The Static Polytope and its applications to a Scheduling Problem

K. Subramani * Ashok Agrawala †

Abstract

In the design of real-time systems, it is often the case that certain process parameters (such as *execution time*) are not known precisely. The challenge in real-time system design is to develop techniques that efficiently meet the requirements of impreciseness. Traditional models tend to simplify the issue of impreciseness by assuming *worst-case* times. This assumption is unrealistic and at the same time, may cause certain constraints to be violated at run-time. In this paper, we shall study the problem of scheduling a set of ordered, non-preemptive processes under non-constant execution times. Typical applications for variable execution time scheduling include process scheduling in Real-time Operating Systems such as Maruti, compiler scheduling, database transaction scheduling and automated machine control. An important feature of application areas such as robotics is the interaction between execution times of various processes. We explicitly model this interaction through the representation of execution time vectors as points in convex sets. We present both sequential and parallel algorithms for determining the existence of a static schedule.

1 Introduction

Scheduling in real-time systems has received considerable attention in system design research [Sak94, SB94, HG93, Gel76]. In real-time system design, the challenge is two-fold: (a) Modeling the underlying system (a non-trivial task if we wish to preserve properties of interest), and (b) Proposing solutions for problems arising in the chosen model. In this paper, we are concerned with the following problems:

1. Modeling constraint systems of tasks with relationships between their execution times. To the best of our knowledge, this represents the first attempt to explicitly include the inter-execution time dependency in a scheduling model.

2. Scheduling an ordered set of non-preemptive tasks, subject to a set of linear constraints, with non-constant execution times.

We approach the modeling and the scheduling problems from a Convex Programming perspective [HuL93] and we techniques from *Convex Analysis* [Roc70] almost exclusively to prove our results. We present both sequential and parallel algorithms for determining the existence of a feasible schedule.

In section §2 we present the static scheduling model and pose the static schedulability query. Interprocess execution time relationships are explicitly modeled through convex domains. The succeeding section, §3 discuss the motivation for our work as well as related approaches to this problem. In particular, we show that scheduling concerns in diverse areas such as robotics and real-time operating systems can be addressed through our model.

§4 commences the process of answering the static scheduling query posed in §2 through the application of *convex minimization* algorithms. We present our algorithm for the case when the execution times belong to general convex domains and analyze its correctness and complexity. A straightforward parallelization of the algorithm is provided as part of the complexity analysis. §5 specializes the algorithm in §4 to the case in which the execution time domain is an axis-parallel hyper-rectangle. We conclude in §6 with a summary of our results and some open problems in this area.

*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 Static 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,
- $\vec{\mathbf{e}} \in \mathbf{E} = [e_1, e_2, \ldots, e_n]$ is an $n-$vector of the execution time of the tasks, $\mathbf{E}$ is a convex domain.
- 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.

Note that we can also use the finish times f_i of tasks in relationships. Since the tasks are non-preemptive, the relation: $s_i + e_i = f_i$ holds for all tasks J_i and hence our expressiveness is not enhanced by the inclusion.

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 the 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 $\mathbf{E}$.

Before we develop the scheduling query, a few definition are in order.

Definition 2.1 *Static Schedule - A schedule for a set of tasks specified by assigning rational numbers to their start times.*

Definition 2.2 *Dispatch Calendar - A data structure indicating the time of dispatch for each task in the schedule.*

In static scheduling, we are interested in a rational vector $\vec{\mathbf{s}}$ that holds for all execution times vectors belonging to the set $\mathbf{E}$.

The following predicate captures our model:

$$\exists \vec{\mathbf{s}} = [s_1, s_2 \ldots s_n] \forall \vec{\mathbf{e}} = e_1, e_2, \ldots e_n \in E \qquad A.[s, e] \leq b \ ? \tag{2}$$

We note that the tasks are ordered and this ordering on the tasks is obtained by imposing the constraints:

$$s_i + e_i \leq s_{i+1}, \forall i = 1, \ldots, n-1.$$

The ordering constraints are included in the A matrix in (2).

Predicate (2) will henceforth be referred to as the *static scheduling query.*

Our model is distinct from traditional models in the following ways:

1. *Non-constant execution times*- These account for more realistic abstractions. In some cases, the interaction between execution times is explicitly given through a convex set (refer §3); in other cases a number of runs of the task sets allow us to determine lower and upper bounds of tasks (refer §5);
2. Generalized linear constraints - Typical models are concerned with sequencing in the presence of ready-times and deadlines [GJ79, Cof76]; our models attempt to capture broader settings where linear relationships constrain task execution.

3 Motivation and Related Work

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

One of the fundamental aspects of real-time scheduling is the recognition of interdependencies between tasks [DMP91, Sak94] and the conversion of event-based specifications into temporal distance constraints between tasks [Das85, JM86]. For example, the event-based requirement *Wait 50 ms before sending the next message* would spawn the following temporal distance constraint: $s_i + 50 \leq s_{i+1}$, where s_i and s_{i+1} denote the start times of successive invocations of the message generating task.

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.

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 is to constrain the sum of the velocities of the axes. 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]. In database transactions, we have an ordered set of processes that interact under certain conditions e.g. *When the account balance exceeds* $\$7k$*, change interest rate to* 8%.

Consider a real-time system used for flight control in the aviation industry. Typically, there is a process (task) in charge of controlling the altitude and another process in charge of controlling the speed of the airplane. Suppose that it is determined that the speed of the plane should exceed $300mph$, when the altitude exceeds $50ft$. Such a condition can be captured through a constraint on the execution of these two processes; i.e. *If altitude exceeds* 50*, increase speed to* 300.

Deterministic sequencing (the problem of determining a feasible sequence) and scheduling have a long history of research [BS74, DL78, Cof76]. Mere ready-time and deadline constraints make the sequencing problem NP-complete [GJ79]. Thus the problem of scheduling under general relative constraints (called *generalized scheduling*), in the absence of ordering information between the tasks is clearly NP-complete, as it subsumes the sequencing problem. In [Sak94] it is shown that the problem of generalized scheduling is also NP-complete for the preemptive case, which is surprising since [HL89b] gives a polynomial time algorithm for the problem of `deterministic sequencing`, when preemption is allowed.

In this paper, we focus on the problem of scheduling a set of tasks, when the ordering sequence is known (and supplied as part of the input), but there exist complex inter-task dependencies, captured through linear relationships between their start and execution times. Although we restrict ourselves to addressing the feasibility of the task system, the judicious use of objective functions can be used to improve the quality of our solution. The determination of a feasible schedule coincides with the generation of a *static dispatch-calendar* that contains the dispatching information for each task: e.g. $s_1 = 2; s_2 = 15; s_3 = 24$, is a *dispatch-calendar* for a 3-task system.

Variations of the problem that we are studying have been dealt with in [Sak94], [HL89a], [HL92b] and [HL92a]. This problem is briefly mentioned in [Sak94] as part of a different problem i.e. *parametric scheduling*, however no algorithm is presented for the general case. The strategy suggested there is *variable elimination*, which works for restricted domains only. In [HL89a, HL92b], the problem of scheduling real-time tasks under distance and separation constraints is considered, but the execution times are regarded as constant.

4 Algorithm for convex domains of execution times

We first establish that using worst-case values for execution times will not provide a valid solution in general. Consider the following constraint system imposed on a 3 task set:$\{J_1, J_2, J_3\}$.

1. J_1 finishes before J_2 commences: $s_1 + e_1 \leq s_2$
2. J_2 commences within 2 unit of J_1 finishing: $s_2 \leq s_1 + e_1 + 2$
3. $e_1 \in [3, 9]$.

Substituting the worst-case time for e_1 i.e. 9 in the constraints, we obtain: $s_1 + 9 \leq s_2$, $s_2 \leq s_1 + 11$. One possible solution to this system is $[s_1, s_2] = [0, 9]$. However, during actual execution, suppose e_1 takes on the value 3, then $s_2 > s_1 + e_1 + 2$, thereby violating the second constraint.

We can interpret the static scheduling query (2) as asking whether there exists a set of start times in *pure number form*, without any dependencies i.e. the only accepted solutions are of the form:$s_i = a_i$, where the $a_i \in Q$ are rational numbers. The static approach is to work individually with each constraint and find the execution times that make the constraint tight (or *binding*). We then argue in §4.2 that the strategy is correct inasmuch as the goal is to produce a single start time vector $\vec{\mathbf{s}}$ that holds for all execution time vectors $\vec{\mathbf{e}} \in E$.

Function STATIC SCHEDULER (`Arbitrary Convex Domains`) $(\mathbf{E}, \mathbf{A}, \vec{\mathbf{b}})$

1: {E is the convex domain and $A[\vec{\mathbf{s}}, \vec{\mathbf{e}}] \leq \vec{\mathbf{b}}$ is the constraint system}
2: Using the techniques discussed in Appendix §C,

   1. Rewrite the constraint matrix as: $G.\vec{s} \leq \vec{\mathbf{b}} - \mathbf{B}.\vec{e}$.
   2. Set $\vec{\mathbf{r}} = [r_1, r_2, \ldots, r_m]^T = \vec{\mathbf{b}} - \mathbf{B}.\vec{e}$ { each r_i is an affine function of $\vec{\mathbf{e}} = [e_1, e_2, \ldots, e_n]$}

3: **for** (i=1 **to** m) **do**
4: Let $\rho_i = \min_S r_i$ { ρ_i is a rational number}
5: **end for**
6: **if** ($\vec{s} : G.\vec{s} \leq \vec{\rho} \neq \phi$) **then**
7: **return**(`System is feasible`) { $G.\vec{s} \leq \vec{\rho}$ is the **Static Polytope**}
8: **else**
9: **return**(`System has no static schedule`)
10: **end if**

Algorithm 4.1: Static Scheduling Algorithm

4.1 Example

Before proceeding with proving the correctness of the STATIC SCHEDULER algorithm, we present an example to demonstrate our approach. Consider the two task set $\tau = \{\tau_1, \tau_2\}$, with execution times $\{e_1, e_2\}$, constrained through the following convex domain:

- $e_1 \in [0, 6], e_2 \in [0, 6]$
- $e_1 + e_2 \leq 4$

Figure 1 describes the domain.

Let the system have the following constraints:

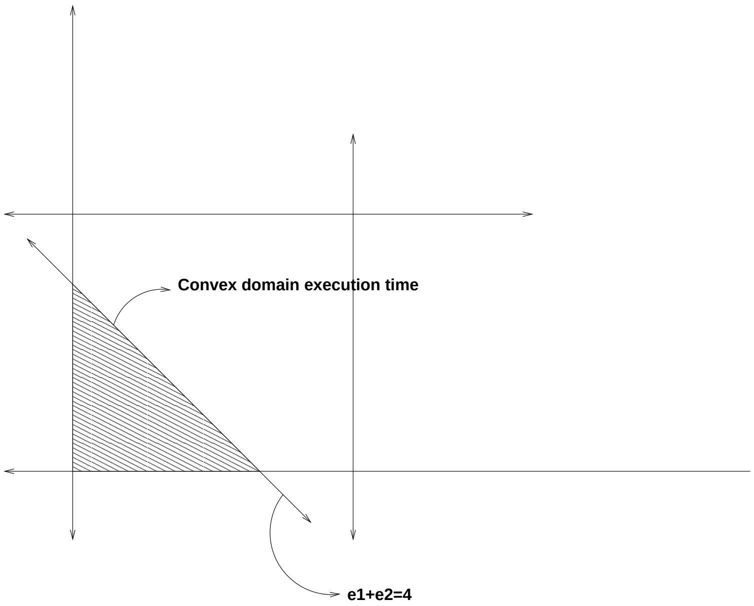

Figure 1: Convex constrained execution times

1. τ_1 finishes execution at or before task τ_2 commences: $s_1 + e_1 \leq s_2$.
2. τ_2 finishes at or before 12 units. $s_2 + e_2 \leq 12$.

Expressing the constraints in matrix form, we get :

$$\begin{bmatrix} 1 & -1 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} s_1 \\ s_2 \\ e_1 \\ e_2 \end{bmatrix} \leq \begin{bmatrix} 0 \\ 12 \end{bmatrix}$$

We first rewrite this system to separate the s and the e vectors:

$$\begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \end{bmatrix} \leq \begin{bmatrix} 0 \\ 12 \end{bmatrix}$$

Moving the $e-$variables to the RHS, we get

$$\begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} \leq \begin{bmatrix} 0 \\ 12 \end{bmatrix} - \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \end{bmatrix}$$

which is equivalent to:

$$\begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} \leq \begin{bmatrix} -e_1 \\ 12 - e_2 \end{bmatrix}$$

Minimizing $-e_1$ over the constraint domain in Figure 1, we get $\rho_1 = -4$. Likewise, minimizing $12 - e_2$ over the constraint domain, we get $\rho_2 = 8$. Thus, the static polytope is determined by:

$$\begin{pmatrix} 1 & -1 \\ 0 & 1 \end{pmatrix} \cdot \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} \leq \begin{bmatrix} -4 \\ 8 \end{bmatrix}$$

as shown in Figure 2.

Applying a linear programming solver like [Ber95] to the above system, we get the following solution:

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

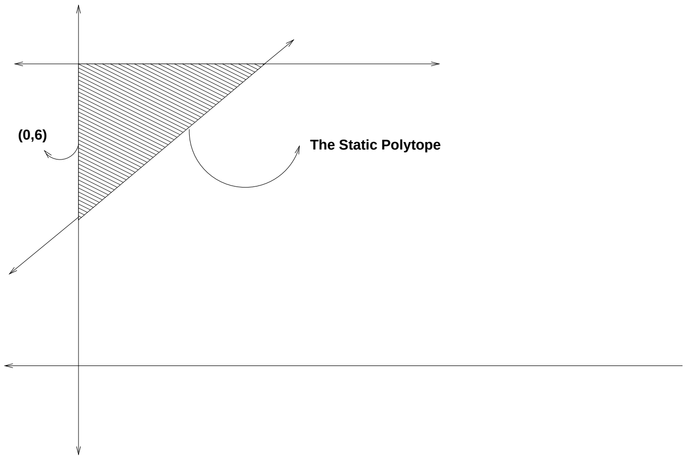

Figure 2: The Static Polytope

4.2 Correctness

Lemma 4.1 *If the final polytope* [1] *in algorithm (4.1) is non-empty, then any point on it serves as a suitable vector of start times* $[s_1, s_2, \ldots, s_n]$.

We use a_i to denote the i^{th} row of $\mathbf{A}$ and g_i to denote the i^{th} row of G.

Proof 4.1 *Given a point* $p = [p_1, p_2, \ldots, p_n]$ *of the final non-empty static polytope* $G.\vec{\mathbf{s}} \leq \vec{\mathbf{m}}$, *let us assume the contrary and that indeed one or more of the input constraints of the input constraint matrix has been violated at a particular execution time vector* $\vec{\mathbf{e}'} = [e'_1, e'_2, \ldots, e'_n] \in \mathbf{E}$. *Pick one such violated constraint, say* $\vec{\mathbf{a_i}}.[\vec{\mathbf{s}}, \vec{\mathbf{e}} \leq b_i$. *The violation of this constraint at* $(\vec{\mathbf{p}}, \vec{\mathbf{e}'}$ *implies that* $\vec{\mathbf{a_i}}.[\vec{\mathbf{p}}, \vec{\mathbf{e}'}] > b_i$ *or after rewriting and separating the variables* $\vec{\mathbf{g_i}}.\vec{\mathbf{p}} > (\vec{\mathbf{b}} - \mathbf{B}.\vec{\mathbf{e}})_i$ *But from the construction of the* static polytope *(refer Algorithm (4.1)), we know that*

$$\rho_i = min_E(\vec{\mathbf{b}} - \mathbf{B}.\vec{\mathbf{e}})_i$$

and

$$\vec{\mathbf{g_i}}.\vec{\mathbf{p}} \leq \rho_i$$

which provides the desired constradiction.

Hence no constraint can be violated by choosing a point on the Static polytope.

4.3 Complexity

We observe that the elimination of each vector r_i and its replacement by a rational number ρ_i involves a call to a Convex minimization Algorithm. m such calls are required to eliminate all the r_i from the constraint system giving a time of $O(m.C)$, where C is the running time of the fastest Convex minimization algorithm (refer Appendix B for details). Finally, one call has to be made to a linear programming algorithm to verify the feasibility of the resultant static polytope. Thus, the total running time of the algorithm is $O(m.C + L)$, where L is the running time of the fastest linear programming algorithm. (Refer Appendix A for details.) Finally, since linear programming is a special case of convex minimization, we have $L \leq C$ and hence the complexity of our algorithm is $O(m.C)$.

[1] The resultant polyhedron will always be bounded because the jobs are ordered i.e. $s_1 < s_2 < \ldots s_n$ and the last job has a deadline i.e. $s_n + e_n \leq D$

4.4 Parallelization of the Static Scheduler

Observe that the ρ_i are created independently for each constraint. Thus the steps involving determination of the ρ_i can indeed be carried out in parallel. That suggests the following parallel[2] algorithm implementation of the **for** loop in Algorithm (4.1): This algorithm has a parallel running time of $O(C)$ with a total work of $O(m.C)$.

Function PARALLEL STATIC SCHEDULER(`Arbitrary Convex Domains`) $(\mathbf{E}, \mathbf{A}, \vec{\mathbf{b}})$

1: Carry out the initialization steps as in Sequential Algorithm
2: **for** ($i = 1$ **to** m **pardo**) **do**
3: Let $\rho_i = \min_S r_i$
4: **end for**
5: Perform feasibility check as in Sequential Algorithm

Algorithm 4.2: Parallel version of Static Scheduling Algorithm

5 Specialization to axis-parallel hyper-rectangle

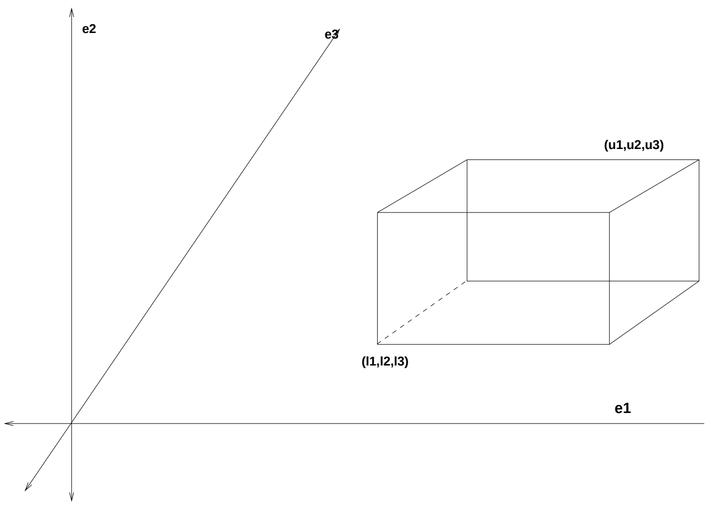

Figure 3: An axis-parallel hyper-rectangle

We now consider an interesting domain of the execution time vectors viz. the class of axis-parallel hyper-rectangles. 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 §2 is now the axis-parallel hyper-rectangle represented by: $R = [l_1, u_1] \times [l_2, u_2] \times \ldots \times [l_n, u_n]$. The static scheduling query (2) for this case is:

$$\exists \vec{s} = [s_1, s_2 \ldots s_n] \forall \vec{e} = [e_1, e_2, \ldots e_n] \in R \qquad A.[s, e] \leq b \ ? \tag{3}$$

We can apply the same algorithm as in section §4, in which case we solve $(m + 1)$ linear programs to give a total running time of $O(m.L)$, where L is the running time of the fastest linear programming algorithm.

However, we can do much better as we shall show shortly.

[2] We are using the PRAM Model, see [Ja'92]

Lemma 5.1 *The minimum of an affine function on an axis-parallel hyper-rectangle (aph) is reached at a vertex of the* `aph`.

Proof 5.1 *From [Sch87], the lemma is true over all polyhedral domains and axis-parallel hyper-rectangles are restricted polyhedral domains.*

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` $R = [l_1, u_1] \times [l_2, u_2] \times \ldots \times [l_n, u_n]$:

- $\forall i$, 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 the STATIC SCHEDULER algorithm runs in $O(mn+L)$ sequential time, where m is the number of constraints.

6 Conclusions and Future Research

The chief contributions of this paper have been:

1. A model to capture the interdependency of process execution times.
2. A systematic procedure to handle transaction scheduling when the execution times are constrained through convex domains.

We also presented a fast algorithm when the domain was an axis-parallel hyper rectangle. The latter case is being currently implemented in Maruti and will be available in [STA00]. Finally, the problem of static scheduling in the absence of ordering information can be decomposed as follows:

- Eliminate the execution time variables from the constraint system, using the algorithm in §4, which results in a generalized scheduling problem (refer §3).
- Use the existing technique (exact or approximate) to solve the resulting generalized scheduling problem

An open area of research is to extend the applicability of our techniques to problems in other domains.

7 Acknowledgements

We thank David Mount for helpful discussions.

A Linear Programming - Complexity Issues

Consider the following linear program in standard form:

$$\begin{aligned} &\max \vec{\mathbf{c}}.\vec{\mathbf{x}},\ s.t. \\ &A.\vec{\mathbf{x}} = \vec{\mathbf{b}}, \vec{\mathbf{x}} \geq 0 \end{aligned} \tag{4}$$

where $\vec{\mathbf{c}}$ is a $n-$vector, $\vec{\mathbf{b}}$ is a $m-$vector, A is $m \times n$ rational matrix and $\vec{\mathbf{x}}$ is a $n-$vector.

The first algorithm for this problem was proposed in [Dan63]. This algorithm has exponential time worst-case complexity [PS82, KM72]. Since then a number of polynomial time algorithms have been developed for this problem [Kha79, Vai87, Kar84].

B Convex Programming - Complexity Issues

Consider the standard nonlinear program, expressed as:

$$\min f(x), s.t.$$
$$g_i(x) \leq 0, \forall i = 1, \dots m \tag{5}$$
$$\tag{6}$$

where the $g_i(x)$ are convex functions and the constraint set formed by their intersection is a convex space in the $n-$dimensional Euclidean space R^n. If f is a convex funtion [HuL93, Roc70], then the problem is called a *convex minimization* or *convex programming* problem.

This problem is known to be solvable in polynomial time [HuL93, PS82]. A fast algorithm for this problem is provided in [KV86].

C Matrix Reorganization

In this section, we focus on manipulating a system of linear inequalities, through the reorganization of its representative matrix.

Consider a linear system of inequalities in two variables $\vec{\mathbf{x}}$ and $\vec{\mathbf{y}}$:$A.[\vec{\mathbf{x}}, \vec{\mathbf{y}}] \leq \vec{\mathbf{b}}$, where,

1. $\vec{\mathbf{x}} = [x_1, x_2, \dots, x_n]^T$,
2. $\vec{\mathbf{y}} = [y_1, y_2, \dots, y_n]^T$,
3. A is a $m \times 2.n$ matrix of rational numbers, and
4. $\vec{\mathbf{b}} = [b_1, b_2, \dots, b_m]^T$ is a rational vector.

Observe that this system can be rewritten in the form:

$$G.\vec{\mathbf{x}} + H.\vec{\mathbf{y}} \leq \vec{\mathbf{b}},$$

$$\Rightarrow G.\vec{\mathbf{x}} \leq \vec{\mathbf{b}} - H.\vec{\mathbf{y}},$$

separating the $\vec{\mathbf{x}}$ and $\vec{\mathbf{y}}$ systems, where G and H are rational $m \times n$ matrices. Further, we can set $\mathbf{vecg} = \vec{\mathbf{b}} - H.\vec{\mathbf{y}}$.

The purpose of this reorganization is to show that a linear system in two variables $\vec{\mathbf{x}}$ and $\vec{\mathbf{y}}$ can be written in the form: $G.\vec{\mathbf{x}} \leq \vec{\mathbf{g}}$, where, g_i as defined above is an affine function of $\vec{\mathbf{y}}$.

References

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

[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.

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

[Dan63] G. B. Dantzig. *Linear Programming and Extensions.* Princeton University Press, Princeton, NJ, 1963.

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

[DMP91] R. Dechter, I. Meiri, and J. Pearl. Temporal constraint networks. *Artificial Intelligence*, 49:61–95, 1991.

[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.

[Gel76] E. Gelenbe, editor. *Modelling and Performance Evaluation of Computer Systems*, chapter Scheduling Dependent Tasks with Different Arrival Times to Meet Deadlines, pages 57–65. North-Holland Publishing Company, 1976.

[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.

[HG93] S. Hong and R. Gerber. Compiling real-time programs into schedulable code. In *Proceedings of the ACM SIGPLAN '93 Conference on Programming Language Design and Implementation.* ACM Press, June 1993. *SIGPLAN Notices, 28(6):166-176.*

[HL89a] C. C. Han and K. J. Lin. Job scheduling with temporal distance constraints. Technical Report UIUCDCS-R-89-1560, University of Illinois at Urbana-Champaign, Department of Computer Science, 1989.

[HL89b] K. S. Hong and J. Y-T. Leung. Preemptive Scheduling with Release Times and Deadlines. *Journal of Real-Time Systems*, 1:265–281, 1989.

[HL92a] C. C. Han and K. J. Lin. Scheduling Distance-Constrained Real-Time Tasks. In *Proceedings, IEEE Real-time Systems Symposium*, pages 300–308, Phoenix, Arizona, December 1992.

[HL92b] C. C. Han and K. J. Lin. Scheduling real-time computations with separation constraints. *Information Processing Letters*, 12:61–66, May 1992.

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

[Ja'92] Ja'Ja'. An introduction to parallel algorithms (contents). *SIGACTN: SIGACT News (ACM Special Interest Group on Automata and Computability Theory)*, 23, 1992.

[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.

[Kar84] N. K. Karmarkar. A new polynomial–time algorithm for linear programming. *Proceedings of the 16th Annual ACM Symposium on Theory of Computing*, pages 302–311, 1984.

[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.

[KM72] V. KLEE and G. J. MINTY. How good is the simplex algorithm. In O. Shisha, editor, *Inequalities - III.* Academic Press Inc., New York and London, 1972.

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

[KV86] S. Kapoor and P. M. Vaidya. Fast algorithms for convex quadratic programming and multicommodity flows. *Proceedings of the 18th Annual ACM Symposium on Theory of Computing*, pages 147–159, 1986.

[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.

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

[Roc70] R. Tyrrell Rockafellar. *Convex Analysis*, volume 28 of *Princeton Mathematics Series.* Princeton University Press, Princeton, 1970.

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

[SB94] A. Stoyenko and T. P. Baker. Real-Time Schedulability Analyzable Mechanisms in Ada9X. *Proceeding of the IEEE*, 82(1):95–106, January 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.

[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.