Optimal mean square control using the continuous stream model of computation

Daniele Fontanelli, Luca Greco, Luigi Palopoli

Optimal Mean Square Control using the Continuous Stream Model of Computation

Daniele Fontanelli[1] and Luca Greco[2] and Luigi Palopoli[3]

***Abstract*— We consider the problem of optimal allocation of computation resources to a set of control tasks sharing a CPU. Each task is used to control a linear and time invariant plant affected by noise and is supposed to have a computation time with known distribution. The metric we use to express the Quality of Control (QoC) is a function of the steady state covariance of the state. We show how the combination of a Resource Reservation scheduler with an unconventional model of computation known as Continuous Stream produces a system that is easy to analyse. In particular, it is possible to compute the QoC of each control loop as a function of the fraction of CPU (bandwidth) that the task receives, and formalise an optimisation problem where a global QoC metric is defined that consolidates the QoC of each task. We show an efficient solution to this optimisation problem and validate its efficacy on a set of numeric examples.**

I. Introduction

Modern embedded controllers are increasingly based on sensors that require a heavy and time-varying processing time to extract the relevant information. Vision based application are a perfect example of this kind. The processing time for these applications can be modeled as stochastic variables with long tailed distribution. When the hardware is shared between several applications (a frequent design choice to reduce the costs and to simplify system engineering) additional delays are introduced by the scheduling interference.

The standard solution to deal with scheduling delays is to force the communications between the plant and the embedded controller to take place at precise points in time [1] and to delegate to the system engineer the responsibility of guaranteeing that all applications will be ready to deliver their output at the correct time. When the processing time changes much, providing temporal guarantees for the execution of every single activation (job) of a task significantly reduces the possibility of sharing the CPU.

Many researchers have investigated on *how to make the design robust* against an irregular timing behaviour of the implementation, focusing on such effects as packet dropout [2], [3], jitter in computation [4], [5] and time varying delays [6]. Other authors have sought suitable ways to modify the scheduling behaviour in overload conditions [7], [8].

In this paper we consider a design problem where multiple and independent linear controllers are implemented by a set of real–time tasks with stochastic computation time sharing a CPU. Roughly speaking the goal is to identify the choice of the scheduling parameter that corresponds to a global optimal performance. Our work is inspired by the following considerations: I) control systems are known to tolerate delays and even occasional losses of data in the feedback loop, as long as the problem is adequately accounted for in system design [9], [10], II) when a task has a variable computation time, standard scheduler using fixed or dynamic priorities [11] are difficult to use, whilst different alternatives such as the Resource Reservations (RR) [12], [13] offer a superior performance, III) if a designer's objective is to enforce system stability with minimal use of resources, unconventional models of computation such as event-triggered [14], [15] or self-triggered [16], [17] control and anytime computing [18] can offer a valuable alternative to periodic sampling.

RR scheduling algorithms have been developed in the multimedia community and enable a fine–grained control of the bandwidth (fraction of CPU time) that each task receives. By using this property, we can derive a discrete–time Markov Chain (DTMC) describing the evolution of the delays incurred by a task for a given distribution of its computation time and for a given assigned bandwidth [19]. In our previous work, we have combined the DTMC associated with the Resource Reservation scheduler with a soft real–time computation model, whereby a control task is activated periodically but is occasionally allowed to execute beyond its deadline. The Markov Jump Linear System (MJLS) that describes the resulting closed loop dynamics is rather cumbersome to analyse [10]. A different possibility [20] is to adopt a simpler model of computation, in which sampling is periodic but a delayed job is cancelled. In this case the use of RR scheduling enables a straightforward computation of the probability of dropping a job as a function of the bandwidth. A standard test to ascertain the stability of a MJLS is to compute the steady state covariance of the state and to verify that it satisfies some eigenvalue conditions (see for instance [21]). Moreover, in case of random input noise, the steady state covariance is usually adopted as a performance measure, since it accounts for the impact of the noise on the output. As a consequence, the steady state covariance can be used to define a Quality of Control (QoC) metrics for the task. The case of multiple control tasks sharing the CPU

This paper has received funding from the European Unions Horizon 2020 Research and Innovation Programme - Societal Challenge 1 (DG CONNECT/H) under grant agreement n° 643644 "ACANTO - A CyberphysicAl social NeTwOrk using robot friends".

D. Fontanelli is with the Department of Industrial Engineering (DII), University of Trento, Via Sommarive 5, Trento, Italy daniele.fontanelli@unitn.it. L. Greco is with the Université Paris Sud, L2S - Supélec - 3, rue Joliot-Curie - 91192 Gif sur Yvette, France luca.greco@lss.supelec.fr. L. Palopoli is with the Department of Information Engineering and Computer Science (DISI), University of Trento, Via Sommarive 5, Trento, Italy palopoli@disi.unitn.it

can be addressed by consolidating the QoC of the different control loops into a global QoC metric. By doing so, the designer translates the design problem into *an optimisation problem*, where the bandwidth assigned to tasks are decision variables that can be efficiently computed. Although simple to analyse, the computation model [20] is based on the outright cancellation of a job and can be a drastic choice, with a negative impact on the QoC of the delivered by the task.

In this paper, we aim to combine the simplicity of the problem formulation with job cancellations with the arguably better performance of a model in which delays are allowed. The key point is the adoption of the so-called Continuous Stream model of computation [22], in which a delayed task is allowed to continue beyond the deadline but the new sampling event is deferred to the termination of the job. This idea allows us to model the delays as an independent and identically distribute (IID) process and to extend the formalisation of the optimisation problem for the job cancellation case proposed in [20] to a more compelling problem. In particular, we can use the steady state covariance of each control loop to define a QoC metric, expose its dependency on the probability of the different delays (and ultimately on the assigned bandwidth), and use a simple technique to compute the optimal value of the global QoC. In the paper, we show the theoretical foundations of this idea considering generic non-monotone QoC measures (only monotone functions are adopted in [20]), and test its efficacy on several numeric examples.

The paper is organised as follows. In Section II, we offer some background information and formulate the design problem. In Section III, we show how to set up constraints and cost function for the optimisation problem. In Section IV we show our solution strategy for the problem. In Section V, we offer numeric evidence of the efficiency of our solution. Finally, in Section VI, we offer our conclusions and announce future work directions.

II. Background Material and Problem Presentation

We consider a set of real–time tasks sharing a CPU. In our setting a task $\tau_i, i = 1, \ldots, n$ is a piece of software implementing a controller. Tasks have a cyclic behaviour: when an event occurs at the start time $r_{i,j}$, task τ_i generates a job $J_{i,j}$ (the j-th job). In this work, we will consider periodic activations: $r_{i,j} = jT_i$ with $j \in \mathbb{Z}_{\geq 0}$. Job $J_{i,j}$ finishes at the finishing time $f_{i,j}$, after using the resource for a computation time $c_{i,j}$. If job $J_{i,j}$ is granted an exclusive use of the resource, then $f_{i,j} = r_{i,j} + c_{i,j}$.

A. Scheduling Mechanism

The scheduling mechanism rules the access to the computing power when multiple job requests are active at the same time. In this paper, we advocate the use of *resource reservations* [12]. In any time interval of length R_i, called the *reservation period* henceforth, the fraction of CPU time granted by the scheduler to the task τ_i during R_i is called the *bandwidth* B_i. The product $Q_i = B_i R_i$ is termed *budget*, which is an integer number ranging in the interval $[0, \ldots, R_i]$. A *reservation* is then (Q_i, R_i). Since the execution of job $J_{i,j}$ requires at most $\left\lceil \frac{c_{i,j}}{Q_i} \right\rceil R_i$ computation units, or briefly $\frac{c_{i,j}}{B_i}$, we can consider the bandwidth B_i as a fraction of the speed of the real processor that is given to task τ_i.

For our purposes, it is convenient to choose a reservation period that is an integer sub-multiple of the task period: $T_i = N_i R_i,\ N_i \in \mathbb{N}$. There are several possible implementation of the resource reservations paradigm, one of the most popular being the Constant Bandwidth Server (CBS) [13], which operates properly as long as the following condition is satisfied.

$$\sum_{i=1}^{n} B_i \leq 1, \tag{1}$$

i.e., the cumulative allocation of bandwidth cannot exceed 100% of the computing power.

B. Model of Computation

The CBS has been already used in the literature assuming a time-triggered model of execution [1]. For example, [10] proposes a feedback scheduling for a set of control tasks, while [20] presents an optimal allocation of bandwidth for a model of computation where delayed jobs are cancelled. The approach followed in this paper is similar in spirit to the latter approach, but extends the result to a more flexible model of computation, called the *Continuous Stream* [22], which we summarise below.

The continuous stream MoC is an "extreme" evolution of the soft real–time model of computation described in [10]. The model can be described as follows: 1. when the j-th job is activated on a sampling event (time $r_{i,j}$), it is assigned a relative deadline to complete (say d), which is a multiple of a base time granularity R_j typically chosen equal to the reservation period, 2. if the job finishes before d, at time $r_j + d$ its output is released a new input is acquired and a new job triggered, 3. in the opposite case the job's execution continues beyond the $r_{i,j} + d$. When the job finishes (time $f_{i,j}$, the system waits for the next release time, equal to $r_{i,j} + \left\lceil \frac{f_{i,j} - r_{i,j}}{R_j} \right\rceil R_j$. When this instant arrives, the result of the computation of job $J_{i,j}$ is released, a new sample is acquired and the new job starts. 4) there is a maximum amount of delay beyond which the job is cancelled.

As apparent from this short description, the idea of time-triggered activation is relaxed: when a job finishes after its soft absolute deadline (equal to the job arrival time plus the nominal period T), it immediately activates the following one (hence the name "continuous stream"). The delivery of the task result takes place at the arrival time of the next job. Since the input for a job is collected at the same time the result of the previous job is released, the model tolerates the delays and, more importantly, the absence of carry-on execution between adjacent jobs prevents an accumulation of delays. This drastically simplifies the model expressing

the evolution of the delays, which become an i.i.d. process. The interested reader is referred to [22].

C. The Control Problem

Each task τ_i is used to control a controllable and observable discrete–time linear system:

$$\begin{aligned} x_i(k+1) &= A_i x_i(k) + F_i u_i(k) + w_i(k) \\ y_i(k) &= C_i x_i(k) \end{aligned} \tag{2}$$

where $x_i(k) \in \mathbb{R}^{n_{x_i}}$ represents the system state, $u_i(k) \in \mathbb{R}^{m_i}$ the control inputs, $w_i(k) \in \mathbb{R}^{n_{x_i}}$ a noise term and $y_i \in \mathbb{R}^{p_i}$ the output functions. One step transition for this discrete time system refers to the evolution of the system step across one temporal unit, which is set equal to the reservation period R_i. The $u_i(k)$ vector is updated at the end of each control task, according to the model of execution described above.

By $P_i(k) = \mathsf{E}\left\{x_i(k)x_i(k)^T\right\}$ we denote the variance of the state resulting from the action of the noise term and of the control action. When $\overline{P}_i = \lim_{k\to\infty} P_i(k) < +\infty$ (i.e., the variance converges) for a given control algorithm, we will say that the closed loop system is *mean square stable* [21] and we will use $\overline{P}_i$ to define Quality of Control (QoC) metrics. Clearly, the smaller is the value of $\overline{P}_i$, the better is the control quality.

D. Problem Formulation

The bandwidth B_i quantifies the amount of computation that the task receives in unit time and can be translated into the delays introduced in the feedback loop and therefore into the probability of violating the deadline. Then, different values for the bandwidth determine a different value of the QoC $\overline{P}_i$.

The objective of this paper is to identify the allocation of resources, i.e., the bandwiths $B_1, \ldots, B_n$, between the different n control tasks that maximises the system-wide *global QoC*. More precisely:

$$\begin{aligned} &\min \Phi(\overline{P}_1, \ldots, \overline{P}_n) \\ &\qquad \text{subject to} \\ &\sum_{i=1}^{n} B_i \leq 1 \\ &\overline{B}_i \geq B_i \geq \underline{B}_i. \end{aligned} \tag{3}$$

where $\underline{B}_i$ is the minimum (critical) bandwidth that the task has to receive in order for the feedback loop that it implements to be mean square stable, while the upper bound $\overline{B}_i$ is to ensure that the task does not receive more bandwidth than it needs to achieve probability 1 of meeting the deadline. The cost function $\Phi(\cdot)$ generate the global QoC index, and it is considered to be the infinity norm in this paper, i.e., $\Phi(\overline{P}_1, \ldots, \overline{P}_n) = \max_i \phi_i(\overline{P}_i)$, where $\phi_i(\cdot)$ are the individual QoC metrics. Finally, the constraint $\sum_{i=1}^{n} B_i \leq 1$ is the schedulability condition (1).

III. Cost function and constraints for the optimisation problems

This section presents the QoC analysis for a generic task. Since we are referring to a single control task, we will drop the i subscript in this section. The jobs control tasks are *nominally* activated on a periodic basis: $r_j = jNR$, where the $N \in \mathbb{N}$ expresses the number of steps (reservation periods) composing a period. Since the control action is effectively computed when the job is executed, the system (2) is controlled by the linear controller

$$\begin{aligned} z_{j+1} &= A_c z_j + B_c y_j \\ u_j &= C_c z_j + H_c y_j, \end{aligned} \tag{4}$$

where j is the index of the j-th job and we have adopted for notation simplicity $z(j+1) = z_{j+1}$. The controller is designed supposing that the control law is computed with a fixed nominal period $T = NR$, while the output of the control task is released with a delay of one period T after the fetch of the output y_j (*unitary delay* assumption). More in depth, by recalling that $D_j = \left\lceil \frac{c_j}{Q} \right\rceil$ is the number of reservation periods R needed for the task computation (see Section II-A), the input-output delay will be at least of N. Because of the computation time and of the presence of other tasks, the output may be further deferred after the relative deadline. In this case, the control task incurs an additional delay, defined by $\Delta_j = D_j - N$ reservation periods. A maximum delay of N_r reservation periods is tolerated, for some fixed integer $N_r > N$. Hence, the integer delay D_j can change for each job and assumes values in the set $\mathcal{D} = \{N, \ldots, N_r\}$.

A. Closed Loop Model

The integer variable $D_j \in \mathcal{D}$ represents the number of reservation periods during which the control value u_{j-1} is held constant, hence the controlled system dynamics in a job-based fashion instead of a time-based one as in (2):

$$\begin{aligned} x_{j+1} &= A^{D_j} x_j + F_{D_j} u_{j-1} + \widetilde{w}(j) \\ y_j &= C x_j, \end{aligned} \tag{5}$$

where $F_{D_j} = \sum_{t=0}^{D_j-1} A^{D_j-t-1} F$ and $\widetilde{w}(j) = \sum_{t=0}^{D_j-1} A^{D_j-t-1} w(j+t)$. Here we are assuming that the output of the system y_j is sampled on a job basis.

If, instead, the delay would be greater then N_r, i.e., $D_j \notin \mathcal{D}$, a job cancellation event takes place (called *drop*), forcing the computation to stop. The cancellation event can be managed either holding the previous control value (drop and hold), or zeroing it (drop and zero). In both cases we consider the controller state z_j to be held ($z_j = z_{j-1}$). Therefore, in the drop case, the dynamics (5) is still unchanged, while the dynamics of the controller (4) has to be modified. To this end, we introduce the state variable ξ_j and re-write the controller dynamics (4) as reported in Table I.

$$\begin{aligned} \zeta_{j+1} &= A_{\phi(j)} \zeta_j + v(j) \\ y_j &= \tilde{C} \zeta_j, \end{aligned} \tag{6}$$

Regular dynamics	Drop and hold	Drop and zero
$z_{j+1} = A_c z_j + B_c y_j$	$z_{j+1} = z_j$	$z_{j+1} = z_j$
$\xi_{j+1} = C_c z_j + H_c y_j$	$\xi_{j+1} = \xi_j$	$\xi_{j+1} = 0$
$u_j = C_c z_j + H_c y_j$	$u_j = \xi_j$	$u_j = 0.$

TABLE I

CONTROLLER DYNAMICS FOR REGULAR AND DROP SITUATIONS.

Regular dynamics $\phi(j) \in \mathcal{D}$	Drop $\phi(j) > N_r$
$\begin{bmatrix} A^{\phi(j)} & 0 & F_{\phi(j)} \\ B_c C & A_c & 0 \\ H_c C & C_c & 0 \end{bmatrix}$	$\begin{bmatrix} A^{\phi(j)} & 0 & F_{\phi(j)} \\ 0 & I & 0 \\ 0 & 0 & \alpha I \end{bmatrix}$

TABLE II

CLOSED LOOP DYNAMICS FOR REGULAR AND DROP SITUATIONS. THE DROP DYNAMIC IS DROP AND HOLD WHEN $\alpha = 1$ AND DROP AND ZERO WHEN $\alpha = 0$.

where $\zeta_j \triangleq [x_j^T, z_j^T, \xi_j^T]^T$, $\tilde{C} \triangleq [C, 0, 0]$, $v(j) \triangleq \left[\widetilde{w}(j)^T, 0\right]^T$ and the piecewise constant function $\phi : \mathbb{Z}_{\geq 0} \to \mathcal{D} \cup \{N_r + 1\}$ rules the switchings among the different subsystems, whit $\phi(j) = N_r + 1$ for a drop event. The structure of $A_{\phi(j)}$ can be easily derived recalling that u_{j-1} in (5) is replaced by ξ_j, thus leading to the matrices in Table II.

In the following, we assume that the noise $w(\cdot)$ and the computation time $\{c_j\}_{j \in \mathbb{Z}_{\geq 0}}$ are mutually independent, stationary and both independent from the state. Moreover, we assume that $w(\cdot)$ is an independent identically distributed (i.i.d.) random process, have zero mean and variance $\mathrm{E}\left\{w(k)w(k)^T\right\} = W$. We further assume that also $\{c_j\}_{j \in \mathbb{Z}_{\geq 0}}$ is i.i.d., which is a realistic assumption since even ignoring its possible correlation structure, a close approximation of the actual process can be obtained [19].

B. Stochastic Analysis

Depending on the choice of the bandwidth B, we get different distributions for the i.i.d. process representing the delay. Let us define the probability μ_t^j of the j-th job to execute *exactly* in t reservation periods as $\mu_t^j \triangleq \mathsf{P}\{D_j = t\}$ for $t = N, \ldots, N_r$. By means of the fluid approximation and the definition of D_j, we have that

$$\mu_t^j \triangleq \begin{cases} \mathsf{P}\{c_j \leq NBR\} & \text{for } t = N \\ \mathsf{P}\{(t-1)BR < c_j \leq tBR\} & \text{for } t = N+1, \ldots, N_r, \end{cases} \tag{7}$$

which expresses the probability in terms of the computation time c_j and of the bandwidth B. Due to the fact that $\{c_j\}_{j \in \mathbb{Z}_{\geq 0}}$ is an i.i.d. process, the probability does not change with the job and we can just drop the subscript j and writing μ_t as a function of t and B only: $\mu_t \triangleq \mu(t, B)$. Here $\mu(\cdot, \cdot)$ represents the cumulative distribution of the computation time $\{c_j\}_{j \in \mathbb{Z}_{\geq 0}}$. We also define the probability $\mu_o \triangleq 1 - \sum_{t=N}^{N_r} \mu_t$ of dropping the computations, i.e., of cancelling the job and going in open loop.

Remark 1: It is important to notice that, due to the dependence of all the probabilities μ_t on the same variable B, the vector $[\mu_N, \ldots, \mu_{N_r}, \mu_o] \in \mathcal{S}^{N_r - N}$ does not span the whole $(N_r - N)$-dimensional probability simplex $\mathcal{S}^{N_r - N}$. Such vector describes a one parameter curve w.r.t. B.

The dynamics of the covariance $P(j+1)$ of the state ζ for the $(j+1)$-th job is then given by

$$\begin{aligned} P(j+1) &= \mathrm{E}\left\{\zeta_{j+1}\zeta_{j+1}^T\right\} \\ &= \sum_{t=N}^{N_r} \mu_t \mathrm{E}\left\{\left(A_t\zeta_j + v(j)\right)\left(A_t\zeta_j + v(j)\right)^T\right\} + \\ &\quad + \mu_o \mathrm{E}\left\{\left(A_o\zeta_j + v(j)\right)\left(A_o\zeta_j + v(j)\right)^T\right\} \\ &= \sum_{t=N}^{N_r} \mu_t A_t P(j) A_t^T + \mu_o A_o P(j) A_o^T + H, \end{aligned} \tag{8}$$

where the last relation is obtained taking into account the mutual independence of the stochastic processes and the fact that $w(\cdot)$ has null mean and constant variance W. Finally, by suitably defining $f(p, q) \triangleq A^{p-q-1} W (A^{p-q-1})^T$ and $g \triangleq \sum_{q=0}^{N-1} f(N, q)$, we have

$$H = H_{N-1} + \sum_{t=N}^{N_r - 1} \mu_t H_t \tag{9}$$

with

$$H_{N-1} \triangleq \begin{bmatrix} g + \sum_{q=N}^{N_r - 1} f(q+1, 0) & 0 \\ 0 & 0 \end{bmatrix}, \tag{10}$$

$$H_t \triangleq \begin{bmatrix} -\sum_{q=t}^{N_r - 1} f(q+1, 0) & 0 \\ 0 & 0 \end{bmatrix}, \ t = N, \ldots, N_r - 1\,. \tag{11}$$

Using Kronecker product properties we can write the dynamics (8) as

$$\mathrm{vec}(P(j+1)) = \left(\sum_{t=N}^{N_r} \mu_t A_t^{[2]} + \mu_o A_o^{[2]}\right) \mathrm{vec}(P(j)) + \mathrm{vec}(H), \tag{12}$$

where $M^{[2]} \triangleq M \otimes M$ and $\mathrm{vec}(\cdot)$ is the linear operator producing a vector by stacking the columns of a matrix. This is a discrete–time linear time–invariant system in the state $\mathrm{vec}(P(j))$. Hence, it admits a steady state solution w.r.t. the constant input $\mathrm{vec}(H) \in \mathbb{R}^{n_c^2}$ if

$$\max_i \left| \lambda_i \left(\sum_{t=N}^{N_r} \mu_t A_t^{[2]} + \mu_o A_o^{[2]} \right) \right| < 1, \tag{13}$$

where with $\lambda_i(M)$ we mean the i-th eigenvalue of M. If such a condition is verified, we look for a steady state solution $\bar{P}$ solving the algebraic equation (see (8))

$$\bar{P} = \sum_{t=N}^{N_r} \mu_t A_t \bar{P} A_t^T + \mu_o A_o \bar{P} A_o^T + H. \tag{14}$$

Using again the Kronecker product we can write the unique solution of (14) as

$$\mathrm{vec}(\bar{P}) = S(\boldsymbol{\mu})^{-1}\mathrm{vec}(H), \tag{15}$$

with

$$S(\boldsymbol{\mu}) \triangleq \sum_{t=N-1}^{N_r} \mu_t \Gamma_t \tag{16}$$

$$\boldsymbol{\mu} \triangleq (\mu_{N-1}, \mu_N, \ldots, \mu_{N_r}) \qquad \mu_{N-1} \triangleq 1 \tag{17}$$

$$\Gamma_{N-1} \triangleq I - A_o^{[2]} \tag{18}$$

$$\Gamma_t \triangleq A_o^{[2]} - A_t^{[2]} \qquad t = N, \ldots N_r. \tag{19}$$

The inverse of $S(\boldsymbol{\mu})$ exists as a result of condition (13). Ensuring mean square stability for the system (6) turns to a problem of Schur stability for the matrix curve $\sum_{t=N}^{N_r} \mu_t A_t^{[2]} + \mu_o A_o^{[2]}$ (see also Remark 1). It is worth noting that we can always find some values of the bandwidth for which condition (13) is verified, since the closed loop system is asymptotically stable. Indeed any bandwidth ensuring $\mu_N = 1$ and $\mu_t = 0$ for any $t = N+1, \ldots, N_r$, satisfies condition (13). As we are not interested in giving to a control task more bandwidth than what is strictly necessary to finish without delay, we define the maximum bandwidth $\overline{B} < +\infty$ as

$$\overline{B} \triangleq \min\{b \geq 0 \mid \mu_N = \mu(N, b) = 1\}. \tag{20}$$

On the other hand, there are always values of the bandwidth for which condition (13) is not verified, as the open loop systems is supposed unstable. We can, thus, define the minimum bandwidth $\underline{B}$ as the limit value for which condition (13) is verified, but it is not verified for any $b < \underline{B}$. The problem with this definition is that it is based on condition (13), which is a strict inequality. Hence, the set of bandwidths verifying such condition is, in general, an open one. We can force a closed set by replacing condition (13) with the following one

$$\max_i \left| \lambda_i \left(\sum_{t=N}^{N_r} \mu_t A_t^{[2]} + \mu_o A_o^{[2]} \right) \right| \leq 1 - \epsilon \tag{21}$$

for a sufficiently small positive ϵ. In particular ϵ must be chosen such that $\max_i \left| \lambda_i \left(A_t^{[2]} \right) \right| < 1 - \epsilon$ in order to ensure that the bandwidth set is not empty. We define the set:

$$\mathcal{B}^\epsilon \triangleq \left\{ 0 \leq b \leq \overline{B} \mid \text{condition (21) is verified} \right\}. \tag{22}$$

We can finally define the minimum bandwidth as

$$\underline{B} \triangleq \min\{b \geq 0 \mid b \in \mathcal{B}^\epsilon\}. \tag{23}$$

Remark 2: The set (22) is closed and bounded, but there is no guarantee that it is also connected. In particular, it can be the union of many disjoint closed subsets. We assume by the following assumption that the number of these sets is finite.

Assumption 1: The number of zeros of the function $\max_i \left| \lambda_i \left(\sum_{t=N}^{N_r} \mu_t A_t^{[2]} + \mu_o A_o^{[2]} \right) \right| - (1-\epsilon)$ w.r.t. B is finite on the set $[\underline{B}, \overline{B}]$.

We also make the following mild assumption.

Assumption 2: The functions $\mu_t(\cdot)$, for any $t = N+1, \ldots, N_r$ have finitely many minima and maxima on on the set $\mathcal{B}^\epsilon$.

IV. Optimal Solution

We express the QoC as a measure of the steady state covariance matrix $\bar{P}$ given in the equations (14). In particular, the QoC of the i-th task is measured by the trace of $\bar{P}_i$ considered as a function of the probabilities μ_t^i and, hence, of the bandwidth B_i. We pose $\phi_i : \mathcal{B}_i^\epsilon \to \mathbb{R}_{>0} \cup \{+\infty\}$ and

$$\phi_i(B_i) \triangleq \text{Trace}\left(\bar{P}_i(B_i)\right). \tag{24}$$

We define the feasibility set of the optimisation problem (3) in terms of the definitions (20), (22) and (23) as

$$\mathcal{B} \triangleq \left\{ (B_1, \ldots, B_n) \in \mathbb{R}^n_{>0} \mid B_i \in \mathcal{B}_i^\epsilon, \quad \sum_{i=1}^n B_i \leq 1 \right\}, \tag{25}$$

where the index i refers to the i-th task. The optimisation problem (3) can be finally written as

$$\min_{B \in \mathcal{B}} \Phi(B) = \min_{B \in \mathcal{B}} \max_{i \in \{1,\ldots,n\}} \phi_i(B_i) \tag{26}$$

where $B \triangleq (B_1, \ldots, B_n)$ and $\mathcal{B}$ is given in (25). This optimisation problem has non trivial solutions only if the set $\mathcal{B}$ is non-empty. Therefore, in addition to the obvious condition $\underline{B}_i \leq \overline{B}_i$ that ensures that the non-emptiness of the set $\mathcal{B}_i^\epsilon$ (see definitions (20), (22) and (23)), we also need the following Assumption.

Assumption 3: The point $\underline{B} = (\underline{B}_1, \ldots, \underline{B}_n)$ is feasible, namely $\sum_{i=1}^n \underline{B}_i \leq 1$.

A. Characterisation of $\phi_i(\cdot)$

In order to proceed with the solution of the optimal problem (26), we need to characterise the functions $\phi_i(\cdot)$. We show that the $\phi_i(\cdot)$ can be expressed as the ratio of two polynomials in the probabilitiesμ_t^i. Recalling that $\text{Trace}\,(AB) = \mathrm{vec}(A^T)^T\mathrm{vec}(B)$ and (15), we can write $\text{Trace}\left(\bar{P}(\boldsymbol{\mu}^i)\right) = \mathrm{vec}(I)^T\mathrm{vec}(\bar{P}(\boldsymbol{\mu}^i)) = \mathrm{vec}(I)^T S(\boldsymbol{\mu}^i)^{-1}\mathrm{vec}(H)$, with $S(\boldsymbol{\mu}^i)$ defined as in (16) and $\boldsymbol{\mu}^i$ as in (17) (except for the index i of the i-th task). In order to compute $S(\boldsymbol{\mu}^i)^{-1}$ we make use of the recursive algorithm in [23], Fact 2.16.28.We can apply this fact in our context as follows. Define the polynomials

$$\Theta_0(\boldsymbol{\mu}^i) \triangleq I$$

$$\Theta_q(\boldsymbol{\mu}^i) \triangleq \sum_{i_1=N-1}^{N_r} \cdots \sum_{i_q=N-1}^{N_r} \mu_{i_1}^i \cdots \mu_{i_q}^i \Lambda_{i_1,\ldots,i_q},$$

$\forall q = 1, \ldots, N_r - N + 1$. The matrix coefficients $\Lambda_{i_1,\ldots,i_q} \in \mathbb{R}^{n_c^2 \times n_c^2}$ are given by the following recursive equations:

$$\Lambda_{i_i} \triangleq I$$

$$\Lambda_{i_1,i_2,\ldots,i_q} \triangleq \Gamma_{i_q}\Lambda_{i_1,\ldots,i_{q-1}} - \frac{1}{q}\text{Trace}\left(\Gamma_{i_q}\Lambda_{i_1,\ldots,i_{q-1}}\right) I,$$

$q \geq 2$. Thus we can write $S(\boldsymbol{\mu}^i)^{-1}$ as

$$\frac{N_r - N + 1}{d(\boldsymbol{\mu}^i)} \sum_{i_1=N-1}^{N_r} \cdots \sum_{i_{N_r-N}=N-1}^{N_r} \mu_{i_1}^i \cdots \mu_{i_{N_r-N}}^i \Lambda_{i_1,\ldots,i_{N_r-N}},$$

with

$$d(\boldsymbol{\mu}^i) \triangleq \sum_{i_1=N-1}^{N_r} \cdots \sum_{i_{N_r-N+1}=N-1}^{N_r} \mu^i_{i_1} \cdots \mu^i_{i_{N_r-N+1}} \gamma_{i_1,\ldots,i_{N_r-N+1}}$$

for suitable coefficients $\gamma_{i_1,i_2,\ldots,i_{N_r-N+1}}$.

Recalling (9), (10) and (11) and the definition $\mu_{N-1} = 1$, we can write $\text{vec}(H) = \sum_{t=N-1}^{N_r} \mu_t^i \text{vec}(H_t)$ and hence $\text{Trace}\left(\bar{P}(\boldsymbol{\mu}^i)\right) = \frac{n(\boldsymbol{\mu}^i)}{d(\boldsymbol{\mu}^i)}$ with

$$n(\boldsymbol{\mu}^i) \triangleq (N_r - N + 1)\cdot$$
$$\cdot \sum_{i_1=N-1}^{N_r} \cdots \sum_{i_{N_r-N}=N-1}^{N_r} \sum_{t=N-1}^{N_r} \mu^i_t \mu^i_{i_1} \cdots \mu^i_{i_{N_r-N}} \alpha_{i_1,\ldots,i_{N_r-N}t}$$

and $\alpha_{i_1,\ldots,i_{N_r-N}t} \triangleq \text{vec}(I)^T \Lambda_{i_1,\ldots,i_{N_r-N}} \text{vec}(H_t)$.

B. Degenerate Problem

The optimisation problem (26) can present some special cases that deserve a separate analysis.

Definition 4: The optimisation problem (26) is said to be *degenerate* if there exist $i, j \in \{1,\ldots,n\}$, $i \neq j$ such that $\phi_i(B_i) \geq \phi_j(B_j)$ for every $B_i \in \mathcal{B}_i^\epsilon$ and $B_j \in \mathcal{B}_j^\epsilon$. In such a case $\phi_i(\cdot)$ is said *to dominate* $\phi_j(\cdot)$.

In a degenerate case the bandwidth associated to a dominated function does not influence the cost function, but only the constraints defining the feasibility set. For this reason, if $\phi_j(\cdot)$ is a dominated function, we have $\Phi(B) = \max_{h\in\{1,\ldots,n\}} \phi_h(B_h) = \max_{h\in\{1,\ldots,n\}\setminus\{j\}} \phi_h(B_h)$. As a consequence, we can fix $B_j = \underline{B}_j$ thus ensuring the largest feasibility set. It can be easily verified that the feasibility set is now a $(n-1)$–dimensional subset of the original set $\mathcal{B}$. If in the optimisation problem there are more than one dominated function, say $n' < n$ functions with indices in the set $I' \subset \{1,\ldots,n\}$, then the feasibility set is the $(n-n')$–dimensional subset:

$$\mathcal{B}_{n'} \triangleq \mathcal{B} \setminus \left\{ B \in \mathbb{R}^n_{>0} \mid B_h = \arg\max_{b\in\mathcal{B}_h^\epsilon} \phi_h(b),\ \forall h \in I' \right\}. \tag{27}$$

C. The general case

In order to analyse the solution set of the non degenerate optimisation problem (26), we define the "rectified" functions $\underline{\phi}_i : [\underline{B}_i, \overline{B}_i] \to \mathbb{R}_{>0} \cup \{+\infty\}$ for every $i \in \{1,\ldots,n\}$ as follows

$$\underline{\phi}_i(B_i) \triangleq \min_{b\in\mathcal{B}_i^\epsilon} \phi_i(b), \tag{28}$$

and we consider also the associated optimisation problem

$$\min_{B\in\underline{\mathcal{B}}} \underline{\Phi}(B) = \min_{B\in\underline{\mathcal{B}}} \max_{i\in\{1,\ldots,n\}} \underline{\phi}_i(B_i), \tag{29}$$

where the feasibility set of the new optimisation problem is $\underline{\mathcal{B}} \triangleq \left\{(B_1,\ldots,B_n) \in \mathbb{R}^n_{>0} \mid B_i \in [\underline{B}_i, \overline{B}_i],\quad \sum_{i=1}^n B_i \leq 1\right\}$.

Remark 3: It is worth noting that, while the functions $\phi_i(\cdot)$ are defined only for bandwidths satisfying condition (13), i.e. for $B \in \mathcal{B}_i^\epsilon$, the rectified functions $\underline{\phi}_i(\cdot)$ are defined in the entire set $[\underline{B}_i, \overline{B}_i]$. Indeed, for any $B_i \in [\underline{B}_i, \overline{B}_i] \setminus \mathcal{B}_i^\epsilon$, the definition (28) and the fact that $\underline{B}_i \in \mathcal{B}_i^\epsilon$ imply that there exists a point $\tilde{B}_i < B_i$ with $\tilde{B}_i \in \mathcal{B}_i^\epsilon$ such that $\underline{\phi}_i(B_i) = \phi_i(\tilde{B}_i)$. In other words, the rectified functions $\underline{\phi}_i(\cdot)$ hold on the values assumed by the corresponding functions $\phi_i(\cdot)$ in some points $\tilde{B}_i \in \mathcal{B}_i^\epsilon$. These points can be the right extremal of one of the closed subsets making up $\mathcal{B}_i^\epsilon$, or a local minimum of $\phi_i(\cdot)$. To show the latter case, let us consider, for simplicity, any $B_i \in \mathcal{B}_i^\epsilon$ such that $\underline{\phi}_i(B_i) \neq \phi_i(B_i)$. It can be easily verified that $B_i^\star = \min_{\alpha\in\{b\in[\underline{B}_i,\overline{B}_i] \mid \underline{\phi}_i(b)=\underline{\phi}_i(B_i)\}} \alpha$ is a local minimum for $\phi_i(\cdot)$ and $\underline{\phi}_i(B_i) = \phi_i(B_i^\star)$..

Remark 4: Differently from the functions $\underline{\phi}_i(\cdot)$, the functions $\phi_i(\cdot)$ cannot assume a constant value on intervals. Indeed, being $\phi_i(\cdot)$ a ratio of polynomials and in view of Assumption 2, its derivative can only have a finite number of isolated zeros.

The rationale of the following Theorem 5 relies on the idea that the optimal solution of problem (29) belongs to the region in which all the functions $\underline{\phi}_i(B_i)$ are equal. If such a region exists, the optimal bandwidth allocation is either where one of the $\underline{\phi}_i(B_i)$ has reached its minimum value (point i)) or where it is not possible to further decrease it (point ii)). If the region in which the functions $\underline{\phi}_i(B_i)$ are equal does not exists, then there exists at least one function that will not play a role in the minimisation even if its bandwidth is equal to the minimum (point iii)).

Theorem 5: Assume that the optimisation problem (29) is not degenerate, the functions $\underline{\phi}_i : [\underline{B}_i, \overline{B}_i] \to \mathbb{R}_{>0} \cup \{+\infty\}$ are defined as in (28) and the functions $\phi_i : \mathcal{B}_i^\epsilon \to \mathbb{R}_{>0} \cup \{+\infty\}$ are continuous and differentiable for every $i \in \{1,\ldots,n\}$. Define the optimal solution set as $\underline{\mathcal{X}}^* \triangleq \arg\min_{B\in\mathcal{B}'} \underline{\Phi}(B)$, the values $\underline{t} \triangleq \max_{i\in\{1,\ldots,n\}} \underline{\phi}_i(\bar{B}_i)$, $\bar{t} \triangleq \min_{i\in\{1,\ldots,n\}} \underline{\phi}_i(\underline{B}_i)$, $\overline{\overline{t}} \triangleq \max_{i\in\{1,\ldots,n\}} \underline{\phi}_i(\underline{B}_i)$, the points $\check{B} = (\check{B}_1,\ldots,\check{B}_n)$ such that $\check{B}_i \triangleq \min_{\alpha\in\{b\in[\underline{B}_i,\overline{B}_i] \mid \underline{\phi}_i(b)=\bar{t}\}} \alpha$ and $\hat{B} = (\hat{B}_1,\ldots,\hat{B}_n)$ such that $\hat{B}_i \triangleq \min_{\alpha\in\{b\in[\underline{B}_i,\overline{B}_i] \mid \underline{\phi}_i(b)=\underline{t}\}} \alpha$ for every $i \in \{1,\ldots,n\}$. The following mutually exclusive cases are given:

1) $\sum_{i=1}^n \hat{B}_i \leq 1$, then $\hat{B} \in \underline{\mathcal{X}}^*$and the optimal value is $t^* = \underline{t}$;
2) $\sum_{i=1}^n \check{B}_i \leq 1$ and $\sum_{i=1}^n \hat{B}_i > 1$, then there exists $\tilde{B} = (\tilde{B}_1,\ldots,\tilde{B}_n)$, $\tilde{B} \in \underline{\mathcal{X}}^*$ such that $\underline{\phi}_i(\tilde{B}_i) = \underline{\phi}_j(\tilde{B}_j)$ and $\sum_{i=1}^n \tilde{B}_i = 1$. The optimal value is $t^* \in (\underline{t}, \overline{\overline{t}}]$;
3) $\sum_{i=1}^n \check{B}_i > 1$, then there exists $B^* = (B_1^*,\ldots,B_n^*)$ such that for $\bar{\imath} = \arg\max_{j\in\{1,\ldots,n\}} \underline{\phi}_j(\underline{B}_j)$, the element $B^*_{\bar{\imath}} = \underline{B}_{\bar{\imath}}$ and $B^* \in \underline{\mathcal{X}}^*$. The optimal values is $t^* \in (\overline{t}, \overline{\overline{t}}]$.

Remark 5: The degenerate case can be addressed by recalling that the subset $\mathcal{B}_{n'}$ in (27) is an $(n-n')$–dimensional set. We can, thus, apply the previous theorem to such a set, with the unique difference that now the feasibility constraint on the bandwidth is no longer to sum up to 1, but $\sum_{i\in\{1,\ldots,n\}\setminus I'} B_i = 1 - \sum_{i\in I'} \underline{B}_i$. The case 3 in Theorem 5 is similar to the degenerate case. Indeed, once

the $\bar{\imath}$-th component of B^* is fixed to $B^*_{\bar{\imath}} = \underline{B}_{\bar{\imath}}$, the other components are found solving the optimal problem on the subset $\mathcal{B}' \triangleq \underline{\mathcal{B}} \cap \{(B_1, \ldots, B_n) \in \mathbb{R}^n_{\geq 0} \mid B_i = \underline{B}_i\}$ with the feasibility constraint $\sum_{i \in \{1,\ldots,n\} \setminus \{\bar{\imath}\}} B_i = 1 - \underline{B}_{\bar{\imath}}$.

We can now formulate a corollary tackling the generic case of non-monotone functions $\phi_i(\cdot)$ and defined only on $\mathcal{B}^\epsilon_i$.

Corollary 6: Assume the optimisation problems (26) and (29) are not degenerate. Define the corresponding optimal solution sets as $\mathcal{X}^* \triangleq \arg\min_{B \in \mathcal{B}} \Phi(B)$ and $\underline{\mathcal{X}}^* \triangleq \arg\min_{B \in \underline{\mathcal{B}}} \underline{\Phi}(B)$. Given $\tilde{B}^* = (\tilde{B}^*_1, \ldots, \tilde{B}^*_n) \in \underline{\mathcal{X}}^*$, we have:

1) the point $B^* = (B^*_1, \ldots, B^*_n)$ such that $B^*_i = \min_{\alpha \in \{b \in [\underline{B}_i, \overline{B}_i] | \underline{\phi}_i(b) = \underline{\phi}_i(\tilde{B}^*_i)\}} \alpha$ for every $i \in \{1, \ldots, n\}$, is optimal for the problem (26), namely $B^* \in \mathcal{X}^*$. Moreover, for the optimal value we have $\Phi(B^*) = \underline{\Phi}(\tilde{B}^*)$;
2) if $\phi_i(\tilde{B}^*_i) = \underline{\phi}_i(\tilde{B}^*_i)$ for every $i \in \{1, \ldots, n\}$, then $\tilde{B}^* \in \mathcal{X}^*$; otherwise the optimal point $B^* \in \mathcal{X}^*$ defined at point 1 is such that there exist at least one index $\bar{\imath} \in \{1, \ldots, n\}$ for which $B^*_{\bar{\imath}} = B^\star_{\bar{\imath}}$ with $B^\star_{\bar{\imath}}$ local minimum of $\phi_{\bar{\imath}}(\cdot)$ or right boundary of one of the closed subset making up $\mathcal{B}^\epsilon_{\bar{\imath}}$.

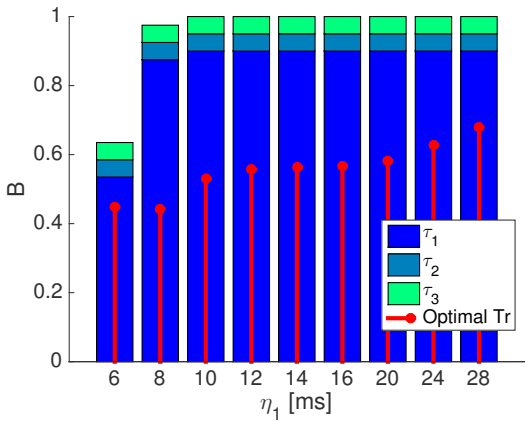
D. Algorithm

The Theorem 5 and the Corollary 6 can be exploited to devise an algorithm for the computation of an optimal solution for the problem (26). With reference to Theorem 5 point 1, the first step is to compute $\hat{B}$ and to check its feasibility. If the test fails, the condition at point 3 of the same theorem is then checked and, in case of success, one bandwidth is fixed to its minimal value and the algorithm is recursively applied on the subset defined in the Remark 5. In the more interesting case of the point 2 of Theorem 5, two nested binary search algorithms are employed. The inner one is responsible for computing the generalised inverse of the functions $\underline{\phi}_i(\cdot)$. That is, given a value $\check{t} \in [\underline{t}, \overline{t}]$, it provides the extreme points $\underline{B}^{\check{t}}_i \triangleq \min_{\alpha \in \{b \in [\underline{B}_i, \overline{B}_i] | \underline{\phi}_i(b) = \check{t}\}} \alpha$ and $\overline{B}^{\check{t}}_i \triangleq \max_{\alpha \in \{b \in [\underline{B}_i, \overline{B}_i] | \underline{\phi}_i(b) = \check{t}\}} \alpha$ for each $i \in \{1, \ldots, n\}$. Note that, if $\underline{\phi}_i(\cdot)$ is locally invertible in $\check{t}$, then $\underline{B}^{\check{t}}_i = \overline{B}^{\check{t}}_i$. The outer algorithm scans the interval $[\underline{t}, \overline{t}]$ searching for the optimal value. At each step the search interval is halved according to the following rule: if $\underline{B}^{\check{t}}$ is not feasible, i.e. $\sum_{i=1}^n \underline{B}^{\check{t}}_i > 1$, then the upper bound of the search interval for the next step is fixed to $\check{t}$; if $\overline{B}^{\check{t}}$ is such that $\sum_{i=1}^n \overline{B}^{\check{t}}_i < 1$, then it is the lower bound of the search interval to be fixed to $\check{t}$; if $\sum_{i=1}^n \underline{B}^{\check{t}}_i = 1$, then $\underline{B}^{\check{t}}$ is the optimal solution and $\check{t}$ is the optimal value. If, instead, $\underline{B}^{\check{t}}$ is feasible and $\overline{B}^{\check{t}}$ is not, then $\check{t}$ is the optimal value. Since we are interested to the solution of the problem (26), we can use the point 1 of the Corollary 6 to conclude that $\underline{B}^{\check{t}}$ is an optimal solution.

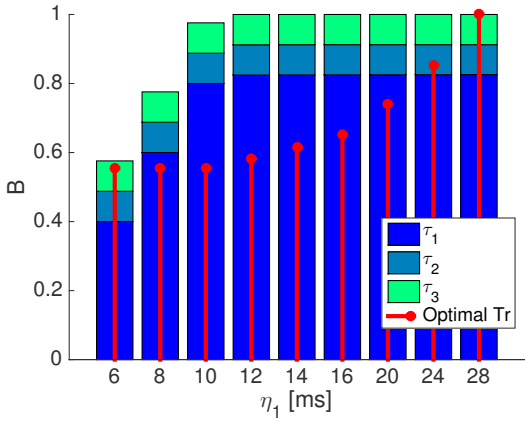
V. Numeric Evaluation

In this section, we provide a numerical analysis of the optimal bandwidth allocation algorithm presented in this paper. We consider a task set comprising three different tasks. The task τ_1 has a nominal sampling period of $T_1 = 20$ ms and a reservation period $R_1 = 4$ ms. Its execution time is drawn from a uniform distribution, parametrised with respect to its mean value η_1, which ranges in the set $\mathcal{M} = \{6, 8, 10, 12, 14, 16, 20, 24, 28\}$ ms to highlight the behaviour of the optimal algorithm in different situations. The uniform distribution $\mathfrak{U}_{[\eta_1,\, b]}$ is thus defined in the range $[b, w_1]$, being $b = R_1$ its best-case execution time and $w_1 = 2\eta_1 - b$ its worst-case execution time (WCET), hence $b \leq c_1 \leq w_1$. The task τ_2 has a sampling period of $T_2 = 56$ ms, a reservation period $R_2 = R_1$ and computation times distributed according to an exponential distribution, i.e., $\mathfrak{E}_{[\eta_2,\, b]} = \frac{1}{\eta_2} \mathrm{e}^{-\frac{c_2 - b}{\eta_2}}$, where the WCET is $w_2 = +\infty$ and where $\eta_2 = 6$ ms. Finally, the task τ_3 has computation times obeying to a Beta distribution $\mathfrak{B}_{[\eta_3,\, b]} = \frac{1}{N} \left(\frac{c_3 - b}{w_3 - b}\right)^{\alpha - 1} \left(1 - \frac{c_3 - b}{w_3 - b}\right)^{\beta - 1}$ with normalisation factor $N = \int_0^1 x^{\alpha-1}(1-x)^{\beta-1} dx$, having $T_3 = T_2$, $\eta_3 = \eta_2$ and WCET $w_3 = 3T_3$ and parameters $\alpha = 2$ and $\beta = \alpha \frac{w_3 - \eta_3}{\eta_3 - b}$. Notice that the task set considered is not hard real time schedulable for any choice of η_1, which remarks how the proposed method outperforms the commonly adopted hard-real time techniques based on the WCET. The relative deadlines were chosen equal to the period, i.e., $d_i = T_i$. We consider randomly generated, open-loop unstable, reachable and observable linear continuous time systems subject to a linear combination of continuous time noises. In order to be general, the number of independent noise sources equals the number of states, which is set to 2, 3 and 4 for each task, respectively. The noise processes are normally distributed with zero mean and standard deviation equal to $\sigma = 0.01$.

We report here the results on the bandwidth optimal synthesis described in the previous sections. Fig. 1-a shows the optimal bandwidths solving the problem (26) for all $\eta_1 \in \mathcal{M}$. For comparison, Fig. 1-b shows the results when the delay is not explicitly considered, as previously presented in [20]. The solid thick line represents the normalised value of the optimal trace with respect to he maximum achieved by the two methods. Fig. 1 clearly shows that the proposed Continuous Stream method tolerating the delays allows an improvement in the QoC with respect to [20], which ranges from a few percent (for $\eta_1 \in \{10, 12, 14\}$) up to 30% in the other cases. The optimal trace increases, as expected, as τ_1 becomes more and more demanding. In the first three cases, the optimal solution is given for $B_1 = \overline{B}_1$, while $\underline{B}_i < B_i < \overline{B}_i$, $i = 2, 3$. In all the other cases, $\underline{B}_i < B_i < \overline{B}_i$, $i = 1, 2, 3$, with $B_1 > B_2 > B_3$. We remark a slight increase of B_1 and a corresponding decrease of B_2 and B_3 (until $B_3 = \underline{B}_3$ for the last experiment) as $\eta_\star$ grows beyond 12 ms, since from $\eta_\star = 12$ ms the system computing power becomes fully exploited. It is worthwhile to note that considering explicitly the delays and adopting the Continuous Stream not only leads to a remarkable improvement in the system performance, but also allows a fully exploitation of the computing power.



(a)



(b)

Fig. 1. Bandwidth optimal allocation minimising Trace (P) with respect to the randomly generated task set. The solid thick lines represent the normalised value of the traces. Results considering the presence of the delay (a) or not (b) are both reported for comparison.

VI. Conclusions

In this paper we have presented a solution for optimal sharing of computation resources between different competing control tasks. We have shown the combination of a RR scheduler with a novel model of computation called Continuous Stream. The resulting problem is easy to analyse and it is possible to establish a functional dependence between the bandwidth allocated to the task and the QoC it delivers, where the latter is related to the steady state covariance of the state. This result is the gateway to a more ambitious effort: setting up an optmisation problem where the QoC of the different tasks is consolidated into a global QoC and the constraint is that the total bandwidth assigned cannot exceed 100%. The optimisation problem can be solved with a limited computational effort. The result of the optimisation problem are shown on a selection of numeric examples.

Our future work directions will be toward finding real–life application examples for the proposed technique in the automotive and in the robot domain. Another possibility for future work is the exploration of different ways of consolidating the QoC of the different tasks into a global QoC metric, different form the infinity norm considered in this paper.

References

[1] H. Kopetz and G. Bauer, "The time-triggered architecture," *Proceedings of the IEEE*, vol. 91, no. 1, pp. 112–126, 2003.

[2] J. Nilsson and B. Bernhardsson, "Analysis of real-time control systems with time delays," in *Proc. IEEE Conf. on Decision and Control*. IEEE, Dec. 1996, pp. 3173–3178.

[3] Q. Ling and M. Lemmon, "Robust performance of soft real-time networked control systems with data dropouts," in *Proc. IEEE Conf. on Decision and Control*. IEEE, Dec. 2002, pp. 1225–1230.

[4] P. Marti, J. Fuertes, G. Fohler, and K. Ramamritham, "Jitter compensation for real-time control systems," in *Proc. IEEE Real-Time Systems Symposium*, Dec. 2001, pp. 39–48.

[5] B. Lincoln and A. Cervin, "JITTERBUG: a tool for analysis of real-time control performance," in *Proc. IEEE Conf. on Decision and Control*. IEEE, Dec. 2002, pp. 1319–1324.

[6] C. Kao and A. Rantzer, "Stability analysis of systems with uncertain time-varying delays," *Automatica*, vol. 43, no. 6, pp. 959–970, June 2007.

[7] G. Buttazzo, M. Velasco, and P. Marti, "Quality-of-Control Management in Overloaded Real-Time Systems," *IEEE Trans. on Computers*, vol. 56, no. 2, pp. 253–266, Feb. 2007.

[8] T. Chantem, X. S. Hu, and M. Lemmon, "Generalized Elastic Scheduling for Real-Time Tasks," *IEEE Trans. on Computers*, vol. 58, no. 4, pp. 480–495, April 2009.

[9] A. Cervin and J. Eker, "The control server: A computational model for real-time control tasks," in *ECRTS*. IEEE Computer Society, 2003, pp. 113–120.

[10] D. Fontanelli, L. Greco, and L. Palopoli, "Soft RealTime Scheduling for Embedded Control Systems," *Automatica*, vol. 49, no. 8, pp. 2330–2338, July 2013.

[11] C. L. Liu and J. Layland, "Scheduling alghorithms for multiprogramming in a hard real-time environment," *Journal of the ACM*, vol. 20, no. 1, 1973.

[12] R. Rajkumar, K. Juvva, A. Molano, and S. Oikawa, "Resource kernels: A resource-centric approach to real-time and multimedia systems," in *Proc. of the SPIE/ACM Conference on Multimedia Computing and Networking*, January 1998.

[13] L. Abeni and G. Buttazzo, "Integrating Multimedia Applications in Hard Real-Time Systems," in *Proc. IEEE Real-Time Systems Symposium*, Dec. 1998, pp. 4–13.

[14] M. Rabi and K. Johansson, "Scheduling packets for event-triggered control," in *Proc. of 10th European Control Conference (ECC)*, 2009, pp. 3779–3784.

[15] X. Wang and M. D. Lemmon, "Event-triggering in distributed networked control systems," *IEEE Trans. Automat. Contr.*, vol. 56, no. 3, pp. 586–601, 2011.

[16] M. Mazo and P. Tabuada, "Input-to-state stability of self-triggered control systems," in *Proc. IEEE Conf. on Decision and Control*. IEEE, Dec. 2009, pp. 928–933.

[17] M. Velasco, P. Martí, and E. Bini, "Control-driven tasks: Modeling and analysis," in *IEEE Real-Time Systems Symposium*. IEEE Computer Society, 2008, pp. 280–290.

[18] A. Quagli, D. Fontanelli, L. Greco, L. Palopoli, and A. Bicchi, "Design of Embedded Controllers Based on Anytime Computing," *IEEE Trans. on Industrial Informatics*, vol. 6, no. 4, pp. 492–502, November 2010.

[19] L. Palopoli, D. Fontanelli, N. Manica, and L. Abeni, "An Analytical Bound for Probabilistic Deadlines," in *Euromicro Conf. on Real-Time Systems (ECRTS)*, july 2012, pp. 179–188.

[20] D. Fontanelli, L. Palopoli, and L. Greco, "Optimal CPU Allocation to a Set of Control Tasks with Soft Real–Time Execution Constraints," in *Hybrid Systems: Computation and Control (HSCC)*. Philadelphia, PA, USA: ACM, April 2013, pp. 233–242.

[21] O. Costa, M. Fragoso, and R. Marques, *Discrete-time Markov jump linear systems*. Springer, 2006.

[22] D. Fontanelli, L. Palopoli, and L. Abeni, "The Continuous Stream Model of Computation for Real–Time Control," in *2013 IEEE 34th Real-Time Systems Symposium (RTSS)*. Vancouver, Canada: IEEE, 4-6 Dec. 2013, pp. 150–159.

[23] D. S. Bernstein, *Matrix mathematics: theory, facts, and formulas*. Princeton University Press, 2009.