Dynamic Dispatching of Cyclic Real-Time Tasks with Relative Constraints *

Seonho Choi Ashok K. Agrawala
Institute for Advanced Computer Studies
Department of Computer Science
University of Maryland
College Park, MD 20742
{seonho,agrawala}@cs.umd.edu

Index Terms: Real-time, operating systems, scheduling, dispatching, relative constraints.

Abstract

In some hard real-time systems, relative timing constraints may be imposed on task executions, in addition to the release time and deadline constraints. A periodic task may have jitter constraints between the start or finish times of any two consecutive executions. Relative constraints such as separation or relative deadline constraints may be given between start or finish times of tasks [4].

One approach is to find a total order on a set of N jobs in a scheduling window, and cyclically use this order at run time to execute the jobs. However, in the presence of the relative constraints, if the job execution times are nondeterministic with defined lower and upper bound, it is not always possible to statically assign start times at pre-runtime without sacrificing the schedulability [4].

We develop a technique called *dynamic cyclic dispatching* to enforce relative constraints along with release time and deadline constraints. An ordered set of N jobs is assumed to be given within a scheduling window and this schedule(ordering) is cyclically repeated at runtime. An off-line algorithm is presented to check the schedulability of the job set and to obtain parametric lower and upper bounds on the start times of jobs, if the job set is schedulable. Then, these parametric bounds are evaluated at runtime to obtain a valid time interval during which jobs can be started. The complexity of this off-line component is shown to be $O(n^2N^3)$ where n is the number of jobs in a scheduling window that have relative constraints with jobs in the next scheduling window. An online algorithm can evaluate these bounds in $O(N)$ time. Especially, for a certain class of relative constraints, it is shown that the off-line component requires $O(N^3 + n^5)$ computation time.

Unlike static approaches which assign fixed start times to jobs in the scheduling window, our approach not only allows us to flexibly manage the slack times with the schedulability of a task set not affected, but also yields a guaranteed schedulability in the sense that, if other dispatching mechanism can schedule the job sequences satisfying all given constraints, then our mechanism can also schedule them.

*This work is supported in part by ONR and ARPA under contract N66001-95-C-8619 to the Computer Science Department at the University of Maryland. The views, opinions, and/or findings contained in this report are those of the author(s) and should not be interpreted as representing the official policies, either expressed or implied, of the Advanced Research Projects Agency, ONR or the U.S. Government.

1 Introduction

A common approach to characterize hard real-time tasks with repetitive requests is to use periodic task model [7]. In the model, every task needs to be executed once during each of its periods, and the executions, called *jobs*, of the same task in different periods are independent. However, in some hard real-time systems, *relative timing constraints* should be satisfied between event occurrence times. as well as *release time* and *deadline* constraints on jobs. For example, control output events in two successive jobs of a periodic task may have to occur with the jitter requirement satisfied. That is, the difference of two event occurrence times, called *jitter*, should lie between a lower and an upper bound. The occurrences of events in different tasks may also be constrained from the requirements and characteristics of the environment by separation or relative deadline constraints [4]. These relative constraints have to be enforced in many real-time control systems such as process control systems and flight control systems [1], etc. For example, in process control systems, it has been shown that jitter constraints have more influence on control systems performance than the frequency constraints [6].

Usually, these relative constraints between events are transformed into relative constraints between start or finish times of the jobs to make feasible the process of scheduling and dispatching of jobs [5, 4]. In [5] a preemptive fixed priority scheduling algorithm is developed to schedule periodic tasks with relative deadline constraints between finish times of two successive jobs of periodic tasks. However, other types of relative constraints are not allowed in that work and it is not possible to flexibly manage slack times at runtime for dynamic tasks. In [4] dispatching of a totally ordered non-preemptive job set with any min/max constraints is studied and a new scheduling mechanism called *parametric scheduling* is developed. In that paper, it is also shown that a traditional static scheduling approach, in which job start times are statically scheduled under the assumption that every job spends its worst case execution time, doesn't work any more for job sets with general min/max constraints even when a total ordering among jobs is given. Furthermore, in parametric scheduling scheme, it is possible to effectively schedule aperiodic tasks at run-time by dynamically managing job start times. However, the job set in parametric scheduling scheme consists of a finite number of jobs with a finite number of constraints. This implies that the approach cannot be applied to a periodic task model, since periodic tasks may invoke an infinite number of jobs with an infinite number of relative constraints. In a traditional time-based scheduling scheme the job start times are statically decided in a scheduling window, and this static schedule is cyclically used at run-time. In the presence of jitter constraints between start times of non-preemptive jobs, the problem of finding a static schedule has been addressed in [2]. However, this static cyclic scheduling approach only allows certain types of min/max constraints to be specified, and it only works under low utilization. Moreover, it is very difficult to flexibly manage job start times at run-time to incorporate any dynamic tasks such as aperiodic tasks into the schedule.

In this paper, we develop a new job dispatching scheme, called *dynamic cyclic dispatching*, that overcomes the above-mentioned limitations of previous approaches. Every job is assumed to be non-preemptive, and a totally ordered N cyclic jobs, $< \tau_1, \tau_2, \ldots, \tau_N >$, are assumed to be given on a scheduling window $[0, L]$ where L is a

scheduling window size[1]. This schedule is cyclically repeated at runtime. Let $\Gamma^j = \{\tau_1^j, \ldots, \tau_N^j\}$, $j \geq 1$, denote a set of N jobs to be scheduled in a j-th scheduling window $[(j-1)L, jL]$. Relative constraints may be given in the form of standard relative constraints [2] between the start or finish times of the jobs in two consecutive scheduling windows, i.e., in Γ^j and Γ^{j+1}, as well as the jobs within one scheduling window Γ^j. These relative constraints as well as the release time and deadline constraints need to be satisfied throughout the system operation time. Figure 1 shows an example job set with their constraints, where each job set, Γ^j, consists of $N = 2$ jobs and the relative constraints are defined across the boundaries of the job sets as well as within a job set. In the figure, s_i^j and f_i^j are runtime variables denoting the actual start and finish times of a job τ_i^j, respectively.

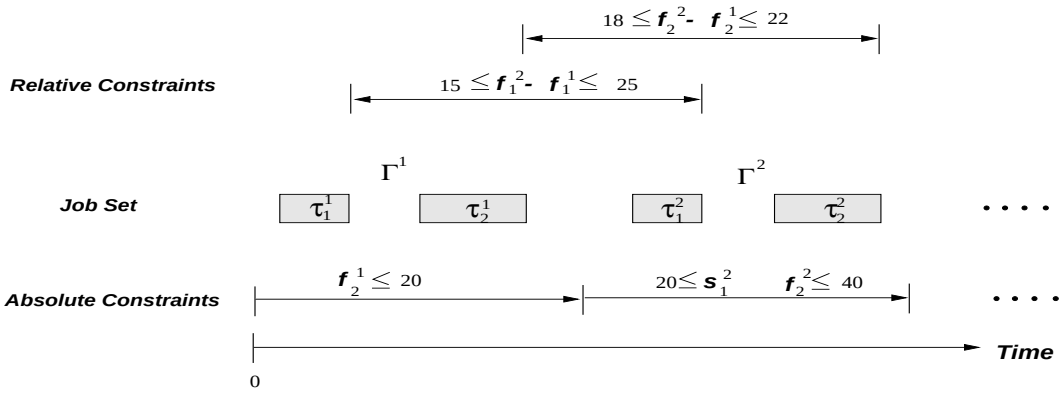

Figure 1: Example Job Sequence

In this paper, we find an off-line algorithm to check the schedulability of job sets, Γ^1, Γ^2, ... And, if they are schedulable, the parametric lower bound and upper bound functions for each job start time are found in terms of the start or finish times of the previous jobs. These bounds are evaluated at runtime within $O(N)$ time. Suppose that τ_i^j belongs to Γ^j, then the parametric lower and upper bound functions of s_i^j, are parameterized in terms of the start and finish times of already executed jobs in Γ^{j-1} and Γ^j. Another important result is that only N pairs of parametric bound functions have to be stored and cyclically used at runtime. The off-line algorithm has a pseudo-polynomial complexity $O(n^2N^3)$, where n is the number of jobs in one scheduling window that have relative constraints with jobs in the next scheduling window. Especially, if only jitter constraints on periodic tasks are allowed, it can be shown that the off-line and online components require $O(n^4N)$ and $O(n)$ times, respectively. Also, it is shown that, for a certain subset of standard constraints, called *restricted standard constraints*, the off-line algorithm requires at most $O(N^3 + n^5)$.

The dynamic cyclic scheduling scheme not only enables us to check the schedulability of the cyclically-constrained job set in the presence of relative constraints, but also makes it possible to flexibly manage the slack times at runtime without affecting the schedulability of the jobs, which is not possible in static scheduling approach.

The rest of the paper is organized as follows. In Section 2, we present a formal definition of the parametric

[1] It is allowed in this paper that the deadline of a job to be greater than the end of a scheduling window to which the job belongs.
[2] Standard constraints are defined in Section 2.

scheduling problem. In Section 3, we summarize the related works on scheduling task sets with relative constraints. And then, in Section 4, the parametric scheduling approach is developed by using the quantifier elimination techniques, and by transforming the constraint set into an equivalent graph. In Section 5, example task instance sequences are given with parametric calendars found from the off-line algorithm. Finally, a conclusion of the paper follows in Section 6.

2 Problem Description

Let $\Gamma^j = \{\tau_i^j \mid i = 1, \ldots, N\}$ denote an ordered set of N jobs to be dispatched sequentially in a j-th scheduling window $[(j-1)L, jL]$ where L is a positive integer denoting a scheduling window size.. The jobs are executed non-preemptively in this order. At runtime, this job set will be cyclically scheduled in consecutive scheduling windows. In other words, τ_i^j and τ_i^k are jobs of the same task.

Then, let $\Gamma^{1,k} = \Gamma^1 \cup \Gamma^2 \cup \ldots \cup \Gamma^k$ denote a set of jobs to be executed in a time interval $[0, kL]$. Each job τ_i^j ($j \geq 1$, $1 \leq i \leq N$) has the following set of parameters that may have integer values:

- A runtime variable s_i^j denoting the actual start time of τ_i^j
- A runtime variable e_i^j representing the actual execution time spent for τ_i^j
- A runtime variable $f_i^j = s_i^j + e_i^j$ denoting the actual finish time of τ_i^j
- A constant l_i^j corresponding to the minimum execution time of τ_i^j
- A constant u_i^j denoting the maximum execution time of τ_i^j.

Note that it is simply assumed that execution times of jobs are nondeterministic and bounded from above and below, which is a realistic assumption in many real-time systems.

Standard constraints are defined next that may be imposed on $\{s_i^j, e_i^j \mid 1 \leq j \leq k,\ 1 \leq i \leq N\}$ for $\Gamma^{1,k}$.

Definition 1 (Standard Constraints) *A standard constraint involves the variables of at most two jobs, τ_a^j and $\tau_b^l (1 \leq a \leq b \leq N, \mid j - l \mid \leq 1)$, where s_a^j (or $s_a^j + e_a^j$) appears on one side of "$\leq$," and s_b^l (or $s_b^l + e_b^l$) appears on the other side of the "$\leq$." For two jobs, τ_a^j, τ_b^l, the following constraints are permitted (where c_i is an arbitrary constant) and called* relative *standard constraints:*

$$
\begin{array}{rcl@{\qquad}rcl}
s_a^j - s_b^l & \leq & c_1 & s_b^l - s_a^j & \leq & c_5 \\
s_a^j - (s_b^l + e_b^l) & \leq & c_2 & s_b^l - (s_a^j + e_a^j) & \leq & c_6 \\
s_a^j + e_a^j - s_b^l & \leq & c_3 & s_b^l + e_b^l - s_a^j & \leq & c_7 \\
s_a^j + e_a^j - (s_b^l + e_b^l) & \leq & c_4 & s_b^l + e_b^l - (s_a^j + e_a^j) & \leq & c_8
\end{array}
\tag{1}
$$

In addition, each job has release time and deadline constraints. These constraints are called absolute *standard constraints. A job τ_a^j has the following absolute constraints:*

$$c_9 \quad \leq \quad s_a^j \qquad\qquad s_a^j + e_a^j \quad \leq \quad c_{10} \tag{2}$$

We also include as standard any constraint that can be rewritten in one of the above forms; e.g., $s_a^j \geq s_b^l + e_b^l - e_a^j + c$ *falls into this category.*

Next, the *k-fold cyclically constrained job set* is formally defined. [3] Any $\Gamma^{1,k}$ considered in this paper belongs to this class.

Definition 2 (k-fold Cyclically Constrained Job Set) *A job set* $\Gamma^{1,k} = \Gamma^1 \cup \Gamma^2 \cup \ldots \cup \Gamma^k$ *(*$k = 1, 2, \ldots, \infty$*) is classified as a* k-fold cyclically constrained job set *if it has the following linear constraints:*

1. *The set of standard* relative *constraints:*

$$\forall j \in [1, k) \ :: \ A_1 \mathbf{x}^j + A_2 \mathbf{x}^{j+1} \leq \mathbf{a} \tag{3}$$

where $\mathbf{x}^j$ *is a* $2N$*-dimensional column vector* $[s_1^j, e_1^j, s_2^j, e_2^j, \ldots, s_N^j, e_N^j]^T$. A_1, A_2 *are* $m_1 \times 2N$ *(*$m_1 \geq 0$*) matrices of* 0, 1, *or* -1, *and* $\mathbf{a}$ *is an* m_1*-dimensional column vector whose elements are integers. Included in the* m_1 *constraints are those denoting the total ordering on jobs:*

$$\forall j \in [1, k] \ :: \ \forall i \in [1, N) \ :: \ s_i^j + e_i^j \leq s_{i+1}^j$$

$$\forall j \in [1, k) \ :: \ s_N^j + e_N^j \leq s_1^{j+1}$$

2. *The set of release time and deadline constraints:*

$$\forall j \in [1, k] \ :: \ B\mathbf{x}^j \leq \mathbf{b}^j \tag{4}$$

$$\forall j \in [1, k] \ :: \ D\mathbf{x}^j \leq \mathbf{d}^j \tag{5}$$

where $\mathbf{b}^j$ *is an* m_2*-dimensional column vector of non-positive integers satisfying:*

$$\mathbf{b}^j = \mathbf{b}^1 + (1 - j)L$$

and $\mathbf{d}^j$ *is an* m_3*-dimensional column vector of non-negative integers satisfying:*

$$\mathbf{d}^j = \mathbf{d}^1 + (j - 1)L$$

[3] Note that k may be equal to ∞.

We define $\mathcal{C}^{1,k}$ to represent the logical conjunction of the constraints induced by each row of (3), (4), and (5).

In the above definition, the same matrices A_1, A_2, B, D are cyclically used to represent the standard constraints on consecutive job sets.

The example job set shown in Figure 1 is presented here with corresponding matrices and vectors defined in (3), (4), and (5).

Example 1 Consider the example job set depicted in Figure 1. Each job set Γ^j, $1 \leq j \leq k$, consists of two jobs, τ_1^j and τ_2^j (i.e. $N = 2$), whose execution time bounds are:

$$\begin{array}{rclcrcl} l_1^j & = & 5 & & u_1^j & = & 8 \\ l_2^j & = & 8 & & u_2^j & = & 10 \end{array}$$

The standard relative constraints defined within Γ^j or within Γ^{j+1} are:

$$\begin{array}{rcl@{\qquad\qquad}rcl} 5 & \leq & s_2^j - (s_1^j + e_1^j) & 5 & \leq & s_2^{j+1} - (s_1^{j+1} + e_1^{j+1}) \\ s_1^j + e_1^j & \leq & s_2^j & s_1^{j+1} + e_1^{j+1} & \leq & s_2^{j+1} \end{array} \tag{6}$$

Similarly, the set of standard relative constraints across the boundary of Γ^j and Γ^{j+1} are:

$$\begin{array}{rcl@{\qquad}c@{\qquad}rcl} s_1^j + e_1^j + 15 & \leq & s_1^{j+1} + e_1^{j+1} & & s_2^j + e_2^j + 18 & \leq & s_2^{j+1} + e_2^{j+1} \\ s_1^{j+1} + e_1^{j+1} & \leq & s_1^j + e_1^j + 25 & s_2^j + e_2^j \leq s_1^{j+1} & s_2^{j+1} + e_2^{j+1} & \leq & s_2^j + e_2^j + 22 \end{array} \tag{7}$$

Finally, the absolute constraints on Γ^j and Γ^{j+1} are:

$$\begin{array}{rcl@{\qquad\qquad}rcl} 20(j-1) & \leq & s_1^j & 20j & \leq & s_1^{j+1} \\ 20(j-1) & \leq & s_2^j & 20j & \leq & s_2^{j+1} \\ s_1^j + e_1^j & \leq & 20j & s_1^{j+1} + e_1^{j+1} & \leq & 20(j+1) \\ s_2^j + e_2^j & \leq & 20j & s_2^{j+1} + e_2^{j+1} & \leq & 20(j+1) \end{array} \tag{8}$$

All standard relative constraints can be denoted by the following inequality:

$$\begin{bmatrix} 1 & 1 & -1 & 0 \\ 1 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ -1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & -1 & -1 \end{bmatrix} \begin{bmatrix} s_1^j \\ e_1^j \\ s_2^j \\ e_2^j \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & -1 & 0 \\ 1 & 1 & -1 & 0 \\ -1 & -1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & -1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} s_1^{j+1} \\ e_1^{j+1} \\ s_2^{j+1} \\ e_2^{j+1} \end{bmatrix} \leq \begin{bmatrix} -5 \\ 0 \\ -5 \\ 0 \\ -15 \\ 25 \\ 0 \\ -18 \\ 22 \end{bmatrix}$$

And, the set of absolute constraints is represented by the following inequality:

$$\begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} s_1^j \\ e_1^j \\ s_2^j \\ e_2^j \end{bmatrix} \leq \begin{bmatrix} -20(j-1) \\ -20(j-1) \\ 20j \\ 20j \end{bmatrix}$$

One traditional approach for scheduling with complex timing constraints is a time-based scheduling scheme that assigns static start times to the jobs in the scheduling window such that the relative constraints are satisfied if the static schedule is cyclically repeated at runtime. However, this approach can't be used in the presence of arbitrary relative constraints between start or finish times of jobs [4]. Also, this approach suffers from the loss of schedulability problem. Some task sets are not schedulable in this approach, even though they are schedulable if our approach is employed. This will be explained through an example later. To cope with some of the above limitations the parametric scheduling scheme was developed in scope of real-time transaction scheduling [4]. However, as far as we know, the solution approach has not been found for general periodic task models where jobs in different scheduling windows may have relative constraints. The objective of this paper is to develop a schedulability test for $\Gamma^{1,\infty}$, and to develop a flexible job dispatching mechanism for schedulable job sets, $\Gamma^{1,\infty}$.

3 Prior Work

In this section, we briefly describe two scheduling schemes closely related to ours. The first one is the static cyclic scheduling scheme [2] and the second one is the parametric scheduling scheme [4].

3.1 Static Cyclic Scheduling

The static cyclic scheduling problem has been studied in [2]. The periodic task model is used, which means that every job has a release time and a deadline constraints, and only the jitter constraints between two job start times are allowed. An important assumption made in the work is that the start times of jobs in Γ^j are statically determined as offsets from the start of the j-th scheduling window $[(j-1)L, jL]$, and this schedule is invoked repeatedly by wrapping around the end point of the current schedule to the start point of the next. In other words, $s_i^{j+1} = s_i^j + L$ holds for all $1 \leq j$.

In the presence of jitter constraints, the job start times should be chosen carefully such that the jitter constraints are satisfied at run-time as well as the absolute constraints. Obtaining the ordering and job start times is an NP-hard problem, since non-preemptive scheduling problem with release time and deadline constraints is NP-hard. Several priority based non-preemptive scheduling algorithms are presented and their performances are compared in [2].

Suppose that a job $\tau_{i_1}^j$ belongs to Γ^j, and a job $\tau_{i_2}^{j+1}$ belongs to Γ^{j+1}, and they have jitter constraints $c_1 \leq s_{i_2}^{j+1} - s_{i_1}^j \leq c_2$ $(0 < c_1 \leq c_2 \leq L)$. From the above assumption, $s_{i_2}^{j+1} = L + s_{i_2}^j$ holds. Thus, a new constraint is created, $c_1 - L \leq s_{i_2}^j - s_{i_1}^j \leq c_2 - L$, which is again equal to $L - c_2 \leq s_{i_1}^j - s_{i_2}^j \leq L - c_1$. Therefore, the jitter constraints across the boundary of Γ^j and Γ^{j+1} are transformed into jitter constraints between two jobs in Γ^j. As a consequence, if we can find a static schedule for Γ^j that satisfy the above transformed constraints and the constraints between jobs within Γ^j, it is clear that all timing constraints will be satisfied if that schedule is repeatedly used at run-time. This approach is depicted in figure 2.

However, this approach suffers from the following limitations:

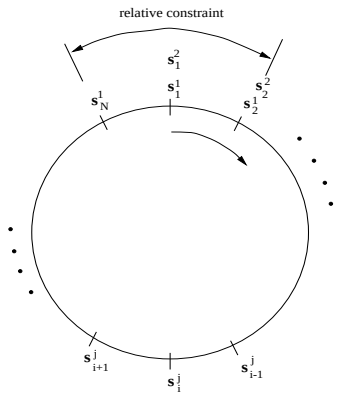

Figure 2: Static Cyclic Scheduling

- The relative constraints allowed are limited to jitter type constraints between start times of two jobs.
- The schedulability of job sets are reduced due to the static start time assignments.
- It is very difficult to effectively incorporate dynamic tasks, such as aperiodic tasks, into a schedule by dynamically adjusting the start times of the jobs.

In some real-time applications, the jitter constraints may be imposed between the finish times of the jobs rather than between the start times [5]. Furthermore, a periodic task may be decomposed into several subtasks and any kind of standard constraints may be defined between these subtasks [4]. In these cases this static scheduling approach is no more applicable without sacrificing the schedulability [4].

By transforming the jitter constraints across the boundary of Γ^j and Γ^{j+1} into those between jobs within Γ^j, we are affecting the schedulability of job sets. We will show that, under our new scheduling scheme in which this transformation is not necessary, the schedulability of job sets is increased, i.e., some job sets are not schedulable according to this scheme whereas it is schedulable by our scheme.

3.2 Parametric Scheduling

Gerber *et al.* [4] proposes a parametric scheduling scheme in the scope of transaction scheduling, in which any standard constraints may be given between jobs in one transaction. Let $\Pi =< \tau_1, \ldots, \tau_N >$ denote a sequence of jobs constituting one transaction with a set of standard constraints, $\mathcal{C}$. Then, a schedulability of Π is defined as follows:

$$Sched \equiv \exists s_1 :: \forall e_1 \in [l_1, u_1] :: \ldots :: \exists s_N :: \forall e_N \in [l_N, u_N] :: \mathcal{C} \tag{9}$$

From this *Sched* predicate, parametric lower and upper bound functions for each start time s_i are obtained by eliminating the variables in an order e_N, s_N, ..., e_i. The parametric lower and upper bound functions, denoted as $\mathcal{F}_{s_i}^{min}$ and $\mathcal{F}_{s_i}^{max}$, are parameterized in terms of the runtime variables, $s_1, e_1, \ldots, s_{i-1}, e_{i-1}$ of already executed jobs. The parametric calendar structure is shown in figure 3.

$\mathcal{F}^{min}_{s_1}()$	$\leq$	s_1	$\leq$	$\mathcal{F}^{max}_{s_1}()$
$\mathcal{F}^{min}_{s_2}(s_1, e_1)$	$\leq$	s_2	$\leq$	$\mathcal{F}^{max}_{s_2}(s_1, e_1)$
	$\vdots$		$\vdots$	
$\mathcal{F}^{min}_{s_N}(s_1, e_1, s_2, e_2, \ldots, s_{N-1}, e_{N-1})$	$\leq$	s_N	$\leq$	$\mathcal{F}^{max}_{s_N}(s_1, e_1, s_2, e_2, \ldots, s_{N-1}, e_{N-1})$

Figure 3: Parametric Calendar Structure

This parametric calendar is obtained from an off-line component of the algorithm by applying variable elimination techniques that will be given later in this section, and the actual bounds of s_i are found at runtime by evaluating the parametric functions in the parametric calendar by using the start times and the finish times of already executed jobs, τ_1, ..., τ_{i-1}. The actual form of these parametric functions are given in the following proposition.

Proposition 1 (Parametric Bound Functions [4]) *A parametric lower bound function for s_j is of the following form:*

$$\begin{aligned} & \mathcal{F}^{min}_{s_j}(s_1, f_1, \ldots, s_{j-1}, f_{j-1}) \\ = \quad & \max(p_1 + c_1, p_2 + c_2, \ldots, p_a + c_a, \alpha^{min}_j) \end{aligned} \tag{10}$$

where each p_i, $1 \leq i \leq a$, belongs to $\{s_1, f_1, \ldots, s_{j-1}, f_{j-1}\}$, and c_i is an arbitrary constant.[4] *And, α^{max}_j is a non-negative integer.*

Similarly, a parametric upper bound function for s_j is of the following form:

$$\begin{aligned} & \mathcal{F}^{max}_{s_j}(s_1, f_1, \ldots, s_{j-1}, f_{j-1}) \\ = \quad & \min(q_1 + d_1, q_2 + d_2, \ldots, q_b + d_b, \alpha^{max}_j) \end{aligned} \tag{11}$$

where each q_i, $1 \leq i \leq b$, belongs to $\{s_1, f_1, \ldots, s_{j-1}, f_{j-1}\}$, and d_i is an arbitrary constant..

The main result obtained by the paper is that, for an arbitrary set of standard constraints on $\Pi = \{\tau_1, \ldots, \tau_N\}$, we can find the parametric calendar in $O(N^3)$ time and the run-time evaluation of each bound function can be carried out in $O(N)$ time.

By applying this parametric scheduling scheme, we are not only able to schedule any sequence of jobs with standard constraints, but also able to take advantage of the flexibility offered by the scheme. That is, the job start times may be decided dynamically at runtime to incorporate other dynamic activities in the system. Even though this scheme is directly applicable to our k-fold cyclically constrained job sets, if the number of jobs in $\Gamma^{1,k}$ becomes large, the bounds need to be found on the size of parametric functions and for the memory requirements for them.

In the rest of this section, the parametric scheduling scheme in the paper is presented with an example.

[4] Note that $f_i = s_i + e_i$.

3.2.1 Elimination of Quantified Variables

Consider a set of linear constraints C in n variables $(x_1, x_2, \ldots, x_n)$,

$$C \equiv H\mathbf{x} \leq \mathbf{h}$$

which must be satisfied with respect to some defined existential and universal quantification over the variables. In this section we show how an innermost universally quantified variable x_n, with associated lower (l_n) and upper (u_n) bounds can be eliminated to obtain a new set of equivalent constraints. The set of constraints C may be partitioned into three subsets, depending on whether the coefficient of x_n is positive, negative or zero. Thus,

$$C \equiv C_P \wedge \ C_N \wedge C_Z$$

where

$$\begin{aligned} C_P &\equiv \{x_n \geq D_i(\mathbf{x}'),\ 1 \leq i \leq p\} \\ C_N &\equiv \{x_n \leq E_j(\mathbf{x}'),\ 1 \leq j \leq q\} \\ C_Z &\equiv \{0 \leq F_k(\mathbf{x}'),\ 1 \leq k \leq r\} \end{aligned}$$

$D_i(\mathbf{x}'), E_j(\mathbf{x}'), F_k(\mathbf{x}')$ are linear functions of $\mathbf{x}' = [x_1, \cdots, x_{n-1}]^T$. The elimination of variable x_n leads to a new system of constraints C' obtained from C by substituting x_n with l_n or u_n, depending on its coefficient:

$$C' \equiv (C_P)_{l_n}^{x_n} \wedge \ (C_N)_{u_n}^{x_n} \wedge \ (C_Z)$$

Lemma 1 ([4]) *Let* C *be a system of linear constraints and let* C′ *be the resulting set of constraints after eliminating a universally quantified variable x_n with lower bound l_n and upper bound u_n. Then the sentence* $\forall x_n \in [l_n, u_n] ::$ C *holds if and only if* C′ *holds.*

The existential quantifier can be eliminated by using Fourier-Motzkin variable elimination technique [3].

Fourier-Motzkin Elimination. Consider a system of linear constraints C in n variables $(x_1, x_2, \ldots, x_n)$. We wish to find a system of linear constraints C' over $\mathbf{x}' = [x_1, \ldots, x_{n-1}]^T$, such that $\mathbf{x}'$ is a solution to C' if and only if $\mathbf{x}'$ is a solution to $\exists x_n :: \ C$. As before, the constraints in C may be partitioned into three subsets.

$$C \equiv \begin{cases} x_n & \geq & D_i(\mathbf{x}'), & 1 \leq i \leq p \\ x_n & \leq & E_j(\mathbf{x}'), & 1 \leq j \leq q \\ 0 & \leq & F_k(\mathbf{x}'), & 1 \leq k \leq r \end{cases}$$

The elimination of variable x_n leads to a new system of constraints:

$$C' \equiv \exists x_n :: \ C \equiv \begin{cases} D_i(\mathbf{x}') & \leq & E_j(\mathbf{x}'), & 1 \leq i \leq p, \quad 1 \leq j \leq q \\ 0 & \leq & F_k(\mathbf{x}'), & 1 \leq k \leq r \end{cases}$$

The correctness of this procedure is stated in the following lemma.

Lemma 2 ([4]) *Let* C *be a set of linear constraints. Let* C$'$ *represent the set of constraints as a result of eliminating* x_n *using Fourier Motzkin elimination as described above. Then,*

$$\exists x_n :: \text{C}$$

holds if and only if C$'$ *holds.*

3.2.2 Example

Here, the variable elimination technique is applied to

$$\exists s_1 :: \forall e_1 \in [5,8] :: \exists s_2 :: \forall e_2 \in [8,10] :: \exists s_3 :: \forall e_3 \in [5,8] :: \exists s_4 :: \forall e_4 \in [8,10] :: \mathcal{C}^{1,2}$$

where $\mathcal{C}^{1,2}$ is a constraint set given on $\Gamma_{1,2}$ in Example 1. Initially, since e_4 is the innermost universally quantified variable, it can be eliminated first. The constraints involving e_4 in $\mathcal{C}^{1,2}$ are:

$$\begin{array}{lcl} s_4 + e_4 & \leq & 40 \\ s_4 + e_4 - (s_2 + e_2) & \leq & 22 \\ 18 & \leq & s_4 + e_4 - (s_2 + e_2) \end{array}$$

The elimination of e_4 from these constraints results in the following derived constraints:

$$\begin{array}{lcll} s_4 & \leq & 30 & (e_4 := u_4 = 10) \\ s_4 - (s_2 + e_2) & \leq & 12 & (e_4 := u_4 = 10) \\ 10 & \leq & s_4 - (s_2 + e_2) & (e_4 := l_4 = 8) \end{array}$$

Therefore, these three constraints are substituted for the original constraints containing e_4. Thus, the complete set of constraints is given below:

$$\begin{array}{rcl} 0 & \leq & s_1 \\ s_2 + e_2 & \leq & 20 \\ 20 & \leq & s_3 \\ s_4 & \leq & 30 \\ s_1 + e_1 & \leq & s_2 \\ s_2 + e_2 & \leq & s_3 \\ s_3 + e_3 & \leq & s_4 \end{array} \qquad \begin{array}{rcl} s_2 - (s_1 + e_1) & \leq & 5 \\ 15 & \leq & s_3 + e_3 - (s_1 + e_1) \\ s_3 + e_3 - (s_1 + e_1) & \leq & 25 \\ 10 & \leq & s_4 - (s_2 + e_2) \\ s_4 - (s_2 + e_2) & \leq & 12 \\ s_4 - (s_3 + e_3) & \leq & 5 \end{array} \tag{12}$$

Next, an existentially quantified variable s_4, which is the innermost one, is eliminated. The constraints containing s_4 in the above constraint set are:

$$\begin{array}{rcl} s_2 + e_2 + 10 & \leq & s_4 \\ s_3 + e_3 & \leq & s_4 \end{array} \qquad \begin{array}{rcl} s_4 & \leq & s_2 + e_2 + 12 \\ s_4 & \leq & s_3 + e_3 + 5 \\ s_4 & \leq & 30 \end{array} \tag{13}$$

From these constraints, the parametric lower and upper bound functions are obtained as follows:

$$\begin{array}{rcl} \mathcal{F}_4^{min}(s_1, e_1, s_2, e_2, s_3, e_3) & = & \max(s_3 + e_3, s_2 + e_2 + 10) \\ \mathcal{F}_4^{max}(s_1, e_1, s_2, e_2, s_3, e_3) & = & \min(s_2 + e_2 + 12, s_3 + e_3 + 5, 30) \end{array}$$

And, as a result of eliminating s_4, the constraints in (13) are replaced by the following constraints:

$$\begin{array}{rcl} 10 & \leq & 12 \\ s_2+e_2+10 & \leq & s_3+e_3+5 \\ s_2+e_2+10 & \leq & 30 \\ s_3+e_3 & \leq & s_2+e_2+12 \\ 0 & \leq & 5 \\ s_3+e_3 & \leq & 30 \end{array} \quad \rightsquigarrow \quad \begin{array}{rcl} s_2+e_2 & \leq & 20 \\ s_2+e_2+5 & \leq & s_3+e_3 \\ s_3+e_3 & \leq & s_2+e_2+12 \\ s_3+e_3 & \leq & 30 \end{array} \tag{14}$$

If we continue this process until s_1 is eliminated, then we will obtain all the parametric bound functions, or the predicate will turn out to be false during the process. Figure 4 shows the obtained parametric bound functions.

0	$\leq$	s_1	$\leq$	2
$\max(8, s_1+e_1)$	$\leq$	s_2	$\leq$	$\min(10, s_1+e_1+5)$
$\max(20, s_1+e_1+10, s_2+e_2)$	$\leq$	s_3	$\leq$	$\min(22, s_1+e_1+17, s_2+e_2+4)$
$\max(s_3+e_3, s_2+e_2+10)$	$\leq$	s_4	$\leq$	$\min(30, s_2+e_2+12, s_3+e_3+5)$

Figure 4: Parametric Calendar for Example

4 Dynamic Cyclic Dispatching

As in the parametric scheduling approach developed for transaction scheduling [4], we want to devise a schedulability test and an efficient dispatching mechanism when an ∞-fold cyclically constrained job set, $\Gamma^{1,\infty}$, is given with its constraint matrices and vectors. We say $\Gamma^{1,k}$, is *schedulable* if there exists any method which can successfully dispatch the jobs in $\Gamma^{1,k}$.

Definition 3 (Schedulability of $\Gamma^{1,k}$) *The k-fold cyclically constrained job set $\Gamma^{1,k}$ $(1 \leq k)$ is schedulable if the following predicate holds:*

$$\begin{aligned} sched^{1,k} \quad \equiv \quad & \exists s_1^1 :: \forall e_1^1 \in [l_1^1, u_1^1] :: \exists s_2^1 :: \forall e_2^1 \in [l_2^1, u_2^1] :: \ \ldots \\ & \exists s_N^k :: \forall e_N^k \in [l_N^k, u_N^k] :: \mathcal{C}^{1,k} \end{aligned} \tag{15}$$

where $\mathcal{C}^{1,k}$ is a set of standard constraints defined on $\{s_1^1, e_1^1, \ldots, s_N^k, e_N^k\}$.

Then, the following proposition holds for all $k \geq 1$.

Proposition 2

$$\forall k \geq 1 \ :: \ sched^{1,k+1} \Longrightarrow sched^{1,k}$$

Proof: Obvious from the definition of a cyclically constrained job set and from the definition of $sched^{1,k}$ in (15). ■

Hence, once $sched^{1,k}$ turns out to be **False**, then all $sched^{1,j}$, $k \leq j$, are **False**, too. By this proposition, the schedulability of $\Gamma^{1,\infty}$ is defined.

Definition 4 (Schedulability of $\Gamma^{1,\infty}$) *$\Gamma^{1,\infty}$ is schedulable if and only if*

$$\lim_{k\to\infty} sched^{1,k} = \textbf{True}$$

In [4], it is shown that checking Predicate (9) is not trivial because of the nondeterministic job execution times and because of the existence of standard relative constraints among the jobs. This applies to the above $sched^{1,k}$ predicate, too. The variable elimination techniques are used in [4] to eliminate variables from Predicate (9). At the end of the variable elimination process parametric bound functions for s_i, that are parameterized in terms of the variables in $\{s_1, e_1, \ldots, e_{i-1}\}$, are found as well as the predicate value.

However, if we want to apply the variable elimination technique to $sched^{1,k}$, the following problems have to be addressed first:

1. On which subset of $\{s_1^1, e_1^1, \ldots, s_{i-1}^j, e_{i-1}^j\}$ does the parametric bound functions for s_i^j depend?
2. Is it required to store parametric bound functions for every job in $\Gamma^{1,k}$?
3. What parametric bound functions have to be used if k is not known at pre-runtime and dynamically decided at runtime?

Let $\mathcal{F}_{s_i^j}^{min,k}$ and $\mathcal{F}_{s_i^j}^{max,k}$ denote parametric lower and upper bound functions for s_i^j, respectively, that are found after the variable elimination algorithms are applied to $sched^{1,k}$. If the number of variables is unbounded with which $\mathcal{F}_{s_i^j}^{min,k}$ or $\mathcal{F}_{s_i^j}^{max,k}$ is parameterized, then it is not possible to evaluate them at run-time within bounded computation times. Also, if it is required that parametric bound functions for every job in $\Gamma^{1,k}$ be stored at runtime, the scheme is not implementable for large k because of memory requirements. Finally, if the value of k is not known at pre-runtime and is decided dynamically at runtime, which is true in most real-time applications, parametric bound functions to be used have to be selected.

In this section, the answers to the above questions are sought by first transforming $sched^{1,k}$ into a constraint graph and by investigating the properties of such graphs. In section 4.1 the transformation rule is presented with lemmas showing the equivalence relationship between $sched^{1,k}$ and its constraint graph with respect to variable elimination process. In section 4.2 several terminologies are defined for constraint graphs, and in section 4.3 the properties of constraint graphs are investigated. Then, in section 4.4 a complete off-line algorithm is presented to check $sched^{1,\infty}$ and to obtain parametric bound functions for job start times if it is schedulable. In addition, for a certain class of standard constraints, it is shown in section 4.5 that the off-line algorithm can be executed within $O(N^3 + n^5)$ time by pre-eliminating certain nodes from the constraint graph.

4.1 Transforming a Constraint Set into a Constraint Graph

Let $\Pi = \{\tau_1, \tau_2, \ldots, \tau_N\}$ be a finite set of jobs with a set of standard constraints, $\mathcal{C}$. Consider eliminating quantified variables from the following predicate:

$$Sched \equiv \exists s_1 :: \forall e_1 \in [l_1, u_1] :: \ldots \exists s_N :: \forall e_N \in [l_N, u_N] :: \mathcal{C}$$

Then, predicates on subsets of $\{s_1, e_1, \ldots, s_N, e_N\}$ are defined next that are found after eliminating variables.

Definition 5 *$Sched(s_a)(1 \leq a \leq N)$ is defined to be a predicate on a set of variables $\{s_1, e_1, \ldots, s_a\}$ that are found after eliminating variables of $< f_N, s_N, \ldots, f_a >$ from Sched. $Sched(e_a)$ is defined similarly.*

That is, $Sched(s_a)$ can be expressed as

$$Sched(s_a) \equiv \exists s_1 :: \forall e_1 \in [l_1, u_1] :: \ldots \exists s_a :: \mathcal{C}(s_a)$$

It will be shown that $Sched$(or $Sched(s_a)$, or $Sched(e_a)$) can be transformed into a directed graph, which is called a *constraint graph*, such that the variable elimination process can be mapped into a corresponding node elimination operation in the graph. Note that, in the following definition of a constraint graph, *semi-exclusive-ORed* edges are defined, which will be used in defining *restricted paths* in constraint graphs. Also, $v_1 \xrightarrow{w} v_2$ denotes an edge from a node v_1 to a node v_2 with a weight w, and $< v_1 \xrightarrow{w_1} v_2 \xrightarrow{w_2} \ldots \xrightarrow{w_{i-1}} v_i >$ denotes a path from a node v_1 to a node v_i with a weight sum $w = \sum_{j=1}^{i-1} w_j$. If no confusion is caused, a brief notation, $v_1 \overset{w}{\rightsquigarrow} v_i$, will be used to denote such a path.

Definition 6 (Constraint Graph) *A constraint graph $G(V, E)$ is found from Sched (or $Sched(s_a)$, or $Sched(e_a)$) as follows:*

1. *node set V is obtained as follows:*
   - $v_0 \in V$
   - $s_i,\ f_i \in V$ *for* $1 \leq i \leq N$ *where* $f_i = s_i + e_i$.
2. *edge set E is obtained as follows:*
   - *For each tuple $< s_i, f_i >$, add the following* semi-exclusive-ORed *edges to E:*
     (a) $s_i \xrightarrow{l_i} f_i$
     (b) $f_i \xrightarrow{-u_i} s_i$
   - *For each constraint in $\mathcal{C}$ that can be converted to:*
     (a) $v_i - v_j \leq c$ $(v_i, v_j \in \{s_i, f_i \mid 1 \leq i \leq N\})$: *add* $v_j \xrightarrow{c} v_i$ *to E.*
     (b) $v_i \leq c$: *add* $v_0 \xrightarrow{c} v_i$ *to E.*
     (c) $-v_i \leq c$: *add* $v_i \xrightarrow{c} v_0$ *to E.*

Definition 7 *The constraint graph found from $Sched(s_a)$ is denoted as $G(s_a)$.*[5] *Similarly, $G(f_a)$ represents a graph found from $Sched(e_a)$.*

Figure 5 shows a graph created from the example job set $\Gamma^{1,2}$ defined in Example 1. Note that v_0 is an extra node created to represent a constant 0 that is used to specify absolute constraints such as the release time and the deadline constraints. In the figure, the edges connected by $\oplus$ are semi-exclusive-ORed edges.

[5] The full notation would be $G(s_a)(V, E)$. But, if no confusion is caused, $G(s_a)$ will be used in this paper.

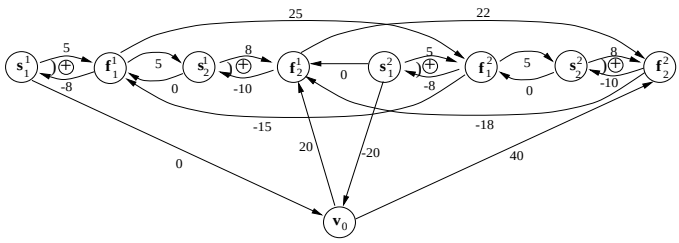

Figure 5: Constraint Graph for $\Gamma^{1,2}$

Note that there may exist only one edge from one node to another from the uniqueness of inequality in the constraint set. For example, if there are two constraints $v_1 - v_2 \leq c_1$ and $v_1 - v_2 \leq c_2$ in $\mathcal{C}$, then one of them is redundant. Therefore, we can denote an edge from v_1 to v_2 in a constraint graph as $v_1 \rightarrow v_2$ without its weight specified. Also, note that any edge from f_i to s_i is semi-exclusive-ORed to any edge from s_i to f_i. That is, even if any of these two edges is created from another constraint in $\mathcal{C}$ rather than from the minimum or maximum execution time constraint, they are semi-exclusive-ORed.

Definition 8 (Restricted Path) *In a constraint graph, a path,* $< v_1 \xrightarrow{w_1} v_2 \xrightarrow{w_2} \ldots v_{i-1} \xrightarrow{w_{i-1}} v_i >$, *is called a* restricted path *from* v_1 *to* v_i *if the following is satisfied:*

- *If* $f_j \rightarrow s_j$ *appears in the path, then its semi-exclusive-ORed edge* $s_j \rightarrow f_j$ *may appear at most once in the path, and vice versa.*
- *If two semi-exclusive-ORed edges,* $f_j \rightarrow s_j$ *and* $s_j \rightarrow f_j$, *appear in the path, then they belong to a sub-path* $< f_j \rightarrow s_j \rightarrow f_j >$.

Note that if a sub-path $< f_j \rightarrow s_j \rightarrow f_j >$ appears once in the path, then neither $f_j \rightarrow s_j$ nor $s_j \rightarrow f_j$ should appear at another place in the path, and vice versa.

Definition 9 (Restricted Cycle) *A* restricted cycle *in a constraint graph is defined to be a cycle*[6] *such that*

1. *it satisfies the definition of a restricted path.*
2. *it is not a sub-path of* $< f_j \rightarrow s_j \rightarrow f_j >$ *for any* $1 \leq j \leq N$.

For example, a path $< f_j \rightarrow s_j \rightarrow f_j \rightarrow s_l \rightarrow f_j >$ is a restricted cycle while a path $< f_j \rightarrow s_j \rightarrow f_j >$ is not. Also, a restricted path without any restricted cycle in it is called an *acyclic restricted path.*

The elimination algorithm of a node f_a from a graph $G(f_a)$ is presented next.

Algorithm 1 (Elimination of f_a from a Graph $G(f_a)$) *Elimination of* f_a *from* $G(f_a)$ *is performed by the following algorithm.*

1. *For each edge pair,* $< y \xrightarrow{w_1} f_a, f_a \xrightarrow{w_2} s_a >$, *that are not semi-exclusive-ORed in* $G(f_a)$:

[6] A cycle is defined to be a path $< y \rightarrow v_1 \ldots \rightarrow v_i \rightarrow y >$ where $i \geq 1$, or to be a path $< y \rightarrow y >$.

- *create an edge* $y \xrightarrow{w_1+w_2} s_a$.
  - (a) *If* $y = s_a$ *and* $w_1 + w_2 < 0$, *then return* **False**.[7]
  - (b) *If* $y = s_a$ *and* $w_1 + w_2 \geq 0$, *then remove this edge.*[8]
  - (c) *If there already exists an edge* $y \xrightarrow{w'} s_a$ *before creating* $y \xrightarrow{w_1+w_2} s_a$, *then the edge with less weight remains, while the other is removed.*

2. *For each edge pair,* $< s_a \xrightarrow{w_1} f_a, f_a \xrightarrow{w_2} z >$, $z \neq s_a$, *that are not semi-exclusive-ORed in* $G(f_a)$:

   - *create an edge* $s_a \xrightarrow{w_1+w_2} z$.
     - (a) *If there already exists an edge* $s_a \xrightarrow{w''} z$ *before creating* $s_a \xrightarrow{w_1+w_2} z$, *then the edge with less weight remains, while the other is removed.*

3. *Set* $V = V - \{f_a\}$ *and remove all edges to or from* f_a *in* $G(f_a)$.

Let $Elim(G(f_a), f_a)$ denote a new graph created after eliminating f_a from the graph $G(f_a)$ according to Algorithm 1 in case **False** is not found. In this case, the following lemma proves the equivalence, with regards to the graph transformation rule, between the elimination of an universal quantifier from the predicate and the elimination of a node, f_a, from the constraint graph.

Lemma 3 *$Elim(G(f_a), f_a)$ is equal to $G(s_a)$.*

Proof: Given in appendix.

Next, we show how a node corresponding to an existential quantifier s_a may be eliminated from the graph $G(s_a)$.

Algorithm 2 (Elimination of s_a from a Graph $G(s_a)$) *Elimination of s_a from $G(s_a)$ is performed by the following algorithm.*

1. *For each edge pair,* $< y \xrightarrow{w_1} s_a, s_a \xrightarrow{w_2} z >$, *in* $G(s_a)$:

   - *create an edge* $y \xrightarrow{w_1+w_2} z$.
     - (a) *If* $y = z$ *and* $w_1 + w_2 < 0$, *then return* **False**.
     - (b) *If* $y = z$ *and* $w_1 + w_2 \geq 0$, *then remove this edge.*
     - (c) *If there already exists an edge* $y \xrightarrow{w'} z$ *before creating* $y \xrightarrow{w_1+w_2} z$, *then the edge with less weight remains, while the other is removed.*

2. *Set* $V = V - \{s_a\}$ *and remove all edges to or from* s_a *in* $G(s_a)$.

[7] This is because $y - y = 0 \leq w_1 + w_2 < 0$ is a contradiction.
[8] This is because $y - y = 0 \leq w_1 + w_2$ is a tautology.

Again, let $Elim(G(s_a), s_a)$ denote a new graph created after eliminating s_a from the graph $G(s_a)$ according to Algorithm 2 in case **False** is not found. Then, the following lemma shows the equivalence between the elimination of a node in the graph and the elimination of an existential quantifier from the constraint set.

Lemma 4 *$Elim(G(s_a), s_a)$ is equal to $G(f_{a-1})$.*

Proof: Given in appendix.

By inductively applying Lemma 3 and 4, the equivalence relationship between node elimination and variable elimination processes can be established. This relationship is shown in Figure 6 with respect to the constraint graph derivation rules.

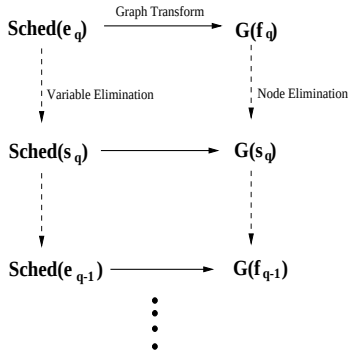

Figure 6: Equivalence between Predicates and Graphs

The elimination process of nodes, f_a and s_a, from the graph $G(f_a)$ can be viewed as preserving the connectivity between any two nodes in $\{v_0, s_1, f_1, \ldots, s_{a-1}, f_{a-1}\}$ through f_a and s_a in $G(f_a)$. That is, if there exists any restricted path from y to z only through s_a and f_a in $G(f_a)$, then a new edge from y to z is created to maintain the connectivity from y to z even after f_a and s_a are eliminated. This is formally proved in Lemma 5.

Figure 7 shows a graph and its node elimination processes for $sched^{1,2}$ that is derived from $\Gamma^{1,2}$ in Example 1.

The following proposition describes a necessary condition for *Sched* to be true in terms of its constraint graph.

Proposition 3 *If a constraint graph for Sched has a negative weight restricted cycle, then Sched* = **False**.

Proof: Given in appendix.

The following lemma shows how the connectivity is maintained during the node elimination process, which is quite an useful property that will be frequently used throughout the paper.

Lemma 5 *Let $\{v_0, s_1, f_1, \ldots, s_a, f_a\}$, $1 \leq a \leq N$, denote a node set of $G(f_a)$ that is found after eliminating nodes of $< f_N, s_N, \ldots, f_{a+1}, s_{a+1} >$ from $G(f_N)$. Also, assume that no contradiction has been found yet. Then, the following two conditions are equivalent:*

1. $y \xrightarrow{W} z \in G(f_a)$

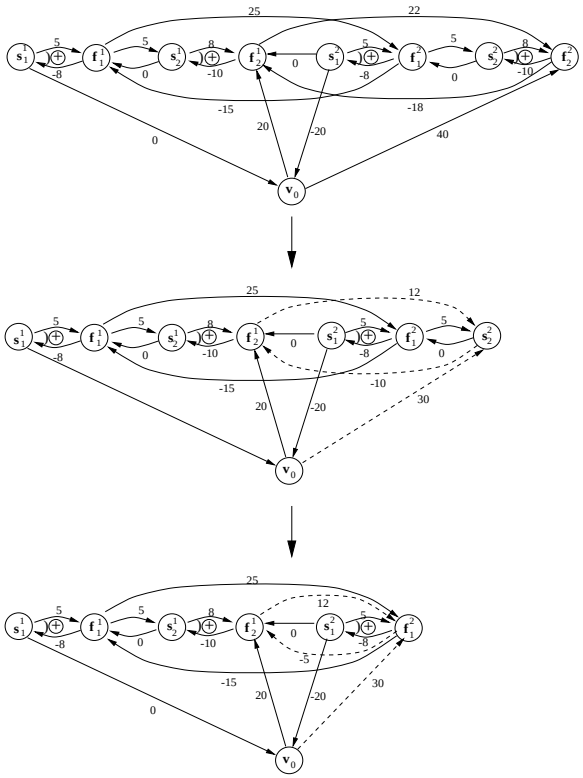

Figure 7: Elimination of f_2^2 and s_2^2 from $\Gamma^{1,2}$

2. *there exists a minimum weight acyclic restricted path* $y \overset{W}{\rightsquigarrow} z$ *in* $G(f_N)$ *where all intermediate*[9] *nodes of the path belong to* $\{s_{a+1}, f_{a+1}, \ldots, s_N, f_N\}$.[10]

Proof: Given in appendix.

In the next corollary it is assumed that v and v' denote any two nodes that are located consecutively in a sequence $< v_0, s_1, f_1, \ldots s_N, f_N >$.

Corollary 1 *Let* $\{v_0, s_1, f_1, \ldots, v\}$, *denote a node set of* $G(v)$ *that is found after eliminating nodes of* $< f_N, s_N, \ldots, v' >$ *from* $G(f_N)$. *Also, assume that no contradiction has been found yet. If an edge from* y *to* z *exists in* $G(v)$, *then there exists a path from* y *to* z *in* $G(f_N)$ *whose intermediate nodes belong to* $\{v', \ldots, s_N, f_N\}$.

Proof: Given in appendix.

For example, in the example shown in figure 7, after eliminating $\{f_2^2, s_2^2\}$ an edge $f_2^1 \xrightarrow{12} f_1^2$ is created since, in the initial graph, there exists a minimum weight acyclic restricted path $< f_2^1 \xrightarrow{22} f_2^2 \xrightarrow{-10} s_2^2 \xrightarrow{0} f_1^2 >$ whose weight sum is 12 and whose intermediate nodes belong to $\{s_2^2, f_2^2\}$. Also, an edge $f_2^1 \xrightarrow{2} f_2^1$ is created in $G^{1,2}(f_1^2)$, since

[9] $\{v_1, v_2, \ldots, v_i\}$ is a set of intermediate nodes of a path $< y \rightarrow v_1 \rightarrow v_2 \ldots v_i \rightarrow z >$ where $i \geq 1$, or $\{\}$ is an intermediate node set if the path consists of one edge.

[10] $y \rightarrow z$ may also be considered as a path whose intermediate nodes belong to $\{s_{a+1}, f_{a+1}, \ldots, s_N, f_N\}$.

there exists a minimum weight restricted path $< f_2^1 \stackrel{22}{\longrightarrow} f_2^2 \stackrel{-10}{\longrightarrow} s_2^2 \stackrel{8}{\longrightarrow} f_2^2 \stackrel{-18}{\longrightarrow} f_2^1 >$ without any intermediate restricted cycle.

4.2 Definitions for Constraint Graphs

In this section, we define several terminologies regarding constraint graphs. They will be used in investigating properties of constraint graphs in the next section. In this section, it is assumed that an initial graph is obtained from the predicate $sched^{1,k}$ that is defined in (15) for $\Gamma^{1,k}$.

Before defining terminologies for constraint graphs, the following function is defined on node sets of constraint graphs.

Definition 10 (g_γ) *g_γ is an one-to-one mapping*

$$\{s_i^j, f_i^j \mid 1 \le i \le N \wedge \max(1, -\gamma + 1) \le j\}$$

$$\longrightarrow \{s_i^j, f_i^j \mid 1 \le i \le N \wedge \max(\gamma + 1, 1) \le j\}$$

by the following rule:

$$g_\gamma(v) = \begin{cases} v_0 & \text{if } v = v_0. \\ s_i^{j+\gamma} & \text{if } v = s_i^j \text{ where } 1 \le i \le N. \\ f_i^{j+\gamma} & \text{if } v = f_i^j \text{ where } 1 \le i \le N. \end{cases}$$

$g_\gamma(V)$ on a node set V is defined to be a set of $g_\gamma(v)$ where v is an element of V.

For example, $s_i^{j_1}$ in Γ^{j_1} can be related to a node $s_i^{j_2}$ in Γ^{j_2} by

$$s_i^{j_2} = g_{(j_2 - j_1)}(s_i^{j_1})$$

In this case $s_i^{j_2}$ is called a *corresponding node* of $s_i^{j_1}$ in a job set Γ^{j_2}, and vice versa.

As in Definition 5, $sched^{1,k}(s_i^j)$ $(1 \le i \le N \wedge 1 \le j \le k)$ is defined to be a predicate on a set of variables $\{s_1^1, e_1^1, \ldots, s_i^j\}$ that is obtained after eliminating the variables, e_N^k, s_N^k, ..., e_i^j, from $sched^{1,k}$. That is, it can be expressed as

$$sched^{1,k}(s_i^j) \equiv \exists s_1^1 :: \forall e_1^1 \in [l_1^1, u_1^1] :: \ldots \exists s_i^j :: \mathcal{C}^{1,k}(s_i^j)$$

where $\mathcal{C}^{1,k}(s_i^j)$ is a set of standard constraints obtained after variable elimination. $sched^{1,k}(e_i^j)$ is defined similarly. Also, as in Definition 7, the graphs found from the above predicates are denoted as follows:

- $G^{1,k}(s_i^j)$ denotes a graph constructed from $sched^{1,k}(s_i^j)$.
- $G^{1,k}(f_i^j)$ denotes a graph constructed from $sched^{1,k}(e_i^j)$.

Note that, from $\mathcal{C}^{1,k}(s_i^j)$(or $G^{1,k}(s_i^j)$), we can find out the parametric lower and upper bound functions for s_i^j in the forms presented in Proposition 1.

First, several terms are defined for constraint graphs. Let $\mathcal{E}$ denote a subset of edges in a graph $G^{1,k}(s_i^j)$, (or $G^{1,k}(f_i^j)$) in the following two definitions.

Definition 11 (Node Set from $\mathcal{E}$) *$Node(\mathcal{E})$ denotes a set of nodes that are connected by any edge in $\mathcal{E}$.*

Definition 12 (Preceding Node Set from $\mathcal{E}$) *$PrecNode(\mathcal{E})$ is defined to be a subset of $Node(\mathcal{E})$ in the graph such that $v \in PrecNode(\mathcal{E})$ if and only if*

- *there exists a node v' that lies after v in the sequence $< v_0, s_1^1, f_1^1, \ldots, s_i^j(, f_i^j) >$ satisfying:*

$$v \rightarrow v' \in \mathcal{E} \ \vee \ v' \rightarrow v \in \mathcal{E}$$

In the example constraint graph shown in Figure 5 let $\mathcal{E}$ be $\{f_2^1 \rightarrow f_2^2, s_2^2 \rightarrow f_2^2, v_0 \rightarrow f_2^2\}$. Then, a node set from $\mathcal{E}$, $Node(\mathcal{E})$ is found to be $\{v_0, f_2^1, s_2^2, f_2^2\}$. Also, the preceding node set, $PrecNode(\mathcal{E})$, is $\{v_0, f_2^1, s_2^2\}$.

In the following definition, let $< v_0, s_1^1, f_1^1, \ldots, y, z, \ldots, s_N^k, f_N^k >$ denote a sequence of nodes in the initial graph $G^{1,k}(f_N^k)$, where y, z denote any two consecutive nodes in the sequence.

Definition 13 (Crossing Edge Set over a Node y) *A crossing edge set $\Phi^{1,k}(y)$ is defined to be a set of edges $v_1 \rightarrow v_2$ in $G^{1,k}(f_N^k)$ satisfying either of the following two conditions:*

1. *$v_1 \in < v_0, s_1^1, f_1^1, \ldots, y >$ and $v_2 \in < z, \ldots, s_N^k, f_N^k >$.*
2. *$v_2 \in < v_0, s_1^1, f_1^1, \ldots, y >$ and $v_1 \in < z, \ldots, s_N^k, f_N^k >$.*

For example, in Figure 8, $\Phi^{1,2}(f_2^1)$ is shown with dashed edges. Informally speaking, any edges created in $G^{1,k}(y)$ after eliminating nodes $< z, \ldots, s_N^k, f_N^k >$ may connect only the nodes that belong to $PrecNode(\Phi^{1,k}(y))$. This is proved in Proposition 5.

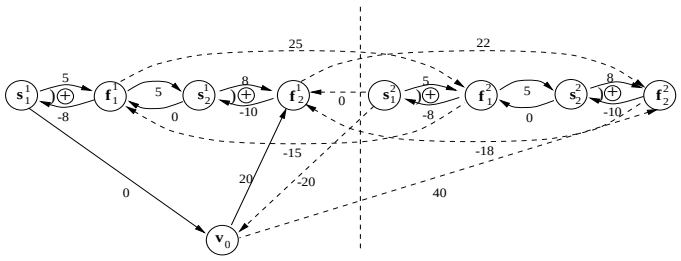

Figure 8: $\Phi^{1,2}(f_2^1)$ is denoted as dashed edges meeting with a vertical line.

Definition 14 (Created Edge Set in $G^{1,k}(f_i^j)$) *A created edge set $\Psi^{1,k}(f_i^j)$, $1 \leq j \leq k-1$, is defined to be a set of edges $v_1 \xrightarrow{w} v_2$ in $G^{1,k}(f_i^j)$ where v_1, v_2 satisfy the following condition:*

- *there exists a path $v_1 \rightsquigarrow v_2$ in $G^{1,k}(f_N^k)$ such that*
  1. *it has at least one intermediate node.*

2. all of its intermediate nodes belong to $\{v_0, s_1^1, f_1^1, \ldots, s_N^k, f_N^k\} - \{v_0, s_1^1, f_1^1, \ldots, f_i^j\}$.

$\Psi^{1,k}(s_i^j)$ *is defined similarly.*

That is, a created edge set in $G^{1,k}(f_i^j)$ contains edges that have possibilities of being newly created during the variable elimination process. Note that, if a newly created edge is implied by an already existing edge in $G^{1,k}(f_N^k)$ with a less weight and thus removed during the elimination process as explained in Algorithm 1 and 2, then the already existing edge is included into the created edge set instead of the removed one that is actually created during the variable elimination process. In figure 9, the constraint graph is shown corresponding to Example 1. Dashed edges are used to represent $\Psi^{1,3}(s_2^3)$ and $\Psi^{1,3}(s_2^2)$.

Next, the semi-homogeneity and homogeneity relationships are defined between two edge sets in two constraint graphs that are found during variable elimination processes from two job sets, $\Gamma^{1,k}$ and $\Gamma^{1,l}(k \leq l)$, respectively.

Definition 15 (Semi-homogeneous Edge Sets) *Let* $\mathcal{E}_1$ *and* $\mathcal{E}_2$ *be subsets of edges in* $G^{1,k}(f_i^{j_1})$ *and* $G^{1,l}(f_i^{j_2})$ *(or,* $G^{1,k}(s_i^{j_1})$ *and* $G^{1,l}(s_i^{j_2})$*), respectively, where* $k \leq l \wedge j_1 \leq k \wedge j_2 \leq l$*. Then,* $\mathcal{E}_1$ *is semi-homogeneous to* $\mathcal{E}_2$ *if and only if*

$$| \mathcal{E}_1 | = | \mathcal{E}_2 | \quad \wedge \quad (v_1 \rightarrow v_2 \in \mathcal{E}_1) \implies (g_{(j_2 - j_1)}(v_1) \rightarrow g_{(j_2 - j_1)}(v_2) \in \mathcal{E}_2)$$

Here, note that, if $\mathcal{E}_1$ is semi-homogeneous to $\mathcal{E}_2$, then

$$(v_3 \rightarrow v_4 \in \mathcal{E}_2) \implies (g_{(j_1 - j_2)}(v_3) \rightarrow g_{(j_1 - j_2)}(v_4) \in \mathcal{E}_1)$$

holds, too, since $| \mathcal{E}_1 | = | \mathcal{E}_2 |$ and $\mathcal{E}_1$ is mapped onto $\mathcal{E}_2$ under the index function $g_{(j_2 - j_1)}$ which is one-to-one.

The homogeneity relationship is defined next which is stronger than semi-homogeneity relationship. Again, let $\mathcal{E}_1$ and $\mathcal{E}_2$ be subsets of edges in $G^{1,k}(f_i^{j_1})$ and $G^{1,l}(f_i^{j_2})$ (or, $G^{1,k}(s_i^{j_1})$ and $G^{1,l}(s_i^{j_2})$), respectively, where $k \leq l \wedge j_1 \leq k \wedge j_2 \leq l$.

Definition 16 (Homogeneous Edge Sets) $\mathcal{E}_1$ *is homogeneous to* $\mathcal{E}_2$*, denoted as* $\mathcal{E}_1 \sim \mathcal{E}_2$*, if and only if*

1. $\mathcal{E}_1$ *and* $\mathcal{E}_2$ *are semi-homogeneous.*
2. *For two nodes* $v_1 (\neq v_0)$*,* $v_2 (\neq v_0)$*,* $(v_1 \xrightarrow{\mathrm{w}} v_2 \in \mathcal{E}_1) \iff (g_{(j_2 - j_1)}(v_1) \xrightarrow{\mathrm{w}} g_{(j_2 - j_1)}(v_2) \in \mathcal{E}_2)$
3. *For two nodes* v_0*,* v_2*, where* $v_2 \neq v_0$*,* $(v_0 \xrightarrow{\mathrm{w}} v_2 \in \mathcal{E}_1) \iff (v_0 \xrightarrow{\mathrm{w+(j_2-j_1)L}} g_{(j_2 - j_1)}(v_2) \in \mathcal{E}_2)$
4. *For two nodes* v_1*,* v_0*, where* $v_1 \neq v_0$*,* $(v_1 \xrightarrow{\mathrm{w}} v_0 \in \mathcal{E}_1) \iff (g_{(j_2 - j_1)}(v_1) \xrightarrow{\mathrm{w-(j_2-j_1)L}} v_0 \in \mathcal{E}_2)$

Homogeneity relations are commutative and transitive, i.e.,

$$\mathcal{E}_1 \sim \mathcal{E}_2 \iff \mathcal{E}_2 \sim \mathcal{E}_1$$

$$(\mathcal{E}_1 \sim \mathcal{E}_2) \wedge (\mathcal{E}_2 \sim \mathcal{E}_3) \implies \mathcal{E}_1 \sim \mathcal{E}_3$$

which can be easily proved from the definition of homogeneity. Two homogeneous created edge sets, $\Psi^{1,3}(s_2^3)$ and $\Psi^{1,3}(s_2^2)$, are shown in Figure 9 with dashed edges where $L = 20$.

A constant, n, is defined next that will be used in obtaining a complexity bound of our algorithm.

Definition 17 (n) $n = \mid PrecNode(\Phi^{1,k}(f_N^1)) \mid, \quad k \geq 2$

Note that $\mid PrecNode(\Phi^{1,k}(f_N^j)) \mid$ is same for all $2 \leq k$ and all $1 \leq j \leq k-1$ from the definition of a cyclically constrained job set and the definition of a preceding node set. $n-1$ is the number of jobs in one scheduling window that have standard relative constraints with jobs in the next scheduling window.

4.3 Characteristics of Constraint Graphs

From now on, the properties of constraint graphs will be examined that remain true during the node elimination process. Note that, from Proposition 3 if a negative weight restricted cycle exists in the constraint graph, Algorithm 1 or 2 will detect it and return **False**. In this case the predicate $sched^{1,k}$ is false and the job set $\Gamma^{1,\infty}$ is not schedulable as well as $\Gamma^{1,k}$. If a constraint graph appears in any of the following propositions, it is assumed that no contradiction has been found in the process of obtaining that graph. First, it is shown that the parametric bound functions for s_i^j found from a constraint graph $G^{1,k}(s_i^j)$ depend on the start or finish times of the jobs in Γ^{j-1} and Γ^j that are already executed. This means that the number of jobs it may actually be dependent on is shown to be bounded by $O(N)$. This bounds the number of variables to $O(N)$ that have to be used in evaluating parametric bound functions at runtime.

Proposition 4 *In a graph $G^{1,k}(s_i^j)$, if s_i^j is connected to a node v, then*

$$v \in PrecNode(\Phi^{1,k}(s_i^j)) \cup P$$

where $P = \{y \mid y \in < v_0, s_1^{j-1}, f_1^{j-1}, \ldots, f_{i-1}^j > \wedge (y \rightarrow s_i^j \in G^{1,k}(f_N^k) \vee s_i^j \rightarrow y \in G^{1,k}(f_N^k))\}$

Proof: Given in appendix.

Similar result holds for a graph $G^{1,k}(f_i^j)$.

Then, the following proposition implies that the set of nodes, to which additionally created edges in $G^{1,k}(s_i^j)$(or $G^{1,k}(f_i^j)$) may be connected, is a subset of $PrecNode(\Phi^{1,k}(s_i^j))$(or $PrecNode(\Phi^{1,k}(f_i^j))$).

Proposition 5

$$Node(\Psi^{1,k}(s_i^j)) \subseteq PrecNode(\Phi^{1,k}(s_i^j))$$

$$Node(\Psi^{1,k}(f_i^j)) \subseteq PrecNode(\Phi^{1,k}(f_i^j))$$

Also, $\mid \Psi^{1,k}(f_N^j) \mid \leq n(n-1)$ *holds.*

Proof: Given in appendix.

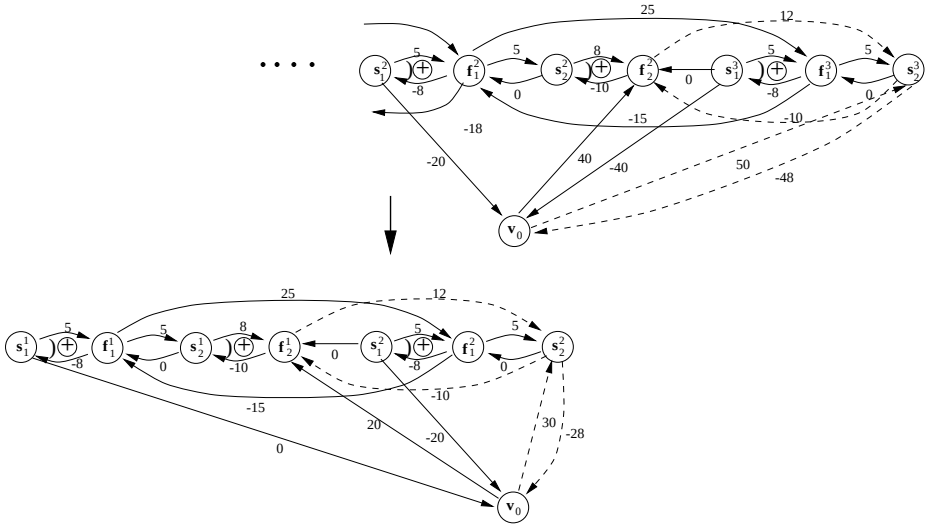

Figure 9: Homogeneous edge sets, $\Psi^{1,3}(s_2^3)$ and $\Psi^{1,3}(s_2^2)$

These two propositions give an upper bound on the actual number of nodes s_i^j may be connected to in $G^{1,k}(s_i^j)$, which is $O(N)$. If only jitter constraints are allowed from periodic tasks, it is easy to see that s_i^j in $G^{1,k}(s_i^j)$ is connected to at most $O(n)$ number of jobs. This is because $\mid PrecNode(\Phi^{1,k}(s_i^j)) \mid \leq n$ and $\mid P \mid \leq 2$.

Then, an interesting property of an additionally created edge set, $\Psi^{1,k}(f_N^j)$, is given in the following proposition. After eliminating $4N$ variables of the last $2N$ jobs (belonging to Γ^k and Γ^{k-1}) from $sched^{1,k}$, we periodically obtain semi-homogeneous created edge sets once eliminating each $2N$ variables for Γ^j, $2 \leq j \leq k-2$.

Proposition 6 *An edge set $\Psi^{1,k}(f_N^j)$ is semi-homogeneous to $\Psi^{1,k}(f_N^{j-1})$ for $2 \leq j \leq k-2$.*

Proof: Given in appendix.

In addition, the edge weight change patterns between two semi-homogeneous edge sets, $\Psi^{1,k}(f_N^{j-1})$ and $\Psi^{1,k}(f_N^j)$, are presented in the following proposition.

Proposition 7 *Consider two semi-homogeneous created edge sets, $\Psi^{1,k}(f_N^{j-1})$ and $\Psi^{1,k}(f_N^j)$, where $2 \leq j \leq k-2$. Suppose $v_1 \xrightarrow{w} v_2 \in \Psi^{1,k}(f_N^j)$ and $g_{(-1)}(v_1) \xrightarrow{w'} g_{(-1)}(v_2) \in \Psi^{1,k}(f_N^{j-1})$. Then, the following is satisfied:*

1. *If $v_1 \neq v_0$ and $v_2 \neq v_0$, $\quad w' \leq w$*
2. *If $v_1 = v_0$ and $v_2 \neq v_0$, $\quad w' \leq w - L$*
3. *If $v_1 \neq v_0$ and $v_2 = v_0$, $\quad w' \leq w + L$*

Proof: Given in appendix.

Once we find two homogeneous created edge sets, $\Psi^{1,k}(f_N^j)$ and $\Psi^{1,k}(f_N^{j-1})$ for some j, then the following proposition enables us to stop the variable elimination process, since homogeneous created edge sets will be found to the ones already obtained, if the node elimination process continues.

Proposition 8 *If an edge set* $\Psi^{1,k}(f_N^j)$ *is homogeneous to an edge set* $\Psi^{1,k}(f_N^{j-1})$*, where* $2 \leq j \leq k-2$*, then*

$$\forall l : 2 \leq l \leq j-1 \ :: \ \forall i : 1 \leq i \leq N \ :: \ \Psi^{1,k}(f_i^l) \sim \Psi^{1,k}(f_i^j) \ \wedge \ \Psi^{1,k}(s_i^l) \sim \Psi^{1,k}(s_i^j)$$

Proof: Given in appendix.

More generalized result is presented next which holds whenever two homogeneous edge sets, $\Psi^{1,k}(f_N^j)$ and $\Psi^{1,k}(f_N^{j-1})$, are found during the variable elimination process.

Proposition 9

$$\Psi^{1,k}(f_N^{j-1}) \sim \Psi^{1,k}(f_N^j) \implies (\forall i : 1 \leq i :: \Psi^{1,k+i}(f_N^{(j-1)+i}) \sim \Psi^{1,k+i}(f_N^{j+i}))$$

Proof: This is obvious from the cyclic structures of constraint graphs, $G^{1,k}(f_N^k)$ and $G^{1,k+i}(f_N^{k+i})$, and from Proposition 7 and 8. ■

From the definition of homogeneity between edge sets in constraint graphs, the following proposition is derived.

Proposition 10 *Suppose* $\Psi^{1,k_1}(s_i^j) \sim \Psi^{1,k_2}(s_i^l)$ *holds. Then,*

1. *the set of edges to* s_i^j *in* $G^{1,k_1}(s_i^j)$ *is homogeneous to the set of edges to* s_i^l *in* $G^{1,k_2}(s_i^l)$*.*
2. *the set of edges from* s_i^j *in* $G^{1,k_1}(s_i^j)$ *is homogeneous to the set of edges from* s_i^l *in* $G^{1,k_2}(s_i^l)$*.*

Note that from the set of edges to s_i^j in $G^{1,k_1}(s_i^j)$ we can obtain the parametric upper bound function $\mathcal{F}_{s_i^j}^{max,k_1}$ for s_i^j, and from the set of edges from s_i^j in $G^{1,k_1}(s_i^j)$ we can obtain the parametric lower bound function $\mathcal{F}_{s_i^j}^{min,k_1}$ for s_i^j by inversely transforming the edge set into constraints. Two parametric lower (upper) bound functions for s_i^j and s_i^l are defined to be *homogeneous* if they satisfy condition 1(2) in the above proposition. If $\mathcal{F}_{s_i^j}^{min,k_1}$ and $\mathcal{F}_{s_i^l}^{min,k_2}$ are homogeneous, it is denoted as:

$$\mathcal{F}_{s_i^j}^{min,k_1} \sim \mathcal{F}_{s_i^l}^{min,k_2}$$

We have the following lemma from Proposition 8 and 9.

Lemma 6 *If* $\Psi^{1,k}(f_N^{j-1}) \sim \Psi^{1,k}(f_N^j)$ *holds for* $2 \leq j \leq k-2$*, then*

1. $\forall l : 2 \leq l \leq j-1 \ :: \ \forall i : 1 \leq i \leq N \ :: \ \mathcal{F}_{s_i^l}^{min,k} \sim \mathcal{F}_{s_i^j}^{min,k} \ \wedge \ \mathcal{F}_{s_i^l}^{max,k} \sim \mathcal{F}_{s_i^j}^{max,k}$
2. $\forall a : 1 \leq a :: \mathcal{F}_{s_i^{(j-1)+a}}^{min,k+a} \sim \mathcal{F}_{s_i^{j+a}}^{min,k+a} \ \wedge \ \mathcal{F}_{s_i^{(j-1)+a}}^{max,k+a} \sim \mathcal{F}_{s_i^{j+a}}^{max,k+a}$

This lemma enables us to obtain *asymptotic* [11] parametric bound functions, $\mathcal{F}_{s_i^j}^{min,\infty}$ and $\mathcal{F}_{s_i^j}^{max,\infty}$, once we find two homogeneous created edge sets during node elimination process from the constraint graph. By using

[11] "Asymptotic" means "converging" in the sense that homogeneous parametric bound functions will be found to the ones already obtained, if the variable elimination process continues.

asymptotic parametric bound functions at run-time we can guarantee that the constraint set $\mathcal{C}^{1,k}$ will be satisfied with any arbitrary value of k.

Note that asymptotic parametric bound functions, $\mathcal{F}^{min,\infty}_{s_i^j}$ and $\mathcal{F}^{max,\infty}_{s_i^j}$, are parameterized in terms of the variables in $\{s_1^{j-1}, f_1^{j-1}, \ldots, f_{i-1}^j\}$ and in terms of the index variable j. By knowing the scheduling window(job set) index j at run-time, only one pair of asymptotic parametric bound functions need to be stored for all s_i^j where i is fixed and $j \leq 2$. In addition to this, another pair of parametric bound functions needs to be stored for s_i^1.

4.4 Off-line Component

In this section, a $4N$-node graph, called *basis graph*, is obtained to which we can cyclically apply variable elimination algorithm without explicitly obtaining a large constraint graph $G^{1,k}(f_N^k)$ for large k. That is, by recursively applying variable elimination algorithm to this smaller graph, it can be decided whether the created edge set sequence, $\Psi^{1,k}(f_N^j)$, $j = k, k-1, \ldots$, will converge or not.

Definition 18 (Basis Graph) *A basis graph $G_b(V_b, E_b)$ is defined as a subgraph of $G^{1,2}(f_N^2)$ as follows*[12].

1. *$V_b = V_{b,1} \cup V_{b,2} \cup \{v_0\}$ where:*

$$V_{b,1} = PrecNode(\Phi^{1,2}(f_N^1)) - \{v_0\}$$

$$V_{b,2} = \{s_1^2, f_1^2, \ldots, s_N^2, f_N^2\}$$

2. *All edges in $G^{1,2}(f_N^2)$ connecting any two nodes in V_b are included into E_b.*

Then, the variable elimination process for a graph $G^{1,k}(f_N^k)$ can be transformed into an equivalent one by using a basis graph as follows:

Algorithm 3 *Cyclic algorithm to obtain $G^{1,k}(f_N^2)$.*

- *Input: k, Basis Graph $G_b(V_b, E_b)$*
- *Output: $G^{1,k}(f_N^2)$*

1. *Initialize $i = 1$.*
2. *Initialize $G_{in}^1(V_b, E_{in}^1) = G_b(V_b, E_b)$.*
3. *From $i = 1$ to $i = k-2$ repeat the following:*
   (a) *Eliminate, from $G_{in}^i(V_b, E_{in}^i)$, the nodes of $V_{b,2}$ by alternately using Algorithm 1 and 2.*
   (b) *If* **False** *is returned from Algorithm 1 or 2, then return* **False**.
   (c) *Let $G_{out}^i(V_{b,1} \cup \{v_0\}, E_{out}^i)$ denote the resulting graph.*

[12] $G^{1,2}(f_N^2)$ is found from $\Gamma^{1,2}$.

(d) If $i \geq 2$ and $G^i_{out}(V_{b,1} \cup \{v_0\}, E^i_{out}) = G^{i-1}_{out}(V_{b,1} \cup \{V_0\}, E^{i-1}_{out})$, then return $G^i_{in}(V_b, E^i_{in})$.

(e) Let $G^{i+1}_{in}(V_b, E^{i+1}_{in}) = G_b(V_b, E_b)$

(f) For each edge $v_1 \xrightarrow{w_{12}} v_2$ in $G^i_{out}(V_{b,1} \cup \{v_0\}, E^i_{out})$,

i. If $v_1 \neq v_0$ and $v_2 \neq v_0$, add an edge $g_{(1)}(v_1) \xrightarrow{w_{12}} g_{(1)}(v_2)$ to $G^{i+1}_{in}(V_b, E^{i+1}_{in})$.

ii. If $v_1 = v_0$, add an edge $g_{(1)}(v_1) \xrightarrow{w_{12}+L} g_{(1)}(v_2)$ to $G^{i+1}_{in}(V_b, E^{i+1}_{in})$.

iii. If $v_2 = v_0$, add an edge $g_{(1)}(v_1) \xrightarrow{w_{12}-L} g_{(1)}(v_2)$ to $G^{i+1}_{in}(V_b, E^{i+1}_{in})$.

(g) Set $i = i + 1$.

At step $3-(d)$ the graph $G^i_{in}(V_b, E^i_{in})$ is returned. By utilizing Proposition 8, this graph can be shown to be equal to $G^{1,k}(f^2_N)$. Once we find homogeneous created edge sets on $V_{b,1} \cup \{v_0\}$ at step $3-(d)$, asymptotic parametric bound functions for job start times can be found from the graph $G^{1,k}(f^2_N)$. From this graph the variables in the sequence $< f^2_N, s^2_N, \ldots, f^2_1, s^2_1 >$ are eliminated to obtain the parametric bound functions for each s^2_i, $1 \leq i \leq N$. During this elimination process, the weights of edges connected to or from v_0 have to be modified appropriately to reflect scheduling window index $j \geq 2$ as well as the node index of the graph. For example,

- if an edge $v_0 \xrightarrow{w} s^2_i$ is obtained after eliminating $< f^2_N, s^2_N, \ldots, f^2_i >$, then a formula $s^j_i \leq w + (j-2)L$ must be used in deriving asymptotic parametric bound functions for s^j_i.
- if an edge $s^2_i \xrightarrow{w} v_0$ is obtained after eliminating $< f^2_N, s^2_N, \ldots, f^2_i >$, then a formula $-w + (j-2)L \leq s^j_i$ must be used in deriving asymptotic parametric bound functions for s^j_i.
- if an edge $s^1_a \xrightarrow{w} s^2_i$, is obtained after eliminating $< f^2_N, s^2_N, \ldots, f^2_i >$, then a formula $s^j_i - s^{j-1}_a \leq w$ must be used in deriving asymptotic parametric bound functions for s^j_i.

After obtaining asymptotic parametric bound functions for s^j_i, $2 \leq j$, we can also find parametric bound functions for Γ^1 by eliminating nodes from $G^{1,k}(f^1_N)$.

Note that, at each iteration in the above algorithm, no explicit transformation of node indices are performed by using $g_{(-1)}$. This is because our purpose is to check the schedulability and obtain asymptotic parametric bound functions, and this may be done without explicit knowledge of node indices. The key property that this algorithm makes use of is that the basis graph is recursively used and transformed until the schedulability is checked. It is clear that this algorithm produces exactly the same result (**True** or **False**) and graph as the node elimination algorithm applied to $G^{1,k}(f^k_N)$ does.

The following theorem provides an upper bound on the number of loop iterations in Algorithm 3 that have to be performed before the schedulability is checked.

Theorem 1 *If Algorithm 3 doesn't terminate within $n^2 - n + 2$ loop iterations, then $sched^{1,\infty}$ is not schedulable.*

Proof: Given in appendix.

Therefore, we obtain the final algorithm for checking $sched^{1,\infty}$ and deriving asymptotic parametric bound functions if $\Gamma^{1,\infty}$ is schedulable. The overview of off-line component is shown in Figure 10.

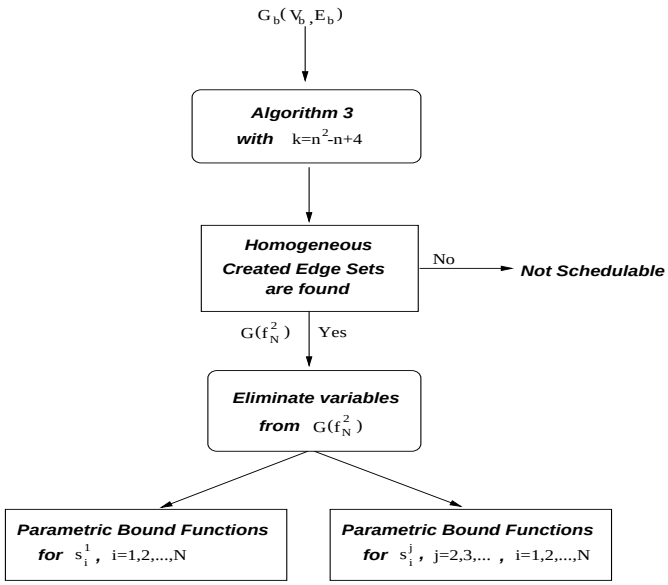

Figure 10: Overview of off-line component

From Theorem 1 the total complexity of the off-line algorithm is $O(n^2N^3)$, since each loop iteration of Algorithm 3 may take at most $O(N^3)$ computation time [4]. If only jitter constraints are allowed from periodic tasks, then the off-line algorithm will be finished within $O(n^4N)$ time where n is the number of periodic tasks that have jitter constraints, since each loop iteration in this case takes at most $O(n^2N)$ time. This is because $\mid PrecNode(\Phi^{1,k}(s_i^j)) \cup P \mid \leq n+2$ holds, and because from Proposition 4 we know that at most $O(n)$ number of edges exist in $G^{1,k}(s_i^j)$ that are connected to or from s_i^j. This implies that the elimination of s_i^j from the graph $G^{1,k}(s_i^j)$ will require at most $O(n^2)$ time, and eliminating nodes of one job set requires $O(n^2N)$ time. Also, the on-line component in this case requires at most $O(n)$ execution time.

4.5 Off-line Component with Restricted Standard Constraints

For a certain class of standard constraints, called *restricted standard constraints*, it will be shown that the off-line component can be carried out in $O(N^3+n^5)$ time instead of $O(n^2N^3)$ time.

Definition 19 (Restricted Standard Constraints) *For two jobs, τ_a^j and τ_b^l, where $(j=l-1)\vee(j=l\wedge a<b)$, the following constraints are defined as* restricted *standard constraints:*

$$\begin{aligned} s_a^j - s_b^l &\leq c_1 & \qquad s_b^l - s_a^j &\leq c_3 \\ s_a^j + e_a^j - s_b^l &\leq c_2 & \qquad s_b^l + e_b^l - s_a^j &\leq c_4 \end{aligned} \tag{16}$$

Also, as in the definition for standard constraints, release time and deadline constraints can also be classified as restricted standard constraints. We also include as restricted standard any constraint that can be rewritten in one of the above forms.

For this class of constraints the following lemma makes it possible to pre-process the basis graph and to obtain a smaller graph that can be used in the off-line algorithm instead of the basis graph. This graph is called a *compact* basis graph.

Lemma 7 ([8]) *If $\Gamma^{1,k}$ is constructed with restricted standard constraints, it is schedulable if and only if it is schedulable for the maximum execution times of the jobs.*

Let the following be a predicate representing a schedulability for a job set Π.

$$Sched \equiv \exists s_1 :: \forall e_1 \in [l_1, u_1] :: \ldots \exists s_i :: \forall e_i \in [l_i, u_i] :: \exists s_N :: \forall e_N \in [l_N, u_N] :: \mathcal{C}$$

From Lemma 7 this predicate is equivalent to the following predicate where $\mathcal{C}$ only consists of restricted standard constraints.

$$\exists s_1 :: \ \ldots \ :: \exists s_i :: \ldots \ \exists s_{N-1} :: \exists s_N :: \ \mathcal{C}[e_j/u_j \ : \ 1 \leq j \leq N]$$

where e_j/u_j denotes a substitution of u_j for a variable e_j. In other words, *Sched* can be checked by first replacing every universally quantified variable e_j with u_j for $1 \leq j \leq N$, and then by eliminating existentially quantified variables s_N, ..., s_1.

However, eliminating the existentially quantified variables, s_N, s_{N-1}, ..., s_{i+1}, in any order will produce the same constraint graph $G(s_i)$. This is because there exists no exclusively-ORed edges between nodes in $\{v_0, s_{i+1}, s_{i+2}, \ldots, s_N\}$ after substituting the maximum execution times for the variables e_j, $1 \leq j \leq N$, and because any minimum weight acyclic restricted paths through the nodes of $\{s_{i+1}, \ldots, s_N\}$ are preserved in the remaining constraint graph after eliminating the variables s_j, $i+1 \leq j \leq N$, regardless of the elimination order.

This property is used to find a *compact* basis graph from $sched^{1,2}(f_N^2)$ as follows:

Algorithm 4 (Compact Basis Graph) *Algorithm to obtain a compact basis graph.*

- *Input: $sched^{1,2}(f_N^2)$*
- *Output: Compact Basis Graph $G_{cb}(V_{cb}, E_{cb})$*

1. *Let $G^{'}(V^{'}, E^{'})$ denote a graph from a predicate that is found by substituting u_j for each universally quantified variable e_j in $sched^{1,2}(f_N^2)$.*
2. *Let $\Phi^{'}(s_N^1)$ denote a crossing edge set of s_N^1 found from $G^{'}(V^{'}, E^{'})$.*
3. *Let $G^{''}(V^{''}, E^{''})$ denote a graph found after eliminating the following nodes from $G^{'}(V^{'}, E^{'})$.*

$$\{s_1^2, s_2^2, \ldots, s_N^2\} - g_{(1)}(PrecNode(\Phi^{'}(s_N^1)))$$

4. *Let $G_{cb}(V_{cb}, E_{cb})$ be a subgraph of $G^{''}(V^{''}, E^{''})$:*

(a) $V_{cb} = V_{cb,1} \cup V_{cb,2} \cup \{v_0\}$ *where:*

$$V_{cb,1} = \{s_1^1, s_2^1 \ldots, s_N^1\} \cap PrecNode(\Phi^{'}(s_N^1))$$
$$V_{cb,2} = g_{(1)}(V_{cb,1})$$

(b) All edges in $G^{''}(V^{''}, E^{''})$ *connecting two nodes of* V_{cb} *defines* E_{cb}.

We can apply Algorithm 3 to this compact basis graph instead of the basis graph. This limits the complexity of obtaining homogeneous created edge sets to $O(N^3 + n^5)$ instead of $O(n^2N^3)$. Once we find homogeneous created edge sets on $V_{cb,1}$, asymptotic parametric bound functions can be found by first unrolling the final graph from the algorithm to obtain $G^{1,\infty}(f_N^2)$ and then by eliminating from this graph the nodes in the sequence $< f_N^2, s_N^2, \ldots, f_1^2, s_1^2 >$. During this elimination process, as in Section 4.4 the weights of edges connecting v_0 have to be modified appropriately to reflect scheduling window index as well as the node indices of the graph.

5 Example

The asymptotic parametric bound functions are found for the job set, $\Gamma^{1,\infty}$, in Example 1. Figure 11 shows the parametric bound functions found from $\Gamma^{1,4}$, and Figure 12 shows asymptotic parametric bound functions for $sched^{1,\infty}$.

0	$\leq$	s_1^1	$\leq$	2
$\max(8, s_1^1 + e_1^1)$	$\leq$	s_2^1	$\leq$	$\min(10, s_1^1 + e_1^1 + 5)$
$\max(20, s_2^1 + e_2^1, s_1^1 + e_1^1 + 10)$	$\leq$	s_1^2	$\leq$	$\min(22, s_1^1 + e_1^1 + 17, s_2^1 + e_2^1 + 4)$
$\max(28, s_1^2 + e_1^2, s_2^1 + e_2^1 + 10)$	$\leq$	s_2^2	$\leq$	$\min(30, s_2^1 + e_2^1 + 12, s_1^2 + e_1^2 + 5)$
$\max(40, s_2^2 + e_2^2, s_1^2 + e_1^2 + 10)$	$\leq$	s_1^3	$\leq$	$\min(42, s_1^2 + e_1^2 + 17, s_2^2 + e_2^2 + 4)$
$\max(48, s_1^3 + e_1^3, s_2^2 + e_2^2 + 10)$	$\leq$	s_2^3	$\leq$	$\min(50, s_2^2 + e_2^2 + 12, s_1^3 + e_1^3 + 5)$
$\max(60, s_2^3 + e_2^3, s_1^3 + e_1^3 + 10)$	$\leq$	s_1^4	$\leq$	$\min(62, s_1^3 + e_1^3 + 17, s_2^3 + e_2^3 + 4)$
$\max(s_1^4 + e_1^4, s_2^3 + e_2^3 + 10)$	$\leq$	s_2^4	$\leq$	$\min(70, s_2^3 + e_2^3 + 12, s_1^4 + e_1^4 + 5)$

Figure 11: Parametric bound functions found from $sched^{1,4}$

$\mathcal{F}^{min}_{s_1^1}$	$=$	0
$\mathcal{F}^{max}_{s_1^1}$	$=$	2
$\mathcal{F}^{min}_{s_2^1}$	$=$	$\max(8, s_1^1 + e_1^1)$
$\mathcal{F}^{max}_{s_2^1}$	$=$	$\min(10, s_1^1 + e_1^1 + 5)$
$\mathcal{F}^{min}_{s_1^j}$	$=$	$\max(20 + (j-2)20, s_2^{j-1} + e_2^{j-1}, s_1^{j-1} + e_1^{j-1} + 10)$
$\mathcal{F}^{max}_{s_1^j}$	$=$	$\min(22 + (j-2)20, s_1^{j-1} + e_1^{j-1} + 17, s_2^{j-1} + e_2^{j-1} + 4)$
$\mathcal{F}^{min}_{s_2^j}$	$=$	$\max(28 + (j-2)20, s_1^j + e_1^j, s_2^{j-1} + e_2^{j-1} + 10)$
$\mathcal{F}^{max}_{s_2^j}$	$=$	$\min(30 + (j-2)20, s_2^{j-1} + e_2^{j-1} + 12, s_1^2 + e_1^2 + 5)$

Figure 12: Asymptotic parametric bound functions for $sched^{1,\infty}$

It is clear from this figure that the following hold:

$$\mathcal{F}^{min,4}_{s^2_1} \sim \mathcal{F}^{min,4}_{s^3_1}$$

$$\mathcal{F}^{max,4}_{s^2_1} \sim \mathcal{F}^{max,4}_{s^3_1}$$

$$\mathcal{F}^{min,4}_{s^2_2} \sim \mathcal{F}^{min,4}_{s^3_2}$$

$$\mathcal{F}^{max,4}_{s^2_2} \sim \mathcal{F}^{max,4}_{s^3_2}$$

Note that $n = \mid PrecNode(\Phi^{1,4}(f^1_2)) \mid = 3$, and $n^2 - n + 2 = 8$ is the iteration bound given in Theorem 1. But, Algorithm 3 found homogeneous created edge sets after 3 loop iterations. This shows that the upper bound on the number of loop iterations given in Theorem 1 is not tight in general, and the schedulability may be checked within less amount of time.

6 Conclusion

In this paper, we proposed a dynamic cyclic dispatching scheme that may be applied to real-time systems with complex timing constraints, such as relative constraints between start or finish times of jobs. A schedule(ordering) of N jobs is assumed to be given on a scheduling window, and it is required that this schedule be repeated at run time. The relative constraints may be cyclically defined across the boundaries of the scheduling windows as well as between jobs in one scheduling window.

Unlike static approaches which assign fixed start times to jobs in a scheduling window, our approach not only allows us to flexibly manage the slack times with the schedulability of a job set not affected, but also yields an guaranteed schedulability in the sense that, if other dispatching mechanism can dispatch the job sequences satisfying all given constraints, then our mechanism can also schedule them.

A pseudo-polynomial time off-line algorithm is presented to check the schedulability of a cyclically constrained job set and to obtain parametric lower and upper bound functions for each job start time. The off-line algorithm requires at most $O(n^2N^3)$ time where n is the number of relative constraints defined across the boundary of two consecutive scheduling windows. Then, the parametric bound functions for each start time can be evaluated by an on-line algorithm within $O(N)$ time. Especially, with restricted standard constraints it is shown that the off-line component requires at most $O(N^3 + n^5)$ execution time.

We believe that the dynamic cyclic dispatching scheme can be applied to many real-time applications that have complex timing constraints and provide more flexibility in managing system resources at runtime.

References

[1] T. Carpenter, K. Driscoll, K. Hoyme, and J. Carciofini. Arinc 659 scheduling: Problem definition. In *Proceedings, IEEE Real-time Systems Symposium*, San Juan, PR, December 1994.

[2] S. Cheng and Ashok K. Agrawala. Scheduling of periodic tasks with relative timing constraints. Technical Report CS-TR-3392, UMIACS-TR-94-135, Department of Computer Science, University of Maryland, December 1994.

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

[4] R. Gerber, W. Pugh, and M. Saksena. Parametric Dispatching of Hard Real-Time Tasks. *IEEE Transactions on Computers*, 44(3), Mar. 1995.

[5] C. Han, C. Hou, and K. Lin. Distance-Constrained Scheduling and Its Applications to Real-Time Systems. *IEEE Transactions on Computers*. To appear.

[6] Seung H. Hong. Scheduling Algorithm of Data Sampling Times in the Integrated Communication and Control Systems. *IEEE Transactions on Control Systems Technology*, 3(2):225–230, June 1995.

[7] C. L. Liu and J. Layland. Scheduling Algorithm for Multiprogramming in a Hard Real-Time Environment. *Journal of the ACM.*, 20(1):46–61, Jan. 1973.

[8] M. Saksena. *Parametric Scheduling for Hard Real-Time Systems*. PhD thesis, University of Maryland, College Park, MD 20742, 1994.

A Proofs

Proof of Lemma 3: It is obvious that there exists an one-to-one correspondence between an edge pair set in $G(f_a)$ from which a new edge will be created after f_a is eliminated, and a constraint in $Sched(e_a)$ to be changed after eliminating e_a. Also, it is clear that a new constraint created in $Sched(s_a)$ will correspond to a new edge created in $G(s_a)$. Therefore, $Elim(G(f_a), f_a)$ is equal to $G(s_a)$.

Proof of Lemma 4: The proof for this lemma is similar to that of Lemma 3, and is omitted.

Proof of Proposition 3: Let π be a negative weight restricted cycle in $G(f_N)$ satisfying:

- no restricted cycle appears as a proper sub-cycle of π.

If there exists a negative weight restricted cycle in $G(f_N)$, then π also exists in $G(f_N)$. Also, let y be a node in π that appears first in a sequence $< v_0, s_1, f_1, \ldots, s_N, f_N >$. Then, π can be denoted as

$$< y \xrightarrow{w_1} v_1 \xrightarrow{w_2} v_2 \ldots v_i \xrightarrow{w_{i+1}} y >$$

where $\sum_{j=1}^{i+1} w_j < 0$. By eliminating nodes that lie after y in the node sequence $< v_0, s_1, f_1, \ldots, s_N, f_N >$, we will obtain a negative weight edge $y \xrightarrow{w'} y$ where $w' < 0$. This is clear from the path preserving property of node elimination algorithms. Then, from the equivalence relationship between constraint graphs and predicates, a contradiction is obtained during the elimination of the variables from $Sched$. Therefore, $Sched$ is equal to **False**. ■

Proof of Lemma 5: Claim 1: If $y \xrightarrow{w} z \in G(f_a)$ holds where $y \neq z$, then there exists an acyclic[13] restricted path $y \overset{w'}{\rightsquigarrow} z$ in $G(f_N)$ where $w' \leq w$ and all its intermediate nodes belong to $\{s_{a+1}, f_{a+1}, \ldots, s_N, f_N\}$.

If $v = f_N$, then the claim holds. Suppose that there exists an edge $y \xrightarrow{w} z$ in $G(f_a)$ where $1 \leq a \leq N-1$. Assume that there exists an acyclic restricted path in $G(f_b)$ with a weight sum w, $a \leq b \leq N-1$,

$$< y \xrightarrow{w_{b,1}} v_1 \xrightarrow{w_{b,2}} v_2 \ldots \xrightarrow{w_{b,i}} v_i \xrightarrow{w_{b,i+1}} z > \tag{17}$$

where $i \geq 0$, and $v_j \in \{s_{a+1}, f_{a+1}, \ldots, s_b, f_b\}$ for $1 \leq j \leq i$. If all edges constituting this path exist in $G(f_{b+1})$ with same weights, then there exists an acyclic restricted path in $G(f_{b+1})$ with a weight sum w where all its intermediate nodes belong to $\{s_{a+1}, f_{a+1}, \ldots, s_{b+1}, f_{b+1}\}$. So, assume that at least one of these edges is created in $G(f_b)$ just after eliminating f_{b+1} and s_{b+1} from $G(f_{b+1})$. Let $\mathcal{J} = \{j_1, j_2, \ldots, j_k\}$, where $1 \leq k \leq i+1$ and $1 \leq j_l \leq i+1$ for $1 \leq l \leq k$, denote an index set of edges in the above path which are newly created in $G(f_b)$. The indices in $\mathcal{J}$ is assumed to be increasing. Each edge $v_{j_l-1} \xrightarrow{w_{b,j_l}} v_{j_l}$, for $1 \leq l \leq k$, is created[14] just after f_{b+1} and s_{b+1} are eliminated from $G(f_{b+1})$.

[13] For a case when $y = z$, it can be similarly shown that a restricted path without any intermediate restricted cycle(i.e., excluding y and z) is obtained, even though the resulting restricted path is not acyclic.

[14] For the purpose of convenience v_0 denotes a node y, and v_{i+1} denotes a node z.

Fact 1: In $G(f_{b+1})$ the weight of an edge $s_{b+1} \rightarrow f_{b+1}$ is equal to l_{b+1}, and the weight of $f_{b+1} \rightarrow s_{b+1}$ is equal to $-u_{b+1}$.

If the fact is not true, then a contradiction should have been derived, which is against the assumption.

From the node elimination algorithm we know that the edge $v_{j_l-1} \stackrel{w_{b,j_l}}{\longrightarrow} v_{j_l}$ is created from one of the following restricted paths in $G(f_{b+1})$ whose weight sum is w_{b,j_l}:

1. $< v_{j_l-1} \stackrel{w^1_{b+1,j_l}}{\longrightarrow} s_{b+1} \stackrel{w^2_{b+1,j_l}}{\longrightarrow} v_{j_l} >$
2. $< v_{j_l-1} \stackrel{w^1_{b+1,j_l}}{\longrightarrow} f_{b+1} \stackrel{-u_{b+1}}{\longrightarrow} s_{b+1} \stackrel{w^2_{b+1,j_l}}{\longrightarrow} v_{j_l} >$
3. $< v_{j_l-1} \stackrel{w^1_{b+1,j_l}}{\longrightarrow} s_{b+1} \stackrel{l_{b+1}}{\longrightarrow} f_{b+1} \stackrel{w^2_{b+1,j_l}}{\longrightarrow} v_{j_l} >$
4. $< v_{j_l-1} \stackrel{w^1_{b+1,j_l}}{\longrightarrow} f_{b+1} \stackrel{-u_{b+1}}{\longrightarrow} s_{b+1} \stackrel{l_{b+1}}{\longrightarrow} f_{b+1} \stackrel{w^2_{b+1,j_l}}{\longrightarrow} v_{j_l} >$

We can extend the path in (17) into a path in $G(f_{b+1})$ by replacing each edge in (17) with an index j_l by one of the above paths via s_{b+1} and f_{b+1}.

If $k = 1$, i.e., only one edge is created after eliminating f_{b+1} and s_{b+1} from $G(f_{b+1})$, then it is obvious that the extended path is also a restricted path with a weight w in $G(f_{b+1})$. So, assume that $k \geq 2$. In this case, there exists a cycle in the extended path.

First, consider two edges, $v_{j_1-1} \stackrel{w_{b,j_1}}{\longrightarrow} v_{j_1}$ and $v_{j_2-1} \stackrel{w_{b,j_2}}{\longrightarrow} v_{j_2}$. For all 16 possible combinations of the above 4 paths from which these two edges will be created, a restricted cycle is obtained after extending these two edges in (17). For example, if both of these two edges are created from the paths of the form 4, then the extended path will be of the following form:

$$\begin{aligned} < \quad y \quad & \rightarrow v_1 \rightarrow v_2 \ldots v_{j_1-1} \rightarrow \\ & < f_{b+1} \rightarrow s_{b+1} \rightarrow f_{b+1} \rightarrow v_{j_1} \ldots v_{j_2-1} \rightarrow f_{b+1} > \\ & \rightarrow s_{b+1} \rightarrow f_{b+1} \rightarrow v_{j_2} \ldots v_i \rightarrow z > \end{aligned}$$

The inner path, $< f_{b+1} \rightarrow s_{b+1} \rightarrow f_{b+1} \rightarrow v_{j_1} \ldots v_{j_2-1} \rightarrow f_{b+1} >$, is a restricted cycle, since the sub-path $< v_{j_1} \ldots v_{j_2-1} >$ is a restricted path and neither s_{b+1} nor f_{b+1} appears in this sub-path. Then, from Proposition 3 the weight sum of this restricted cycle is non-negative. If it is negative, then a **False** should have been derived during eliminating the nodes in $\{f_N, s_N, \ldots, f_{a+1}, s_{a+1}\}$, which is a contradiction to the assumption. Therefore, if we reduce this restricted cycle into a single node f_{b+1}, then we obtain the following restricted path whose weight sum is less than or equal to w:

$$< y \rightarrow v_1 \rightarrow v_2 \ldots v_{j_1-1} \rightarrow f_{b+1} \rightarrow s_{b+1} \rightarrow f_{b+1} \rightarrow v_{j_2} \ldots v_i \rightarrow z >$$

As a result, two edges, $v_{j_1-1} \stackrel{w_{b,j_1}}{\longrightarrow} v_{j_1}$ and $v_{j_2-1} \stackrel{w_{b,j_2}}{\longrightarrow} v_{j_2}$, are merged into one sub-path

$$v_{j_1-1} \rightarrow f_{b+1} \rightarrow s_{b+1} \rightarrow f_{b+1} \rightarrow v_{j_2}$$

Similarly, for other combinations for two edges, $v_{j_1-1} \xrightarrow{w_{b,j_1}} v_{j_1}$ and $v_{j_2-1} \xrightarrow{w_{b,j_2}} v_{j_2}$, the similar results can be obtained.

If we continue this merging process for an edge, $v_{j_3-1} \xrightarrow{w_{b,j_3}} v_{j_3}$, and for the sub-path $< v_{j_1-1} \rightarrow f_{b+1} \rightarrow s_{b+1} \rightarrow f_{b+1} \rightarrow v_{j_2} >$ found above, we will obtain a merged acyclic sub-path from v_{j_1-1} to v_{j_3} through f_{b+1} or s_{b+1}.

Therefore, after $k-1$ iterations of the above process, we will obtain an acyclic restricted path in $G(f_{b+1})$ whose intermediate nodes belong to $\{s_{a+1}, f_{a+1}, \ldots, s_{b+1}, f_{b+1}\}$ and whose weight sum is less than or equal to w.

Therefore, by inductively applying the above argument, we know that there exists an acyclic restricted path in $G(f_N)$ whose intermediate nodes belong to $\{s_{a+1}, f_{a+1}, \ldots, s_N, f_N\}$ and whose weight sum is less than or equal to w.

Claim 2: If there exists an acyclic [15] restricted path $y \overset{w}{\rightsquigarrow} z$ in $G(f_N)$ whose intermediate nodes belong to $\{s_{a+1}, f_{a+1}, \ldots, s_N, f_N\}$, then $y \xrightarrow{w'} z \in G(f_a)$ holds where $w' \leq w$.

The proof for this claim is similar to that for Proposition 3, and is omitted.

From claim 1 and 2 the lemma is proved. ■

Proof of Corollary 1: Suppose that an edge $y \rightarrow z$ exists in $G(v)$. If $v = f_a$ for some a, then from Lemma 5 it is obvious that there exists a path $y \rightsquigarrow z$ in $G(f_N)$ whose intermediate nodes belong to $\{v', \ldots, s_N, f_N\}$. So, assume that $v = s_a$ for some a in $[1, N]$.

If there exists an edge from y to z in $G(f_a)$, then the condition 2 holds. Hence, further assume that an edge $y \rightarrow z$ is created just after eliminating f_a from $G(f_a)$. From the node elimination algorithm, the edge is created from either of the following paths:

1. $y \rightarrow f_a \rightarrow s_a$
2. $s_a \rightarrow f_a \rightarrow z$

From Lemma 5 we know that there exist two acyclic restricted paths whose intermediate nodes belong to $\{s_{a+1}, f_{a+1}, \ldots, s_N, f_N\}$. By merging these paths, we obtain a path from y to z whose intermediate nodes belong to $\{f_a, s_{a+1}, f_{a+1}, \ldots, s_N, f_N\}$. ■

Proof of Proposition 4: If there exists an edge connecting s_i^j and v in $G^{1,k}(f_N^k)$, then it is obvious that v belongs to a node set P. So, assume that there exists no such edge in $G^{1,k}(f_N^k)$.

Two cases must be considered.

Case 1: $v \xrightarrow{w} s_i^j \in G^{1,k}(s_i^j)$

From Corollary 1 there exists a path from v to s_i^j in $G^{1,k}(f_N^k)$ whose intermediate nodes belong to $\{f_i^j, \ldots, s_N, f_N\}$. Note that this path has at least one intermediate node. From the definition of a crossing edge set $\Phi^{1,k}(s_i^j)$, it is clear that $v \in PrecNode(\Phi^{1,k}(s_i^j))$.

[15] For a case when $y = z$, it can be similarly shown that the claim holds for a restricted path without any intermediate restricted cycle(i.e., excluding y and z).

Case 2: $s_i^j \xrightarrow{w} v \in G^{1,k}(s_i^j)$

Similarly, the proposition can be proved in this case. ■

Proof of Proposition 5: Suppose that v belong to $Node(\Psi^{1,k}(s_i^j))$. Then, there exist an edge $v' \in \{v_0, s_1^1, f_1^1, \ldots, s_i^j\}$ such that an edge $v \rightarrow v'$ exists in $G(s_i^j)$. Then from Corollary 1, we know that there exists a path $v \rightsquigarrow v'$ in $G^{1,k}(f_N^k)$ where all intermediate nodes in the path belong to $\{f_i^j, \ldots, s_N^k, f_N^k\}$. From the definition of $\Psi^{1,k}(s_i^j)$ there exist two edges $v \rightarrow v_1$ and $v_2 \rightarrow v'$ in $v \rightsquigarrow v'$ where v_1 and v_2 belong to $\{f_i^j, \ldots, s_N^k, f_N^k\}$. Note that v_1 may be equal to v_2. This means that v is an element of $PrecNode(\Phi^{1,k}(s_i^j))$. Thus, $Node(\Psi^{1,k}(s_i^j)) \subseteq PrecNode(\Phi^{1,k}(s_i^j))$ is proved. The second assertion, $Node(\Psi^{1,k}(f_i^j)) \subseteq PrecNode(\Phi^{1,k}(f_i^j))$, can be proved in a similar way. Also, from these we know that a maximum number of edges in $\Psi^{1,k}(f_N^j)$, $1 \leq j \leq k-1$, is less than or equal to $n(n-1)$, since n is the number of nodes in $PrecNode(\Phi^{1,2}(f_N^1))$. ■

Proof of Proposition 6: Claim 1: If there exists an edge from v_1 to v_2 in $\Psi^{1,k}(f_N^{j-1})$, then there also exists an edge from $g_{(1)}(v_1)$ to $g_{(1)}(v_2)$ in $\Psi^{1,k}(f_N^j)$.

First suppose that $v_1 \rightarrow v_2 \in \Psi^{1,k}(f_N^{j-1})$ where $1 \leq j-1 \leq k-3$. Then, from the definition of a created edge set, there exists a path from v_1 to v_2 that has at least one intermediate node and whose intermediate nodes belong to $\{s_1^j, f_1^j, \ldots, s_N^k, f_N^k\}$. By applying a technique similar to the one used in the claim 1 of the proof for Lemma 5, we can reduce this path into an acyclic restricted path from v_1 to v_2 that has at least one intermediate node. Let this reduced path be denoted as $< v_1 \rightarrow x_1 \rightarrow x_2 \ldots \rightarrow x_l \rightarrow v_2 >$, $l \geq 1$, where every intermediate node x_h $(1 \leq h \leq l)$ belongs to $< s_1^j, f_1^j, \ldots, s_N^k, f_N^k >$. If all nodes x_h, $1 \leq h \leq l$, belong to $\{s_1^j, f_1^j, \ldots, s_N^{k-1}, f_N^{k-1}\}$, then it is clear from the cyclic nature of constraint graphs that there exists an acyclic restricted path from $g_{(1)}(v_1)$ to $g_{(1)}(v_2)$ in $G^{1,k}(f_N^k)$ whose intermediate nodes belong to $\{s_1^{j+1}, f_1^{j+1}, \ldots, s_N^k, f_N^k\}$.

Hence, assume that there exists at least one x_m, $1 \leq m \leq l$, that belongs to $\{s_1^k, f_1^k, \ldots, s_N^k, f_N^k\}$. Note that $x_1, x_l \in \{s_1^j, f_1^j, \ldots, s_N^j, f_N^j\}$. There are two possible cases to be considered:

1. x_1 is located later than x_l in the node sequence $< s_1^j, f_1^j, \ldots, s_N^j, f_N^j >$.

   - In this case there exists an acyclic restricted path $< v_1 \rightarrow x_1 \rightsquigarrow x_l \rightarrow v_2 >$ whose intermediate nodes belong to $\{s_1^j, f_1^j, \ldots, s_N^j, f_N^j\}$. This is because every node in constraint graphs has an edge to its previous node in the node sequence $< s_1^1, f_1^1, \ldots, s_N^k, f_N^k >$. In other words, $g_{(1)} < v_1 \rightarrow x_1 \rightsquigarrow x_l \rightarrow v_2 >$ is an acyclic restricted path from $g_{(1)}(v_1)$ to $g_{(1)}(v_2)$ in $G^{1,k}(f_N^k)$ whose intermediate nodes belong to $\{s_1^{j+1}, f_1^{j+1}, \ldots, s_N^k f_N^k\}$.[16] Hence, from Lemma 5 there exists an edge $g_{(1)}(v_1) \rightarrow g_{(1)}(v_2)$ in $G^{1,k}(f_N^j)$. Because there exists a path from $g_{(1)}(v_1)$ to $g_{(1)}(v_2)$ satisfying the condition given in definition of a created edge set, this edge belongs to $\Psi^{1,k}(f_N^j)$

2. x_1 is located before x_l.

   - Let the reduced path be denoted as $< v_1 \rightarrow x_1 \rightsquigarrow x_i \rightsquigarrow x_m \rightarrow x_l \rightarrow v_2 >$ where x_i is a first node appearing in this path that lies after x_1 in the node sequence $< s_1^j, f_1^j, \ldots, s_N^k, f_N^k >$. Note that

[16] $g_{(a)} < y_1 \rightarrow y_2 \ldots \rightarrow y_i >$ is defined to be $< g_{(a)}(y_1) \rightarrow g_{(a)}(y_2) \ldots \rightarrow g_{(a)}(y_i) >$.

$x_i \in \{s_1^j, f_1^j, \ldots, s_N^{j+1}, f_N^{j+1}\}$. Again, since $j+1 \leq k-1$ and every node has a path to its predecessor in the node sequence, there exists an acyclic restricted path $< v_1 \rightarrow x_1 \rightsquigarrow x_i \rightsquigarrow x_l \rightarrow v_2 >$ that doesn't have a node for a job in Γ^k. Hence, there exists an acyclic restricted path $g_{(1)} < v_1 \rightarrow x_1 \rightsquigarrow x_i \rightsquigarrow x_l \rightarrow v_2 >$ whose intermediate nodes belong to $\{s_1^{j+1}, f_1^{j+1}, \ldots, s_N^k, f_N^k\}$. This means that $g_{(1)}(v_1) \rightarrow g_{(1)}(v_2) \in G^{1,k}(f_N^j)$. Also, because the above path satisfies the definition for a created edge set, this edge belongs to $\Psi^{1,k}(f_N^j)$

Claim 2: If there exists an edge from v_3 to v_4 in $\Psi^{1,k}(f_N^j)$, then there also exists an edge from $g_{(-1)}(v_3)$ to $g_{(-1)}(v_4)$ in $\Psi^{1,k}(f_N^{j-1})$.

Suppose that there exists an edge from v_3 to v_4 in $\Psi^{1,k}(f_N^j)$. Then, from the definition of a created edge set, there exists a path from v_3 to v_4 that has at least one intermediate node and whose intermediate nodes belong to $\{s_1^{j+1}, f_1^{j+1}, \ldots, s_N^k, f_N^k\}$. By applying the technique in the claim 1 of the proof for Lemma 5, we can reduce this path into an acyclic restricted path from v_3 to v_4 that has at least one intermediate node. Let this path be denoted as $< v_3 \rightsquigarrow v' \rightsquigarrow v_4 >$ where v' belongs to $\{s_1^{j+1}, f_N^{j+1}, \ldots, s_N^k, f_N^k\}$. In this case, the path $g_{(-1)} < v_3 \rightsquigarrow v' \rightsquigarrow v_4 >$ is also an acyclic restricted path in $G(f_N^k)$ whose intermediate nodes belong to $\{s_1^j, f_N^j, \ldots, s_N^{k-1}, f_N^{k-1}\}$. Then, from Lemma 5 there exists an edge $g_{(-1)}(v_3) \rightarrow g_{(-1)}(v_4)$ in $G^{1,k}(f_N^{j-1})$. Also, because the path $g_{(-1)} < v_3 \rightsquigarrow v' \rightsquigarrow v_4 >$ satisfies the condition in the definition of a created edge set, this edge belongs to $\Psi^{1,k}(f_N^{j-1})$, too.

From Claim 1 and 2, we conclude that $\Psi^{1,k}(f_N^{j_1})$ is semi-homogeneous to $\Psi^{1,k}(f_N^{j_2})$ for $1 \leq j_1 \leq j_2 \leq k-2$. ■

Proof of Proposition 7: From Lemma 5 there exists a minimum weight acyclic restricted path $\pi_1 =< v_1 \overset{w}{\rightsquigarrow} v_2 >$ whose intermediate nodes belong to $\{s_1^{j+1}, f_1^{j+1}, \ldots, s_N^k, f_N^k\}$, and a minimum weight acyclic restricted path $\pi_2 =< g_{(-1)}(v_1) \overset{w'}{\rightsquigarrow} g_{(-1)}(v_2) >$ whose intermediate nodes belong to $\{s_1^j, f_1^j, \ldots, s_N^k, f_N^k\}$. Three cases must be examined:

Case 1: $v_1 \neq v_0$ and $v_2 \neq v_0$

In this case it is clear that w' is less than or equal to w, since the set of acyclic restricted paths from v_1 to v_2 in $G^{1,k}(f_N^k)$ whose intermediate nodes belong to $\{s_1^{j+1}, f_1^{j+1}, \ldots, s_N^k, f_N^k\}$ is a subset of a set of acyclic restricted paths from v_1 to v_2 in $G^{1,k}(f_N^k)$ whose intermediate nodes belong to $\{s_1^j, f_1^j, \ldots, s_N^k, f_N^k\}$.

Case 2: $v_1 = v_0$

The path $g_{(-1)}\pi_1$ is also an acyclic restricted path. The weight of a path $g_{(-1)}\pi_1$ is equal to $w - L$, since every edge weight in this new path is the same as that of corresponding edge in π_1 except for the first edge $v_0 \rightarrow g_{(-1)}(x_1)$ of $g_{(-1)}\pi_1$ where x_1 denotes the first node appearing after v_0 in π_1. The weight of this edge is L less than that of $v_0 \rightarrow x_1$ which is the first edge of π_1. This implies $w' \leq w - L$ from Lemma 5.

Case 3: $v_2 = v_0$

The path $g_{(-1)}\pi_1$ is also an acyclic restricted path. The weight of a path $g_{(-1)}\pi_1$ is equal to $w + L$, since every edge weight in this new path is the same as that of corresponding edge in π_1 except for the last edge $g_{(-1)}(x_l) \rightarrow v_0$ of $g_{(-1)}\pi_1$. The weight of this edge is L more than the weight of $x_l \rightarrow v_0$ which is the last edge of π_1. This implies $w' \leq w + L$ from Lemma 5. ■

Proof of Proposition 8: Note that two created edge sets, $\Psi^{1,k}(f_i^{j-1})$ and $\Psi^{1,k}(f_i^j)$, can be shown to be semi-homogeneous by employing similar proof to that for Proposition 6 where $2 \leq j \leq k-2$

The following claim is proved where i is any integer satisfying $1 \leq i \leq N$.

Claim 1: $\Psi^{1,k}(f_i^{j-1}) \sim \Psi^{1,k}(f_i^j)$

First, suppose that $v_1 \xrightarrow{w_1} v_2 \in \Psi^{1,k}(f_i^{j-1})$, where $v_1 \neq v_0$,$v_2 \neq v_0$. Consider a graph $G^{1,k}(f_N^{j-1})$. From Lemma 5, we can find a minimum weight acyclic restricted path within this graph, $\pi_1 =< v_1 \overset{w_1}{\rightsquigarrow} v_2 >$ whose intermediate nodes belong to $\{s_{i+1}^{j-1}, f_{i+1}^{j-1}, \ldots, s_N^{j-1}, f_N^{j-1}\}$. From the assumption of homogeneity between $\Psi^{1,k}(f_N^j)$ and $\Psi^{1,k}(f_N^{j-1})$, every edge $x_1 \rightarrow x_2$ in $G^{1,k}(f_N^{j-1})$, where $x_1, x_2 \in \{s_{i+1}^{j-1}, \ldots, s_N^{j-1}, f_N^{j-1}\}$, has the same weight as an edge $g_{(1)}(x_1) \rightarrow g_{(1)}(x_2)$ in $G^{1,k}(f_N^j)$. This one-to-one correspondence between created edge sets implies that an acyclic restricted path $g_{(1)}\pi_1$ has the same weight w_1 as that of π_1, and $g_{(1)}\pi_1$ is a minimum weight acyclic restricted path among the acyclic restricted paths in $G^{1,k}(f_N^j)$ whose intermediate nodes belong to $\{s_{i+1}^j, \ldots, s_N^j, f_N^j\}$. Hence, $g_{(1)}(v_1) \xrightarrow{w_1} g_{(1)}(v_2) \in G^{1,k}(f_i^j)$ holds from Lemma 5. Because $\Psi^{1,k}(f_i^{j-1})$ and $\Psi^{1,k}(f_i^j)$ are semi-homogeneous, this edge also belongs to $\Psi^{1,k}(f_i^j)$.

Second, suppose that $v_3 \xrightarrow{w_2} v_4 \in \Psi^{1,k}(f_i^j)$, where $v_3 \neq v_0$,$v_4 \neq v_0$. Consider a graph $G^{1,k}(f_N^j)$. From Lemma 5, we can find a minimum weight acyclic restricted path within this graph, $\pi_2 =< v_3 \overset{w_2}{\rightsquigarrow} v_4 >$ whose intermediate nodes belong to $\{s_{i+1}^j, f_{i+1}^j, \ldots, s_N^j, f_N^j\}$. Again, from the one-to-one correspondence between created edge sets, a path $g_{(-1)}\pi_2$ has the same weight w_2 as that of π_2, and the path is also a minimum weight acyclic restricted path among the acyclic restricted paths in $G^{1,k}(f_N^{j-1})$ whose intermediate nodes belong to $\{s_{i+1}^{j-1}, \ldots, s_N^{j-1}, f_N^{j-1}\}$. Hence, $g_{(-1)}(v_3) \xrightarrow{w_2} g_{(-1)}(v_4) \in G^{1,k}(f_i^{j-1})$ holds from Lemma 5. Because $\Psi^{1,k}(f_i^{j-1})$ and $\Psi^{1,k}(f_i^j)$ are semi-homogeneous, this edge also belongs to $\Psi^{1,k}(f_i^{j-1})$.

Therefore, the following is proved where $v_1 \neq v_0$ and $v_2 \neq v_0$:

$$(v_1 \xrightarrow{w} v_2 \in \Psi^{1,k}(f_i^{j-1})) \iff (g_{(1)}(v_1) \xrightarrow{w} g_{(1)}(v_2) \in \Psi^{1,k}(f_i^j))$$

For cases where one of v_1 or v_2 is equal to v_0, the condition 3 or 4 in the definition of homogeneous edge sets may be proved in a similar way to the above one by using the definition of homogeneity between created edge sets and Lemma 5.

Therefore, the Claim 1 is proved. Then, from the transitivity of homogeneous relations, it is clear that the following holds:

$$\forall l : 2 \leq l \leq j-1 \ :: \ \forall i : 1 \leq i \leq N \ :: \ \Psi^{1,k}(f_i^l) \sim \Psi^{1,k}(f_i^j)$$

Claim 2: $\forall l : 2 \leq l \leq j-1 \ :: \ \forall i : 1 \leq i \leq N \ :: \ \Psi^{1,k}(s_i^l) \sim \Psi^{1,k}(s_i^j)$

For fixed l and i, we know that $\Psi^{1,k}(f_i^l) \sim \Psi^{1,k}(f_i^j)$ holds from claim 1. From this homogeneity, it is clear that $\Psi^{1,k}(s_i^l) \sim \Psi^{1,k}(s_i^j)$ holds from node elimination algorithms. That is, $\Psi^{1,k}(f_i^l)$ is obtained after eliminating f_i^l from $G(f_i^l)$, and $\Psi^{1,k}(f_i^j)$ is obtained after eliminating f_i^j from $G(f_i^j)$. ■

Proof of Theorem 1: Let $G_b(V_b, E_b)$ denote a basis graph obtained from an initial constraint graph for a cyclically constrained job set.

Claim: If the Algorithm 3 applied to $G_b(V_b, E_b)$ doesn't terminate within n^2-n+2 loop iterations, then there exists a negative weight cycle in $G^{1,k}(f_N^k)$ for $k \gg n^2$.

Suppose that the algorithm doesn't terminate within n^2-n+2 loop iterations. From Proposition 6, we know that $\Psi^{1,k}(f_N^{k-2})$, $\Psi^{1,k}(f_N^{k-3})$, ..., $\Psi^{1,k}(f_N^1)$ are semi-homogeneous. Thus, $G^i_{out}(V_{b,1} \cup \{v_0\}, E^i_{out})$, $2 \leq i \leq n^2-n+2$, are semi-homogeneous, too. This means that after each loop iteration for $i \geq 3$ in the algorithm, there exists at least one edge in $G^i_{out}(V_{b,1} \cup \{v_0\}, E^i_{out})$, $3 \leq i \leq n^2-n+2$, whose weight has been reduced from the corresponding one in $G^{i-1}_{out}(V_{b,1} \cup \{v_0\}, E^{i-1}_{out})$. If not, then the algorithm should have been completed within n^2-n+2 loop iterations at step $3-(d)$, because homogeneous created edge sets are already found, which is against the assumption. For the purpose of clarity, each node $v_i (\in V_b)$ used in this proof will be denoted as v_i^j to represent that v_i belongs to a node set V_b in a graph $G^j_{in}(V_b, E^j_{in})$, or to a node set $V_{b,1} \cup \{v_0\}$ of a graph $G^j_{out}(V_{b,1} \cup \{v_0\}, E^j_{out})$.

Let $v_1^{n^2-n+2} \longrightarrow v_2^{n^2-n+2}$, $v_1, v_2 \in V_{b,1} \cup \{v_0\} (v_1 \neq v_2)$, denote one such edge in $G^{n^2-n+2}_{out}(V_{b,1} \cup \{v_0\}, E^{n^2-n+2}_{out})$ whose weight is less than that of the corresponding edge in $G^{n^2-n+1}_{out}(V_{b,1} \cup \{v_0\}, E^{n^2-n+1}_{out})$. Equivalently, from the cyclic operation performed at step $3-(f)$ in Algorithm 3 we can say that $v_1^{n^2-n+2} \longrightarrow v_2^{n^2-n+2}$ is an edge in $G^{n^2-n+2}_{out}(V_{b,1} \cup \{v_0\}, E^{n^2-n+2}_{in})$ whose weight is less than or equal to

- $w-1$, if the edge doesn't connect v_0.
- $w-L-1$, if the edge is from v_0.
- $w+L-1$, if the edge is to v_0.

where w is a weight of an edge $(g_{(1)}(v_1))^{n^2-n+2} \longrightarrow (g_{(1)}(v_2))^{n^2-n+2}$ of $G^{n^2-n+2}_{in}(V_b, E^{n^2-n+2}_{in})$.

Let p_1 denote a minimum weight acyclic restricted path from $v_1^{n^2-n+2}$ to $v_2^{n^2-n+2}$ with a weight w_{12} in $G^{n^2-n+2}_{in}(V_b, E^{n^2-n+2}_{in})$ whose intermediate nodes belong to $V_{b,2}$. Note that no intermediate node, if there exists any, is equal to v_0. p_1 exists from Lemma 5. Then, after (n^2-n+2)-th loop iteration, the weight of $v_1^{n^2-n+2} \rightarrow v_2^{n^2-n+2}$ will be changed to w_{12} in $G^{n^2-n+2}_{out}(V_{b,1} \cup \{v_0\}, E^{n^2-n+2}_{out})$.

sub-claim 1: In $G^{n^2-n+2}_{in}(V_b, E^{n^2-n+2}_{in})$, p_1 has at least one edge connecting two different nodes that belong to $g_{(1)}(V_{b,1}) \cup \{v_0\}$.

Suppose the claim is not true. Then, p_1 is also a minimum weight acyclic restricted path from $v_1^{n^2-n+1}$ to $v_2^{n^2-n+1}$ with a weight w_{12} in $G^{n^2-n+1}_{in}(V_b, E^{n^2-n+1}_{in})$, since only the weights of edges connecting two different nodes of $g_{(1)}(V_{b,1}) \cup \{v_0\}$ may be reduced after each loop iteration of the algorithm. This contradicts to the definition of the path p_1.

Then, the following is proved. Here, it is assumed that v_3, v_4 $(v_3 \neq v_4)$ belong to $g_{(1)}(V_{b,1}) \cup \{v_0\}$, and thus $g_{(-1)}(v_3)$, $g_{(-1)}(v_4)$ belong to $V_{b,1} \cup \{v_0\}$.

sub-claim 2: There exists at least one edge in p_1, $v_3^{n^2-n+2} \xrightarrow{w_{34}} v_4^{n^2-n+2}$, $v_3, v_4 \in g_{(1)}(V_{b,1}) \cup \{v_0\}$, satisfying

$$w_{34} < w_{34}^{'}$$

where $w_{34}^{'}$ is a weight of an edge $v_3^{n^2-n+1} \longrightarrow v_4^{n^2-n+1}$ in $G^{n^2-n+1}_{in}(V_b, E^{n^2-n+1}_{in})$.

Suppose that the claim is not true. Then, all edges lying in p_1 that connect two nodes of $g_{(1)}(V_{b,1}) \cup \{v_0\}$ don't satisfy the above condition. In other words, all edge weights of p_1 in $G_{in}^{n^2-n+2}(V_b, E_{in}^{n^2-n+2})$ are not reduced compared to the edge weights of p_1 in $G_{in}^{n^2-n+1}(V_b, E_{in}^{n^2-n+1})$. This means that p_1 is also a minimum weight acyclic restricted path with a weight w_{12} in $G_{in}^{n^2-n+1}(V_b, E_{in}^{n^2-n+1})$, which is clear from Proposition 7. From Lemma 5 this implies that the weight of $v_1^{n^2-n+1} \rightarrow v_2^{n^2-n+1}$ in $G_{in}^{n^2-n+1}(V_b, E_{in}^{n^2-n+1})$ is equal to w_{12}. This contradicts to the definition of the path p_1. Therefore, sub-claim 2 is proved.

Hence, we know that in path p_1 there exists an edge $v_3 \xrightarrow{w_{34}} v_4$ whose weight is less than that of the corresponding edge $v_3^{n^2-n+1} \xrightarrow{w'_{34}} v_4^{n^2-n+1}$ in $G_{in}^{n^2-n+1}(V_b, E_{in}^{n^2-n+1})$.

From the cyclic operation performed at step $3-(f)$ in Algorithm 3 and from Lemma 5, we know that there exists a minimum weight acyclic restricted path from $g_{(-1)}(v_3)$ to $g_{(-1)}(v_4)$ in $G_{in}^{n^2-n+1}(V_b, E_{in}^{n^2-n+1})$ whose intermediate nodes belong to $V_{b,2}$ and which is equal to one of the following forms:

1. If $v_3 \neq v_0$ and $v_4 \neq v_0$,
$$(g_{(-1)}(v_3))^{n^2-n+1} \overset{w_{34}}{\rightsquigarrow} (g_{(-1)}(v_4))^{n^2-n+1}$$

2. If $v_3 = v_0$ and $v_4 \neq v_0$,
$$v_0^{n^2-n+1} \overset{w_{34}-L}{\rightsquigarrow} (g_{(-1)}(v_4))^{n^2-n+1}$$

3. If $v_3 \neq v_0$ and $v_4 = v_0$,
$$(g_{(-1)}(v_3))^{n^2-n+1} \overset{w_{34}+L}{\rightsquigarrow} v_0^{n^2-n+1}$$

Note that, if any edge weight in the above minimum weight acyclic restricted path is reduced, the weight of an edge $v_3 \rightarrow v_4$ in $G_{in}^{n^2-n+2}(V_b, E_{in}^{n^2-n+2})$ will also be reduced by at least the same amount after (n^2-n+1)-th loop iteration of Algorithm 3.

Hence, p_1 can be denoted as:

$$< v_1^{n^2-n+2} \rightsquigarrow v_3^{n^2-n+2} \xrightarrow{w_{34}} v_4^{n^2-n+2} \rightsquigarrow v_2^{n^2-n+2} >$$

where the edge $v_3^{n^2-n+2} \xrightarrow{w_{34}} v_4^{n^2-n+2}$ can be replaced by one of the above minimum weight paths. Then p_1 can be denoted as:

$$< v_1^{n^2-n+2} \rightsquigarrow < (g_{(-1)}(v_3))^{n^2-n+1} \rightsquigarrow (g_{(-1)}(v_4))^{n^2-n+1} > \rightsquigarrow v_2^{n^2-n+2} >$$

where the inner path, $< (g_{(-1)}(v_3))^{n^2-n+1} \rightsquigarrow (g_{(-1)}(v_4))^{n^2-n+1} >$, will be reduced to an edge $v_3^{n^2-n+2} \xrightarrow{w_{34}} v_4^{n^2-n+2}$ after (n^2-n+1)-th loop iteration if Algorithm 3 is applied to the above extended path.

Note that applying Algorithm 3 to this new path will produce an edge $v_1^{n^2-n+2} \xrightarrow{w_{12}} v_2^{n^2-n+2}$, and if some edge weight is reduced, w_{12} will be reduced, too.

From the above result and from $w_{34} < w'_{34}$, we know that the edge weight of $g_{(-1)}(v_3) \rightarrow g_{(-1)}(v_4)$ in a graph $G_{out}^{n^2-n+1}(V_{b,1} \cup \{v_0\}, E_{out}^{n^2-n+1})$ is:

- $w'_{34} - 1$ or less, if $v_3 \neq v_0$ and $v_4 \neq v_0$.

- $w_{34}^{'} - L - 1$ or less, if $v_3 = v_0$.
- $w_{34}^{'} + L - 1$ or less, if $v_4 = v_0$.

where $w_{34}^{'}$ is an edge weight of $v_3 \rightarrow v_4$ in $G_{in}^{n^2-n+1}(V_b, E_{in}^{n^2-n+1})$.

This enables us to repeatedly apply the same procedure to a new minimum weight acyclic restricted path $g_{(-1)}(v_3) \rightsquigarrow g_{(-1)}(v_4)$ in $G_{in}^{n^2-n+1}(V_b, E_{in}^{n^2-n+1})$. Therefore, we obtain the following extension of path p_1:

$$\begin{aligned} < v_1^{n^2-n+2} \rightsquigarrow < (g_{(-1)}(v_3))^{n^2-n+1} \quad &\rightsquigarrow \quad < (g_{(-1)}(v_5))^{n^2-n} \rightsquigarrow (g_{(-1)}(v_6))^{n^2-n} > \\ &\rightsquigarrow \quad (g_{(-1)}(v_4))^{n^2-n+1} > \rightsquigarrow v_2^{n^2-n+2} > \end{aligned}$$

where the intermediate nodes of $< (g_{(-1)}(v_5))^{n^2-n} \rightsquigarrow (g_{(-1)}(v_6))^{n^2-n} >$ in the above path belong to $V_{b,2}$ of $G_{in}^{n^2-n}(V_b, E_{in}^{n^2-n})$.

And, this extension may be continued until the following is obtained:

$$\begin{aligned} < v_1^{n^2-n+2} \quad &\rightsquigarrow \quad < (g_{(-1)}(v_3))^{n^2-n+1} \rightsquigarrow < (g_{(-1)}(v_5))^{n^2-n} \rightsquigarrow \dots \\ &\rightsquigarrow \quad < (g_{(-1)}(v_{2(n^2-n+1)-1}))^2 \rightsquigarrow (g_{(-1)}(v_{2(n^2-n+1)}))^2 > \rightsquigarrow \\ \dots \quad &\rightsquigarrow \quad (g_{(-1)}(v_6))^{n^2-n} > \rightsquigarrow (g_{(-1)}(v_4))^{n^2-n+1} > \rightsquigarrow v_2^{n^2-n+2} > \end{aligned}$$

Consider the following set of node pairs in $V_{b,1} \cup \{v_0\}$ of $G_{in}^{j}(V_b, E_{in}^{j})$, $2 \leq j \leq n^2 - n + 2$, that have been included in the extension of path p_1 at each iteration of the process.

$$\begin{aligned} \{(v_1^{n^2-n+2}, v_2^{n^2-n+2}), \quad &((g_{(-1)}(v_3))^{n^2-n+1}, (g_{(-1)}(v_4))^{n^2-n+1}), \dots, \\ &((g_{(-1)}(v_{2(n^2-n+1)-1}))^2, (g_{(-1)}(v_{2(n^2-n+1)}))^2)\} \end{aligned}$$

Note that this set has $n^2 - n + 1$ node pairs. Because there exist n nodes in $V_{b,1} \cup \{v_0\}$, there may exist only $n^2 - n$ distinct node pairs. Hence, there should exist at least one node pair that appears twice in the above node pair set. Let $(v_{i_1}^{j}, v_{i_2}^{j}), (v_{i_3}^{l}, v_{i_4}^{l})$, $l < j$, denote two such node pairs where $i_1 = i_3 \wedge i_2 = i_4$. Therefore, in the extension process of p_1 performed above, we should have encountered the following path:

$$< v_{i_1}^{j} \rightsquigarrow < v_{i_1}^{l} \rightarrow v_{i_2}^{l} > \rightsquigarrow v_{i_2}^{j} > \tag{18}$$

Because the extension process choose an edge $v_{i_1}^{j} \rightarrow v_{i_2}^{j}$ in $G_{in}^{j}(V_b, E_{in}^{j})$ whose weight is less than $v_{i_1}^{j-1} \rightarrow v_{i_2}^{j-1}$ at the $(n^2 - n + 2 - j + 1)$-th iteration of Algorithm 3, we know that the weight of an edge $v_{i_1}^{l} \rightarrow v_{i_2}^{l}$ is greater than the weight of the edge $v_{i_1}^{j} \rightarrow v_{i_2}^{j}$ since $j > l$.

This implies that there exists a path that reduces the the edge weight of $v_{i_1}^{j} \rightarrow v_{i_2}^{j}$ from that of $v_{i_1}^{l} \rightarrow v_{i_2}^{l}$ after j-th loop iteration in Algorithm 3. Then, from Proposition 7 we know that, after $l + k(j - l)$ loop iteration in Algorithm 3 where $k \geq 1$, the edge weight of $v_{i_1} \rightarrow v_{i_2}$ in the resulting graph will be reduced from the corresponding edge weight in the graph found after $l + (k - 1)(j - l)$ loop iteration. This means that the edge weight of $v_{i_1} \rightarrow v_{i_2}$ will be infinitely decreased. But, since every job has a release time and a deadline constraints,

this repeated process will eventually create a negative weight cycle during the variable elimination process applied to a constraint graph for $sched^{1,\infty}$.

This contradicts to the assumption, and proves Claim 1 and the theorem. ■