# Probabilistically Certified Management of Data Centers Using Predictive Control

A. Daniel Carnerero, Daniel R. Ramirez, Teodoro Alamo and Daniel Limon

*Abstract*—**Data centers are facilities with large number of servers providing cloud services. The increasing number of data centers in use along the last years has generated environmental concern due to the immense amounts of energy consumed by them. This also includes some auxiliary services such as the cooling equipment which is known to be very costly. For that reason, efficient data center strategies are needed in order to provide an acceptable Quality of Service (QoS) and suitable temperature for every server while using the least amount of resources possible. This paper presents some strategies to deal with the unified workload and temperature problem that appears in the data center. As the system is modeled as a queue and the control variables have an hybrid nature, some highly parallelizable particle based optimization algorithms are proposed to solve the optimization problem. Numerical simulations are provided in order to illustrate the effectiveness of the strategy. These simulations also show the improvements obtained from the GPU computing. Finally, a probabilistic evaluation approach is developed in order to provide certificates on the probability of constraint satisfaction without increasing the computational burden of the online problem.**

*Note to Practitioners*—**This paper addresses the problem of deciding in real-time the number of active servers in a data center that is required to meet the Quality of Service (QoS) demands while keeping energy consumption at a minimum. The temperature set point of the cooling equipment must be also taken into account, as it is advisable to use the minimum cooling that keeps the servers running in safe conditions. The management strategy proposed is based on Predictive Control. In this way, the number of active servers and temperature set point will be chosen so that the future energy consumption is minimized while guaranteeing that QoS and safety demands are met under different possible operating conditions. Furthermore, the proposed management strategy can be tuned depending on the QoS that it is desirable to provide. The proposed strategy will lead to energy-consumption improvements while having guarantees on the data center performance.**

*Index Terms*—**Model Predictive Control, Constrained Optimization, Data Center, Energy Efficiency, Particle based Algorithms, Probabilistic Evaluation.**

## I. Introduction

Data centers are facilities composed by a large amount of servers and the associated support infrastructure. The variety of tasks that can be carried out by these infrastructures, such as batch and interactive computation, web portals, etc [1] have led to a sustained growing of the number of active data centers in the last decades [2]. Some studies predict that data centers within the U.S. will consume around 3000 TWh electricity by 2030 [3]. In addition to the economic burden derived from the electricity prices, this extremely high power consumption also leads to serious concerns due to environmental issues [4]. Thus, improvements in the energy efficiency of data center management are critical for a sustainable society.

The energy consumption of a data center can be broadly associated to Information Technology (IT) equipment (i.e. servers, storage) and infrastructure facilities, mainly the cooling equipment [5], [6]. The total amount of energy consumed will depend on both the design and the efficiency of the policies used to manage these facilities. An integrated data center management must consider not only the thermal constraints of the IoT equipments and the total energy consumption but also keep the performance near the required levels at all times. Thus, the way in which both the IT and cooling equipments are used to operate the data center will have a great impact on the overall energy consumption.

Many works in the literature are focused on developing cooling policies to minimize the energy consumption within data centers using only linear thermal models [7]–[9]. In spite of being simple first order linear models, interesting improvements were reported. However, better solutions could be attained with a unified data center management. In this field, works are rather scarce. For example, the paper [1] proposes a control architecture where both thermal and tasks management are taken into account. The data center is posed as a linear first-order continuous system in its thermal component and in the computational part. Task arrival rates are considered deterministic, thus tasks arrive at fixed intervals. A QoS constraint is used to establish stability limits to the system. On the other hand, in [10], several control policies assuming that the maximum temperature of the servers are soft constraints are proposed. Other techniques available are based on workload distribution with some thermal aware criteria [11], [12]. Also, a solution to the problem under the assumption that the recirculation of hot air is at constant temperature can be found in [13].

Model Predictive Control [14] has also been applied to this problem. In [15], a scheduling methodology based on MPC and electricity prices is proposed in order to reduce the economic impact. On the other hand, [16] unifies the management with the cooling scheme, assuming implicitly that servers have no limits to be overclocked, which is rather unrealistic. Due to the complexity of the problem, in [17] it is provided an approximation algorithm focused on providing fast evaluations of the complex constraints that have to be taken into account. However, the models appearing in the literature, under different assumptions, usually evade the fact that the system is actually a queue model, simplifying drastically the problem at the expense of a less realistic modeling.

In addition, due to the stochastic nature of the data center,

Dep. de Ingeniería de Sistemas y Automática, Escuela Técnica Superior de Ingeniería,Universidad de Sevilla, Camino de los Descubrimientos s/n, 41020 Sevilla, Spain (e-mail: acarnerero,danirr,dlm,talamo@us.es).

some considerations should be done. In the case considered in this paper, the uncertainty comes from the random nature of the task arrival times and their workload. Since it is difficult to obtain guaranteed bounds for these uncertainties, robust MPC based on deterministic settings [18], [19] are hardly applicable or lead to very conservative results. Thus, strategies based on randomized settings [20] like chance constrained MPC [21] and the scenario approach [22], [23] are more suitable. The main limitation of these approaches is that the number of scenarios to be generated increases with the dimension of the problem, becoming an intractable problem in many cases. A possible solution to this problem is to consider a limited number of scenarios in the control law computation, resulting in a tractable online computational burden. Then, an offline probabilistic validation could be carried out. This would certify that the controller fulfills the state and control constraints with a specified probability and confidence [24]–[26]. However, this method only provides a “yes” or “no” answer if a specific controller fulfills these probabilistic guarantees, being hard to compare among all of them. Also, this could lead to non feasible solutions if none of the controllers satisfies the probabilistic constraint. For that reason, a probabilistic evaluation approach is proposed here. This allows us to compute the safety level of each one of the proposed controllers to study the trade-off between constraint satisfaction level and overall costs within the data center. Note that these controllers can be completely different between them or be the same controller with different tuning hyper-parameters (back-offs, number of scenarios, etc). That is, there is no restriction to the nature of the controllers to be considered. This probabilistic evaluation is carried out by means of closed loop simulations and does not depend on the dimension of the problem, making the problem itself tractable.

This paper proposes an MPC framework for the optimization of cold aisle data centers. The optimization objective is to guarantee a certain QoS to the users and keep a suitable temperature of the servers while consuming the least amount of energy possible. The data center is modelled as a queue system where the arriving tasks can be computed by multiple servers at the same time. Although it moves away from the usual M/M/c queue and makes the treatment more complicated, it allows more generality within the model, where we can take into account some specific kind of data centers like render farms [27]. Unlike other works available in the literature, the full queue model is used to predict the evolution of the system through the horizon, thus achieving a more realistic modeling than that obtained from linear models. Random arrival rates and workloads will be managed through scenarios, adding more complexity to the model and thus better reflecting the working conditions of a real data center. The QoS is managed by imposing a hard constraint on the number of powered servers in the optimization problem, thus guaranteeing that tasks are executed within the pre-specified time limits. Although there exist very optimized mixed-integer solvers nowadays such as Gurobi [28], they are not suitable for the related optimization problem. For that reason, particle algorithms were chosen to deal with the optimization problem [29]. These algorithms are highly parallelizable and can obtain important speed ups with a GPU implementation [30]. This allows us to solve the optimization problem within the sampling time. An offline probabilistic evaluation scheme based on closed-loop simulations is developed and it is used to provide robustness to the proposed controllers.

The paper is organized as follows. In Section II, the model of the data center is presented. Section III shows the proposed controllers together with the particle based optimization algorithms. In Section IV, the probabilistic evaluation scheme is presented. Section V shows the simulation results of the proposed strategies. Finally, in Section VI, conclusions and some future works are presented.

## II. Data Center Model Description

Data centers are composed of thermally-isolated units in which server racks are allocated typically following a Cold Aisle (CA) structure as shown in figure 1. An important feature of this arrangement is that air flow is separated into two different flows, being the first a cold one which reaches every server as it is blown from below the floor by means of suitable built-in fans. This cold flow is provided by a Computer Room Air Conditioning (CRAC) unit. The cold air travels through the servers and gets heated. This hot air returns to the CRAC unit through the ceiling. It should be noted that servers are isolated from the hot air to avoid further heating.

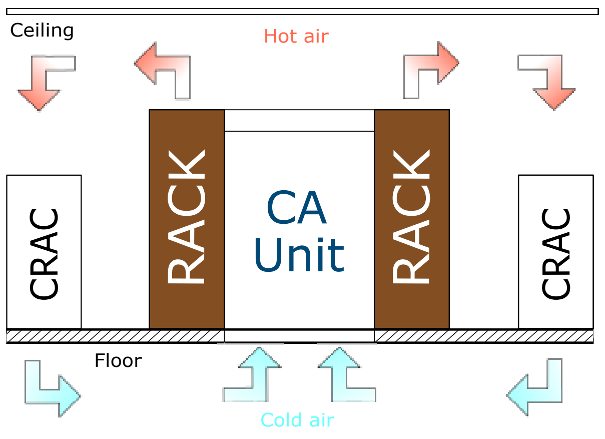

Fig. 1. Scheme of a cold aisle data center structure.

The CRAC unit is necessary to keep the server temperatures below certain security levels in order to ensure the servers reliability. As almost all power consumed by servers is dissipated as heat, CRAC operation is essential, but also very costly because of the high number of servers involved in a typical data center operation [5]. Thus any measure aimed to achieve a more efficient thermal management of the data center will have a great impact not only in operating costs and environmental impact, but also in the QoS provided. In the following, a discretized model of the data center dynamics with an integration time step of $t_s$ is presented. The discrete time unit will be denoted as $k$.

### A. Tasks Model

In this section, a description of a data center operation based on a queue model [10] is presented. The data center has $M$

available servers, although $m$, the number of booted servers, can be lower. In this way, the number of servers working in a specific moment can be manipulated according to the needs of the users (workload) and under efficiency and QoS constraints.

For the whole data center, there exists a unique queue where the task requests wait for their turn until they can be processed. As it is accepted in the literature [10], [16], the time between arrivals is assumed to follow an exponential distribution whose mean is $k_a$ and probability density function

$$f(k) = \frac{1}{k_a} e^{-\frac{k}{k_a}}. \tag{1}$$

Also, let $L(k)$ be the request rate at instant $k$. That is, the number of tasks that arrive to the data center at instant $k$. Due to the exponential nature of the time between arrivals, this can be modeled as a random variable following a Poisson distribution with mean $\frac{1}{k_a}$ and probability mass function

$$g(n) = \frac{1}{n!}\left(\frac{1}{k_a}\right)^n e^{-\frac{1}{k_a}}. \tag{2}$$

For an interval of length $k_u > 0$ , $k_u \in \mathbb{Z}$, the number of task arriving between $[k \quad k-k_u]$, is denoted $L_{k_u}(k)$ and the probability mass function is

$$g_{k_u}(n) = \frac{1}{n!}\left(\frac{k_u}{k_a}\right)^n e^{-\frac{k_u}{k_a}}, \tag{3}$$

with mean $\frac{k_u}{k_a}$.

It is also considered, as it holds in practice, that every task has a workload potentially different from another. Then, for a certain task $y$, the computational time $W_y$ required to accomplish it in a single server, is assumed to follow an exponential distribution like (1) with mean $\frac{1}{\mu}$. In this way, the workload $W(k)$ will be defined as the average workload of the $L(k)$ tasks that arrived at instant $k$. Similarly, $W_{k_u}(k)$ is the average workload of the $L_{k_u}(k)$ tasks that arrived between $k$ and $k-k_u$. The parameters $k_a$ and $\mu$ also define the minimum number of servers needed in the data center in order to prevent that the queue does not grow infinitely. Then, $M$ must satisfy the following condition

$$M \geq \frac{1}{k_a \mu}. \tag{4}$$

## B. Server Model

The variables needed to characterise the state $x_i(k)$ of a certain server $i$ at instant $k$ are the number of tasks currently running in the data center ($\alpha$), the number of tasks in the queue ($\beta$), the temperature of the cold air ($T_c$), the temperature of the server ($T_i$) and the time instant in which the server $i$ is turned on ($k_i$).

In order to determine the state of the whole data center, it suffices to include the remaining $T_i$ and $k_i$. Therefore, the complete situation $x$ will be

$$x^T(k) = \left[\alpha(k),\ \beta(k),\ T_c(k),\ T_1(k)\ \ldots\ T_M(k),\ k_1\ \ldots\ k_M\right] \tag{5}$$

Servers will switch between four possible working conditions:

- Off. The server does not draw any power.
- Booting. It is a transition from off to working or idle. It takes a fixed amount of time $k_{on}$ as a pure delay. While in this transition time, the server draws power at the same rate as the idle condition.
- Working. The server is "on" and processing a certain task. For simplicity, power is assumed to be drawn at a constant rate. In practice this is equivalent to assume that no CPU frequency scaling is used.
- Idle. The server is "on" but it is not processing any tasks. It draws less power than in the previous condition. However, the server is consuming energy for doing nothing.

As usual in the literature and also for simplicity reasons, the time needed to transition from "on" to "off" is assumed to be negligible.

The transition from working to idle and vice-versa is assumed to be instantaneous. Note that because of the transition time $k_{on}$ and the power drawn in the booting process, it could be advantageous to keep a server in idle state if it is expected to be needed in a near future. Furthermore, the transition from "off" to "on" is always immediately available (although it takes a $k_{on}$ time to be completed), but the reverse, that is, from "on" to "off", is done in a deferred way, because the server must be "on" until the current job is finished.

Taken into account the previous working conditions, denote by $u_i$ if the server $i$ is switched on ($u_i = 1$) or off ($u_i = 0$). Then, the power consumption of a server $i$ will be defined by the following

$$p_i(k, x_i, u_i) = \begin{cases} 0 & \text{if off } (u_i = 0) \\ a_2 & \text{if booting} \\ & (u_i = 1,\ \alpha \geq 0,\ k - k_i < k_{on}) \\ a_2 & \text{if idle} \\ & (u_i = 1,\ \alpha = 0,\ k - k_i \geq k_{on}) \\ a_1 + a_2 & \text{if working} \\ & (u_i = 1,\ \alpha > 0,\ k - k_i \geq k_{on}) \end{cases}$$

where $a_1$ is the marginal consumption and $a_2$ the minimum consumption.

The management approach researched in this paper does not deal with the task scheduling of the data center [31]. We assume that the distribution of the tasks among the servers follows a certain known policy. It is assumed that a task can be split among multiple servers (up to $M$). However, a single server can only work for the completion of a single task. These assumptions complicate the mathematical modelling of the data center as it will not work as a $M/M/c$ queue in which each task is scheduled to be executed in a single server, but it will result in a more general data center model. On the other hand, no server will be idle if there is at least one running task. This also implies that every idle server starts to work at the moment a task request arrives.

These server assignment policies imply, from an implementation point of view, the existence of a pool of running tasks no larger than the number of servers in "on" condition (i.e., "working" or "idle"). Once a task is ready to be processed (i.e., is at the front of the queue and the pool has one or more empty slots), it is assigned to at least one server and never leaves the pool until it is finished. That is, the remaining workload of each task is strictly decreasing.

Let $m(k)$ be the number of servers turned on at instant $k$ (i.e. working or idle). Then, the processing of a task is done in the following way. Assume that only a certain task $y$ with workload $W_y$ is within a pool of $m(k)$ servers ($\alpha(k) = 1$). The workload of the task $W_y$ is the number of "work packages" that have to be processed in order to complete the task. At instant $k$, $m(k)$ servers are assigned to this task (due to being the only task in the pool). Thus, every server will compute a "work package" resulting in $m(k)$ "work packages" executed. If there are no "work packages" left for the instant $k+1$, the task is completed and dropped from the pool. If the pool is full at a certain instant $k$ (i.e. $\alpha(k) = m(k)$), the assignation is trivial, that is, every task will be assigned a unique server. For the case where $1 < \alpha(k) < m(k)$, the tasks will be assigned to one server and the task with largest remaining time will be assigned to $m(k) - \alpha(k)$ servers. This makes sense because it will achieve better QoS.

## C. Thermal Model

The thermal model of each server is derived from the following thermal balance equation

$$K_t \frac{dT_i(t)}{dt} \;=\; c_p\, q_a(t)(T_c(t) - T_i(t)) + p_i(t)\,, \tag{6}$$

where $T_c$ and $q_a$ are the temperature and flow of the cold air provided by the CRAC unit, $T_i$ and $p_i$ are the temperature and power consumption of server $i$ respectively, $K_t$ the server thermal capacity and $c_p$ the air heat capacity. In a discrete setting with an integration step $t_s$, equation (6) turns into

$$T_i(k+1) \;=\; T_i(k) + \frac{t_s}{K_t}\left(c_p\, q_a(k)\left(T_c(k) - T_i(k)\right) + p_i(k)\right)\,. \tag{7}$$

In order to guarantee server reliability, it is necessary to keep the servers temperature under certain safety levels as stated by the following hard constraints

$$T_i \;\leq\; 80^\circ\text{C} \quad \forall i \in M\,. \tag{8}$$

From the variables that affect $T_i$ in (7), in this paper it will be considered controllable $p_i$ (through the state of the server) and $T_c$ which is assumed to be regulated by the set point $T_r$, that is, the desired temperature of the cold air provided by the CRAC. Thus $T_c$ will follow $T_r$ with a closed loop dynamics that it is assumed to be first order with a certain time constant $\tau$ and unity gain

$$\tau \frac{dT_c(t)}{dt} = T_r(t) - T_c(t)\,. \tag{9}$$

Same as before, equation (9) turns into

$$T_c(k+1) \;=\; T_c(k) + \frac{t_s}{\tau}\left(T_r(k) - T_c(k)\right)\,. \tag{10}$$

On the other hand, the possible values of $T_r$ must satisfy

$$15^\circ\text{C} \;\leq\; T_r(t) \;\leq 25^\circ\text{C}\,. \tag{11}$$

Also, the CRAC unit coefficient of perfomance (CoP) will change depending on the cold air temperature ($T_c$). The CoP represents how expensive is to cool the inlet air till a certain temperature, meaning that at lower temperatures it will need more power consumption to reach it. As the CoP increases, the cost will decrease. It can be calculated from the following equation

$$\text{CoP}(T_c(k)) \;=\; 0.0068\, T_c(k)^2 + 0.0008\, T_c(k) + 0.458\,, \tag{12}$$

which is widely adopted in the literature [12]. Thus, let $\sum_{i=1}^{M} p_i(k, x_i, u_i)$ be the server power consumption at time instant $k$. The power consumption at the CRAC unit is

$$\frac{\sum_{i=1}^{M} p_i(k, x_i, u_i)}{\text{CoP}(T_c(k))}\,. \tag{13}$$

The total power consumption will be then the power drawn by the servers added to the CRAC power consumption, thus

$$\sum_{i=1}^{M} \left(1 + \frac{1}{\text{CoP}(T_c(k))}\right) p_i(k, x_i, u_i)\,. \tag{14}$$

## D. Quality of Service

The Quality of Service (QoS) of a task is defined as the time that is required to complete it since its arrival until its completion. It includes the wait time in the queue and the execution time once it is within pool. It is necessary to guarantee the data center clients that this time will be lower than an agreed one.

If the tasks were assigned to just one server, like in an $M/M/c$ queue (or Erlang-C model), the mean service time would be

$$t_c(k) \;=\; \frac{1}{\frac{m(k)}{W(k)} - L(k)}\,. \tag{15}$$

This measurement would be used in practice with estimations of $W(k)$ and $L(k)$, denoted as $\hat{W}(k)$ and $\hat{L}(k)$ respectively. However, in the proposed approach, assuming that a task can be executed concurrently by a number of servers, the aforementioned measure is used to provide an upper bound of the mean service time since in this case it only can be lower or equal

$$t_{sv}(k) \;\leq\; \frac{1}{\frac{m(k)}{\hat{W}(k)} - \hat{L}(k)} \tag{16}$$

where $\hat{W}(k)$ is the expected value of $W(k)$ at instant $k$ for the next $\hat{L}(k)$ requests and $\hat{L}(k)$ is the estimation of $L(k)$ at instant $k$. We consider that the QoS is satisfied at instant $k$ if the mean service time is no larger than a specified value $D$. Thus, the QoS constraint is satisfied if the following condition holds

$$\frac{1}{\frac{m(k)}{\hat{W}(k)} - \hat{L}(k)} \;\leq\; D$$

which implies that

$$m(k) \geq \hat{W}(k)\left(\frac{1}{D} + \hat{L}(k)\right)\,. \tag{17}$$

The term on the right represents the minimum number of servers to fulfill the QoS constraint with a certain $\hat{L}$, $\hat{W}$ and $D$. This can be written as

$$m_D(k) = \hat{W}(k)\left(\frac{1}{D} + \hat{L}(k)\right). \tag{18}$$

## III. Data center management approach

In this paper, an optimal management policy inspired on predictive control strategies is proposed. This controller determines the number of servers that are to be on or off and the set point temperature of the CRAC machines. The criteria to decide the optimal values for those variables will be the energy consumption and the control effort of the inputs (i.e. server switching and temperature reference changes) subject to thermal and QoS constraints.

First of all, we assume that the dynamics of the queue, task arrivals, etc. run much faster than the process of switching on a server (i.e. $k_{on} \gg 1$). With such a large delay in the control actions, the control decisions have to be made in a superior time scale and separate them over time. This leads us to a scheme where the predictive controller is not executed at every instant $k$ but it is executed at every instant $k_m t_s$ where $k_m$ is the number of instants between the controller execution. Thus, the sample time of the controller is $k_m t_s$. In order to avoid switching off a server that is still booting (that is, never reached the "on" state and did nothing but consuming power), we take a sampling larger than the switching on time, i.e. $k_m \geq k_{on}$.

Based on the model, the expected evolution of the data center can be estimated and then, we can associate a predicted cost to a sequence of candidate future control inputs. For clarity purposes, we denote $\ell_j = k + jk_m$. This will work as the time scale of the MPC controller. In this paper, we propose the following predicted cost function to measure the expected performance of the data center:

$$V(\ell_0, \mathbf{x}, \mathbf{u}, \mathbf{T_r}) = \sum_{j=0}^{N_p}\sum_{i=0}^{M}\left(1 + \frac{1}{\text{CoP}(T_c(\ell_j))}\right) p_i(\ell_j, x_i, u_i)$$
$$+ \lambda_u \sum_{j=0}^{N_c} |\Delta u(\ell_j)| + \lambda_{T_r} \sum_{j=0}^{N_c} |\Delta T_r(\ell_j)|, \tag{19}$$

where $N_p$ and $N_c$ are the prediction and control horizons, $\mathbf{x}$ is the sequence of $x(\ell_j)$ over the prediction horizon, $\mathbf{u}$ and $\mathbf{T_r}$ are the sequences of "on"-"off" control actions of all servers and temperature set points through the control horizon respectively, $\Delta u(\ell_j)$ is the total number of commutations to either "on" or "off" at instant $\ell_j$, $\Delta T_r(\ell_j)$ is the increment in $T_r(\ell_j)$, $\lambda_u$ is a term weighting $\Delta u(\ell_j)$ and $\lambda_{T_r}$ is a term weighting $\Delta T_r(\ell_j)$. Also, the control actions further from the control horizon are considered to remain constant.

In order to derive the proposed controller, it is necessary to determine a prediction model such that for a given state at time $\ell_j$, $x(\ell_j)$ and for given control actions $u(\ell_j)$ and $T_r(\ell_j)$ (that will remain constant throughout the sampling time $k_m$), the state of the data center predicted at the next sampling time $\hat{x}(\ell_{j+1})$ is calculated depending on the estimation of the number of tasks and their workload $\hat{L}_{km}(\ell_{j+1})$ and $\hat{W}_{km}(\ell_{j+1})$. This prediction model can be posed as:

$$\hat{x}(\ell_{j+1}) = f(x(\ell_j),\, u(\ell_j),\, T_r(\ell_j), \hat{L}_{k_m}(\ell_{j+1}),\, \hat{W}_{k_m}(\ell_{j+1})), \tag{20}$$

being $f(\cdot)$ the function that compute the following state given the previous one, the inputs and the realisations of $L_{k_m}$ and $W_{k_m}$. Note that the function $f(\cdot)$ must compute all events happening thorough the interval $k_m$ for every instant $k$ in order to be able to return the state at the following sample time.

The optimal predicted number of servers and set point temperatures for the CRAC will be then computed as the solution of the optimization problem

$$\begin{aligned}
\min_{m(\ell_j), T_r(\ell_j)} \quad & V(\ell_0, \mathbf{x}, \mathbf{u}, \mathbf{T_r}) \\
\text{s. t.} \quad & \hat{x}(\ell_{j+1}) = f(x(\ell_j), u(\ell_j), \\
& T_r(\ell_j),\, \hat{L}_{k_m}(\ell_{j+1}),\, \hat{W}_{k_m}(\ell_{j+1})) \\
& m(\ell_j) \in [1,\, M] \quad \forall j \in [0, N_c] \\
& u(\ell_j) = \text{switch}(m(\ell_j)) \quad \forall j \in [0, N_c] \\
& 15°C \leq T_r(\ell_j) \leq 25°C \quad \forall j \in [0, N_c] \\
& T_i(\ell_j) \leq 80°\text{C} \quad \forall i \in M \quad \forall j \in [1, N_p] \\
& m(\ell_j) \geq m_D(\ell_j) \quad \forall j \in [1, N_p],
\end{aligned} \tag{21}$$

where $\text{switch}(\cdot)$ represents the policy to select which servers are to be switched on. In this paper, a simple scheme is proposed for such policy. That is, for a given number of servers, the first $m(k)$ will be switched on.

As it is customary in predictive controllers, the solution of (21) is applied in a receding horizon manner, meaning that only the control actions and temperature set points are actually applied (i.e., $u_i(\ell_0)$ and $T_r(\ell_0)$) while the remaining decision variables $(u_i(\ell_1), \ldots, u_i(\ell_{N_c-1})$ and $T_r(\ell_1), \ldots, T_r(\ell_{N_c-1}))$ computed at time $k$ are discarded. The optimization of (21) is then repeated at each sampling time so that the decision variables to be applied are computed using the real state of the data center at that sampling time.

It should be noted that in this optimization problem some of the variables are integer (number of servers on) whereas other are real valued (the set-point temperatures). This, together with the complexity of the data center model motivates the use of specialized optimization algorithms to solve (21), such as a particle based optimization technique [29], [30] that will be exposed in the following subsection.

### A. Particle based optimization

In these techniques of iterative nature, a set of possible candidate solutions (called particles) are evaluated at each iteration and used to generate a new candidate solution set that may be nearer to the solution of the optimization problem. Here, the evaluation of each particle will be done by means of simulations that will be used to assess the performance of each candidate solution. The main ingredients of the proposed technique are:

- *Particles* are candidate solutions of the optimization problem. That is, a sequence of control actions over the control horizon. At higher problem complexity (i.e., more servers and longer control horizons), more particles are needed in order to obtain a solution close to the optimal one [32].

- *Weights.* Particles have associated a weight that represents how good is a particle compared with the others. In this work, the weight is based on the performance cost of each particle computed by means of a computer simulation of the data center model. These simulations predict the evolution of the data center along the prediction horizon if the decision variables are those of the particles. Once the simulations for all particles are completed (this can be done in parallel), a scaling of the performance costs of the particles is made. Let $V_{max}$ and $V_{min}$ be the maximum and minimum costs attained on the set of particles under evaluation. If a given particle $z \triangleq (\mathbf{m}(\cdot), \mathbf{T_r}(\cdot))$ has a performance cost $V_z$, then the scaled cost will be
$$\tilde{V}_z = \frac{V_z - V_{min}}{V_{max} - V_{min}} \tag{22}$$
which guarantees that $\tilde{V}_z \in [0, 1]$. Then the weight of the particle $z$ will be defined as
$$\sigma_z = 1 - \tilde{V}_z, \tag{23}$$
meaning that particles with higher costs will have lower weights and vice versa. This is very important for the resampling phase.
- Feasibility checking. Those particles that do not satisfy the constraints in (21) will be assigned a zero weight so that they cannot be selected in the resampling stage.
- *Resampling* is the step where a new generation of particles is created based on the performance (weights) at the previous iteration. This is done by means of the Kitagawa resampling algorithm with stratified scheme [33]. The aim of this method is to form groups of particles with good performance in different areas of the feasible set of solutions. In this way, the risk of getting stuck at a local minimum is reduced because the algorithm considers all particles and not only the best one. Particles with higher weights are more likely to be resampled at next iteration (i.e., selected to be included in the next particle set).
- *Perturbation.* This process is inherently connected to the previous one. Once particles are resampled, it is necessary to "move" them along the local search space in order to discover new feasible solutions with potentially lower costs. In this work, the perturbation is made by adding a gaussian white noise. Figure 2 shows the result of the resampling and perturbation steps in a 2 degree of freedom minimization example. At first, there exists a set of random-generated particles. At next iteration, after resampling and perturbation, particles move towards better possible solutions according to the costs obtained previously. In our case, the optimization variables are integer, thus this process ends rounding to the nearest integer.
- Reseeding. The initial set of particles should be generated in a random manner. However, for the subsequent sample times, the reseeding can be done by exploiting the receding horizon nature of the predictive controller. Then consider the solution of (21) for a given sampling time $k$
$$\mathbf{m}^*(\ell_0) = \begin{bmatrix} m^*(\ell_0 \,|\, k) \\ m^*(\ell_1 \,|\, k) \\ \vdots \\ m^*(\ell_{N_c-1} \,|\, k) \end{bmatrix},$$
$$\mathbf{T_r}^*(\ell_0) = \begin{bmatrix} T_r^*(\ell_0 \,|\, k) \\ T_r^*(\ell_1 \,|\, k) \\ \vdots \\ T_r^*(\ell_{N_c-1} \,|\, k) \end{bmatrix}.$$
Thus, a particle generated as the shifted version of the solution for $\ell_0$
$$\mathbf{m}(\ell_1) = \begin{bmatrix} m^*(\ell_1 \,|\, k) \\ \vdots \\ m^*(\ell_{N_c-1} \,|\, k) \\ m^*(\ell_{N_c-1} \,|\, k) \end{bmatrix},$$
$$\mathbf{T_r}(\ell_1) = \begin{bmatrix} T_r^*(\ell_1 \,|\, k) \\ \vdots \\ T_r^*(\ell_{N_c-1} \,|\, k) \\ T_r^*(\ell_{N_c-1} \,|\, k) \end{bmatrix}.$$
would be a good candidate solution for (21) in $\ell_1$ although it may not be optimal. Note that the last control action is repeated. Thus, the reseeding scheme is based on including in the initial particle set some shifted versions of the best particles (those with greater weight) of the final set obtained in the previous sampling time. These reseed particles will not be perturbed so that they can not be lost in the resampling process. Furthermore, in the resampling step, the reseed particles will lead to more particles close to them, and those may attain a higher weight. Note that the fact that a set of reseed particles is used instead of just one reduces the probability of getting stuck in a local minima.

The particle algorithm is shown in Algorithm 1. According to the receding horizon approach described earlier in section III, Algorithm 1 is executed at each sampling time, yielding as a result a sequence of number of servers to be on and set point temperatures.

The stop condition can be just a fixed number of iterations or a more elaborated scheme such as a small improving rate of the cost of the best particle at each iteration.

*1) Prediction of the future values of $L_{k_m}$ and $W_{k_m}$:* In the proposed strategy, the QoS constraint is evaluated for time intervals of length $k_m$, thus it is needed to estimate $L_{k_m}$ and $W_{k_m}$. It is assumed within the controller that tasks will arrive uniformly every $k_a$ time units (that is, the mean time of the exponential distribution (1)). On the other hand, the workload of these incoming tasks will also be assumed to be the expected value of the corresponding exponential probability distribution, that is, $\frac{1}{\mu}$. These values will be used by the controller to evaluate the QoS constraint.

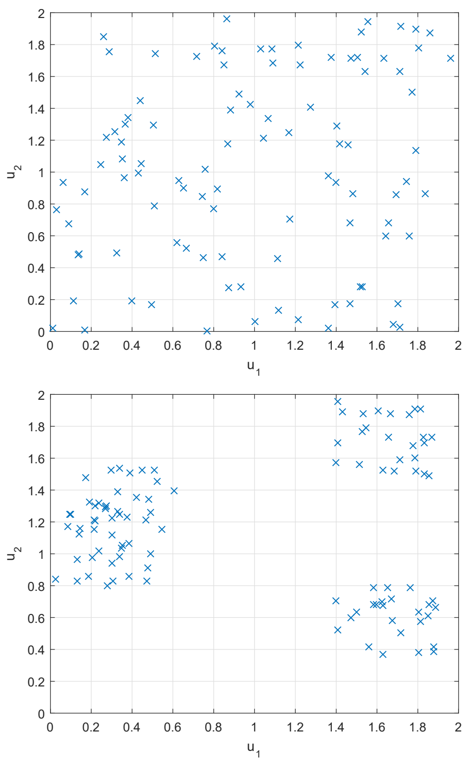

Fig. 2. Optimization example with two real decision variables.

```
Algorithm 1: Particle based optimization.
  Data: $\chi$ (Number of particles), $N_p$ , $N_c$ (Prediction
        and Control Horizon respectively), $V(\cdot,\cdot)$
        (Cost Function), $\chi_t$ (Number of particles to
        reseed).
  Result: Proposed control sequence $\mathbf{m}, \mathbf{T_r}$.
1 Reseed $\chi_t$ particles that showed best performance at
    previous time step;
2 Generate randomly new control actions for $\chi - \chi_t$
    particles;
3 Estimate $\hat{L}_{k_m}$ and $\hat{W}_{k_m}$;
4 repeat
5     Predict the system evolution for each particle
        along the prediction horizon;
6     Verify the feasibility of each particle and set
        weight zero for the unfeasible ones;
7     Calculate cost and weight for each particle;
8     Resampling process considering all particles;
9     Perturbation of particles excluding the $\chi_t$ reseed
        particles;
10 until a stop condition is fulfilled;
11 Choose as result the particle with greatest weight;
```

## B. Scenario based optimization

Another scheme for the estimation of $L_{k_m}$ and $W_{k_m}$ is proposed at the expense of a heavier computation burden. As the probability distribution of $L_{k_m}$ and $W_{k_m}$ are assumed to be known, it is possible to draw random sequences of those variables which will be called scenarios. These scenarios represent hypothetical realizations of the variables in the future and, as such, will be considered in the computation of the optimal number of servers and temperature set points. Algorithm 2 shows how to consider the scenarios.

```
Algorithm 2: Scenario PbO.
  Data: $\chi$ (Number of particles), $N_p$ , $N_c$ (Prediction
        and Control Horizon respectively), $V(\cdot,\cdot)$
        (Cost Function), $\chi_t$ (Number of particles to
        reseed), S (Number of scenarios).
  Result: Best control sequence $\mathbf{m}^*, \mathbf{T_r}^*$.
1 Reseed $\chi_t$ particles that showed best performance at
    previous time step;
2 Generate randomly new control actions for $\chi - \chi_t$
    particles;
3 Set the weights to an initial value $\frac{1}{\chi}$;
4 repeat
5     for every scenario do
6         Extract a sequence of arrivals and workloads
            from the distributions;
7         Estimate $\hat{L}$ and $\hat{W}$;
8         Predict the system evolution for each particle
            along the prediction horizon;
9         Verify the feasibility of each particle and set
            weight zero for the unfeasible ones;
10        Calculate cost and update weight for each
            particle;
11    end
12    Scale particle weights;
13    Resampling process considering all particles;
14    Perturbation of particles excluding the $\chi_t$ reseed
        particles;
15    Reset the values of the weights to $\frac{1}{\chi}$;
16 until a stop condition is fulfilled;
17 Choose as result the particle with greatest weight;
```

The main change from Algorithm 1 is the addition of a "for" loop over all the scenarios. As the first step in each iteration of this "for" loop, each scenario is drawn according to the probability distributions of $L$ and $W$. Once this has been done, the evolution of the system is predicted like in algorithm 1. The same goes for the feasibility verification of the particles.

There are also several changes about the costs and weights computation and meaning. In algorithm 1, the weight was assigned directly once the cost was calculated applying equation (23). This could be done because there was only one scenario (the expected values of the random variables). Now, it is necessary that weights show the performance of each particle for all scenarios. We chose to give them an initial value of $\frac{1}{\chi}$ and update them at each iteration of the "for" loop with the information obtained from the scenario considered at that iteration. The particle with the best cost will maintain its weight meanwhile the rest will decrease theirs according to how good their performance were. This is done by means of the following equation:

$$\sigma_z \leftarrow \sigma_z(1 - \tilde{V}_z)\,. \tag{24}$$

The weights are now updated as the product of the old weight and the new one computed from the cost in a specific scenario. At the end of the "for" loop, it is required by the Kitagawa resampling that the sum of the weights must be equal to one. This is because the weights represent a discrete probability density function. For that reason, the weights are scaled in line 12. This is done using the following equation:

$$\sigma_z \leftarrow \frac{\sigma_z}{\sum_{z=1}^{\chi} \sigma_z} . \tag{25}$$

Also, it is important to reset the values at line 15 due to the new weight updating from equation (24).

Taking into account different realisations of the random variables will turn out to be a more conservative solution of the initially proposed algorithm because of step 9. It means that for a certain sequence of arrivals and workloads, particles which do not fulfill the conditions on maximum temperature and QoS will not be resampled at next iteration. That is, they will disappear. In algorithm 1, this condition was checked only once for the sequence obtained from the mean values of the distributions. However, in algorithm 2, it is necessary to fulfill this condition in every scenario (which is more restrictive).

## C. Parallel implementation using GPU computations

Since its initial release in 2007, CUDA has become the first programming platform that allowed general purpose computing on GPUs [34] and it has still better performance than other interfaces such as OpenCL [35]. The main advantage of GPU computing is the decrease of the computation time [36] provided that the tasks considered fit in the CUDA programming model which is based on a Single Instruction Multiple Thread (SIMT) scheme [37].

Because of its nature, the particle algorithms proposed are highly parallelizable in some of their steps such as the prediction of the system evolution, feasibility verification, cost calculation and weight updating. Others can not be completely parallelizable (i.e. the resampling process) due to the nature of the Kitagawa resampling [33]. The reason of this is that the computation of the cumulative probability density function requires that a certain thread $z$ knows the value of the thread $z-1$ to do its job. Thus this resampling must be done in a pure sequential manner.

For simplicity, the simulations presented in this section have been coded in CUDA C kernels called from Matlab scripts. Although not the most efficient way of using GPU computing with CUDA, it allows an easy integration with Matlab CPU code.

## IV. Probabilistic Evaluation

Once reached to this point, we notice that the algorithms proposed in the previous sections have some design parameters that have to be chosen (i.e. $N_c$, the control horizon and $S$, number of scenarios). Depending on these parameters, a different controller will be obtained. In this section, we propose a method to compute the constraint satisfaction level for a given controller in order to be able to compare their performance. This is in general an unmanageable problem under the presence of unbounded uncertainty and non linearity within the model [38]. For example, in a statistical learning theory approach, the number of samples to be considered can be very high when the accuracy parameter is relatively small [39], [40], making the online problem intractable in some cases.

An alternative would be to use an offline probabilistic *validation* scheme [24] that reduces the online computational burden required to endow the controller with probabilistic guarantees. Those guarantees imply that the controllers does not violate the imposed constraints with a pre-specified probability and confidence.

However, in this case, we opt for a probabilistic *evaluation* scheme of our controllers instead. The main difference is that the probabilistic validation scheme only gives a "yes" or "no" answer to a certain controller if, with a pre-specified probability and confidence, the controller achieves the desired performance, in this case the constraint satisfaction. On the other hand, the proposed probabilistic evaluation scheme computes a different constraint satisfaction level for each controller given the desired confidence level $1-\delta$, which would let us better compare the results.

For the sake of simplicity, only the QoS constraint will be considered in the following, although it can be easily extended to the temperature constraint as it is done in the numerical results.

Let $Y_i$ be the number of tasks that a certain controller $i$ must complete and $s_i \in [0, Y_i]$ be a random variable showing the total number of QoS violations for that controller. Then, an empirical violation rate of the controller $i$ for a certain number of completed tasks $Y_i$ will be defined as

$$\frac{\bar{s}_i}{Y_i} = \frac{\text{QoS violations}}{\text{Total number of tasks}} .$$

*Definition 4.1:* Failure. Assuming that the real violation rate is greater than a certain quantity $\rho_i + \Delta\rho_i$, with $\rho_i \in [0,1)$, $\Delta\rho_i > 0$, a failure is considered to occur when an empirical violation rate of the controller is lower or equal than $\rho_i$.
That is, a *failure* happens when the empirical violation rate leads to misleading results. The following theorem proves that, with a properly chosen $\Delta\rho_i$, failures occur with a probability lower than a small confident parameter $\delta$. This definition of failure is slightly different than the one presented, for example, in [39]. Denoting $E(\cdot)$ as the expectation operator, the following theorem summarizes the result of this section.

*Theorem 4.1 (Bound on the empirical constraint satisfaction level):* Assume that

$$E\left(\frac{s_i}{Y_i}\right) > \rho_i + \Delta\rho_i ,$$

with $\rho_i \in [0,1]$ and $\Delta\rho_i$ satisfying

$$\Delta\rho_i \geq \frac{1}{Y_i}\log\frac{1}{\delta} + 2\sqrt{\frac{\rho_i}{Y_i}\log\frac{1}{\delta}} . \tag{26}$$

Then,

$$\Pr\left\{\frac{\bar{s}_i}{Y_i} \leq \rho_i\right\} \leq \delta ,$$

where $\bar{s_i}$ is an empirical value of $s_i$. This implies that the real constraint satisfaction level is bounded by $\rho_i + \Delta\rho_i$ with probability greater than $1-\delta$.

*Proof.* We are interested in upper bounding $\Pr\left\{\frac{\bar{s_i}}{Y_i} \leq \rho_i\right\}$ under the assumption that $E\left(\frac{s_i}{Y_i}\right) > \rho_i + \Delta\rho_i$. Since the number of empirical violations grows with $\mathrm{E}\left(\frac{s_i}{Y_i}\right)$, we have that this probability has a maximum when $\mathrm{E}\left(\frac{s_i}{Y_i}\right) = \rho_i + \Delta\rho_i$. That is,

$$\Pr\left\{\frac{\bar{s_i}}{Y_i} \leq \rho_i \,|\, \mathrm{E}\left(\frac{s_i}{Y_i}\right) > \rho_i + \Delta\rho_i\right\} \leq$$
$$\Pr\left\{\frac{\bar{s_i}}{Y_i} \leq \rho_i \,|\, \mathrm{E}\left(\frac{s_i}{Y_i}\right) = \rho_i + \Delta\rho_i\right\} .$$

Once reached this point, we notice that the probability of QoS violation for a certain task and controller $i$ can be interpreted as a Bernoulli random variable. Therefore, the probability of having less than a certain number of violations for some number of trials can be expressed as the binomial tail.

On the other hand, consider a random variable $X$ following a binomial random distribution $B(N,p)$ and given a constant $x$, we have from Chernoff's bound [41], [42] that if $x \leq p$ then

$$\Pr\left\{\frac{\bar{X}}{N} \leq x \,|\, \mathrm{E}\left(\frac{X}{N}\right) = p\right\} \leq \exp\left(-N\varphi(x,p)\right)$$

where $\exp(\cdot)$ refers to the exponential function and

$$\varphi(x,p) = x\log\frac{x}{p} + (1-x)\log\frac{1-x}{1-p} .$$

The convergence rate provided by this bound is known to be tight from Cramér's theorem of large deviations when $N \to \infty$ ( [43], chapter 23 in [44]). It is easy to see that the above Chernoff's bound can be applied to our problem making $\bar{X} = \bar{s_i}$, $N = Y_i$ and $\Delta\rho_i > 0$, obtaining

$$\Pr\left\{\frac{\bar{s_i}}{Y_i} \leq \rho_i \,|\, \mathrm{E}\left(\frac{s_i}{Y_i}\right) = \rho_i + \Delta\rho_i\right\} \leq$$
$$\exp\left(-Y_i\varphi(\rho_i, \rho_i + \Delta\rho_i)\right) .$$

Thus, we can design this upper-bound to be lower than the confidence $\delta$ by imposing

$$\exp\left(-Y_i\varphi(\rho_i, \rho_i + \Delta\rho_i)\right) \leq \delta .$$

Taking logarithms in both sides and rearranging the terms, we obtain

$$\varphi(\rho_i, \rho_i + \Delta\rho_i) \geq \frac{1}{Y_i}\log\frac{1}{\delta} . \tag{27}$$

We have that $\Delta\rho_i$ is embedded within the function $\varphi(\rho_i, \rho_i + \Delta\rho_i)$ so, in order to be able to obtain a closed expression for $\Delta\rho_i$, we apply the bound presented by Okamoto (Lemma 2 in [42])

$$\varphi(\rho_i, \rho_i + \Delta\rho_i) \geq \left(\sqrt{\rho_i + \Delta\rho_i} - \sqrt{\rho_i}\right)^2 .$$

Thus, we obtain the following sufficient condition for equation (27)

$$\left(\sqrt{\rho_i + \Delta\rho_i} - \sqrt{\rho_i}\right)^2 \geq \frac{1}{Y_i}\log\frac{1}{\delta} .$$

For $\Delta\rho_i \geq 0$, this is equivalent to

$$\sqrt{\rho_i + \Delta\rho_i} \geq \sqrt{\rho_i} + \sqrt{\frac{1}{Y_i}\log\frac{1}{\delta}} ,$$

and thus

$$\rho_i + \Delta\rho_i \geq \left(\sqrt{\rho_i} + \sqrt{\frac{1}{Y_i}\log\frac{1}{\delta}}\right)^2 .$$

This is easily rewritten as

$$\Delta\rho_i \geq \frac{1}{Y_i}\log\frac{1}{\delta} + 2\sqrt{\frac{\rho_i}{Y_i}\log\frac{1}{\delta}} .$$

This completes the proof. ■

Using the previous results, the probabilistic evaluation scheme will be carried out using the following steps *for each controller* to be validated:

1) Determine a number of closed loop simulations ($H$). Note that higher $H$ implies higher $Y_i$ and thus lower $\Delta\rho_i$. In practice, $H$ should be chosen as the highest affordable value.
2) Run $H$ closed loop simulations with different values of the parameters in the probability distributions in the data center model to reflect different operating conditions.
3) Denote $Y_{i,H}$ as the number of tasks executed during the $H$ simulations for controller $i$ and $s_{i,H}$ as the number of QoS violations in the $Y_{i,H}$. Then, after running these simulations for every controller, $Y_{i,H}$ and $s_{i,H}$ are known. Then compute $\rho_i$ as
$$\rho_i = \frac{s_{i,H}}{Y_{i,H}} . \tag{28}$$
4) Once done that, $\Delta\rho_i$ can be calculated for each controller with equation (26) for the specified confidence $1-\delta$. In case that the resulting $\Delta\rho_i$ is considered too large, it is possible to repeat from step (1) with an increased number of simulations.
5) Choose the best controller according to the desired criteria. For example, minimum constraint violation performance, trade-off between constraint satisfaction level and operating costs, etc.

## V. Numerical Simulations

The proposed management schemes and the probabilistic validation method will be illustrated by means of examples. The improvements from the parallel implementation will be also shown.

Four controllers with different parameters will be compared by means of simulations. These simulations have a fixed time step of 2500 time units. A number of $H = 100$ simulations were enough to have a large number of tasks to apply the probabilistic evaluation method explained above. In each one of these simulations, to reflect different operating conditions, the variables $L$ and $W$ were given different mean values taking into account the maximum workload and number of tasks that a data center of $M = 25$ servers can handle with respect to equation (4). Note that the number of completed tasks $Y_i$ will depend on the values of $L$ and $W$.

On the other hand, the integration step is $t_s = 1s$, the sampling time of the controller $k_m = 100$ time units (i.e. $100s$) and the prediction horizon is set to $N_p = 10$ sampling times (i.e. 1000 time units). Also, the maximum temperature and QoS are set to $80°C$ and 50 time units respectively. On the other hand, the time constant of the CRAC unit is $\tau = 180$ time units. The values of the power consumption are $a_1 = 180\,W$ and $a_2 = 120\,W$. Other thermal parameters like the server thermal capacity and the product of the air heat capacity times the air flow are $K_t = 160\,\frac{J}{K}$ and $c_p\,q_a = 5\,\frac{W}{K}$. The weighting factors in (19) are $\lambda_u = 200$ and $\lambda_{T_r} = 100$. Finally, table I shows the parameters of the four controllers considered and the line pattern for every controller in the following figures.

| | | | |
|---|---|---|---|
| $C_1$ | S-Pbo, $N_c = 1$, $S = 25$ | Solid line | Orange |
| $C_2$ | S-Pbo, $N_c = 5$, $S = 25$ | Dotted line | Blue |
| $C_3$ | Pbo, $N_c = 1$ | Dashed-dotted line | Purple |
| $C_4$ | Pbo, $N_c = 5$ | Dashed line | Yellow |

TABLE I
CONTROLLERS CONSIDERED AND LINE PATTERNS. PBO REFERS TO ALGORITHM 1 WHEREAS S-PBO REFERS TO ALGORITHM 2

Figure 3 shows the mean queue length, number of "on" servers in and number of idle servers. It can be seen how the controller without scenarios and longest control horizon $N_c$ ($C_4$) has the largest mean queue length in the experiments. This implies longer waiting times and, thus, the elapsed time to be expected for a certain task with this controller will be higher. Taking into account multiple scenarios, as in $C_2$, results in a lower queue length. This come from the fact that considering multiple scenarios yields a more robust and conservative management. The other controllers ($C_1$ and $C_3$) shows the effect of working with minimal control horizons. In that case, the controller is forced to satisfy the constraints along a large prediction horizon with few degrees of freedom, resulting in a more conservative management in the case of $C_1$ that keeps the queue length very small at the cost of a greater power consumption and very similar to $C_4$ for $C_3$. That is, without taking into account scenarios, the operation of the controllers is practically not affected by the control horizon. A more conservative management also implies a higher number of "on" servers, as shown in the middle subplot, and also more idle servers (bottom subplot) when the real operating conditions are less demanding than the predicted ones. Having more servers "on" but idle also make sense in order to be able to deal with the unknown upcoming tasks, due to the delay time $k_{on}$ needed to get the servers active from the "off" condition.

Figure 4 shows the mean values of the terms of the cost function (19) for each controller. The top subplot shows the power consumption term in Watts ($W$). As it can be guessed, the controllers with larger number of active servers will have greater power consumption, due to the consumption of the servers themselves but also for the greater consumption of the CRAC unit (which following (12) is assumed proportional to that of the servers). The lower subplots are referred to the control efforts of the control actions. Here, it can be seen

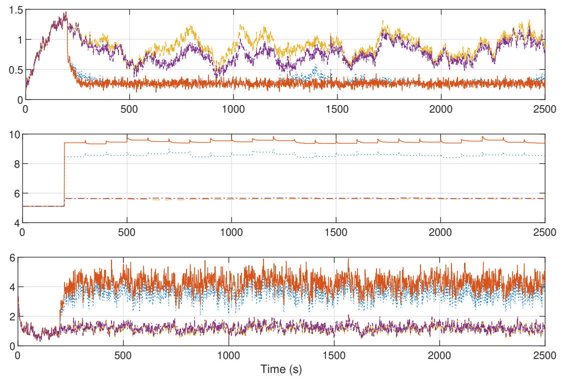

Fig. 3. From top to bottom, Mean Queue Length, Mean Number of On Servers and Mean Number of Idle Servers

that the controllers with $N_c = 5$ tend to change more their control actions which leads to greater control efforts, but also facilitates a more tuned application of the control action that results in lower overall costs as seen in Figure 5. In this figure, the instantaneous total cost for each controller is shown. As expected, the more conservative controllers are the most expensive ones in terms of performance cost.

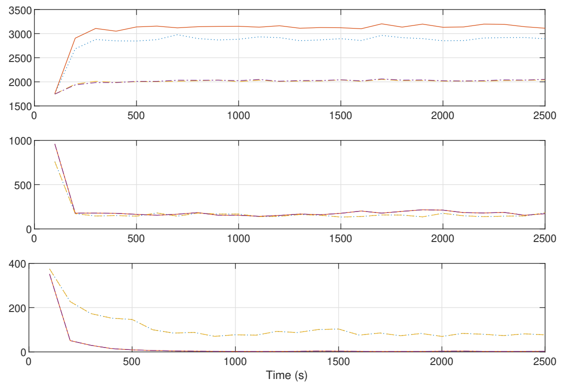

Fig. 4. From top to bottom: Mean Power Consumption, Mean Control Effort for booting servers and Mean Control Effort for changing the CRAC temperature reference

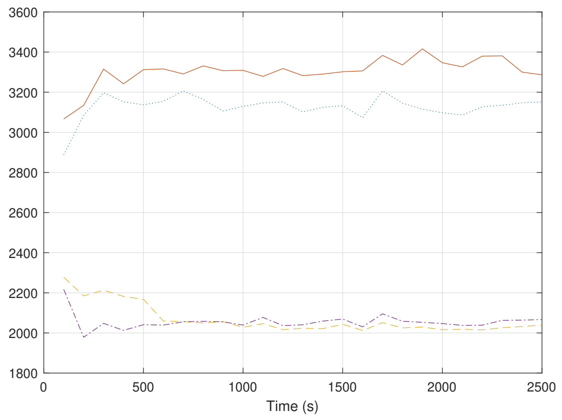

Fig. 5. Instantaneous total cost

### A. Probabilistic evaluation

Once all simulations are carried out, the probabilistic evaluation scheme proposed in the section IV can be applied to the four controllers. Table II shows the results of the probabilistic evaluation for the QoS constraint. The final result is shown in the $\rho_i+\Delta\rho_i$ column which is the upper bound (with confidence $1-\delta$) of the probability of not meeting the agreed service time $D$. Lower numbers imply better probabilistic guarantees, which as expected corresponds to the more conservative controllers ($C_1$ and $C_2$).

| | $\rho_i$ | $\rho_i+\Delta\rho_i$ | $Y_i$ |
|---|---|---|---|
| S-Pbo, $N_c=1$ | 0.0032 | 0.0050 | 65960 |
| S-Pbo, $N_c=5$ | 0.0040 | 0.0060 | 65926 |
| Pbo, $N_c=1$ | 0.0278 | 0.0328 | 65708 |
| Pbo, $N_c=5$ | 0.0337 | 0.0392 | 65678 |

TABLE II
PROBABILISTIC EVALUATION OF THE PROPOSED CONTROLLERS FOR THE QOS CONSTRAINTS WITH $\delta=10^{-6}$

The probabilistic evaluation has also been applied to the temperature constraint with table III summarizing the results. In this case, $Y$ stands as the number of total time instants for all the simulations $H$ (which were all of the same length), and the upper bound is on the probability of reaching a temperature higher than 80°C. Again the more conservative a controller is, the better its probabilistic guarantees, but at the expense of a higher cost.

| | $\rho_i$ | $\rho_i+\Delta\rho_i$ | $Y_i$ |
|---|---|---|---|
| S-Pbo, $N_c=1$ | 0.0000 | 0.0055 | 2500 |
| S-Pbo, $N_c=5$ | 0.0000 | 0.0055 | 2500 |
| Pbo, $N_c=1$ | 0.0001 | 0.0071 | 2500 |
| Pbo, $N_c=5$ | 0.0000 | 0.0055 | 2500 |

TABLE III
PROBABILISTIC EVALUATION OF THE PROPOSED CONTROLLERS FOR THE MAXIMUM TEMPERATURE CONSTRAINT WITH $\delta=10^{-6}$

### B. Parallel computation improvement

In order to prove the speed-ups obtained with a parallel implementation, the algorithms have been implemented in the CPU and also in a CUDA capable GPU using the indications of section III.C The CPU version is coded completely in Matlab while the GPU version uses Matlab code for the serial operations and C code for the CUDA kernels. Table IV shows the execution time of the controllers for a different number of particles. Cells with a hyphen mean that the time exceeded the sampling time. On the other hand, figure 6 shows the relative speed-up for different numbers of particles.

As it can be expected, at a higher number of particles, the execution time within the CPU grows strongly whereas the GPU time yields almost constant in comparison which leads to the increasing slope shown in figure 6. It also should be noted that once reached a number of particles greater than 1000, the execution times of the CPU become unmanageable and the simulations are extremely costly. However, this improvement will stop increasing and remain constant once reached the point where the GPU is saturated.

| | S-Pbo GPU | S-Pbo CPU | Pbo GPU | Pbo CPU |
|---|---|---|---|---|
| 10 | 17.0931 | 23.4282 | 0.6974 | 1.0001 |
| 100 | 17.8046 | - | 0.7230 | 9.4192 |
| 1000 | 30.2881 | - | 1.2267 | 93.6546 |
| 10000 | 91.4658 | - | 3.7208 | - |

TABLE IV
MEAN COMPUTATION TIME OF THE CONTROLLERS IN SECONDS.

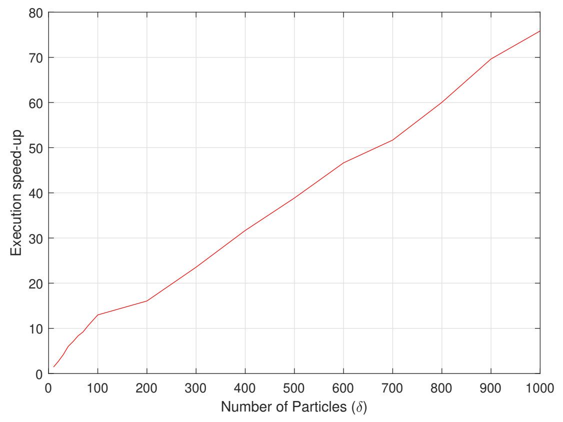

Fig. 6. Speed-up obtained with the GPU computing for the controller execution.

### C. Computation time analysis

Also, an analysis of the computation time of the algorithm with respect to different number of servers has been done in order to study its behavior for the problem of large data centers. The results are shown in figure 7. We also add to the figures a polynomial of order $n=2$ that confirms a quadratic complexity. Thus, the algorithm is feasible from a computational point of view and it can be used with larger data centers. Nevertheless, for higher numbers of servers, besides adding more computational power, one could consider multiple controllers each one for each cold aisle or consider that the control actions handle clusters of servers instead of individual ones.

## VI. CONCLUSIONS

The use of predictive control in the energy efficient management of data center under service and temperature constraints has been considered. The data center dynamics have been simulated using a queue model based on more realistic assumptions than others found in the literature. The control variables are the temperature set points of the CRAC unit and the number of servers "on" through the control horizon. A massive-parallelizable particle algorithm is used to deal with the MPC optimization problem to be solved. Each particle is a candidate solution to the problem and generates a simulation of the future behaviour of the system for its unique control actions so that its performance can be measured. A probabilistic evaluation procedure to choose among different values of the controller parameters has been presented. This scheme provides guarantees of constraint satisfaction with a certain probabilistic performance level. Finally, the results of the paper have been illustrated by means of simulated examples. Future

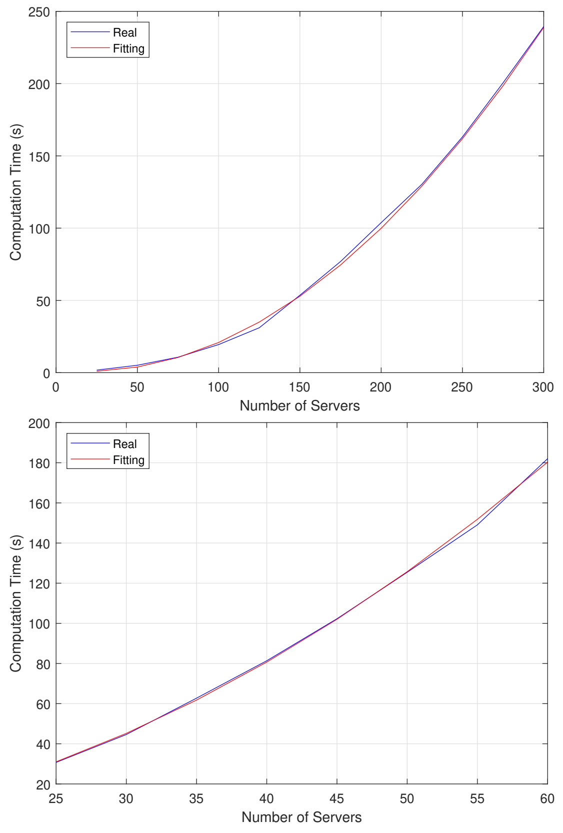

Fig. 7. Computation times of the algorithms for an increasing number of servers. Top: Algorithm without scenarios. Bottom: Algorithm with scenarios

work involves the addition of new control variables such as CPU frequency scaling in order to improve the management. On the other hand, hyperparameter selection is a task to be tackled. While the data center dynamics is modelled, making the control task suitable for MPC, there is no model for the influence of the hyper-parameters of the proposed algorithm. Learning techniques like Reinforcement Learning [45] can be considered for hyper-parameter selection as this technique does not need such model. In this context, the probabilistic evaluation scheme could be used in the design of the reward function of a Reinforcement Learning algorithm that explores the possible values of the controller hyperparameters (e.g., prediction and control horizons, weight factors, etc.) in search of the best possible controller.

## Acknowledgment

This work was supported in part by the Agencia Estatal de Investigación (AEI) under Grant PID2019-106212RB-C41/AEI/10.13039/501100011033, by MINERCO-Spain and FEDER funds under Grant DPI2016-76493-C3-1-R and Junta de Andalucia under Grant PY20_00546.

## References

[1] L. Parolini, B. Sinopoli, B. H. Krogh, and Z. Wang, "A cyber–physical systems approach to data center modeling and control for energy efficiency," *Proceedings of the IEEE*, vol. 100, no. 1, pp. 254–268, 2011.

[2] J. Scaramella, "Worldwide server power and cooling expense 2006-2010 forecast," *Market analysis, IDC Inc*, 2006.

[3] A. S. Andrae and T. Edler, "On global electricity usage of communication technology: trends to 2030," *Challenges*, vol. 6, no. 1, pp. 117–157, 2015.

[4] P. X. Gao, A. R. Curtis, B. Wong, and S. Keshav, "It's not easy being green," in *Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication*, 2012, pp. 211–222.

[5] M. Dayarathna, Y. Wen, and R. Fan, "Data center energy consumption modeling: A survey," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 732–794, 2015.

[6] W. Van Heddeghem, S. Lambert, B. Lannoo, D. Colle, M. Pickavet, and P. Demeester, "Trends in worldwide ict electricity consumption from 2007 to 2012," *Computer Communications*, vol. 50, pp. 64–76, 2014.

[7] N. Lazic, C. Boutilier, T. Lu, E. Wong, B. Roy, M. Ryu, and G. Imwalle, "Data center cooling using model-predictive control," in *Advances in Neural Information Processing Systems*, 2018, pp. 3814–3823.

[8] H. Endo, S. Suzuki, H. Kodama, T. Hatanaka, H. Fukuda, and M. Fujita, "Development of predictive control system using just-in-time modeling and enthalpy-aware control in air conditioners for large-scale data center," in *2018 18th International Conference on Control, Automation and Systems (ICCAS)*. IEEE, 2018, pp. 1278–1283.

[9] M. Ogawa, H. Fukuda, H. Kodama, H. Endo, T. Sugimoto, T. Kasajima, and M. Kondo, "Development of a cooling control system for data centers utilizing indirect fresh air based on model predictive control," in *2015 7th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*. IEEE, 2015, pp. 132–137.

[10] L. Fu, J. Wan, J. Yang, D. Cao, and G. Zhang, "Dynamic thermal and it resource management strategies for data center energy minimization," *Journal of Cloud Computing*, vol. 6, no. 1, p. 25, 2017.

[11] C. Bash and G. Forman, "Cool job allocation: Measuring the power savings of placing jobs at cooling-efficient locations in the data center." in *USENIX Annual Technical Conference*, vol. 138, 2007, p. Page 140.

[12] J. D. Moore, J. S. Chase, P. Ranganathan, and R. K. Sharma, "Making scheduling" cool": Temperature-aware workload placement in data centers." in *USENIX annual technical conference*, 2005, pp. 61–75.

[13] S. Li, H. Le, N. Pham, J. Heo, and T. Abdelzaher, "Joint optimization of computing and cooling energy: Analytic model and a machine room case study," in *2012 IEEE 32nd International Conference on Distributed Computing Systems*, 2012, pp. 396–405.

[14] E. F. Camacho and C. Bordons Alba, *Model Predictive Control*, 2nd ed. Springer London, 2007.

[15] L. Parolini, B. Sinopoli, and B. H. Krogh, "Model predictive control of data centers in the smart grid scenario," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 10 505–10 510, 2011.

[16] M. Ogura, J. Wan, and S. Kasahara, "Model predictive control for energy-efficient operations of data centers with cold aisle containments," *IFAC-PapersOnLine*, vol. 51, no. 20, pp. 209–214, 2018.

[17] Q. Fang, J. Wang, and Q. Gong, "QoS-driven power management of data centers via model predictive control," *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 4, pp. 1557–1566, 2016.

[18] A. Bemporad and M. Morari, "Robust model predictive control: A survey," in *Robustness in identification and control*. Springer, 1999, pp. 207–226.

[19] D. Q. Mayne, M. M. Seron, and S. Raković, "Robust model predictive control of constrained linear systems with bounded disturbances," *Automatica*, vol. 41, no. 2, pp. 219–224, 2005.

[20] R. Tempo, G. Calafiore, and F. Dabbene, *Randomized algorithms for analysis and control of uncertain systems: with applications*. Springer Science & Business Media, 2012.

[21] A. T. Schwarm and M. Nikolaou, "Chance-constrained model predictive control," *AIChE Journal*, vol. 45, no. 8, pp. 1743–1752, 1999.

[22] G. C. Calafiore and L. Fagiano, "Robust model predictive control via scenario optimization," *IEEE Transactions on Automatic Control*, vol. 58, no. 1, pp. 219–224, 2012.

[23] S. Grammatico, X. Zhang, K. Margellos, P. Goulart, and J. Lygeros, "A scenario approach for non-convex control design," *IEEE Transactions on Automatic Control*, vol. 61, no. 2, pp. 334–345, 2015.

[24] B. Karg, T. Alamo, and S. Lucia, "Probabilistic performance validation of deep learning-based robust NMPC controllers," *arXiv preprint arXiv:1910.13906*, 2019.

[25] M. Alamir, "On probabilistic certification of combined cancer therapies using strongly uncertain models," *Journal of theoretical biology*, vol. 384, pp. 59–69, 2015.

[26] T. Alamo, R. Tempo, A. Luque, and D. R. Ramirez, "Randomized methods for design of uncertain systems: Sample complexity and sequential algorithms," *Automatica*, vol. 52, pp. 160–172, 2015.

[27] J. Yao, Z. Pan, and H. Zhang, "A distributed render farm system for animation production," in *Entertainment Computing – ICEC 2009*, S. Natkin and J. Dupire, Eds. Springer Berlin Heidelberg, 2009, pp. 264–269.
[28] B. Bixby, "The gurobi optimizer," *Transp. Re-search Part B*, vol. 41, no. 2, pp. 159–178, 2007.
[29] A. L. Visintini, W. Glover, J. Lygeros, and J. Maciejowski, "Monte carlo optimization for conflict resolution in air traffic control," *IEEE Transactions on Intelligent Transportation Systems*, vol. 7, no. 4, pp. 470–482, 2006.
[30] N. Kantas, J. Maciejowski, and A. Lecchini-Visintini, "Sequential monte carlo for model predictive control," in *Nonlinear model predictive control*. Springer, 2009, pp. 263–273.
[31] G. Andreadis, L. Versluis, F. Mastenbroek, and A. Iosup, "A reference architecture for datacenter scheduling: design, validation, and experiments," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis*, 2018, p. 37.
[32] J. P. De Villiers, S. Godsill, and S. Singh, "Particle predictive control," *Journal of Statistical Planning and Inference*, vol. 141, no. 5, pp. 1753–1763, 2011.
[33] G. Kitagawa, "Monte carlo filter and smoother for non-gaussian nonlinear state space models," *Journal of computational and graphical statistics*, vol. 5, no. 1, pp. 1–25, 1996.
[34] J. Sanders and E. Kandrot, *CUDA by example: an introduction to general-purpose GPU programming, portable documents*. Addison-Wesley Professional, 2010.
[35] K. Karimi, N. G. Dickson, and F. Hamze, "A performance comparison of CUDA and OpenCL," *arXiv preprint arXiv:1005.2581*, 2010.
[36] S. Che, M. Boyer, J. Meng, D. Tarjan, J. W. Sheaffer, and K. Skadron, "A performance study of general-purpose applications on graphics processors using cuda," *Journal of parallel and distributed computing*, vol. 68, no. 10, pp. 1370–1380, 2008.
[37] P. Bialas and A. Strzelecki, "Benchmarking the cost of thread divergence in CUDA," in *International Conference on Parallel Processing and Applied Mathematics*. Springer, 2015, pp. 570–579.
[38] V. D. Blondel and J. N. Tsitsiklis, "A survey of computational complexity results in systems and control," *Automatica*, vol. 36, no. 9, pp. 1249–1274, 2000.
[39] T. Alamo, R. Tempo, and E. F. Camacho, "Randomized strategies for probabilistic solutions of uncertain feasibility and optimization problems," *IEEE Transactions on Automatic Control*, vol. 54, no. 11, pp. 2545–2559, 2009.
[40] T. Alamo, V. Mirasierra, F. Dabbene, and M. Lorenzen, "Safe approximations of chance constrained sets by probabilistic scaling," in *2019 18th European Control Conference (ECC)*. IEEE, 2019, pp. 1380–1385.
[41] H. Chernoff *et al.*, "A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations," *The Annals of Mathematical Statistics*, vol. 23, no. 4, pp. 493–507, 1952.
[42] M. Okamoto, "Some inequalities relating to the partial sum of binomial probabilities," *Annals of the institute of Statistical Mathematics*, vol. 10, no. 1, pp. 29–35, 1959.
[43] H. Cramér, "Les sommes et les fonctions de variables aléatoires," *Actualités Scientifiques et Induestrielles. Conférences Internationales de Sciences.* Paris Hermann, vol. 3, 1938.
[44] A. Klenke, *Probability theory: a comprehensive course*. Springer Science & Business Media, 2013.
[45] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of artificial intelligence research*, vol. 4, pp. 237–285, 1996.

**A. Daniel Carnerero** received his M. Eng. Degree in Industrial Engineering from the University of Seville in 2019. He is currently a PhD candidate in the Department of Systems Engineering and Automation at the University of Seville. His current research interests include GPU computing, metaheuristic optimization, model predictive control, randomized algorithms and data-driven methods.

**Daniel R. Ramirez** received M.Eng. and PhD degrees in Computer Engineering from the University of Seville in 1996 and 2002, respectively. From 2009 he is Associate Professor of the Department of System Engineering and Automation of the University of Seville. He has authored and co-authored more than 70 technical papers in international journals and conference proceedings. His current research interests include randomized algorithms, model predictive control, data-based forecasting and soft computing techniques.

**Teodoro Alamo** received the M.Eng. degree in telecommunications engineering from the Polytechnic University of Madrid in 1993, and the PhD degree in telecommunications engineering from the University of Seville in 1998. He has been a Full Professor of the Department of System Engineering and Automation in the University of Seville since 2010. He is the author or co-author of more than 200 publications, including books, book chapters, journal articles and conference proceedings. His current research interests include decision-making, model predictive control, data-driven methods, randomized algorithms, and optimization strategies.

**Daniel Limon** received the M.Eng. and Ph.D. degrees in electrical engineering from the University of Seville, Seville, Spain, in 1996 and 2002, respectively. From 2017 he is Full Professor of the Department of System Engineering and Automation in the University of Seville. He has been visiting researcher at the University of Cambridge and the Mitsubishi Electric Research Labs in 2016 and 2018 respectively. His current research interests include model predictive control, stability and robustness analysis, tracking control and data-based control.