Robust Task Allocation for Dynamic Distributed Real-time Systems Subject to Multiple Environmental Parameters

Dazhang Gu, Frank Drews, Lonnie Welch
Center for Intelligent, Distributed, and Dependable Systems
School of Electrical Engineering and Computer Science
Ohio University, Athens, Ohio 45701 U.S.A.
dgu@bobcat.ent.ohiou.edu {drews,welch}@ohio.edu

Abstract

Some distributed real-time systems interact with external environments that change dynamically, and it is necessary to take the external variables into account when performing task allocation. We developed an approximation algorithm for task allocation, and it finds allocations that are maximally robust against dynamic changes in multiple external variables. Such an algorithm will help to reduce expensive reallocations triggered by changes in unpredictable environments. The algorithm has a polynomial running time, and its robustness optimality is given by an approximation ratio, which equals 2.41 asymptotically, when workloads are large and workload independent utilization of tasks is insignificant.

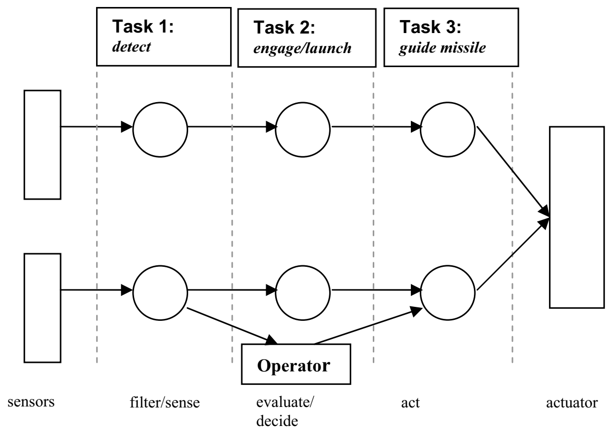


Figure 1. A motivating example [21]

1. Introduction

Execution times of dynamic distributed real-time systems (DDRTSs) can be affected by many unpredictable external variables (EVs) that originate in their environments [8]. The time complexities of algorithms generally depend on the sizes of their inputs, and the algorithms in real-time tasks are no exception. The unpredictable environmental parameters can result in varying execution times of system tasks that cannot be known in advance [17]. Thus it has been said that traditional periodic task scheduling and allocation based on worst case execution time is not applicable for many applications [19, 9, 20, 14]. In this paper, we develop a strategy for feasible task allocations that explicitly considers robustness of the allocations against dynamic changes in external variables.

The notion of tasks, which have workload-dependent execution times, originated from the study of the generic air defense system [21] depicted in Figure 1. The *detect* task identifies threats to a defended entity. The task runs periodically and performs the functions of filtering and evaluating radar tracks. When a threat is detected, the *detect* task triggers the *engage* task, which fires a missile at the threat. After a missile is in flight, the *guide missile* task keeps the missile on the correct course. The *guide missile* task executes periodically; uses sensor data to track the threat; recalculates the flight path for the missile; and issues guidance commands to the missile.

All three of these tasks have resource needs that are environment-dependent. The execution time of the *detect* task is primarily workload-dependent. Since the task evaluates each radar track to determine if it is a potential threat, its execution time is a function of the number of radar tracks in the environment. The workload of the *engage* task is also variable since it is activated by events which occur at rates that are determined by the external environment. Similarly, the work performed by the *guide missile* depends on the number of missiles in flight. Unlike traditional approaches to real-time computing, which characterize the resource needs of a task by a worst case execution time (WCET) [13], we characterize the resource needs of these tasks by execution profile functions. These functions compute the resource need as a function of workload [18] by curve-fitting [22].

We note that concurrency is often used in order to meet the real-time constraints of the tasks. Thus, during operation, there may be many replicas of the three tasks in the air defense system. When the number of radar tracks grows too large for a single replica of the *detect* task to process all tracks within the required time bound, one or more replicas are created and the radar tracks are partitioned among them. Additionally, pipeline concurrency is obtained by replicating the subtasks of the *detect* task. In a similar manner, the *guide missile* task is replicated as necessary to meet deadlines, and its subtasks are distributed. Replication is also used for the *engage* task when heavy workloads are anticipated. Thus, an important problem to solve for this system is how to allocate the tasks and subtasks in a manner that allows real-time constraints to be met and that minimizes the need for reallocations (which create overhead in the system). Also, we want to know the maximum numbers of missiles and radar tracks that can be sustained by a given hardware configuration.

Researchers have started to investigate this class of problems. Determining the maximal allowable increase in load for a feasible allocation was studied in [7]. Latency requirement was from end to end, and both computation and communication latency were considered. However, some real-time scheduling issues were not considered such as release time and subdeadline assignments [4, 15]. The variances of all execution times were characterized by linear functions of one system parameter α. The parameter was maximized using mixed integer programming after applying a simplifying heuristic. The performance was validated through simulations. The definition of a general robustness metric for multiple environmental parameters was introduced [3]. The metric was defined as a radius for a maximal allowable perturbation along any direction of the parameter space without violating system performance boundaries. Actual optimization algorithm was not the topic of the paper, although convex optimization was mentioned.

We previously studied a task allocation problem with a single workload variable for dynamic systems [10]. The goal was to find the maximal allowable workload (MAW) for a system of independent real-time tasks allocated to processors scheduled rate monotonically (RM). Tasks' execution times had workload dependent portions that were non-decreasing. An allocation algorithm was developed based on binary search and greedy first fit. During the allocation, the rate monotonic schedulable utilization for single processor was used to test feasibility. Heuristic algorithms were introduced to experimentally compare with performance of the approach which were simulated annealing, hill-climbing and random search. The set of algorithms was later expanded to include tabu search, genetic algorithm, dynamic programming, and optimal branch-and-bound [1]. The results indicated that the first-fit based algorithm performs well in finding the MAW compared to the heuristic or exhaustive algorithms, while it offers great cost-efficiency in time and space complexity. However, the work was restricted to a *single* environmental variable. The logical extension to multiple environmental variables will be addressed in this paper.

This research addresses the task allocation problem that seeks allocations with the maximal robustness measure against *multiple* external variables. The maximization will allow large changes in environment parameters to be absorbed while the same allocation remains feasible. Significance of the robustness lies in that no new allocation has to be recomputed, and no reallocation needs to be enacted. Either action can be very time consuming. High latency may result from a poorly allocated system, when changes in external variables frequently trigger reallocations, and the overhead may cause task deadline misses and serious consequences. The algorithm developed in this research will be used for task allocation decisions in the *Resource Management Service* of QARMA, the Quality-based Adaptive Resource Management Architecture [6]. Figure 2 shows the architecture, which consists of three major components: the *System Repository Service*, the *Resource Management Service*, and the *Enactor Service*. The *System Repository* stores both static and dynamic information that describe the software systems and resources in the computing environment. The *Resource Management Service* (dash boxed) is responsible for using information in a system repository to decide what actions should be performed to ensure that performance requirements are satisfied and overall optimized. The *Enactor Service* receives instructions from the resource management service about actions to perform and enacts them. Actions may include adjustment of quality of service settings and allocation of tasks. The algorithm, as part of the *Resource Management Service*, will provide robustness in such actions. We will start off the development with a system model and the problem definition.

System model: We assume a set of n periodic tasks $T = \{T_1, T_2, ..., T_n\}$ and a set of m identical processors $P = \{P_1, P_2, ..., P_m\}$. Each processor is assumed to use a rate monotonic scheduler. Each task T_i is characterized by a period $T_i.p$, and its deadline is assumed to be equal to its period. We assume l external variables $\vec{w} = (w_1, w_2, ..., w_l)$. For each task T_i the profile function $T_i.e(\vec{w})$ is given as a function of these external variables. The system utilization $U(\vec{w}) = \sum_{i=1}^{n} \frac{T_i.e(\vec{w})}{T_i.p}$, is also expressed as a function of the external variables, and $U(0)$ is the utilization of system that is independent of external variables.

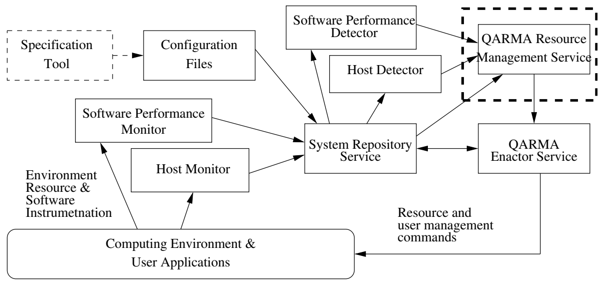

Specification Tool
Configuration Files
Software Performance Detector
Host Detector
QARMA Resource Management Service
Software Performance Monitor
System Repository Service
QARMA Enactor Service
Host Monitor
Environment Resource & Software Instrumetnation
Resource and user management commands
Computing Environment & User Applications

Figure 2. QARMA architecture

Problem definition: Define an allocation robustness metric $M = M(\vec{w})$ ($M \in \Re$) so that the maximization of M effectively maximizes the allowable ranges (robustness) $[0, w_k^{max}]$ for every external variable. Determine an allocation $T \rightarrow P$ for which $M(\vec{w})$ is maximized, subject to the feasibility constraints imposed by rate monotonic scheduling on each processor,

$$\forall j : 1 \le j \le m, \sum_{i:T_i \rightarrow P_j} \frac{T_i.e(\vec{w})}{T_i.p} \le n_j(2^{\frac{1}{n_j}} - 1),$$

where n_j is the number of tasks allocated to processor P_j. Return the feasible allocation corresponding to the $M^{max}(\vec{w})$, and the corresponding maximal allowable ranges for external variables $[0, w_k^{max}]$ ($1 \le k \le l$).

The problem calls for the maximization of a robustness measure against multiple external variables while the allocation stays feasible on every processor. This involves two issues: the choice of a meaningful robustness metric to maximize, and the development of an algorithm that finds the allocation with maximal robustness metric. It's also necessary to know the quality of maximal robustness it finds. We noticed that it is not a regular constrained optimization problem, because the constraints are not given explicitly but are functions of allocations. Exhaustively enumerating them is exponential and inefficient. These issues will be addressed in the paper. Rate monotonic scheduling was used because we have built the analysis upon the utilization bound developed by Oh and Baker [16].

The paper is organized as follows. Section 2 deals with finding robust allocations in the case of one environmental variable. An approximation ratio is developed for the robustness found by a first fit based allocation algorithm. The result will build the foundation for subsequent extensions to multiple environmental parameters. Section 3 addresses the problem of how to define a robustness metric in the case of multidimensional variables. Section 4 proposes a polynomial time algorithm for the multi-dimensional case, and an approximation ratio is derived. Section 5 experimentally compares our heuristic with branch and bound and random search. Finally, section 6 concludes this paper and discusses future research.

2. Robust allocation in one-dimensional case

In the previous work on single external workload variable (MAW) [10], a first-fit based allocation algorithm was developed to find the allocation with maximal robustness. By imposing two well-behaved restrictions on execution time profile functions, Juedes further developed a conditional two component approximation bound and an asymptotic approximation ratio for this algorithm [11]. In this section, we will extend the definition of well-behaved profile functions and present an improved bound on the one dimensional robustness when this first-fit based algorithm is used. The bound will later be used when dealing with the multiple dimensional case. Because the analysis relies on Oh and Baker's utilization bound [16], processors are assumed homogenous and use rate monotonic scheduler.

2.1. Well-behaved profile functions

We begin with some mathematical preparations on profile functions. Domains of running time profile functions $T_i.r(w)$ are first expanded from $\mathbb{Z}^+$ to $\mathbb{R}^+$, since the profile function forms (polynomial, logarithmic, exponential...) generally have mathematical definition on $\mathbb{R}^+$. For instance, $e(n) = nlogn$ of a sorting task is indeed well defined for $n \in \mathbb{R}$ and $n \in (0, \infty)$. Usually their domains are restricted to integers due to practical consideration (e.g. $\sqrt{2}$ items make no sense). However, working with the full domain of a function enables us to conveniently exploit intrinsic function properties like local derivatives, which is useful to speculate global function behaviors. We assume the profile functions are differentiable to the second order. This won't be a problem for normal profile function forms like polynomial, logarithmic, exponential... If a function profile is given in some non-continuous manner, it is always possible to bound or interpolate it with polynomials [12]. Next we make an assumption on profile function that will be useful to speculate a global bound later.

Definition 2.1 *A task's profile function $T_i.e(w)$ is* well-behaved *if: $\frac{d}{dw}T_i.e(w) \ge 0$ and $\frac{d^2}{dw^2}T_i.e(w) \ge 0$ ($w \in [0, w_{max}]$).*

The "well-behaved" functions are nondecreasing and convex within the range of allowable workload. For example, the function $e(w) = 0.1w + 10$ can be considered well-behaved, so can $wlogw$. Although $logw$ cannot. However, we may bound it with a linear function to render it well-behaved. The consequence is we would ignore the further saving of execution time offer by a logarithmic function beyond the linear function, and, thus, would not appreciate the potential improvement in maximal workload if logarithmic function were indeed used. But since we seek a worst case bound on the maximal workload, the bound will still hold under the improvement. From now on we assume profile functions are well-behaved.

We may formally introduce a *well-behaved function space* W, and a function $f(x) \in W$ if it is well-behaved by above definition. Subsequently, we may define well-behaved operators that conserve the space. With these operators, we may easily construct/destruct unknown well-behaved functions from/to known well-behaved functions.

Definition 2.2 *An operator $\odot$ is a well-behaved operator if: for $h = \odot(f, g, ...)$ with $f, g, ... \in W$, there is $h \in W$.*

Theorem 2.1 *Addition $+$ and multiplication $\times$ are well-behaved operators, and the unary operator* multiplication by constant *is also a well-behaved operator.*

The proofs are fairly straightforward by applying derivative chain rules, and they will be omitted to save space. As a result of the theorem, we may conveniently find system utilization function $U(w)$ is also well-behaved because: $U(w) = \sum_{i=1}^{n} T_i.e(w)/T_i.p$ and $\forall i, T_i.e(w) \in W$.

2.2. Robustness bound of a first fit based algorithm

Equipped with well-behavedness in task execution time profiles, we continue to examine the quality of robust allocation produced by a first fit based algorithm under one workload variable.

Briefly, the first fit based algorithm [10] combined a binary search with a first fit allocation to find the allocation with maximal robustness. It used a robustness metric that was defined as the maximal workload value allowed by an allocation (MAW). The algorithm performed binary search along the workload value while first fitting tasks with execution times fixed at that workload onto processors; the search terminated when the workload value could not be increased any further. The resultant feasible workload was the MAW and the allocation by first fit was the most robust allocation.

If this allocation algorithm is used and tasks have well-behaved profile functions, we found that it has an *absolute approximation ratio* and then an *asymptotic performance ratio* for large instances. For our maximization problem, the former is the tightest bound on $R_A(I) = \frac{OPT(I)}{A(I)}$ for all instances I, and the latter is the bound for large instances [5]. Let $OPT(I)$ be the optimal MAW workload that exists in the allocation problem I, and $FF(I)$ be the MAW produced by this first fit based algorithm; let r_{FF}^{1} be the absolute approximation ratio for 1 dimension and r_{FF}^{1*} be the asymptotic ratio, there is:

Theorem 2.2 $r_{FF}^{1} < \frac{2-2\delta}{\sqrt{2}-1-\delta}$, *where $\delta = \frac{U(0)}{m}$. Asymptotically when $OPT(I) \to \infty$, $r_{FF}^{1*} \le \frac{1-\delta}{\sqrt{2}-1-\delta}$.*

PROOF. The proof made use of the knowledge of well-behavedness in system utilization function $U(w)$ to speculate its behavior without knowing its exact form. Since the well-behavedness in $U(w)$ supplies local differential properties over a range, a good mathematical tool to exploit this is the Taylor expansion. $U(w)$ can be accurately expanded as a Taylor series when the *integral remainder* is used. Taylor expansion with the integral remainder has this general form:

$$f(x) = \sum_{k=0}^{n} \frac{1}{k!} f^{(k)}(x-c)^k + R_n(x), \text{ where}$$
$$R_n(x) = \frac{1}{n!} \int_c^x f^{(n+1)}(t)(x-t)^n dt. \quad (1)$$

It will be used in the proof next.

Let w_0 be the real value for which $U(w_0) = \sum_{i=1}^{n} \frac{T_i.e(w_0)}{T_i.p} = (\sqrt{2}-1)m$. We may assume $w_0 \ge 1$ since otherwise only zero workload ($\lfloor w_0 \rfloor$) can satisfy this utilization for feasible allocation, which becomes the problem of fixed execution time task allocation, and there is no value to consider the feasible range for allowed workload. Now for all $w \le w_0$, there is $U(w) \le U(w_0)$ since the system utilization function is well-behaved and nondecreasing. Therefore, by Oh and Baker [16], all workload value $w \le \lfloor w_0 \rfloor$ has a feasible allocation by first fit. Thus $FF(I) \ge \lfloor w_0 \rfloor$.

If we use the Taylor expansion to expand the system utilization $U(w)$ to first order about $w = 0$ and evaluate at w_0, we have

$$U(w_0) = U(0) + U'(0)w_0 + \int_0^{w_0} U''(t)(w_0 - t)dt.$$

Similarly, if evaluated at point cw_0 where $c \ge 1$ and $c \in R$,

$$U(cw_0) = U(0) + U'(0)cw_0 + \int_0^{cw_0} U''(t)(cw_0 - t)dt.$$

Then: $\quad \dfrac{U(cw_0) - U(0)}{cw_0} - \dfrac{U(w_0) - U(0)}{w_0} =$

$$\frac{1}{cw_0} \int_0^{cw_0} U''(t)(cw_0 - t)dt - \frac{1}{w_0} \int_0^{w_0} U''(t)(w_0 - t)dt$$
$$= \dots$$
$$= \frac{c-1}{cw_0} \int_0^{w_0} U''(t)t\, dt + \frac{1}{cw_0} \int_{w_0}^{cw_0} U''(t)(cw_0 - t)dt$$

The first term $\frac{c-1}{cw_0} \int_0^{w_0} U''(t)t\ dt \ge 0$ because $c \ge 1$ and $U''(t) \ge 0$ (well-behaved) and $t \ge 0$ for $t \in [0, w_0]$; the second term $\frac{1}{cw_0} \int_{w_0}^{cw_0} U''(t)(cw_0 - t)dt \ge 0$ because $U''(t) \ge 0$ and $cw_0 \ge t$ for $t \in [0, cw_0]$. As a result, $\frac{U(cw_0)-U(0)}{cw_0} - \frac{U(w_0)-U(0)}{w_0} \ge 0$, thus:

$$U(cw_0) \ge c[U(w_0) - U(0)] + U(0)$$
$$= c[(\sqrt{2}-1)m - U(0)] + U(0). \quad (2)$$

If we choose $c = \frac{m-U(0)}{(\sqrt{2}-1)m-U(0)}$, then $U(cw_0) \geq m$. Because it is impossible to utilize processors more than full, and $U(w)$ is non-decreasing, we have:

$$OPT(I) \leq cw_0 = \frac{m-U(0)}{(\sqrt{2}-1)m-U(0)} w_0 \qquad (3)$$

Since $FF(I) \geq \lfloor w_0 \rfloor$, there is:

$$\frac{OPT(I)}{FF(I)} \leq \frac{OPT(I)}{\lfloor w_0 \rfloor} \leq \frac{m-U(0)}{(\sqrt{2}-1)m-U(0)} \cdot \frac{w_0}{\lfloor w_0 \rfloor} < \frac{m-U(0)}{(\sqrt{2}-1)m-U(0)} \cdot 2 = \frac{2-2\delta}{\sqrt{2}-1-\delta}. \qquad (4)$$

We have expressed the workload independent system utilization $U(0)$ as δm, and used the fact that $\frac{w_0}{\lfloor w_0 \rfloor} < 2$ since $w_0 \geq 1$. Thus the absolute approximation ratio $r^1_{FF} < \frac{2-2\delta}{\sqrt{2}-1-\delta}$.

Asymptotically when $OPT(I) \to \infty$, the bound becomes tighter. From equation 3, there is $w_0 \geq OPT(I)/c$. When $OPT(I) \to \infty$, $w_0 \to \infty$. This leads to $\frac{w_0}{\lfloor w_0 \rfloor} = 1$. Plugging it into equation 4, we have:

$$\frac{OPT(I)}{FF(I)} \leq \frac{1-\delta}{\sqrt{2}-1-\delta}, \qquad (5)$$

and thus the asymptotic approximation ratio $r^{1*}_{FF} \leq \frac{1-\delta}{\sqrt{2}-1-\delta}$.

Notice that there also exists a special case where $FF(I) = OPT(I)$. This occurs when there is a task T_i for which $T_i.e(w_p)/T_i.p = 1$ and $U(w_p) \leq 0.414m$; in this case $w_p = OPI(I)$. Since first fit can always find an allocation when system utilization is less than Oh and Baker bound, it will have feasible allocation for w_p: $FF(I) = w_p$. Thus $FF(I) = OPT(I)$. □

3. A robustness metric for multiple dimensions

When a system depends on multiple external variables, the magnitude of a single workload is inadequate to capture overall robustness for all variables, and a more comprehensive robustness metric needs to be defined. Since a set of extrinsic attributes may be regarded as a vector, norm seems a potential metric for robustness. However, caution must be taken about how the metric mixes components, which should reflect the interest in this problem. The l_2 norm did not seem appropriate because it implies that the same robustness value may potentially be achieved by any point of external variables lying on a sphere of radius $\|x\|_2$, even if the point has all components equalling to 0 but just one: $x_k = \|x\|_2$. This is not desirable since external variables may not be tradable with each other in value and the loss of all but one may not be beneficial. For example, (a) the ability to handle 15 missile tracks alone cannot substitute (b) the ability to handle 5 missile tracks and 5 torpedo tracks, even though scenario (a) may appear to offer more robustness than (b) with such norm $15 > \sqrt{5^2+5^2} = 7.07$. This can guide the optimization algorithm to prefer the former and cause undesirable results. Another consideration in the choice of metric is relative importance among the components. It is natural that some external variable dimension may be regarded more important than others. Military statistics may indicate that for a combat type A, incoming missiles are twice as likely as torpedoes. Then handling more missile tracks becomes more important than torpedo tracks. Therefore, the metric construction also needs to take this into consideration, such as by the use of weighted components.

We chose our metric similar to l_∞. However, instead of picking the maximal component, we pick the minimal. To address the relative importance factor, weights are also attached. If we label it M:

$$M(\vec{w}) \equiv \min_{1\leq i\leq l}(k_i|w_i|) = \min_{1\leq i\leq l}(k_i w_i), \qquad (6)$$

assuming $k_i > 0$, and $w_i \geq 0$.

This choice is not uncommon [7, 15, 2]. It maximizes the minimal value range in *every* external variable while weighted by importance. Thus a system optimized by this metric can absorb maximum change in any external variable. If we look back on scenarios (a) and (b), now (a) would have a metric of 0 while (b) have a metric of 5 (equal weights), thus (b) would now be preferred.

To better understand the properties of this metric, its equi-value contour lines are plotted for $M(\vec{w}) = 1, 2$ in two dimensions in Figure 3. The metric value increases as the lines go out, and they are shaped as right angles parallel to the coordinate frame. In addition, 'origins' of all these contours reside on the line: $y = \frac{k_1}{k_2}x$, which can be generalized in multi-dimensional space to:

$$k_i w_i = t, \qquad (7)$$

where t is the parameter in the parametric format of the line.

In Figure 3, a point $\vec{w}_0 = (x_0, y_0) = (1/k_1, 1/k_2)$ was shown on this line. It has: $k_1 x_0 = k_2 y_0 = 1$ and thus a metric: $M(\vec{w}) = min(1,1) = 1$. Generally, if w is on the line, $M(\vec{w}) = min_{1\leq i\leq l}(k_i w_i) = t$, thus parameter t turns out to be the metric itself, and the line may also be characterized by:

$$w_i = \frac{M(\vec{w})}{k_i} \qquad (1 \leq i \leq l). \qquad (8)$$

For convenience of reference, we label it the *ray of origins*. Geometrically, any point on the line determines the whole $M(\vec{w}) = c$ contour line.

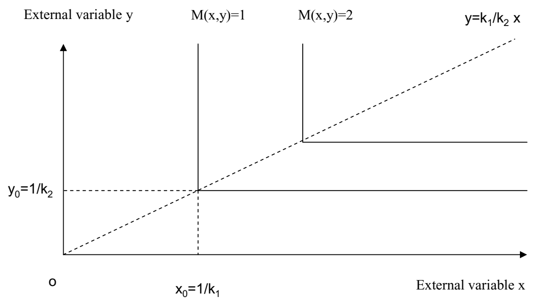


Figure 3. Contour lines of robustness metric in two dimensions

3.1. Search space for the maximal robustness metric

Now we further examine the robustness metric when it is considered in conjunction with properties of system tasks, since that is where the metric will be used. This understanding will help us design an efficient search for the maximal robustness value under multiple dimensions. Generally, an algorithm's running time does not decrease as its input size grows larger, thus we previously made the assumption that execution time functions are non-decreasing in one workload dimension [10]. When a profile function depends on multiple environmental variables, it is reasonable to assume the function also to be non-decreasing in each dimension while keeping others fixed, or mathematically

$$\frac{\partial T_i.e(\vec{w})}{\partial w_j} \geq 0 \qquad (1 \leq j \leq l, 1 \leq i \leq n). \tag{9}$$

For instance, a bin-packing approximation algorithm may run in $O(mn)$, where m and n is number of bins and blocks. Then the function is non-decreasing in either dimension. This yields an important property:

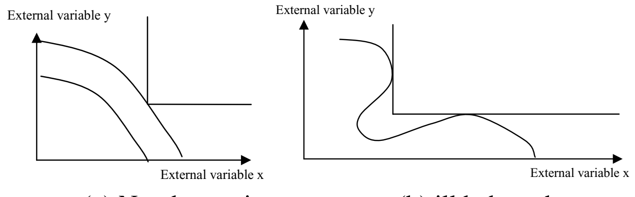
Lemma 3.1 *If tasks of a* DDRTS *have execution time profile functions that are non-decreasing in all EV dimensions, then for every allocation of the* DDRTS, *there can exist only one tangent point between the boundary line (or surface) of its feasible area for EVs, $f(\vec{w}) = c_1$, and the robustness metric's contour line (or surface), $M(\vec{w}) = c_2$. The tangent point occurs at the metric contour's origin.*

PROOF. Without loss of generality, we give the proof only for the two-dimensional case. Let the feasibility boundary of an allocation be described by $f(w_1, w_2) = c_1$, or equivalently $w_2 = g(w_1)$.

We first show that for every allocation, everywhere on the feasibility boundary line there is $\frac{dw_2}{dw_1} \leq 0$. Assume that $\exists$ somewhere $\frac{dw_2}{dw_1} > 0$. Let there be $w_2 = g(w_1)$ and $w_2' = g(w_1')$. Then it is possible that when $w_1' = w_1 + \Delta$ (for any $\Delta > 0$), $w_2' > w_2$. It follows that since the allocation is feasible at (w_1', w_2'), it should also be feasible at (w_1, w_2'), because $w_1 < w_1'$ decreases execution times of all tasks according to Equation 9, and tasks allocated on every processor stays schedulable. However, since (w_1, w_2) is on the feasibility boundary, (w_1, w_2') should not be feasible because $w_2' > w_2$. Thus we have reached a contradiction and the assumption was false. There is $\frac{dw_2}{dw_1} \leq 0$ everywhere on the feasibility boundary line.

This means that at any point on the feasibility boundary line of each allocation, the angle is greater than $\frac{\pi}{2}$, and a tangent contact with the metric's right angle contour at its origin will prevent any further crossing of the two lines. □

An illustration is depicted in Figure 4(a). An illustration of an ill-behaved feasibility boundary without the non-decreasing condition is demonstrated in a counterexample in Figure 4(b). As can be seen in (a), the tangent point corresponds to the origin of the metric contour and is easily found, while in the ill-behaved case, the tangent points have to be searched in general or solved for in special functions. Each allocation of a non-decreasing *DDRTS* contains one such special point; these points all reside on the same "ray of origins" line. Next we will see that one of them corresponds to the maximum robustness metric value, and the containing allocation is the robust allocation we wish to find.



(a) Nondecreasing (b) ill behaved

Figure 4. Interaction between feasible boundary and robustness metric contour line

Theorem 3.1 *If a* DDRTS *has tasks with non-decreasing execution time profile functions, then its maximum achievable robustness metric value, M^{max}, can be found by incrementally searching along the "ray of origins" line of the metric, $w_i = \frac{1}{k_i}t$ $(1 \leq i \leq l)$, until the metric value fails to correspond to any feasible allocation. The resulting maximum allowable values for environmental variables are $w_i^{max} = \frac{M^{max}}{k_i}$ $(1 \leq i \leq l)$.*

PROOF. For each allocation, the tangent point(s) between the feasibility boundary and the metric contour mathematically mean that the metric value is the maximum at the point(s). Because such a point is unique now and resides on the "ray of origins" line (by the lemma above), to find the overall maximum of this value among all allocations, we may search the line incrementally starting from $t = M(\vec{w}) = 0$. The search proceeds while at each value a feasible allocation can be found (with task execution times fixed by the value). If no feasible allocation can be found for $M(\vec{w}) + 1$, then $M^{max} = M(\vec{w})$. This is because if $M(\vec{w})+2$ or more were to produce a feasible allocation (say A) again, then $M(\vec{w})+1$ must have a feasible allocation (at least the same A) due to the non-decreasing condition (9) (a contradiction). Therefore it must be $M^{max} = M(\vec{w})$, and the resulting maximum allowable values for environmental variables are $w_i^{max} = \frac{M^{max}}{k_i}(1 \le i \le l)$ since the point resides on the "ray of origins." $\square$

The observation serves as the basis to develop a first-fit based approximation algorithm in multiple environmental variable dimensions next.

4. Multi-dimensional robust allocation and an optimality bound

In the previous section, it was shown that searching for the chosen maximal robustness metric only involves a special line, and it is equivalent to exhaustively searching the whole l dimensional space. The result is valuable to significantly reduce the algorithm's time complexity.

The algorithm to find the maximally robust allocation in multiple external variable dimension and its optimality is the topic of this section. It was demonstrated in one dimension that a binary search coupled with first-fit allocation efficiently produces good maximal allowable workload [10, 1]. Observing the linear, one-dimensional nature of the robustness metric search, we decided to derive from the first-fit based algorithm that has been tested. A linear search along the *ray of origins* will now serve as a driver that provides a set of constant external variable values to fix tasks' execution times, and a first-fit allocation algorithm then tests for a feasible allocation of tasks when given these constant execution times. If the allocation algorithm returns a feasible allocation, the search proceeds to the next set of external variable values; if the allocation algorithm reports no feasible allocation can be found, the search terminates. Thus the external variable values with the maximal robustness metric as well as corresponding allocation will be found. Quality of the algorithm will be proved analytically by an approximation ratio, and experimentally in the next section.

This algorithm of linear search coupled with an arbitrary allocation algorithm plug-in is listed in Algorithm 1; Algorithm 2 then lists the algorithm plug-in we chose: the greedy first-fit by Oh and Baker [16]. The plug-in will later allow us to compare performance of the allocation algorithm of first-fit with others such as branch-and-bound or random search. The rest of the section focuses on finding out the quality of the maximal robustness metric produced by this algorithm, as we did earlier for the one-dimensional case. The extension likewise starts from the definition of well-behaved functions in higher dimensions.

Algorithm 1 Linear search for maximal robustness metric

```
Input: $\langle T, P \rangle$
Output: A set of external variable values found to have maximal robustness metric
and their associated allocation
$metric = 0$
while AllocationAlgorithm($T(\{\frac{metric+1}{k_i}\}), P$)="Feasible" do
  $metric = metric + 1$
  store FeasibleAllocation
end while
$\{w_i^{max}\} = \{\frac{metric}{k_i}\}$
return $\{w_i^{max}\}$ and FeasibleAllocation
```

Algorithm 2 Greedy First Fit Allocation Algorithm of Oh and Baker [16]

```
Input: $\langle T(\{w_i\}), P \rangle$
Output: "Feasible" or "Not Feasible," and $FeasibleAllocation : T \longrightarrow P$
for each task $i$ do
  set $j = 1; n = |P|$;
  while job $i$ has not been allocated and $j \le n$ do
    set $n_j = |\{T_k | alloc(T_k) = P_j\}| + 1$;
    if $\left( \sum_{alloc(T_k)=P_j} \frac{T_k.e(\{w_i\})}{T_k.p} \right) + \frac{T_i.e(\{w_i\})}{T_i.p} \le n_j(2^{\frac{1}{n_j}} - 1)$ then
      set $alloc(T_i) = P_j$;
    else
      set $j = j + 1$;
    end if
  end while
  if $j > n$ then
    return "Not Feasible";
  end if
end for
FeasibleAllocation = alloc
return "Feasible"
```

Definition 4.1 *A task's profile function $T_i.e(\vec{w})$ is* well-behaved *in l dimensional external variable space if:* $\frac{\partial}{\partial w_i} T_i.e(\vec{w}) \ge 0 \ (1 \le i \le l)$, *and* $\frac{\partial^2}{\partial w_i \partial w_j} T_i.e(\vec{w}) \ge 0 \ (1 \le i, j \le l)$.

Many practical functions satisfy the condition, e.g. $O(n^2 + mn)$. Well-behaved function space and its operators can be defined similar to the one-dimensional case.

Theorem 4.1 *When l external variables exist, if the system of tasks have well-behaved execution time profile functions and are allocated by this algorithm, then its robustness as measured by the metric has an absolute approximation ratio $r_{FF}^l < \frac{2-2\delta}{\sqrt{2}-1-\delta}$, where $\delta = \frac{U(0)}{m}$, and an asymptotic performance ratio $r_{FF}^{l*} \le \frac{1-\delta}{\sqrt{2}-1-\delta}$.*

PROOF. We noticed that these well-behaved functions are sufficiently non-decreasing, therefore the linear search approach is applicable. We first express the ray of origins in vector form: $\vec{l} = \sum_{i=1}^{l} \frac{t}{k_i} \vec{e}_i$, where $t \in R$ and $t \ge 0$. System utilization is a function of the external variables vector, $U(\vec{w})$, so its change with respect to t along the ray of origins is:

$$dU(\vec{w}) = \nabla U(\vec{w}) \cdot d\vec{l} = \sum_{i=1}^{l}(\vec{e}_i \frac{\partial U(\vec{w})}{\partial w_i}) \cdot \sum_{j=1}^{l}(\vec{e}_j \frac{dt}{k_j})$$

$$= \sum_{i=1}^{l} \frac{1}{k_i} \frac{\partial U(\vec{w})}{\partial w_i} dt \quad \Rightarrow \quad \frac{dU(\vec{w})}{dt} = \sum_{i=1}^{l} \frac{1}{k_i} \frac{\partial U(\vec{w})}{\partial w_i} \geq 0,$$

since $U(\vec{w})$ is well-behaved and $\frac{\partial U(\vec{w})}{\partial w_i} \geq 0$. In addition, its double derivative is:

$$\frac{d^2U(\vec{w})}{dt^2} = \sum_{i=1}^{l} \frac{1}{k_i} \frac{d}{dt}(\frac{\partial U(\vec{w})}{\partial w_i})$$

$$= \sum_{i=1}^{l} \frac{1}{k_i} \sum_{j=1}^{l} \frac{\partial^2 U(\vec{w})}{\partial w_j \partial w_i} \frac{dw_j}{dt} = \sum_{i=1}^{l}\sum_{j=1}^{l} \frac{1}{k_i} \frac{\partial^2 U(\vec{w})}{\partial w_i \partial w_j} \frac{1}{k_j} \geq 0$$

since $U(\vec{w})$ is well-behaved and $\frac{\partial^2 U(\vec{w})}{\partial w_i \partial w_j} \geq 0$.

Now, in effect, parameter t is identical to the workload variable in the one-dimensional approximation ratio since $t \in R$, $t \geq 0$, and system utilization function $U(\vec{w}) = U(t)$ is well-behaved in one dimension since $U'(t) \geq 0$ and $U''(t) \geq 0$. Further, we note that it was previously shown by Equation (8) that the parameter t is equivalent to the robustness metric $M(\vec{w})$. Therefore, $r_{FF}^{l} < \frac{2-2\delta}{0.414-\delta}$ and asymptotically $r_{FF}^{l*} \leq \frac{1-\delta}{0.414-\delta}$. $\Box$

5. Experiments

In order to verify the analytical results, four experiments were conducted. **Experiment 1** compared the outcome of the maximal robustness metric found with linear search against that of exhaustive search. Its purpose was to verify the claim in Section 3 that a special linear search is sufficient and efficient under multiple external variable space. **Experiment 2** measured robustness metrics produced by first-fit along with an optimal brand-and-bound approach. Its purpose was to check whether Section 4's approximation ratio indeed holds. **Experiment 3** compared *running time* and *robustness metric quality* produced by first-fit against the baselines of random search and branch-and-bound. The purpose was to reveal whether first-fit is still a good choice in multiple dimensions as it was in the single workload case[10, 1]. **Experiment 4** solved an example to demonstrate application of this approach.

The experiments were done through simulations in which various system parameters were controlled, and these include: task period, deadline, processor number, processor speed, number of external variables, number of external variable dependent tasks, number of external variable independent tasks, and the particular execution time profile function form of each task. The profile functions were constructed on a set of function basis: $\{w_i^2 \log w_i, w_i^2, w_i \log w_i, w_i | 1 \leq i \leq l\}$. The number of basis to use was chosen randomly according to a distribution supplied by user, and the projection on each base was also generated randomly. By doing so, it was intended to represent general problem instances with the randomly produced samples. All experiments were performed on a Pentium 4 PC running Linux 2.4.22. Two other allocation algorithms were chosen to compare with first-fit which are random search and optimal branch-and-bound, because they are good baselines to measure performance against. However for the algorithms' speed considerations, we assumed there were two external variables. The result of the first experiment is next.

Table 1 Max robustness metric found using Branch-and-Bound allocation coupled with: (1) linear search (LS) and (2) exhaustive search (ES) in external variable space

Dep tasks	Indep. tasks	Procs	Max robust. metric (LS)	Max robust. metric (ES)	Equal (Y/N)	LS Time (s)	ES Time (s)	LS Faster (Y/N)
8	2	3	42	42	Y	0	0.04	Y
13	2	4	32	32	Y	0	0.21	Y
18	2	5	30	30	Y	0.02	0.37	Y
23	2	6	27	27	Y	0.09	0.92	Y
28	2	7	28	28	Y	0.16	1.05	Y
33	2	8	30	30	Y	0.73	1.73	Y
18	2	5	91	91	Y	0.06	1.12	Y
23	2	6	86	86	Y	0.29	2.88	Y
21	4	6	36	36	Y	0.03	0.75	Y
19	6	6	31	31	Y	0.02	0.5	Y
17	8	6	34	34	Y	0.02	0.64	Y
15	10	6	39	39	Y	0.05	0.69	Y

Experiment 1 This experiment was used to validate that a special linear search is equivalent to exhaustively searching the external variables space. The linear search as shown in Algorithm 1 increased the magnitude of the robustness metric which supplied external variable values to drive the allocation algorithm; the exhaustive search, on the other hand, enumerated the entire space of external variable values and recorded the point with maximal robustness metric for which the allocation algorithm finds a feasible allocation. To ensure no feasible allocation would be missed, we chose optimal branch-and-bound as the allocation algorithm. The results are listed in Table 1.

Data from twelve instances are given in the table. In each instance, the number of external variables' dependent tasks, independent tasks, and processors were varied. We expected to see that under various instances, algorithms using linear search produced the same max robustness metric as the exhaustive search. This was verified from the 4th and 5th columns. In addition, it was shown from the 7th and 8th columns that the algorithm using linear search is significantly faster than exhaustive search.

Experiment 2 The second experiment was designed to verify the robustness bound of first-fit that was derived in Theorem 4.1. Six instances were given with 5 to 30 tasks to be allocated to 5 processors. Tasks in each instance contained 80% of external variable dependent tasks and the remaining were tasks with fixed execution time. Branch-and-bound algorithm was used to find the allocation with optimal robustness metric value. The results are listed in Table 2. The experiment showed that optimality of robustness metric found by first fit did fall within the approximation ratio we computed analytically. Under many instances we tested, its solution quality was found to be very close or just equal to the optimal. A careful look at the data assures this was not simply a result of the special case in Theorem 2.2. Therefore this suggested that the first-fit based algorithm may have a much better average case behavior than the worst-case bound.

Table 2 Experimental verification of robustness metric bound of FF algorithm

	5 Processors		Robustness Analysis		Robustness Experiment			Bound Correct?
#	Dep Tasks	Indep Tasks	δ	$r_{FF}^{l=2}$	Metric (FF)	Metric (BB)	BB/FF	(BB/FF<r)
1	4	1	0.012	4.91	3104	3104	1.00	yes
2	8	2	0.024	5.01	2387	2387	1.00	yes
3	12	3	0.047	5.19	1697	1697	1.00	yes
4	16	4	0.058	5.29	854	868	1.02	yes
5	20	5	0.077	5.48	810	830	1.02	yes
6	24	6	0.092	5.64	722	728	1.01	yes

Experiment 3 As mentioned at the beginning of section 4, we wanted to experimentally find out whether first-fit is a reasonably good choice of allocation algorithm. Comparisons were made based on maximum robustness metric found and algorithm running time with two baseline allocation algorithms: optimal branch-and-bound, and random search. To speed up the exhaustive process, branch-and-bound was programmed to begin searching on top of first-fit's results. In order to improve result quality of random search, 10000 iterations of trials were allowed, which meanwhile boosted its running time. Results of the experiment are shown in Table 3. The data showed that robustness found by first-fit approximated branch-and-bound; when processors were faster and more numerous, differences between first-fit and random search became more significant. This could be due to the fact that more state combinations were available and the probability that random search stumbled onto good solutions within a given number of trials was reduced. On criterion of robustness quality, the grades for random, first-fit, and branch-and-bound were ok, good, and best, respectively. Running time wise, first-fit was the best and the advantage against branch-and-bound grew larger when instance grew larger. Running time of random search was not on the same order as the other two. This was due to the large number of trials required for good robustness quality. Using the two criteria, our evaluation was that first-fit is an efficient allocation algorithm to choose for the multiple external variable problem. It gives good solutions with fast running time, which is valuable for online applications, and scales better than the other algorithms examined.

Table 3 Comparisons of max robustness metric quality and alg running time among FF, BB, and RN

Dep Tasks	Indep. Tasks	Procs	Proc speed factor	Max robust. metric (RN)	Max robust. metric (FF)	Max robust. metric (BB)	RN Time (sec)	FF Time (sec)	BB Time (sec)
8	2	3	100	14	14	14	23.77	0	0.01
12	3	4	100	14	14	14	34.99	0	0.01
16	4	5	100	14	14	15	44.06	0	0
20	5	6	100	14	14	14	55.18	0.01	0.02
24	6	7	100	12	12	12	66.37	0.01	0.07
16	4	7	3000	126	129	130	8.48	0.01	0.02
16	4	10	3000	134	149	152	8.67	0.01	1.66
			Evaluation:	**ok**	**good**	**best**	**poor**	**best**	**ok**

Table 4 Max robustness metric found with RN,FF,BB algorithms under three scenarios

#	Detect	Engage	Guide	Procs	RN	FF	BB(OPT)
1	20	5	10	8	170	178	178
2	6	5	3	5	98	105	105
3	8	8	8	5	53	54	54

Experiment 4 In the last experiment, we applied the approaches we developed to the modelling and analysis of a *Dynbench* missile defense testbed [18]. There were three type of tasks in the system: Detect, Engage and Guidance, and their execution times were dependent on two external variables: r and m, which were the number of radar tracks and real threats. Their profiling functions were found to be:
Detect: $e_1(r, m) = 0.0869r^2 + 15.4374r + 615\ \mu s$
Engage: $e_2(r, m) = 12897m + 45610\ \mu s$
Guide: $e_3(r, m) = 0.0869r^2 + 15.4374r + 12903.909m + 46476\ \mu s$
The objective was to maximize the minimum of r and m that can be handled by an allocation. Three scenarios were tested for the system and three algorithms were used to find the maximum robustness. Their results are shown next to each other in Table 4. Similar to the previous experiment, first-fit produced very good robustness, in this case the same as the optimum achieved by branch-and-bound, while random search with 10000 iterations did not perform as well. The scenarios of this experiment further corroborated previous discoveries.

6. Conclusions

Robust task allocation strategy is desired by distributed real-time systems affected by dynamically changing environments. We have introduced a metric that measures robustness in an allocation and then developed a task allocation algorithm that finds the maximally robust allocation as measured by the metric. Quality of the robustness was derived analytically and was experimentally verified in four experiments. The algorithm has a polynomial running time which is ideal to be used online. In future work, we will continue to improve the robustness results. The algorithm will be tested in software systems on real hardware platforms and be incorporated into resource management middleware.

References

[1] E. Aber, F. Drews, D. Gu, D. Juedes, A. Lenharth, D. Parrott, L. Welch, H. Zhao, and D. Fleeman. Experimental comparison of heuristic and optimal resource allocation algorithms for maximizing allowable workload in dynamic, distributed real-time systems. In *6th Brazilian Workshop on Real-Time Systems*, 2004.

[2] S. Ali, J.-K. Kim, Y. Yu, S. B. Gundala, S. Gertphol, H. J. Siegel, A. A. Maciejewski, and V. Prasanna. Utilization-based heuristics for statically mapping real-time applications onto the HiPer-D heterogeneous computing system. In *11th IEEE Heterogeneous Computing Workshop (HCW 2002) in Proceedings of the 16th International Parallel and Distributed Processing Symposium*, 2002.

[3] S. Ali, A. A. Maciejewski, H. J. Siegel, and J.-K. Kim. Definition of a robustness metric for resource allocation. In *Proceedings International Parallel and Distributed Processing Symposium*, page 10, 2003.

[4] R. Bettati and J. W. Liu. End-to-end scheduling to meet deadlines in distributed systems. In *Proceedings of the 12th International Conference on Distributed Computing Systems*, pages 452–459, 1992.

[5] J. Blazewicz, K. H. Ecker, E. Pesch, G. Schmidt, and J. Weglarz. *Scheduling Computer and Manufacturing Processes*. Springer Heidelberg, New York, 2001.

[6] D. Fleeman, M. Gillen, A. Lenharth, M. Delany, L. Welch, D. Juedes, and C. Liu. Quality-based adaptive resource management architecture (qarma): A corba resource management service. In *The 12th IPDPS Workshop on Parallel and Distributed Real-Time Systems (WPDRTS 04)*, Santa Fe, New Mexico, April 2004.

[7] S. Gertphol, Y. Yu, S. B. Gundala, and V. Prasanna. A metric and mixed-integer-programming-based approach for resource allocation in dynamic real-time systems. In *Proceedings of the International Parallel and Distributed Processing Symposium*, 2002.

[8] D. Gu, F. Drews, and L. Welch. A characterization of task allocation problems for dynamic distributed real-time systems. In *16th IASTED International Conference on Parallel and Distributed Computing and Systems, Cambridge, MA*, 2004.

[9] X. S. Hu, T. Zhou, and E. H.-M. Sha. Estimating probabilistic timing performance for real-time embedded systems. In *IEEE Transactions on Very Large Scale Integration Systems*, volume 9, pages 833–844, 2001.

[10] D. Juedes, F. Drews, L. Welch, and D. Fleeman. Heuristic resource allocation algorithms for maximizing allowable workload in dynamic, distributed, real-time systems. In *The 12th Workshop on Parallel and Distributed Real-Time Systems*, 2004.

[11] D. Juedes, L. Welch, F. Drews, and D. Fleeman. Resource allocation algorithms for maximizing allowable workload in dynamic, distributed real-time systems. Technical report, Center for Intelligent, Distributed, and Dependable Systems, Ohio University, 2003.

[12] D. R. Kincaid and E. W. Cheney. *Numerical Analysis: Mathematicas of Scientific Computing*. Brooks Cole, Pacific Grove, CA, 3 edition, 2001.

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

[14] S. Manolache, P. Eles, and Z. Peng. Optimization of soft real-time systems with deadline miss ratio constraints. In *10th IEEE Real-Time and Embedded Technology and Applications Symposium*, 2004.

[15] M. D. Natale and J. A. Stankovic. Dynamic end-to-end guarantees in distributed real time systems. In *Proceedings. Real-Time Systems Symposium*, pages 216–227, 1994.

[16] D.-I. Oh and T. P. Baker. Utilization bounds for N-processor rate monotonic scheduling with stable processor assignment. *Real Time Systems Journal*, 15(1):183–193, 1998.

[17] B. Ravindran, L. R. Welch, and B. A. Shirazi. Resource management middleware for dynamic, dependable real-time systems. *The Journal of Real-time Systems*, 20(2):183–196, 2000.

[18] Z. Tan. Producing application cpu profiles in dynbench via curve fitting. Technical report, Center for Intelligent, Distributed, and Dependable Systems, Ohio University, 2003.

[19] T.-S. Tia, Z. Deng, M. Shankar, M. Storch, J. Sun, L.-C. Wu, and J.-S. Liu. Probablistic performance guarantee for real-time tasks with varying computation times. In *IEEE Real-Time Technology and Applications Symposium*, 1995.

[20] E. Wandeler, A. Maxiaguine, and L. Thiele. Quantitative characterization of event streams in analysis of hard real-time applications. In *10th IEEE Real-Time and Embedded Technology and Applications Symposium*, 2004.

[21] L. R. Welch and B. A. Shirazi. A dynamic real-time benchmark for assessment of qos and resource management technology. In *Proceedings of the IEEE Real-Time Technology and Applications Symposium*, pages 36–45, 1999.

[22] Y. Zhou, L. R. Welch, E.-N. Huh, C. Alexander, and D. Lawrence. Execution time analysis for dynamic, periodic processes. In *The 9th Workshop on Parallel and Distributed Real-Time Systems*, 2001.