A Dual Interpretation of "Standard Constraints" in Parametric Scheduling

K. Subramani * Ashok Agrawala

Abstract

The problem of *parametric scheduling* in hard real-time systems, (in the presence of linear relative constraints between the start and execution times of tasks) was posed in [Sak94] and [GPS95]. In [GPS95], a polynomial time algorithm is presented for the case when the constraints are restricted to be *standard* (defined in §6) and the execution time vectors belong to an axis-parallel hyper-rectangle. In this paper, we extend their results in two directions. We first present a polynomial time algorithm for the case when the execution time vectors belong to arbitrary convex domains. We then show that the set of *standard constraints* can be extended to include arbitrary network constraints. Our insights into the problem occur primarily as a result of studying the dual polytope of the constraint system.

1 Introduction

The problem of *parametric scheduling* for hard real-time systems was introduced in [Sak94]. In particular, they considered the scheduling of processes subject to linear relative constraints between the start and execution times of tasks. In [GPS95], a polynomial time algorithm was presented for the case, where the constraints are "standard" (defined in §6). In this paper, we extend the results in [GPS95] in the following two ways:

- We present a polynomial time algorithm for parametric scheduling when the execution time vectors belong to arbitrary convex domains,
- We extend the class of parametrically schedulable constraints to include arbitrary network constraints.

Our insights into the problem occur primarily as a result of studying the dual polytope of the constraint system. In Section §2, we present the parametric scheduling model and pose the *parametric schedulabity query.* In the suceeding section, viz. §3, we discuss the motivation behind the problem and related approaches in the litreature.

Section §4 commences our analysis by looking at the complement of the parametric scheduling problem. In the succeeding section viz. §5, we study the dual of the complement problem and apply Farkas' lemma to derive the termination condition of our algorithm. §6 presents the "Standard Constraints Model". We also discuss the structure of the *standard constraint matrix* and interpret the complement of the parametric schedulabity query in this model. We show that we show that the infeasibility of the input constraint system coincides with the existence of a loop having infinite negative cost in a certain weighted graph. This implies that a *symbolic* version of the Bellman-Ford Algorithm for the Single-Source-Shortest-Paths problem (SSSP) in a network can be used to solve the parametric scheduling problem. Section §7 provides such an algorithm, while §7.1 discusses its correctness and complexity. Section §8 deals with extending the class of constraints, for which we can efficiently provide a parametric schedule. In particular, we provide a polynomial time algorithm when arbitrary network constraints exist between tasks. We conclude in §9 by summarizing our results and posing problems for further research.

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

2 The Parametric 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 interdependency 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*, as discussed below.

For example, consider the two task system $J = \{J_1, J_2\}$ with start times $\{s_1, s_2\}$, execution times $\{e_1 \in [2, 4], e_2 \in [4, 5]\}$ 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:

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

The elimination strategies used in [GPS95] establish that a parametric schedule need only have linear functions.

3 Motivation and Related Work

Our investigations have been motivated by two orthogonal concerns viz. real-time operating systems and real-time applications.

In real-time operating systems like Maruti [LTCA89, MAT90, MKAT92] and MARS [DRSK89], the interaction of processes is constrained through linear relationships between their start and execution times. Real-time specification languages like the Maruti Specification Language [SdSA94] permit 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.

The automation of machining operations [Y.K80, Kor83, SE87, SK90] provides a rich source of problems in which execution time vectors belong to convex domains. Consider the contouring system described in [TSYT97], where the task is to machine a workpiece through cutting axes. In general, there are multiple axes of motion that move with different velocities. In a two axis system, a typical requirement would be to constrain the sum of the velocities of the axes to exceed a certain quantity. This is captured through:$e_1 + e_2 \geq a$.

Real-time database applications involve the scheduling of transactions and the execution of these transactions is constrained through linear relationships [BFW97]. More applications of constrained scheduling models can be found in [STA00, SA00a].

Deterministic sequencing and scheduling have a rich history of research [BS74, DL78, Cof76]. Our focus is on a particular scheduling model viz. the *parametric scheduling* model proposed in [Sak94]. In [GPS95] a polynomial time algorithm is presented for the *standard constraints* case, when the execution time vectors belong to an axis-parallel hyper-rectangle. They use the Fourier-Motzkin (FM) elimination method [Sch87] to successively eliminate the variables in query (3). The FM algorithm takes exponential time in the worst case; in the case where the constraints are *standard* they show that they can prevent the exponential increase in the number of constraints. Hochbaum, et. al. [HN94] have shown that it is possible to implement FM elimination in strongly polynomial time for *network* constraints. We shall be using their result in §8 to provide a polytope inclusion algorithm that determines parametric feasibility. In a previous paper [SA00a], we showed that the parametric scheduling problem is NP-complete in the general case. It was also established that it is sufficient to determine whether a system is *parametrically schedulable*; explict construction of the parametric functions is not necessary.

In this paper, we extend the results in [GPS95] to provide polynomial time algorithms for more general domains (arbitrary convex domains) and constraint sets (arbitrary network constraints).

4 Complement of Parametric Scheduling

We commence our analysis by looking at the complement of the parametric scheduling query (3). Observe that the answer to the complement of a query is true iff the answer to the query is false. The complement of query (3) is:

$$\neg(\exists s_1 \forall e_1 \in E_1 \exists s_2 \forall e_2 \in E_2, \ldots \exists s_n \forall e_n \in E_n \quad A[\vec{\mathbf{s}}, \vec{\mathbf{e}}] \leq \vec{\mathbf{b}} \quad ?), \tag{4}$$

which gives

$$\forall s_1 \exists e_1 \in E \forall s_2 \exists e_2 \in E, \ldots \forall s_n \exists e_n \in E \quad A[\vec{\mathbf{s}}, \vec{\mathbf{e}}] \not\leq \vec{\mathbf{b}} \quad ?$$

As observed in [SA00a], the execution times are independent of the start times of the tasks and hence we can restate the query above as:

$$\exists e_1 \in E_1 \exists e_2 \in E_2, \ldots \exists e_n \in E_n \forall s_1 \forall s_2, \ldots \forall s_n \quad A[\vec{\mathbf{s}}, \vec{\mathbf{e}}] \not\leq \vec{\mathbf{b}} \quad ? \tag{5}$$

which implies

$$\exists \vec{\mathbf{e}} = [e_1, e_2, \ldots, e_n] \forall s_1 \forall s_2, \ldots \forall s_n \quad A[\vec{\mathbf{s}}, \vec{\mathbf{e}}] \not\leq \vec{\mathbf{b}} \quad ? \tag{6}$$

Query (6) basically asks whether there exists an execution time vector $\vec{\mathbf{e}} = [e'_1, e'_2, \ldots, e'_n] \in \mathbf{E}$ such that the linear system resulting from substituting these execution times in $A.[\vec{\mathbf{s}}, \vec{\mathbf{e}}] \leq \vec{\mathbf{b}}$ is infeasible, i.e. as shown in [SA00a], (6) asks whether the polyhedral set $\{\vec{\mathbf{s}}{:}A.[\vec{\mathbf{s}}.\vec{\mathbf{e}}] \leq \vec{\mathbf{b}} | \vec{\mathbf{e}} = [e'_1, e'_2, \ldots, e'_n]\}$ is empty.

For the rest of the paper, we focus on finding such a *witness* execution time vector; if we succeed in finding such a vector, it means that the input system does not have a parametric schedule. On the other hand, if we can say definitely, that no such execution vector exists within the convex domain E, then query (3) can be answered in the affirmative and the input system does have a parametric schedule.

5 The Parametric Dual

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

$$G.\vec{\mathbf{s}} \leq \vec{\mathbf{b}} - B.\vec{\mathbf{e}} \tag{7}$$

where,

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

Accordingly, query (6) gives

$$\exists \vec{\mathbf{e}} = [e_1, e_2, \ldots, e_n] \forall s_1 \forall s_2, \ldots \forall s_n \quad G.\vec{\mathbf{s}} \not\leq \vec{\mathbf{b}} - B.\vec{\mathbf{e}} \quad ? \tag{8}$$

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 (8) as

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

The matrix G will henceforth be referred to as the *constraint matrix*. Note that G is a $m \times n$ rational matrix.

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 5.1 *Either* $\{\vec{\mathbf{x}} \in R^n_+ : A\vec{\mathbf{x}} \leq \vec{\mathbf{b}}\} \neq \phi$ *or (exclusively)* $\exists \vec{\mathbf{y}} \in R^m_+$*, such that,* $\vec{\mathbf{y}}^{\mathbf{T}} A \geq \vec{\mathbf{0}}^T$ *and* $\vec{\mathbf{y}}^{\mathbf{T}}.\vec{\mathbf{b}} = -\infty$.

Proof 5.1 *See [Sch87, PS82, NW88].*

The lemma is interpreted as follows: Either the primal system viz. $\{A.[\vec{\mathbf{x}}] \leq \vec{\mathbf{b}}, \vec{\mathbf{x}} \geq \vec{\mathbf{0}}\}$ is feasible, in which case the associated polyhedron is non-empty *or* (exclusively) the vector $\vec{\mathbf{b}}$ lies in the polar cone of of the dual space viz. $\{\vec{\mathbf{y}}^{\mathbf{T}}.\mathbf{A} \geq \vec{\mathbf{0}}, \vec{\mathbf{y}} \geq \vec{\mathbf{0}}\}$. In the latter case, the function $\vec{\mathbf{y}}^{\mathbf{T}}.\vec{\mathbf{b}}$ is unbounded below and its minimum is $-\infty$. For a geometric interpretation of the lemma, refer [PS82].

Query (9) requires the system $G.\vec{\mathbf{s}} \leq \vec{\mathbf{g}}$ to be infeasible for a particular $\vec{\mathbf{e}} \in \mathbf{E}$. Farkas' lemma assures us that this is possible only if $\exists \vec{\mathbf{y}'} \in R^+_m$, such that

$$\vec{\mathbf{y}'}^T.G \geq \vec{\mathbf{0}}, \quad \vec{\mathbf{y}'}^{\mathbf{T}}.(\vec{\mathbf{b}} - B.\vec{\mathbf{e}}) = -\infty \tag{10}$$

which implies that

$$G^T.\vec{\mathbf{y}'} \geq \vec{\mathbf{0}}, \quad \vec{\mathbf{y}'}^{\mathbf{T}}.(\vec{\mathbf{b}} - B.\vec{\mathbf{e}}) = -\infty. \tag{11}$$

Equation (11) 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 :$\{\vec{\mathbf{y}} : \vec{\mathbf{y}} \geq \vec{\mathbf{0}}, G^T.\vec{\mathbf{y}} \geq \vec{\mathbf{0}}\}$ and $\mathbf{E}$. If $z = -\infty$, the input system of constraints does not have a parametric schedule.

6 "Standard Constraints" Model

As discussed in [GPS95, Sch87] the Fourier-Motzkin elimination method suffers from the *curse of dimensionality* i.e. it is an exponential time method in the worst-case. [Sak94] shows that for an important subset of constraints, viz. *Standard Constraints*, the elimination method runs in polynomial time. As described in [Sak94],

Definition 6.1 *A standard constraint involves the start times of at most two tasks J_i and J_k, such that exactly one of s_i or s_j appears on one side of the $\leq$ relation. Further the coefficients of all start and execution variables are unity.*

For example, the following set of constraints are standard:

1. $s_i \leq s_j + 2$,
2. $s_i + e_i \leq s_j$,
3. $s_i + e_i \leq s_j + e_j + 2$

The constraint $s_i + s_j \leq 2$ is not standard, because both s_i and s_j appear on the same side of the relational operator $\leq$. *Absolute* constraints i.e. constraints in which the start time of a task is constrained by an absolute value (e.g. $s_1 \geq 5$) are also permitted and considered standard. In order to make the treatment of constraints (absolute and relative) uniform, we introduce an additional task J_0 with start time s_0 and execution time $e_0 = 0$. Further we impose the constraint $s_0 + e_0 \leq s_1$. Absolute constraints are modified as follows:

- Constraints of the form $s_i \leq a$ are replaced by $s_i - s_0 \leq a$;
- Constraints of the form $s_i \geq a$ are replaced by $s_0 - s_i \leq -a$

For the rest of the discussion, we assume that the matrix G in (11) has been altered to reflect the above changes. Accordingly, G^T has $n+1$ rows 0 through n and $m+1$ columns (m of the initial constraints and the additional constraint between s_0 and s_1.)

6.1 Structure of the Transpose of the Standard Constraint Matrix

When the constraints are standard, the transpose of the constraint matrix (i.e. G^T in (11)) has the following structure:

1. All entries belong to the set $\{0, 1, -1\}$.
2. There are *exactly* 2 non-zero entries in any column; one of the entries is 1 and the other entry is -1.

In this case, we can show that the problem: Is $z = \vec{\mathbf{y}}^{\mathbf{T}}.\vec{\mathbf{b}} = -\infty$, subject to :

$$G^T.\vec{\mathbf{y}} = \vec{\mathbf{g}}, \quad \vec{\mathbf{y}} \geq \vec{\mathbf{0}} \tag{12}$$

where

- G^T is a $(n+1) \times (m+1)$ rational matrix, with the structure discussed above,
- $\vec{\mathbf{y}}$ is a $m+1-$vector,
- $\vec{\mathbf{b}}$ is a rational $m+1-$vector; $b_0 = 0$.
- $\vec{\mathbf{g}}$ is a rational $n+1-$vector.

has a *min-cost flow* interpretation in a constraint network [1] The network $M = < V, R >$ corresponding to the constraint system $G^T.y = g$ is constructed as follows:

[1]These constraints are a subset of *montone constraints*, in which only relations of the form: $a.x_1 - b.x_2 \leq c, a, b > 0$ are allowed.

1. A vertex for v_j for the each row $j, j = 0, \ldots n$ of G^T, giving a total of $n+1$ vertices. Note that v_i corresponds to row G_i i.e. task J_i.
2. Associated with each vertex v_j is a *supply* equal to g_j; Set $g_0 = 0$.
3. An edge r_i for each column of $i, i = 0, \ldots m$ of G^T giving a total of $m+1$ edges. Let v_a denote the vertex corresponding to the $+1$ entry and v_b denote vertex corresponding to the -1 entry. Direct the edge from the vertex v_a to v_b.
4. Associated with edge r_i is *cost* b_i, where b_i is the coefficent of y_i;
5. $y_i(\geq 0), \forall i = 0, \ldots, m$ represents the flow on edge r_i. The flow on the edge $v_0 - v_1$ i.e. y_0 does not contribute to the total cost as the cost on this edge is 0.

The vertex v_0 is the source of this network. Each constraint is now a *mass balance* condition i.e. it states that the net flow into node v_i which is the difference between the total flow into v_i and the total flow out of v_i must equal the supply at v_i. $z = \vec{\mathbf{y}}^T.\vec{\mathbf{b}}$ represents the cost of the flow.

Let us analyze the case where all vertices two special cases, which directly bear upon our scheduling problem:

1. $\vec{\mathbf{g}} = \vec{\mathbf{0}}$ i.e. the supply at all vertices is zero.

   Lemma 6.1 *In this case the condition $z = -\infty$ is possible only iff there is a negative cost loop in the network.*

   Proof 6.1 *Clearly, if there is a negative cost loop, we can pump flow in that loop decreasing the cost arbitrarily, while meeting the mass balance constraints.*

   Now assume that $z = -\infty$. This is possible only if the flow vector $\vec{\mathbf{y}}$ is unbounded in some of its elements. Let us pick some element y_k that is unbounded i.e. $y_k = +\infty$. Thus there is an infinite flow on edge e_k. In order to satisfy the zero supply *requirement at the vertices corresponding to its end-points, e_k must belong to a closed loop and all the edges in that loop have the same flow equal to $+\infty$. Since $z = -\infty$, it follows that the cost around that loop is negative.*

2. $g_1 = R(\geq 0)$; $g_i = 0, i = 2, \ldots n$. In this case the first node has a supply that could be non-zero. This is now a Single-Source Shortest Path Problem with vertex v_1 being the source. Using arguments identical to the case above, it is clear that that $z = -\infty$ coincides with the existence of a negative cost cycle i.e. the shortest path from the source v_0 to any vertex on this cycle is of length $-\infty$.

Our dual system (11) though is in the form $G^T.\vec{\mathbf{y}} \geq \vec{\mathbf{0}}, \vec{\mathbf{y}} \geq \vec{\mathbf{0}}$. Before we apply the flow-related concepts and results derived above, the system needs to be converted into equality form. We use the *Complementary Slackness* property [Sch87, PS82] to aid us in this conversion. Observe that in the primal system, the start time variables are strictly ordered i.e. we have $s_i \leq s_{i+1}, i = 0, \ldots n-1$. We impose $s_1 \geq \epsilon$ to simpleify the analysis. Thus in any solution (including the optimal), we must have $s_i > 0, \forall i = 1, \ldots n$. According to the Complementary Slackness property, *if the primal variable is non-zero at optimality, then the corresponding constraint in the dual must be met with equality.* Thus, all the constraints in the system $G^T.\vec{\mathbf{y}} \geq \vec{\mathbf{0}}$, except the first one, are met with equality, which is exactly what we need. Hence, we can rewrite condition (11) for infeasibility in the primal as:

$$\exists \vec{\mathbf{y}} \in R_+^{m+1} \quad G^T.\vec{\mathbf{y'}} = [R, 0, \ldots, 0]^T \quad \vec{\mathbf{y'}}^{\mathbf{T}}.(\vec{\mathbf{b}} - B.\vec{\mathbf{e}}) = -\infty. \tag{13}$$

Thus our problem is equivalent to the *SSSP* problem as discussed in case (2) of the above analysis. There is one major difference, viz. in our case the edge costs are *symbolic* (e.g. $e_1, e_1 - e_2$, etc.) and not rational numbers. However, if we can find values $e'_i \in \mathbf{E}$ for the e_i variables, then our techniques and results still hold. We now require an algorithm that detects *symbolic* negative cost cycles in the constraint graph corresponding to the given constraint system. We provide such an algorithm in the following section.

7 The Symbolic Bellman-Ford Algorithm

Algorithms 7.1 together with procedures 7.2 and 7.3 represent the Symbolic Bellman-Ford Algorithm. The key modification to the algorithm in [CLR92] is the addition of procedure 7.3. In the case, where the edge weights are rational numbers, it is trivial to check whether $d[v]$ exceeds $d[u] + w(u, v]$.

The input to the algorithm is a graph $G = (V, E)$, with V denoting the vertex set and E denoting the edge set. The weights on the edges are parameterized linear functions in the execution times e_i as discussed above. The function Initialize-Single-Source sets the source s to be at a distance of 0 from itself and all other vertices at a distance of ∞ from the source. A detailed exposition of the Bellman-Ford Algorithm is presented in [CLR92]. Let $\delta[v_0, v_i]$, $d[v_i]$ denote the length of the shortest path from v_0 to vertex v and the current estimate of the shortest path respectively.

Function Symbolic Bellman-Ford(G, w, s)
Initialize-Single-Source
for ($i \leftarrow 1$ **to** $|V(G)| - 1$) **do**
  for (each edge $(u, v) \in E[G]$) **do**
    Symbolic-Relax(u, v, w)
  end for
end for

for (each edge $(u, v) \in E[G]$) **do**
  if ($d[v] >_{sym} d[u] + w(u, v)$) **then**
    return(**false**)
  end if
end for

return(**true**)

Algorithm 7.1: Symbolic Bellman Ford

Procedure Symbolic-Relax(u, v, w)
  if ($d[v] >_{sym} d[u] + w(u, v)$) **then**
    $d[v] = d[u] + w(u, v)$
  end if

Algorithm 7.2: Symbolic-Relax

Function Symbolic $>(u, v, w)$
  if ($\min_E .(d[v] - d[u] - w(u, v)) < 0$) **then**
    return(**true**)
  else
    return(**false**)
  end if

Algorithm 7.3: Implementation of $>_{sym}$

Assuming that all vertices are reachable from the source (a valid assumption in our case), a return value of **true** means that there is a finite shortest path from the source to every other vertex. Likewise, the detection of a negative cost cycle (indicating that certain vertices are at a distance of $-\infty$ from the source), causes the value **false** to be returned. When dealing with rational numbers, all the above operations are relatively straightforward. In our case, the weights on the edges are no longer rational numbers, but parameterized linear forms in the e_i

variables, as indicated above. The algorithm implementing SYMBOLIC $>$ is a *convex minimization* algorithm [PS82, HuL93].

7.1 Analysis - Correctness and Complexity

The correctness of the algorithm follows from the correctness of the Bellman-Ford algorithm [CLR92]. The following two cases arise:

1. There is no point $\vec{\mathbf{e'}} \in \mathbf{E}$ such that substituting $\vec{\mathbf{e'}}$ on the edge costs results in a negative cost cycle.

   Claim 7.1 *Algorithm (7.1) returns* `true`.

   Proof 7.1 *Observe that in the absence of a witness vector $\vec{\mathbf{e'}} \in \mathbf{E}$, the shortest path from v_0 to every vertex is finite. Using the inductive technique from [CLR92], it is clear that after $|V(G)| - 1$ iterations of the* **for** *loop in Step 2 of Algorithm (7.1) the distance of each vertex has converged to its true shortest path from the source. Consequently the test in the succeeding* **for** *loop fails and the value* `true` *is returned.*

2. There exists a point $\vec{\mathbf{e'}} = [e'_1, e'_2, \ldots, e'_n] \in \mathbf{E}$ such that substituting $\vec{\mathbf{e'}}$ on the edge costs results in a negative cost cycle.

   Claim 7.2 *Algorithm (7.1) returns* `false`.

   Proof 7.2 *Once again, we use the same technique as in [CLR92].*

The time taken by the algorithm is dominated by the $O(n^3)$ loop represented by Steps $2 - 6$ of Algorithm 7.1. Each call to SYMBOLIC-RELAX takes time $O(C)$ where C is the time taken by a convex programming algorithm [HuL93]. Accordingly, the total time taken by SYMBOLIC-BELLMAN-FORD is $O(n^3.C)$.

7.2 Example

Let us apply our techniques to the following problem.

We have four tasks $\{J_1, J_2, J_3, J_4\}$ with execution times $\{e_1 \in [4,8], e_2 \in [0,11], e_3 \in [10.13], e_4 \in [3,9]\}$ and start times $\{s_1, s_2, s_3, s_4\}$ constrained through:

- Task J_4 finishes before time 56; $s_4 + e_4 \leq 56$
- Task J_4 finishes within 12 units of J_3;$s_4 + e_4 \leq s_3 + e_3 + 12$
- Task J_4 starts no earlier than 18 units of T_2 completing: $s_2 + e_2 + 18 \leq s_4$
- Task J_3 finishes within 31 units of J_1 completing: $s_3 + e_3 \leq s_1 + e_1 + 31$

Implicit are the ordering constraints:

$$0 \leq s_1, \;\; s_1 + e_1 \leq s_2, \;\; s_2 + e_2 \leq s_3, \;\; s_3 + e_3 \leq s_4$$

Based on (3), the parametric schedulabilty query is:

$$\exists s_1 \forall e_1 \in [4,8] \exists s_2 \forall e_2 \in [6,11] \exists s_3 \forall e_3 \in [10,13] \exists s_4 \forall e_4 \in [3,9] \{(2),(3),(4),(5),(6)\} \tag{14}$$

We construct the graph in Figure (1) as per the discussion in §6.1.

In this case, the convex domain is the axis-parallel hyper-rectangle $\mathbf{E} = [4,8] \times [6,11] \times p10,13] \times [3,9]$. We provide the graph $M =< V, E >$ and $\mathbf{E}$ as the input to Algorithm (7.1). The tables below (1-2) detail the the iterations of the algorithm.

At the end of the 2^{nd} iteration, the shortest path values converge and after applying Steps (8-12) of Algorithm (7.1), we conclude that there is no negative cost loop in the graph.

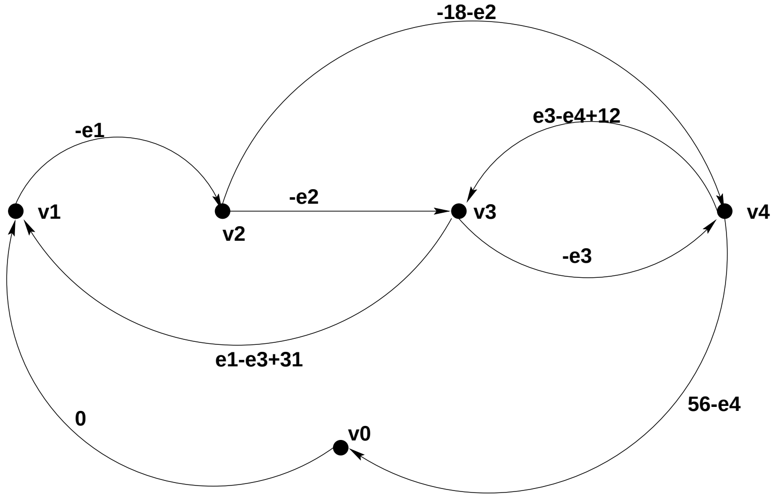

Figure 1: Constraint Graph Corresponding to Example

Edge	Effect
$v_0 - v_1$	$v_1 = 0$
$v_1 - v_2$	$v_2 = -e_1$
$v_2 - v_3$	$v_3 = -e_1 - e_2$
$v_2 - v_4$	$v_4 = -18 - e_2$
$v_3 - v_4$	$v_4 = -e1 - e_2 - e_3$
$v_3 - v_1$	$v_1 = 0$
$v_4 - v_3$	$v_3 = -e_1 - e_2$
$v_4 - v_2$	$v_2 = -18 - 2.e_2 - e_3$
$v_4 - v_0$	$v_0 = 0$

Table 1: Iteration 1

Edge	Effect
$v_0 - v_1$	$v_1 = 0$
$v_1 - v_2$	$v_2 = -18 - 2.e_2 - e_3$
$v_2 - v_3$	$v_3 = -e_1 - e_2$
$v_2 - v_4$	$v_4 = -e1 - e_2 - e_3$
$v_3 - v_4$	$v_4 = -e1 - e_2 - e_3$
$v_3 - v_1$	$v_1 = 0$
$v_4 - v_3$	$v_3 = -e_1 - e_2$
$v_4 - v_2$	$v_2 = -18 - 2.e_2 - e_3$
$v_4 - v_0$	$v_0 = 0$

Table 2: Iteration 2 (Final iteration)

8 Network Constraint Scheduling

In [HN94], a strongly polynomial algorithm is presented for checking feasibility of a linear program when there are at most two variables per constraint i.e. all constraints are of the form :$a.x_i + b.y_j \leq c_k$, where $a, b \in Q$. In other words, it is no longer required that the constraints be *monotone* or *standard* as in §6. In this section, we show that their technique can be extended to the problem of parametric scheduling in a straightforward fashion, thereby enhancing the class of constraints for which we can efficiently determine the existence of a parametric schedule. Consider the complement of query (6)

$$\forall \vec{\mathbf{e}} \in \mathbf{E} \exists s_1, s_2, \ldots, s_n \quad A.[\vec{\mathbf{s}}, \vec{\mathbf{e}}] \leq \mathbf{b} \quad ? \tag{15}$$

Suppose we project the object represented by $A.[\vec{\mathbf{s}}, \vec{\mathbf{e}}] \leq \mathbf{b}$ onto the execution space i.e. the space of the e_i variables to get a new polyhedron $P.\vec{\mathbf{e}} \leq \vec{\mathbf{p}}$. Then the query (15) can be interpreted as the following polytope inclusion problem:

Does the convex body $\mathbf{E}$ lie completely within the polytope represented by $P.\vec{\mathbf{e}} \leq \vec{\mathbf{p}}$?

The input system has a parametric schedule *iff* the above question can be answered affirmatively. So now we are faced with 2 sub-problems:

1. Projection of $A.[\vec{\mathbf{s}}, \vec{\mathbf{e}}] \leq \mathbf{b}$ onto execution space.

2. Determing whether the convex body $\mathbf{E}$ lies completely within the projected object.

A linear constraint $a.x \leq c$ is a half-hyperplane in $n-$space. A convex body ρ is said to lie on the *valid* side of $a.x \leq c$, iff for all points $y \in \rho$ $a.y \leq c$. When there is no confusion, we say that $a.x \leq c$ is valid for **rho**.

Lemma 8.1 *A convex body ρ lies on the valid side of $a.x \leq c$, iff $max_{y \in \rho} a.y \leq c$*

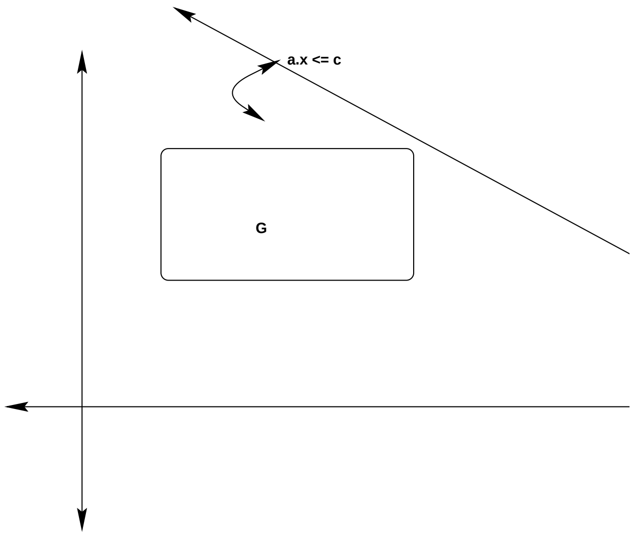

Figure 2: Validity of hyperplanes

Proof 8.1 *This is obvious as indicated by Figure 2. We also note that the validity of $a.x \leq c$ for ρ can be checked in $O(C)$ time, through a convex miniization algorithm.*

Lemma 8.2 *Given a polyhedral system $A.\vec{\mathbf{x}} \leq \vec{\mathbf{b}}$ and a convex body $\mathbf{X}$ in the same space, it is possible to determine whether X lies completely within $A.\vec{\mathbf{x}} \leq \vec{\mathbf{b}}$ in time $O(m.C)$ where m is the number of constraints in $A.\vec{\mathbf{x}} \leq \vec{\mathbf{b}}$*

Proof 8.2 *We first note that $\mathbf{X}$ is contained within $A.\vec{\mathbf{x}} \leq \vec{\mathbf{b}}$ iff each constraint $a_i.\vec{\mathbf{x}} \leq b_i$ is valid for $\mathbf{X}$. Accordingly, we can apply a convex minimization algorithm m times to verify that each constraint is valid and then declare whether or not $\mathbf{X}$ is contained within $A.\vec{\mathbf{x}} \leq \vec{\mathbf{b}}$.*

We are now left with the problem of projecting the polyhedral object (1) onto execution time space to get another polyhedron of the form $P.\vec{\mathbf{e}} \leq \vec{\mathbf{p}}$. We achieve the projection through the Fourier-Motzkin elimination procedure [DE73, Sch87]. In general, the procedure creates exponentially many inequalities [Sch87]; however there are good results for the case when the number of variables per constraint is restricted to just 2 or less. Nelson [Nel78] gave the first sub-exponential algorithm for the restricted case of two variables per constraint. He showed that the total number of inequalities was bounded by $O(m.n^{\log n})$. This result was improved in [HN94], where it is shown that elimination procedure can be implemented efficiently i.e. in time time $O(m.n^2.\log m)$, where m is the number of constraints and n is the number of variables.

We use their implementation to project the object (1) onto execution time space in time $O((m.n^2.\log m).C)$. We then use Lemma (8.2) to verify the existence of a parametric schedule in time $O((m.n^2.\log m).C)$.

9 Concluding Remarks

In this paper, we set out to address two issues in parametric scheduling:

1. Are there polynomial time algorithms for execution times in domains other than axis-parallel hyper-rectangles ?
2. Can we extend the class of constraints on the start times of tasks, for which a parametric schedule can be determined in polynomial time ?

We answered the first question by providing a polynomial algorithm for the case when the execution times belong to arbitrary convex domains. Likewise, the second constraint is answered in the affirmative, by an algorithm that runs in polynomial time when the constraints between the start times of the tasks can be represented by a network.

Our algorithms are simple and easy-to-implement extensions of existing algorithms for network problems. Our work is currently being implemented in the Maruti Operating System [2] [STA00].

It would be interesting to see if the techniques presented in this paper can be extended to a wider class of constraints in real-time scheduling.

References

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

[BS74] K. R. Baker and Z. Su. Sequencing with Due-Date and Early Start Times to Minimize Maximum Tardiness. *Naval Res. Log. Quart.*, 21:171–176, 1974.

[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.

[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.

[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.

[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.

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

[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.

[2] Maruti is the registered trademark of the Maruti Real-Time Operating System, developed at the University of Maryland, College Park; http://www.cs.umd.edu/projects/maruti

[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.

[Nel78] C. G. Nelson. An n log n algorithm for the two-variable-per-constraint linear program satisfiability problem. Technical Report Technical Note STA, Stanford University, Computer Science Department,-78–689, November 1978.

[NW88] G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization*, chapter I.6.4: A projective algorithm for linear programming, pages 164–172. John Wiley & Sons, New York, 1988.

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

[SA00a] K. Subramani and A. K. Agrawala. The parametric polytope and its applications to a scheduling problem. *Manuscript in Preparation*, March 2000.

[SA00b] K. Subramani and A. K. Agrawala. The static polytope and its applications to a scheduling problem. *Manuscript in Preparation*, February 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.

[STA00] K. Subramani, Bao Trinh, and A. K. Agrawala. Implementation of static and parametric schedulers in maruti. *Manuscript in Preparation*, March 2000.

[TSYT97] M. Tayara, Nandit Soparkar, John Yook, and Dawn Tilbury. Real-time data and co-ordination control for reconfigurable manufacturing systems. In Azer Bestavros and Victor Fay-Wolfe, editors, *Real-Time Database and Information Systems, Research Advances*, pages 23–48. Kluwer Academic Publishers, 1997.

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