The Parametric Polytope and its applications to a Scheduling Problem

K. Subramani [*] Ashok Agrawala [†]

Abstract

An important feature in Real-time systems is *parameter impreciseness* i.e. the inability to accurately determine certain parameter values. The most common such parameter is *task execution time*. A second feature is the presence of complex relationships between tasks that constrain their execution. Traditional models do not accomodate either feature completely: (a) Variable execution times are modeled through a fixed value (*worst-case*), and (b) Relationships are limited to those that can be represented by precedence graphs. We present a task model that effectively captures *variable task execution time*, while simultaneously permitting arbitrary linear relationships between tasks. Our model finds applications in diverse areas such as real-time task scheduling, compiler scheduling, real-time database scheduling and machine control. This paper focuses primarily on the computational complexity of answering queries posed in our model; in particular we demonstrate the existence of constraint classes that make the scheduling problem *hard.*

1 Introduction

One of the primary concerns in Real-Time System design is the accomodation of *non-constant* parameters. Task execution time is a prime example of such a parameter. A scheduling model that ignores variability in parameters, runs the risk of a catastrophic breakdown during execution [SA00b]. A second feature, peculiar to real-time systems is the existence of complex constraints between tasks [Sta88, LTCA89, Das85, JM86]. Traditional models such as the ones proposed in [Cof76, DL78] and [Pin95, Bru81] used fixed values for process time. Likewise, relationships between tasks are restricted to precedence constraints. Our goals in this paper are twofold:

- Proposing a model that captures the intricacies involved in real-time scheduling, and
- Studying the complexity of queries posed in this model.

Our model is very general in that a wide range of issues in real-time scheduling can be accomodated. Interaction between execution times of processes are explicitly modeled through convex domains. This captures the nature of scheduling in real-time applications such as machine control [SA00b, Y.K80]. However we pay a price for this generality; determining schedulability for certain classes of constraints becomes inherently difficult. The rest of this paper is organized as follows: Section §2 describes the general parametric scheduling model in detail and contrasts it with static scheduling models [SA00b]. We present the *parametric schedulabilty query* as part of the model. The restriction of the general model to domains where the execution times belong to axis-parallel hyper-rectangles is discussed in §2.1. Indeed, we shall be studying complexity issues for this restricted domain only. In the succeeding section, we discuss the motivation for our model, through examples from problem domains. We also discuss related approaches to our problem. Our analysis commences in §4, where we complement the parametric schedulabilty query and use properties of the model to convert the complement to a global minimization problem. In §5, the computational complexity of the parametric query is addressed and we provide proof of its intractabilty. Polynomial time algorithms for special cases are discussed in §6. We conclude in §7 by summarizing our contributions and mentioning some open problems of interest.

[*]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

2 The Parametric Scheduling Model

We are a given a set of ordered non-preemptive tasks $\{J_1, J_2, \ldots J_n\}$, with linear constraints imposed on their respective start times $\{s_1, s_2, \ldots, s_n\}$ and execution times $\{e_1, e_2, \ldots, e_n\}$. The constraint system is expressed in matrix form as :

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

where,

- $\vec{\mathbf{s}} = [s_1, s_2, \ldots, s_n]$ is an $n-$vector of the start times of the tasks,
- $\vec{\mathbf{e}} = [e_1, e_2, \ldots, e_n]$ is an $n-$vector of the execution time of the tasks,
- A is a $m \times 2.n$ matrix of rational numbers,
- $\vec{\mathbf{b}} = [b_1, b_2, \ldots, b_m]$ is an $m-$vector of rational numbers.

System (1) is a convex polyhedron in the $2.n$ dimensional space, spanned by the start time axes $\{\vec{\mathbf{s_1}}, \vec{\mathbf{s_2}}, \ldots, \vec{\mathbf{s_n}}\}$ and the execution time axes $\{\vec{\mathbf{e_1}}, \vec{\mathbf{e_2}}, \ldots, \vec{\mathbf{e_n}}\}$. The execution time of the i^{th} task e_i is *not constant*, but belongs to the set E_i where E_i is the projection of a convex set $\mathbf{E}$ on axis $\vec{\mathbf{e_i}}$. 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 by the set $\mathbf{E}$. We regard the execution times as $n-$vectors belonging to the set $\mathbf{E}$.

The goal is to come up with a start time vector $\vec{\mathbf{s}}$, that satisifes the constraint system (1), for all execution time vectors belonging to the set $\mathbf{E}$. One way of approaching this problem is through *Static Scheduling* techniques, as discussed in [SA00b]. However, Static Scheduling results in the phenomenon known as *loss of schedulability* discussed below.

Consider the two task system $J = \{J_1, J_2\}$ with start times $\{s_1, s_2\}$, execution times in the set $\{(e_1 \in)[2, 4] \times (e_2 \in)[4, 5]\}$ and the following set of constraints:

- Task J_1 must finish before task J_2 commences; i.e. $s_1 + e_1 \leq s_2$;
- Task J_2 must commence within 1 unit of J_1 finishing; i.e. $s_2 \leq s_1 + e_1 + 1$;

A static approach forces the following two constraints:

- $s_1 + 4 \leq s_2$,
- $s_2 \leq s_1 + 2 + 1 \Rightarrow s_2 \leq s_1 + 3$

Clearly the resultant system is inconsistent and there is no static solution. Now consider the following start time vector assigment:

$$\vec{\mathbf{s}} = \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} = \begin{bmatrix} 0 \\ s_1 + e_1 \end{bmatrix} \tag{2}$$

This assignment clearly satisfies the input set of constraints and is hence a valid solution. The key feature of the solution provided by (2) is that the start time of task J_2 is no longer an absolute time, but a (parameterized) function of the start and execution times of task J_1. This phenomenon in which a static scheduler declares a system infeasible in the presence of a valid solution (albeit parameterized) is termed as *loss of schedulability.*

In the parametric scheduling model, we are interested in checking whether an input constraint system has a *parameteric schedule*, i.e. a schedule in which the start time of a task can depend on the start and execution times of tasks that are sequenced before it.

Definition 2.1 *A parametric solution of an ordered set of tasks, subject to a set of linear relative constraints (expressed by (1)) is a vector* $\vec{\mathbf{s}} = [s_1, s_2, \ldots, s_n]$*, where* s_1 *is a rational number and each* $s_i, i \neq 1$ *is a function of the variables* $\{s_1, e_1, s_2, e_2, \ldots, s_{i-1}, e_{i-1}\}$*. Further, this vector should satisfy the constraint system, for all vectors* $\vec{\mathbf{e}} \in E$*.*

Based on the discussion above, we are in a position to state the parametric schedulabilty query:

$$\exists s_1 \forall e_1 \in E_1 \exists s_2 \forall e_2 \in E_2, \ldots \exists s_n \forall e_n \in E_n \quad A.[\vec{\mathbf{s}}, \vec{\mathbf{e}}] \leq \vec{\mathbf{b}} \quad ? \tag{3}$$

2.1 Axis-parallel hyper-rectangle domain

The parametric scheduling model discussed above is very general and allows execution time vectors to belong to arbitrary convex domains. In this paper, we focus on one particular domain, viz. the class of axis-parallel hyper-rectangle, refer Figure (1). This domain finds applicability scheduling processes in real-time operating systems.

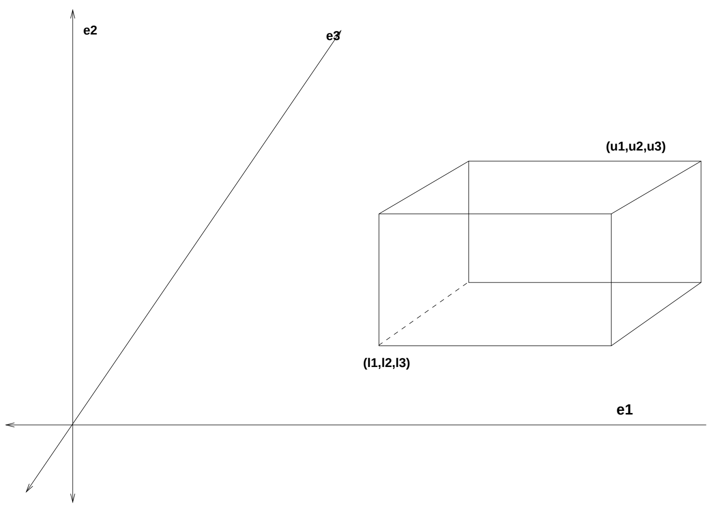

Figure 1: An axis-parallel hyper-rectangle

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 a task J_i i.e. 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.

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 parametric scheduling query (3) for this case is:

$$\exists s_1 \forall e_1 \in [l_1, u_1] \exists s_2 \forall e_2 \in [l_2, u_2] \ldots \exists s_n \forall e_n \in [l_n, u_n] \qquad A.[\vec{\mathbf{s}}, \vec{\mathbf{e}}] \leq \vec{\mathbf{b}} \text{ ?} \tag{4}$$

For the rest of the paper, we shall focus on the complexity of query (4).

2.2 Obviation of Parametric functions through dynamic dispatching

Lower bound function	$\leq$ Start time $\leq$	Upper bound function
a	s_1	b
$f_1(s_1, e_1)$	s_2	$f'_1(s_1, e_1)$
$f_2(s_1, e_1, s_2, e_2)$	s_3	$f'_2(s_1, e_1, s_2, e_2)$
$\vdots$	$\vdots$	$\vdots$
$f_{n-1}(s_1, e_1, s_2, e_2, \ldots, s_{n-1}, e_{n-1})$	s_n	$f'_{n-1}(s_1, e_1, s_2, e_2, \ldots, s_{n-1}, e_{n-1})$

Table 1: List of parametric functions

In [Sak94], query (4) was answered by providing a parametrized list of linear functions for each start time s_i, as shown in Table (2.2). During actual execution, s_1 can take on any value in the range $[a, b]$. Upon termination of task J_1, we know e_1 which along with s_1 can be plugged into $f_1()$ and $f_1'()$ thereby providing a range $[a', b']$ for s_2 and so on, till task J_n is scheduled and completes execution.

We argue here that explicit construction of the parameterized function list is unnecessary. Determination of feasibility is sufficient, thereby eliminating the need for storing the parameterized function list. Observe that at any point in the scheduling window, the first task that has not yet been scheduled has a start time that is independent of the start and execution times of all other tasks. Once this task is executed, we can determine a rational range e.g.$[a'', b'']$ for the succeeding task and the same argument applies to this task. In essence, all that we require to do is determine the start time of the first `unexecuted` task in the sequence. Let us assume the existence of an oracle $\mathbf{\Omega}$ that decides query (4). The following methods can be used to determine the start time of the first *unexecuted* task (say J_ρ)[1] in the schedule:

1. The bisection method - Let M be the deadline of J_n. We must have $s_\rho \leq M$ and the goal is to determine the exact value we can safely assign to s_ρ, without violating the constraint set. We assume that a call to the oracle $\mathbf{\Omega}$ has determined the existence of a parametric schedule for the constraint system $A.[\vec{\mathbf{s}}, \vec{\mathbf{e}}] \leq \vec{\mathbf{b}}$.

```
Function DETERMINE-START $(A, \vec{\mathbf{b}}, M)$
1: $m' = \frac{M}{2}$
2: if $(\Omega(A.[\vec{\mathbf{s}}, \vec{\mathbf{e}}] \leq b, s_1 \geq m'))$ then
3:     $s_\rho = m.$
4:     return
5: else
6:     DETERMINE-START $(A, \vec{\mathbf{b}}, m)$
7: end if
```

Algorithm 2.1: Online Dispatcher

The cost of this strategy is $O(\log M)$ calls to the oracle $\mathbf{\Omega}$.

2. We can improve the online dispatching time as follows: Let UP_ρ denote the set of constraints that can be written in the form $s_\rho \geq ()$. Clearly these are the only constraints that prevent s_ρ from being assigned the value $\overline{a}$, where $\overline{a}$ is the time at which task $J_{\rho-1}$ finished execution. Inspection of UP_ρ provides a suitable assignment for s_ρ. The online dispatching time is the size of UP_ρ, which could be $O(m.n)$ in the worst case.

Thus, we have established that the principal complexity of the parametric scheduling problem is in determining the answer to query (4)

3 Motivation and Related Work

Real-Time Operating Systems such as Maruti [LTCA89, MAT90, MKAT92] and MARS [DRSK89], permit interaction of processes through linear relationships, between their start and execution times. The Real-time specification Language MPL (Maruti Programming Language) [SdSA94] explicitly includes programmer constructs such as:

- **within** 10 ms; **do**
  Perform Task 1 od
- **Perform Task 1**;
  Delay at most 17 ms;
  Perform Task 2

These constructs are easily transformed into linear constraints between the start and execution times of the tasks. For instance, the first construct can be expressed as: $s_1 \geq 10$, while the second construct is captured through: $s_2 \geq f_1 + 17$. Note that f_1 is the finish time of task 1 and since we are dealing with non-preemptive tasks, we can write $f_i = s_i + e_i, \forall i$, where f_i denotes the finish time of task i.

[1]At commencement, $\rho = 1$

Other application areas include avionics (flight control) [SA00b], machining [Y.K80, Kor83, SE87, SK90] and real-time databases [BFW97]. A detailed survey of applications and models can be found in [Sub00].

The *parametric model* for **axis-parallel hyper-rectangle** domains was proposed in [Sak94]. In [GPS95], polynomial time algorithms were presented for the case, where the constraints were restricted to be "standard" i.e. *monotone* constraints involving the start times of at most two tasks. In [SA00a], we provided a dual interpretation of the *standard constraint* case and provided polynomial time algorithms for arbitrary convex domains. We also showed that a fast implementation of the Fourier-Motzkin elimination procedure [HN94] could be used to determine parametric feasibility in case of arbitrary **network constraints** i.e. constraints of the form $a.s_i + b.s_j \le c.e_i + d.e_j; a, b, c, d \in \mathcal{Q}$. However, this paper represents the first attempt to study the problem from a computational complexity perspective and our investigations reveal that the parametric schedulability query is intrinsically hard (§4).

4 Complement of Parametric Scheduling

We commence our analysis by looking at the complement of the parametric scheduling query (4). Observe that the answer to the complement of a query is true iff the answer to the query is false. The complement of query (4) is:

$$\neg(\exists s_1 \forall e_1 \in E_1 \exists s_2 \forall e_2 \in E_2, \ldots \exists s_n \forall e_n \in E_n \quad A[\vec{\mathbf{s}}, \vec{\mathbf{e}}] \le \vec{\mathbf{b}} \quad ?), \tag{5}$$

where $E_i = [l_i, u_i]$.

$$\Rightarrow \forall s_1 \exists e_1 \in [l_1, u_1] \forall s_2 \exists e_2 \in [l_2, u_2], \ldots \forall s_n \exists e_n \in [l_n, u_n] \quad A[\vec{\mathbf{s}}, \vec{\mathbf{e}}] \not\le \vec{\mathbf{b}} \quad ?$$

But the execution times are independent of the start times of the tasks and hence we can restate the query above as:

$$\exists e_1 \in [l_1, u_1] \exists e_2 \in [l_2, u_2], \ldots \exists e_n \in [l_n, u_n] \forall s_1 \forall s_2, \ldots \forall s_n \quad A[\vec{\mathbf{s}}, \vec{\mathbf{e}}] \not\le \vec{\mathbf{b}} \quad ? \tag{6}$$

which implies

$$\exists \vec{\mathbf{e}} = [e_1, e_2, \ldots, e_n] \in \mathbf{\Upsilon} \forall s_1 \forall s_2, \ldots \forall s_n \quad A[\vec{\mathbf{s}}, \vec{\mathbf{e}}] \not\le \vec{\mathbf{b}} \quad ? \tag{7}$$

Query (7) basically asks whether there exists an execution time vector $\vec{\mathbf{e}} = [e'_1, e'_2, \ldots, e'_n] \in \mathbf{\Upsilon}$ such that the linear system resulting from substituting these execution times in $A.[\vec{\mathbf{s}}, \vec{\mathbf{e}}] \le \vec{\mathbf{b}}$ is infeasible, i.e. (7) asks whether the polyhedral set $\{\vec{\mathbf{s}}:A.[\vec{\mathbf{s}}.\vec{\mathbf{e}}] \le \vec{\mathbf{b}} | \vec{\mathbf{e}} = [e'_1, e'_2, \ldots, e'_n]\}$ is empty.

Thus, the existence of such an execution time vector, or *witness vector* would imply that the non-existence of a parametric schedule. On the other hand proof of non-existence of any such vector implies that query (4) can be answered in the affirmative and that there exists a parametric schedule.

4.1 The Parametric Dual

We first rewrite the constraint system (1) in the form:

$$G.\vec{\mathbf{s}} \le \vec{\mathbf{b}} - B.\vec{\mathbf{e}} \tag{8}$$

where,

$$A.[\vec{\mathbf{s}}, \vec{\mathbf{e}}] = G.\vec{\mathbf{s}} + B.\vec{\mathbf{e}}$$

and G and B are $m \times n$ rational matrices.

Accordingly, query (7) gives

$$\exists \vec{\mathbf{e}} = [e_1, e_2, \ldots, e_n] \forall s_1 \forall s_2, \ldots \forall s_n \quad G.\vec{\mathbf{s}} \not\le \vec{\mathbf{b}} - B.\vec{\mathbf{e}} \quad ? \tag{9}$$

Note that $\vec{\mathbf{b}} - B.\vec{\mathbf{e}}$ is an $m-$vector, with each element being an affine function in the e_i variables. We set $\vec{\mathbf{g}} = \vec{\mathbf{b}} - B.\vec{\mathbf{e}}$, so that we can rewrite query (9) as

$$\exists \vec{\mathbf{e}} = [e_1, e_2, \ldots, e_n] \forall s_1 \forall s_2, \ldots \forall s_n \quad G.\vec{\mathbf{s}} \not\le \vec{\mathbf{g}}? \tag{10}$$

In order to find an execution time vector, which serves as a witness to the infeasibilty of the input constraint system, we study the dual of the complement problem. The following lemma called Farkas' lemma [NW88, Sch87] is crucial to understanding and analyzing the dual.

Lemma 4.1 *Either* $\{\vec{\mathbf{x}} \in R^n_+ : A\vec{\mathbf{x}} \le \vec{\mathbf{b}}\} \ne \phi$ *or (exclusively)* $\exists \vec{\mathbf{y}} \in R^m_+$*, such that,* $\vec{\mathbf{y}}^{\mathbf{T}} A \ge \vec{\mathbf{0}}^T$ *and* $\vec{\mathbf{y}}^{\mathbf{T}}.\vec{\mathbf{b}} = -\infty$*.*

Proof 4.1 *See [Sch87, PS82, NW88].*

The lemma is interpreted as follows: If the primal system is infeasible, then there exists a proof of this infeasibility. This proof takes the form of a witness vector which is unbounded in the dual space.

Applying the lemma to our problem, we note that query (10) requires the system $G.\vec{s} \le \vec{\mathbf{g}}$ to be infeasible for a particular $\vec{\mathbf{e}} \in \mathbf{\Upsilon}$. Farkas' lemma assures us that this is possible only if $\exists \vec{\mathbf{y}} \in R^+_m$, such that

$$\vec{\mathbf{y}}^T.G \ge \vec{\mathbf{0}}, \quad \vec{\mathbf{y}}^{\mathbf{T}}.(\vec{\mathbf{b}} - B.\vec{\mathbf{e}}) = -\infty \tag{11}$$

which implies that

$$G^T.\vec{\mathbf{y}} \ge \vec{\mathbf{0}}, \quad \vec{\mathbf{y}}^{\mathbf{T}}.(\vec{\mathbf{b}} - B.\vec{\mathbf{e}}) = -\infty. \tag{12}$$

Equation (12) is interpreted algorithmically in the following way:

Let z be the minimum of the bilinear form $\vec{\mathbf{y}}^{\mathbf{T}}.(\vec{\mathbf{b}} - B.\vec{\mathbf{e}})$ over the two convex bodies :$\{\vec{\mathbf{y}} : \vec{\mathbf{y}} \ge \vec{\mathbf{0}}, G^T.\vec{\mathbf{y}} \ge \vec{\mathbf{0}}\}$ and $\mathbf{\Upsilon}$. If $z = -\infty$, the input system of constraints does not have a parametric schedule.

We make the following observations:

1. From an algorithmic perspective, we need only check if $z < 0$; in the dual cone i.e. $G^T.\vec{\mathbf{y}} \ge \vec{\mathbf{0}}$, $z < 0 \Rightarrow z = -\infty$ [Sch87].

2. The bilinear form $z = \vec{\mathbf{y}}^{\mathbf{T}}.(\vec{\mathbf{b}} - B.\vec{\mathbf{e}})$ can be expressed as the following quadratic form:

$$z = [y_1, y_2, \ldots, y_m, e_1, e_2, \ldots, e_n]\mathcal{F} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ e_n \end{bmatrix}, \tag{13}$$

   for suitably chosen $\mathcal{F}$.

3. The axis-parallel hyper-rectangle $\mathbf{\Upsilon}$ is a polyhedron and can be expressed in the form $\{\vec{\mathbf{e}} : N.\vec{\mathbf{e}} \le \vec{\mathbf{f}}\}$. The minimum of the bilinear form is achieved at a vertex (extreme point) of this polyhedron [BSS93]. This implies that if a *witness* execution time vector exits, then there must exist a witness vector $\vec{\mathbf{e}}$ with elements $e_i \in \{l_i, u_i\}$ i.e. only the end-points of the execution time ranges matter.

4. Let us assume the existence of the following constraint $s_1 \ge \epsilon$ i.e. we are precluding $s_1 = 0$ in any solution. We use the *Complementary Slackness* property [NW88] to infer that the dual variables $\vec{\mathbf{y}}$ must satisy $G^T.\vec{\mathbf{y}} = \vec{\mathbf{0}}$

The complement of the parametric scheduling query (called the *co-scheduling query* henceforth) is thus:

$$\exists \vec{\mathbf{y}} \ge \vec{\mathbf{0}} \in G^T.\vec{\mathbf{y}} = \vec{\mathbf{0}}, \vec{\mathbf{e}} \in \mathbf{\Upsilon} = \{N.\vec{\mathbf{e}} \le \vec{\mathbf{f}}\}, s.t. \quad [\vec{\mathbf{y}}, \vec{\mathbf{e}}]^T \mathcal{F} [\vec{\mathbf{y}}, \vec{\mathbf{e}}] = -\infty \quad ? \tag{14}$$

In the succeeding section, we analyze the complexity of the co-scheduling query (14).

5 Complexity of Parametric Scheduling

We now show that we cannot hope to do substantially better in the deterministic case.

Lemma 5.1 *The co-scheduling query is in the class NP*

Proof 5.1 *From the discussion in §4, we know that optimality occurs at an extreme point of the polyhedron* $\Upsilon = \{N.\vec{\mathbf{e}} \leq \vec{\mathbf{f}}\}$. *Hence an oracle need only guess one of these points and check that the corresponding linear function in* $\vec{\mathbf{y}}$ *is unbounded below in* $G^T.\vec{\mathbf{y}} = \vec{\mathbf{0}}$. *Since all the extreme point compenents belong* $[o\ l_i, u_[i]$, *the lemma follows.*

Lemma 5.2 *Let* A *be a* $p \times q, p < q$ *matrix with entries in the set* $\{0, 1, -1\}$. *The query,*

$$\exists \vec{\mathbf{x}}, s.t. \quad A.\vec{\mathbf{x}} = \vec{\mathbf{0}}, x_i \in \{0, 1\}, \sum_{i=1}^{n} x_i \geq p \quad ?,$$

is NP-complete.

Proof 5.2 *See [GJ79, CLR92]. The above query is basically asking whether there is a a* $p-$*subset of columns of* A*, such that the submatrix formed by those columns is singular. In fact, this problem is NP-complete, even if each row has at most* 3 *non-zero entries (by a reduction from* $3 - SAT$ *).*

Theorem 5.1 *The co-scheduling is NP-complete.*

Proof 5.3 *Clearly, any non-trivial solution* $\vec{\mathbf{y}}$ *to* $G^T.\vec{\mathbf{y}} = \vec{\mathbf{0}}$*, identifies a* $n-$*subset of columns of* G^T *such that the submatrix formed by those columns is singular. Thus it subsumes the query in Lemma 5.2. The NP-completeness of co-scheduling follows* [2]. *We note that:*

- *The co-scheduling query is* `strongly` *NP-complete, since the problem is hard, even when the entries in* G *belong to the set* $\{-1, 0, 1\}$.
- *The query is NP-complete, when each row has at most* 3 *non-zero entries.*

Corollary 5.1 *Parametric Scheduling is coNP-complete.*

Proof 5.4 *Follows from the completeness of the co-scheduling query for the class NP.*

6 Special Cases

In this section, we enumerate the cases where it is possible to determine parametric schedulability in polynomial time.

1. The matrix $\mathcal{F}$ in query (14) is positve semidefinite - In this case, we can use the ellipsoid algorithm [HuL93, Kha79], which is guaranteed to run in polynomial time. If the global minimum of (14) is less than 0, we know that the constraint system does not have a parametric schedule.
2. There are at most two start time variables per constraint - This case has been the focus of [SA00a], where Fourier-Motzkin elimination and polytope collapsing are used to present polynomial time algorithms to determine the existence of parametric schedules.

[2] A proof based on the completeness of bilinear programming is given in [HT90]

7 Conclusion

In this paper, our focus was to approach the problem of parametric scheduling from the perspective of computational complexity. Previous research had determined that the existence of a parametric schedule over a restricted set of constraints and restricted execution domains could be determined in polynomial time. However, no attempt was made to characterize the complexity of this problem.

Our work establishes that the problem is intractable in a general setting and it is unlikely that new strategies will succeed in finding a parametric schedule efficiently. In fact, we showed that the problem was intractable when there were at most 3 start times per constraint and all execution times belonged to the set $[0, 1]$ i.e. **strongly NP-complete**. One of the principal insights was the conversion of the alternating quantifier form into a global minimization problem, through appropriate results from polyhedral combinatorics. Through our analysis, we also showed that determining parametric feasibility in polynomial time is possible in two special cases.

An interesting open problem is determining the performance of approximation algorithms over the bilinear dual.

8 Acknowledgements

We thank David Mount, David Kueker and Samir Khuller for helpful discussions.

References

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

[Bru81] P. Brucker. *Scheduling.* Akademische Verlagsgesellschaft, Wiesbaden, 1981.

[BSS93] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty. *Nonlinear Programming: Theory and Algorithms.* John Wiley, New York, second edition, 1993.

[CLR92] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to algorithms.* MIT Press and McGraw-Hill Book Company, 6th edition, 1992.

[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.

[DL78] M. I. Dessouky and R. E. Larson. Symmetry and optimality properties of the single machine problem. *AIIE Transactions*, 10(2):170–175, June 1978.

[DRSK89] A. Damm, J. Reisinger, W. Schwabl, and H. Kopetz. The Real-Time Operating System of MARS. *ACM Special Interest Group on Operating Systems*, 23(3):141–157, July 1989.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability, a Guide to the Theory of NP-Completeness.* W. H. Freeman Company, San Francisco, 1979.

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

[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.

[HT90] R. Horst and H. Tuy. *Global Optimization.* Springer, Berlin, 1990.

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

[JM86] F. Jahanian and A.K. Mok. Safety analysis of timing properties in real-time systems. *IEEE Transactions on Software Engineering*, SE-12(9):890–904, September 1986.

[Kha79] L. G. Khachiyan. A polynomial algorithm in linear programming. *Doklady Akademiia Nauk SSSR*, 224:1093–1096, 1979. English Translation: *Soviet Mathematics Doklady*, Volume 20, pp. 1093–1096.

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

[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.

[NW88] G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization.* John Wiley & Sons, New York, 1988.

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

[PS82] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization.* Prentice Hall, 1982.

[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 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.

[SdSA94] M. Saksena, J. da Silva, and A. Agrawala. Design and Implementation of Maruti-II. In Sang Son, editor, *Principles of Real-Time Systems.* Prentice Hall, 1994. Also available as CS-TR-2845, University of Maryland.

[SE87] K. Shin and M. Epstein. Intertask communication in an integrated multi-robot system. *IEEE Journal of Robotics and Automation*, 1987.

[SK90] K. Srinivasan and P.K. Kulkarni. Cross-coupled control of biaxial feed drive mechanisms. *ASME Journal of Dynamic Systems, Measurement and Control*, 112:225–232, 1990.

[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.

[Sub00] K. Subramani. Real-time scheduling-models, algorithms and complexity. Technical report, University of Maryland, College Park, Department of Computer Science, June 2000. Manuscript in preparation.

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