# Modeling Clairvoyance and Constraints in Real-Time Scheduling

K. Subramani

Department of Computer Science and Electrical Engineering,
West Virginia University,
Morgantown, WV, USA
{ksmani@csee.wvu.edu}

**Abstract.** Scheduling in Real-time systems differs from scheduling in conventional models in two principal ways: (a) Parameter variability, (b) Existence of complex constraints between jobs. Our work focusses on variable execution times. Whereas traditional models assume fixed values for job execution time, we model execution times of jobs through convex sets. The second feature unique to real-time systems, is the presence of temporal relationships that constrain job execution. Consider for instance the requirement that job 1 should conclude 10 units before job 2. This can be modeled through a simple, linear relationship, between the start and execution times of jobs 1 and 2. In real-time scheduling, it is important to guarantee a priori, the scheduling feasibility of the system. Depending upon the nature of the application involved, there are different schedulability specifications viz. Static, Co-Static and Parametric. Each specification comes with its own set of flexibility issues. In this paper, we present a framework that enables the specification of real-time scheduling problems and discuss the relationship between flexibility and complexity in the proposed model. We motivate each aspect of our model through examples from real-world applications.

## 1 Introduction

In this paper, we describe the features of our real-time scheduling framework called the E-T-C ( Execution-Time-Constraints ) Real-Time Scheduling model. Real-time scheduling differs from traditional scheduling in two fundamental ways, viz. non-constant execution times and the existence of complex constraints (such as relative timing constraints) between the constituent jobs of the underlying system. A traditional scheduling model such as the one discussed in [14] and [3] assumes that the execution time of a job is a fixed constant. This assumption is not borne out in practice; for instance the running time of an input dependent loop structure such as **for(** $i = 1$ **to** $N$ **)** will depend upon the value of $N$. Secondly, jobs in a real-time system are often constrained by complex relationships such as: Start job $J_a$ within $5$ units of Job $J_b$ completing. Traditional scheduling literature does not accommodate constraints more complex than those that can be represented by precedence graphs.

Our scheduling model is composed of 3 sub-models, viz. the Job model, the Constraint model and the Query model. The Job model describes the type of jobs that we are interested in scheduling. The Constraint model is concerned with the nature of relationships constraining the execution of the jobs. The Query model specifies what it means

for a set of jobs to be schedulable, subject to constraints imposed as per the Constraint model. *An instance of a problem in the* E-T-C *model is specified by instantiating the variables in the sub-models.*

We focus on the following issues:

(a) Designing a framework that enables specification of real-time scheduling problems, and
(b) Studying instantiations of interest in this framework.

The rest of this paper is organized as follows: Section §2 describes the Job model within the E-T-C scheduling framework. The Constraint model is discussed in the succeeding section viz. Section §3. Section §4 details the Query model and presents the 3 types of queries that we consider in this thesis. Each aspect of the E-T-C scheduling framework is motivated through an example from real-time design. A classification scheme for Scheduling problems in the E-T-C model is introduced in §5.

## 2 Job Model in E-T-C

Assume an infinitely extending time axis, starting at time $t = 0$. This axis is divided into intervals of length $L$; these intervals are ordered and each interval is called a scheduling window e.g. $[0, L]$ represents the first scheduling window, $[L, 2.L]$ represents the second scheduling window and in general, $[(i-1).L, i.L]$ represents the $i^{th}$ scheduling window. We are given a set of *ordered, non-preemptive,* jobs $\mathcal{J} = \{J_1, J_2, \ldots J_n\}$, with start times $\{s_1, s_2, \ldots, s_n\}$ and execution times $\{e_1, e_2, \ldots, e_n\}$. $L$ is the period of the job-set and all jobs execute periodically in each scheduling window. We remark that non-preemptive jobs form the bulk of real-time applications in mission-critical tasks [11].

## 3 Constraint Model in E-T-C

The executions of the jobs in the job-set $\mathcal{J}$ ( discussed in the above section ) are constrained through relationships that exist between their start times and execution times. In the E-T-C model, we permit only linear relationships; thus the constraint system on the job-set is expressed in matrix form as :

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

where,

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

Observe that System (1) can be rewritten in the form:

$$\mathbf{G}.\vec{\mathbf{s}} + \mathbf{H}.\vec{\mathbf{e}} \leq \vec{\mathbf{b}}, \tag{2}$$

where,

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

*We can also use the finish times $f_i$ of jobs in relationships. Since the jobs are non-preemptive, the relation: $s_i + e_i = f_i$ holds for all jobs $J_i$ 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 $\{s_1, s_2, \ldots, s_n\}$ and the execution time axes $\{e_1, e_2, \ldots, e_n\}$.

The execution times are independent of the start times of the jobs; however they may have complex interdependencies among themselves. This interdependence is expressed by setting

$$\vec{\mathbf{e}} \in \mathbf{E} \tag{3}$$

where $\mathbf{E}$ is an arbitrary convex set. We regard the execution times as $n-$vectors belonging to the set $\mathbf{E}$.

*The ordering on the jobs 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 $\mathbf{A}$ matrix in (1).*

The Constraint model can be adapted to special situations by restricting either $\mathbf{E}$ or $\mathbf{A}$ or both. The following advantages result from such restrictions:

- A model that more accurately describes the requirements of the current situation,
- Faster algorithms for schedulability queries, and
- More efficient dispatching schemes.

In §3.1, §3.2 and §3.3 we discuss restrictions to the convex set $\mathbf{E}$, while §3.4, §3.5 and §3.6 deal with restrictions to the constraint matrix $\mathbf{A}$.

### 3.1 The Axis-parallel Hyper-rectangle domain

As specified above, the set $\mathbf{E}$ in the Constraint model can be an arbitrary convex domain. One domain that finds wide applicability is the axis-parallel hyper-rectangle domain ( henceforth abbreviated as `aph` ). The Maruti Operating System [8–10] estimates running times of jobs by performing repeated *runs* so as to determine upper and lower bounds on their execution time. Accordingly, the running time of job $J_i$, viz. $e_i$, belongs to the interval $[l_i, u_i]$, where $l_i$ and $u_i$ denote the lower and upper bounds on the execution time as determined by empirical observation. `These independent range variations are the only constraints on the execution times.` Observe that during actual execution, $e_i$ can take any value in the range $[l_i, u_i]$.

The `aph` domain possesses two useful features:

- A specification that is tractable for this domain is also tractable for arbitrary convex domains [23],
- A specification that is provably "hard" for arbitrary convex domains is also "hard" for this domain [18].

Thus when proving complexity results ( especially *hardness results* ), it suffices to focus on the `aph` domain only.

### 3.2 The Polyhedral domain

A feature of machining systems such as the ones discussed in [25] and [7] is the active interdependence of execution times on each other. For instance, the requirement that the sum of the speeds of two axes $J_1$ and $J_2$ not exceed $k$ is captured by: $e_1 + e_2 \leq a$. Polyhedral domains are generalizations of the aph domains discussed above.

### 3.3 Arbitrary Convex Sets

Even polyhedral domains cannot capture the requirements of Power Systems in which there exists quadratic constraints on the execution times. For instance, the spherical constraint $e_1^2 + e_2^2 + \ldots e_n^2 \leq r, r \geq 0$ captures the requirement that the total power spent in the system is bounded by $r$ [2] .

### 3.4 Standard Constraints

The class of "standard constraints" was introduced in [16], as a restriction to the constraint matrix $\mathbf{A}$ for which the Parametric Schedulability query ( see Section §4.3 ) could be decided efficiently.

**Definition 1.** *A constraint is said to be a standard constraint, if it can be expressed as a strict difference relationship between at most two jobs. The relationship could be expressed between their start or finish times.*

These constraints are also known as *monotone constraints* in the literature [6]. Standard constraints serve to model relative positioning requirements between two jobs and absolute constraints on a single job. When the constraints are standard, the matrix $\mathbf{G}$ in System (2) is network unimodular [5, 13] and hence the constraint system can be represented as a network graph [21, 4].

The advantage of the network representation is that certain feasibility queries in the primal system can be expressed as shortest-path queries in the corresponding dual network [4]. Standard constraints are widely used to model temporal relationships in flight-control systems [11, 12].

### 3.5 Network Constraints

Network constraints are a straightforward generalization of standard constraints.

**Definition 2.** *A constraint is said to be a network constraint, if it can be put in the following form:*

$$a.s_i + b.s_j \leq c.e_i + d.e_j + k, \tag{4}$$

*where* $a, b, c, d, k \in \Re$.

Network constraints can also be represented as graphs [6, 1]; however the relationships between adjacent vertices form a polyhedron and are not adequately represented through edges, as in the case of standard constraints. Once again, the advantage of the graph representation is the existence of faster algorithms for feasibility checking as opposed to general constraints. Network constraints find wide applicability in approximating certain measures [19].

### 3.6 Arbitrary Constraints

Job completion statistics such as *Sum of Completion times* and *Weighted Sum of Completion times* of jobs are of interest to the designers of real-time systems [25]. These statistics are aggregate constraints $\sum_{i=1}^{n}(s_i + e_i)$ and cannot be captured through either standard or network constraints.

## 4 Query model in E-T-C

Goal: *We wish to determine a start time vector $\vec{\mathbf{s}}$, in each scheduling window, such that the constraint system (1) holds ( is not violated )* at run-time *for any execution time vector $\vec{\mathbf{e}} \in \mathbf{E}$.*

The above specification ( called the *schedulability specification* ) is rather vague and is intended to be so; in this section, we shall present three different formalizations of the informal specification above. Each formalization ( specification ) has a different notion of what it means for a job-set to be schedulable and is characterized by a distinct set of complexity issues and flexibility concerns. *However, in all the specifications the guarantees provided are absolute i.e. if the schedulability query is decided affirmatively, then the constraint set will not be violated at run time.* We also use the terms *schedulability query* and *schedulability predicate* to refer to the schedulability specification.

### 4.1 Static Scheduling

Static scheduling ( also called *Scheduling with no Clairvoyance* ) is concerned with deciding the following predicate:

$$\mathcal{P}_s \equiv \exists \vec{\mathbf{s}} = [s_1, s_2, \ldots, s_n] \forall \vec{\mathbf{e}} = [e_1, e_2, \ldots, e_n] \in \mathbf{E} \quad \mathbf{A}.[\vec{\mathbf{s}}, \vec{\mathbf{e}}] \leq \vec{\mathbf{b}} \quad ? \tag{5}$$

In other words, the goal is to determine the existence of a single start-time vector $\vec{\mathbf{s}} \in \Re^n$, such that the constraint system represented by (1) holds. The only information that is available prior to the dispatching of jobs in the $i^{th}$ scheduling window is the knowledge of the execution time domain $\mathbf{E}$.

In [23], we showed that the above proposition can be decided efficiently for arbitrary convex domains. From a computational perspective, query (5) is the easiest to answer. Static Scheduling is the only mode of scheduling at one's disposal, if the dispatcher does not have the power to perform online computations; in fact $O(1)$ dispatching time is one of the advantages of static scheduling [24].

### 4.2 Co-Static Scheduling

Static scheduling is unduly restrictive in that even simple constraint sets will fail to have static schedules [22]. The restrictiveness of Static Scheduling stems from the insistence on rational solution vectors. If however, the solution vector is allowed to be a function of the execution time vector, then a greater amount of flexibility results. In Co-Static Scheduling ( also called *Scheduling with total Clairvoyance* ), the assumption is that the execution time vector is known at the start of the scheduling window, although it

may be different in different windows. Accordingly, we wish to decide the following predicate:

$$\mathcal{P}_c \equiv \forall \vec{\mathbf{e}} = [e_1, e_2, \ldots, e_n] \in \mathbf{E} \quad \exists \vec{\mathbf{s}} = [s_1, s_2, \ldots, s_n] \quad \mathbf{A}.[\vec{\mathbf{s}}, \vec{\mathbf{e}}] \leq \vec{\mathbf{b}} \quad ? \tag{6}$$

*Co-static scheduling permits maximum flexibility during the dispatching phase, in that if a constraint system is not co-statically schedulable, then it is not schedulable.* However, query (6) is `coNP-complete` for arbitrary constraint sets, as shown in [22]. We have recently shown that the co-static schedulable query is solvable in polynomial time for standard and network constraints [20]. Co-static scheduling queries are applicable in Flow-shops [15].

### 4.3 Parametric Scheduling

Co-static scheduling requires knowledge of the execution time vector for a particular scheduling window, prior to determining the start time vector for that window. This may not be feasible in all real-time systems. Parametric scheduling ( also called *Scheduling with limited Clairvoyance* ) attempts to provide a balance between the Static an Co-Static scheduling modes. In a parametric schedule, the start time of a job is permitted to depend upon the start and execution times of jobs that have been sequenced before it *and only on those times.* In this mode, we restrict our discussion to `aph` domains, inasmuch as this simple domain preserves the hardness of schedulability queries. Thus, the parametric schedulability predicate is:

$$\mathcal{P}_p \equiv \exists s_1 \; \forall e_1 \; \in [l_1, u_1] \; \exists s_2 \; \forall e_2 \; \in [l_2, u_2] \; \ldots \exists s_n \; \forall e_n \; \in [l_n, u_n] \; \mathbf{A}.[\vec{\mathbf{s}}, \vec{\mathbf{e}}] \leq \vec{\mathbf{b}} \quad ? \tag{7}$$

## 5 A Taxonomy of Scheduling problems

From the discussion in the above sections, it is clear that in order to specify an instance of a scheduling problem in the `E-T-C` scheduling framework, it is necessary to specify:

- The nature of the execution time domain ( $\mathbf{E}$ ),
- The type of constraints on the jobs ( $\mathbf{A}$ ), and
- A description of the schedulability query ( $\mathcal{P}_s, \mathcal{P}_c, \mathcal{P}_p$ ).

Thus, a problem instance can be specified by instantiating the tuples in the $< \alpha|\beta|\gamma >$ triplet, where,

- $\alpha$ represents the execution time domain $\mathbf{E}$ - The following values are permissible for $\alpha$:
  - `aph` - $\mathbf{E}$ is an axis-parallel hyper-rectangle,
  - `poly` - $\mathbf{E}$ is a polyhedron
  - `arb` - $\mathbf{E}$ is an arbitrary convex domain.

  Clearly `aph` is the weakest domain in terms of what can be specified and `arb` is the strongest.
- $\beta$ represents the constraint matrix $\mathbf{A}(\mathbf{G}, \mathbf{H})$ - $\beta$ can assume the following values:

- `stan` - The constraints are *standard* which implies that $\mathbf{G}$ and $\mathbf{H}$ are network, unimodular matrices.
- `net` - The constraints are *network* which implies that $\mathbf{G}$ and $\mathbf{H}$ have at most two non-zero entries in any row
- `arb` - $\mathbf{G}$ and $\mathbf{H}$ are an arbitrary $m \times n$ rational matrices

Once again `stan` is the weakest constraint class, in terms of real-time constraints that it can model, whereas `arb` is the strongest.

– $\gamma$ represents the schedulability predicate - The schedulability predicate specifies what it means for a set of jobs to be schedulable; the following values are permitted:
  - `stat` - The query is concerned with static schedulability,
  - `co-stat` - The query is concerned with co-static schedulability,
  - `param` - The query is concerned with parametric schedulability.

Clearly `co-stat` is the most flexible query and `stat` is the least flexible.

Accordingly, $< aph|arb|stat >$ represents an instance of a real-time scheduling problem, in which the execution time domain is an axis-parallel hyper-rectangle, the constraints are arbitrary and the schedulability predicate is static. Our notation scheme is similar to the $< \alpha|\beta|\gamma >$ scheme for traditional scheduling models [14, 3].

## 6 Offline Analysis versus Online Dispatching

Scheduling algorithms in the `E-T-C` model possess an offline schedulability analyzer and an online dispatching component The analyzer examines the constraints on the system and the type of schedulability query involved, to determine whether a feasible schedule is possible. *This analysis is always carried out offline.* The dispatching component is concerned with determining the exact start times of the jobs in the current scheduling window. *Dispatching is always carried out online.*

For a given instance of a scheduling problem, the offline analyzer is executed exactly once. If the schedulability query is decided affirmatively, the online dispatcher is executed in every scheduling window.

## References


1. Bengt Aspvall and Yossi Shiloach. A fast algorithm for solving systems of linear equations with two variables per equation. *Linear Algebra and its Applications*, 34:117–124, 1980.
2. M. S. Bazaraa, H. D. Sherali, and C. M. Shetty. *Nonlinear Programming: Theory and Algorithms*. John Wiley, New York, second edition, 1993.
3. P. Brucker. *Scheduling*. Akademische Verlagsgesellschaft, Wiesbaden, 1981.
4. T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press and McGraw-Hill Book Company, 6th edition, 1992.
5. G. B. Dantzig. *Linear Programming and Extensions*. Princeton University Press, Princeton, NJ, 1963.
6. 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.
7. Y. Koren. *Computer Control of Manufacturing Systems*. McGraw-Hill, New York, 1983.

8. 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.
9. 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.
10. 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.
11. N. Muscettola, B. Smith, S. Chien, C. Fry, G. Rabideau, K. Rajan, and D. Yan. In-board planning for autonomous spacecraft. In *The Fourth International Symposium on Artificial Intelligence, Robotics, and Automation for Space (i-SAIRAS)*, July 1997.
12. Nicola Muscettola, Paul Morris, Barney Pell, and Ben Smith. Issues in temporal reasoning for autonomous control systems. In *The Second International Conference on Autonomous Agents*, Minneapolis, MI, 1998.
13. G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization*. John Wiley & Sons, New York, 1988.
14. M. Pinedo. *Scheduling: theory, algorithms, and systems*. Prentice-Hall, Englewood Cliffs, 1995.
15. M. Pinedo. *Scheduling: theory, algorithms, and systems*, chapter 5. In [14], 1995.
16. Manas Saksena. *Parametric Scheduling in Hard Real-Time Systems*. PhD thesis, University of Maryland, College Park, June 1994.
17. K. Subramani. *Duality in the Parametric Polytope and its Applications to a Scheduling Problem*. PhD thesis, University of Maryland, College Park, July 2000.
18. K. Subramani. On the complexity of co-static scheduling. Technical Report 2000-0006, West Virginia University, December 2000.
19. K. Subramani. Parametric scheduling for network constraints. In *The Seventh Annual International Computing and Combinatorics Conference*, 2001.
20. K. Subramani. Polynomial time algorithms for co-static scheduling. Technical report, West Virginia University, April 2001. Manuscript in Preparation.
21. K. Subramani and A. K. Agrawala. A dual interpretation of standard constraints in parametric scheduling. In *The Sixth International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems*, September 2000.
22. K. Subramani and A. K. Agrawala. The parametric polytope and its applications to a scheduling problem. Technical Report CS-TR-4116, University of Maryland, College Park, Department of Computer Science, March 2000.
23. K. Subramani and A. K. Agrawala. The static polytope and its applications to a scheduling problem. $3^{rd}$ *IEEE Workshop on Factory Communications*, September 2000.
24. I. Tsamardinos, N. Muscettola, and P. Morris. Fast transformation of temporal plans for efficient execution. In *The Fifteenth National Conference on Artificial Intelligence (AAAI-98)*.
25. Y.Koren. Cross-coupled biaxial computer control for manufacturing systems. *ASME Journal of Dynamic Systems, Measurement and Control*, 102:265–272, 1980.