Abstract—In this paper, we assume that a set of non preemptive controller tasks should be implemented on a limited computational resource platform, and look for a sampling period assignment that allows to obtain the desirable performance. The problem is formulated as a multi-objective optimization problem under a resource constraint, where the cost functions depend on the sampling period. Linear-quadratic controllers are used, resulting on feedback gains that also depend on the sampling period. The global cost function is chosen as a weighted sum of all plants performances, translating the multi-objective optimization problem into a single-objective one which provides an additional degree of freedom and leads to a set of solutions denoted as Pareto efficient. To handle this additional variable, we assume a Nash bargaining cooperative game. An upper level task performs the update of the sampling period and of the plant input, to be used on a finite-horizon control strategy, for each control loop. A numerical example is provided to illustrate our approach.

I. INTRODUCTION

Controllers implemented over computing platforms are more and more common nowadays. In this kind of systems, the computational resource is usually limited and shared between several control tasks. Therefore, it might be desirable to combine some characteristics of control theory and real-time scheduling theory, the so-called co-design, providing an appropriate performance level for each loop.

In this matter, two issues of interest are often investigated in the literature [15]: i) Given a performance level requirement, obtain the lowest level of resource utilization needed; ii) Given an allowable resource utilization level, obtain the best possible control performance level.

Some interesting approaches in the first one are the event-triggered and self-triggered methods, in which a triggering mechanism determines when the control input has to be updated [10], and consequently the use of the shared resource.

In the present work we investigate the second issue. In this context, [16] presents a study about the performance optimization for a set of tasks under resource constraints. The authors consider an off-line approach and a performance index approximated by an exponential function for each loop. Later, their work has been improved into an online approach [7], where each loop performance was described approximating an LQ-cost as a quadratic function of the sampling period, but without taking into account the current plant states.

Online sampling period assignment is also analyzed in [11], where the solution is obtained by solving an optimization problem regarding the expected future performance of the control loops, taking into account the actual states values and the expected noise. By using this strategy, resources can be accordingly distributed with respect to the current control performance. Furthermore, in [5], an algorithm that approximates the solution at run-time, based on a set of possible sampling periods, for the generic case of convex cost functions on the sampling period, is also presented.

As in [5], [11], in this paper we consider a system where a set of non preemptive control tasks share one processor with limited computational resources, and an online adjustment of the sampling periods is possible. Although previous works present solutions for this co-design problem, they do not allow the designer to provide more information about how the resource might be distributed between the control loops. Thereby, we generalize the results already presented in the literature using a vector of weighting parameters, providing an additional degree of freedom to the assignment of the sampling period.

The overall problem is actually a multi-criteria minimization problem, since we are looking for the control signal and the sampling period for each control loop that results in a suitable performance of the entire system. Nevertheless, it is possible to solve it in two steps. First determining a solution for the control design, based on a quadratic cost function used for each system, and dependent on the sampling period. Later, the sampling periods are allowed to be re-assigned based on the considered performance requirement, subjected to a resource allocation constraint.

The performance of each plant is captured in a finite-time horizon quadratic cost function, that should be minimized and takes into account the sampling period and the current state of the plant, and may also be interpreted as a predictive control [2]. Nevertheless, the global minimization of all performance indicators is usually not possible, and characterizes a multi-objective optimization problem. Thus, in order to obtain a single-objective optimization problem, the global objective function is translated into a weighted sum of the control loops performance, which leads to a Pareto efficient solution, requiring a decision maker in order to choose a single suitable solution among all possible ones from the optimal set. Although decision making is a crucial aspect in the design, it is seldom mentioned in the multi-objective
optimization control theory [9]. In our approach, despite
the possibility that the weights may be chosen arbitrarily,
we consider an iterative procedure issued from a Nash
bargaining game. Therefore, an additional task is performed,
that may update the sampling period of the control loops.

The paper is organized as follows. In Section II the
main problem is formulated. Some preliminary concepts
regarding Pareto efficiency and Nash bargaining are
presented in Section III. In Section IV we depict the main
approach regarding the sampling period assignment, from
the constrained optimization problem to the weighting parameter
computation. Finally, Section V presents a discussion about
the studied approach, based on a numerical example, and
Section VI presents some concluding remarks.

II. PROBLEM FORMULATION

We deal with a real-time system running \( p \) controller tasks
that share a computing resource, the central processing unit
(CPU), and where each task is responsible for sampling,
control computation and actuation of a plant. The controlled
plants dynamics are given by:

\[
x_i(t) = A_ix_i(t) + B_iu_i(t) + w_{c,i}(t), \quad \forall i = 1, \ldots, p
\]

(1)

where \( x_i(t) \in \mathbb{R}^{n_i} \) are the plant state vectors, \( u_i(t) \in \mathbb{R}^{m_i} \)
are the control signal vectors, \( A_i \in \mathbb{R}^{n_i \times n_i} \)
are the dynamic matrices, \( B_i \in \mathbb{R}^{n_i \times m_i} \)
are the input matrices, and \( w_{c,i}(t) \) are white noises with incremental variances \( R_{c,i} \). Also, assume
that the initial state conditions are known and equal to \( x_i(0) = x_0_i \).

The control design is formulated as solving an optimization
problem, where the cost function to minimize is the expression of a compromise between several requirements
(stability or more precisely remaining in the vicinity of the origin,
minimizing the energy of the state and of the control
input, in the presence of noise). Thus, the cost function is
defined as a finite-time horizon quadratic one, defined as:

\[
J_i(x_{0,i}, u_i) = \mathbb{E}\left\{ \int_0^{T_{RM}} \left( x_{1c,i}^T(t)Q_{1c,i}x_{1c,i}(t) + 2x_{1c,i}^T(t)Q_{12c,i}u_i(t) + u_i^T(t)Q_{2c,i}u_i(t) \right) dt + x_{1c,i}(T_{RM})Q_{0,1c,i}(T_{RM}) \right\}
\]

(2)

where \( \mathbb{E}\{\cdot\} \) stands for the mathematical expectation
due to the noise variable. The weighting matrices
\( \begin{bmatrix} Q_{1c,i} & Q_{12c,i} \\ Q_{12c,i} & Q_{2c,i} \end{bmatrix} \) are design parameters,
with the last one related to the importance given to the final states
of the plant.

The controller tasks behave as digital controllers. Thus,
sampling the continuous-time plants (1) with periods \( h_i \),
gives the discrete-time systems:

\[
x_{i,j+1} = \Phi_i(h_i)x_{i,j} + \Gamma_i(h_i)u_{i,j} + w_{i,j},
\]

(3)

with \( \Phi_i(h_i) = e^{A_i h_i}, \Gamma_i(h_i) = \int_0^{h_i} e^{A_i \tau}B_i d\tau \). Also, we have that
\( R_i(h_i) = \mathbb{E}\{w_{i,j}w_{i,j}^T\} = \int_0^{h_i} \mathbb{E}\{\Phi_i(s)\}R_{c,i}\Phi_i(s) ds \).

Assuming that \( T_{RM} \) is a multiple of the sampling periods,
and consequently \( N_i = \frac{T_{RM}}{h_i} \) are integers, the sampled version
of the cost functions (2) are then given by [5]:

\[
J_i(x_{0,i}, u_i, h_i) = \mathbb{E}\left\{ \sum_{t=0}^{N_i-1} \left( x_{1c,i}^T(t)Q_{1c,i}(h_i)x_{1c,i}(t) + 2x_{1c,i}^T(t)Q_{12c,i}(h_i)u_{i,j} + u_{i,j}^T(h_i)Q_{2c,i}(h_i)u_{i,j} + R_{w,i}(h_i) \right) + x_{1c,i}(T_{RM})Q_{0,1c,i}(T_{RM}) \right\},
\]

(4)

with weighting matrices \( \begin{bmatrix} Q_{1c,i}(h_i) & Q_{12c,i}(h_i) \\ Q_{12c,i}(h_i) & Q_{2c,i}(h_i) \end{bmatrix} \) are design parameters [1] due to the inter-sample noise
\( R_{w,i}(h_i) = \mathbb{E}\{\Phi_i(s)\}R_{c,i}(s)ds \), where \( \mathbb{E}\{\cdot\} \) stands for the
trace of a matrix.

As stated previously, all controller tasks share the same
computational resource, thus a schedulability test has to
be performed in order to guarantee that all deadlines will
be respected. To meet this goal, we assume the following
properties for each task: the worst case execution time is
given by \( c_i \); the period is equal to the sampling period,
as well as the relative deadline; and the task utilization level is
given by \( U_i = \frac{c_i}{h_i} \).

Hence, considering a deadline monotonic strategy and
based on the utilization of the shared resource, the schedulability
of the task set can be guaranteed if:

\[
\sum_{j=1}^{p} \frac{c_j}{h_j} \leq U_{sp},
\]

(5)

where \( U_{sp} \) is the global utilization level available.

In the sequel we consider the problem of a dynamic
sampling period assignment, which can be performed online
by a supervisor task, namely Resource Manager (RM). The
RM computations take into account the plants individual
performances, as well as the sharing resource constraint.
Furthermore, the supervisor is assumed to have the knowledge
of each player’s dynamics and cost functions.

III. BASIC CONCEPTS

A. Pareto efficiency and the bargaining game

We deal with the situation where more than one controller
task, that we may call players, running on the same processor.
Due to the shared resource, a policy used to minimize the
cost function from one player, may have a negative effect
on another player’s performance. Thus, a multi-objective
optimization procedure must be taken in account.

Let \( J(h) = [J_1(h_1), \ldots, J_p(h_p)]^T \) be a vector with all \( p \) cost
functions, and \( h = [h_1, \ldots, h_p]^T \), a vector with the sampling
periods. Due to the presence of multiple cost functions, it
is not possible to define an optimal solution. It is required
to consider compromises and equilibria, which can be determined
thanks to game theory [8].

Among several possibilities, the notion of Pareto efficiency
is considered here. A solution point is Pareto efficient if it is
not possible to move from that point and improve at least one
objective function without detriment to any other objective function. That is, a point \( h^* \) is Pareto efficient if and only if there does not exist any other point, \( h \), such that \( J(h) \leq J(h^*) \), and \( J_i(h) < J_i(h^*) \) for at least one function.

Generally, there is not a unique Pareto efficient point \( h^* \). The outcome \( \{J_1(h^*_1), \ldots, J_p(h^*_p)\} \) associated with a Pareto efficient point \( h^* \) is called the Pareto payoff. The set of all Pareto payoffs defines the Pareto frontier, which is depicted in the criteria space and is denoted as the \( P_f \) curve in the sequel and in Fig 1.

Let define the simplex \( \mathcal{B} = \{ \beta = [\beta_1, \ldots, \beta_p] \mid \beta_i \geq 0 \) and \( \sum_{i=1}^p \beta_i = 1 \} \). For any \( \beta \in \mathcal{B} \), \( h^* \in \arg\min_h \sum_{i=1}^p \beta_i J_i(h) \) is Pareto efficient. The reciprocal is true only if the cost functions \( J_i(h) \) are convex with respect to \( h \). That is, when \( J_i(h) \) is convex, if \( h^* \) is Pareto efficient, there exists \( \beta^* \in \mathcal{B} \) such that \( h^* \in \arg\min_h \sum_{i=1}^p \beta_i J_i(h) \). In the convex case, the latter implication is an elegant way to parameterize the Pareto frontier. The Pareto frontier translates the multi-objective optimization problem into a single-objective one, by using a parameterized objective function.

### B. Nash bargaining game

It is important to note that, for applications, it is often necessary to incorporate some decision rule in order to determine a single suitable Pareto efficient point to be used. This means that from each player’s set of possible outcomes, only one has to be cooperatively chosen. The question is which outcome might the players possibly cooperatively choose. This dealing is carried out by a decision making process [14]. More specifically, here we consider a bargaining game approach, namely Nash bargaining.

The Nash bargaining rule deals with the idea in which players, through cooperation, can achieve better outcomes than the one which becomes effective when they do not cooperate. The non-cooperative outcome is called disagreement point, and is given by the point \( d(d_1, \ldots, d_p) \).

In Figure 1, a sketch of a typical Nash bargaining for a two players game is depicted. The curve \( P_f \) is the set of Pareto efficient outcomes of the game (Pareto frontier), the point \( d \) is the disagreement point, and the point \( N_b \) corresponds to the obtained solution. The asymptotes are related to the utopia point, i.e., when there is a \( \beta_i = 1 \), and \( \beta_j = 0, \forall j \neq i \).

The Nash bargaining rule chooses the point on \( P_f \) that maximizes the product of the players gains from disagreement point \( d \). That is, for \( J(h) \in P_f \) with \( J_i(h) \leq d_i \):

\[
N_b = \arg \max_{J(h) \in P_f} \prod_{i=1}^p (d_i - J_i(h)).
\]

Under the assumption that all cost functions \( J_i(h) \) are convex, and since the solution \( N_b \) is also located on the Pareto frontier, we conclude that for some \( N_b \) there is also a \( \beta^{N_b} \in \mathcal{B} \), and it is given by [8]:

\[
\beta_i^{N_b} (d_1 - J_1^{N_b}) = \cdots = \beta_p^{N_b} (d_p - J_p^{N_b})
\]

or equivalently

\[
\beta_i^{N_b} = \frac{\prod_{j \neq i} (d_j - J_j^{N_b})}{\sum_{j=1}^p \prod_{\ell \neq j} (d_\ell - J_\ell^{N_b})}, \quad i = 1, \ldots, p.
\]

We remark that the selection of the disagreement point \( d \), depending on the problem to be dealt with, may not be a trivial task. A possible choice might be using, for each \( d_i \), the upper limit for the related cost function \( J_i(h) \), given by physical, operational or even design constraints.

### IV. Sampling Period Assignment

Let now associate the schedulability constraint (5) to a weighted sum of the performance requirements (4), with respect to a \( k \)-th instant:

\[
\min_{u_1, \ldots, u_p} \sum_{i=1}^p \beta_i J_i(x_{k,i}, u_i, h_i) \quad \text{s.t.} \quad \sum_{j=1}^p c_j U_{sp} \leq U_{sp},
\]

where \( \sum_{i=1}^p \beta_i = 1 \), \( \beta_i \geq 0 \), \( i = 1, \ldots, p \), are the weighting parameters.

We first solve this problem in terms of the control vectors \( u_i \), following by an approach to determine the sampling periods to be used for each task. The parameters \( \beta_i \) are used to weight the plants performance, and are computed by an iterative algorithm based on a Nash bargaining approach.

#### A. Finite-horizon linear quadratic controller

The control law to be used for each plant \( i \) is actually dependent only on the plant \( i \) dynamics. We solve this by computing the finite-horizon linear quadratic controller independently for each plant, as a function of its own sampling period. Thus, we obtain minimum cost functions

\[
J_i^*(x_{i,k}, h_i) = \min_{u_i} J_i(x_{i,k}, u_i, h_i),
\]

that leads to

\[
J_i^*(x_{i,k}, h_i) = x_{i,k}^T S_{0,i}(h_i) x_{i,k} + q_{0,i}(h_i),
\]

which is a quadratic function in the current state vector \( x_{i,k} \).

The positive semidefinite weighting matrix \( S_{0,i}(h_i) \) and the additional scalar term due to the disturbances, \( q_{0,i}(h_i) \geq 0 \),
are both given by backward induction, from where we have, for each instant \( t = 0, \ldots, N_t - 1 \):
\[
S_{t,i}(h_i) = \Phi_t^i(h_i)S_{t+1,i}(h_i) + Q_t,i(h_i)
\]
\[
- \left( \Phi_t^i(h_i)S_{t+1,i}(h_i)\Gamma_t(h_i) + Q_{t+1,i}(h_i) \right)
\]
\[
\times \left( \Gamma_t^i(h_i)S_{t+1,i}(h_i)\Phi_t(h_i) + Q_{t+1,i}(h_i) \right)^{-1}
\]
\[
\times \left( \Gamma_t^i(h_i)S_{t+1,i}(h_i)\Phi_t(h_i) + Q_{t+1,i}(h_i) \right)
\]
and \( q_t,i(h_i) = \text{tr}(S_{t+1,i}(h_i)R_t(h_i)) + R_{t+1,i}(h_i) + q_{t+1,i}(h_i) \). The cost function at \( t = N_t \) is assumed to be given by \( J^*_t(x_{N_t}) = x_{N_t}^T S_{N_t}x_{N_t} \), with \( q_{N_t,i} = 0 \) and \( S_{N_t,i} = Q_{0,i} \). Furthermore, as \( \Phi_t(h_i), \Gamma_t(h_i), Q_{t,i}(h_i), Q_{t+1,i}(h_i) \) and \( Q_{t+1,i}(h_i) \) depend on the sampling period \( h_i \), this implies a solution \( S_{0,i}(h_i) \) that also depends on the sampling period.

Finally, the obtained controller is time-varying, and depends on both, the sampling period and the state of the plant, and is given by
\[
u_{ij} = K_{ij}(h_i)x_{i,t} = -\left( \Gamma_t^i(h_i)S_{t+1,i}(h_i)\Gamma_t(h_i) + Q_{t+1,i}(h_i) \right)^{-1}
\]
\[
\times \left( \Gamma_t^i(h_i)S_{t+1,i}(h_i)\Phi_t(h_i) + Q_{t+1,i}(h_i) \right)x_{i,t}
\]

B. Resource allocation

Notice that now the objective in (9) has become the minimization of a weighted sum of the cost functions (10):
\[
\min_{h_i \ldots = h_p} \sum_{i=1}^p \beta_i J^*_i(x_{k,i}, h_i) \quad \text{s.t.} \quad \sum_{j=1}^p \frac{c_j}{h_j} \leq U_{sp},
\]
Equation (11) is convex thanks to the convexity of the cost functions \( J^*_i \). Moreover, if the cost functions can be described in the form
\[
J^*_i(x_{k,i}, h_i) = \alpha_i + \gamma_i h_i,
\]
with \( \gamma_i, h_i \geq 0, i = 1, \ldots, p \), then it is possible to find an explicit solution for the optimization problem (11).

However, we recall that, in general, the cost functions are not always described as (12). Nevertheless, it is possible to approximate the cost functions by linearizing them around the current sampling periods, \( h_0^i \). We may then, solve (11) iteratively as follows.

First, let the cost functions, \( g_i(h_i) = I^*_i(x_{k,i}, h_i) \), be decomposed into a first order approximation:
\[
g_i(h_i) = g_i(h_0^i) + g_i'(h_0^i)(h_i - h_0^i) + \theta_i
\]
where \( g_i' \) is the first order derivative of \( g_i \) with respect to the current sampling period \( h_0^i \), and \( \theta_i \) corresponds to a remainder.

Note that if the plants are sampled reasonably fast, the term \( \theta_i \) might be small enough, and thus it can be neglected. Thereby, assuming a limited development for the equation (13), the cost functions (10) can be approximated by affine functions of \( h_i \):
\[
J^*_i(x_{k,i}, h_i) \approx \alpha_i + \gamma_i h_i.
\]

Thereby, we get:
\[
\gamma_i = g_i'(h_0^i) = x_{k,i}^T \frac{\partial S_{0,i}(h_0^i)}{\partial h_i} x_{k,i} + \frac{\partial q_{0,i}(h_0^i)}{\partial h_i}.
\]

Second, new sampling periods are computed using the affine functions (14), to find a solution to the optimization problem (11). For ease of notation, let us define a vector \( h = [h_1, \ldots, h_p]^T \) and functions \( J_p(h) = \sum_{i=1}^p \beta_i J^*_i(x_{k,i}, h_i) \) and \( f(h) = \sum_{i=1}^p \frac{c_i}{h_i} - U_{sp} \). Hence, the optimization problem (11) can be rewritten as
\[
\min_{h} J_p(h) \quad \text{s.t.} \quad f(h) \leq 0.
\]

Since (16) is a constrained optimization problem, we apply Karush-Kuhn-Tucker conditions [12], which leads to the equivalent unconstrained problem:
\[
\min_{h} J_p(h) + \lambda f(h),
\]
where the scalar \( \lambda \geq 0 \) is the Lagrangian multiplier. Supposing an \( h^* \) that minimizes (17), then
\[
\nabla J_p(h^*) + \lambda \nabla f(h^*) = 0.
\]

which after some manipulations gives, for \( i = 1, \ldots, p \):
\[
h_i^* = \sqrt{\frac{c_i}{\beta_i}} \sum_{j=1}^p \sqrt{\frac{c_j \beta_j}{U_{sp}}}. \]

Finally, these two steps may be repeated to improve the solution. Observe that the involved derivatives must be computed off-line and stored in the memory.

C. Nash bargaining algorithm

Here, we briefly outline an algorithm to compute the \( N_h \)-solution. As often occurs in applications, the Pareto frontier can be very flat and the solution of this kind of problem is not straightforward, even if we have a convex surface. However, the existence of relations (8) facilitates the approach.

Thus, the following steps may be used for an iterative computation of the weighting parameters:

Step 0 Set tuning parameters \( \Delta, \delta_1, \delta_2 \in (0,1) \) and \( \epsilon > 0 \).

Step 1 Set the disagreement point \( d(\delta_1, \ldots, \delta_p) \).

Step 2 Set initial weight parameters \( \beta_i^0, i = 1, \ldots, p, \sum_{i=1}^p \beta_i = 1 \). For example, \( \beta_i^0 = \frac{1}{p} \).

Step 3 Compute \( h_i(\beta_i^0) \).

Step 4 Verify wether \( J_i(h_i(\beta_i^0)) \leq d_i, i = 1, \ldots, p \). If not, then there is an \( I \) for which \( J_i(h_i(\beta_i^0)) > d_i \). In that case update \( \beta_i^0 := \beta_i^0 + \Delta, \beta_i^0 := \beta_i^0 - \Delta \frac{d - d}{p-1} \), for \( i \neq I \) and return to Step 3.

Step 5 For \( i = 1, \ldots, n \), compute
\[
\beta_i = \frac{\prod_{j\neq(k)}(d_j - J_j(h_i(\beta_i^0)))}{\sum_{j=1}^p \prod_{j\neq(k)}(d_j - J_j(h_i(\beta_i^0)))}.
\]

Step 6 If \( |\beta_i - \beta_i^0| < \epsilon, i = 1, \ldots, p \), then finish the algorithm and set \( \beta_i^* = \beta_i \). Else \( \beta_i^0 := \delta_i \beta_i^0 + \delta_i \beta_i^* \), and return to Step 3.

The tuning parameters used in the previous steps may be, to a certain level, chosen arbitrarily. In Step 4, \( \Delta \) corresponds
to an updating parameter used in case of failure of the condition described in the referred step. In Step 6, $e$ is used as a precision parameter between two consecutive iterations. Also in Step 6, the updating parameters $\delta_1$ and $\delta_2$ must be chosen in order to prevent too large steps in the update process, which might result in values $\beta_i^0$ for which the inequalities $J_i(h_i(\beta_i^0)) \leq d_i$ are no longer satisfied. Furthermore, this algorithm shows to have a fast convergence, as suggested in [8] and also verified in our simulation experiments.

D. Online resource management

We may now define a general procedure for the online sampling period computation, as follows.

a) Plant states $x_i(t)$ are sampled every $T_{RM}$ and then used as initial conditions $x_{k,i}$ for the current finite-time horizon.

b) The values of $\gamma$ (15) are updated using the current states $x_{k,i}$ and sampling periods $h_i^0$.

c) Decision making procedure takes place to find weighting parameters $\beta$.

d) New sampling periods to be applied to each plant are computed from (19) and updated over the controller tasks.

We recall that this procedure may be repeated from Step b to Step d to improve the results.

V. Example

In this section, we illustrate the approach studied in this paper with an example comprised of two cruise control systems. We provide a discussion about the obtained results with respect to the choice of disagreement point.

The purpose of the cruise control system is to maintain constant the vehicle speed despite external disturbances. Thus, for our example we consider a simplified first order model of a vehicle dynamics, borrowed from [6]:

$$\dot{x}_i(t) = v_i(t) = -\frac{5}{100} v_i(t) + \frac{1}{1000} u_i(t) + w_i(t),$$

with $w_i(t)$ a white noise with incremental variance $R_{ei} = 1$.

Sampling the plants with periods $h_i$ leads to the discrete-time systems:

$$v_i(t+1) = e^{-h_i} v_i(t) + \frac{1}{50} \left(1 - e^{-h_i}\right) u_i(t) + w_i(t).$$

For further analysis, we assume different initial condition for each plant: $x_1(0) = 2.7$ and $x_2(0) = -10$.

The RM runs once every $1.5s$ (at the instants $k_r = rT_{RM}$), for simplicity, on a dedicated processor unit. We assume $c_i = 0.1s$ for both control tasks. Furthermore, we consider a rate monotonic scheduling algorithm, which gives an available utilization level $U_{sp} = 2(2^7 - 1) \approx 0.828$ [13].

The linear quadratic controllers are designed with $Q_{1ei} = 10^4$, $Q_{2ei} = 0$, $Q_{2ei} = 10^{-4}$ and $Q_{0i} = Q_i(h_i)$ as weighting matrices for the cost functions. We note that $Q_{0i}$ is chosen in order to keep constant the quadratic relation of the plant state, for each system $i$, throughout the finite-time horizon cost function (4).

We consider the overall iterative process described in Section IV-D, which converges independently of the initial guess used for $h_i$. As a matter of clarification, in the first run we started the algorithm using an equal distribution of the computational resource ($h_i^0 = 0.2414$), and for the subsequent ones the current value of the sampling period is used. The simulations are performed using the toolbox TrueTime [4].

As specified in the procedure presented in Section IV-C, the Nash bargaining strategy requires the designer to set a disagreement point $d$. In our case, we arbitrarily choose $d_1 = d_2 = 3.5 \times 10^4$, which has lead to the weighting parameters and sampling periods shown in Table I. The corresponding Pareto frontiers and Nash bargaining solutions are shown in Figure 2, and the plants outputs in Figure 3.

We may notice that, at the beginning of the simulation, as we have a larger difference between both plant outputs, the solution found through the sampling period assignment procedure gives a relative preference to the system that is farther from the equilibrium point. On the second RM run, as both system trajectories are close to their respective equilibrium, the solution of the bargaining tends to an equal distribution of the shared resource. Like in the first run, at $k_r = 3.0s$ we have again one system output that is farther from equilibrium than the other one, thus the resource manager accordingly assigns new sampling periods to deal with the disturbance.

<table>
<thead>
<tr>
<th>TABLE I</th>
<th>SAMPLING PERIOD SEQUENCE FOR $d_1 = d_2 = 3.5 \times 10^4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k_0$, (0.0s)</td>
<td>(0.3615, 0.4100) (0.6385, 0.1711)</td>
</tr>
<tr>
<td>$k_1$, (1.5s)</td>
<td>(0.5004, 0.2405) (0.4996, 0.2423)</td>
</tr>
<tr>
<td>$k_2$, (3.0s)</td>
<td>(0.5196, 0.2116) (0.4804, 0.2811)</td>
</tr>
<tr>
<td>$k_3$, (4.5s)</td>
<td>(0.5001, 0.2411) (0.4999, 0.2417)</td>
</tr>
</tbody>
</table>

Fig. 2. Pareto frontiers and Nash bargaining solution
A. On the choice of the disagreement point

As an alternative to a fixed disagreement point as used before, we studied two alternatives to dynamically compute the point $d$ at each RM run. The first one uses disagreement weights $\beta_i$, that lead to $d_i(\beta_i) = J_i(h_i(\beta_i))$. Notice that these disagreement weights do not belong to a simplex, actually $\Sigma_{i=1}^p \beta_i < 1$. The second one uses disagreement sampling periods $h_i$, that lead to $d_i(h_i) = J_i(h_i)$. Both choices ensure that $d$ is computed as a disagreement cost function. We considered the same simulation conditions used previously.

Let us consider disagreement weights $\bar{\beta}_1 = 0.05$, equal for both plants. This guarantees that there exist $J_i(h) \leq d_i(\beta_i)$. The results obtained through the simulation are shown in Table II. Notice that using this strategy the weighting parameters are constantly chosen as $\bar{\beta}_1 = \bar{\beta}_2$, which is similar to the choice in [11].

Consider now disagreement sampling periods $\bar{h}_i = 1.5s$, again equal for both plants. The obtained weighting parameters and sampling periods are shown in Table III. In this case, we obtain an equal distribution of the available computational resource for all runs of the RM, independently of the current trajectories of the plants.

VI. CONCLUSION

The online dynamic sampling period assignment problem, where a set of control tasks runs on the same processor, has been studied. A global weighted cost function was used, providing an additional degree of freedom. Although the weights may be not trivial to define, when accordingly used they can lead to an overall performance that also takes into account a suitable individual performance for each system. In order to obtain a single Pareto efficient solution, a strategy issued from a Nash bargaining game was presented, making use of an iterative procedure. An example has been proposed to illustrate the proposed approach.

REFERENCES