Math. Struct. in Comp. Science (2005), *vol.* 15, *pp.* 383–406. © 2005 Cambridge University Press
doi:10.1017/S0960129504004657 Printed in the United Kingdom

Periodic Linear Programming with applications to real-time scheduling

K. SUBRAMANI[†]

Lane Department of Computer Science and Electrical Engineering, West Virginia University, Morgantown, WV 26506
Email: ksmani@csee.wvu.edu

Received 12 February 2003; revised 1 September 2004

In this paper we introduce a new mathematical modelling technique called Periodic Linear Programming; the periodic properties of Periodic Linear Programs (PLPs) permit the specification of inter-period constraints in embedded systems, in a straightforward and natural manner. We analyse PLPs in which the relationship between program variables is restricted to the class of difference constraints. Our analysis establishes that such PLPs can be reduced to simple linear programs, and hence decided in polynomial time. The class of difference constraints is extremely important from the perspective of embedded systems design, in that it permits the specification of complex timing constraints in real-time specification languages. A PLP can be thought of as a finite-description tool that represents infinite-state systems; although we use this tool purely for the purpose of modelling real-time scheduling problems, PLPs also find applications in other areas, such as concurrency design. In studying this programming paradigm, we develop novel techniques that, to the best of our knowledge, are not part of the literature. We build on the PLP structure to introduce a generalisation called Periodic Quantified Linear Programming; this programming paradigm permits the specification and analysis of uncertainty in the parameters of a PLP. Consequently, a Periodic Quantified Linear Program (PQLP) is the natural modelling tool to capture the requirements of periodic, embedded systems that are characterised by uncertainty in the execution times of processes, periodicity and relative timing constraints. In this paper, we use the PQLP structure to model and solve the periodic version of the zero-clairvoyant scheduling problem. Modelling uncertainty in the problem description is a typical technique used to incorporate a measure of fault-tolerance in the specification.

1. Introduction

Real-time scheduling problems are characterised by the presence of complex timing constraints between tasks and the existence of execution time variability. It is important to note that constraints such as relative timing constraints cannot be represented by precedence graphs (which are necessarily acyclic). A traditional scheduling model, such as the one described in Pinedo (1995) fails to account for either of the above characteristics.

A third issue in real-time scheduling problems is the issue of clairvoyance, which specifies when the execution time of a job is known and can be used in the computation

[†] This work was conducted in part at the VLSI CAD Laboratory at the University of California, San Diego, where the author was a Visiting Associate Professor.

of schedules. To this end, the E-T-C (Execution Time Constraints) scheduling framework was proposed in Subramani (2002b). This model addresses the issues of specifying constraints, execution time variability and clairvoyance in real-time scheduling problems in a very broad and flexible manner. The E-T-C scheduling model is built on extensions of Linear Programming and Quantified Linear Programming (Subramani 2003b); these mathematical programming models are single-shot in nature. Consequently, the model itself is suitable only for expressing constraint relationships between jobs in the same period (*intra-period*). However, applications abound in which there are relationships between successive invocations of job-sets (*inter-period*). One such application can be found in Ancilloti *et al.* (1993), which is concerned with the modelling of a traffic control system; additional applications are discussed in Section 4. In such systems, the positioning of jobs within a scheduling window is affected by the positioning of jobs in the previous scheduling window, and in turn affects the positioning of jobs in the succeeding scheduling window, thereby necessitating the use of *inter-period constraints*.

Observe that even the task of constraint specification in such systems is a non-trivial task and requires the generalisation of familiar mathematical programming models. Accordingly, we focus our efforts on developing a model that permits the specification of such inter-period constraints, and introduce the Periodic Linear Programming structure, which is a generalisation of Linear Programming. This generalisation permits a finite specification of an infinite-state system in a manner similar to Timed Automata (Alur and Dill 1994) and the structures in Esparza (1997) and Baldan *et al.* (2001).

We establish that a restricted class of PLPs can be reduced to traditional linear programs, and hence decided in polynomial time; the restrictions that we impose on PLP structure, nevertheless, permit the capturing of requirements in typical embedded systems (Kuchinski 1997). The analysis of Periodic Linear Programs (PLPs) requires the development of novel techniques that, to the best of our knowledge, have not been studied in the literature.

We build on the PLP structure to introduce an additional mathematical programming paradigm called Periodic Quantified Linear Programming, which is used to give effective models of uncertainty in zero-clairvoyant real-time systems.

The main contributions of this paper are:

(i) the development of new mathematical modelling tools in the form of Periodic Linear Programs (PLPs) and Periodic Quantified Linear Programs (PQLPs);
(ii) the establishing of sufficient conditions for the existence of periodic solutions for PLPs; and
(iii) the development of a polynomial time algorithm for zero-clairvoyant scheduling systems, in the presence of inter-period constraints.

The rest of this paper is organised as follows. Section 2 defines the general form of the Periodic Linear Programming structure and the restrictions that we shall be studying in this paper. The PQLP model and the associated decision problem are discussed in Section 3. In Section 4, we discuss issues in the specification of embedded systems that can be effectively captured through PLPs. Section 5 provides a detailed analysis of the restricted PLP problem that permits efficient decidability. In Section 6, we describe the zero-clairvoyant scheduling problem with inter-period constraints (ZCIPC) in real-time systems. A polynomial time algorithm for ZCIPC is developed in Section 7. We extend the ideas in Section 7, to the case in which the execution times of jobs are constrained through arbitrary convex domains. An example is detailed in Section 9, while implementation results are discussed in Section 10. Section 11 describes related work in the literature, and we conclude in Section 12 by summarising our contributions and outlining directions for future research.

2. Periodic Linear Programs

Linear programming models, for example, $\exists\, \vec{\mathbf{x}} \;\; \mathbf{A} \cdot \vec{\mathbf{x}} \leqslant \vec{\mathbf{b}}$, have been widely used in the modelling of relationships between the program variables of complex systems. However, these models can only represent *single-shot* situations, that is, all relationships must be completely and explicitly specified before the model is solved. Real-time embedded systems are often periodic in nature, which means that the same program variables and constraints occur repetitively in fixed-length periods. Furthermore, there are temporal constraints among instantiations of program variables in successive periods; such relationships cannot be captured by simple linear programming models. To this end, we introduce the concept of a Periodic Linear Program (PLP).

In what follows, we assume that the program variables represent instants in time and that the time axis is broken into periods of fixed-length.

Definition 2.1. A mathematical program of the form

$$\begin{aligned} \exists\, \vec{\mathbf{x^i}},\ i = 1, 2, \ldots, \infty \qquad & \mathbf{A} \cdot \vec{\mathbf{x^i}} \leqslant \vec{\mathbf{b^i}}, \\ & \mathbf{C} \cdot [\vec{\mathbf{x^i}} \;\; \vec{\mathbf{x^{i+1}}}]^{\mathbf{T}} \leqslant \vec{\mathbf{d^i}}, \end{aligned} \tag{1}$$

where $\vec{\mathbf{x^i}} = [x_1^i, x_2^i, \ldots, x_n^i]^T$ represents the program variables of the ith period, is called a Periodic Linear Program.

In Definition 2.1, $\mathbf{A}$ is an $m \times n$ rational matrix called the *intra-period constraint matrix*, $\mathbf{C}$ is an $m' \times 2 \cdot n$ rational matrix called the *inter-period constraint matrix*, $\vec{\mathbf{b^i}}$ is an m-vector and $\vec{\mathbf{d^i}}$ is an m'-vector. Note that inter-period constraints are permitted between successive periods only. We assume that the length of the period is L, that is, the window for $\vec{\mathbf{x^i}}$ is the continuous real interval $[(i-1) \cdot L,\ i \cdot L]$. We enforce the requirement that $\vec{\mathbf{x^i}} \leqslant \mathbf{i} \cdot \vec{\mathbf{L}}$, since the program variables represent time instants, where $\mathbf{i} \cdot \vec{\mathbf{L}}$ is the vector in which every component is $i \cdot L$; these constraints are part of the intra-period constraint matrix $\mathbf{A}$.

Observe that we must be able to derive $\vec{\mathbf{b^i}}$ from $\vec{\mathbf{b^1}}$; likewise, we must be able to derive the vectors $\vec{\mathbf{d^i}}$ fom $\vec{\mathbf{d^1}}$. Otherwise, the PLP does not have a compact description and the size of the problem is undefined. We only consider PLPs in which $b_j^i = b_j^1 + u_j \cdot (i-1) \cdot L,\ u_j \in \mathscr{Z}$ and $d_j^i = d_j^1 + v_j \cdot (i-1) \cdot L,\ v_j \in \mathscr{Z}$. Furthermore, for each constraint l_i in $\mathbf{A} \cdot \vec{\mathbf{x^i}} \leqslant \vec{\mathbf{b^i}}$, the u_i value is fixed and part of the input; the same holds for each constraint l_j of the system $\mathbf{C} \cdot [\vec{\mathbf{x^i}} \;\; \vec{\mathbf{x^{i+1}}}]^{\mathbf{T}} \leqslant \vec{\mathbf{d^i}}$ and the variable v_j. Thus, we see that a PLP has a compact description, in that it is completely described by $\{\mathbf{A},\ \vec{\mathbf{b}},\ \mathbf{C},\ \vec{\mathbf{d}}\}$ and the variable set $M = \{u_1, u_2, \ldots, u_m, v_1, v_2, \ldots, v_{m'}\}$. Each element of M is called a constraint interpreter or a constraint multiplier, for reasons that will become apparent later. In other words, we have a finite and *implicit* description of a (possibly) infinite state system.

For instance, Systems (2) and (3) together represent a PLP.

$$(\mathbf{A}) \begin{bmatrix} 1 & -1 \\ -1 & 1 \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} x_1^i \\ x_2^i \end{bmatrix} \leqslant \begin{bmatrix} -2 \\ 5 \\ 15 + (i-1).15 \end{bmatrix} (\vec{\mathbf{b^i}}) \tag{2}$$

$$(\mathbf{C}) \begin{bmatrix} 1 & 0 & -1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 5 \end{bmatrix} (\vec{\mathbf{d^i}}) \tag{3}$$

In the first two constraints of the intra-period constraint matrix, the constraint multiplier is 0, whereas in the third constraint, the constraint multiplier is 1.

Definition 2.2. An Order k restriction of a PLP as described in System (1), is the simple Linear Program that results by restricting the constraint system to k successive periods, starting from the first period.

Accordingly, the Order 1 restriction of a PLP is $\mathbf{A} \cdot \vec{\mathbf{x^1}} \leqslant \vec{\mathbf{b^1}}$, that is, the intra-period constraints only, the Order 2 restriction is: $\mathbf{A} \cdot \vec{\mathbf{x^1}} \leqslant \vec{\mathbf{b^1}}$, $\mathbf{C} \cdot [\vec{\mathbf{x^1}} \;\; \vec{\mathbf{x^2}}]^{\mathbf{T}} \leqslant \vec{\mathbf{d^1}}$, $\mathbf{A} \cdot \vec{\mathbf{x^2}} \leqslant \vec{\mathbf{b^2}}$ and so on, where the comma operator represents conjunction.

Definition 2.3. An Order k solution of a PLP is the Linear Programming solution of the Order k restriction.

Note that an Order k restriction has $k \times n$ variables in all.

Definition 2.4. The solution of a PLP is the solution to the Order ∞ restriction; a PLP is said to be feasible, if it has an Order ∞ solution.

The PLP-decidability problem is as follows: *Given a PLP specification, is it feasible?*

Observe that the above decision problem may not have a succinct solution, in that the Order ∞ restriction may be feasible, but require the specification of infinitely many different program variables for the description of the solution. We now focus on a class of solutions that can be succinctly described.

Definition 2.5. The solution of the Order ∞ restriction of a PLP is said to be cyclic, with period k, if the solution vector for the jth period can be computed using Algorithm 1.

Algorithm 1 Cyclic solutions to a PLP

```
Function CYCLIC-SOLUTION ($\mathbf{A}, \vec{\mathbf{b^i}}, \mathbf{C}, \vec{\mathbf{d^i}}, j$)
1: if ($j \leqslant k$) then
2:     Form the Order $j$ restriction of the PLP.
3:     Solve the restriction as a Linear Program over $n \times j$ variables.
4:     Return $\vec{\mathbf{x^j}}$.
5: else
6:     Let $\vec{\mathbf{z}}$ = CYCLIC-SOLUTION ($\mathbf{A}, \vec{\mathbf{b^i}}, \mathbf{C}, \vec{\mathbf{d^i}}, j-k$).
7:     Set $\vec{\mathbf{x^j}} = \vec{\mathbf{z}} + k \cdot \vec{L}$.
8: end if
```

The motivation behind the definition of Cyclic solutions with Period k, is that once the solution to the Order k restriction has been computed, the solution to the Order p restriction for any p, can be computed through Algorithm 1; in other words, if a PLP has a cyclic solution, this solution can be succinctly described. Furthermore, as will be shown in Section 5, if we are assured that a feasible PLP has a cyclic solution, the process of deciding it is greatly simplified. Note that depending upon the nature of the constraints involved, it is perfectly possible for a feasible PLP not to have a cyclic solution.

In order to reduce the cumbersomeness of the notation, we shall use $\vec{\mathbf{a}} + c_1$ to mean that every component of the vector $\vec{\mathbf{a}}$ is increased by c_1.

2.1. *Restrictions to PLPs*

As defined above, PLPs are extremely general structures and, at this point, it is not clear they are even decidable, much less in polynomial time. We now place the following restrictions on the PLPs that we shall be analysing:

(a) The vectors $\vec{\mathbf{b^i}}$ and $\vec{\mathbf{d^i}}$ are integral for all $i = 1, 2, \ldots,$.
(b) Every constraint in $\mathbf{A}$ is either a (strict) difference constraint or an absolute constraint. Note that a constraint of the form $x_i - x_j \leqslant c_1$ is called a difference constraint (Cormen *et al.* 1992). A constraint of the form $x_i \leqslant c_1$ or $x_i \geqslant c_2$ is called an absolute constraint.
(c) The constraints in $\mathbf{C}$ are difference constraints only.
(d) The constraint multiplier for a difference constraint is 0, whereas the constraint multiplier for a constraint of the form $x_i \leqslant c_1$ is $+1$ and the constraint multiplier for a constraint of the form $x_i \geqslant c_2$ is -1.

The reasoning behind the restrictions on the constraint multipliers is as follows: an intra-period difference constraint in the first period, say $x_1^1 - x_2^1 \leqslant 7$, will stay the same in the second period, except that the variables will be x_1^2 and x_2^2, respectively. The same argument applies for inter-period difference constraints as well. Hence, the constraint multipliers for these constraints should be zero. In the case of absolute constraints (which are only permitted in the intra-period constraint matrix), a constraint of the form $x_1^1 \leqslant c_1$ in the first period will be transformed into to $x_1^2 \leqslant c_1 + L$ in the second period and $x_1^i \leqslant c_1 + (i-1) \cdot L$, in the ith period, that is, the constraint multiplier is 1. Likewise, a constraint of the form $x_1^1 \geqslant c_1$ will be transformed into $x_1^i \geqslant c_1 + (i-1) \cdot L$ in the ith period, and hence $-x_1^i \leqslant -c_1 - (i-1) \cdot L$. In other words, the constraint multiplier is -1.

Restricting the PLP structure as discussed above results in the following properties.

(a) The vector $\vec{\mathbf{d^i}}$ stays the same for all periods i, that is, we can set $\vec{\mathbf{d^i}} = \vec{\mathbf{d}}$.
(b) If the jth constraint of the intra-period constraint matrix is a difference constraint, b_j^i is the same for all periods i. If the jth constraint of the intra-period constraint matrix is an absolute constraint of the form $x_k \leqslant ()$, then $b_j^i = b_j^1 + (i-1) \cdot L$, and if it is a constraint of the form $x_k \geqslant ()$, then $b_j^i = b_j^1 - (i-1) \cdot L$.
(c) The intra-period constraint system for the ith period can be derived from the intra-period constraint system for the first period by shifting the origin to the point $((i-1) \cdot L,\ (i-1) \cdot L, \ldots, (i-1) \cdot L)$. Hence, the Order 1 restriction PLP, *viz.*, $U_1 : \mathbf{A} \cdot \vec{\mathbf{x^1}} \leqslant \vec{\mathbf{b^1}}$ is feasible if and only if the intra-period constraint system for the

ith period, *viz.*, $U_i : \mathbf{A} \cdot \vec{\mathbf{x}}^{\mathbf{i}} \leqslant \vec{\mathbf{b}}^{\mathbf{i}}$ is feasible, for $i = 2, 3, \ldots, \infty$. Furthermore, if $\vec{\mathbf{z}}_{\mathbf{1}}$ is a solution to U_1, then $\vec{\mathbf{z}}_{\mathbf{i}} = \vec{\mathbf{z}}_{\mathbf{1}} + (i-1) \cdot L$ is a solution to U_i, and *vice versa*. The name given to this property is *Solution Preservation through Shift of Origin* (SPSO).

Accordingly, we can refer to an arbitrary PLP as

$$\exists\, \vec{\mathbf{x}}^{\mathbf{i}},\ i = 1, 2, \ldots, \infty \qquad \begin{aligned} \mathbf{A} \cdot \vec{\mathbf{x}}^{\mathbf{i}} &\leqslant \vec{\mathbf{b}}^{\mathbf{i}}, \\ \mathbf{C} \cdot [\vec{\mathbf{x}}^{\mathbf{i}}\ \ \mathbf{x}^{\vec{\mathbf{i+1}}}]^{\mathbf{T}} &\leqslant \vec{\mathbf{d}} \end{aligned} \tag{4}$$

From this point onwards, when we refer to a PLP, we mean a Periodic Linear Program with the above restrictions, that is, System (4).

3. Periodic Quantified Linear Programs

Linear Programming models do not permit the specification of uncertainty, even in single-shot situations. One technique for permitting the specification of uncertainty in linear programs is through the use of Quantified Linear Programs.

Definition 3.1. A linear program in which some of the program variables are universally quantified over a specified domain is called a Quantified Linear Program (QLP).

For instance, the mathematical program described in System (5) is a QLP:

$$\exists x_1 \forall y_1 \in [0,4] \exists x_2\ \exists x_3 \ldots \forall y_2 \in [9,14]\ \ \mathbf{A} \cdot [x_1\ x_2\ x_3\ y_1\ y_2]^T \leqslant \vec{\mathbf{b}} \tag{5}$$

Typically, x variables are existentially quantified and y variables are universally quantified. It is also customary to separate the coefficients of the existentially quantified and universally quantified variables. Accordingly, we write a QLP as

$$\mathbf{Q}(\vec{\mathbf{x}},\ \vec{\mathbf{y}})\ \ \mathbf{G} \cdot \vec{\mathbf{x}} + \mathbf{H} \cdot \vec{\mathbf{y}} \leqslant \vec{\mathbf{b}} \tag{6}$$

where $\mathbf{Q}(\vec{\mathbf{x}},\ \vec{\mathbf{y}})$ represents the quantifier specification of the QLP. As is the case with Linear Programming models, Quantified Linear Programming models are restricted in that they can only capture single-shot situations. A QLP is completely described by its quantifier string, which specifies the ordering on the variables, and the constraint system. The semantics of a QLP is as follows: an existential player non-deterministically makes moves for the x variables, and a universal player non-deterministically guesses values for the y variables. The guesses are made in the order specified by the quantifier string; furthermore, the guesses made by either player could depend on the values guessed up to the current juncture by the other player. When all the guesses have been made, we check whether all the constraints of the constraint system $\mathbf{G} \cdot \vec{\mathbf{x}} + \mathbf{H} \cdot \vec{\mathbf{y}} \leqslant \vec{\mathbf{b}}$ hold; if they have, the existential player has won the game, otherwise the universal player wins. We say that the QLP is true if the existential player has a winning strategy against any strategy adopted by the universal player. These programming structures find wide application in the specification of clairvoyant scheduling problems (Subramani 2002b). A detailed introduction to Quantified Linear Programming and methodologies to decide QLPs is available in Subramani (2003b).

Definition 3.2. A Periodic Quantified Linear Program (PQLP) is a Periodic Linear Program in which some of the program variables are universally quantified over a specified domain.

Just as PLPs are implicit descriptions of an infinite Linear Program, we intend that a PQLP is an implicit description of an infinite QLP. There is one issue though that must be addressed, *viz*., the description of the quantifier string of a PQLP. Definition 3.2 does not explicitly account for the length of the quantifier specification. However, if a PQLP is to be succinctly described, there must be a compact description for its infinite quantifier string, that is, the description must incorporate some form of periodicity. Observe that a PLP could be specified as in System (4), and the interpretation of the constraint system is straightforward, since all the quantifiers are existential. However, in the case of a PQLP the presence of both existential and quantifiers in the quantifier specification requires a careful explanation of the semantics involved. We shall postpone the discussion of this issue till Section 6, where a PQLP is used to model a real-world problem and the quantifier string will be interpreted in a manner that is consistent with the demands of the problem. For the present, a PQLP is denoted by:

$$\begin{aligned}
&\mathbf{Q_i}(\vec{\mathbf{x}}^{\mathbf{i}},\ \vec{\mathbf{y}}^{\mathbf{i}})\\
\mathbf{G}\cdot\vec{\mathbf{x}}^{\mathbf{i}} + \mathbf{H}\cdot\vec{\mathbf{y}}^{\mathbf{i}} &\leqslant \vec{\mathbf{b}}^{\mathbf{i}}\\
\mathbf{I}\cdot\vec{\mathbf{x}}^{\mathbf{i}} + \mathbf{J}\cdot\vec{\mathbf{y}}^{\mathbf{i}} &\leqslant \vec{\mathbf{d}}\\
i = 1, 2, \ldots, \infty &
\end{aligned} \tag{7}$$

where,

(a) The constraint matrices **G** and **I** are restricted in exactly the same way as the constraint matrices of a PLP,

(b) The application of the quantifier specification $\mathbf{Q_i}(\vec{\mathbf{x}}^{\mathbf{i}},\ \vec{\mathbf{y}}^{\mathbf{i}})$ will be decided by the problem being modelled.

4. Motivation

Inter-period constraints are useful for modelling relationships that exist between jobs across scheduling windows, where a scheduling window is a period. In a typical application it is important to control *jitter*, that is, the variation in the arrival time of tasks across periods. Consider the requirement that the jitter between the fourth task in the current period and the first task in the succeeding period be no greater than 5. Such a relationship can easily be represented through an inter-period constraint as: $x_1^{i+1} - x_4^i \leqslant 5, \quad \forall i = 1, 2, \ldots, \infty$. (Choi 2000) describes a number of application areas in which jitter minimisation is of paramount importance.

4.1. *A traffic control system*

Consider the operation of a typical traffic control system, such as the one discussed in Ancilloti *et al.* (1993) (See Figure 1).

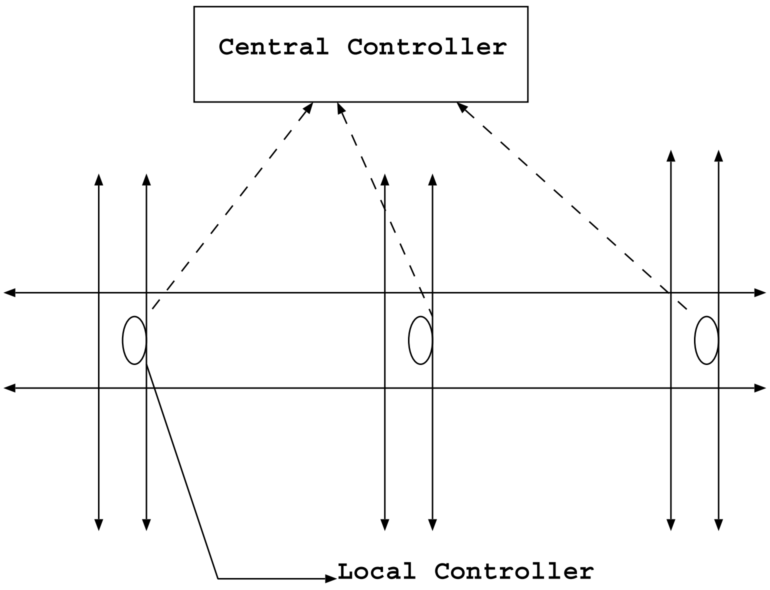

Fig. 1. A traffic controller.

Each intersection in the traffic system possesses a local controller, which carries out local control functions in addition to communicating with a central controller.

In any scheduling window the local controller displays red, green and yellow lights in strict sequential order. The repetition of the sequential order in each period represents a *cyclic schedule*. The duration for which a particular light is displayed depends upon the traffic pattern, as monitored by detectors at the intersection. For instance, the green light for a particular direction stays on longer when the traffic is heavy than when it is light. The other lights are controlled in similar fashion. The detectors issue signals when they recognise the presence of an automobile. These signals are used by the local controller to modify the parameters of the cyclic schedule. Some of the information collected from the sensors is passed on to the central controller, which then uses it to improve the traffic flows taking the current conditions into account.

An inter-period constraint would place restrictions between the signal lengths in the same period, whereas an intra-period constraint would place timing restrictions between signal lengths of adjacent periods.

4.2. *Real-time embedded systems*

Larsen *et al.* (2003) discusses the design of an embedded controller for a real-time coffee machine. In this machine there is a continuous operation that begins with the user selecting the number of coffee cups that he wants. The process of delivering the coffee to the user is constituted of a number of sub-tasks. For instance, there is a task that decides how much coffee is to be released; a second task that controls the creamer amount and yet another task that controls the sugar levels. All these sub-tasks are performed in a strict sequential order. Furthermore, there are relative timing constraints between these sub-tasks of the form:

(a) Release the creamer at least 8 seconds after the coffee has been poured;
(b) Move the coffee cup within 2 inches of the sugar orifice (distance constraints can be converted into timing requirements, (Muscatella *et al.* 1997; Muscatella *et al.*, 1998).)

There are also constraints between successive instantiations of the same task: for example,

(a) Wait at least 15 seconds, between orders so that the milk can be replenished.

4.3. *Protocols in RDBMS*

Consider a real-time data base system as described in Bestavros and Fay-Wolfe (1997). The query-processor has to execute a sequence of steps to process each query. In order to maintain a consistent database, certain delay requirements need to be met between processes in successive invocations. For instance,

(a) Wait at least 10 seconds between successive *update*() operations (inter-period constraint),
(b) A *Change-Balance*() operation precedes an *update*() operation by at least 4 seconds (intra-period constraint),
(c) A *Change-Balance*() operation occurs only after a *Check-Balance*() operation (intra-period constraint).

We thus see that the PLP framework can be put to good use in modelling constraints in a wide variety of periodic embedded systems.

5. PLP decidability

In this section we show that the class of PLPs that have been restricted as discussed in Section 2 can be decided in polynomial time. Before we proceed with our analysis, we introduce the concept of *projection in polyhedral spaces*, as discussed in Chandru and Rao (1999) and Schrijver (1987). Given a polyhedral system in Euclidean space $\Re^d$, we can project this system onto a lower dimensional space $\Re^{d'}$, $d' < d$, while preserving the set of solutions to the original system (Schrijver 1987). One of the commoner techniques for achieving this projection is the Fourier–Motzkin (FM) elimination method (Dantzig and Eaves 1973; Hochbaum and Naor 1994), which is based on variable elimination.

Let us focus on the PLP described by System (8):

$$\begin{array}{rcl} \mathbf{A} \cdot \vec{\mathbf{x^i}} & \leqslant & \vec{\mathbf{b^i}}, \\ \mathbf{C} \cdot [\vec{\mathbf{x^i}} \;\; \vec{\mathbf{x^{i+1}}}]^{\mathbf{T}} & \leqslant & \vec{\mathbf{d}} \\ & i = 1, 2, \ldots, \infty & \end{array} \tag{8}$$

Observe that System (8) is in fact a progression of linear programs, with additional constraints being added in every period. Let $S_1 = \mathbf{A} \cdot \vec{\mathbf{x^1}} \leqslant \vec{\mathbf{b^1}}$ denote the Order 1 restriction to System (8). Now consider the Order 2 restriction of the PLP, that is,

$$\mathbf{A} \cdot \vec{\mathbf{x^1}} \leqslant \vec{\mathbf{b^1}}, \quad \mathbf{A} \cdot \vec{\mathbf{x^2}} \leqslant \vec{\mathbf{b^2}}, \quad \mathbf{C} \cdot [\vec{\mathbf{x^1}} \;\; \vec{\mathbf{x^2}}]^{\mathbf{T}} \leqslant \vec{\mathbf{d}}. \tag{9}$$

When System (9) is projected onto the space spanned by $\vec{\mathbf{x^1}}$ by eliminating the variables in $\vec{\mathbf{x^2}}$, we get a polyhedral set $S_2 = \mathbf{A}' \cdot \vec{\mathbf{x^1}} \leqslant \vec{\mathbf{b'^1}}$. Observe that any solution to System (8) must belong to S_2. Accordingly, we can *reformulate* System (8) as

$$\begin{array}{rcl} \mathbf{A}' \cdot \vec{\mathbf{x^i}} & \leqslant & \vec{\mathbf{b'^i}}, \\ \mathbf{C} \cdot [\vec{\mathbf{x^i}} \;\; \vec{\mathbf{x^{i+1}}}]^{\mathbf{T}} & \leqslant & \vec{\mathbf{d}} \\ & i = 1, 2, \ldots, \infty & \end{array} \tag{10}$$

It is clear that the original PLP (described by System (8)) is feasible if and only if the reformulated PLP (described by System (10)) is too. From the properties of Fourier–Motzkin elimination, we know that projecting out a variable in a system of difference and absolute constraint with an integral right-hand side, results in a system with difference and absolute constraints with an integral right-hand side. In other words, the class of restricted PLPs is closed under FM elimination and the intra-period constraint system $\mathbf{A}' \cdot \vec{\mathbf{x^i}} \leqslant \vec{\mathbf{b'^1}}$ also satisfies the restrictions discussed in Section 2. Each reformulation of the intra-period constraint matrix results in a (possibly) different polyhedral set. Let S_i denote the polyhedral system that results from projecting the Order i restriction of System (8) onto $\vec{\mathbf{x^1}}$ space.

Lemma 5.1. $S_1 \supseteq S_2 \supseteq S_3 \ldots$

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

We need to consider the following two possibilities only:

(i) The sequence of sets S_i converges to some polyhedron in $\vec{\mathbf{x^1}}$ space.
(ii) $S_k = \phi$ for some k, that is, the reformulation process results in an infeasible set.

Corollary 5.1 follows immediately from Lemma 5.1.

Corollary 5.1. If $S_k \neq S_{k+1}$, then $S_k \supset S_{k+1}$.

Lemma 5.2. If $S_k = S_{k+1}$, for some k, then $S_j = S_k, \forall j \geqslant k$.

Proof. The inter-period constraint system $\mathbf{C} \cdot [\vec{\mathbf{x^i}} \;\; \vec{\mathbf{x^{i+1}}}]^{\mathbf{T}} \leqslant \vec{\mathbf{d}}$ can be thought of as a *reformulation operator* ∇ applied to the intra-period constraint system (input set), in that at each stage its application results in a (possibly) new intra-period constraint matrix (output set), that is, $\nabla(S_i) = S_{i+1}$. Since $S_k = S_{k+1}$, this means that $\nabla(S_k) = S_{k+1} = S_k$. Now observe that the reformulation operator is independent of the periods involved. So, reapplying ∇ to S_k will result in the same set S_k. In other words, S_k is a fixed-point of the reformulation operator (Istrescu 1981). □

Lemma 5.3. The reformulation operator ∇, described in Lemma 5.2 can be applied at most $O(n^3 \cdot L)$ times to the intra-period constraint system, at which time, either a fixed-point is reached or the reformulated system becomes infeasible.

Proof. In order to prove Lemma 5.3, we need to further analyse the structure of the constraint systems that constitute a (restricted) PLP. It is well known that a system of m linear inequalities on n variables constituted entirely of absolute and difference constraints can be represented as a constraint network on $(n+1)$ vertices having m edges (Cormen *et al.* 1992; Dechter *et al.* 1991).

In the constraint network there is a vertex for each variable, and a relative constraint between variables x_i and x_j is represented by an edge between these two vertices. Absolute constraints are represented as edges between the vertex corresponding to the variable and a special vertex x_0. The constraint network has a negative cost cycle if and only if the system is infeasible (Cormen *et al.* 1992; Dechter *et al.* 1991).

Furthermore, if the system is feasible, that is, the constraint network has no negative cost cycles, then the single-source shortest path distances from x_0 to the vertices representing the other variables is a solution to the constraint system.

From the previous discussion, it is clear that the sets S_i, $i = 1, 2, \ldots$ can be represented as constraint networks. Let G_i denote the constraint network representing S_i. Without loss of generality, we can assume that the network has an edge between each pair of vertices; if the constraint set does not specify an edge between a vertex pair, we consider it to be an edge of weight ∞. Consider the set $S_{k+1} = \nabla(S_k)$. If $S_k \supset S_{k+1}$, then, as per the mechanics of the FM elimination procedure, one or more edges in G_k (possibly an edge with infinite weight) has been replaced by an edge of smaller weight to get G_{k+1}. The weight of a newly created edge is at least one unit lesser than the weight of the edge it replaces. In other words, the weight of an edge can only decrease by the application of the ∇ operator.

We make the following observations:

(i) The constraints $0 \leqslant x_i^1 \leqslant L$, $i = 1, 2, \ldots n$ are part of the S_i unless they have been replaced by tighter constraints. Accordingly, the single-source shortest path to all vertices in the constraint network G_i must be non-negative in a feasible solution.

(ii) If the weight of an edge is finite, it can never exceed L, since the constraints specify relative distance constraints between points in time.

If the weight of an edge drops to $-(n+1) \cdot L$, at least one vertex will have negative shortest path from the source, causing a constraint violation. From the construction of the constraint network, we know that there are at most $O(n^2)$ edges in the network. Consequently, after $O(n^2 \cdot (n+1) \cdot L) = O(n^3 \cdot L)$ applications, the ∇ operator reaches a fixed point, or the reformulated system is inconsistent. □

Theorem 5.1. A feasible PLP must have a cyclic solution with period 1.

Proof. As argued in Lemma 5.3, if the PLP is feasible, the ∇ operator will produce a fixed-point of the intra-period constraint system after at most $k = O(n^3 \cdot L)$ applications. Thus, $S_k = S_{k+1} = \ldots = S_\infty$. Let $\mathbf{A}' \cdot \vec{\mathbf{x^i}} \leqslant \vec{\mathbf{b'^i}}$ denote the reformulated intra-period constraint system, when the fixed-point is reached. Now, the inter-period constraint system is redundant, so the PLP can be described as

$$\exists \vec{\mathbf{x^i}} \quad i = 1, 2, \ldots, \infty \quad \mathbf{A}' \cdot \vec{\mathbf{x^i}} \leqslant \vec{\mathbf{b'^i}} \tag{11}$$

Let $\vec{\mathbf{z}}^{\mathbf{1}}$ denote a solution to the system $\mathbf{A}' \cdot \vec{\mathbf{x}}^{\mathbf{1}} \leqslant \vec{\mathbf{b}}'^{\mathbf{1}}$. From the structure of the intra-period constraint matrix and the solution preservation through shift of origin property, it follows that $\vec{\mathbf{z}}_{\mathbf{i}} = \vec{\mathbf{z}}_{\mathbf{1}} + (i-1) \cdot L$ is a solution to the intra-period constraint system for the ith window. □

Observe that the existence of a period 1 cyclic solution trivially implies that the PLP is feasible. Hence, we have the following theorem.

Theorem 5.2. A PLP is feasible if and only if it has a cyclic solution with period 1.

The principal consequence of the above analysis is that we can set

$$\vec{\mathbf{x}}^{\mathbf{i+1}} = \vec{\mathbf{x}}^{\mathbf{i}} + L \tag{12}$$

in System (4), while preserving its solution space. Thus the inter-period constraints between window $[(i-1) \cdot L, i \cdot L]$ and $[i \cdot L, (i+1) \cdot L]$ can be merged using Equation (12), and System (8) can be expressed in terms of the $\vec{\mathbf{x}}^{\mathbf{i}}$ only. This linear system is denoted by

$$\begin{aligned} \mathbf{M} \cdot \vec{\mathbf{x}}^{\mathbf{i}} \; &\leqslant \; \vec{\mathbf{f}}^{\mathbf{i}} \\ & i = 1, 2, \ldots, \infty \end{aligned} \tag{13}$$

The polyhedral system (13) is called the `Periodic Polytope` corresponding to the PLP (1); this system is composed of n variables and $m + m'$ constraints.

Algorithm 2 Deciding a PLP

```
Function PLP-Solve ($\mathbf{A}, \vec{\mathbf{b}}^{\mathbf{i}}, \mathbf{C}, \vec{\mathbf{d}}$)
1: Form the composite system $\mathbf{M} \cdot \vec{\mathbf{x}}^{\mathbf{i}} \leqslant \vec{\mathbf{f}}^{\mathbf{i}}$, by using equation (12).
2: Set $i = 1$ to get the system $\mathbf{M} \cdot \vec{\mathbf{x}}^{\mathbf{1}} \leqslant \vec{\mathbf{f}}^{\mathbf{1}}$.
3: if ($\{\mathbf{M} \cdot \vec{\mathbf{x}}^{\mathbf{1}} \leqslant \vec{\mathbf{f}}^{\mathbf{1}}\} \neq \phi$) then
4:     Let $\vec{\mathbf{z}}^{\mathbf{1}}$ denote a solution to this system.
5:     "PLP is feasible."
6:     return(Cyclic solution $\vec{\mathbf{z}}^{\mathbf{i}} = \vec{\mathbf{z}}^{\mathbf{1}} + (i-1) \cdot L$)
7: else
8:     "PLP is infeasible."
9: end if
```

Algorithm 2 formalises our strategy for solving PLPs. Observe that because of the nature of the constraints involved, Algorithm 2 can be implemented in $O((m + m') \cdot n)$ steps using a variation of the Bellman–Ford algorithm (Cormen *et al.* 1992).

Given a periodic vector of the form $\mathbf{z}^{\mathbf{i}} = \mathbf{z}^{\mathbf{1}} + (i-1) \cdot L$ and a PLP as specified by System (4), $\vec{\mathbf{z}}^{\mathbf{i}}$ is not a solution to the PLP if and only if at least one of the constraints in the Order k system is violated for some finite k. (See Definition 2.2.)

The analysis of this section proving the existence of period 1 cyclic solutions for PLPs exploited their structure. In particular, for the analysis to hold, the following two conditions must be met:

P1: The inter-period constraint set, that is, $\nabla = \mathbf{C} \cdot [\vec{\mathbf{s^1}}\ \ \vec{\mathbf{s^2}}]^{\mathbf{T}} \leqslant \vec{\mathbf{d}}$ is independent of the period involved. Hence, it acts as a contractive operator that contracts the solution space with each successive application until either a fixed-point is reached or the reformulated system becomes infeasible. This is the gist of Lemma 5.2.

P2: If $\nabla(S_i) \neq S_i$, there is a discrete, measurable difference in the two sets. This is the gist of Lemma 5.3 where we exploited the fact that the weight of an edge is always integral and drops by at least unity as a result of applying the ∇ operator. This is not true for general constraint systems in which the changes effected by the ∇ operator are not measurable. Consequently, a PLP defined by such a constraint system may have an infinite-dimensional solution, but not a cyclic one. In other words, it is possible for a PLP to be feasible, but not have a fixed-point solution; in this case, Algorithm 2 is an incomplete decision procedure.

6. Zero-clairvoyant scheduling with inter-period constraints

In this section we describe the modelling of a practical real-time scheduling problem using Periodic Quantified Linear programs.

6.1. *Job model*

Assume an infinite time axis starting at time $t = 0$ and divided into intervals of length L. These intervals are called *scheduling windows*, that is, the first scheduling window is $[0, L]$, the second scheduling window is $[L, 2 \cdot L]$, and, in general, the ith scheduling window is $[(i-1) \cdot L, i \cdot L]$. The scheduling windows are also referred to as periods. We are given a set of ordered, non-preemptive jobs $\{J_1, J_2, \ldots J_n\}$ with instances in each scheduling window. The set $\Gamma^1 = \{J_1^1, J_2^1, \ldots, J_n^1\}$ corresponds to the instance of the job set in the first scheduling window; the sets $\Gamma^i, i = 2, \ldots, \infty$ are defined similarly. The instances of a job within a scheduling window are called tasks. Associated with each job J_i is its execution time e_i. During execution, J_i can take anywhere from l_i to u_i to complete; we denote this by $e_i \in [l_i, u_i]$. Note that the value of e_i may be different for instances of the same task in different windows. Using intervals to model the execution time of a job permits a degree of fault-tolerance, since it is rarely the case that a job will have exactly the same running time in each instance of its invocation. Let $\vec{\mathbf{e^i}} = [e_1^i, e_2^i, \ldots, e_n^i]^T$ denote the execution times of the tasks in the ith window and $\vec{\mathbf{s^i}} = [s_1^i, s_2^i, \ldots, s_n^i]^T$ denote their start times. Let $\mathbf{E}$ denote the axis-parallel hyper-rectangle (aph) $\Pi_{i=1}^n [l_i, u_i]$. Since the jobs are non-preemptive, the finish time of job J_k in scheduling window i is $(s_k^i + e_k^i)$.

6.2. *Constraint model*

The constraints on the system are described in terms of scheduling window i:

(i) *Intra-period constraints* These constraints are difference constraints between the start or finish time of a task in a period, and the start or finish time of another task *in the same period*. For instance, the requirement that the job J_2 should start at least 10 seconds after job J_1 finishes in each period is represented as $s_2^i \geqslant s_1^i + e_1^i + 10$.

Intra-period constraints also include absolute constraints, that is, a constraint of the form $s_1^i \geqslant 7 + (i-1) \cdot L$. Thus, the intra-period constraints can be described by System (14):

$$\mathbf{G} \cdot \vec{\mathbf{s^i}} + \mathbf{H} \cdot \vec{\mathbf{e^i}} \leqslant \vec{\mathbf{b^i}} \tag{14}$$

(ii) *Inter-period constraints* These constraints are difference constraints between the start or finish time of a task in a period, and the start or finish time of a task in the adjacent period. For instance, the requirement that job J_n in the current period completes at least 15 seconds before job J_2 in the next period commences is represented as $s_2^{i+1} \geqslant s_n^i + 15$. Thus, the inter-period constraints can be described by System (15):

$$\mathbf{I} \cdot [\vec{\mathbf{s^i}} \ \ \vec{\mathbf{s^{i+1}}}]^{\mathbf{T}} + \mathbf{J} \cdot [\vec{\mathbf{e^i}} \ \ \vec{\mathbf{e^{i+1}}}]^{\mathbf{T}} \leqslant \vec{\mathbf{d}} \tag{15}$$

Observe that System (14) and System (15) together constitute a PLP, as described in Section 2.1, in that the constraints are restricted to be of the form required by a PLP.

In System (14), $\mathbf{G}$ and $\mathbf{H}$ are $m \times n$ matrices and $\vec{\mathbf{b^i}}$ is an integral m-vector; similarly, in System (15), $\mathbf{I}$ and $\mathbf{J}$ are $m' \cdot n$ matrices and $\vec{\mathbf{d}}$ in an integral m'-vector.

Note that we are interested in 'hard' real-time systems only (Stankovic and Ramamritham 1998), in that the constraints cannot be violated, regardless of the values assumed by the execution times of tasks.

6.3. *Query model*

In order to completely characterise a real-time scheduling problem, we need to specify the type of schedulability query involved. As described in Subramani (2002b), there are three types of schedulability queries as far as real-time scheduling is concerned: *viz.*, zero-clairvoyant, partially clairvoyant and totally clairvoyant. The point at which the execution time of a job becomes available to the dispatcher determines the type of clairvoyance. In this paper, we shall be concerned with zero-clairvoyant scheduling; in this form of scheduling, the dispatcher is not aware of the execution time of a job, *even after it has finished execution*. There are a number of practical instances for which zero-clairvoyant scheduling is the only methodology available, since online computation is not permitted. For an overview of zero-clairvoyant scheduling in the presence of intra-period constraints only, see Subramani (2002a).

For the case in which there are no inter-period constraints, the zero-clairvoyant scheduling query is given by

$$\exists \vec{\mathbf{s}} \ \forall \vec{\mathbf{e}} \in \mathbf{E} \ \ \mathbf{G} \cdot \vec{\mathbf{s}} + \mathbf{H} \cdot \vec{\mathbf{e}} \leqslant \vec{\mathbf{b}}, \tag{16}$$

where $\mathbf{E}$ is the aph $\Pi_{i=1}^n [l_i, u_i]$, representing the variation in the execution time of each job. Note that there is only one period in System (16) and the query is asking whether there exists a single start-time vector $\vec{\mathbf{s}}$ that holds for all valid execution times of each job in the job set. Let $\vec{\mathbf{z_s}}$ denote such a vector. Then, we must have $\mathbf{G} \cdot \vec{\mathbf{z_s}} \leqslant \vec{\mathbf{b}} - \mathbf{H} \cdot \vec{\mathbf{e}}$ for all $\vec{\mathbf{e}} \in \mathbf{E}$.

In the presence of inter-period constraints, we are essentially interested in the same guarantee, that is, we desire a start-time vector that does not know the execution time of a job, *even after it has completed executing*. Thus, we are interested in the following schedulability query:

$$\exists \vec{\mathbf{s}}^{\mathbf{1}}, \vec{\mathbf{s}}^{\mathbf{2}} \ldots \forall \vec{\mathbf{e}}^{\mathbf{1}}, \vec{\mathbf{e}}^{\mathbf{2}}, \ldots \quad [(14), (15)] \tag{17}$$

In System (17), the start-time vector $\vec{\mathbf{s}}^{\mathbf{i}}$ for the ith window does not depend on the execution time vector of any scheduling window.

System (17) can be described succinctly in the following form:

$$\begin{aligned} &\exists \vec{\mathbf{s}}^{\mathbf{i}} \quad \forall \vec{\mathbf{e}}^{\mathbf{i}} \in \mathbf{E} \quad i = 1, 2, \ldots, \infty \\ &\qquad\qquad \mathbf{G} \cdot \vec{\mathbf{s}}^{\mathbf{i}} + \mathbf{H} \cdot \vec{\mathbf{e}}^{\mathbf{i}} \leqslant \vec{\mathbf{b}}^{\mathbf{i}} \\ &\mathbf{I} \cdot [\vec{\mathbf{s}}^{\mathbf{i}} \ \ \vec{\mathbf{s}}^{\mathbf{i+1}}]^{\mathbf{T}} + \mathbf{J} \cdot [\vec{\mathbf{e}}^{\mathbf{i}} \ \ \vec{\mathbf{e}}^{\mathbf{i+1}}]^{\mathbf{T}} \leqslant \vec{\mathbf{d}} \end{aligned} \tag{18}$$

Observe that System (18) is, in fact, a Periodic Quantified Linear Program (PQLP). It is to be understood that each execution time vector $\vec{\mathbf{e}}^{\mathbf{i}}$ belongs to its own execution time domain $\mathbf{E}$, that is, the execution time vectors of different scheduling windows are independent of each other. The PQLP, as specified, can be interpreted in a variety of ways; for the purposes of this paper, we attach the semantics of System (17) to System (18). This issue was first raised in Section 3 and has now been adequately addressed.

Definition 6.1. The zero-clairvoyant scheduling problem, with inter-period constraints (ZCIPC), is concerned with deciding whether System (18) is true.

System (18) is also referred to as the schedulability query or the schedulability predicate.

The chief features of the zero-clairvoyant scheduling problem are as follows:

(a) The jobs are ordered, that is, the execution sequence is known. This is unlike traditional scheduling models, where the goal is to determine the execution sequence so as to optimise an objective function.

(b) The matrices (**G**, **H**) and (**I**, **J**) permit the specification of complex constraints, such as relative timing requirements; constraints of traditional scheduling problems such as precedence requirements, deadline and ready-time requirements are easily represented in this framework.

(c) The execution times of the jobs are not fixed constants, but range-bound variables. Specifying the execution time of a job as a range, greatly enhances the fault-tolerance of the system.

7. The scheduling algorithm

In this section, we shall focus on developing a polynomial time algorithm for the ZCIPC problem.

As was the case with PLPs, we define the Order k restriction of the PQLP specified by System (18).

Definition 7.1. An Order k restriction of a PQLP as described in System (1) is the simple Quantified Linear Program that results by restricting the constraint system to k successive periods, starting from the first period.

Accordingly,

(a) The Order 1 restriction of the PQLP is

$$\begin{aligned} &\exists \vec{\mathbf{s^1}} \forall \vec{\mathbf{e^1}} \in \mathbf{E} \\ \mathbf{G} \cdot \vec{\mathbf{s^1}} + \mathbf{H} \cdot \vec{\mathbf{e^1}} &\leqslant \vec{\mathbf{b^1}} \end{aligned}$$

That is, only the intra-period constraints of the first window are in the constraint set.

(b) The Order 2 restriction is

$$\begin{aligned} &\exists \vec{\mathbf{s^1}},\ \vec{\mathbf{s^2}}\ \ \forall \vec{\mathbf{e^1}},\ \vec{\mathbf{e^2}} \in \mathbf{E} \\ \mathbf{G} \cdot \vec{\mathbf{s^1}} + \mathbf{H} \cdot \vec{\mathbf{e^1}} &\leqslant \vec{\mathbf{b^1}} \\ \mathbf{G} \cdot \vec{\mathbf{s^2}} + \mathbf{H} \cdot \vec{\mathbf{e^2}} &\leqslant \vec{\mathbf{b^2}} \\ \mathbf{I} \cdot [\vec{\mathbf{s^1}}\ \ \vec{\mathbf{s^2}}]^{\mathbf{T}} + \mathbf{J} \cdot [\vec{\mathbf{e^1}}\ \ \vec{\mathbf{e^2}}]^{\mathbf{T}} &\leqslant \vec{\mathbf{d}} \end{aligned}$$

(c) The Order k restriction is given by

$$\begin{aligned} &\exists \vec{\mathbf{s^1}},\ \vec{\mathbf{s^2}} \ldots \vec{\mathbf{s^k}}\ \forall \vec{\mathbf{e^1}}\ \vec{\mathbf{e^2}}, \ldots, \vec{\mathbf{e^k}} \in \mathbf{E} \\ \mathbf{G} \cdot \vec{\mathbf{s^i}} + \mathbf{H} \cdot \vec{\mathbf{e^i}} &\leqslant \vec{\mathbf{b^i}} \\ \mathbf{I} \cdot [\vec{\mathbf{s^i}}\ \ \vec{\mathbf{s^{i+1}}}]^{\mathbf{T}} + \mathbf{J} \cdot [\vec{\mathbf{e^i}}\ \ \vec{\mathbf{e^{i+1}}}]^{\mathbf{T}} &\leqslant \vec{\mathbf{d}} \\ i = 1, 2, \ldots, k-1& \\ \mathbf{G} \cdot \vec{\mathbf{s^k}} + \mathbf{H} \cdot \vec{\mathbf{e^k}} &\leqslant \vec{\mathbf{b^k}} \end{aligned}$$

If P_s^k is used to define the Order k restriction of the PQLP, the ZCIPC problem can be thought of as deciding the truth value of P_s^∞.

7.1. *Constraint analysis*

Note that for each k, P_s^k is a fixed-dimensional Quantified Linear Program, and hence can be decided using the quantifier elimination procedure discussed in Subramani (2003b). This procedure uses variable substitution to eliminate all the universally quantified variables, that is, $\vec{\mathbf{e^k}}$ through $\vec{\mathbf{e^1}}$, and then the Fourier–Motzkin elimination procedure is used to eliminate the existentially quantified variables $\vec{\mathbf{s^k}}$ through $\vec{\mathbf{s^2}}$. In each step, the dimension of the QLP is reduced by 1; the procedure terminates when either an inconsistency is reached or a feasible range is found for the variable s_1^1. As discussed previously, the Fourier–Motzkin procedure preserves the difference constraint structure. The intra-period constraint set, $\mathbf{I} \cdot [\vec{\mathbf{s^i}}\ \ \vec{\mathbf{s^{i+1}}}]^{\mathbf{T}} + \mathbf{J} \cdot [\vec{\mathbf{e^i}}\ \ \vec{\mathbf{e^{i+1}}}]^{\mathbf{T}} \leqslant \vec{\mathbf{d}}$, which serves as the reformulation operator ∇, is independent of the period involved and thus condition **P1** is satisfied. ∇ is repeatedly applied to the intra-period constraint set, $\mathbf{G} \cdot \vec{\mathbf{s^1}} + \mathbf{H} \cdot \vec{\mathbf{e^1}} \leqslant \vec{\mathbf{b^1}}$, until a fixed-point is obtained or an inconsistency is detected. Once again, the reformulation operator can be applied at most $O(n^3 \cdot L)$ times, on account of the restricted structure of the constraint set.

Furthermore, the elimination of either an existentially quantified variable or a universally quantified variable from the restricted constraint set preserves integrality and hence, if the reformulated polyhedron is different from the current polyhedron, the change is discrete and measurable. It follows that condition **P2** is also satisfied and thus the entire discussion in Section 5 that demonstrated the existence of fixed-point solutions for PLPs can be applied to the case of PQLPs as well. The existence of a fixed-point solution implies the existence of a cyclic solution with period 1. We therefore have the following theorem.

Theorem 7.1. System (18) is feasible if and only if it has a cyclic solution with period 1.

Proof. The proof follows from the discussions in Section 5 and this section. □

Theorem 7.1 permits us to set $\vec{\mathbf{s^{i+1}}} = \vec{\mathbf{s^i}} + L$ in the schedulability specification System 18. By combining all the constraints, we get the following PQLP:

$$\begin{aligned} &\exists \vec{\mathbf{s^i}} \quad \forall \vec{\mathbf{e^i}} \in \mathbf{E} \quad i = 1, 2, \ldots, \infty \\ &\mathbf{M} \cdot \vec{\mathbf{s^i}} + \mathbf{N} \cdot [\vec{\mathbf{e^i}} \ \ \vec{\mathbf{e^{i+1}}}]^{\mathbf{T}} \leqslant \vec{\mathbf{f^i}} \end{aligned} \tag{19}$$

Setting $i = 1$, we get the system

$$\begin{aligned} &\exists \vec{\mathbf{s^1}} \ \forall \vec{\mathbf{e^1}} \ \vec{\mathbf{e^2}} \in \mathbf{E} \\ &\mathbf{M} \cdot \vec{\mathbf{s^1}} + \mathbf{N} \cdot [\vec{\mathbf{e^1}} \ \ \vec{\mathbf{e^2}}]^{\mathbf{T}} \leqslant \mathbf{f^1} \end{aligned} \tag{20}$$

In the case of PLPs it was straightforward to compute $\vec{\mathbf{z^1}}$ by using the Bellman–Ford procedure (See Section 5). In the current case, we have the universally quantified execution time variables that need to be resolved. However, note that System (20) is a standard QLP. We can therefore apply the quantifier elimination procedure of Subramani (2003b) to obtain a solution, say $\vec{\mathbf{z^1}}$. Then the solution vector for the ith scheduling window can be computed as $\vec{\mathbf{z^i}} = \vec{\mathbf{z^1}} + (i-1) \cdot L$.

To recapitulate,

(a) The ZCIPC (zero-clairvoyant scheduling with inter-period constraints) problem is captured through the PQLP (18). The semantics of the specification are as described in Section 6.3.

(b) Restricting the number of scheduling windows to k results in System (18) becoming a k-dimensional QLP. For each k, the resultant QLP can be solved by using the variable elimination methods described in Subramani (2003b). As is the case with PLPs, the PQLP must converge to a fixed-point solution. Since a fixed-point solution implies the existence of a cyclic solution with period 1, we can set $\vec{\mathbf{s^{i+1}}} = \vec{\mathbf{s^i}} + L$ in the schedulability specification.

(c) Setting $i = 1$, we get a QLP on the variables $\vec{\mathbf{s^i}}$, $\vec{\mathbf{e^i}}$ and $\vec{\mathbf{e^2}}$. This QLP is composed of $(m + m')$ constraints on $3 \cdot n$ variables. The universally quantified variables can be eliminated in $O((m+m') \cdot n)$ time (Subramani 2003b) and the resultant set of difference constraints can be solved in $O((m+m') \cdot n)$ time using a Bellman–Ford type approach.

8. Extending the execution time domain to an arbitrary convex set

Thus far we have focussed on the case in which the execution time domain, *viz.* **E**, is an axis-parallel hyper-rectangle. However, there are applications in which there are complex relationships between the execution times themselves. For instance, in real-time embedded systems it is of paramount importance to bound the power that is expended during execution. Such a requirement can be captured through the constraint $\sum_{i=1}^{n} e_i^2 \leqslant r^2$ (Subramani 2002b), where e_i represents the execution time of the ith job. Note that modelling execution time constraints as convex sets is far more general than modelling them as uncertainty ranges. We now show that the fixed-point techniques developed in the previous section can be used in this case as well.

Let **C** denote an arbitrary convex set. The schedulability query then becomes

$$\begin{aligned} \exists \vec{\mathbf{s^i}} \quad \forall \vec{\mathbf{e^i}} \in \mathbf{C} \quad & i = 1, 2, \ldots, \infty \\ \mathbf{G} \cdot \vec{\mathbf{s^i}} + \mathbf{H} \cdot \vec{\mathbf{e^i}} &\leqslant \vec{\mathbf{b^i}} \\ \mathbf{I} \cdot [\vec{\mathbf{s^i}} \;\; \vec{\mathbf{s^{i+1}}}]^{\mathbf{T}} + \mathbf{J} \cdot [\vec{\mathbf{e^i}} \;\; \vec{\mathbf{e^{i+1}}}]^{\mathbf{T}} &\leqslant \vec{\mathbf{d}} \end{aligned} \qquad (21)$$

Although the set **C** is not an aph, the semantics used in Section 6 to describe System (21) are still applicable. Accordingly, we can define the Order k restriction of this system in precisely the same fashion as before. This restriction is given by

$$\begin{aligned} \exists \vec{\mathbf{s^1}}, \vec{\mathbf{s^2}} \ldots \vec{\mathbf{s^k}} \; \forall \vec{\mathbf{e^1}} \; \vec{\mathbf{e^2}}, \ldots, \vec{\mathbf{e^k}} \in \mathbf{C} & \\ \mathbf{G} \cdot \vec{\mathbf{s^i}} + \mathbf{H} \cdot \vec{\mathbf{e^i}} &\leqslant \vec{\mathbf{b^i}} \\ \mathbf{I} \cdot [\vec{\mathbf{s^i}} \;\; \vec{\mathbf{s^{i+1}}}]^{\mathbf{T}} + \mathbf{J} \cdot [\vec{\mathbf{e^i}} \;\; \vec{\mathbf{e^{i+1}}}]^{\mathbf{T}} &\leqslant \vec{\mathbf{d}} \\ i = 1, 2, \ldots, k-1 & \\ \mathbf{G} \cdot \vec{\mathbf{s^k}} + \mathbf{H} \cdot \vec{\mathbf{e^k}} &\leqslant \vec{\mathbf{b^k}} \end{aligned} \qquad (22)$$

Observe that we can write the Order k restriction as

$$\begin{aligned} \exists \vec{\mathbf{s^1}}, \vec{\mathbf{s^2}} \ldots \vec{\mathbf{s^k}} \; \forall \vec{\mathbf{e^1}} \; \vec{\mathbf{e^2}}, \ldots, \vec{\mathbf{e^k}} \in \mathbf{C} & \\ \mathbf{G} \cdot \vec{\mathbf{s^i}} + &\leqslant -\mathbf{H} \cdot \vec{\mathbf{e^i}} + \vec{\mathbf{b^i}} \\ \mathbf{I} \cdot [\vec{\mathbf{s^i}} \;\; \vec{\mathbf{s^{i+1}}}]^{\mathbf{T}} &\leqslant -\mathbf{J} \cdot [\vec{\mathbf{e^i}} \;\; \vec{\mathbf{e^{i+1}}}]^{\mathbf{T}} + \vec{\mathbf{d}} \\ i = 1, 2, \ldots, k-1 & \\ \mathbf{G} \cdot \vec{\mathbf{s^k}} &\leqslant -\mathbf{H} \cdot \vec{\mathbf{e^k}} + \vec{\mathbf{b^k}} \end{aligned} \qquad (23)$$

Each constraint on the right-hand side of System (23) is an affine function over the $\vec{\mathbf{e^i}}$ variables. We now show that this affine function can be replaced by a single rational number using convex minimisation. The reduction results in the transformation of the original scheduling model to a simple linear program over difference constraints.

Note that each intra-period and inter-period constraint is independently reduced in the above transformation procedure. Consider the vector $\vec{\mathbf{r_1}}$ of the first scheduling window. Its first component, say $r_1(1)$, is obtained by minimising $(\vec{\mathbf{b^1}} - \mathbf{H} \cdot \vec{\mathbf{e^1}})_1$ over **C**; the other components of $\vec{\mathbf{r_1}}$ are similarly obtained. Now consider the vector $\vec{\mathbf{q_1}}$. Its components are computed by minimising the appropriate inter-period constraint over the domain $\mathbf{C} \times \mathbf{C}$.

Algorithm 3 Transformation procedure

```
Function ORDER-k-TO-LP ($\mathbf{A}, \vec{\mathbf{b}}, \mathbf{C}, \vec{\mathbf{d}}$)
1: for ($i = 1$ to $k - 1$) do
2:     {For each period}
3:     Let $\vec{\mathbf{r_i}} = \min_{\mathbf{C}}(\vec{\mathbf{b^i}} - \mathbf{H} \cdot \vec{\mathbf{e^i}})$. {Reducing the intra-period constraints}
4:     Let $\vec{\mathbf{q_i}} = \min_{\mathbf{C}\times\mathbf{C}}(\vec{\mathbf{d}} - \mathbf{J} \cdot [\vec{\mathbf{e^i}} \ \ \mathbf{e^{i+1}}]^T)_i$. {Reducing the inter-period constraints}
5: end for
6: Let $\vec{\mathbf{r_i}} = \min_{\mathbf{C}}(\vec{\mathbf{b^k}} - \mathbf{H} \cdot \vec{\mathbf{e^k}})$.
```

Note that the cartesian product of two convex domains is convex, so this minimisation is, in fact, also a convex minimisation call.

Because of the structure of the constraint set, we need to compute $\vec{\mathbf{r_1}}$ and $\vec{\mathbf{q_1}}$ only. The vectors $\vec{\mathbf{r_2}}$ through $\vec{\mathbf{r_k}}$ can be computed through offsets, while the vectors $\vec{\mathbf{q_2}}$ through $\mathbf{q}_{\vec{\mathbf{k-1}}}$ are identical to $\vec{\mathbf{q_1}}$. Thus the entire procedure consists of $(m + m')$ convex minimisation calls, and hence the total time taken is $O((m+m')\cdot\mathscr{C})$, where $\mathscr{C}$ is the fastest convex minimisation algorithm (Hiriart-urruty and Lemarechal 1993). Subramani (2002a) provides a detailed argument establishing that replacing each constraint by a number as specified by Algorithm 3 preserves the solution space. Thus, in the Order k restriction, the execution time variables can be eliminated through convex minimisation to get a simple linear program, which in turn can be solved using Bellman–Ford techniques.

Observe that the inter-period constraint set is identical in structure to the case in which the execution time domain is an aph, and hence condition **P1** is trivially satisfied. In the case of PQLPs it was straightforward to argue that the elimination procedure preserved integrality and hence condition **P2** is satisfied as well. In the case of arbitrary convex sets though, the result of a convex minimisation call may not be integral, and hence we cannot directly conclude that **P2** is satisfied. However, observe that in any implementation of **C** on a computer, there exists only finite precision. Let Θ denote the resolution of the computer, that is, the smallest number that can be represented on it. If the application of the ∇ operator results in a reformulated system, then at least one of the edges in the corresponding constraint network drops by at least Θ. Thus, the analysis in Section 5 lets us conclude that ∇ can be applied at most $O(n^3 \cdot \frac{L}{\Theta})$ times, before a fixed-point is reached or an infeasibility is detected. It follows that condition **P2** is satisfied also, and hence System (21) has a cyclic solution with period 1.

Algorithm 4 formalises the procedure discussed above. It is clear it takes time at most $O((m+m')\cdot n + (m+m')\cdot\mathscr{C})$, where $\mathscr{C}$ is the fastest convex minimisation algorithm.

9. Example

Consider the following set of specifications for a periodic job set consisting of two jobs $\{J_1, J_2\}$. Let the period of the job set, that is, L, be 15, and the execution times belong to the aph $\tau = (e_1^i \in)[2,4] \times (e_2^i) \in)[1,3]$. Assume that the following constraints have been imposed on the execution of the jobs.

Algorithm 4 Zero-clairvoyant scheduler with convex-constrained execution time domain

```
Function ZERO-CLAIRVOYANT-CONVEX($\mathbf{G}$, $\mathbf{H}\ \vec{\mathbf{b^i}}$, $\mathbf{I}$, $\mathbf{J}$, $\vec{\mathbf{d}}$)
1: Set $\vec{\mathbf{s^{i+1}}} = \vec{\mathbf{s^i}} + L$ in System (21) to get the combined system:
       $$\exists \vec{\mathbf{s^i}} \quad \forall \vec{\mathbf{e^i}} \in \mathbf{C} \quad i = 1, 2, \ldots, \infty$$
       $$\mathbf{M} \cdot \vec{\mathbf{s^i}} + \mathbf{N} \cdot [\vec{\mathbf{e^i}}\ \mathbf{e^{i+1}}]^{\mathbf{T}} \leqslant \vec{\mathbf{f^i}} \quad (24)$$
2: Set $i = 1$ in System (24) to get the system:
       $$\exists \vec{\mathbf{s^1}}\ \forall \vec{\mathbf{e^1}}\ \forall \vec{\mathbf{e^2}} \in \mathbf{C}$$
       $$\mathbf{M} \cdot \vec{\mathbf{s^1}} + \mathbf{N} \cdot [\vec{\mathbf{e^1}}\ \vec{\mathbf{e^2}}]^{\mathbf{T}} \leqslant \vec{\mathbf{f^1}} \quad (25)$$
3: Solve System (25) using convex minimisation and the Bellman–Ford approach.
4: if ( no solution exists ) then
5:     return(The specification is infeasible.)
6: end if
7: Let $\vec{\mathbf{z^1}}$ be a solution to the System (25).
8: Set $\vec{\mathbf{z^i}} = \vec{\mathbf{z^1}} + (i-1) \cdot L$ to be the cyclic solution.
```

(i) In each period, Job J_2 starts at least 2 units after job J_1 finishes: $s_1^i + e_1^i + 2 \leqslant s_2^i$, $\forall i = 1, 2, \ldots$;

(ii) In each period, Job J_2 starts within 5 units of job J_1 finishing: $s_2^i \leqslant s_1^i + e_1^i + 5$, $\forall i = 1, 2, \ldots$;

(iii) Job J_1 in each period starts at least 5 units after job J_1 finishes in the previous period finishes:
$s_2^i + e_2^i + 5 \leqslant s_1^{i+1}$, $\forall i = 1, 2, \ldots$;

(iv) Job J_2 finishes before the end of the period:
$s_2^i + e_2^i \leqslant 15 + (i-1) \cdot 15$, $\forall i = 1, 2, \ldots$.

Expressing the constraint system in form [(14)–(15)], we get

$$\mathbf{G}\begin{bmatrix} 1 & -1 \\ -1 & 1 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} s_1^i \\ s_2^i \end{bmatrix} + \mathbf{H}\begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} e_1^i \\ e_2^i \end{bmatrix} \leqslant \begin{bmatrix} -2 \\ 5 \\ 15 + (i-1).15 \end{bmatrix} \vec{\mathbf{b^i}} \tag{26}$$

and

$$\mathbf{C}\begin{bmatrix} 0 & 1 & -1 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} s_1^i \\ s_2^i \\ s_1^{i+1} \\ s_2^{i+1} \\ e_1^i \\ e_2^i \\ e_1^{i+1} \\ e_2^{i+1} \end{bmatrix} \leqslant \begin{bmatrix} 5 \end{bmatrix} \vec{\mathbf{d}} \tag{27}$$

Table 1. *Machine characteristics.*

Speed	450 Mhz
Processor	AMD K-6
Memory	256 Mb RAM
Cache	1 MB L2
Operating System	Suse Linux 7.0
Kernel	2.2.16
File System	ReiserFS
Language	Perl 5.005-03
Software	*lp-solve*

Table 2. *Summary of results for zero-clairvoyant scheduling.*

(n)	IrPC (m)	IPC (m')	Time (seconds)
5	11	20	0.05
10	21	40	0.05
15	31	60	0.09
20	41	80	0.10
25	51	100	0.18
30	61	120	0.17
35	71	140	0.28
40	81	160	0.38
45	91	180	0.51
50	101	200	0.59
100	201	400	1.81

We set $\vec{\mathbf{s^{i+1}}} = \vec{\mathbf{s^{i}}} + L$ to get the following fixed-point solution:

$$\begin{bmatrix} s_1^i \\ s_2^i \end{bmatrix} = \begin{bmatrix} 0 + (i-1) \cdot 15 \\ 6 + (i-1) \cdot 15 \end{bmatrix} \tag{28}$$

Inspection of the solution confirms that the constraint set is indeed satisfied in each scheduling window.

10. Implementation

We implemented the above-mentioned strategy for instances of the ZCIPC problem in which the execution time domain was an aph.

Table 1 describes system specifications, while Table 2 presents our implementation results. We chose to use the less-efficient, but more general *lp-solve* software, as opposed to Bellman–Ford procedures.

In Table 2, n denotes the number of jobs, m denotes the number of intra-period constraints and m' denotes the number of inter-period constraints.

11. Related work

Linear programming models have been widely used in the AI community to model and decide constraint-based schedulability problems (Dechter *et al.* 1991; Muscatella *et al.* 1997). The general framework of Quantified Linear Programming was outlined in Subramani (2003b), in which a polynomial time procedure was discussed for certain classes of QLPs.

The `E-T-C` scheduling framework was proposed in Subramani (2002b) to model general, intra-period constraints using linear programs and QLPs. Within this model, zero-clairvoyant scheduling has been analysed in Subramani (2002a), and partially clairvoyant scheduling has been analysed in Gerber *et al.* (1995) and Subramani (2003a). The problem of deciding schedulability queries for non-preemptable jobs in the presence of inter-period constraints was proposed in Choi (1997). Choi (2000) considered the problem of deciding partially clairvoyant schedulability when all constraints are strictly relative and demonstrated the existence of convergent schedules in the presence of inter-period constraints. These techniques are fundamentally different from ours and do not generalise easily. Our work here, that is, the use of fixed-point theory represents the first attempt to specify a formal framework for capturing inter-period constraints and generalises the work in Subramani (2002a) to the case of inter-period constraints.

In other real-time scheduling work, Fohler (Fohler 1995) studied the joint scheduling of distributed, complex periodic and hard, aperiodic jobs. In their model, the periodic jobs are considered preemptable. Liu and Layland (1973) is a seminal paper in the area of preemptive scheduling of periodic jobs; schedulability conditions are derived in terms of the Earliest Deadline First (EDF) heuristic. They showed that EDF scheduling was optimal, in that if there exists a preemptive schedule, then there exists a preemptive schedule using the EDF heuristic. Precedence constrained scheduling of real-time jobs is discussed in Chetto *et al.* (1990).

12. Conclusions

The principal contributions of this paper include the following:

1 Definition of the Periodic Linear Programming and Periodic Quantified Linear Programming frameworks;
2 Analysis of the conditions under which these frameworks have cyclic solutions;
3 Capture of the requirements of a problem in real-time scheduling using Periodic Quantified Linear Programs;
4 Generalisation of the execution time domain of jobs to arbitrary convex sets.

The analysis that we have used is, to the best of our knowledge, completely novel and not part of the literature. It would be interesting to see if the results hold for a class of constraints that is wider than difference constraints, such as sum constraints. From Subramani (2003b), it follows that the analysis of this paper does indeed hold for Totally Unimodular constraint matrices.

The principal open question concerns the applicability of our techniques to partially clairvoyant and totally clairvoyant scheduling systems. In the case of partially clairvoyant scheduling, a pseudo-polynomial time algorithm is known (Choi 2000), but there are no known results for totally clairvoyant scheduling. We suspect that our techniques can be extended to these systems to provide polynomial time algorithms.

We are also interested in additional real-world problems that can be modelled through Periodic Linear Programs. It would be interesting to see if there is any equivalence between the PQLP framework and the Timed Automata framework of Alur and Dill (1994). We would like to mention that Papadimitriou (1994) describes various periodic machine scheduling problems using variants of Periodic-SAT, and exploring the connections between PLPs and Periodic-SAT problems would be rewarding.

13. Acknowledgements

I am indebted to Michael Bond of the West Virginia Library System for his efforts in the implementation of the algorithms. I am also grateful to Kim G. Larsen for drawing my attention to Larsen *et al.* (2003).

References

Alur, R. and Dill, D. L. (1994) A theory of timed automata. *Theoretical Computer Science* **126** (2) 183–235.

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

Baldan, P., Corradini, A. and Konig, B. (2001) A static analysis technique for graph transformation systems. In: *Proceedings of CONCUR 2001*, Springer-Verlag 381–395.

Bestavros, A. and Fay-Wolfe, V. (eds.) (1997) *Real-Time Database and Information Systems, Research Advances*, Kluwer Academic Publishers.

Chandru, V. and Rao, M. R. (1999) Linear programming. In: *Algorithms and Theory of Computation Handbook*, CRC Press, Boca Raton, Florida.

Chetto, H., Silly, M. and Bouchentouf, T. (1990) Dynamic scheduling of real-time tasks under precedence constraints. *JRTS* **2** 181–194.

Choi, S. (1997) *Dynamic Time-based scheduling for Hard Real-Time Systems*, Ph.D. thesis, University of Maryland, College Park.

Choi, S. (2000) Dynamic time-based scheduling for hard real-time systems. *Journal of Real-Time Systems.*

Cormen, T. H., Leiserson, C. E. and Rivest, R. L. (1992) *Introduction to Algorithms*, MIT Press and McGraw-Hill Book Company, Boston, Massachusetts, 2*nd* edition.

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

Dechter, R., Meiri, I. and Pearl, J. (1991) Temporal constraint networks. *Artificial Intelligence* **49** 61–95.

Esparza, J. (1997) Decidability of model-checking for infinite-state concurrent systems. *Acta Informatica* **34** 85–107.

Fohler, G. (1995) Joint scheduling of distributed complex, periodic and hard aperiodic tasks in statically scheduled systems. In: *Proceedings IEEE Real-Time Systems Symposium*, IEEE Computer Society Press.

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

Hiriart-urruty, J. B. and Lemarechal, C. (1993) *Convex Analysis and Minimization Algorithms*, Springer-Verlag.

Hochbaum, D. S. and Naor, J. (1994) Simple and fast algorithms for linear and integer programs with two variables per inequality. *SIAM Journal on Computing* **23** (6) 1179–1192.

Istratescu, V. I. (1981) *Introduction to Linear Operator theory*, Marcel Dekker Inc., New York.

Kuchinski, K. (1997) Embedded system synthesis by timing constraints solving. In: *International Symposum on Systems Synthesis*, IEEE Computer Society 50–57.

Larsen, K. G., Steffen, B. and Weise, C. (2003) Continuous modelling of real time and hybrid systems. BRICS Technical Report, Aalborg Universitet.

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

Muscettola, N., Smith, B., Chien, S., Fry, C., Rabideau, G., Rajan, K. and Yan, D. (1997) In-board planning for autonomous spacecraft. In: *The Fourth International Symposium on Artificial Intelligence, Robotics, and Automation for Space* (i-SAIRAS).

Muscettola, N., Morris, P., Pell, B. and Smith, B. (1998) Issues in temporal reasoning for autonomous control systems. In: *The Second International Conference on Autonomous Agents*, Minneapolis.

Papadimitriou, C. H. (1994) *Computational Complexity*, Addison-Wesley.

Pinedo, M. (1995) *Scheduling: theory, algorithms, and systems*, Prentice-Hall.

Schrijver, A. (1987) *Theory of Linear and Integer Programming*, John Wiley and Sons.

Stankovic, J. A., Spuri, M., Ramamritham, K. and Buttazzo, G. C. (1998) (eds.) *Deadline Scheduling for Real-Time Systems*, Kluwer Academic Publishers.

Subramani, K. (2002a) An analysis of zero-clairvoyant scheduling. In: Katoen, J.-P. and Stevens, P. (eds.) Proceedings of the 8th International Conference on Tools and Algorithms for the construction of Systems (TACAS). *Springer-Verlag Lecture Notes in Computer Science* **2280** 98–112.

Subramani, K. (2002b) A specification framework for real-time scheduling. In: Grosky, W. I. and Plasil, F. (eds.) Proceedings of the 29th Annual Conference on Current Trends in Theory and Practice of Informatics (SOFSEM). *Springer-Verlag Lecture Notes in Computer Science* **2540** 195–207.

Subramani, K. (2003a) An analysis of partially clairvoyant scheduling. *Journal of Mathematical Modelling and Algorithms* **2** (2) 97–119.

Subramani, K. (2003b) An analysis of quantified linear programs. In: Calude, C. S. *et al.* (eds.) Proceedings of the 4th International Conference on Discrete Mathematics and Theoretical Computer Science (DMTCS). *Springer-Verlag Lecture Notes in Computer Science* **2731** 265–277.