Holistic Design Parameter Optimization of Multiple Periodic Resources in Hierarchical Scheduling

Man-Ki Yoon[*], Jung-Eun Kim[*], Richard Bradford[†], and Lui Sha[*]
[*]University of Illinois at Urbana-Champaign, Urbana, IL 61801
Email: {mkyoon, jekim314, lrs}@illinois.edu
[†]Rockwell Collins, Cedar Rapids, IA 52498
Email: rmbradfo@rockwellcollins.com

***Abstract*—Hierarchical scheduling of periodic resources has been increasingly applied to a wide variety of real-time systems due to its ability to accommodate various applications on a single system through strong temporal isolation. This leads to the question of how one can optimize over the resource parameters while satisfying the timing requirements of real-time applications. A great deal of research has been devoted to deriving the analytic model for the bounds on the design parameter of a single resource as well as its optimization. The optimization for multiple periodic resources, however, requires a holistic approach due to the conflicting requirements of the limited computational capacity of a system among resources. Thus, this paper addresses a holistic optimization of multiple periodic resources with regard to minimum system utilization. We extend the existing analysis of a single resource in order for the variable interferences among resources to be captured in the resource bound, and then solve the problem with Geometric Programming (GP). The experimental results show that the proposed method can find a solution very close to the one optimized via an exhaustive search and that it can explore more solutions than a known heuristic method.**

I. Introduction

As the processing power of processors has grown, there has been an increasing trend toward integrating many real-time applications on a single system and thus efficiently utilizing the system by allowing the applications to share common hardware devices. In such systems, temporally partitioned hierarchical scheduling [1]–[5] has been widely adopted because of its strong isolation among sets of real-time applications, which either are independently developed or have different functionalities or criticalities. For example, in IMA (Integrated Modular Avionics) architecture [6], applications are often grouped into different partitions according to their design-assurance levels in order to protect high-criticality applications from the faulty behavior of other applications and guarantee their timing requirements.

In such a partitioned hierarchical scheduling, one important question is how much of the computational resource needs to be allocated to each resource in order for the system to be optimized for a certain metric. For instance, it is desirable in system design to minimize the system utilization while guaranteeing the timing requirements of both resources and their applications. This is true since a lower-utilized system can be more utilized by accommodating additional workload or, alternatively, the same workload can be implemented by a lower-speed system.

This work is supported in part by Rockwell Collins under RPS#6 45038, by Lockheed Martin under 2009-00524, by NSF under A17321, and by ONR under N00014-12-1-0046. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of sponsors.

978-3-9815370-0-0/DATE13/©2013 EDAA

For a single resource case, the optimized *resource design parameters*, that is, *period* and *execution length*, can be obtained by a method based either on an exact schedulability test [4], [5] or on resource supply and demand functions [1]–[3]. However, it is often intractable to find the optimal set of resource parameters mainly because the local optimality of each resource does not necessarily lead to the globally optimal solution [4], [5]. Accordingly, each design parameter cannot be chosen independently; the optimal selection requires a brute-force search, which is only practical when some parameters are fixed and/or the number of resources is small.

Thus, in this paper, we are interested in finding a sub-optimal set of resource design parameters that minimizes the schedulable system utilization; both resources and their tasks are schedulable. Specifically, we consider the periodic resource model introduced in [1]–[5]; each resource $\mathcal{R}$ is periodically released at every T and supplies an execution amount of L to its tasks. For global and local scheduling, we consider fixed-priority scheduling with the assumption that priorities are pre-assigned. The results on the resource parameter bound in previous work were derived by calculating the lower-bound on a resource supply that can satisfy the worst-case demand of the workload. When other resource parameters are unknown, however, a pessimistic assumption on the minimum supply needs to be made; each resource suffers the maximum possible delay. We tackle this problem by parameterizing the worst-case resource supply with the unknown parameters of other resources that can be holistically optimized via *Geometric Programming (GP)* [7], [8]. GP is a non-linear optimization method that can solve a specially formed non-convex problem by transforming it into a convex one through a logarithmic transformation, thus finding the optimal solution efficiently. We present a GP formulation as the solution to the design parameter optimization of multiple periodic resources. We show that our method can find a solution that is close to the one that can be found by an exhaustive search, and it can explore more solutions than a known heuristic method [5].

II. Problem Description

A. System Model

We consider a uniprocessor consisting of a set of independent periodic resources $\mathfrak{R} = \{\mathcal{R}_i | i = 1, \ldots, N^{\mathfrak{R}}\}$. Each $\mathcal{R}_i$ is characterized by an unknown tuple of (T_i, L_i), $T_i, L_i \in \mathbf{R}^+$, where T_i and L_i are the period and the execution length of the resource, respectively.[1] In $\mathcal{R}_i$, a set $\mathbf{\Gamma_i} = \{\tau_j | j = 1, \ldots, N^{\mathbf{\Gamma_i}}\}$ of tasks run in a fixed-priority preemptive schedule such as Rate Monotonic [10]. Each τ_j is represented by $\tau_j := (e_j, p_j, d_j)$, where e_j is the worst-case execution time, p_j is the minimum inter arrival time between successive releases, and d_j is the relative deadline.[2] In this paper, we assume that $d_j = p_j$. We then further assume that there is no synchronization or precedence constraints among tasks, and task releases are not bound to resources [5].

The resources are also scheduled in a fixed-priority manner and we assume their deadline, D_i, is equal to the period. In addition, we consider that resource priorities are given, assuming, for example, the priorities are assigned according to criticalities. We note that the optimization method in this paper cannot be applied to cases when resource priorities are not given. Additionally, a resource is idled if there is no task ready to execute. We also assume that there is no resource release jitter.

B. *Problem Description*

Given a set of resources $\{\mathcal{R}_i\}$ and the corresponding task sets $\{\mathbf{\Gamma_i}\}$, our problem is to find the set of the resource parameters, $\{(T_i, L_i)\}$ for $i = 1, \ldots, N^{\mathfrak{R}}$, which minimizes the overall system utilization, U_s, while guaranteeing the schedulabilities of the resources and the tasks. Here, U_s can be represented by

$$U_s = \sum_{i=1}^{N^{\mathfrak{R}}} \frac{c_1 \cdot \delta + c_2 \cdot L_i}{T_i}, \tag{1}$$

where δ is the resource context-switch overhead, and c_1, c_2 are weights given by the system designer [3]. In this paper, we set both c_1 and c_2 to 1 and assume that each resource will consume a context-switch overhead of δ at each release.

III. Parameter Bounds of Multiple Resources

In this section, we present the analysis of multiple resource bounds by extending those of the single resource bound in the previous literature [1], [2], which is summarized in [9]. In a partitioned resource whose period T_i and length L_i are unknown, we can derive the lower-bound of L_i (or the upper-bound of T_i) with respect to T_i (or L_i) that makes τ_j in $\mathcal{R}_i$ schedulable by using the periodic resource model [1], [2]. Informally speaking, the key idea of previous work is that a task can be schedulable if the minimum resource supply ($\mathbf{sbf}_{\mathbf{\Gamma}}(t)$ [1] or $A\_s(t)$ [2]) can match the maximum workload demand generated by τ_j and its higher-priority tasks during a time interval t. In fixed-priority global scheduling, the minimum supply of a resource is delivered to its tasks when its $(k-1)^{th}$ execution has just been finished at time 0 with minimum interferences from higher-priority resources, and then the subsequent executions are maximally delayed by higher-priority resources. When the parameters of higher-priority resources are unknown, it is a safe assumption that the subsequent executions from the k^{th} release are delayed by $T_i - L_i$. This is pessimistic, since, in reality, high priority resources would suffer no or only a few preemptions. Thus, an exact method is required [4], [5], which, however, is not useful for an optimization of multiple resource parameters due to its high time complexity. This necessitates holistic optimization of multiple resource parameters, and thus in the following we present the analysis of multiple resource bounds via a parameterization of unknown resource parameters, which will be formulated and solved by Geometric Programming in Sec. IV. The optimization of a single resource has been extensively studied. Interested readers can refer to [1]–[5], [11]–[15].

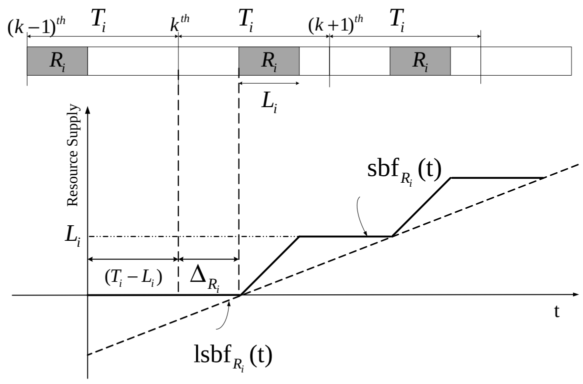

Fig. 1: The worst-case release pattern of $\mathcal{R}_i$ considering the periods and execution lengths of higher-priority resources.

A. *Lower-bound Supply Function Considering Unknown Parameters of Higher-Priority Resources*

We consider a worst-case release pattern of $\mathcal{R}_i$ occurring when it suffers zero delay in the $(k-1)^{th}$ release and the maximum delay from higher-priority resources thereafter, i.e., $\Delta_{\mathcal{R}_i}$, as depicted in Fig. 1. The worst-case busy period of $\mathcal{R}_i$, denoted as $w_{\mathcal{R}_i}$, is the maximum time duration that $\mathcal{R}_i$ can take to execute L_i when it is released simultaneously with its higher-priority resources, $hp(\mathcal{R}_i)$, at the k^{th} release, which can be obtained by the traditional exact analysis:

$$w_{\mathcal{R}_i}^{k+1} = L_i + \sum_{\mathcal{R}_h \in hp(\mathcal{R}_i)} \left\lceil \frac{w_{\mathcal{R}_i}^{k}}{T_h} \right\rceil \cdot L_h, \tag{2}$$

where $w_{\mathcal{R}_i}^0 = L_i$ and $w_{\mathcal{R}_i} = w_{\mathcal{R}_i}^k$ when it converges for some k. Thus, the worst-case delay at the k^{th} release and thereafter (called *initial latency* in [2]) can be represented as

$$\Delta_{\mathcal{R}_i} = \sum_{\mathcal{R}_h \in hp(\mathcal{R}_i)} \left\lceil \frac{w_{\mathcal{R}_i}}{T_h} \right\rceil \cdot L_h. \tag{3}$$

However, the iterative method of (2) can only be applicable to brute-force optimization. Thus, we approximate $\Delta_{\mathcal{R}_i}$. During a time interval of T_i, the maximum workload generated by $\mathcal{R}_i$ and $hp(\mathcal{R}_i)$ can be represented by:

$$w_{\mathcal{R}_i} = L_i + \sum_{\mathcal{R}_h \in hp(\mathcal{R}_i)} \left\lceil \frac{T_i}{T_h} \right\rceil \cdot L_h.$$

We can avoid iterative calculation by assuming the number of invocations of higher-priority resources during T_i, not during the exact busy period. Note that it is a safe bound as long as $\mathcal{R}_i$ meets its deadline, i.e., $D_i = T_i$. Now, we linearize $w_{\mathcal{R}_i}$ as follows:

$$w_{\mathcal{R}_i} = L_i + \sum_{\mathcal{R}_h \in hp(\mathcal{R}_i)} \left(\frac{T_i}{T_h} + 1 \right) \cdot L_h,$$

because $\lceil x \rceil \leq x + 1$. Then, the worst-case linear lower-bound supply function[3] of $\mathcal{R}_i$ during a time interval t parameterized

[1]When $T_i, L_i \in \mathbf{N}$, a branch-and-bound or a variable rounding heuristic is required. We note that the choice of such method is orthogonal to the optimization presented in this paper. In [9], we describe the effect of such constraints on the considered optimization problem.

[2]More precisely, each task should be represented as $\tau_{i,j}$ if it belongs to $\mathcal{R}_i$. For the simplicity of notations, however, we use the abbreviation τ_j.

[3]It is identical to $A'\_s(t)$ with $\Delta = \Delta_{\mathcal{R}_i}$ and $\alpha = \frac{L_i}{T_i}$ in [2].

with $\Delta_{\mathcal{R}_i}$ is derived as follows:

$$\mathbf{lsbf}_{\mathcal{R}_\mathbf{i}}(t) = \frac{L_i}{T_i} \cdot (t - (T_i - L_i) - \Delta_{\mathcal{R}_i}), \tag{4}$$

where

$$\Delta_{\mathcal{R}_i} = \sum_{\mathcal{R}_h \in hp(\mathcal{R}_i)} \left(\frac{T_i}{T_h} + 1\right) \cdot L_h. \tag{5}$$

B. *Sufficient Resource Bound for Task Schedulability*

Let us now consider τ_j in $\mathcal{R}_i$ whose period T_i is fixed. Then, let us define $L_i^{min}(\tau_j, T_i)$ as the minimum required length of $\mathcal{R}_i$ that guarantees to schedule τ_j. In order to derive $L_i^{min}(\tau_j, T_i)$, we can consider the situation in which τ_j barely meets its deadline at time $t = d_j$ with the worst-case interference from higher-priority tasks $hp(\tau_j)$. Since we make no assumption on task offsets, the worst-case response time of τ_j occurs when it and $hp(\tau_j)$ are released simultaneously at the end of $\mathcal{R}_i$'s $(k-1)^{th}$ execution and then suffers the worst-case preemptions from $hp(\tau_j)$ in k^{th} release and thereafter, which we define as *the critical instant*.[4] Now, let us denote I_j as the worst-case workload generated by τ_j and $hp(\tau_j)$ from the critical instant to the deadline of τ_j as follows:

$$I_j = e_j + \sum_{\tau_h \in hp(\tau_j)} \left\lceil \frac{d_j}{p_h} \right\rceil \cdot e_h.$$

τ_j is guaranteed to be schedulable if the minimum supply delivered by the resource is greater than or equal to the worst-case workload generated during the time interval d_j. Thus,

$$\mathbf{lsbf}_{\mathcal{R}_\mathbf{i}}(d_j) = \frac{L_i}{T_i} \cdot (d_j - (T_i - L_i) - \Delta_{\mathcal{R}_i}) \geq I_j. \tag{6}$$

Accordingly, the minimum required resource length, $L_i^{min}(\tau_j, T_i)$, for τ_j with a given T_i can be obtained by solving the quadratic inequality in (6), which result in

$$L_i^{min}(\tau_j, T_i) = \frac{-(d_j - T_i - \Delta_{\mathcal{R}_i}) + \sqrt{(d_j - T_i - \Delta_{\mathcal{R}_i})^2 + 4I_jT_i}}{2}. \tag{7}$$

Note that (7) is equivalent to Eq. (23) in [1] with $\Delta_{\mathcal{R}_i} = T_i - L_i$ and to Eq. (12) in [2] with $\beta = 1$.

In order to find the minimum required length of $\mathcal{R}_i$ for a given T_i, we take the maximum of the bounds $L_i^{min}(\tau_j, T_i)$ over all tasks in $\mathbf{\Gamma_i}$, which therefore can be defined as follows:

$$L_i^{min}(T_i) = \max_{\tau_j \in \mathbf{\Gamma_i}} \Big(L_i^{min}(\tau_j, T_i)\Big). \tag{8}$$

Thus, if we take L_i from $[L_i^{min}(T_i), T_i]$, all $\tau_j \in \mathbf{\Gamma_i}$ are guaranteed to meet their deadlines. It is now important to note that (6) is only sufficient and not necessary condition; τ_j can be schedulable if and only if there exists a time instant $t \leq d_j$ such that $\mathbf{lsbf}_{\mathcal{R}_\mathbf{i}}(t) \geq I_j$ [1]. In this paper, however, we use the sufficient condition in (6) because the presence of time in the necessary condition makes the proposed optimization method not applicable to the problem under consideration. Although the bound is not exact and may incur approximation error, it enables us to optimize multiple resources holistically with high efficiency, as will be described in Sec. IV.

[4]In this paper, we do not consider task jitters. However, without loss of generality, the presented analyses can be similarly applied to cases with jitters. For example, the worst-case situation of τ_j is when $hp(\tau_j)$ have experienced their maximum jitters and are released at the same time with τ_j.

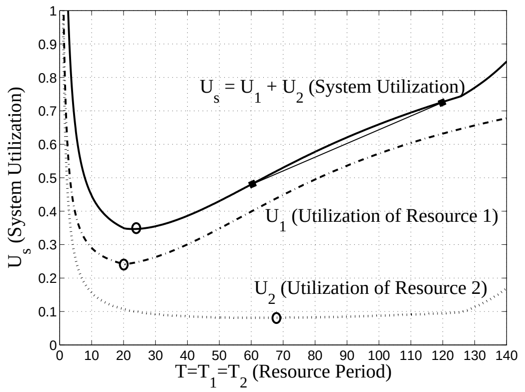

Fig. 2: System utilization of two resources and its non-convexity.

C. *Non-convexity of Multiple Resource Optimization*

We present an example of two resources in order to show the non-convexity of multiple resource optimization. Let us consider Fig. 2, which shows the utilization functions, U_1 and U_2, of two randomly generated resources, $\{\mathcal{R}_1, \mathcal{R}_2\}$, and the system utilization function, $U_s = U_1 + U_2$, over the period in $[1, 140]$. In this example, the resources have the same period for simplicity of representation, and δ is set to 1. $\mathcal{R}_1$ has a higher priority than $\mathcal{R}_2$, thus $\Delta_{\mathcal{R}_1} = 0$ and $\Delta_{\mathcal{R}_2} = 2 \cdot L_1$. From the graphs, we can first see that U_s is not convex, which is shown by the straight line drawn between $T = 60$ and 120. Furthermore, while the resources achieve minimum utilization at $T_1 = 20.3$ and $T_2 = 63.9$, respectively, these do not lead to the global optimality, which occurs at $T = T_1 = T_2 = 23.1$; this issue is addressed also in [4]. In this example, the resources have the same period. The optimization of multiple resource parameters will be harder to solve once we consider a higher number of resources and arbitrary resource periods.

IV. Holistic Optimization of Multiple Resource Parameters via Geometric Programming

In this section, we formulate the parameter optimization problem of multiple periodic resources with Geometric Programming (GP) [7], [8]. GP can solve a non-linear, non-convex optimization problem if it can be formulated in the following form:

$$\begin{array}{ll} \text{Minimize} & f_0(\mathbf{x}) \\ \text{Subject to} & f_i(\mathbf{x}) \leq 1,\ i = 1, \ldots, n_p, \\ & g_j(\mathbf{x}) = 1,\ j = 1, \ldots, n_m, \end{array}$$

where f and g are *posynomial* and *monomial* functions, respectively, and $\mathbf{x}$ are the optimization variables. A function $g_j(\mathbf{x})$ is monomial if it can be represented as:

$$g_j(\mathbf{x}) = c_j \prod_{k=1}^{n_j} x_k^{a_k},$$

where $c_j \in \mathbf{R}^+$ and $a_k \in \mathbf{R}$. A posynomial function is a sum of monomials, and thus can be expressed as:

$$f_i(\mathbf{x}) = \sum_{k=1}^{n_i} c_k x_1^{a_{1k}} x_2^{a_{2k}} \cdots x_n^{a_{nk}},$$

where $c_k \in \mathbf{R}^+$ and $a_{jk} \in \mathbf{R}$. Also, f/g is a posynomial and f^{a_x} is also a posynomial if $a_x \in \mathbf{R}^+$. In summary, only the objective function and the inequality constraints can be posynomial.

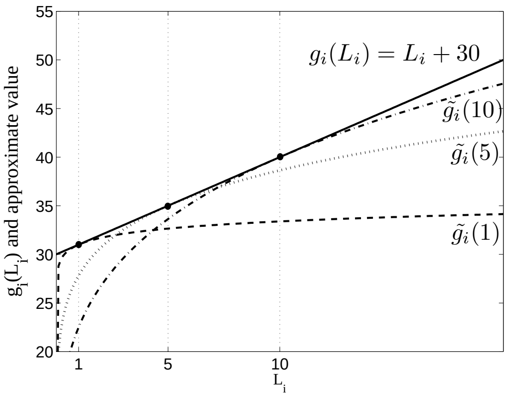

Fig. 3: $g_i(L_i) = L_i + 30$ and $\tilde{g}_i(L_i)$ at $x_0 = 1, 5$, and 10. $\tilde{g}_i(x_0)$ is tangent to $g_i(L_i)$ at $L_i = x_0$.

We now formulate the optimization problem of the multiple resource parameters in GP form. As stated in Sec. II-B, we are given a set of periodic resources $\{\mathcal{R}_i\}$ with unknown parameters, T_i and L_i; their task sets $\{(e_j, p_j, d_j)|\forall\tau_j \in \mathbf{\Gamma_i}\}$, and the resource context-switch overhead δ are known. Optimization variables are $\mathbf{T} = (T_1, \ldots, T_{N^{\mathfrak{R}}})$ and $\mathbf{L} = (L_1, \ldots, L_{N^{\mathfrak{R}}})$.

Objective Function

The objective function (1) in Sec. II-B is already in a posynomial form, thus it can be represented as follows:

$$f_o(\mathbf{T}, \mathbf{L}) = \sum_{i=1}^{N^{\mathfrak{R}}} (c_1\delta + c_2 L_i) \cdot T_i^{-1}, \tag{9}$$

where $c_1, c_2, \delta \geq 0$.

Resource Bound Constraint

The resource bound for each $\mathcal{R}_i$ is constrained by (6) for each $\tau_j \in \mathbf{\Gamma_i}$, which can be reexpressed as follows:

$$\frac{T_i \cdot (L_i + I_j) + \Delta_{\mathcal{R}_i} \cdot L_i}{L_i \cdot (L_i + d_j)} \leq 1, \tag{10}$$

where d_j and I_j are constants for a given input. However, the inequality above does not conform to a posynomial form because of the posynomial term in the denominator, i.e., $L_i \cdot (L_i + d_j) = L_i^2 + L_i \cdot d_j$ (Recall that a denominator must be monomial). Observe, however, that $L_i + d_j$ can be approximated with a monomial by the following geometric mean approximation [16]. Let us first denote it as $g_i(L_i) = u_1(L_i) + u_2(L_i)$ where $u_1(L_i) = L_i$ and $u_2(L_i) = d_j$. Then, we now approximate $g_i(L_i)$ with

$$\tilde{g}_i(L_i) = \Big(u_1(L_i)/\gamma_1\Big)^{\gamma_1} \cdot \Big(u_2(L_i)/\gamma_2\Big)^{\gamma_2}, \tag{11}$$

where $\gamma_1 = \frac{u_1(x_0)}{g_i(x_0)}$ and $\gamma_2 = \frac{u_2(x_0)}{g_i(x_0)}$ where $x_0 \in \mathbf{R}^+$ is a constant that satisfies $\tilde{g}_i(x_0) = g_i(x_0)$. The approximated monomial $\tilde{g}_i(L_i)$ then can be rewritten as:

$$\tilde{g}_i(L_i) = \Big(L_i/\gamma_1\Big)^{\gamma_1} \cdot \Big(d_j/\gamma_2\Big)^{\gamma_2},$$

with $\gamma_1 = \frac{x_0}{x_0+d_j}$ and $\gamma_2 = \frac{d_j}{x_0+d_j}$. Finally, (10) can be formulated as the following posynomial constraint:

$$(T_i \cdot (L_i + I_j) + \Delta_{\mathcal{R}_i} \cdot L_i) \cdot (L_i \cdot \tilde{g}_i(L_i))^{-1} \leq 1, \tag{12}$$

where $\Delta_{\mathcal{R}_i}$ is $\sum_{\mathcal{R}_h \in hp(\mathcal{R}_i)} \big((T_i + T_h) \cdot T_h^{-1} \cdot L_h\big)$. Note that the approximation quality of $\tilde{g}_i(L_i)$ depends on the choice of x_0, as shown in Fig. 3. Thus, in the optimization procedure, we iteratively approximate $\tilde{g}_i(L_i)$ by updating γ_1 and γ_2 according to the intermediate solution of L_i; until the objective value converges, we use L_i at k^{th} step as x_0 at $(k+1)^{th}$ step. The objective value converges normally in 1 or 2 iterations.

Resource Schedulability Constraint

Each resource must be schedulable, that is, $L_i + \Delta_{\mathcal{R}_i} \leq T_i$, which corresponds to the following posynomial constraint:

$$\Big(L_i + \sum_{\mathcal{R}_h \in hp(\mathcal{R}_i)} \big((T_i + T_h) \cdot T_h^{-1} \cdot L_h\big)\Big) \cdot T_i^{-1} \leq 1. \tag{13}$$

V. Evaluation

A. Evaluation Method

Parameter	Value
Number of resources, $N^{\mathfrak{R}}$	$\{2, 3, 4, 5\}$
Number of tasks per resource, $N^{\mathbf{\Gamma_i}}$	$[2, 8]$
Task execution time, e_j	$[1, 30]$
Task period, p_j	$[50, 2000]$
Context-switch overhead, δ	1

The table above summarizes the experimental parameters used for the evaluations. We consider the cases with $2, 3, 4,$ and 5 resources, and for each case, we generated 100 random input sets with the parameters. The number of tasks per resource, the task execution time and period are uniformly randomly chosen in the given range. For the simplicity of evaluations, the context-switch overhead was fixed to 1. With these parameters, we compare the following methods:

- Exhaustive Search: From the highest priority resource to the lowest one, we recursively assign each resource period from 1 to T_{max} with a step size of s. For each period T_i, L_i^{min} is determined by (8) and (7) with $\Delta_{\mathcal{R}_i}$ calculated by (3). Recall that the system utilization obtained with this exhaustive search is still not the exact globally optimal solution as explained in Sec. III.
- GP with the upper-bound on T_{max}: The GP optimization method with additional constraints on the upper-bound on resource periods, that is, $T_i \cdot T_{max}^{-1} \leq 1$.
- GP without the upper-bound on T_{max}: Identical to the above except that there is no upper-bound on T_{max}. Note that in our GP-based optimization, the upper-bound on resource period is unnecessary.

For the evaluation purpose only, the priority of each resource is assigned according to the utilization sum of tasks in each resource; the higher the utilization is, the higher its priority. Readers interested in priority optimization can refer to [4], [5]. Task priorities in each resource are assigned by RM priority assignment [10]. The GPs were solved using GGPLAB [17].

B. Evaluation Metric

We compare the methods above in terms of the minimum system utilization, i.e., Eq (1). We denote the solution of each method as U_s^{Exh}, $U_s^{GP_1}$, and $U_s^{GP_2}$, respectively. For each input, we calculate the difference of U_s^{Exh} from $U_s^{GP_1}$ and $U_s^{GP_2}$, that is, $U_s^{GP_1} - U_s^{Exh}$ and $U_s^{GP_2} - U_s^{Exh}$, respectively, and then take the average of 100 random input sets for each setting. It should be noted that we do not compare the solving time of each method because while GP can solve a problem within a few seconds, the exhaustive search normally takes 10–60 minutes or more depending on the problem size and the choices of T_{max} and s.

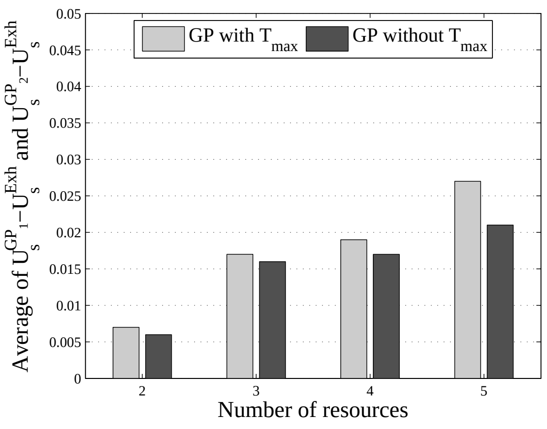

Fig. 4: The average differences of $U_s^{GP_1}$ and $U_s^{GP_2}$ to U_s^{Exh} with different numbers of resources.

C. Evaluation Results

Fig. 4 compares the minimum system utilization found by the exhaustive search and our GP methods increasing the number of resources from 2 to 5.[5] T_{max} and s were set to 100 and 0.5, respectively. As we can see from the result, the average difference of $U_s^{GP_1}$ and U_s^{Exh} increases with the number of resources. This is mainly because of the approximation error in $\Delta_{\mathcal{R}_i}$ (see Sec. III). Recall that while the exhaustive search calculates the minimum supply of each resource by using the iterative equation (Eq. (3)), our GP method approximately calculates the interference from higher-priority resources during the interval of its period (Eq. (5)). With two resources, the error of the approximation is small; however as the number of resources increases, the error accumulates from higher-priority resources to lower-priority ones. Nevertheless, we can see that the differences between the two methods are quite small, indicating that our method can find a solution that is close to the one that can be found by the exhaustive search; 0.007–0.027 average difference compared to the exhaustive search. Another interesting observation is that GP_2 can find better solutions than GP_1. This result might be expected because of the limited search space of GP_1: when the workload of a resource is significantly lower than the other resources, its optimal period may appear beyond T_{max}. In fact, for some input sets, $U_s^{GP_2}$ were lower than U_s^{Exh}. One can find better solutions with the exhaustive search by setting T_{max} higher, however, this can be limited by the input size.

With the same inputs, we evaluated the performance of our GP method (GP_2) with various base utilizations, i.e., the sum of all task utilizations. From Fig. 5, we can see that the differences increase with the base utilizations. A similar argument as above can be used to explain this correlation. That is, a higher base utilization implies that there exist resources with higher utilizations, and thus those tend to have shorter periods and longer execution lengths. We attribute this, again, to the approximation error of $\Delta_{\mathcal{R}_i}$ in (5).

Lastly, we compared our GP method with the heuristic proposed in [5]. The method finds the optimal parameters for each resource in turn from the highest to the lowest resources; for each resource, it iterates over a range of periods and for each

[5]The exhaustive search takes a longer time to solve cases with six or more resources, and thus we evaluated cases with 2, 3, 4, and 5 resources.

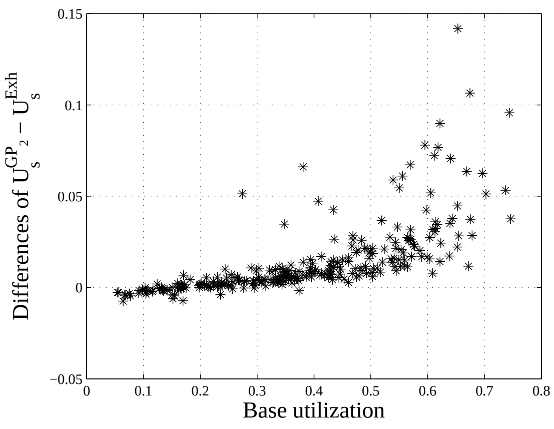

Fig. 5: $U_s^{GP_2}$ - U_s^{Exh} with various base utilizations.

period, it finds the optimal resource length by a binary search. The same process is applied to the next priority resource. For the comparison, we used the same inputs as above. In the heuristic, each T_i is assigned from 1 to 1000 with s of 0.1, and each L_i was found at the granularity of 0.1. Fig. 6 shows i) the numbers of solutions found by each method and ii) the average difference of the minimum system utilization between the methods. As can be seen from the bars, our method finds more solutions than the heuristic, and in the experiment, all input sets for which a solution was found by the heuristic were also solved by our method. We can also see that as the number of resources increase, the gap also increases; with 5 resources, our method found 49 solutions among 100 input sets, but only 7 solutions were found by the heuristic.[6] This follows from the greedy nature of the heuristic; the parameters for a high-priority resource were locally optimized without considering the feasibilities of lower-priority resources. In contrast, although our method is not a global optimal method either, it can explore more solutions due to its ability to take into account the variable interferences among resources simultaneously in the GP optimization process. However, the qualities of the solutions found by our method are worse than those found by the heuristic, as the line plot in Fig. 6 shows. Each marker on the line is the average of the difference of $U_s^{GP_2}$ and U_s^{Heu}, for the input sets the heuristic found; with 2, 3 and 4 resources, the average differences are between 0.055 and 0.065, and with 5 resources, the difference is 0.108. The spike at 5 resources could be explained by the low number of solutions found. Although our method achieved lower system utilization for some input sets, the heuristic could find better solutions in most cases. Such differences mainly arise from the optimality of the analysis used by the heuristic. That is, when the parameters of higher-priority resources and the period of the resource under analysis are fixed, the (local-)optimal resource length is found by the binary search which is based on the exact analysis [5]. On the other hand, as explained Sec. III, our analysis considers the worst-case scenarios that are sufficient but not necessary, and it is also based on the approximation of $\Delta_{\mathcal{R}_i}$, both of which

[6]The exact number of feasible solutions is unknown as this requires a true optimal method. Among 100 input sets for each case, some input sets might be infeasible in the first place. Also, the main reason that each method finds fewer number of solutions as the number of resources increase is because the base system utilization also increases. For example, the average base utilizations of 100 input sets with 2 and 5 resources are 0.281 and 0.663, respectively.

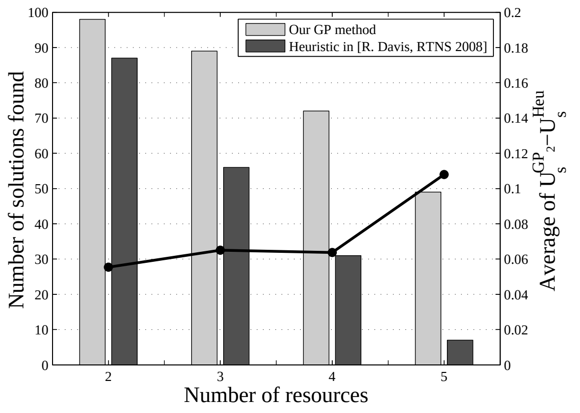

Fig. 6: The number of solutions found by the proposed GP method and the heuristic in [5] (Bars), and the average differences of $U_s^{GP_2}$ and U_s^{Heu} (Line).

lead to schedulability loss. From this evaluation, we can conclude that there is a trade-off between the solution feasibility (our GP method) and the solution quality (the heuristic of [5]).

VI. Related Work

Shin *et al.* [1] proposed the periodic resource model in a hierarchical scheduling that facilitates the schedulability analysis of the workload of tasks (child) under a periodic resource supply (parent). The authors presented the exact schedulability analysis of a workload set in a periodic resource under RM and EDF scheduling and derived the utilization bounds. In [2], Almeida *et al.* analyzed a similar periodic server model by introducing the server availability function. They also developed a heuristic algorithm for a server parameter optimization for the minimum system utilization, in which the search space is reduced to a set of *deadline points*. Lipari *et al.* [3] also considered the same problem with a different approach of schedulability analysis using a notion of characteristic function. These three works all used linear resource supply models. In contrast, Davis *et al.* [4], [5] presented the exact worst-case response time analysis of tasks under deferrable server, periodic server, and sporadic server. Through an empirical investigation, the authors claimed that the optimal parameter selection for multiple resources is a holistic problem and provided a greedy algorithm, which we compare with our GP-based method in Sec. V. Additionally, in [18], Saewong *et al.* developed a response time analysis for real-time guarantees of tasks under sporadic server and deferrable server. In [11], Easwaran introduced a generalized periodic resource model called *Explicit Deadline Periodic* (EDP) resource, and proposed an exact algorithm for determining the optimal resource parameter that minimizes the ratio of length to period of an EDP resource. The same problem for periodic resource model was addressed by Shin et al. [13], in which the authors presented a polynomial-time sufficient algorithm. Both problems were addressed by Dewan *et al.* [15] and Fisher [14] by proposing fully-polynomial-time approximation algorithms that improve both the optimality and time complexity. None of these papers, however, consider the problem of optimizing the parameters of multiple resources.

Geometric Programming [7], [8] has been widely applied to a broad range of non-linear, non-convex optimization problems such as digital circuit gate sizing [19], resource allocation in communication systems [16], information theory [20], etc. An extensive discussion of GP can be found in [8].

VII. Conclusion

In this paper we addressed the problem of design parameter optimization of multiple periodic resources in hierarchical scheduling. We extended the existing analysis on a single resource in order for our resource supply model to be able to capture the variable parameters of higher-priority resources. The presented analysis on the resource bounds and the GP-based optimization is not a globally optimal method due to the approximation error of worst-case resource interference and task schedulability. However, we believe that one can benefit from the presented optimization method in designing a hierarchical system with a large number of partitioned resources due to its ability to yield a high-quality solution with a high scalability. For future work, we plan to apply the presented method to a hierarchical system under non-preemptive global scheduling such IMA scheduling.

References

[1] I. Shin and I. Lee, "Periodic resource model for compositional real-time guarantees," in *Proceedings of the 24th IEEE Real-Time Systems Symposium*, 2003, pp. 2–13.

[2] L. Almeida and P. Pedreiras, "Scheduling within temporal partitions: response-time analysis and server design," in *Proceedings of the 4th ACM international conference on Embedded software*, 2004, pp. 95–103.

[3] G. Lipari and E. Bini, "Resource partitioning among real-time applications," in *Proceedings of the 15th Euromicro Conference on Real-Time Systems*, 2003, pp. 151–158.

[4] R. I. Davis and A. Burns, "Hierarchical fixed priority pre-emptive scheduling," in *Proceedings of the 24th IEEE Real-Time Systems Symposium*, 2005, pp. 389–398.

[5] R. Davis and A. Burns, "An investigation into server parameter selection for hierarchical fixed priority pre-emptive systems," in *Proccedings of Real-Time and Network Systems, RTNS*, 2008.

[6] *ARINC Specification 651: Design Guidance for Integrated Modular Avionics*, ser. ARINC report. Airlines Electronic Engineering Committee (AEEC) and Aeronautical Radio Inc, Nov. 1991.

[7] R. J. Duffin and E. Peterson and C. Zener, *Geometric Programming - Theory and Application*. John Wiley, New York, 1967.

[8] S. P. Boyd, S.-J. Kim, L. Vandenberghe, and A. Hassibi, "A tutorial on geometric programming," *Optimization and Engineering*, vol. 8, pp. 67–127, 2007.

[9] M.-K. Yoon, J.-E. Kim, R. Bradford, and L. Sha, "Geometric programming based optimization of multiple periodic resources in hierarchical scheduling," Dept. of Computer Science, University of Illinois at Urbana-Champaign, Technical report, May 2012, https://www.ideals.illinois.edu/handle/2142/35279.

[10] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard real-time environment," *Journal of the ACM*, vol. 20, no. 1, pp. 46–61, January 1973.

[11] A. Easwaran, "Compositional schedulability analysis supporting associativity, optimality, dependency and concurrency," *PhD thesis, Computer and Information Science, University of Pennsylvania*, 2007.

[12] A. Easwaran, M. Anand, I. Lee, and O. Sokolsky, "On the complexity of generating optimal interfaces for hierarchical systems," in *Workshop on Compositional Theory and Technology for Real-Time Embedded Systems*, 2008.

[13] I. Shin and I. Lee, "Compositional real-time scheduling framework with periodic model," *ACM Transactions on Embedded Computing Systems*, vol. 7, no. 3, pp. 30:1–30:39, May 2008.

[14] N. Fisher, "An FPTAS for interface selection in the periodic resource model," in *Proceedings of the 17th International Conference on Real-Time and Network Systems*, 2009, pp. 127–136.

[15] F. Dewan and N. Fisher, "Approximate bandwidth allocation for fixed-priority-scheduled periodic resources," in *Proceedings of the 16th IEEE Real-Time and Embedded Technology and Applications Symposium*, 2010, pp. 247–256.

[16] M. Chiang, "Geometric programming for communication systems," *Commun. Inf. Theory*, vol. 2, pp. 1–154, Jul. 2005.

[17] "GGPLAB: A Simple Matlab Toolbox for Geometric Programming," http://www.stanford.edu/~boyd/ggplab/.

[18] S. Saewong, R. R. Rajkumar, J. P. Lehoczky, and M. H. Klein, "Analysis of hierarhical fixed-priority scheduling," in *Proceedings of the 14th Euromicro Conference on Real-Time Systems*, 2002, pp. 152–160.

[19] S. P. Boyd, S.-J. Kim, D. D. Patil, and M. A. Horowitz, "Digital circuit optimization via geometric programming," *Operations Research*, vol. 53, pp. 899–932, 2005.

[20] M. Chiang and S. P. Boyd, "Geometric programming duals of channel capacity and rate distortion," *IEEE Trans. Inform. Theory*, vol. 50, pp. 245–258, 2004.