Real-Time Systems, 21, 199–217, 2001

Trade-Off Analysis of Real-Time Control Performance and Schedulability*

DANBING SETO
Pratt & Whitney, United Technologies Corp., 400 Main Street, MS 163-14 East Hartford, CT 06108

JOHN P. LEHOCZKY
Department of Statistics, Carnegie Mellon University, Pittsburgh, PA 15213

LUI SHA
Department of Computer Science, University of Illinois, 1304 W. Springfield Avenue, Urbana, IL 61801

KANG G. SHIN
Real-Time Computing Laboratory, Department of Electrical Engineering and Computer Science, The University of Michigan, Ann Arbor, MI 48109

Abstract. Most real-time computer-controlled systems are developed in two separate stages: controller design followed by its digital implementation. Computational tasks that implement the control algorithms are usually scheduled by treating their execution times and periods as unchangeable parameters. Task schedulability therefore depends only on the limited computing resources available. On the other hand, controller design is primarily based on the continuous-time dynamics of the physical system being controlled. The set of tasks resulting from this controller design may not be schedulable with the limited computing resources available. Even if the given set of tasks is schedulable, their overall performance may not be optimal in the sense that they do not make a full use of the computing resources. In this paper, we propose an integrated approach to controller design and task scheduling. Specifically, task frequencies (or periods) are allowed to vary within a certain range as long as such changes do not affect critical control functions such as the maintenance of system stability. We present an algorithm that determines the task frequencies such that a prescribed aspect of system performance is optimized subject to satisfaction of computing resource constraints. The tasks are then scheduled with the chosen frequencies. The proposed approach also addresses the issue of choosing controller processors.

Keywords: real-time control, task schedulability, resource management

1. Introduction

The control of physical systems has changed from analog to digital technology, and computer control has been applied to perform more complex, higher-level functions. In applications ranging from flight control to micro-surgery, real-time control plays a crucial role in coordination of the dynamics of these systems. Although the application domain of digital real-time control has been expanded significantly, there still remain many issues in control implementation before its full potential can be achieved. For example, the

* This research was supported in part by the Office of Naval Research under contract N00014-92-J-1524, by the Software Engineering Institute of Carnegie Mellon University, by the NSSN program, by the ISC program, and by the JSF program.

design of controllers and the scheduling of control tasks are usually considered separately, and this separation may result in suboptimal system designs. In this paper we investigate the interaction between control task performance and task scheduling.

A real-time computer-controlled system consists of two major parts: the physical system and the computer system. The physical system includes the physical plant to be controlled, sensors to provide information on the plant's real-time behavior, and actuators that drive the physical plant. The computing unit can be a computer (a CPU) or a computer network which generates control commands to the actuators in the physical system. Figure 1 shows two different computer-controlled systems, where one involves a single computer controlling one physical system, and the other has one computer controlling multiple physical systems. In the former case, there are multiple control algorithms executing concurrently, and the computation of each of the control laws is considered to be one task. In the latter system, all of the physical systems are running simultaneously, and generation of the control command for each one of them is a task. In both cases, the physical systems are sampled periodically, and the computation of the control commands must be finished within the sampling period of the corresponding physical systems. This set of periodic tasks needs to be scheduled carefully so that the overall systems will work properly.

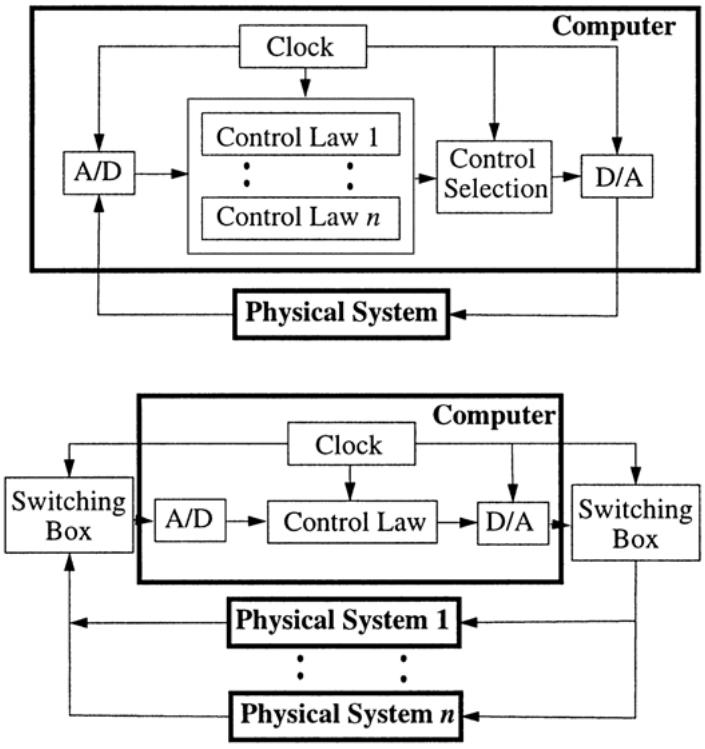

Figure 1. Examples of real-time computer-controlled systems.

Task scheduling is a fundamental issue in any real-time control algorithm implementation. A seminal contribution was made by Liu and Layland [2] who developed optimal static and dynamic priority scheduling algorithms for hard real-time task sets. They showed for such task sets that dynamic priority scheduling algorithms can achieve 100% schedulable utilization, while the optimal static priority algorithm, the rate monotonic algorithm, has a least upper bound of 69% on its schedulable utilization. Nevertheless, over the last two decades, significant progress has been made on generalizing these algorithms and making them suitable for real applications (e.g., Sha et al. [7]). Still, nearly all of these developments have assumed that the task set characteristics (e.g., computation times, periods and deadlines) are fixed and known.

There have been several papers which relate task scheduling and system performance. For example, Gerber et al. [1], addressed the issue of task design in relation to system performance; however, their focus was on distributed systems, and they did not use the performance index approach we present. Task scheduling and system performance have also been addressed by Locke [3] and other authors using best-effort scheduling (see Locke [3] for references). This approach is especially designed to handle transient overloads, and its premise is that a task will obtain a value which depends on the time at which it is completed. Again, this work did not focus on any particular application area such as control algorithm scheduling nor were performance indices introduced.

The control algorithm is usually designed to optimize some system performance index based on physical system properties, and this is often carried out with the assumption that the resulting controller will always be implementable on a digital controller computer. With a digital implementation, a controller can be designed using two distinct approaches. One approach is direct digital design which first discretizes the associated continuous-time system dynamics and then designs a control algorithm for the discretized system. The second approach is continuous-time design followed by digitization. In this approach, the control algorithm is designed based on the continuous-time system dynamics, and the resulting control law is digitized for implementation on a computer. Neither of these design strategies takes into account any limitations on the available computing resources.

When there are insufficient computing resources for the real-time control computations, one may naturally think of adding additional processors or replacing the computer with a faster one. Both approaches certainly solve the computing resource problem, but they may be neither efficient nor economical. For example, in the automotive industry, resource constraints are common due to cost concerns and the limited physical space inside each automobile. The reduction of overall resource usage without sacrificing system performance is one of the challenging software issues that needs to be addressed. On the other hand, when the computing resources are ample for scheduling the given set of tasks, the issue becomes how to best exploit the extra computing resources to further improve the control system's performance. In summary, a better approach to real-time computer-controlled system design would be to optimize the global system performance by simultaneously considering both control performance and computing resource availability.

The proposed integrated approach requires knowledge of the relationship between control system performance and sampling frequency (inverse of task period), and real-

time scheduling theory. In this paper, we assume that (i) the control algorithm has been developed using a continuous-time design followed by digitization, and (ii) the resulting algorithm is ''optimal'' (in the sense of a certain given objective function). To implement such a control strategy, we would like the sampling frequency to be as high as possible in order to make a better match between the (optimal) continuous-time control and its digital implementation. Note, however, that the limitations on computing resources shared among multiple tasks impose an upper bound on the sampling frequency for each periodic task. These upper bounds, one for each periodic task, must be considered in the integrated design approach, while they need not be considered in standard control designs. On the other hand, to correctly capture and control the system dynamics, the sampling frequencies are normally chosen to be 5 to 10 times the corresponding system's characteristic frequencies. This requirement gives a lower bound on the sampling frequency from the control system point of view.

By allowing the sampling frequencies to vary within the ranges defined by the lower and upper bounds mentioned above, we can actually enhance periodic task schedulability. That is, one can change the periods of some of the tasks in the given set (within these ranges) so that all of the periodic tasks in the set become schedulable, if the initial task periods make the task set unschedulable. We will adjust the task frequencies to optimize the overall system control performance, subject to two constraints: (1) the lower bounds on task frequencies and (2) the limitations on available computing resources.

The paper is organized as follows. In Section 2, we briefly review some of the basic concepts in control theory, especially the optimization of control system performance and digital control implementation. We describe in detail the rationale for combining control design and its digital implementation (i.e., task scheduling) in the design of real-time computer-controlled systems. The main results of this paper are presented in Section 3, where we derive an algorithm for choosing the optimal task frequencies such that all the tasks are schedulable with a dynamic priority assignment scheduling algorithm, and the system performance using the digital control implementation is optimized for the limited computing resources. The issue of choosing processors is discussed through examples. Conclusions are drawn in Section 4.

2. Control Design and Its Digital Implementation

Here we will briefly review some of the relevant concepts in control theory. In particular, we will first present an example to demonstrate the relationship between the control system performance index and the control task frequency when the control input is to be produced by a digital computer. Then we discuss in general the issues of optimizing system performance and digital control implementation.

To illustrate the effect of sampling frequency on system performance, we have chosen an actual real-time control application, a bubble control system. Such a system is a simplified model designed to study diving control in submarines. For a discussion of real-time computing issues in real submarines, the reader may refer to Molini et al. [5]. The bubble control system considered here consists of a tank filled with water and a diver, an inverted cup partially filled with air and immersed in the water. Depth control of the diver

is achieved by adjusting the air volume inside the diver, which is controlled by adjusting the piston connected to the air bubble. A schematic diagram of the system is given in Figure 2.

Let $x = (x_1, x_2, x_3) = (y, \dot{y}, h - h_e)$, where h_e is the air-volume height at equilibrium state. The equations of motion can be written as:

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -c_1|x_2|x_2 - c_2 x_3 \\ \dot{x}_3 &= -a(x)x_2 - b(x)u \end{aligned} \tag{1}$$

where u is the control variable defined as the piston velocity $\dot{l}$, c_1 and c_2 are positive coefficients of the water resistance and buoyancy, respectively, and $a(x)$ and $b(x)$ are positive definite functions obtained from the law of conservation of mass for the air. Suppose the control objective is to drive the diver to move at a given speed v_d. Then, a tracking problem can be formulated, for example, to have the diver follow a reference trajectory

$$y_r = y(0) + v_d t$$

From the control design point of view, the control goal is to make $|y(t) - y_r(t)| = 0$ as $t \to \infty$. Moreover, we wish to design the control u, the law for adjusting the piston velocity, such that the diver trajectory y converges to the reference y_r as fast as possible. Apparently, if the piston can be moved arbitrarily fast, it is possible to drive the diver to follow the reference at an arbitrarily fast convergence rate. However, due to the physical limitation, all the pistons have an upper bound on their velocity, and therefore, the tracking problem must be solved with a limited control regime. In other words, the control objective can be restated as: drive the diver such that its trajectory converges to

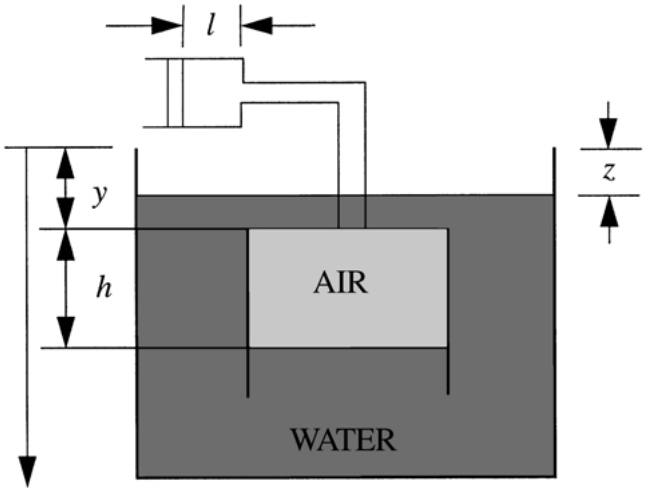

Figure 2. A schematic diagram of the bubble control system.

the reference as fast as possible with limited control. To put such objective in the perspective of optimal control, we define an objective function, or a performance index as

$$J(u) = \int_0^{t_f} \left\{ r[u(t)]^2 + (y(t) - y_r(t))^2 \right\} dt$$

where $r > 0$ is a weighting constant. The physical meaning of this performance index can be interpreted as a measure of the total cost of control and tracking error generated in the time period $[0, t_f]$ by the control u. By minimizing this performance index with a properly designed control law u, we can make both the tracking error and the control cost to be small, which results in a fast convergence of the diver trajectory to the reference with a small amount of control. Formally, the optimization problem can be stated as: find the optimal control function u^* such that

$$J^* = J(u^*) = \min_u J(u)$$

subject to: Equation (1),

$$H_0 \le y \le H - H_c, \quad 0 \le h \le H_c,$$

$$|u| \le u_{\max}$$

where H_0, H_c and H are the water level, the height of the diver, and the height of the tank, respectively, and $u_{\max}$ is the maximum piston velocity.

When a continuous-time control law u is implemented digitally, the system performance will deviate from the value obtained with the continuous-time control, J^*, and the deviation will depend on the sampling frequency. Let f and $T = 1/f$ be the sampling frequency and period, respectively. Suppose the control is implemented as a zero-order hold, i.e., $u(t) = u(kT), \forall t \in [kT, (k+1)T)$. Then the performance index with the digitized control can be written as

$$J_D(f) = \sum_{k=0}^{n-1} \int_{kT}^{(k+1)T} [r(u(kT))^2 + (y(t) - y_r(t))^2] dt$$

where $y(t)$ is the trajectory of the diver under the digitized control. Hence we conclude that the performance index of a given control law u with a digital implementation is a function of the sampling frequency (period). Figure 3 shows the simulation result of $\Delta J^*(f) = J_D^*(t) - J^*$ for the bubble control system with f_m representing the lower bound of frequency.

Remark 2.1 Figure 3 illustrates that there can be a wide range over which the sampling frequency can vary. Specifically, any frequency above the lower bound f_m will keep the system stable with a particular value of the performance index. Furthermore, the performance index is a convex function of sampling frequency, and it is this convexity property that allows us to select optimal frequencies for a set of tasks, such as in the computation of the bubble control command, and schedule them in a single CPU. This will be elaborated on in the next section.

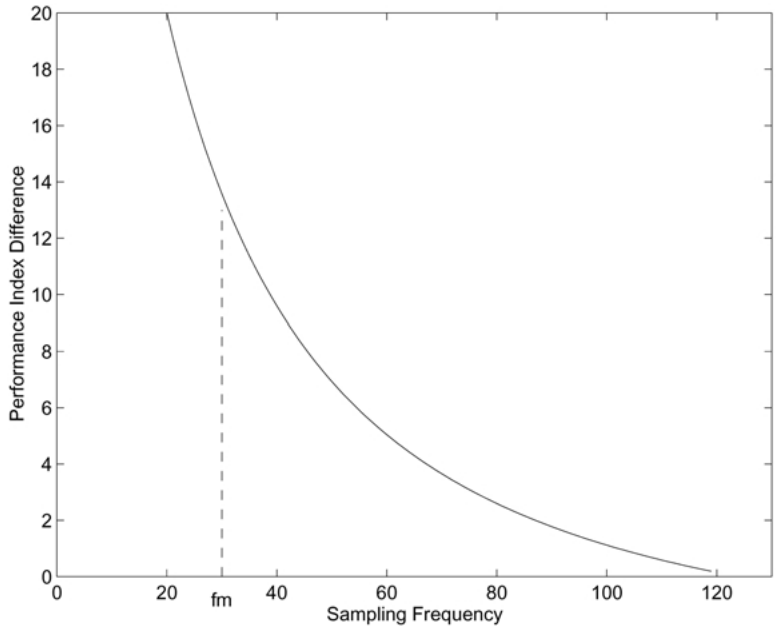

Figure 3. $\Delta J^*(f)$ for the bubble control system.

We now extend the results obtained from the above example to general real-time control systems. As shown in the example, the system performance is usually measured by a performance index, and the control algorithm is often derived to optimize this index subject to the system dynamics and constraints. For example, the objective for a radar system is to track a target, and the performance index would be a measure of the tracking error. For this example, one would like to design a control algorithm to minimize the performance index. For mechanical systems, on the other hand, the performance index might be some measure of the total work the system produces, and in this case we would want to maximize the performance index. Other examples may involve minimization of time (e.g., minimizing the system response time) or energy (e.g., minimizing the cost). The problem of optimizing the performance index can be stated formally as follows.

$$(\max_{u\in\Omega})\ \min_{u\in\Omega} J(u) = (\max_{u\in\Omega})\ \min_{u\in\Omega}\left[S(x(t_f),t_f) + \int_0^{t_f} L(x(t),u(t),t)dt\right] \tag{2}$$

$$\text{subject to:}\quad \dot{x} = f(x(t),u(t),t) \tag{3}$$

$$c(x(t),u(t)) \le 0 \tag{4}$$

where $J(u)$ is the system performance index, $\Omega \subset R^m$ is a set containing valid control values, $[0,t_f]$ is the time interval of interest, $S(\cdot)$ and $L(\cdot)$ are the weight (or cost) functions depending on system states, time and control inputs. Equation 3 describes the dynamics of the underlying system with state $x(t) \in R^n$ and control input $u(t) \in \Omega$ for each

$t > 0$, while Equation (4) represents the constraints on the system trajectory and the control input with $c(\cdot) \in R^p$. The complete statement of the control optimization problem can be summarized as: find the optimal control such that the performance index defined by Equation (2) will be minimized (maximized) subject to the system dynamics and constraints given by Equation (3) and Equation (4).

The optimal control for the problem described above can be derived by direct digital design or continuous-time design and then digitization. We adopt the latter design approach; similar results can be obtained for the direct digital design. Suppose the optimization problem in Equations (2)–(4) can be solved with the optimal control $u^*(t)$ resulting in performance index J^*. Then, the control implementation determines if we can actually obtain the performance for which the controller is designed. Discretizing the control input $u^*(t)$ in the time domain with sampling period T and zero-order hold, we obtain the performance index with the digitized control as

$$J_D^*(f) = S(x^*(t_f), t_f) + \sum_{k=0}^{n-1} \int_{kT}^{(k+1)T} L(x^*(t), u^*(kT), t)dt \tag{5}$$

where again, $f = 1/T$ is the sampling frequency, $t_f = nT$, and $x^*(t)$ is determined by

$$\dot{x}^*(t) = f(x^*(t), u^*(kT), t), \ kT \le t \le (k+1)T, \ k = 0, \ldots, n-1$$

Equation (5) shows that the performance index is a function of the sampling frequency, and Figure 4 illustrates possible performance indices. In this paper, we will consider only

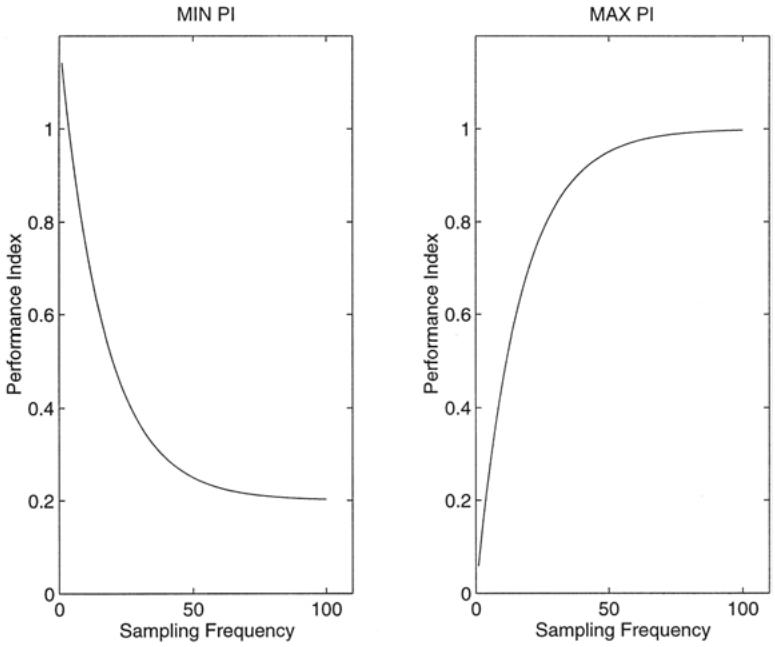

Figure 4. Control system performance indices versus sample frequency.

monotone, convex or concave functions as shown in Figure 4. The physical meaning of these functions is clear: as the sampling frequency increases, the performance index with digitized optimal control (PIDOC) will tend to converge to the performance index with continuous-time optimal control (PICOC). On the other hand, as the sampling frequency decreases, the difference between PIDOC and PICOC will increase, and eventually the system will become unstable. To prevent this from happening, a lower bound on the sampling frequency must be imposed. For convenience, we will consider the difference $\Delta J^*(f) = |J_D^*(f) - J^*|$. Clearly, $\lim_{f\to\infty} \Delta J^*(f) = 0$. Since the PIDOC $J_D^*(f)$ considered in this paper is assumed to be a monotone, convex or concave function as shown in Figure 4 and J^* is a constant, the difference $\Delta J^*(f)$ will always be a monotone, convex function. Moreover, we will approximate the function $\Delta J^*(f)$ by an exponential decay function, i.e., $\Delta J^*(f) = \alpha e^{-\beta f}$, where α is a magnitude coefficient and β is the decay rate. While there are various ways to obtain the approximation, we consider the least-square fitting approach. Let $\Delta J_s^*(f)$ be a simulation result with $n+1$ points at $(f_m, \Delta J_s^*(f_m))$, $(f_1, \Delta J_s^*(f_1)) \ldots, (f_n, \Delta J_s^*(f_n))$, and $L = \sum_{i=1}^{n} [\Delta J_s^*(f_i) - \alpha e^{-\beta f_i}]^2$, with $\alpha = \Delta J_s^*(f_m) e^{\beta f_m}$ by setting $\Delta J_s^*(f_m) = \alpha e^{-\beta f_m}$. Then by searching for β such that L is minimized, we obtain an exponential expression for $\Delta J^*(f)$.

Remark 2.2 The algorithm developed in this paper is not restricted to the optimal control problem. Let J and $J_D(f)$ be the performance indices generated by a continuous-time control which may not be optimal and its digital implementation at sampling frequency f, respectively. Then the algorithm can be applied provided $\Delta J(f) = |J_D(f) - J|$ is convex and monotonically decreasing. We shall therefore use $\Delta J(f)$ in the rest of the paper. Again, note that ΔJ is a function of both the sampling frequency f and the control function u. Since our goal is to investigate the effect of sampling frequency on the performance index for a given control function u, we omit u from the argument list of ΔJ.

For multiple physical systems controlled by a single CPU, we consider designing the control algorithm for each system such that the performance of the overall system is optimized. Suppose there are n control systems under consideration and each of them is required to achieve a control goal independent of the others. Apparently, for each control system to have an optimal performance, a control law can be derived by solving the optimization problem as stated above. To determine the sampling frequency for each control system, we have to make a trade-off among the systems. In other words, if we make one system's behavior close to the performance it would have with the continuous-time optimal control by allowing it to be sampled at a high frequency, then the rest systems will suffer from relatively slow sampling rates due to the limited computing resources. Therefore, we will determine the sampling frequency for each control system with respect to a weighted sum of the performance indices

$$\Delta J(f_1, \ldots, f_n) = \sum_{i=1}^{n} w_i \Delta J_i(f_i) \tag{6}$$

which is defined as the performance index of the overall system. In Equation (6), w_i, $i = 1, \ldots, n$, are the design parameters determined from the application, for instance, the importance of the associated control system to have a better performance than the others. For this set of control systems, we will develop an algorithm to find the optimal choice of $f_1, \ldots, f_n$ such that $\Delta J(f_1, \ldots, f_n)$ is minimized subject to the availability of the computing resource. In this paper, the algorithm is developed for a general class of control systems in which the functions $\Delta J(f)$ are monotonically decreasing and convex. Many control systems belong to this class; for example, the aircraft landing control application studied in Shin et al. [8] offers a second example.

Remark 2.3 The tasks to be scheduled may not all be control-related tasks. For example, some tasks might involve data processing and display. The periods for some of these tasks may not be changeable, while others may not have associated performance indices. When the system involves such tasks, we will schedule them as a part of the real-time task set without optimizing their periods.

3. Task Scheduling

In this section, we address the issue of determining the task frequencies such that all the tasks are schedulable and the system performance indices are optimized. Specifically, for a given set of tasks $\{\tau_1, \ldots, \tau_n\}$ with minimal allowable frequencies, $f_{m1}, \ldots, f_{mn}$, and performance indices

$$\Delta J_i(f_i) = \alpha_i e^{-\beta_i t_i}, \quad i = 1, \ldots, n \tag{7}$$

we choose the frequencies $f_i \geq f_{mi}$ such that all the tasks are schedulable and the performance index for the overall system, $\sum_{i=1}^{n} \omega_i \Delta J_i(f_i)$, is minimized. In this paper, we will concentrate on developing the algorithm for selection of task frequencies with dynamic priority assignment (DPA) scheduling schemes. Determination of the optimal frequencies for tasks schedule with a static priority algorithm (such as rate monotonic) involves a different approach which is out of the scope of this paper. The static priority case is addressed in Seto et al. [6].

To guarantee the schedulability of a task set $\{\tau_1, \ldots, \tau_n\}$ with a DPA scheduling algorithm, the frequencies of the tasks $f_1, \ldots, f_n$ need to be chosen such that $\sum_{i=1}^{n} C_i f_i \leq A$, where C_i is the execution time of task τ_i and $0 < A \leq 1$ is the given utilization. If the task set contains two types of tasks, ones with variable frequencies and the others with unchangeable frequencies, we need to modify the given utilization by subtracting $\sum_{j \in J} C_j f_j$ from A, where J is the collection of indices of the tasks whose periods are fixed. So far as the determination of the frequencies is concerned, we only consider the tasks which have variable frequencies and performance indices characterized by Equation (7). Then the optimization problem can be stated as:

$$\min_{(f_1, \ldots, f_n)} \Delta J(f_1, \ldots, f_n) = \min_{(f_1, \ldots, f_n)} \sum_{i=1}^{n} w_i \alpha_i e^{-\beta_i f_i} \tag{8}$$

$$\text{subject to } \sum_{i=1}^{n} C_i f_i \le A, \quad f_i \ge f_{mi}, \quad i = 1, \ldots, n \tag{9}$$

Before formally solving this nonlinear constrained optimization problem, we offer some insight into the form of the optimal solution. If $\sum_{i=1}^{n} C_i f_{mi} < A$, then at least one of the frequencies can be increased above its minimum value. The decrease in the system performance index per unit increase in utilization of task i, i.e., $\partial \Delta J / \partial (C_i f_i)$, is given by $U_i(f_i) = \Gamma_i e^{-\beta_i f_i}$ where $\Gamma_i = \frac{w_i \alpha_i \beta_i}{C_i}$ and task i has frequency f_i. We label the tasks so that

$$U_1(f_{m1}) \le U_2(f_{m2}) \le \cdots \le U_n(f_{mn}) \tag{10}$$

Consequently, task n should have its frequency increased first. As long as sufficient processor utilization is available, this increase should continue until the point at which $U_n(f_n) = U_{n-1}(f_{m(n-1)})$. Both f_n and f_{n-1} should then be increased to maintain the equality $U_n(f_n) = U_{n-1}(f_{n-1})$. Only these two frequencies will be increased until $U_n(f_n) = U_{n-1}(f_{n-1}) = U_{n-2}(f_{m(n-2)})$. At this point, if additional processor utilization is available, all three of these frequencies will be increased to maintain equality in the marginal performance indices. This process continues until either processor utilization is exhausted or all task frequencies are increased above their minimum value. If p of the frequencies are set to their minimum values for some $1 \le p \le n-1$, then total processor utilization of $A - \sum_{i=1}^{p} C_i f_{mi}$ is available for tasks $p+1, \ldots, n$. The optimal frequencies are easily found using standard Lagrange multiplier methods and are given by $f_i^* = \frac{1}{\beta_i}(\ln \Gamma_i - Q)$, where $p+1 \le i \le n$ and Q is chosen to ensure the total processor utilization is A. This method is illustrated in Figure 5 and formalized in the following proposition.

Proposition 3.1 Given a set of tasks $\tau_1, \ldots, \tau_n$ to be scheduled in a single CPU with the objective function given by (8) and constraints given by (9). Suppose $\sum_{i=1}^{n} C_i f_{mi} \le A$. Then there exists a unique optimal set of sampling frequencies, $f_1^*, \ldots, f_n^*$, given by

$$\begin{aligned} f_i^* &= f_{mi}, && i = 1, \ldots, p, \\ f_j^* &= \frac{1}{\beta_j}(\ln \Gamma_j - Q), && j = p+1, \ldots, n \end{aligned} \tag{11}$$

where

$$\Gamma_j = \frac{w_j \alpha_j \beta_j}{C_j}, \quad Q = \frac{1}{\sum_{i=p+1}^{n} \frac{C_i}{\beta_i}} \left(\sum_{i=1}^{p} C_i f_{mi} + \sum_{i=p+1}^{n} \frac{C_i}{\beta_i} \ln \Gamma_i - A \right)$$

$f_{m1}, \ldots, f_{mn}$ are ordered to satisfy (10), and $p \in \{1, \ldots, n\}$ is the largest integer such that

$$\sum_{i=1}^{p} C_i f_{mi} + \sum_{i=p+1}^{n} \frac{C_i}{\beta_i} \left(\beta_p f_{mp} + \ln \frac{\Gamma_i}{\Gamma_p} \right) \ge A$$

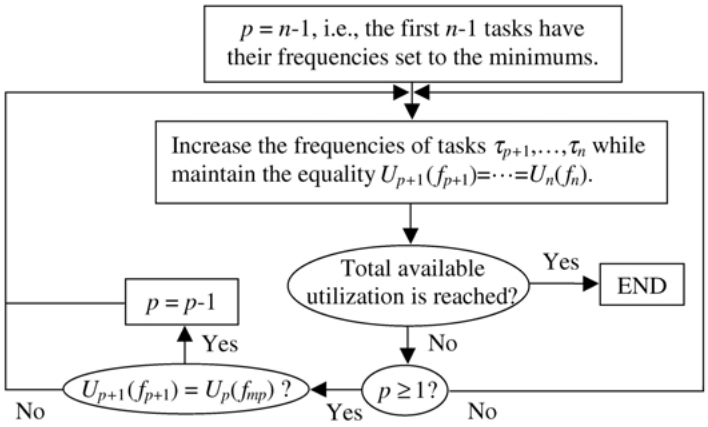

Figure 5. Algorithm of determining the optimal frequencies.

If no such integer p exists, then

$$f_j^* = \frac{1}{\beta_j}(\ln \Gamma_j - Q), \quad Q = \left[\sum_{i=1}^{n} \frac{C_i}{\beta_i} \ln \Gamma_i - A\right] \Big/ \sum_{i=1}^{n} \frac{C_i}{\beta_i}, \quad 1 \le j \le n$$

Proof: To solve the constrained optimization problem, we introduce $n+1$ Lagrange multipliers λ, $\lambda_i \ge 0$, it $>= 1, \ldots, n$, and minimize the augmented objective function

$$\Delta J(f_1, \ldots, f_n) = \sum_{i=1}^{n} w_i \alpha_i e^{-\beta_i f_i} + \lambda\left(\sum_{i=1}^{n} C_i f_i - A\right) + \sum_{i=1}^{n} \lambda_i (f_{mi} - f_i)$$

subject to (9), $\lambda\left(\sum_{i=1}^{n} C_i f_i - A\right) = 0$, and $\lambda_i(f_{mi} - f_i) = 0$, $i = 1, \ldots, n$. Since both the objective function and the constraint functions are convex, this optimization problem has a unique solution given by $(f_1^*, \ldots, f_n^*)$ if and only if the following Kuhn-Tucker conditions are satisfied,

$$\begin{cases} -\Gamma_i e^{-\beta_i f_i^*} + \lambda - \lambda_i / C_i = 0 \\ \sum_{i=1}^{n} C_i f_i^* - A \le 0, & f_{mi} - f_i^* \le 0 \\ \lambda\left(\sum_{i=1}^{n} C_i f_i^* - A\right) = 0, & \lambda_i (f_{mi} - f_i^*) = 0 \\ \lambda,\ \lambda_i \ge 0 \end{cases}$$

for all $i = 1, \ldots, n$. See Mangasarian [4] for details of the Kuhn-Tucker condition. Since $\lambda_i \ge 0$, $\Gamma_i > 0$, $\forall i$, we have

$$\lambda > 0, \quad \text{which implies} \quad \sum_{i=1}^{n} C_i f_i = A \tag{12}$$

Note that $\sum_{i=1}^{n} C_i f_{mi} \le A$ by assumption. If $\sum_{i=1}^{n} C_i f_{in} = A$, then $f_i^* = f_{mi}$, $1 \le i \le n$, which agrees with the proposition. It remains to consider the case $\sum_{i=1}^{n} C_i f_{mi} < A$, which

is assumed in the follows. In this case, some (or all) of $f_1, \ldots, f_n$ must be increased from their minimal values so that Equation (12) is satisfied. Next we will show that the process of increasing frequencies must begin with f_n. Suppose to the contrary that we start increasing f_j, $j < n$. Then $f_j > f_{mj}$ which results in $\lambda_j = 0$ and $\lambda = \Gamma_j e^{-\beta_j f_j}$. On the other hand,

$$\frac{\lambda_n}{C_n} = \lambda - \Gamma_n e^{-\beta_n f_{mn}} = \Gamma_j e^{-\beta_j f_j} - \Gamma_n e^{-\beta_n f_{mn}}$$

Since $\Gamma_j e^{-\beta_j f_{mj}} \leq \Gamma_n e^{-\beta_n f_{mn}}$ and $\Gamma_j e^{-\beta_j f_j} < \Gamma_j e^{-\beta_j f_{mj}}$, we conclude that $\Gamma_j e^{-\beta_j f_j} < \Gamma_n e^{-\beta_n f_{mn}}$, and $\lambda_n < 0$, which is not allowed. Thus, f_n has to be increased first. In this case, we have $f_n > f_{mn}$, $\lambda_n = 0$, and $\lambda = \Gamma_n e^{-\beta_n f_n}$. Clearly, λ decreases as f_n increases. Now consider the frequencies of the other tasks. Before f_j, $j < n$ can be increased, we have $\lambda = \Gamma_j e^{-\beta_j f_{mj}} + \lambda_j / C_j$, which has a minimum value $\Gamma_j e^{-\beta_j f_{mj}}$. If λ falls below $\Gamma_j e^{-\beta_j f_{mj}}$, f_j must be increased above f_{mj}. Therefore, when f_n is increased to the value at which $\lambda = \Gamma_{n-1} e^{-\beta_{n-1} f_{m(n-1)}}$ and the total utilization $\sum_{i=1}^{n-1} C_i f_{mi} + C_n f_n$ is still less than A, f_{n-1} will start to increase. Then both f_n and f_{n-1} will be increased with $\lambda = \Gamma_n e^{-\beta_n f_n} = \Gamma_{n-1} e^{-\beta_{n-1} f_{n-1}}$ until $\lambda = \Gamma_{n-2} e^{-\beta_{n-2} f_{m(n-1)}}$, in which case, f_{n-2} starts increasing if the total utilization is still less than A. This process will continue until the total utilization reaches A, which can be checked by finding the largest integer p such that

$$\sum_{i=1}^{p} C_i f_{mi} + \sum_{i=p+1}^{n} C_i f_i \geq A \quad \text{with} \quad \lambda = \Gamma_p e^{-\beta_p f_{mp}} \quad \text{and}$$

$$f_i = -\frac{1}{\beta_i} \ln\left(\frac{\lambda}{\Gamma_i}\right), i = p+1, \ldots, n$$

which is equivalent to

$$\sum_{i=1}^{p} C_i f_{mi} + \sum_{i=p+1}^{n} \frac{C_i}{\beta_i}\left(\beta_p f_{mp} + \ln\frac{\Gamma_i}{\Gamma_p}\right) \geq A$$

as given in the proposition. After the integer p has been identified, we conclude that the frequencies $f_1, \ldots, f_p$ must be chosen to be $f_{m1}, \ldots, f_{mp}$ and the remaining frequencies are computed from

$$\sum_{i=1}^{p} C_i f_{mi} + \sum_{i=p+1}^{n} C_i f_i = A \quad \text{and} \quad \lambda = \Gamma_p e^{-\beta_p f_p}$$

which results in Equation (11) in the proposition. If no $p \in \{1, \ldots, n\}$ exists, then all frequencies must be increased above their minimum values, which can be determined by solving $\sum_{i=1}^{n} C_i f_i = A$ and $\lambda = \Gamma_i e^{-\beta_i f_i}$, $\forall i$, as summarized in the statement of the proposition. ■

Remark 3.1 The solution to the optimization problem and the associated algorithm rely on the fact that both the objective function and the constraint region are convex. It is important to note that the constraint region is convex because we are assuming the use of

the earliest deadline first processor scheduling algorithm. This algorithm allows the processor to schedule periodic tasks with deadlines at the end of each task period up to 100% utilization regardless of the individual task periods and computation requirements. Had we assumed that processor scheduling was done using a static priority algorithm, then the constraint region would not in general be convex, and it would depend upon the individual task periods and computation requirements in a non-linear way. A more complicated search algorithm would need to be employed to determine the optimal set of task periods in this case. See Seto et al. [6] for details.

Proposition 3.1 provides a method for optimally determining the sampling frequency at each level of CPU utilization such that task schedulability with a DPA scheduling algorithm is guaranteed. Its application can be found in a wide range of real-time control systems. In fact, most control systems can have a flexible sampling frequency above the lower bound. This feature was defined as the control system deadline and discussed in detail in Shin and Kim [9] where the authors derived the number of consecutive control updates that can be missed without losing system stability. Proposition 3.1 presents an integrated approach to deriving the sampling frequencies for the tasks that optimize overall system performance subject to task schedulability and system stability.

Example 3.1 Consider an open-loop temperature control problem. Suppose there are five units whose temperatures need to be automatically controlled by one processor, and controlling the temperature in each unit is considered to be one task. The execution time C_i, the given frequency f_i and the minimum required frequency f_{mi} of each task are listed in Table 1.

According to the given data, the total utilization with the given frequencies will be $1.55 > 1$, and therefore, the tasks are not all schedulable. However, by investigating the underlying physical systems, we may find that these tasks are schedulable with a set of ''redesigned'' frequencies. In fact, if the minimum required frequencies were used, the task set would be schedulable with a DPA scheduling algorithm. Suppose the temperature for each unit is governed by the dynamic equation

$$\dot{\gamma}_i = -a_i\gamma_i + b_i u_i \tag{13}$$

where $\gamma_i(t)$ is the temperature difference between the i-th unit and the ambient temperature with $\gamma(0) = 0$; a_i and b_i are positive constants depending on the insulation of the unit; u_i is the rate of heat (cold air) supplied to the unit. Suppose we need to change

Table 1. Data for temperature control scheduling.

	C_i (ms)	f_{mi} (Hz)	f_i (Hz)	Utilization (%)
Unit 1	10	20	30	30
Unit 2	15	12.5	20	30
Unit 3	20	10	20	40
Unit 4	25	6	10	25
Unit 5	30	4	10	30

the temperature in the i-th unit, and we require such a change to be completed in no more than t_f time units and to consume a minimum amount of fuel. Let γ_{di} be the difference between the desired temperature and the ambient temperature. We also require that $|\gamma_i(t_f) - \gamma_{di}| \leq \delta_i$. Then, the optimization problem can be formulated as

$$\min_{u_i} J_i = \min_{u_i}\left[\frac{1}{2}p_i(\gamma_i(t_f) - \gamma_{di})^2 + \frac{1}{2}\int_o^{t_f} u_i^2(t)dt\right]$$

where p_i is a weight coefficient. Then, the continuous-time optimal control and the final state are determined by

$$u^*(t) = \frac{\gamma_{di}p_i a_i b_i e^{a_i t}}{a_i e^{a_i t_f} + p_i b_i^2 \sinh(a_i t_f)}, \quad \gamma_i^*(t_f) = \frac{\gamma_{di}p_i b_i^2 \sinh(a_i t_f)}{a_i e^{a_i t_f} + p_i b_i^2 \sinh(a_i t_f)}$$

Digitizing $u^*(t)$ with sampling period T_i and choosing $p_i \gg 1$, we obtain the approximations:

$$\gamma_i^*(t_f) = \gamma_{di}, \quad \text{and} \quad J_{di} = \frac{1}{2}p_i\gamma_{di}^2\left(\frac{1 - e^{-a_i T_i}}{1 + e^{-a_i T_i}}\right)^2$$

To satisfy the condition $|\gamma_i(t_f) - \gamma_{di}| \leq \delta_i$, we need

$$\gamma_{di}\left(\frac{1 - e^{-a_i T_i}}{1 + e^{-a_i T_i}}\right) \leq \delta_i \quad \Rightarrow \quad T_i \leq \frac{1}{a_i}\ln\frac{\gamma_{di} + \delta_i}{\gamma_{di} - \delta_i}$$

Furthermore, by approximating $1 + e^{-a_i T_i} \approx 3/2$, $1 - e^{-a_i T_i} \approx e^{-\beta_i f_i}$ with $\beta_i = 1/a_i$ and $f_i = 1/T_i$, we finally obtain

$$\Delta J_i = \frac{2}{3}e^{-\beta_i f_i} \quad \text{with} \quad f_{mi} = \frac{a_i}{\ln\frac{\gamma_{di} + \delta_i}{\gamma_{di} - \delta_i}}$$

To determine the sampling frequencies in relation with the system performance and task schedulability, we assume the physical parameters as given in Table 2.

Following the algorithm in Proposition 3.1, we determine the optimal frequency for each task as follows. Let

$$F(p) = \sum_{i=1}^{p} C_i f_{mi} + \sum_{i=p+1}^{5} \frac{C_i}{\beta_i}\left(\beta_p f_{mp} + \ln\frac{\Gamma_i}{\Gamma_p}\right)$$

Table 2. Data for temperature control tasks.

	α_i	β_i	C_i (ms)	f_{mi} (Hz)	w_i
Unit 1	2/3	0.3	10	20	1
Unit 2	2/3	0.4	15	12.5	2
Unit 3	2/3	0.5	20	10	3
Unit 4	2/3	0.6	25	6	4
Unit 5	2/3	0.7	30	4	5

Then, a simple calculation shows

$$F(5) = 0.8875 < 1, \quad F(4) = 0.9 < 1, \quad F(3) = 1.04 > 1$$

Therefore, we will assign

$$f_i = f_{mi}, \quad \text{for } i = 1, 2, 3, \quad \text{i.e.,} \quad f_1 = 20 \text{ Hz}, \quad f_2 = 12.5 \text{ Hz}, \quad f_3 = 10 \text{ Hz}$$

and compute f_4 and f_5 to be

$$f_4 = (\ln \Gamma_4 - Q)/\beta_4 = 7.97 \text{ Hz}, \quad f_5 = (\ln \Gamma_5 - Q)/\beta_5 = 7.1 \text{ Hz}$$

This choice of frequencies yields a total utilization 99.97%, and the final set of tasks is schedulable. Furthermore, these frequencies yield a performance index difference $\Delta J = 0.0696$ which is better than the one obtained with the minimal frequencies, $\Delta J = 0.2997$.

Example 3.2 Consider the bubble control system discussed in the previous section. Suppose four such systems with different physical dimensions are installed on an underwater vehicle to control the depth and orientation of the vehicle, and they are controlled by one on-board processor. For each bubble control system i, let C_i be the control task execution time in each sampling period, f_{mi} be the lower bound on sampling frequency, and w_i be the weight. The following data are given for the control design and scheduling problem: $\Delta J_i = \alpha_i e^{-\beta_i f_i}$, $i = 1, \ldots, 4$, and where the frequencies, f_i, $i = 1, \ldots, 4$, must be determined. The coordinator is the control unit which coordinates the bubble subsystems to perform a desired mission, for instance, changing depth, roll or pitch angles. The frequency for this particular task is fixed, and it is determined by the underlying coordination requests. For example, to keep the roll angle θ_r within a certain range, say $|\theta_r| < \theta_{rm}$, we may want to choose the task period to be $\theta_{rm}/\max(|\dot{\theta}_r|)$.

A simple calculation shows that the total CPU utilization of the overall bubble system is 63% when the minimum task frequencies are assigned, and the total CPU utilization available for the bubble systems is 95%, excluding the utilization of the coordinator. These imply that for any $A \in [0.63, 0.95]$, all the tasks are schedulable with a DPA scheduling algorithm, using the task frequencies determined by the algorithm described in Proposition 3.1. Again, letting

$$F(p) = \sum_{i=1}^{p} C_i f_{mi} + \sum_{i=p+1}^{4} \frac{C_i}{\beta_i} \left(\beta_p f_{mp} + \ln \frac{\Gamma_i}{\Gamma_p} \right)$$

Table 3. Data for bubble control scheduling.

	α_i	β_i	C_i (ms)	f_{mi} (Hz)	w_i	f_i (Hz)
b1	1	0.5	10	15	5	—
b2	1	0.7	10	10	3	—
b3	1	0.3	10	18	2	—
b4	1	0.1	10	20	1	—
coordinator	—	—	5	—	—	10

we find

$$F(4) = 0.63, \quad F(3) = 0.7908, \quad F(2) = 0.8371, \quad F(1) = 0.8852$$

Hence we conclude (also as indicated in Figure 6): when $A \in (0.8852, 0.95]$, there is no frequency chosen to be the corresponding lower bound. When $A \in (0.8371, 0.8852]$, there is one frequency that needs to assume its minimum value, i.e., $f_1 = 15$ (Hz). When $A \in (0.7908, 0.8371]$, two frequencies are set to their minimum value, i.e., $f_1 = 15$ (Hz) and $f_2 = 10$ (Hz). When A is reduced further to range $(0.63, 0.7908]$, three frequencies must be set to their lower bounds, namely, $f_1 = 15$ (Hz), $f_2 = 10$ (Hz) and $f_3 = 18$ (Hz). Finally, when $A = 0.63$, all the frequencies must be assigned their minimum values. Table 4 shows, at different utilization levels, the optimal frequencies computed from the algorithm in Proposition 3.1 and the resulting performance index difference.

The results in Table 4 demonstrate that the control system performance will be improved through increasing tasks' sampling frequencies as more computing resource is available. This is also illustrated in Figure 6 as the relation between the performance index difference ΔJ and the value of A.

Examples 3.1 and 3.2 outline general guidelines for the selection of processors and the determination of processor load. As shown in Example 3.1, by using the optimal sampling frequencies, not only is the task set made schedulable from a unschedulable set, but also the control system performance index difference is reduced from $\Delta J = 0.2997$, when the minimal frequencies are used, to $\Delta J = 0.0696$, a 76% improvement. In this case, if a faster processor is chosen, the gain on performance will be within the extra 24% range, and a significant change on computing speed is required to realize the gain on

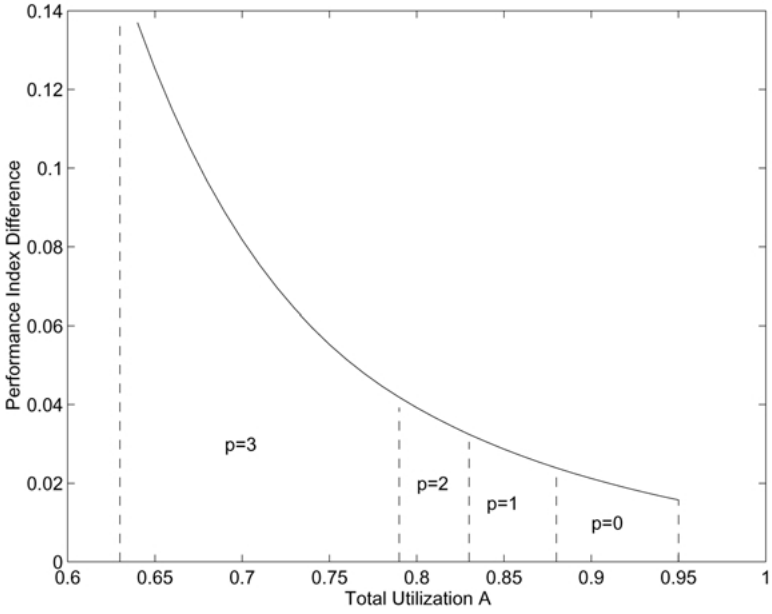

Figure 6. ΔJ versus total utilization A.

Table 4. Optimal frequencies and corresponding ΔJ at different utilization levels.

A	p	$[f_1,f_2,f_3,f_4]$	ΔJ
0.95	0	[15.77, 11.02, 21.53, 46.68]	0.0157
0.8852	1	[15, 10.465, 20.24, 42.81]	0.0232
0.8371	2	[15, 10, 19.16, 39.55]	0.0310
0.7908	3	[15, 10, 18, 36.08]	0.0416
0.63	4	[15, 10, 18, 20]	0.1499

performance. Then an economic decision needs to be made if a faster processor is indeed needed. The same conclusion can be made in Example 3.2, namely, as the performance index difference correspoinding to the optimal frequencies, $\Delta J = 0.0157$, is 89% better than $\Delta J = 0.1499$ at the minimal frequencies, a faster processor will only gain another 11% extra improvement on performance, and such an improvement will require significant increase of processor speed. Furthermore, since the utilization when tasks are assigned minimal frequencies, i.e., $A = 0.63$, is far below the available utilization $A = 0.95$ in this example, we would question if it is economical to have the four bubble control systems use up all the available computing resource. We must determine if more tasks should be scheduled on the same processor without significantly affecting the performance of the bubble systems. Figure 6 shows that there is indeed a possibility of increasing the load on the processor. In fact, for low levels of available utilization, great gains in the overall performance can occur from relatively small increases in the available utilization. However, a point of diminishing marginal returns will be reached. At this point, any increase in available utilization begins to contribute relatively little to the overall system performance. Therefore, for a high level of available utilization, as in Example 3.2, loading more tasks on the processor will not significantly reduce the overall system performance.

4. Conclusion

In this paper, we have considered the periodic task schedulability issue from a new point of view, optimization of the performance of a real-time computer-controlled system. For a set of tasks from a class of control systems, whose performance indices are monotonically decreasing (increasing) and convex (concave) functions of the sampling frequencies, an algorithm is proposed to determine the task frequencies such that a weighted sum of the system performance indices is optimized and all the tasks are schedulable with a dynamic priority assignment scheduling algorithm.

The main contributions of this paper are three-fold. First, the developed algorithm enhances the processor's task schedulability by allowing the task frequencies to change in relation to their system performance. For a set of real-time control tasks with given sampling frequencies, they may not be schedulable with the chosen DPA scheduling algorithm, but will become schedulable if their frequencies can be adjusted and their minimum allowable frequencies render the schedulability. In this case, the algorithm guarantees that the tasks will be scheduled with the system performance being optimized.

Second, even for a set of tasks which are originally schedulable, the proposed algorithm can still be used to improve the overall system performance. This feature distinguishes our algorithm from the others where system performance was not considered to be a factor in scheduling the tasks. Finally, the proposed integrated system design approach provides a guideline for choosing the processors and loads during the design phase. It offers an analytical means for improving system performance in terms of hardware costs.

Future work along this line will include generalizing the algorithm developed here to address the problem of task re-allocation over a network of processors and optimizing the overall system performance. Preliminary results on this extension is documented in Shin and Meissner [10].

References

1. Gerber, R., Hong, S., and Saksena, M. 1994. Guaranteeing end-to-end timing constraints by calibrating Intermediate Processes. In *Proceedings of the IEEE Real-Time Systems Symposium.*
2. Liu, C. L., and Layland, J. W. 1973. Scheduling algorithms for multiprogramming in a hard real-time environment. *Journal of Association for Computing Machinery* 20(1): 46–61.
3. Locke, C. D. 1986. Best-effort decision making for real-time scheduling. Ph.D. Dissertation, Computer Science Department, Carnegie Mellon University.
4. Mangasarian, O. L. 1969. *Nonlinear Programming*. McGraw-Hill Book Company.
5. Molini, J. J., Maimon, S. K., and Watson, P. H. 1990. Real-time system scenarios. In *Proceedings of the IEEE Real-Time Systems Symposium*.
6. Seto, D., Lehoczky, J. P., and Sha, L. 1998. Task period selection and schedulability in real-time systems. In *Proceedings of the 20th IEEE Real-Time Systems Symposium.*
7. Sha, L., Rajkumar, R., and Sathaye, S. S. 1994. Generalized rate-monotonic scheduling theory: A framework for developing real-time systems. *Proceedings of the IEEE* 82(1): 68–82.
8. Shin, K. G., Krishna, C. M., and Lee, Y.-H. 1985. A unified method for evaluating real-time computer controllers and its application. *IEEE Transactions on Automatic Control* 30(4): 357–366.
9. Shin, K. G., and Kim, H. 1992. Derivation and application of hard deadlines for real-time control systems. *IEEE Transactions on Systems, Manufacturing, and Cybernetics* 22(6): 1103–1413.
10. Shin, K. G., and Meissner, C. 1999. Adaptation and graceful degradation of control system performance by task re-allocation and period adjustment. *Proceedings of the 1999 European Real-Time Systems Conference*. York, UK.