Designing High-Quality Embedded Control Systems with Guaranteed Stability

Amir Aminifar[1], Soheil Samii[1,2], Petru Eles[1], Zebo Peng[1], Anton Cervin[3]
[1]Department of Computer and Information Science, Linköping University, Sweden
[2]Semcon AB, Linköping, Sweden
[3]Department of Automatic Control, Lund University, Sweden
{amir.aminifar,petru.eles,zebo.peng}@liu.se, soheil.samii@semcon.com, anton@control.lth.se

***Abstract*—Many embedded systems comprise several controllers sharing available resources. It is well known that such resource sharing leads to complex timing behavior that degrades the quality of control, and more importantly, can jeopardize stability in the worst-case, if not properly taken into account during design. Although stability of the control applications is absolutely essential, a design flow driven by the worst-case scenario often leads to poor control quality due to the significant amount of pessimism involved and the fact that the worst-case scenario occurs very rarely. On the other hand, designing the system merely based on control quality, determined by the expected (average-case) behavior, does not guarantee the stability of control applications in the worst-case. Therefore, both control quality and worst-case stability have to be considered during the design process, i.e., period assignment, task scheduling, and control-synthesis. In this paper, we present an integrated approach for designing high-quality embedded control systems, while guaranteeing their stability.**

I. Introduction and Related Work

Many embedded systems comprise several control applications sharing available resources. The design of embedded control systems involves two main tasks, synthesis of the controllers and implementation of the control applications on a given execution platform. Controller synthesis comprises period assignment, delay compensation, and control-law synthesis. Implementation of the control applications, on the other hand, is mostly concerned with allocating computational resources to applications (e.g., mapping and scheduling).

Traditionally, the problem of designing embedded control systems has been dealt with in two independent steps, where first the controllers are synthesized and, second, applications are implemented on a given platform. However, this approach often leads to either resource under-utilization or poor control performance and, in the worst-case, may even jeopardize the stability of control applications because of timing problems which can arise due to certain implementation decisions [1], [2]. Thus, in order to achieve high control performance while guaranteeing stability even in the worst-case, it is essential to consider the timing behavior extracted from the system schedule during control synthesis and to keep in view control performance and stability during system scheduling. The issue of control–scheduling co-design [2] has become an important research direction in recent years.

Rehbinder and Sanfridson [3] studied the integration of off-line scheduling and optimal control. They found the static-cyclic schedule [4] which minimizes a quadratic cost function. Although the stability of control applications is guaranteed, the authors mention the intractability of their method and its applicability only for systems limited to a small number of controllers. Majmudar et al. [5] proposed a performance-aware static-cyclic scheduler synthesis approach for control systems. The optimization objective is based on the $\mathcal{L}_\infty$ to RMS gain. Recently, Goswami et al. [6] proposed a time-triggered [4] implementation method for mixed-criticality automotive software. The optimization is performed considering an approximation of a quadratic cost function and with assuming the absence of noise and disturbance. Considering time-triggered implementation, it is straightforward to guarantee stability, in the absence of noise and disturbance, as a consequence of removing the element of jitter. However, to completely avoid jitter, the periods of applications are constrained to be related to each other (harmonic relationship). Therefore, such approaches can lead to resource under-utilization and, possibly, poor control performance due to long sampling periods [3], [7]. Nevertheless, the approaches in [3], [5], and [6] are restricted to static-cyclic and time-triggered scheduling.

Seto et al. [8] found the optimal periods for a set of controllers on a uniprocessor platform with respect to a given performance index. Bini and Cervin [9] proposed a delay-aware period assignment approach to maximize the control performance measured by a standard quadratic cost function on a uniprocessor platform. Ben Gaid et al. [10] considered the problem of networked control systems with one control loop. The objective is to minimize a quadratic cost function. Cervin et al. [11] proposed a control–scheduling co-design procedure to yield the same relative performance degradation for each control application. Zhang et al. [12] considered control–scheduling

co-design problem where the objective function to be optimized is the sum of the H_∞ norm of the sensitivity functions of all controllers. In our previous work [13] and [14], we proposed optimization methodologies for integrated mapping, scheduling, and control synthesis to maximize control performance by minimizing a standard quadratic cost on a distributed platform.

In order to capture control performance, two kinds of metrics are often used: (1) stochastic control performance metrics and (2) robustness (stability-related) metrics. The former identify the expected (*mathematical expectation*) control performance of a control application, whereas the latter are considered to be a measure of the worst-case control performance. Although considering both the expected control performance and worst-case control performance during the design process is crucial, previous work only focuses on one of the two aspects. The main drawback of such approaches, e.g., based solely on the expected control performance, is that the resulting high expected performance design solution does not necessarily satisfy the stability requirements in the worst-case scenario. On the other hand, considering merely the worst-case, often results in a system with poor expected control performance. This is due to the fact that the design is solely tuned to a scenario that occurs very rarely. Thus, even though the overall design optimization goal should be the expected control performance, taking the worst-case control stability into consideration during design space exploration is indispensable.

In this paper, for the first time, to the best of our knowledge, we propose an integrated control–scheduling approach to design embedded control systems with guarantees on the worst-case robustness, where the optimization objective is the expected control performance.

The paper is organized as follows. In the next section, we present the system model, i.e., plant, platform and application models. Section III discusses the metrics for the expected and the worst-case control performance, and control synthesis. Analysis of real-time attributes is outlined in Section IV. In Section V, we present motivational examples and in Section VI, we formulate our control–scheduling co-design problem. The proposed design flow is discussed in Section VII. Section VIII contains the experimental setup and results. Finally, the paper will be concluded in Section IX.

II. System Model

The system model is determined by the plant model, the platform and application model.

A. Plant Model

Let us consider a given set of plants $\mathbf{P}$. Each plant P_i is modeled by a continuous-time system of equations

$$\begin{aligned} \dot{\boldsymbol{x}}_i &= A_i \boldsymbol{x}_i + B_i \boldsymbol{u}_i + \boldsymbol{v}_i, \\ \boldsymbol{y}_i &= C_i \boldsymbol{x}_i + \boldsymbol{e}_i, \end{aligned} \tag{1}$$

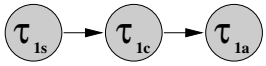


Figure 1. Example of modeling a controller as a task graph

where $\boldsymbol{x}_i$ and $\boldsymbol{u}_i$ are the plant state and control signal, respectively. The additive plant disturbance $\boldsymbol{v}_i$ is a continuous-time white-noise process with zero mean and given covariance matrix R_{1i}. The plant output is denoted by $\boldsymbol{y}_i$ and is sampled periodically with some delays at discrete time instants—the measurement noise $\boldsymbol{e}_i$ is discrete-time Gaussian white-noise process with zero mean and covariance R_{2i}. The control signal will be updated periodically with some delays at discrete time instants and is held constant between two updates by a hold-circuit in the actuator [15].

For instance, an inverted pendulum can be modeled using Equation 1 with $A_i = \left[\begin{smallmatrix} 0 & 1 \\ g/l_i & 0 \end{smallmatrix}\right]$, $B_i = \left[\begin{smallmatrix} 0 & g/l_i \end{smallmatrix}\right]^\mathsf{T}$, and $C_i = \left[\begin{smallmatrix} 1 & 0 \end{smallmatrix}\right]$, where $g \approx 9.81$ m/s^2 is the gravitational constant and l_i is the length of pendulum P_i. The two states in $\boldsymbol{x}_i = \left[\begin{smallmatrix} \phi_i \\ \dot{\phi}_i \end{smallmatrix}\right]$ are pendulum position ϕ_i and speed $\dot{\phi}_i$. For plant disturbance and measurement noise, we have $R_{1i} = B_i B_i^\mathsf{T}$ and $R_{2i} = 0.1$, respectively.

B. Platform and Application Model

The platform considered in this paper is a uniprocessor. For each plant $P_i \in \mathbf{P}$ there exists a corresponding control application denoted by $\Lambda_i \in \mathbf{\Lambda}$, where $\mathbf{\Lambda}$ indicates the set of applications in the system.

Each application Λ_i is modeled as a task graph. A task graph consists of a set of tasks and a set of edges, identifying the dependencies among tasks. Thus, an application is modeled as an acyclic graph $\Lambda_i = (\mathbf{T}_i, \mathbf{\Gamma}_i)$, where $\mathbf{T}_i$ denotes the set of tasks and $\mathbf{\Gamma}_i \subset (\mathbf{T}_i \times \mathbf{T}_i)$ is the set of dependencies between tasks. We denote the j-th task of application Λ_i by τ_{ij}. The execution-time, c_{ij}, of the task τ_{ij} is modeled as a stochastic variable with probability function ξ_{ij},[1] bounded by the best-case execution-time c_{ij}^b and the worst-case execution-time c_{ij}^w. Further, the dependencies between tasks τ_{ij} and τ_{ik} are indicated by $\gamma_{ijk} = (\tau_{ij}, \tau_{ik}) \in \mathbf{\Gamma}_i$.

Control applications can typically provide a satisfactory control performance within a range of sampling periods [15]. Hence, each application Λ_i can execute with a period $h_i \in \mathbf{H}_i$, where $\mathbf{H}_i$ is the set of suggested periods application Λ_i can be executed with. However, the actual period for each control application is determined during the co-design procedure, considering the direct relation between scheduling parameters and control-synthesis.

[1]The probability function ξ_{ij} is only used for the system simulation which is utilized in delay compensation for controller design and computing the expected control performance. Worst-case stability guarantees only depend on the worst-case and best-case execution times.

A simple example of modeling a control application as a task graph is shown in Figure 1. The control application Λ_i has three tasks, where τ_{is}, τ_{ic}, and τ_{ia} indicate the sensor, computation, and actuator tasks, respectively. The arrows between tasks indicate the dependencies, meaning that, for instance, the computation task τ_{ic} can be executed only after the sensor task τ_{is} terminates its execution.

III. Control Performance and Synthesis

In this section, we introduce control performance metrics both for the expected and worst-case. We also present preliminaries related to control synthesis.

A. Expected Control Performance

In order to measure the expected quality of control for a controller Λ_i for plant P_i, we use a standard quadratic cost [15]

$$J^{\mathrm{e}}_{\Lambda_i} = \lim_{T\to\infty} \frac{1}{T}\mathbb{E}\left\{\int_0^T \begin{bmatrix} \boldsymbol{x}_i \\ \boldsymbol{u}_i \end{bmatrix}^{\mathsf{T}} Q_i \begin{bmatrix} \boldsymbol{x}_i \\ \boldsymbol{u}_i \end{bmatrix} dt\right\}. \quad (2)$$

The weight matrix Q_i is given by the designer, and is a positive semi-definite matrix with weights that determine how important each of the states or control inputs are in the final control cost, relative to others ($\mathbb{E}\{\cdot\}$ denotes the expected value of a stochastic variable). To compute the expected value of the quadratic control cost $J^{\mathrm{e}}_{\Lambda_i}$ for a given delay distribution, the Jitterbug toolbox is employed [16].

The stability of a plant can be investigated using the Jitterbug toolbox in the mean-square sense if all time-varying delays can be modeled by independent stochastic variables. However, since it is not generally possible to model all time-varying delays by independent stochastic variables, this metric (Equation 2) is not appropriate as a guarantee of stability in the worst-case.

B. Worst-Case Control Performance

The worst-case control performance of a system can be quantified by an upper bound on the gain from the uncertainty input to the plant output. For instance, assuming the disturbance d to be the uncertainty input, the worst-case performance of a system can be measured by computing an upper bound on the worst-case gain G from the plant disturbance d to the plant output y. The plant output is then guaranteed to be bounded by

$$\|y\| \leq G\|d\|.$$

In order to measure the worst-case control performance of a system, we use the Jitter Margin toolbox [17]. The Jitter Margin toolbox provides sufficient conditions for the worst-case stability of a closed-loop system. Moreover, if the closed-loop system is stable, the Jitter Margin toolbox can measure the worst-case performance of the system. The worst-case control cost is captured by

$$J^{\mathrm{w}}_{\Lambda_i} = G(P_i, \Lambda_i, L_i, \Delta^{\mathrm{w}}_{is}, \Delta^{\mathrm{w}}_{ia}). \quad (3)$$

The discrete-time controller designed for sampling period h_i is denoted by Λ_i. The nominal sensor–actuator (input–output) delay for control application Λ_i is denoted by L_i. The *worst-case* jitters in response times of the sensor (input) and the actuator (output) tasks of the controller Λ_i are denoted by Δ^{w}_{is} and Δ^{w}_{ia} (see Section IV, Figure 2), respectively. It should be noted that a finite value for the worst-case control cost $J^{\mathrm{w}}_{\Lambda_i}$ represents a guarantee of stability for control application Λ_i in the worst-case.

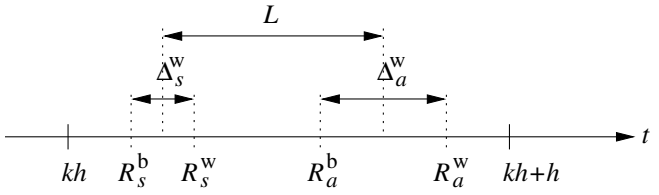


Figure 2. Graphical interpretation of the nominal sensor–actuator delay L, worst-case sensor jitter Δ^{w}_s, and worst-case actuator jitter Δ^{w}_a

C. Control Synthesis

For a given sampling period h_i and a given, constant sensor–actuator delay (i.e., the time between sampling the output $\boldsymbol{y}_i$ and updating the controlled input $\boldsymbol{u}_i$), it is possible to find the control-law $\boldsymbol{u}_i$ that minimizes the expected cost $J^{\mathrm{e}}_{\Lambda_i}$ [15]. Thus, the optimal controller can be designed if the delay is constant at each periodic instance of the control application. Since the overall performance of the system is determined by the expected control performance, the controllers shall be designed for the expected average behavior of the system. System simulation is performed to obtain the delay distribution and the expected sensor–actuator delay and the controllers are designed to compensate for this expected amount of delay. In order to synthesize the controller we use MATLAB and the Jitterbug toolbox [16].

The sensor–actuator delay is in reality not constant at runtime due to interference from other applications competing for the shared resources. The quality of the constructed controller is degraded if the sensor–actuator delay distribution is different from the constant one assumed during the control-law synthesis. The overall expected control quality of the controller for a given delay distribution is obtained according to Section III-A.

IV. Jitter and Delay Analyses

In order to apply the worst-case control performance analysis, we shall compute the three parameters mentioned in Section III-B (Equation 3), namely, the nominal sensor–actuator delay L_i, worst-case sensor jitter Δ^{w}_{is},

and worst-case actuator jitter Δ^{w}_{ia} for each control application Λ_i. Figure 2 illustrates the graphical interpretation of these three parameters. To obtain these parameters, we can apply response-time analysis as follows,

$$\begin{aligned} \Delta^{\mathrm{w}}_{is} &= R^{\mathrm{w}}_{is} - R^{\mathrm{b}}_{is}, \\ \Delta^{\mathrm{w}}_{ia} &= R^{\mathrm{w}}_{ia} - R^{\mathrm{b}}_{ia}, \\ L_i &= \left(R^{\mathrm{b}}_{ia} + \frac{\Delta^{\mathrm{w}}_{ia}}{2} \right) - \left(R^{\mathrm{b}}_{is} + \frac{\Delta^{\mathrm{w}}_{is}}{2} \right), \end{aligned} \tag{4}$$

where R^{w}_{is} and R^{b}_{is} denote the worst-case and best-case response times for the sensor task τ_{is} of the control application Λ_i, respectively. Analogously, R^{w}_{ia} and R^{b}_{ia} are the worst-case and best-case response times for the actuator task τ_{ia} of the same control application Λ_i.

In this paper, we consider fixed-priority scheduling and we give a brief overview on computing the worst-case and best-case response times.

A. Worst-Case Response Time Analysis

Under fixed-priority scheduling, assuming deadline $D_i \leq h_i$ and an independent task set[2], the worst-case response time of task τ_{ij} can be computed by the following equation [18],

$$R^{\mathrm{w}}_{ij} = c^{\mathrm{w}}_{ij} + \sum_{\tau_{ab} \in hp(\tau_{ij})} \left\lceil \frac{R^{\mathrm{w}}_{ij}}{h_a} \right\rceil c^{\mathrm{w}}_{ab}, \tag{5}$$

where $hp(\tau_{ij})$ denotes the set of higher priority tasks for the task τ_{ij}. Equation 5 is solved by fixed-point iteration starting with e.g., $R^{\mathrm{w}}_{ij} = c^{\mathrm{w}}_{ij}$.

B. Best-Case Response Time Analysis

For the best-case response time analysis under fixed priority scheduling, we use the bound given by equation [19]

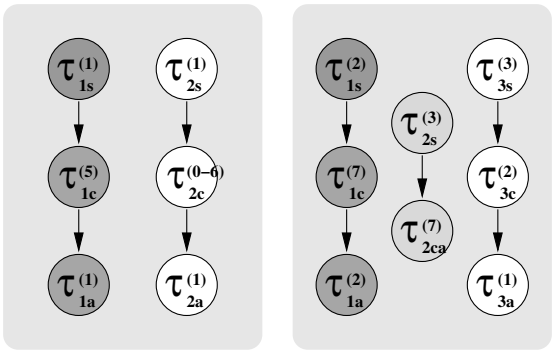
$$R^{\mathrm{b}}_{ij} = c^{\mathrm{b}}_{ij} + \sum_{\tau_{ik} \in pr(\tau_{ij})} c^{\mathrm{b}}_{ik}, \tag{6}$$

where the set $pr(\tau_{ij})$ is the set of predecessors for task τ_{ij} in the task graph model of application Λ_i.

V. Motivational Example

The examples in this section motivate the need for a proper design space exploration in order to be able to design a high expected performance embedded control system with guaranteed worst-case stability. The first example illustrates that considering only the expected control performance could jeopardize the stability of a control system. The second example illustrates that the design space exploration should be done considering the expected control cost (Equation 2) as the objective function, subject to the constraints on worst-case cost (Equation 3).

[2]Since we consider the same priority for all tasks within a task graph, the worst-case response-time analysis can be readily extended to the sensor (input) and actuator (output) tasks in a task graph, without introducing any pessimism.



(a) Example 1: two control applications (b) Example 2: three control applications

Figure 3. Motivational examples

We consider (see Equation 2) the weight matrix $Q_i = \mathrm{diag}(1, 0, 0.001)$ for each application Λ_i. All time quantities are given in units of 10 milliseconds throughout this section. We synthesize the discrete-time Linear-Quadratic-Gaussian (LQG) controllers for a given period and constant expected sensor–actuator delay using the Jitterbug toolbox [16] and MATLAB. We consider (see Equation 1) the continuous-time disturbance $\boldsymbol{v}_i$ to have a covariance $R_{1i} = 1$ and discrete-time measurement noise $\boldsymbol{e}_i$ to have the covariance $R_{2i} = 0.01$. The total expected control cost, for a set of plants $\mathbf{P}$, is $J^{\mathrm{e}}_{\mathrm{total}} = \sum_{P_i \in \mathbf{P}} J^{\mathrm{e}}_{\Lambda_i}$, whereas the total worst-case control cost is defined to be $J^{\mathrm{w}}_{\mathrm{total}} = \sum_{P_i \in \mathbf{P}} J^{\mathrm{w}}_{\Lambda_i}$.

A. Example 1: System Design Driven by Expected Control Quality

In this example, we consider two plants P_1 and P_2 ($\mathbf{P} = \{P_1, P_2\}$). The corresponding applications Λ_1 and Λ_2 are two controllers modeled as two task graphs each consisting of a sensor task τ_{is}, a computation task τ_{ic}, and an actuator task τ_{ia} as shown in Figure 3(a). The numbers in parentheses specify the execution times of the corresponding tasks. The execution-time of the control task τ_{2c} is uniformly distributed in the interval of $(0, 6)$. The other tasks, on the other hand, have constant execution times. Moreover, let us consider that both applications are released synchronously.

The applications Λ_1 and Λ_2 have periods $h_1 = h_2 = 30$. Considering the control application Λ_2 to be the higher priority application, the total expected control cost for control applications Λ_1 and Λ_2, $J^{\mathrm{e}}_{\mathrm{total}}$, computed by the Jitterbug toolbox, is minimized and is equal to 14.0. However, if we use the Jitter Margin toolbox [17], we realize that there is no guarantee that the plant P_1 will be stable (the cost $J^{\mathrm{w}}_{\Lambda_1}$ is infinite). The values of the nominal sensor–actuator delay L_1, sampling jitter Δ^{w}_{1s}, and actuator jitter Δ^{w}_{1a} are equal to 6.0.

Inverting the priority order of the applications, we can

guarantee the stability of control application Λ_1 as a result of decrease in the sampling jitter Δ^{w}_{1s} and actuator jitter Δ^{w}_{1a}, without jeopardizing the stability of control application Λ_2 (since the nominal sensor–actuator delay L_2, sampling jitter Δ^{w}_{2s}, and actuator jitter Δ^{w}_{2a} remain the same). For this priority order, the values of the nominal sensor–actuator delay L_1, sampling jitter Δ^{w}_{1s}, and actuator jitter Δ^{w}_{1a} are 6.0, 0.0, and 0.0, respectively. This solution guarantees the stability of both control applications, although the total expected control cost $J^{\text{e}}_{\text{total}}$ for control applications Λ_1 and Λ_2 is slightly larger than before, i.e., 15.2. Thus, considering only the maximization of the expected control performance during the design process can jeopardize the worst-case stability of applications. We also observe that the guaranteed worst-case stability could be obtained with only marginal decrease in the overall expected control quality.

B. Example 2: Stability-Aware System Design with Expected Control Quality Optimization

We consider three plants P_1, P_2, and P_3 ($\mathbf{P} = \{P_1, P_2, P_3\}$) to be controlled, where task graph models of the corresponding control applications Λ_1, Λ_2, and Λ_3 are shown in Figure 3(b). Let us assume that application Λ_i has a higher priority than application Λ_j iff $i > j$. Our objective is to find a stable solution with high control performance.

Considering the initial period assignment to be $h_1 = 60$, $h_2 = 60$, and $h_3 = 20$, the stability of the plant P_1 is not guaranteed (the worst-case control cost $J^{\text{w}}_{\Lambda_1}$ is not finite) even though this design solution provides high expected control performance, $J^{\text{e}}_{\text{total}} = 2.7$. For this period assignment, we obtain $L_1 = 12$, $\Delta^{\text{w}}_{1s} = 16$, $\Delta^{\text{w}}_{1a} = 22$ for application Λ_1.

In order to decrease the amount of interference experienced by low priority application Λ_1 from high priority applications Λ_2 and Λ_3, we increase the period of application Λ_3 to $h_3 = 60$. Further, since a smaller period often leads to higher control performance, we decrease the period of the control application Λ_1 to $h_1 = 30$. Applying delay and jitter analyses for this period assignment, we compute the values of the nominal sensor–actuator delay $L_1 = 9$, worst-case sensor jitter $\Delta^{\text{w}}_{1s} = 16$, and worst-case actuator jitter $\Delta^{\text{w}}_{1a} = 16$, which are smaller than the corresponding values for the initial period assignment. However, this modification does not change the nominal sensor–actuator delay, worst-case sensor jitter, and worst-case actuator jitter for the control applications Λ_2 and Λ_3. As a result, we can guarantee the stability of all applications in the worst-case scenario since the worst-case control costs for all three applications is finite. Moreover, the total worst-case and expected control costs for this period assignment are $J^{\text{w}}_{\text{total}} = 2.3$ and $J^{\text{e}}_{\text{total}} = 20.4$, respectively.[3] We observe that stability in the worst-case has been achieved with a severe degradation of the expected control performance.

Another possible solution would be $h_1 = 60$, $h_2 = 60$, and $h_3 = 60$, that leads to a total worst-case control cost equal to $J^{\text{w}}_{\text{total}} = 5.7$ and a total expected cost $J^{\text{e}}_{\text{total}} = 3.0$. It should be mentioned that the values of the nominal sensor–actuator delay, worst-case sensor jitter, and worst-case actuator jitter remain the same as the previous period assignment for all three applications. Considering the applications are released simultaneously, as for the previous period assignment, the low priority application Λ_1 experiences interference by high priority applications every other time (since it executes twice as frequently as the high priority applications). Therefore, the decrease in the expected control cost is due to removing the variation in the response-time of the control application Λ_1.

Having finite total worst-case costs for both period assignments indicates the stability of both design solutions. However, though the total worst-case control cost ($J^{\text{w}}_{\text{total}}$) is smaller in the former period assignment, the latter is the desirable design solution since the overall performance of the system is determined by the expected control performance ($J^{\text{e}}_{\text{total}}$). Although it is of critical importance to guarantee stability, it is highly desirable to keep the inherent quality degradation in the expected control performance as small as possible.

We conclude that, just optimizing the expected control quality can lead to unsafe solutions which are unstable in the worst-case. Nonetheless, focusing only on stability, potentially, leads to low overall quality. Therefore, it is essential and possible to achieve both safety (worst-case stability guarantees) and high level of expected control quality.

VI. Problem Formulation

The inputs for our co-design problem are

- a set of plants $\mathbf{P}$ to be controlled,
- a set of control applications $\mathbf{\Lambda}$,
- a set of sampling periods $\mathbf{H}_i$ for each control application Λ_i,
- execution-time probability functions of the tasks with the best-case and worst-case execution times.

The outputs of our co-design tool are the period h_i for each control application Λ_i, unique priority ρ_i for each application Λ_i, and the control law $\boldsymbol{u}_i$ for each plant $P_i \in \mathbf{P}$ (the tasks within an application have the same priority equal to the application priority). The outputs related to

[3]It is good to mention that the metrics for the worst-case (J^{w}) and expected case (J^{e}) are expressing different properties of the system and, thus, are not comparable with each other ($J^{\text{w}}_{\text{total}} = 2.3$, and $J^{\text{e}}_{\text{total}} = 20.4$, does not mean that the worst-case control cost is better than the expected one.).

the controller synthesis are the period h_i and the control law $\boldsymbol{u}_i$ for each plant P_i.

As mentioned before, there exists a control application $\Lambda_i \in \boldsymbol{\Lambda}$ corresponding to each plant $P_i \in \mathbf{P}$ and the final control quality is captured by the weighted sum of the individual control costs $J^{\mathrm{e}}_{\Lambda_i}$ (Equation 2) of all control applications $\Lambda_i \in \boldsymbol{\Lambda}$. Hence, the cost function to be minimized is

$$\sum_{P_i \in \mathbf{P}} w_{\Lambda_i} J^{\mathrm{e}}_{\Lambda_i}, \tag{7}$$

where the weights w_{Λ_i} are determined by the designer.

To guarantee stability of control application Λ_i, the worst-case control cost $J^{\mathrm{w}}_{\Lambda_i}$ (Equation 3) must have a finite value. However, in addition to stability, the designer may require an application to satisfy a certain degree of robustness which is captured by the following criteria,

$$J^{\mathrm{w}}_{\Lambda_i} < \bar{J}^{\mathrm{w}}_{\Lambda_i}, \quad \forall P_i \in \mathbf{P}, \tag{8}$$

where $\bar{J}^{\mathrm{w}}_{\Lambda_i}$ is the limit on tolerable worst-case cost for control application Λ_i and is decided by the designer. If the requirement is for an application Λ_i only to be stable in the worst-case, the constraint on the worst-case cost $J^{\mathrm{w}}_{\Lambda_i}$ is to be finite.

VII. Co-design Approach

The overall flow of our proposed optimization approach is illustrated in Figure 4. In each iteration, each application is assigned a period by our period optimization method (see Section VII-A). Having assigned a period to each application, we proceed with determining the priorities of the applications (see Section VII-B). This process takes place in two main steps. In the first step, we analyze the worst-case sensitivity (worst-case sensitivity is illustrated in Section VII-B1) of control applications and at the same time we synthesize the controllers (see Section VII-B2). The analysis of worst-case sensitivity is done based on the worst-case control performance constraints for control applications. The higher the sensitivity level of a control application, the smaller the amount of jitter and delay it can tolerate before hitting the worst-case control performance limit. Therefore, in this step, the applications are clustered in several groups according to their sensitivity and the priority level of a group is identified by the sensitivity level of its applications. In addition, in the first step, we can figure out if there exists, at all, any possible priority assignment which satisfies the worst-case robustness requirements with the current period assignment. The outputs of this step are a sequence of groups, in ascending order of sensitivity, and the synthesized controllers. While the first step has grouped applications according to their sensitivity, the second step is concerned with assigning priorities to each individual application (see Section VII-B3). This will be taken care of by an inside group optimization, where

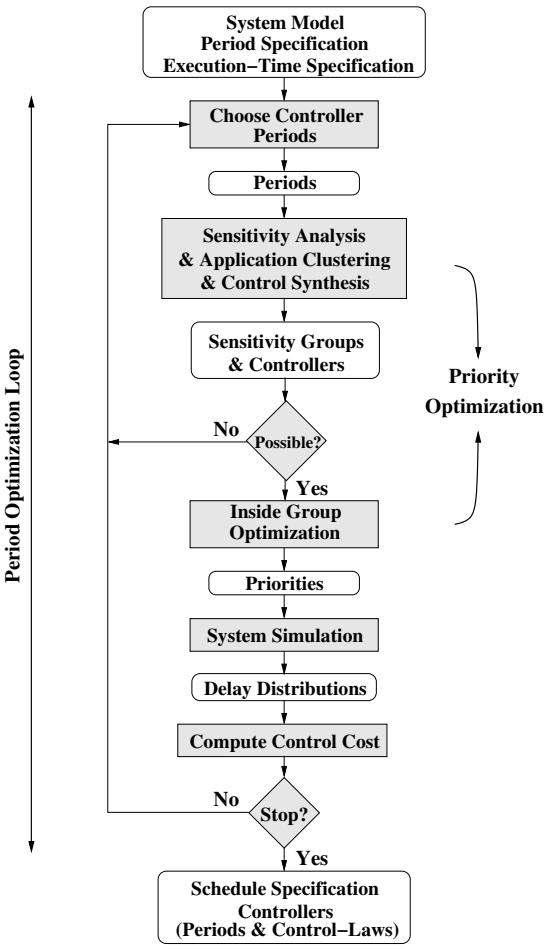


Figure 4. Overall flow of our approach

we take the expected control performance into account during the priority optimization step. Having found the periods and priorities, we perform system simulation to obtain the delay distributions for all sensor and actuator tasks. These delay distributions are then used by the Jitterbug toolbox to compute the total expected control cost. The optimization will be terminated once a satisfactory design solution is found.

A. Period Optimization

The solution space exploration for the periods of control applications is done using a modified coordinate search [20] combined with the direct search method [21]. Both methods belong to the class of derivative-free optimization methods, where the derivative of the objective function is not available or it is time consuming to obtain. These methods are desirable for our optimization since the objective function is the result of an inside optimization loop (i.e., the objective function is not available explicitly) and it is time consuming to approximate the gradient of the objective function using finite differences [20]. We describe our period optimization approach in two steps, where the first step identifies a promising

region in the search space and the second step performs the search in the region identified by the first step, to achieve further improvements.

In the first step, we utilize a modified coordinate search method to acquire information regarding the search space and to drive the optimization towards a promising region. The coordinate search method cycles through the coordinate directions (here, the coordinates are along the application periods) and in each iteration performs a search along only one coordinate direction. Our modified coordinate search method is guided by the expected control performance, taking into consideration the worst-case requirements of the applications. In the initial step, all control applications are assigned their longest periods in their period sets. If there exists a control application violating its worst-case control performance criterion, its period is decreased to the next longest period in the period set; otherwise, the period of the control application with the highest expected control cost is decreased to the next longest period in the period set. This process is repeated until the expected control performance cannot be further improved under the worst-case control performance requirements of the applications.

The solution obtained in the first step (period assignment), places us in a promising region of the solution space. In the second step, the direct search method is utilized to achieve further improvements. This method employs two types of search moves, namely, exploratory and pattern. The exploratory search move is used to acquire knowledge concerning the behavior of the objective function. The pattern move, however, is designed to utilize the information acquired by the exploratory search move to find a promising search direction. If the pattern move is successful, the direct search method attempts to perform further steps along the pattern move direction with a series of exploratory moves from the new point, followed by a pattern move.

Figure 5 illustrates the coordinate and direct search methods using a simple example considering two control applications. Therefore, in this example, the search space is two-dimensional, where each dimension corresponds to the period of one of the control applications. The dashed curves are contours of the objective function and the minimizer of the objective function in this example is $x^* = (h_1^*, h_2^*)$. The green arrows are the moves performed by the coordinate search method in the first step. The blue and red arrows are exploratory and pattern moves, respectively, performed by the direct search method in the second step.

In this section, we have considered the period optimization loop in Figure 4. Inside the loop, application priorities have to be determined and controllers have to be synthesized such that the design goals are achieved.

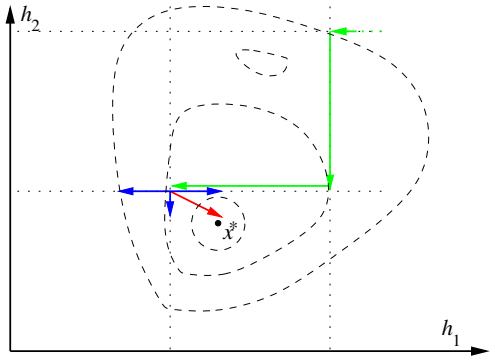


Figure 5. The coordinate and direct search methods

This will be described in the following section.

B. Priority Optimization and Control Synthesis

Our priority optimization approach consists of two main steps. The first step makes sure that the worst-case control performance constraints are satisfied, whereas the second step improves the expected control performance of the control applications.

1) Worst-Case Sensitivity and Sensitivity Groups

The notion of worst-case sensitivity is closely related to the amount of delay and jitter a control application can tolerate before violating the robustness requirements. The sensitivity level of an application is captured by Algorithm 1. Further, Algorithm 1 clarifies the relation between sensitivity level and priority level of a group and identifies the sensitivity groups ($\mathbf{G}_i$). The notion of sensitivity is linked to priority by the fact that high priority applications experience less delay and jitter.

2) Sensitivity-Based Application Clustering

The algorithm for the worst-case sensitivity analysis and application clustering is outlined in Algorithm 1. The main idea behind this algorithm is to cluster the applications which have the same level of sensitivity in the same group.

To cluster applications, we look for the set of applications which can satisfy their worst-case control performance requirements even if they are at the lowest priority level. This group ($\mathbf{G}_1$) of applications can be considered the least sensitive set of applications. We remove these applications from the set of applications (Line 24) and proceed with performing the same procedure for the remaining applications. This process continues until either the set of remaining applications ($\mathbf{S}$) is empty (Line 17) or none of the remaining applications can satisfy its requirements if it is assigned the lowest priority among the remaining applications (Line 20), which indicates that there does not exist any priority assignment for the

Algorithm 1 Worst-Case Sensitivity Analysis

```
1:  % S: remaining applications set;
2:  % G_i: the i-th sensitivity group;
3:  Initialize set S = Λ;
4:  for n = 1 to | Λ | do
5:     G_n = ∅;
6:     for all Λ_i ∈ S do
7:        • Response-time analysis for sensor and actuator (Eq 5
          and 6), considering hp(Λ_i) = S \ {Λ_i};
8:        • Jitter and delay analyses Δ^w_is, Δ^w_ia, L_i (Eq 4);
9:        • Simulation considering hp(Λ_i) = S \ {Λ_i} to find the
          expected sensor–actuator delay for Λ_i;
10:       • Control-law synthesis and delay compensation;
11:       • Worst-case control performance J^w_Λi analysis (Eq 3);
12:       if J^w_Λi < J̄^w_Λi then
13:          G_n = G_n ∪ {Λ_i};
14:       end if
15:    end for
16:
17:    if S == ∅ then
18:       % Terminate!
19:       return ⟨G_1, G_2, ..., G_n⟩;
20:    else if G_n == ∅ then
21:       % No possible solution meets the requirements!
22:       return ⟨⟩;
23:    else
24:       S = S \ G_n;
25:    end if
26: end for
```

current assigned periods which can guarantee the worst-case control performance requirements for all applications.

In order to figure out whether an application meets its worst-case robustness requirements, we perform the best-case and worst-case response-time analyses (Equations 5 and 6) for the sensor and actuator tasks considering $hp(\Lambda_i) = \mathbf{S} \setminus \{\Lambda_i\}$ (Line 7) and compute the nominal sensor–actuator delay, worst-case sensor jitter, and worst-case actuator jitter (Equation 4) for the application under analysis (Line 8). Moreover, we shall synthesize the control-law and compensate for the proper amount of delay. Therefore, we design an LQG controller and compensate for the expected sensor–actuator delay (Line 10). The expected sensor–actuator delay is extracted from the schedule in our system simulation environment, where the remaining applications constitute the set of higher priority applications ($hp(\Lambda_i) = \mathbf{S} \setminus \{\Lambda_i\}$) for the application under analysis (Line 9). Having the controller and the values of the nominal sensor–actuator delay, worst-case sensor jitter, and worst-case actuator jitter, we can calculate the worst-case control cost (Equation 3) (Line 11) and check if the worst-case requirements (Equation 8) are fulfilled (Line 12).

3) Inside Group Optimization

Based on Algorithm 1 in the previous section, we have grouped the applications according to their sensitivity level. In this section, we shall assign a unique priority to each application such that the expected control performance is improved. *The grouping realized by Algorithm 1 guarantees the worst-case control performance and robustness requirements (Equation 8) of an application Λ_i assigned to group $\mathbf{G}_j$, as long as all applications assigned to a group $\mathbf{G}_k$, $k < j$, have a priority smaller than the priority of application Λ_i.* Considering the grouping, we can make the following observation: priority order of applications inside a group can be assigned arbitrarily without jeopardizing the worst-case performance and stability guarantees.

As mentioned before, an intrinsic property of Algorithm 1 is that the priority order of the applications inside a group can be changed arbitrarily. Therefore, our proposed approach optimizes the priority order of the applications within each group with respect to the expected control performance without the need for rechecking the worst-case requirements of applications. Thus, the combinatorial optimization problem is divided into several smaller problems, inside each group, which are exponentially less time consuming to solve. In this paper, the priorities are assigned considering the dynamics of the plants, A_i, and the sampling periods, h_i. Therefore, we choose to assign priorities based on the values of $\hat{\lambda}_i h_i$, where $\hat{\lambda}_i = \max_j\{\mathrm{Re}(\lambda_{ij}) | \det(A_i - \lambda_{ij} I) = 0\}$ is a proper representative of dynamics of plant P_i (λ_{ij} are the eigenvalues of matrix A_i). We consider that the larger the value of $\hat{\lambda}_i h_i$ is, the higher the priority of application Λ_i, leading to smaller induced delays due to interference from other applications.

VIII. Experimental Results

We have performed several experiments to investigate the efficiency of our proposed design approach. We compare our approach (EXP–WST) against three other approaches (NO_OPT, WST, and EXP). We have 125 benchmarks with varying number of plants. The number of plants varies from 2 to 15. The plants are taken from a database with inverted pendulums, ball and beam processes, DC servos, and harmonic oscillators [15]. Such benchmarks are representative of realistic control problems and are used extensively for experimental evaluations. For each plant, there exists a corresponding control application modeled as a task graph with 2 to 5 tasks. The set of suggested periods of each control applications includes 6 periods generated based on common rules of thumb [15]. Without loss of generality, we wish to find a high-quality stable design solution, i.e., the constraint on the maximum tolerable worst-case control cost is that the cost is finite (Equation 8).

A. Efficiency of our proposed approach

As for the first comparison, we run the same algorithm as our proposed approach, however, it terminates as soon as it finds a stable design solution. Therefore, this approach, called NO_OPT, does not involve any expected

Table I
EXPERIMENTAL RESULTS

Number of control applications	NO_OPT approach	WST approach	EXP approach	
	Improvement $\left(\frac{J^e_{\text{NO\_OPT}} - J^e_{\text{EXP–WST}}}{J^e_{\text{NO\_OPT}}}\right) \times 100$	Improvement $\left(\frac{J^e_{\text{WST}} - J^e_{\text{EXP–WST}}}{J^e_{\text{WST}}}\right) \times 100$	Difference $\left(\frac{J^e_{\text{EXP}} - J^e_{\text{EXP–WST}}}{J^e_{\text{EXP}}}\right) \times 100$	Percentage of invalid solutions [4]
2	74%	10%	-2.4%	42%
3	72%	49%	4.6%	20%
4	44%	25%	-5.4%	44%
5	55%	37%	-0.4%	40%
6	52%	27%	-6.1%	30%
7	58%	33%	4.4%	33%
8	56%	23%	-4.1%	44%
9	54%	28%	-4.9%	44%
10	46%	20%	-4.9%	37%
11	51%	24%	-5.5%	50%
12	43%	13%	-1.6%	44%
13	65%	28%	1.9%	71%
14	46%	24%	-2.4%	60%
15	38%	18%	-5.5%	60%
Average	53%	26%	-2.3%	44%

control performance optimization but guarantees worst-case stability. We calculate the relative expected control cost improvements $\frac{J^e_{\text{NO\_OPT}} - J^e_{\text{EXP–WST}}}{J^e_{\text{NO\_OPT}}}$, where $J^e_{\text{EXP–WST}}$ and $J^e_{\text{NO\_OPT}}$ are the expected control costs produced by our approach and the NO_OPT approach, respectively. The second column of Table I shows the result of this set of experiments. As it can be observed, our approach produces solutions with guaranteed stability and an overall control quality improvement of 53% on average, compared to an approach which only addresses worst-case stability.

The second comparison is made with an optimization approach driven by the worst-case control performance. The approach, called WST, is exactly the same as our approach but the objective function to be optimized is the worst-case control cost given in Equation 3. Similar to the previous experiment, we are interested in the relative expected control cost improvements $\frac{J^e_{\text{WST}} - J^e_{\text{EXP–WST}}}{J^e_{\text{WST}}}$, where J^e_{WST} is the expected control cost of the final solution obtained by the WST approach. Our proposed approach, while still guarantees worst-case stability, has an average improvement of 26% in the expected control cost as shown in the third column of Table I.

The third comparison is performed against an optimization approach, called EXP, which ONLY takes into consideration the expected control performance. The priority assignment in this approach is done by a genetic algorithm approach similar to our previous work [13]. Since the worst-case control performance constraints are ignored, the search space is larger, and the algorithm should be able to find a superior design solution in terms of the expected control performance. The comparison has been made considering the relative expected control cost difference $\frac{J^e_{\text{EXP}} - J^e_{\text{EXP–WST}}}{J^e_{\text{EXP}}}$, where J^e_{EXP} is the expected control cost of the final solution found by the EXP approach. The results of the comparison are shown in the forth column of Table I. Since the worst-case control performance constraints are relaxed, the final solution of this approach can turn out to be unstable. The percentage of designs produced by the EXP approach for which the worst-case stability was not guaranteed is reported in the fifth column of Table I. The first observation is that, on average, for 44% of the benchmarks this algorithm ended up with a design solution for which the stability could not be guaranteed. The second observation is that our approach is on average 2.3% away from the relaxed optimization approach exclusively guided by expected control performance. This clearly states that we are able to guarantee worst-case stability with a very small loss on expected control quality. As discussed before, the relaxed optimization approach should in principle outperform our proposed approach. However, our proposed approach performs slightly better in a few cases which is due to the fact that heuristics with different constraints can be guided to different regions of the huge search space.

B. Runtime of our proposed approach

We measured the runtime of our proposed approach on a PC with a quad-core CPU running at 2.83 GHz with 8 GB of RAM and Linux operating system. The average runtime of our approach based on the number of control application is shown in Figure 6. It can be seen that our approach could find a high-quality stable design solution for large systems (15 control applications) in less than 7 minutes. Also, we report the timing of the relaxed optimization approach, EXP, which only considers the expected control performance (no guarantee for stability), where the priority assignment is performed by a genetic algorithm similar to [13]. For large systems (15 control applications), it takes 178 minutes for this approach to terminate.

[4]Percentage of design solutions produced by the EXP approach for which stability is not guaranteed.

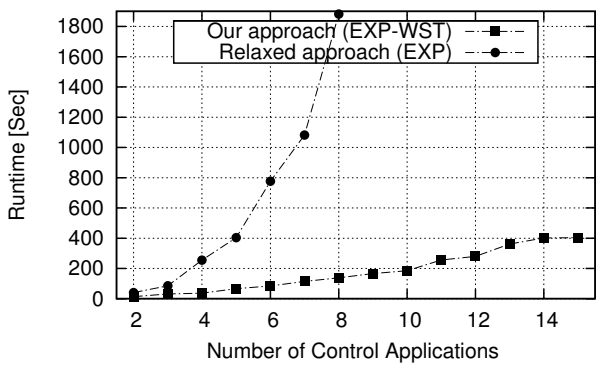


Figure 6. Runtime of our approach (EXP–WST) and comparison with the relaxed optimization approach (EXP)

We conclude that the design optimization time is reduced by an order of magnitude compared to previous co-design methods for embedded control systems. In addition, our integrated design approach gives formal guarantees on stability based on timing and jitter analysis, and at the same time it practically achieves the same level of control-quality optimization as related design approaches.

IX. Conclusions

The design of embedded control systems requires special attention due to the complex interrelated nature of timing properties connecting scheduling and control synthesis. In order to address this problem, not only the control performance but also the robustness requirements of control applications have to be taken into consideration during the design process since having high control performance is not a guarantee for worst-case stability. On the other hand, a design methodology solely grounded on worst-case scenarios leads to poor control performance due to the fact that the design is tuned to a scenario which occurs very rarely. We proposed an integrated design optimization method for embedded control systems with high requirements on robustness and optimized expected control performance. Experimental results have validated that both control quality and worst-case stability of a system have to be taken into account during the design process.

References

[1] B. Wittenmark, J. Nilsson, and M. Törngren, "Timing problems in real-time control systems," in *Proceedings of the American Control Conference*, 1995, pp. 2000–2004.

[2] K. E. Årzén, A. Cervin, J. Eker, and L. Sha, "An introduction to control and scheduling co-design," in *Proceedings of the* 39^{th} *IEEE Conference on Decision and Control*, 2000, pp. 4865–4870.

[3] H. Rehbinder and M. Sanfridson, "Integration of off-line scheduling and optimal control," in *Proceedings of the* 12^{th} *Euromicro Conference on Real-Time Systems*, 2000, pp. 137–143.

[4] H. Kopetz, *Real-Time Systems—Design Principles for Distributed Embedded Applications*. Kluwer Academic, 1997.

[5] R. Majumdar, I. Saha, and M. Zamani, "Performance-aware scheduler synthesis for control systems," in *Proceedings of the* 9^{th} *ACM international conference on Embedded software*, 2011, pp. 299–308.

[6] D. Goswami, M. Lukasiewycz, R. Schneider, and S. Chakraborty, "Time-triggered implementations of mixed-criticality automotive software," in *Proceedings of the* 15^{th} *Conference for Design, Automation and Test in Europe (DATE)*, 2012.

[7] K. E. Årzén and A. Cervin, "Control and embedded computing: Survey of research directions," in *Proceedings of the* 16^{th} *IFAC World Congress*, 2005.

[8] D. Seto, J. P. Lehoczky, L. Sha, and K. G. Shin, "On task schedulability in real-time control systems," in *Proceedings of the* 17^{th} *IEEE Real-Time Systems Symposium*, 1996, pp. 13–21.

[9] E. Bini and A. Cervin, "Delay-aware period assignment in control systems," in *Proceedings of the* 29^{th} *IEEE Real-Time Systems Symposium*, 2008, pp. 291–300.

[10] M. M. Ben Gaid, A. Cela, and Y. Hamam, "Optimal integrated control and scheduling of networked control systems with communication constraints: Application to a car suspension system," *IEEE Transactions on Control Systems Technology*, vol. 14, no. 4, pp. 776–787, 2006.

[11] A. Cervin, B. Lincoln, J. Eker, K. E. Årzén, and G. Buttazzo, "The jitter margin and its application in the design of real-time control systems," in *Proceedings of the* 10^{th} *International Conference on Real-Time and Embedded Computing Systems and Applications*, 2004.

[12] F. Zhang, K. Szwaykowska, W. Wolf, and V. Mooney, "Task scheduling for control oriented requirements for cyber-physical systems," in *Proceedings of the* 29^{th} *IEEE Real-Time Systems Symposium*, 2008, pp. 47–56.

[13] S. Samii, A. Cervin, P. Eles, and Z. Peng, "Integrated scheduling and synthesis of control applications on distributed embedded systems," in *Proceedings of the Design, Automation and Test in Europe Conference*, 2009, pp. 57–62.

[14] A. Aminifar, S. Samii, P. Eles, and Z. Peng, "Control-quality driven task mapping for distributed embedded control systems," in *Proceedings of the* 17^{th} *IEEE Embedded and Real-Time Computing Systems and Applications (RTCSA) Conference*, 2011, pp. 133 –142.

[15] K. J. Åström and B. Wittenmark, *Computer-Controlled Systems*, 3rd ed. Prentice Hall, 1997.

[16] B. Lincoln and A. Cervin, "Jitterbug: A tool for analysis of real-time control performance," in *Proceedings of the* 41^{st} *IEEE Conference on Decision and Control*, 2002, pp. 1319–1324.

[17] A. Cervin, "Stability and worst-case performance analysis of sampled-data control systems with input and output jitter," in *Proceedings of the 2012 American Control Conference (ACC)*, 2012.

[18] M. Joseph and P. Pandya, "Finding response times in a real-time system," *The Computer Journal*, vol. 29, no. 5, pp. 390–395, 1986.

[19] J. Palencia Gutierrez, J. Gutierrez Garcia, and M. Gonzalez Harbour, "Best-case analysis for improving the worst-case schedulability test for distributed hard real-time systems," in *Proceedings of the* 10^{th} *Euromicro Workshop on Real-Time Systems*, 1998, pp. 35–44.

[20] J. Nocedal and S. Wright, *Numerical Optimization*, 2nd ed. Springer, 1999.

[21] R. Hooke and T. A. Jeeves, ""direct search" solution of numerical and statistical problems," *J. ACM*, vol. 8, no. 2, pp. 212–229, 1961.