` constraints i.e. the constraints between the start times of tasks in adjacent scheduling windows.

The execution times e_i, are independent of the start times of the tasks; however they may have complex interdependencies among themselves. This interdependence is captured through the set $\mathbf{E}$. We regard the execution times as $n-$vectors belonging to the set $\mathbf{E}$. Observe that $e_i \in E_i$, where, where E_i is the projection of $\mathbf{E}$ on execution axis $\vec{\mathbf{e_i}}$.

Note that System $[(1)-(2)]$ is an infinite constraint system expressed in finite form.

Definition 2.1 *Order k constraint system: The constraint system obtained by expanding the Systems (1) and (2) over k scheduling windows is called an Order k constraint system.*

Accordingly, $A.\vec{\mathbf{s^1}} \leq \vec{\mathbf{b}}$ is the Order 1 constraint system, the Order 2 constraint system consists of: $A.\vec{\mathbf{s^1}} \leq \vec{\mathbf{b}}$, $A.\vec{\mathbf{s^2}} \leq \vec{\mathbf{b}}$, $C.[\vec{\mathbf{s^1}}, \vec{\mathbf{s^2}}] \leq \vec{\mathbf{d}}$, and so on.

Definition 2.2 *Order k Solution: An assignment $f : \vec{\mathbf{s^i}} \rightarrow R^n, i = 1, 2, \ldots, k$ is said to be a valid Order k solution, if it satisfies the Order k constraint System in Definition (2.1) for all vectors $\vec{\mathbf{e}} \in \mathbf{E}$.*

Definition 2.3 *Infinite Solution: An Infinite Solution is an Order ∞ solution*

Definition 2.4 *Cyclic Solution: An Infinite Solution is said to be cyclic with period p if $\vec{\mathbf{x^{p+i}}} = \vec{\mathbf{x^i}} + p.L$, $\forall i = a, a+1, \ldots, \infty$, where a denotes the number of initial transient windows prior to the onset of cycling.*

The following questions arise:

(a) Do *cyclic solutions* exist for the system represented by [(1)- (2)] ?

(b) What is the period of the cyclic solution ?

(c) Can cyclic solutions be determined efficiently ?

The rest of this paper is devoted to addressing the above questions and we shall show that they can be answered in the affirmative.

Let P^k denote the predicate which holds if there is an Order k solution. The Periodic Scheduling problem is concerned with checking whether P^∞ holds. We are now in a position to state the static periodic scheduling query:

$$\exists \vec{\mathbf{s^i}} = [s_1^i, s_2^i, \ldots, s_n^i] \forall \vec{\mathbf{e}} = [e_1, e_2 \ldots e_n] \in \mathbf{E}\ A.[\vec{\mathbf{s^i}}, \vec{\mathbf{e}}] \leq \vec{\mathbf{b}}, C.[\vec{\mathbf{s^i}}, \vec{\mathbf{s^{i+1}}}, \vec{\mathbf{e}}] \leq \vec{\mathbf{d}}, \quad \forall i = 1, 2, \ldots \infty \ ? \tag{3}$$

3 Motivation and Related Work

Periodic task systems form the bulk of real-time systems studied and analyzed in the litreature [DL78, LL73, CC89, Cho00, Foh95]. [LL73] was a seminal paper in the area of preemptive scheduling of periodic tasks, in which schedulability conditions were derived in terms of the Earliest Deadline First (EDF) heuristic. They showed that EDF scheduling was optimal, in that if there exists a preemptive schedule, then there exists a preemptive schedule using the EDF heuristic. Most manufacturing control systems are inherently periodic in that the same set of tasks repeats at regular intervals [Y.K80, Kor83]. [Ram90, Foh95] consider preemptive periodic task sets while [Cho97] considers non-preemptive task sets. In [Cho00] explicit mention is made of the contrast between time-based models such as the ones proposed in [Sak94, SA00d, SA00a] and priority- based models, such as the ones proposed in [HKL91, LL73, CC89]. The model differences occur primarily as a result of the real-time task system under consideration. Thus, `periodicity` is a common feature among real-time applications.

Real-time systems have non-trivial, complex constraints that constrain their execution [Ram90, Das85, Sak94]. Constraints are used to express relationships between tasks that need to be satsified at run-time e.g. the scheduling requirement that task T_1 should precede task T_2 is captured through: $s_1 + e_1 \leq s_2$, where all entities are as defined in §2. In periodic task sets, it becomes necessary to control *jitter* i.e. the variation in the arrival time of tasks across periods. Consider the requirement that the jitter between the fourth task in the current period and the first task in the succeeding period be no greater than 5. This is expressed as: $s_1^{i+1} - s_4^i \leq 5$. In general, far more complex relationships constrain the execution of jobs [SA00b]. Even a relatively unsophisticated traffic control system can be modeled as a set of periodic processes with inter-period and intra-period constraints [ABNM93]. A discussion of the wide range of applicability of constraint models in real-time scheduling can be found in [Das85, SA00d] and [ABNM93].

An important feature of real-time systems is the lack of preciseness in measurable parameters, such as the *execution time* of a process. It is vital that schedulabilty of task systems be guaranteed under variations of execution time. Suppose we know that the execution time of task T_1 i.e. e_1 can take on any value in the range $[4, 7]$. A schedule that is appropriate for $e_1 = 4$ may not satisfy the constraint set if during actual execution e_1 takes on the value 7. Attempts to mitigate the effects of variable execution time through *worst-case* assumptions, result in systems that fail at run-time [SA00d]. Thus, we need a model that explicitly accounts for parameter variability. [Sak94, GPS95] proposed a model in which the constraints are "standard" [SA00a, Cho97] and the execution time of a task is range-bound i.e. e_1 can take any value in the range $[l_1, u_1]$ at run-time, l_1 denoting the lower bound and u_1 denoting the upper bound respectively. This model can be represented by restricting the convex set $\mathbf{E}$ in §2 to an axis-parallel hyper-rectangle [SA00b, SA00d] and the matrices A and C to be network unimodular [Sch87]. However their model fails to account for task execution time interactions and dependencies. Consider for instance, the requirement in [Y.K80], that the cutting axes across a work-piece must be constrained in their combined velocities. Such a requirement is captured by setting $e_1 + e_2 \leq a$. Further, they considered aperiodic task sets only. [Cho00] considers periodic non-preemptive task models similar to the one that we have proposed in §2; in their model too the execution time vectors are restricted to axis-parallel hyper-rectangles and the constraint matrices are network unimodular. They provide an extremely complicated algorithm to address the *parametric* schedulabilty query. In this paper, we shall prove properties of periodic linear systems and use those properties in developing a simple, efficient algorithm for answering (3). In a forthcoming paper [SA00c], the parametric version of the schedulability query will be addressed.

Static scheduling in aperiodic processes has been the focus of [SA00d], where convex minimization techniques are used to guarantee run-time schedulabilty of tasks. We shall be using similar methods in §5.1 to ensure that the solutions we provide do not violate system constraints at run-time.

4 Periodic Linear programs (PLPs)

In this section, we shall study a special class of infinite linear programs called `periodic linear programs`. The goal is to formalize the development of solutions to linear systems such as [(1)-(2)] in §2. We shall show that periodic linear programs have a fixed-point solution.

Assume that the time-axis is divided into intervals of length L starting at time $t = 0$ and extending infinitely. Each interval $[(i-1).L, iL]$ represents a scheduling window. Let

$$A.\vec{\mathbf{x^i}} \leq \vec{\mathbf{b}}, \vec{\mathbf{x}} \geq \vec{\mathbf{0}} \tag{4}$$

where $\vec{\mathbf{x^i}} = [x_1^i, x_2^i, \ldots, x_n^i]$, (and the matrix A and vector $\vec{\mathbf{b}}$ are as defined in §2), denote a linear system that models a system in which the control variables i.e the x_j^i represent points in time. We desire the values of the control variables in all intervals. The constraints imposed by the linear system (4) exist in all scheduling windows. Further there exist additional constraints between the variables in one window and the variables in the immediately succeeding window. These constraints are called *inter-period* constraints as opposed to the intra-period constraints represented by (4). We focus on the case when inter-period constraints exist *only between adjacent scheduling windows.* We term this infinite linear program a *Periodic Linear Program* (PLP), since we are dealing with an infinite linear program with a regular structure both within and between periods. Let the inter-period constraints be represented by:

$$C.[\vec{\mathbf{x^i}}, \mathbf{x^{i+1}}] \leq \vec{\mathbf{d}} \qquad (5)$$

where C and $\vec{\mathbf{d}}$ are defined as in §2.

4.1 Collapsing Polytopes and a fixed-point theorem

We first study the nature of solutions to a periodic linear program represented by [(4)-(5)]. We show that if a PLP is feasible then it must have a fixed-point solution.

Observe that the periodic linear system is in fact a progression of systems that increasingly constrained. One way to look at this progression is to organize the PLP as a series of *rounds* as discussed below:

- In round r_1, the PLP consists of the system $A.[\vec{\mathbf{x^1}}] \leq \vec{\mathbf{b}}$;
- In round r_2, the PLP consists of the systems $A.[\vec{\mathbf{x^1}}] \leq \vec{\mathbf{b}}$,$A.[\vec{\mathbf{x^2}}] \leq \vec{\mathbf{b}}$, $C.[\vec{\mathbf{x^1}}, \vec{\mathbf{x^2}}] \leq \vec{\mathbf{d}}$;
- In round r_p, the PLP is the set of systems: $A.[\vec{\mathbf{x^i}}] \leq \vec{\mathbf{b}}, \ \forall i = 1, \ldots, p, \quad C.[\vec{\mathbf{x^i}}, \vec{\mathbf{x^{i+1}}}] \leq \vec{\mathbf{d}}, \ \forall i = 1, \ldots p-1$;

We have thus characterized the PLP as an infinite sequence of linear programs. We now have to prove that the solution set of this sequence converges if the PLP is feasible. For the rest of the discussion, we assume that the input PLP is feasible and derive a fixed-point solution. If no fixed-point solution exists, then we know that the input PLP is infeasible.

Figure (1) captures the main ideas in our analysis.

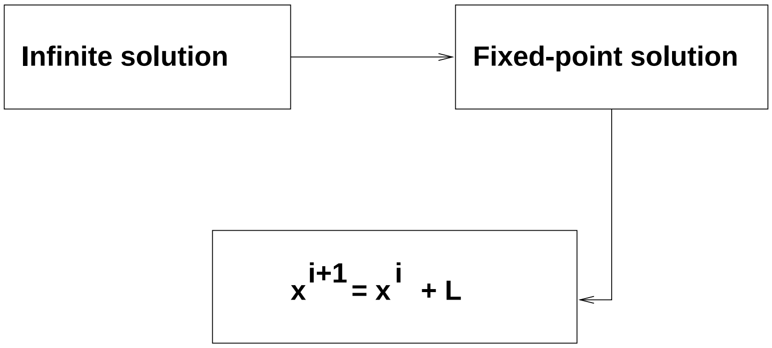

Figure 1: Existence of fixed-point solutions

Before we proceed with our analysis, we need to introduce the concept of *projection in polyhedral spaces.* Given a polyhedral system in Euclidean space R^d, it is a known fact that we can project this system onto a lower dimensional space $R^{d'}, d > d'$, while preserving the set of solutions to the original system [Sch87]. One of the more common techniques for achieving this projection is the Fourier-Motzkin elimination method [DE73, HN94], which is based on variable elimination. We associate the operator ∇ with the projection operation i.e. ∇ will henceforth be used as the projection operator. We need two properties of ∇ in our discussion.

Claim 4.1 ∇ *is a finite operator i.e.given a finite dimensional polyhedral space, its projection onto a lower dimensional space can be computed in finite time.*

Proof 4.1 *Refer [Sch87]. In fact, [DE73] gives an algorithm to construct the lower dimensional space.*

Claim 4.2 ∇ *is continuous i.e. points close to each other in the orginal space, will stay close to each other in the projected space.*

Proof 4.2 *Refer [Ist81b].*

It is clear that each round r_i represents an Order i constraint system (Definition (2.1)). Let us add the projection operator to each round. For instance, at the end of round r_2, we project the $2.n-$dimensional space spanned by $[\vec{\mathbf{x^1}}, \vec{\mathbf{x^2}}]$ onto the $n-$dimensional space spanned by $\vec{\mathbf{x^1}}$. Thus at the end of each round, we are always in the $n-$dimensional space spanned by $\vec{\mathbf{x^1}}$. As we proceed from round r_i to round r_{i+1}, we first move to a space with dimension n greater than the dimension of the original space and then return to the original space.

Essentially, two additional constraint matrices are introduced in each round. Constraint matrices are also linear operators which are continous and finite [Str80]. Λ denotes the constraint matrix operator .

Let S_1 denote the feasible polytope at the end of round r_1. In round r_2, we solve the linear system $[(4)-(5)]$ in the variables $[\vec{\mathbf{x^1}}, \vec{\mathbf{x^2}}]$ to get a new feasible set S_1'. This space is then projected onto $\vec{\mathbf{x^1}}-$space to get S_2. Each successive round (say r_i) consists of the following two steps:

1. Two constraint matrices operate on S_{i-1} to get a new set of feasible points (say S_{i-1}') in the $2n-$dimensional space spanned by $[\vec{\mathbf{x^1}}, \vec{\mathbf{x^i}}]$;
2. The projection operator ∇ projects the set S_{i-1}' onto $n-$dimensional space spanned by $\vec{\mathbf{x^1}}$.

Lemma 4.1 $S_1 \supseteq S_2 \supseteq S_3 \ldots$

Proof 4.3 *Note that each S_i, $i = 1, 2, \ldots, \infty$ polytope is formed by adding constraints to the polyhedral set S_{i-1}, further restricting the feasible space. The claim follows.*

Lemma 4.2 *If $S_k = S_{k+1}$ for some k, then $S_j = S_k, \forall j \geq k$.*

Proof 4.4 *S_{k+1} is obtained from S_k by the application of two operators; the constraint matrix operator (Λ) and the projection operator (∇). If $S_{k+1} = S_k$, it means that the operator application has returned the same set. Consequently repeatedly applying the same set of operators is not going to result in a new set i.e. $S_j = S_k \; \forall j \geq k$.*

Let Δ denote the composition of the matrix operator and the projection operator i.e. $\Delta = \Lambda \circ \nabla$. Thus, we can regard S_i as the set that results by the application of Δ to the set S_{i-1} i.e. $S_i = \Delta(S_{i-1})$.

Lemma 4.3 *Δ is a finite, continuous operator*

Proof 4.5 *From elementary calculus, we know that the composition of two finite, continuous functions is finite and continuous.*

We are now ready to state Brouwer's fixed-point theorem.

Theorem 4.1 *Let K be a compact (i.e. bounded and closed) convex non-empty subset of R^n and suppose that $f : K \rightarrow K$ is continuous. Then f has a fixed point.*

Proof 4.6 *Refer [Ist81a].*

In our case, S_1 is a polyhedral set and consequently compact. The boundedness of S_1 follows from the fact that all tasks have to be completed by the period L. Likewise, Δ is a continuous function. Hence we can apply Brouwer's fixed-point theorem to conclude that

Corollary 4.1 *If a periodic linear program is feasible i.e. it has an infinite solution, then it must have a fixed-point solution.*

Proof 4.7 *Follows from the above discussion.*

Corollary 4.2 *If a PLP has an infinite solution, then the solution is cyclic with period* 1.

Proof 4.8 *Follows from Corollary (4.1).*

The principal consequence of Corollary (4.2) is that we can set

$$\vec{\mathbf{x^{i+1}}} = \vec{\mathbf{x^{i}}} + L \tag{6}$$

in System [(4)−(5)] and solve the resultant system. Thus the inter-period constraints between window $[(i-1).L, i.L]$ and $[i.L, (i+1).L]$ can be merged using (6) and all constraints can be expressed in terms of the $\vec{\mathbf{x^i}}$ only. Let this linear system be denoted by

$$E.\vec{\mathbf{x^{i}}} = \vec{\mathbf{f}} \tag{7}$$

The polyhedral system (7) is called the `Periodic Polytope` corresponding to the PLP [(4-(5)].

Claim 4.3 *The PLP [(4-(5)] has a solution iff the polyhedron represented by (7) i.e. the periodic polytope is non-empty.*

Proof 4.9 *Let the periodic polytope (7) be non-empty. Thus we have a fixed-point solution of the system. From the discussion above, we know that this solution is valid for all scheduling windows* $[(i-1).L, i.L]$, $\forall i = 1, 2, \ldots, \infty$.

Let the periodic polytope (7) be empty. It follows that there is no fixed-point solution for the PLP. From Theorem (4.1) it follows that the PLP is infeasible.

The claim follows.

Observation 4.1 *The composite system (7) has n variables and $m + m'$ constraints*

Algorithm (4.1) summarizes our strategy solving PLPs.

Function PLP-Solve $(A, \vec{\mathbf{b}}, C, \vec{\mathbf{d}})$

Form the composite system $E.\vec{\mathbf{x}} \leq \vec{f}$, by using equation (6) .
if $(\{\vec{\mathbf{x}} : E.\vec{\mathbf{x}} \leq \vec{\mathbf{f}}\} \neq \phi)$ **then**
  `System has infinite Schedule` $\vec{\mathbf{x}}$
  return
else
  `System does not have an infinite schedule`
end if

Algorithm 4.1: PLP-Solver

Based on Observation (4.1), we note that the time taken by Algorithm (4.1) is $O(L(m+.m', n))$, where $L(m, n)$ is the fastest linear programming algorithm [Vai87], with m constraints and n variables.

In passing, we note that although we confined our model to inter-period constraints between adjacent scheduling windows only, our derivations did not require this restriction. In fact, the same technique works if we allow inter-period constraints between variables in a given scheduling window and variables of a fixed number of other scheduling windows. The only feature that is important is the regularity of the constraint set across scheduling windows !

5 Transformation of Scheduling Model to a PLP

Let us rewrite the constraint system [(1)-(2)] by moving all the execution time (i.e. the e_i) variables to the Right Hand Side (RHS). Accordingly, the periodic constraint system becomes:

$$\begin{aligned} G.\vec{\mathbf{s^{i}}} &\leq \vec{\mathbf{b}} - H.\vec{\mathbf{e}}, \\ I.[\vec{\mathbf{s^{i}}}, \vec{\mathbf{s^{i+1}}}] &\leq \vec{\mathbf{d}} - J.\vec{\mathbf{e}}, \forall i = 1, 2, \ldots, \infty \end{aligned} \tag{8}$$

where,

- G is an $m \times n$ rational matrix,
- H is an $m \times n$ rational matrix,
- I is an $m' \times 2.n$ rational matrix,
- J is an $m' \times n$ rational matrix
- $A.[\vec{\mathbf{s^i}}, \vec{\mathbf{e}}] = G.\vec{\mathbf{s^i}} + H.\vec{\mathbf{e}}$,
- $C.[\vec{\mathbf{s^i}}, \vec{\mathbf{s^{i+1}}}, \vec{\mathbf{e}}] = I.[\vec{\mathbf{s^i}}, \mathbf{s^{i+1}}] + J.\vec{\mathbf{e}}$

Observe that each constraint in (8) has an affine function of the execution time variables e_i on the RHS. In [SA00d], we showed that we could replace this affine function by using convex minimization. We apply the same technique here to reduce the RHS of every constraint in (8) to a number. The reduction results in the transformation of the original scheduling model into a PLP and we can apply the results obtained in §4 to obtain a valid solution.

Function PERIODIC-MODEL- TO-PLP $(A, \vec{\mathbf{b}}, C, \vec{\mathbf{d}})$
Split up the input system as discussed above to get System (8).
for $(l_1 \leftarrow 1$ **to** $m)$ **do**
  Let $r_i = \min_E(\vec{\mathbf{b}} - H.\vec{\mathbf{e}})_i$
end for
for $(l_2 \leftarrow 1$ **to** $m')$ **do**
  Let $q_i = \min_E(\vec{\mathbf{d}} - J.\vec{\mathbf{e}})_i$
end for

Algorithm 5.1: Transformation Procedure

The transfomation procedure consists of $(m + m')$ convex minimization calls and hence the total time taken is $O((m + m')C)$, where C is the fastest convex minimization algorithm [HuL93].

Algorithm (5.1) converts the system [(1)-(2)] into the following system:

$$\begin{aligned} G.[\vec{\mathbf{s^i}}] &\leq \vec{\mathbf{r}}, \\ I.[\vec{\mathbf{s^i}}, \vec{\mathbf{s^{i+1}}}] &\leq \vec{\mathbf{q}} \end{aligned} \tag{9}$$

Lemma 5.1 *Any solution that satisfies system (9) satisfies system [(1)-(2)].*

Proof 5.1 *Follows from the construction of System (9). Let us say that the solution to system (9) violates a constraint (say $\vec{\mathbf{a_p}}.[\vec{\mathbf{s_i}}, \vec{\mathbf{e}}] \leq b_p$) at execution time $\vec{\mathbf{e'}} = [e'_1, e'_2, \ldots, e'_n]$. We thus have $\vec{\mathbf{a_p}}.[\vec{\mathbf{s_i}}, \vec{\mathbf{e'}}] > b_p$. This implies that $\vec{\mathbf{g_p}}.\vec{\mathbf{s_i}} > r_p$, explicitly violating (9). This provides the necessary contradiction.*

Observation 5.1 *System (9) is a Periodic Linear Program and we can now use the techniques discussed in §4 to solve it.*

Observation 5.2 *We remark at this point that the non-existence of a static solution does not rule out the existence of alternate forms of solutions [SA00b, Sak94, SA00c].*

5.1 Axis-parallel hyper-rectangle domain

The scheduling model discussed in §2 is very general and allows execution time vectors to belong to arbitrary convex domains. We now focus on one commonly occurring domain, viz. the class of axis-parallel hyper-rectangles (`aph`). This domain finds applicability in scheduling processes in real-time operating systems. Figure (2) is an example of this domain in 3− dimensions. By focussing on this domain, we obtain better bounds on the running time of Algorithm (5.1).

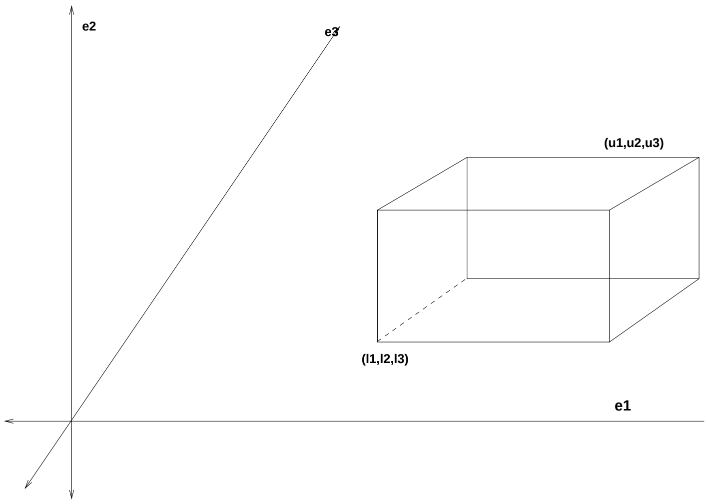

Figure 2: An axis-parallel hyper-rectangle

The Maruti Operating System [LTCA89, MAT90, MKAT92] estimates running times of tasks by performing repeated *runs* so as to determine upper and lower bounds on their execution time. Accordingly, the running time of task J_i, viz. e_i, belongs to the interval $[l_i, u_i]$, where l_i and u_i denote the lower and upper bound on the execution time as determined by the empirical observation. `These independent range variations are the only constraints on the execution times.` Observe that during actual execution, e_i can take any value in the range $[l_i, u_i]$.

Essentially, the convex domain $\mathbf{E}$ in (3) is now the axis-parallel hyper-rectangle represented by: $\mathbf{\Upsilon} = [l_1, u_1] \times [l_2, u_2] \times \ldots \times [l_n, u_n]$. The periodic scheduling query (3) for this case is:

$$\exists \vec{\mathbf{s^i}} = [s_1^i, s_2^i, \ldots, s_n^i] \forall e_1 \in [l_1, u_1] \forall e_2 \in [l_2, u_2] \ldots \forall e_n \in [l_n, u_n] \;\; A.[\vec{\mathbf{s^i}}, \vec{\mathbf{e}}] \leq \vec{\mathbf{b}}, C.[\vec{\mathbf{s^i}}, \vec{\mathbf{s^{i+1}}}, \vec{\mathbf{e}}] \leq \vec{\mathbf{d}}, \forall i = 1, 2, \ldots \infty \tag{10}$$

In this special case, variable substitution (instead of the more expensive convex minimzation) can be used to determine the vectors $\vec{\mathbf{r}}, \vec{\mathbf{q}}$ in Algorithm (5.1) and thus the periodic polytope.

Lemma 5.2 *When the domain is an* `aph`*, an affine function can be minimized by minimizing over each dimension individually.*

Proof 5.2 *Refer [Sch87].*

Lemma (5.2) gives us the following strategy to minimize an affine function $a_1.e_1 + a_2.e_2 + \ldots + a_n.e_n + c$ over an `aph` $\Upsilon = [l_1, u_1] \times [l_2, u_2] \times \ldots \times [l_n, u_n]$:

- $\forall i$, if $a_i > 0$, set $e_i = l_i$
- $\forall i$, if $a_i < 0$, set $e_i = u_i$

The resulting function can then be evaluated to yield a rational minimum.

Using this strategy it is clear that Algorithm (5.1) runs in $O((m + m')n)$ time, where m is the number of intra-period constraints and m' is the number of inter-period constraints.

6 The Periodic Scheduling Algorithm

In this section, we formalize the strategies and conclusions of §4 and §5 into Algorithm (6.1). Given a periodic system $[(1) - (2)]$, we first eliminate the execution time variables using Algorithm (5.1). The second step consists of using Algorithm (4.1) to check whether the associated periodic polytope is non-empty.

The complexity of Algorithm (6.1) is $O((m + m')C + L(m + m', n))$ for general convex domains and $O((m + m').n + L((m + m'), n))$ for `aph` domains.

Function PERIODIC SCHEDULER $(A, \vec{\mathbf{b}}, C, \vec{\mathbf{d}})$

1: Call PERIODIC-MODEL-TO-PLP $(A, \vec{\mathbf{b}}, C, \vec{\mathbf{d}})$ to get new system $G.[\vec{\mathbf{s^i}}] \leq \vec{\mathbf{r}}, I.[\vec{\mathbf{s^i}}, \vec{\mathbf{s^{i+1}}}] \leq \vec{\mathbf{q}}$.
2: Let $\vec{s} =$ PLP-SOLVE $(A, \vec{\mathbf{b}}, C, \vec{\mathbf{d}})$
3: **if** ($\vec{s} \neq \phi$) **then**
4: `System has periodic Schedule` $\vec{x}$
5: **return**
6: **else**
7: `System does not have a periodic schedule`
8: **end if**

Algorithm 6.1: Periodic Scheduler

6.1 Example

Consider the following set of specifications for a periodic task set consisting of two jobs $\{J_1, J_2\}$. Let the period of the task set i.e. L be 15 and the constraints on the execution times be $e_1 \in [2, 4], e_2 \in [1, 3]$.

1. Task J_2 starts at least 2 units after task J_1 finishes, in each period: $s_1^i + e_1 + 2 \leq s_2^i, \forall i = 1, 2, \ldots$;
2. Task J_2 starts within 5 units of task J_1 finishing, in each period: $s_2^i \leq s_1^i + e_1 + 5, \forall i = 1, 2, \ldots$;
3. Task J_1 in each period starts at least 5 units after task J_1 in the previous period finishes: $s_2^i + e_2 + 5 \leq s_1^{i+1}, \forall i = 1, 2, \ldots$;
4. Task J_2 finishes before the end of the period: $s_2^i + e_2 \leq 15 + (i-1).15, \forall i = 1, 2, \ldots$.

Expressing the constraint system in form [(1)-(2)], we get

$$\begin{bmatrix} 1 & -1 & 1 & 0 \\ -1 & 1 & -1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} . \begin{bmatrix} s_1^i \\ s_2^i \\ e_1 \\ e_2 \end{bmatrix} \leq \begin{bmatrix} -2 \\ 5 \\ 15 + (i-1).15 \end{bmatrix} \tag{11}$$

and

$$\begin{bmatrix} 0 & 1 & -1 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} s_1^i \\ s_2^i \\ s_1^{i+1} \\ s_2^{i+1} \\ e_1 \\ e_2 \end{bmatrix} \leq -5 \tag{12}$$

Applying Algorithm (6.1), we obtain the following fixed-point solution:

$$\begin{bmatrix} s_1^i \\ s_2^i \end{bmatrix} = \begin{bmatrix} 0 \\ 6 \end{bmatrix}$$

Inspection of the solution confirms that the constraint set is indeed satisfied [1].

[1] We used the LP-SOLVE program to obtain the solution [Ber95].

7 Conclusions and Future Research

In this paper, we focussed on two issues:

- Building a scheduling model that adequately captures the principal issues in periodic real-time scheduling, and
- Designing strategies for answering queries of interest in this model.

We presented the `Periodic Scheduling model` in §2, which we believe addresses the main issues in periodic real-time systems and developed a polynomial time algorithm for the *schedulability query* posed there. Our main contribution in this paper has been the proof of existence of fixed-point solutions for periodic linear programs. We proved the existence of infinite schedules through recourse to fixed-point theorems in functional analysis. We believe that it should be possible to derive fixed-point conditions, using a purely combinatorial approach.

The limitations of the static approach have been described in [SA00b, SA00d, Sak94], where explicit mention is made of the `Loss of Schedulability` phenomenon. A *parametric* approach is required to eliminate this phenomenon and our investigations along this line will be available in [SA00c]. Our work was motivated principally by the requirements of the Maruti Operating System. A survey of applications that use sophisticated constraint systems akin to the one that we studied would be informative.

References

[ABNM93] P. Ancilloti, G. Buttazo, M. Di Natale, and A.K. Mok. Tracs: A flexible real-time environment for traffic control systems. In *Proceedings IEEE Workshop on Real-Time Applications*, pages 50–53, May 1993.

[Ber95] M. Berkelaar. Linear programming solver. *Software Library for Operations Research, University of Karlsruhe*, 1995.

[BFR71] P. Bratley, M. Florian, and P. Robillard. Scheduling with Earliest Start and Due Date Constraints. *Naval. Res. Log. Quart.*, 18:511–519, Dec. 1971.

[BFW97] Azer Bestavros and Victor Fay-Wolfe, editors. *Real-Time Database and Information Systems, Research Advances.* Kluwer Academic Publishers, 1997.

[CC89] Houssine Chetto and Maryline Chetto. Scheduling periodic and sporadic tasks in a real-time system. *Information Processing Letters*, 30(4):177–184, February 1989.

[Cho97] Seonho Choi. *Dynamic Time-based scheduling for Hard Real-Time Systems.* PhD thesis, University of Maryland, College Park, jun 1997.

[Cho00] Seonho Choi. Dynamic dispatching of cyclic real-time tasks with relative time constraints. *JRTS*, pages 1–35, 2000.

[Cof76] E. G. Coffman. *Computer and Job-Shop Scheduling Theory, Ed.* Wiley, New York, 1976.

[Das85] B. Dasarathy. Timing Constraints of Real-Time Systems: Constructs for Expressing Them, Methods of Validating Them. *IEEE Transactions on Software Engineering*, SE-11(1):80–86, January 1985.

[DE73] G. B. Dantzig and B. C. Eaves. Fourier-Motzkin Elimination and its Dual. *Journal of Combinatorial Theory (A)*, 14:288–297, 1973.

[DL78] S. K. Dhall and C. L. Liu. On a real-time scheduling problem. *Operations Research*, 26(1):127–140, Jan. 1978.

[Foh95] Gerhard Fohler. Joint scheduling of distributed complex, periodic and hard aperiodic tasks in statically scheduled systems. In *Proceedings IEEE Real-Time Systems Symposium.* IEEE Computer Society Press, December 1995.

[GH93] R. Gerber and S. Hong. Semantics-based compiler transformations for enhanced schedulability. In *Proceedings IEEE Real-Time Systems Symposium*, pages 232–242. IEEE Computer Society Press, December 1993.

[GPS95] R. Gerber, W. Pugh, and M. Saksena. Parametric Dispatching of Hard Real-Time Tasks. *IEEE Transactions on Computers*, 1995.

[HKL91] M.G. Harbour, M.H. Klein, and J.P. Lehoczky. Fixed priority scheduling of tasks with varying execution priority. In *Proceedings IEEE Real-Time Systems Symposium*, page 1160128. IEEE Computer Society Press, December 1991.

[HN94] Dorit S. Hochbaum and Joseph (Seffi) Naor. Simple and fast algorithms for linear and integer programs with two variables per inequality. *SIAM Journal on Computing*, 23(6):1179–1192, December 1994.

[HuL93] J. B. Hiriart-urruty and C. Lemarechal. *Convex Analysis and Minimization Algorithms.* Springer-Verlag, 1993.

[Ist81a] Vasile I. Istratescu. *Fixed-point theory:An Introduction.* Kluwer-Boston, Inc., Hingham, MA, 1981.

[Ist81b] Vasile I. Istratescu. *Introduction to Linear Operator theory.* Marcel Dekker Inc., New York, 1981.

[Kor83] Y. Koren. *Computer Control of Manufacturing Systems.* McGraw-Hill, New York, 1983.

[LL73] C. L. Liu and James W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM*, 20(1):46–61, January 1973.

[LTCA89] S. T. Levi, S. K. Tripathi, S. D. Carson, and A. K. Agrawala. The MARUTI Hard Real-Time Operating System. *ACM Special Interest Group on Operating Systems*, 23(3):90–106, July 1989.

[MAT90] D. Mosse, Ashok K. Agrawala, and Satish K. Tripathi. Maruti a hard real-time operating system. In *Second IEEE Workshop on Experimental Distributed Systems*, pages 29–34. IEEE, 1990.

[MKAT92] D. Mosse, Keng-Tai Ko, Ashok K. Agrawala, and Satish K. Tripathi. MARUTI an Environment for Hard Real-Time Applications. In Ashok K. Agrawala, Karen D. Gordon, and Phillip Hwang, editors, *Maruti OS*, pages 75–85. IOS Press, 1992.

[Pin95] M. Pinedo. *Scheduling : theory, algorithms, and systems.* Prentice-Hall, Englewood Cliffs, 1995.

[Ram90] K. Ramamritham. Allocation and scheduling of complex periodic tasks. In *The* 10^{th} *International Conference on Distributed Computing*, pages 108–115, 1990.

[SA00a] K. Subramani and A. K. Agrawala. A dual interpretation of standard constraints in parametric scheduling. Technical Report CS-TR-4112, University of Maryland, College Park, Department of Computer Science, March 2000. Submitted to FTRTFT 2000.

[SA00b] K. Subramani and A. K. Agrawala. The parametric polytope and its applications to a scheduling problem. Technical Report CS-TR-4116, University of Maryland, College Park, Department of Computer Science, March 2000. Submitted to ESA 2000.

[SA00c] K. Subramani and A. K. Agrawala. The periodic polytope and its application to a scheduling problem – a parametric perspective. Technical report, University of Maryland, College Park, Department of Computer Science, April 2000. Manuscript in preparation.

[SA00d] K. Subramani and A. K. Agrawala. The static polytope and its applications to a scheduling problem. *Submitted to* 3^{rd} *IEEE Workshop on Factory Communications*, March 2000.

[Sak94] Manas Saksena. *Parametric Scheduling in Hard Real-Time Systems.* PhD thesis, University of Maryland, College Park, June 1994.

[Sch87] Alexander Schrijver. *Theory of Linear and Integer Programming.* John Wiley and Sons, New York, 1987.

[Sta88] J. A. Stankovic. Real-time computing systems: The next generation. In J. A. Stankovic and K. Ramamritham, editors, *Tutorial: Hard Real Time Systems*, page 14:38. IEEE, 1988.

[Str80] Gilbert Strang. *Linear Algebra and its Applications.* New York Academic Press, 1980.

[Vai87] P. M. Vaidya. An algorithm for linear programming which requires $O(((m+n)n^2+(m+n)^{1.5}n)L)$ arithmetic operations. In Alfred Aho, editor, *Proceedings of the 19th Annual ACM Symposium on Theory of Computing*, pages 29–38, New York City, NY, May 1987. ACM Press.

[Y.K80] Y.Koren. Cross-coupled biaxial computer control for manufacturing systems. *ASME Journal of Dynamic Systems, Measurement and Control*, 102:265–272, 1980.