Measurement-based Efficient Resource Allocation with Demand-Side Adjustments*

Georgios C. Chasparis†

December 19, 2017

Abstract

The problem of efficient resource allocation has drawn significant attention in many scientific disciplines due to its direct societal benefits, such as energy savings. Traditional approaches in addressing online resource allocation problems neglect the potential benefit of feedback information available from the running tasks/loads as well as the potential flexibility of a task to adjust its operation/service level in order to increase efficiency. The present paper builds upon recent developments in the area of bandwidth allocation in computing systems and proposes a unified design approach for efficient resource allocation which is based upon a measurement- or utility-based learning scheme. We demonstrate through analysis the potential of the proposed scheme in providing efficient allocation of resources in a large class of resource allocation problems and when only measurements of the performances of the tasks are available.

1 Introduction

Resource allocation has become an indispensable part of the design of any engineering system that consumes resources, such as electricity power in home energy management [1], access bandwidth and battery life in wireless communications [8], computing bandwidth and memory in parallelization algorithms [3], computing bandwidth in CPU cores [2].

When resource allocation is performed online and the number, arrival and departure times of the tasks are not known a priori (as in the case of CPU bandwidth allocation), the role of a resource manager (RM) is to guarantee an *efficient* operation of all tasks by appropriately distributing resources to the tasks and also assigning their operation/service level. However, guaranteeing efficiency through the adjustment of resources and/or service levels requires the formulation of a centralized optimization problem (e.g., mixed-integer linear programming formulations [2]), which further requires information about the specifics of each task and their response under different resource-service level pairs. Such information may not be available to neither the RM nor the task itself.

Given the difficulties involved in the formulation of centralized optimization problems, not to mention their computational complexity, feedback from the running tasks in the form of performance

*This work has been supported by the European Union grant EU H2020-ICT-2014-1 project RePhrase (No. 644235). It has also been partially supported by the Austrian Ministry for Transport, Innovation and Technology, the Federal Ministry of Science, Research and Economy, and the Province of Upper Austria in the frame of the COMET center SCCH. An earlier version of parts of this paper appeared in [5].

†G. Chasparis os with the Department of Data Analysis Systems, Software Competence Center Hagenberg GmbH, Softwarepark 21, A-4232 Hagenberg, Austria; E-mail: georgios.chasparis@scch.at.

measurements may provide valuable information for the establishment of efficient allocations. Such (feedback-based) techniques have recently been considered in several scientific domains, such as in the case of application parallelization (where information about the memory access patterns or affinity between threads and data are used in the form of scheduling hints) [4], or in the case of allocating virtual processors to computing applications [12].

To tackle the issues of centralized optimization techniques, resource allocation problems have also been addressed through distributed or game-theoretic optimization schemes. The main goal of such approaches is the solution of centralized (global) optimization problems through agent-based (local) objectives, where agents may represent the tasks to be allocated. Examples include the cooperative game formulation for allocating bandwidth in grid computing [13], the non-cooperative game formulation in the problem of medium access protocols in communications [14] or for allocating resources in cloud computing [15].

Recently, a distributed learning scheme has been proposed [6] specifically tailored for the problem of CPU allocation for time-sensitive applications. This scheme exhibits the benefits of measurement- or feedback-based methods, while, in parallel, allows applications for adjusting their own operation/service levels. Motivated by this work and the potential of exploiting both measurements available from the tasks and the flexibility of some tasks in changing their own operation level, in the present paper we propose a unified design methodology for addressing a general class of online resource allocation problems.

According to the proposed scheme, the RM is responsible for adjusting both the resource allocation and the operation level of each task, where the adjustment processes are based only on performance measurements received from each task. The proposed scheme exhibits adaptivity and robustness in the number, type and performance variations of the tasks. It may also be implemented in a distributed fashion, where both resource- and operation-levels are updated locally by each task independently. We demonstrate through analysis the potential of the proposed scheme in the establishment of efficient allocations for a large class of resource allocation problems.

Summarizing, the contributions of the proposed scheme (as compared to [5, 6]), are as follows:

- it introduces a measurement- or feedback-based resource allocation framework that applies to a general class of resource allocation problems, not necessarily restricted to time-sensitive applications;

- tasks may be of different nature (e.g., any application that may request bandwidth in a computing system);

- the analysis incorporates measurement-noise, and provides a robust analysis of the convergence properties in the presence of noisy observations;

- the analysis provides global convergence guarantees under minimal assumptions in the design of performance functions of the tasks (and not only for the case of peak demand as in [5, 6]).

The paper is organized as follows. Section 2 formulates the centralized optimization problem for a general class of resource allocation problems and provides examples. Section 3 presents a learning scheme for addressing adaptive resource allocation. Section 4 presents convergence properties of the resource-level updates, while Section 5 presents convergence properties of the combined resource- and operation-levels update scheme. Section 6 provides a simulation study in the context of power management in residential buildings. Finally, Section 7 presents concluding remarks.

Notation:

– $\Pi_{[a,b]}$ is the projection onto the set $[a, b]$.

– For any $x \in \mathbb{R}^n$ and set $A \subset \mathbb{R}^n$, define

$$\mathrm{dist}\,(x, A) \doteq \inf_{y \in A} |x - y|,$$

where $|\cdot|$ denotes the Euclidean norm.

– For some set $A \subset \mathbb{R}^n$ and $\delta > 0$,

$$\mathcal{B}_\delta(A) \doteq \{x \in \mathbb{R}^n : \mathrm{dist}\,(x, A) \leq \delta\}.$$

– The probability simplex of dimension n is denoted by $\mathbf{\Delta}\,(n)$ and defined as

$$\mathbf{\Delta}\,(n) \doteq \Big\{x = (x_1, ..., x_n) \in [0,1]^n : \sum_{i=1}^{n} x_i = 1\Big\}.$$

– For some finite set A, $|A|$ denotes the cardinality of A.

2 Problem Formulation

2.1 Framework

We consider a resource allocation framework where one or multiple users request a finite number of tasks $\mathcal{I} = \{1, 2, ..., n\}$ to be executed. We denote such requests by $d_i \in \mathcal{D}_i$, indicating the *demand* of a user with respect to task i. Each of these tasks may run at a different *operation* or *service level*, denoted by $s_i \in \mathcal{S}_i$ indicating the level of comfort provided to the user through task i. We admit a normalization of the space of operation levels, i.e., we consider $\mathcal{S}_i \doteq [0, 1]$, ranging between its two extreme values.

In order for a task to be executed, an amount of resources needs to be assigned to it which corresponds to a portion $v_i \in \mathcal{V}_i \doteq [0, 1]$ of the overall available resource. In other words, v_i corresponds to the rate of accessing a common good. Here it is implicitly assumed that there is one type of available resource. Examples include a) electrical power in residential buildings where tasks represent electricity loads, b) computing bandwidth in CPU cores or computing grids, and c) communication bandwidth in communication systems.

The operation level of each task, s_i, and the amount of resources assigned to it, v_i, are determined by a *resource manager* (RM) which is responsible for maintaing a desirable performance of the overall system (according to some user-defined criterion). The RM makes decisions about the resources and service levels of the tasks at regular time instances denoted by $k = 0, 1, 2, ...$ The assignment of resources and service levels to the tasks is based upon measurements related to the performance of each task, denoted by $\tilde{u}_i$. Throughout the paper, we consider the following assumption.

Assumption 2.1 *The* RM *satisfies the following design properties:*

– *(D1) The internal characteristics of the tasks may not be known to the* RM. *Instead, the* RM *may only have access to measurements related to their performance.*

– *(D2) Tasks may not be split, rescheduled or postponed. Instead, the goal of the* RM *is to assign the* currently *available resources to the* currently *requested tasks in an efficient manner.*

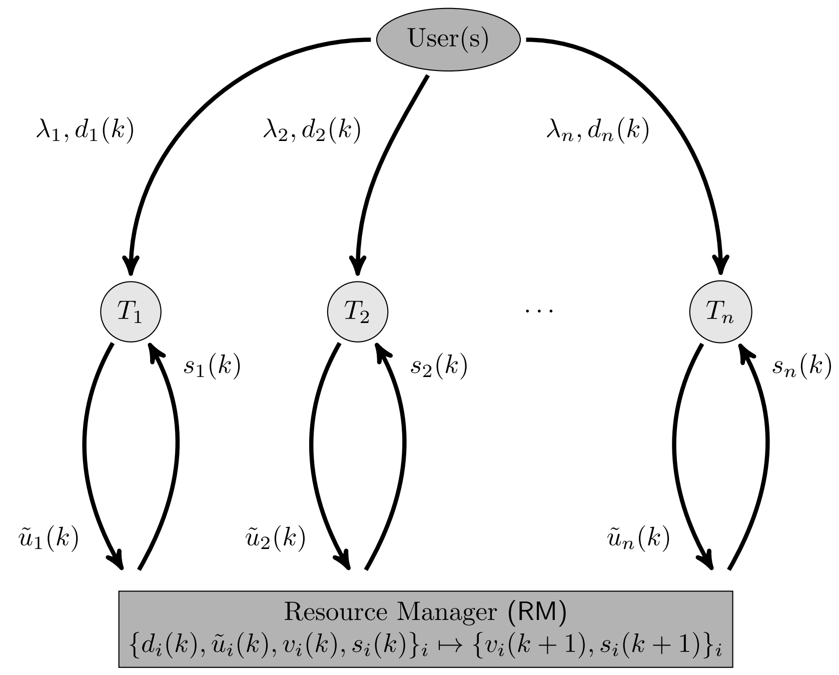

Figure 1: Schematic of resource allocation framework.

Note that the design assumptions (D1) and (D2) describe a framework at which the starting time of a task is *not* an optimization parameter. The RM does not have the necessary information to make such scheduling decisions (e.g., it does not know the duration time of a task). Thus, the main question is how to efficiently assign resources to the tasks assuming that they should immediately start running upon creation.

The overall framework is illustrated in Figure 1 describing the flow of information, starting from the users who determine the requests and ending to the RM which recursively allocates resources $v = (v_1, ..., v_n)$ and operation levels $s = (s_1, ..., s_n)$ to the tasks in $\mathcal{I}$ based on the recorded measurements. Figure 2 demonstrates schematically how an allocation of resources v may look like for a set of tasks requesting resources at regular time intervals. We assume that allocations belong to the set $\mathcal{V} \doteq \boldsymbol{\Delta}(n)$, since each resource v_i may only be a portion of the total available resource corresponding to 1. Figure 3 demonstrates how an allocation of operation levels $s_i \in \mathcal{S}_i$ to a task i may look like as updated by the RM over time.

It is important to point out that *the amount of resources assigned by the* RM *to the currently requested tasks may not necessarily correspond to the amount of resources used by each task.* The amount of resources used by each task depend solely on the operation level s_i and the corresponding demand level d_i. Informally, we may say that the resource allocation v constitutes a form of recommendation provided by the RM. Whether this recommendation is indeed implemented depends on whether the operation level allocation is appropriately set to use efficiently the recommended amount of resources. These points will become more obvious shortly when we discuss these terms through some application scenarios.

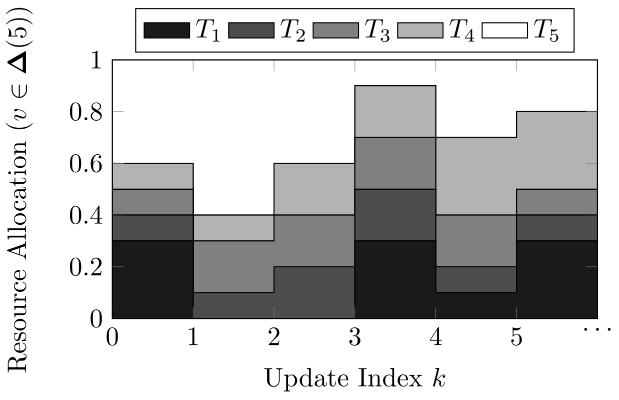

Figure 2: Schematic of resource-allocation evolution.

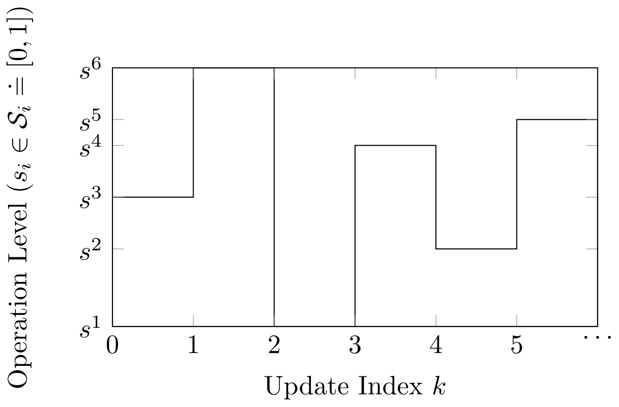

Figure 3: Schematic of operation-level evolution.

2.2 Utility function and efficiency

The objective is two fold. On the one hand, the RM is responsible for maintaining a *fair* allocation, v, among the requested tasks, while, on the other hand, the service level of each task i should guarantee an *efficient* operation given the amount of resources v_i provided by the RM. Before introducing the notions of *fairness* and *efficiency*, we first need to introduce the *performace measure* or *utility function* for each task i.

2.2.1 Utility function

The utility of a task i is introduced to capture the fitness of the task conditional to the amount of resources v_i provided by the resource manager, its operation level s_i and the user demand d_i. It is defined as a function of the form $u_i : \mathcal{S}_i \times \mathcal{V}_i \times \mathcal{D}_i \mapsto \mathbb{R}_+$, where we employ the following conditions.

Assumption 2.2 (Utility function) *The utility function $u_i : \mathcal{S}_i \times \mathcal{V}_i \times \mathcal{D}_i \mapsto \mathbb{R}_+$ of a task $i \in \mathcal{I}$ is continuous with respect to its arguments and satisfies:*

– *(U1) There exists a positive constant $c_i > 1$, such that*

$$1 \leq u_i(s_i, v_i, d_i) \leq c_i$$

uniformly on $v_i \in \mathcal{V}_i$*,* $s_i \in \mathcal{S}_i$ *and* $d_i \in \mathcal{D}_i$*.*

– *(U2) For any given allocation* v_i *and demand* d_i*,* $u_i(\cdot, v_i, d_i)$ *is concave with respect to its first argument* $s_i \in \mathcal{S}_i$*.*

Note that these are design assumptions that may be used to well represent the reasoning of a performance index in a resource allocation problem. In particular, we should always expect (given the boundedness of the provided resources) that (U1) the utility function is uniformly bounded from above. The condition of the utility function being greater than the unity is introduced for technical reasons and can always be met by appropriately shifting the utility function. Finally, we should further expect that as the operation level increases, then (U2) the corresponding performance should increase, however the gradient of the performance should saturate given the limited amount of resources.

2.2.2 Efficiency

Assuming that the utility function for each task has been designed, we introduce the following ***fairness measure***:

$$\Phi_i(s, v, d) \doteq \\ (1 - v_i)\lambda_i[u_i(s_i, v_i, d_i)]^{-1} - v_i \sum_{j \neq i} \lambda_j [u_j(s_j, v_j, d_j)]^{-1},$$

for some constants, $\lambda_i \in (0, 1]$, $i \in \mathcal{I}$.

The function Φ_i *captures the deficiency in resources of task* i *as compared to the rest of the tasks.* When task i is not performing well in comparison with the rest of tasks, i.e., $[u_i(s_i, v_i, d_i)]^{-1}$ is significantly larger than $[u_j(s_j, v_j, d_j)]^{-1}$, $j \neq i$, and its available resources v_i are small, we should expect that Φ_i admits large (positive) values (indicating deficiency of resources for task i). If, instead, task i is performing well, while it also has large amount of resources v_i, then we should expect that Φ_i admits small (negative) values (indicating sufficiency of resources for task i). The factor $\lambda_i \in [0, 1]$ which scales the inverse utility represents the importance of the task and it is user defined.

Definition 2.1 (Efficient allocation) *For some given demand profile* $d = (d_1, ..., d_n) \in \mathcal{D}$*, an allocation of resources* $v^* \in \mathcal{V}$ *and operation levels* $s^* \in \mathcal{S}$ *is called efficient if the following two conditions are satisfied:*

– *(E1)* $\Phi_i(s^*, v^*, d) \equiv 0$ *for all applications* i*.*

– *(E2)* $u_i(s_i^*, v_i^*, d_i) \to \max$ *for all applications* i*.*

We will often denote $\mathcal{E}^* = \mathcal{E}^*(d)$ as the set of efficient allocations. The first condition (E1) provides a *fairness* condition for allocating resources to the tasks, while condition (E2) guarantees the *efficient* operation of the task itself. Thus, a pair (v^*, s^*) will provide an ideal operation with respect to both a) the allocation of resources, and b) the operation of each task separately.

According to Definition 2.1 (E1), an allocation of resources v^* is *fair* for application i only if $v_i^* \neq 0$, since at zero resources $u_i(s_i, 0) > 0$. Thus, an allocation v^* is *fair* if and only if the following identity holds:

$$\frac{v_i^*}{\sum_{j \neq i} v_j^*} = \frac{\lambda_i [u_i(s_i, v_i^*, d_i]^{-1}}{\sum_{j \neq i} \lambda_j [u_j(s_j, v_j^*, d_j)]^{-1}},$$

given that $1 - v_i^* = \sum_{j \neq i} v_j^*$. In this case, *the resources are balanced with the performances.* To understand this identity, let us consider the case of equal weights, i.e., $\lambda_1 = ... = \lambda_n$. If v_i^* is large as compared to the rest of the resources, $\sum_{j \neq i} v_j^*$, then $[u_i]^{-1}$ has to be sufficiently large, i.e., task i should not perform so well in comparison with the rest of the tasks. Informally, there could *not* be a task i that monopolizes the resources at a fair allocation when i performs well and the others do not. Large amount of resources in a single task may only be justified if the task is performing poorly in comparison with the rest of the tasks. If we allow for non-uniform weights λ_i, then large amount of resources in a single task may also be justified by a large weight.

A similar fairness measure has been introduced within the context of CPU Bandwidth Allocation problem in [6]. The above definition is more general since it is not restricted to any specific application scenario.

2.3 Examples

To demonstrate the utility of the proposed framework, let us discuss the following practical scenarios.

2.3.1 Home-energy management

A simplified version of the smart-home paradigm considers a central resource manager which controls the amount of electrical power assigned to the electricity loads demanded by the user. In this case, $v_i \in [0, 1]$ represents the power assigned to each load where the maximum value 1 corresponds to the (desirable) maximum power available. Note that this might *not* be the actual power used.

Loads may correspond to flexible loads, such as the operation of the climate control, heat pumps or lighting. In this scenario, the user may define set-point temperatures for the operation of the heating system and the heat pumps, and desired luminance levels for lighting. Such set points may be considered as demand requirements, d_i.

The operation level s_i of each task i may correspond to the different levels of the service provided. For example, in the case of the heating system, it may correspond to the heating input provided to each thermal zone of the building, while in the case of the lighting equipment, it may represent the luminance level provided to each zone.

The definition of a utility function that may represent the operation of these tasks is open-ended. Consider the trivial example of the greedy objective for maximizing the comfort level provided by each task, which may be represented by a utility function of the form:

$$u_i(s_i, v_i, d_i) \doteq a\ell_i(s_i, d_i) + b(v_i - e_i(s_i)) + c, \tag{1}$$

for some positive constants a, b and c, where the function ℓ captures the comfort of the user, while the function e_i corresponds to the energy rate consumed by task i. Note that any excess energy rate from the assigned v_i (i.e., when $v_i < e_i(s_i)$) is penalized, while any energy rate savings (i.e., when $v_i > e_i(s_i)$) is encouraged. Alternative functions can be defined depending on the application and the performance indices which can be measured. The parameters of such objective function may be user-defined.

Let us consider the example of the heating system in a residential building as described in detail in [7]. In this example, the comfort of the user can be described as

$$\ell_i(s_i, d_i) \doteq \kappa - (s_i - d_i)^2,$$

for some positive constant $\kappa > 0$. The comfort admits its maximum value κ when the operation level meets the corresponding demand, i.e., $s_i \equiv d_i$. In any other case, the comfort admits lower values than κ. Furthermore, the heating cost of a radiant heating system can be approximated by a linear function of the flow rate of the thermal medium (in this case, the operation level), i.e.,

$$e_i(s_i) \doteq hs_i,$$

for some $h > 0$. Thus, in the case of the heating system, the utility function of the task takes on the following form:

$$u_i(s_i, v_i, d_i) = a\left(\kappa - (s_i - d_i)^2\right) + b(v_i - hs_i) + c,$$

for some positive constants a, b and c such that condition (U1) is satisfied. It is also straightforward to verify that the utility function is continuous with respect to its arguments and that (U2) the utility function is concave with respect to s_i due to the fact that $\nabla^2_{s_i} u_i(s_i, v_i, d_i) = -2a < 0$.

2.3.2 CPU bandwidth allocation

The above framework may also accommodate resource allocation problems encountered in the context of CPU bandwidth allocation. Recent work [6] has focused on designing such utility functions for the case of time sensitive applications. In this scenario, the RM is responsible for assigning *virtual-platforms* v_i to each application i. We may think of v_i as the percentage/portion of the CPU assigned to application i, which determines the rate with which an application i executes a job and the corresponding time interval assigned to the application.

Specifically in the case of time sensitive applications, including for example multimedia and control applications, the performance of the application depends on the relation between the *response-time* of a job R_i and the corresponding *soft-deadline* for executing a job, D_i, (determined by v_i). Good performance translates to $R_i \equiv D_i$. A natural definition of such a performance function for time-sensitive applications may take on the following form,

$$u_i(s_i, v_i, d_i) \doteq -a(D_i(d_i, s_i) - R_i(s_i, v_i))^2 + b, \tag{2}$$

for some constants $a, b > 0$ selected appropriately so that condition (U1) is satisfied. Note that the utility function attains a unique maximum when the deadline D_i approaches R_i, which is the desired property.

As described in [6], and in the context of multimedia applications, the soft deadline D_i can be considered constant, e.g., $D_i = h > 0$, while the response time can be defined as $R_i = {}^{C_i}/_{v_i}$, where $C_i = \rho_i s_i$ is the execution time per job (at a service level s_i), for some $\rho_i > 0$, and v_i is the speed of execution. In this case, the utility of application i takes on the following form:

$$u_i(s_i, v_i, d_i) = -a\Big(h - \rho_i \frac{s_i}{v_i}\Big)^2 + b. \tag{3}$$

It is straightforward to check that this function is continuous. Furthermore, it is concave with respect to the service level, since $\nabla^2_{s_i} u_i(s_i, v_i, d_i) = -2a(\rho_i/v_i)^2 < 0$.

2.4 Objective

Ideally, we would like to set up a centralized optimization problem, solved by the RM, such that at each update instance k and depending on the number of tasks and their importance, it would assign resources in a *efficient* manner to all tasks. Definition 2.1 introduces a potential centralized problem for efficient allocations, a candidate form of which is:

$$\begin{array}{ll} \min_{s\in\mathcal{S}, v\in\mathcal{V}} & \sum_{i\in\mathcal{I}} |\Phi_i(s, v, d)|^2 \\ \text{s.t.} & s_i = \arg\max_{s\in\mathcal{S}_i} u_i(s, v_i, d_i), \quad i \in \mathcal{I}, \end{array} \tag{4}$$

for some given $d_i \in \mathcal{D}_i$, $i \in \mathcal{I}$.

Whether such an optimization problem is well posed and the type of solutions it may accept depend on the characteristics of the designed utility functions u_i. Our goal is not to address such centralized optimization problem. This is because the definition of the utility function will necessarily be based upon measurements of quantities related to the performance of a task, whose explicit relation to the (internal) variables s_i and the provided resources v_i is *not* known in general.

To see this, let us consider the example of home energy management discussed in Section 2.3.1. Note that the function $e_i(s_i)$ captures the energy consumed by the task. It can be measured, however its explicit relation to the operation level s_i is not known a-priori to the RM (i.e., the parameter h is unknown). Similar is also the case in the CPU Bandwidth Allocation problem, where the deadline D_i and the response-time R_i can be measured by the RM, however their explicit dependence on the resource level v_i and operation level s_i is not known.

The RM may only respond to measurements available, and thus addressing a centralized optimization problem as stated above is *not* possible. *The goal of this paper is to investigate a class of utility- or measurement-based learning dynamics in addressing computation of efficient allocation pairs (s^*, v^*) as defined in Definition 2.1.*

3 Learning Dynamics

Given the difficulties in formulating centralized optimization problems in the absence of explicit knowledge of the characteristics of the tasks requesting resources, we propose an adaptive scheme which is based on *learning-based* (or *measurement-based*) dynamics. According to the proposed scheme, the RM is responsible for updating both the resource allocation v and the service levels s of the tasks. The goal is to attain convergence to an efficient allocation when only measurements of the utility functions are provided.

3.1 Resources update

At time instances t_k, $k = 0, 1, \ldots$ the RM measures the utility function of each task $i \in \mathcal{I}$ and updates the resources assigned to i as follows:

$$v_i(k+1) = v_i(k) + \epsilon F_i(k, v_i(k)), \tag{5}$$

for each $i = 1, ..., n$, where F_i is the *observed* fairness index defined as follows:

$$\begin{aligned} F_i(k, v_i(k)) &\doteq \\ &(1 - v_i(k))\lambda_i[\tilde{u}_i(k)]^{-1} - v_i(k) \sum_{j \neq i} \lambda_j [\tilde{u}_j(k)]^{-1}. \end{aligned}$$

The quantity $\tilde{u}_i(k)$ denotes the measurement of the utility function of task i which admits the form:

$$\tilde{u}_i(k) = u_i(s_i(k), v_i(k), d_i(k)) + \sigma_i(k), \tag{6}$$

where $\sigma_i(k)$ is a zero-mean bounded measurement noise, i.e., $\sup_{i \in \mathcal{I}} |\sigma_i| \leq \overline{\sigma}$ for some $\overline{\sigma} > 0$. We further assume that this noise process is independently distributed for each $i \in \mathcal{I}$. The introduction of the bounded noise process is necessary in order to capture some irregularities of the tasks (e.g., processes in computing systems). However, the type of the noise process cannot be known a-priori. The boundedness of the noise process is an indirect implication of the nature of the problems considered here, given that the performance indicators considered cannot deviate significantly from a nominal value (e.g., energy, time response, processing speed, etc.). However, even in the case that large noise values can be justified, for security reasons lower and upper bounds should be artificially introduced in all measured quantities.

According to the definition of F_i, if there is a deficiency of resources for i, i.e., $F_i > 0$, then v_i will increase, otherwise it will decrease. We consider a *constant* step size $\epsilon > 0$, since it provides an adaptive response to changes in the number of applications.

The above recursion (5) is motivated by the standard *replicator dynamics* (cf., [16, Chapter 3]), whose relevance and significance in resource allocation problems has been pointed out in [6] when addressing the problem of fair CPU Bandwidth Allocation. The difference here is that we include measurement noise in the observations of the performances of the tasks. Furthermore, the current framework is independent of the type of the task, so we do not assume any special form for the performance function u_i.

3.2 Operation-level update

Due to the concavity of the utility function u_i with respect to the operation level s_i, a gradient-based learning dynamics can be introduced for updating the operation level s_i, for each task i. Similarly to the case of the resource update, the explicit form of the utility function may not be known to the RM thus we may only make use of measurements of the utility function.

We would like that the operation level updates take place at a faster timescale as compared to the resource update (5). The reason for this choice is the better control over the resulting convergence properties of the overall dynamics, since any decision over the allocation of resources will be performed with the operation level updates being nearly equilibrated. To this end, we introduce the following recursion for the operation level of each task i.

$$s_i(k+1) = \Pi_{[0,1]}\left[s_i(k) + \epsilon\mu(\epsilon)\tanh\left(\frac{\tilde{U}_i(k)}{\tilde{S}_i(k)}\right) + \epsilon\mu(\epsilon)\zeta_i(k)\right] \tag{7}$$

where $\mu(\epsilon)$ is defined so that

$$\lim_{\epsilon\downarrow 0}\epsilon\mu(\epsilon) = 0, \quad \lim_{\epsilon\downarrow 0}\frac{\epsilon}{\epsilon\mu(\epsilon)} = 0, \tag{8}$$

i.e., ϵ goes faster to zero than $\epsilon\mu(\epsilon)$, as $\epsilon \downarrow 0$. Thus, the update recursion (7) moves on a faster timescale than recursion (5). The term $\zeta_i(k)$ corresponds to an artificially introduced noise term, i.e., $\zeta_i(k) \doteq \mathrm{rand}([-\overline{\zeta}, \overline{\zeta}])$, for some positive constant $\overline{\zeta} > 0$. The quantities $\tilde{U}_i(k)$ and $\tilde{S}_i(k)$ are approximations of the gradient of the measured performance $\tilde{u}_i(k)$ and the operation level $s_i(k)$, respectively. They can be generated by the following recursions

$$\begin{aligned}\tilde{U}_i(k) &\doteq \gamma\cdot(\tilde{u}_i(k) - \rho_i(k))\\ \tilde{S}_i(k) &\doteq \gamma\cdot(s_i(k) - \xi_i(k))\end{aligned}$$

for some $\gamma > 0$, where $\rho_i(k)$ and $\xi_i(k)$ are updated as follows:

$$\rho_i(k+1) = \rho_i(k) + \epsilon\mu(\epsilon)\cdot\tilde{U}_i(k) \tag{9a}$$

$$\xi_i(k+1) = \xi_i(k) + \epsilon\mu(\epsilon)\cdot\tilde{S}_i(k). \tag{9b}$$

Note that the higher the value of $\gamma > 0$, the better the approximation of the gradients. Thus, as γ increases, we should expect that s_i changes in the direction of increasing the utility $\tilde{u}_i$. This will become more clear when we discuss the convergence properties of the recursion.

3.3 Overall update recursion

It will be helpful to analyze the overall recursion dynamics as a whole, leading to the following set of recursions

$$\begin{pmatrix} v_i \\ s_i \\ \rho_i \\ \xi_i \end{pmatrix}(k+1) = \begin{pmatrix} v_i \\ s_i \\ \rho_i \\ \xi_i \end{pmatrix}(k) + \\ \epsilon \begin{pmatrix} F_i(k, v_i(k)) \\ \mu(\epsilon)\tanh\left(\frac{\tilde{U}_i(k)}{\tilde{S}_i(k)}\right) + \mu(\epsilon)\zeta_i(k) \\ \mu(\epsilon)\tilde{U}_i(k) \\ \mu(\epsilon)\tilde{S}_i(k) \end{pmatrix} + \epsilon \begin{pmatrix} 0 \\ z_i^s(k) \\ 0 \\ 0 \end{pmatrix} \tag{10}$$

$i \in \mathcal{I}$, where $z_i^s(k)$ are correction terms for the resource- and operation-level updates, respectively, that keeps them within the domain $[0,1]$. It is worth noting that the above recursion evolves in two timescales, the fast timescale of the operation level update, $s_i(k)$ (including the approximations $\tilde{U}_i(k)$ and $\tilde{S}_i(k)$) and the slow timescale of the resource-level update, $v_i(k)$. For convenience, in several cases, we will denote $x_i(k)$ as the overall state vector of task i, i.e.,

$$x_i(k) \doteq (v_i(k), s_i(k), \rho_i(k), \xi_i(k))$$

which evolves on $\mathcal{X}_i \doteq [0,1] \times [0,1] \times \mathbb{R} \times \mathbb{R}$. In the remainder of this paper, we will provide a characterization of the asymptotic behavior of the collection $x(k) = (x_1(k), ..., x_n(k)) \in \mathcal{X}_1 \times ... \times \mathcal{X}_n$ as the time index k increases. Note that the overall update recursion is stochastic in nature due to the presence of measurement noise, $\sigma_i(k)$ in the recordings of the performance of a task and secondly due to the artificial perturbation term, $\zeta_i(k)$, in the update of the operation level. In the following analysis, we will often use the probability and expectation operator $\mathbb{P}_x$ and $\mathbb{E}_x$, initiated at state x, defined on the canonical path space generated by the sequences of the recursion (10) for each $i \in \mathcal{I}$.

4 Resource-Level Convergence Properties

In this section, we demonstrate the convergence properties of the resource update recursion (5) *independently of the operation-level update.*

Before proceeding, it is important to find a bound for the *expected* utility measurement as well as the incremental difference of the resources. Let us introduce the notation: $\underline{\lambda} = \inf_{i\in\mathcal{I}} \lambda_i > 0$, $\underline{c} = \inf_{i\in\mathcal{I}} c_i > 1$, and $\overline{c} = \sup_{i\in\mathcal{I}} c_i > 1$. We will also make frequent use of the following sets $L_\alpha \doteq [0,\alpha)$ (i.e., 'less than α') and $G_\alpha \doteq (\alpha, 1]$ (i.e., 'greater than α'), for some constant $\alpha \in (0,1)$.

Proposition 4.1 (Bounded inverse utility) *As $\overline{\sigma} \downarrow 0$,*

$$[\tilde{u}_i(k)]^{-1} \approx [u_i(s_i, v_i, d_i)]^{-1} + \mathcal{O}\left(\overline{\sigma}^2\right),$$

and

$$\frac{1}{\overline{c}} \leq [\tilde{u}_i(k)]^{-1} \leq 1 + \mathcal{O}\left(\overline{\sigma}^2\right), \tag{11}$$

where $\mathcal{O}(\cdot)$ denotes the order of the approximation error of the equality/inequality.

Proof. See Appendix 8.1. □

Proposition 4.2 (Bounded fairness) *As $\overline{\sigma} \downarrow 0$, the incremental difference of the resource update satisfies*

$$\begin{array}{rcl} \underline{\Lambda}(v_i) & \doteq & \underline{\lambda}/\overline{c} - v_i n \left(1 + \mathcal{O}\left(\overline{\sigma}^2\right)\right) \\ & \leq & F_i(k, v_i(k)) \\ & \leq & 1 + \mathcal{O}\left(\overline{\sigma}^2\right) - v_i n \underline{\lambda}/\overline{c} \quad \doteq \quad \overline{\Lambda}(v_i). \end{array} \tag{12}$$

Proof. See Appendix 8.2. □

4.1 Feasibility

The first property of the proposed adjustment process is the feasibility of the resulting vector of resources. In fact, we would like that the resource vector $v(k)$ remains within the probability simplex $\mathbf{\Delta}(n)$ for all future times k.

Proposition 4.3 (Feasibility) *For any number of tasks n and noise size $\overline{\sigma} > 0$, there exists $\epsilon^* = \epsilon^*(n, \overline{\sigma}) > 0$, such that for any $\epsilon < \epsilon^*$, the resources update recursion (5) generates a sequence of resources $\{v(k)\}$ which satisfies $v(k) \in \mathbf{\Delta}(n)$ for all $k = 1, ...$ as long as $v(0) \in \mathbf{\Delta}(n)$.*

Proof. The sum of resources satisfies:

$$\begin{array}{l} \sum_{i=1}^{n} v_i(k+1) \\ \quad = \sum_{i=1}^{n} v_i(k) + \epsilon \sum_{j=1}^{n} \lambda_j [\tilde{u}_j(k)]^{-1} \Big(1 - \sum_{i=1}^{n} v_i(k)\Big). \end{array}$$

Note that the second part of the r.h.s. becomes identically zero when $\sum_{i=1}^{n} v_i(k) = 1$. Thus, if the initial allocation satisfies $\sum_{i=1}^{n} v_i(0) = 1$, then $\sum_{i=1}^{n} v_i(k) = 1$ for all $k = 1, 2, ...$

It remains to check under which conditions $v_i(k) \in [0, 1]$. From Proposition 4.2, we have that the incremental difference of v_i at time k satisfies:

$$|v_i(k+1) - v_i(k)| \leq \epsilon \left(1 + \mathcal{O}\left(\overline{\sigma}^2\right)\right) \doteq \omega(\epsilon) > 0. \tag{13}$$

In order for $v_i(k+1)$ to drop below zero, $v_i(k)$ should be at least within $\omega(\epsilon)$ distance from zero. Take $v_i(k) \in [0, \omega(\epsilon))$. Then,

$$\begin{array}{l} F_i(k, v_i(k)) \\ \quad = \lambda_i [\tilde{u}_i(k)]^{-1} - v_i(k) \sum_{j=1}^{n} \lambda_j [\tilde{u}_j]^{-1} \\ \quad \geq \underline{\lambda}/\overline{c} - \omega(\epsilon) n (1 + \mathcal{O}\left(\overline{\sigma}^2\right)). \end{array}$$

Given that $\underline{\lambda}, \overline{c} > 0$, there exists $\epsilon_1^* = \epsilon_1^*(n, \overline{\sigma})$ sufficiently small, such that if $\epsilon < \epsilon_1^*$, we have that $F_i(k, v_i(k)) \geq 0$ for all $v_i(k) \in [0, \omega(\epsilon))$.

Similarly, in order for $v_i(k+1)$ to become larger than 1, $v_i(k)$ should be at least within $\omega(\epsilon)$-distance from 1. Take $v_i(k) \in (1-\omega(\epsilon), 1]$. Then, we have

$$\begin{aligned}
&F_i(k, v_i(k)) \\
&\quad= (1 - v_i(k))\lambda_i[\tilde{u}_i(k)]^{-1} - v_i(k)\sum_{j \neq i}\lambda_j[\tilde{u}_j]^{-1} \\
&\quad\leq \omega(\epsilon)\lambda_i[\tilde{u}_i(k)]^{-1} - (1-\omega(\epsilon))\sum_{j \neq i}\lambda_j[\tilde{u}_j]^{-1} \\
&\quad\leq \omega(\epsilon) - (1-\omega(\epsilon))\underline{\lambda}(n-1)/\overline{c}
\end{aligned}$$

where we have used the properties $\underline{\lambda} \leq \lambda_i \leq 1$ and $[\tilde{u}_i]^{-1} \geq 1/\overline{c}$ for all $i \in \mathcal{I}$. Given that $\underline{\lambda}, \overline{c} > 0$, there exists $\epsilon_2^* = \epsilon_2^*(n)$ such that, for any $\epsilon < \epsilon_2^*(n)$, we have $F_i(k, v_i(k)) \leq 0$ for all $v_i(k) \in (1-\omega(\epsilon), 1]$.

In conclusion, for any $\epsilon < \epsilon^* \doteq \min\{\epsilon_1^*(n, \overline{\sigma}), \epsilon_2^*(n)\}$, we have $v_i(k) \in [0,1]$ for any $k = 1, 2, ...$ □

For the remainder of the paper, we will assume that the step size ϵ is chosen appropriately (according to Proposition 4.3), so that the resource level is always within the feasible region for all tasks.

4.2 Starvation Avoidance

The adjustment process guarantees *starvation avoidance*, i.e., a positive amount of resources to all tasks and at all times.

Proposition 4.4 (Starvation Avoidance) *Given a number of tasks $n \in \mathbb{N}$ and as $\overline{\sigma} \downarrow 0$, there exists $\alpha^* = \alpha^*(n) \doteq \underline{\lambda}/(n\overline{c}) > 0$ such that, for any task $i \in \mathcal{I}$, and any $0 < \alpha < \alpha^*$, the following holds*

$$\mathbb{P}_x\left[\liminf_{k \to \infty} \text{dist}\,(v_i(k), G_\alpha) = 0\right] = 1. \tag{14}$$

Proof. Let $\alpha > 0$. We restrict the analysis to the per-task process $\{v_i(k)\}$. Let also consider the non-negative function $V(k, v_i) \doteq 1 - v_i \geq 0$. The expected incremental difference of $V(k, v_i)$ satisfies

$$\begin{aligned}
&\Delta V(k) \\
&\quad\doteq \mathbb{E}_x\left[V(k+1, v_i(k+1)) - V(k, v_i(k)) | v_i(k) = v_i\right] \\
&\quad= \mathbb{E}_x\left[v_i(k) - v_i(k+1) | v_i(k) = v_i\right] \\
&\quad= -\epsilon\mathbb{E}_x\left[F_i(k, v_i(k)) | v_i(k) = v_i\right] \\
&\quad\leq -\epsilon\underline{\Lambda}(v_i),
\end{aligned}$$

for all $v_i \in [0,1]$, where $\underline{\Lambda}(v_i)$ is defined in Proposition 4.2. For any $v_i \in L_{\alpha-\delta} \doteq [0, \alpha-\delta)$, $\delta > 0$, we have that:

$$\underline{\Lambda}(v_i) \geq \underline{\lambda}/\overline{c} - (\alpha - \delta)n\left(1 + \mathcal{O}\left(\overline{\sigma}^2\right)\right).$$

As $\overline{\sigma} \downarrow 0$, there exists $\alpha^* = \alpha^*(n) \doteq \underline{\lambda}/(n\overline{c})$ such that, if $\alpha < \alpha^*$, then we have

$$\lim_{\overline{\sigma} \downarrow 0} \inf_{v_i \in L_{\alpha-\delta}} \underline{\Lambda}(v_i) = \underline{\lambda}/\overline{c} - (\alpha-\delta)n > \delta n > 0$$

for all $\delta > 0$. Then, the conclusion follows directly from [11, Theorem 5.1]. □

Proposition 4.4 states that the allocation of resources v_i that the RM will approach infinitely often an amount of resources that it is at least $\alpha^*(n) > 0$, i.e., it is always bounded away from zero for some fixed number of tasks n. Practically, this condition assures that zero amount of resources to one or more tasks cannot be sustainable.

4.3 Balance

Another property of the resource update recursion (that is complementary to the *starvation avoidance* property) assures that a task may never monopolize the amount of resources. This is especially important in the case of large number of tasks, thus establishing a form of *balance* between tasks.

Proposition 4.5 (Balance) *Pick* $0 < \beta \leq 1$. *There exists* $n^* = n^*(\beta) \doteq \lceil \overline{c}/(\beta\underline{\lambda}) \rceil$ *such that, for any set of applications* $\mathcal{I}$ *with* $|\mathcal{I}| \geq n^*$ *and for any* $i \in \mathcal{I}$, *the following hold:*

$$\mathbb{P}_x \left[\liminf_{k\to\infty} \operatorname{dist}(v_i(k), L_\beta) = 0 \right] = 1. \tag{15}$$

Also, as $\beta \to 0$, $n^*(\beta) \to \infty$ *and* $\beta n^*(\beta) \to \lceil \overline{c}/\underline{\lambda} \rceil$.

Proof. At time instance k, let $\mathcal{I}' \subseteq \mathcal{I}$ be the set of tasks with resources greater than β, i.e., $\mathcal{I}'(\beta) \doteq \{i \in \mathcal{I} : v_i(k) > \beta\}$. Pick $0 < \beta \leq 1$. For any $i \in \mathcal{I}'$, let us define the nonnegative function $V(k, v_i) \doteq v_i \geq 0$. The expected incremental difference of this function, as $\overline{\sigma} \downarrow 0$, satisfies

$$\begin{aligned} &\Delta V(k) \\ &\quad \doteq \quad \mathbb{E}_x \left[v_i(k+1) - v_i(k) | v_i(k) = v_i \right] \\ &\quad = \quad \epsilon \mathbb{E}_x \left[F_i(k, v_i(k)) | v_i(k) = v_i \right] \\ &\quad \leq \quad \epsilon \overline{\Lambda}(v_i) \end{aligned}$$

As $\overline{\sigma} \downarrow 0$, there exists $n^* = n^*(\beta) \doteq \lceil \overline{c}/(\beta\underline{\lambda}) \rceil$, such that if $n \geq n^*(\beta)$, then,

$$\lim_{\overline{\sigma}\downarrow 0} \sup_{v_i(k) \in G_{\beta+\delta}} \overline{\Lambda}(v_i) \leq 1 - v_i/\beta < 0,$$

for any $\delta > 0$. According to [11, Theorem 5.1], the conclusion follows. Finally, note that as $\beta \downarrow 0$, then $n^* \to \infty$ and $\beta n^* \to \lceil \overline{c}/\underline{\lambda} \rceil$. $\square$

Proposition 4.5 states that if we pick any $\beta \in (0, 1]$ and we consider a sufficiently large number of tasks $n \geq n^*(\alpha)$, then all tasks will end up with an amount of resources less than β infinitely often. Informally, we may say that *no task can monopolize the available resources when the number of tasks increases.*

4.4 Efficiency

Lastly, we demonstrate one of the most attractive properties of the proposed resource update recursion, i.e., the fact that, for any given allocation of the operation level s and the demand level d, the allocation of resources will approach an *efficient* (or *fair*) *allocation*, as defined by Definition 2.1. This is formally stated as follows. Let us define the set

$$\mathcal{E} = \mathcal{E}(s, d) \doteq \{v \in \mathbf{\Delta}(n) : \Phi_i(v_i, s_i, d_i) = 0\},$$

which correspond to the efficient allocations of Definition 2.1 with respect to the resource allocation v. Let us further denote the δ-neighborhood of this set by $\mathcal{B}_\delta(\mathcal{E})$.

Proposition 4.6 (Efficiency) *As $\overline{\sigma}, \epsilon \downarrow 0$ and for any $\delta > 0$,*

$$\mathbb{P}_x\left[\liminf_{k\to\infty} \operatorname{dist}\left(v(k), \mathcal{B}_\delta(\mathcal{E})\right) = 0\right] = 1.$$

Proof. Let us define the nonnegative function

$$W(k, v) \doteq \sum_{i\in\mathcal{I}} F_i(k, v_i)^2 \geq 0.$$

We can approximate the expected incremental gain of $W(k, v)$ by applying a Taylor series expansion. All expectations in this proof are conditioned to $v(k) = v$. We have:

$$\begin{aligned}
&\Delta W(k)\\
&\quad\doteq\ \mathbb{E}_x\Big[\sum_{i\in\mathcal{I}} \left[F_i(k+1, v_i(k+1))^2 - F_i(k, v_i(k))^2\right]\Big]\\
&\quad\approx\ \epsilon\sum_{i\in\mathcal{I}} \mathbb{E}_x\Big[[\nabla_{v_i} F_i(k, v_i)^2]^{\mathrm{T}} F_i(k, v_i)\Big] + \mathcal{O}\left(\epsilon^2\right).
\end{aligned}$$

Note that

$$\nabla_{v_i}[F_i(k, v_i)^2] = -2F_i(k, v_i)\sum_{j\in\mathcal{I}} \lambda_j[\tilde{u}_j(k)]^{-1},$$

where the observations $\tilde{u}_j$ are considered exogenous parameters. Thus,

$$\begin{aligned}
&\Delta W(k)\\
&\quad\approx\ \epsilon\sum_{i\in\mathcal{I}} \mathbb{E}_x\left[\Big(-2\sum_{j\in\mathcal{I}} \lambda_j[\tilde{u}_j(k)]^{-1}\Big) F_i(k, v_i)^2\right] + \mathcal{O}\left(\epsilon^2\right)\\
&\quad=\ -2\epsilon\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{I}} \lambda_j \mathbb{E}_x\left[[\tilde{u}_j(k)]^{-1} F_i(k, v_i)^2\right] + \mathcal{O}\left(\epsilon^2\right).
\end{aligned}$$

Given the boundedness of the performance function and the measurement noise (6), we have that $\sup_{v_i\in[0,1]} \tilde{u}_i = c_i + \overline{\sigma}$, which results in

$$\inf_{v_i\in[0,1]} [\tilde{u}_i]^{-1} = \frac{1}{c_i + \overline{\sigma}} \geq \frac{1}{\overline{c} + \overline{\sigma}}.$$

Thus, we have that

$$\Delta W(k) \leq \frac{-2\epsilon}{\overline{c} + \overline{\sigma}} \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{I}} \lambda_j \mathbb{E}_x\left[F_i(k, v_i)^2\right] + \mathcal{O}\left(\epsilon^2\right)$$

Let us define the set

$$\tilde{\mathcal{E}} \doteq \left\{v \in \mathbf{\Delta}(n) : \mathbb{E}_x\left[F_i(v_i, s_i, d_i)^2\right] = 0, \forall i \in \mathcal{I}\right\}.$$

Let also $\mathcal{G}_\delta \doteq \mathbf{\Delta}(n) \setminus \mathcal{B}_\delta(\tilde{\mathcal{E}})$, i.e., the set $\mathcal{G}_\delta$ contains all non-efficient allocations that are at least δ far from the ones in $\tilde{\mathcal{E}}$. It is evident that, for any $v \in \mathcal{G}_\delta$, $\sum_{i\in\mathcal{I}} \mathbb{E}_x\left[F_i(k, v_i)^2\right] > 0$, by definition of the set $\mathcal{G}_\delta$. Thus, by [11, Theorem 5.1], we have that, as $\epsilon \downarrow 0$ and for any $\delta > 0$,

$$\mathbb{P}_x\left[\liminf_{k\to\infty} \operatorname{dist}\Big(v(k), \mathcal{B}_\delta(\tilde{\mathcal{E}})\Big) = 0\right] = 1.$$

By convexity of the function x^2 and Jensen's inequality, we have that

$$\begin{aligned}\mathbb{E}_x\left[F_i(k,v_i(k))^2\right] &\geq \left(\mathbb{E}_x\left[F_i(k,v_i(k))\right]\right)^2 \\ &\approx \left(\Phi_i(v_i,s_i(k),d_i(k))+\mathcal{O}\left(\overline{\sigma}^2\right)\right)^2.\end{aligned}$$

Thus, as $\overline{\sigma}\downarrow 0$, $\tilde{\mathcal{E}}\subseteq\mathcal{E}$, which concludes the proof. $\square$

Proposition 4.6, states that for any $\delta > 0$, the allocation v of resources will be in the δ-neighborhood of the set of efficient allocations $\mathcal{B}_\delta(\mathcal{E})$ infinitely often with probability 1. Thus, it establishes a guarantee over the asymptotic properties of the algorithm *independently of the values of the operation levels and the demand levels.* However, as pointed out in the definition of the set of efficient allocations, the set of efficient allocations may change as operation levels and demand levels change.

4.5 Discussion

The above properties of *feasibility*, *starvation avoidance*, *balance* and *efficiency* provide guarantees that any scheduling mechanism should provide independently of the application of interest. In the application of CPU Bandwidth Allocation these properties are essential and should be satisfied by any operating system, i.e., tasks should always receive resources and resources should be balanced. Furthermore, note that these properties were shown only under Assumption 2.2, and thus they are relevant to a large class of resource allocation problems.

In the following section we wish to go one step further by providing a characterization of the convergence properties of the overall update recursion (10).

5 Overall Convergence Properties

The previous convergence results were shown under no assumption in the operation level $s_i(k)$, for each task $i\in\mathcal{I}$. In fact, the operation level may be constant or varying with no influence in the conclusions of Propositions 4.3–4.6. In this section, we wish to provide a more detailed characterization of the properties of the global attractors of the overall dynamics (10).

5.1 ODE approximation

The asymptotic behavior of the overall recursion (10) can be analyzed by the ODE-method for stochastic approximations [10]. In particular, the convergence behavior can be associated with the limit points of the following system of ordinary differential equations (ODE's):

$$\left(\begin{array}{c}\dot{\bar{v}}_i\\ \dot{\bar{s}}_i\\ \dot{\bar{\rho}}_i\\ \dot{\bar{\xi}}_i\end{array}\right)=\left(\begin{array}{c}\Phi_i(\bar{s},\bar{v},\bar{d})\\ \mu\tanh\left(\frac{u_i(\bar{s}_i,\bar{v}_i,\bar{d}_i)-\rho_i}{\bar{s}_i-\bar{\xi}_i}\right)+Z_i^s\\ \mu\gamma\left(u_i(\bar{s}_i,\bar{v}_i,\bar{d}_i)-\bar{\rho}_i\right)\\ \mu\gamma\left(\bar{s}_i-\bar{\xi}_i\right)\end{array}\right), \tag{16}$$

for each $i\in\mathcal{I}$, as the step size ϵ approaches zero, where the terms $\bar{v}_i$, $\bar{s}_i$, $\bar{d}_i$, $\bar{\rho}_i$, and $\bar{\xi}_i$ denote the linear-time interpolations[1] of the corresponding discrete-time variables $v_i(k)$, $s_i(k)$, $d_i(k)$, $\rho_i(k)$ and $\xi_i(k)$, respectively. The scalar Z_i represents the minimum effort required to drive $\bar{s}_i(t)$ back to $[0,1]$.

[1]The linear-time interpolation of a recursion $v(k)$, $k=0,1,...$, is defined as $\overline{v}(\tau)=v(k)$ for all $\epsilon t_k\leq\tau<\epsilon(t_{k+1})$.

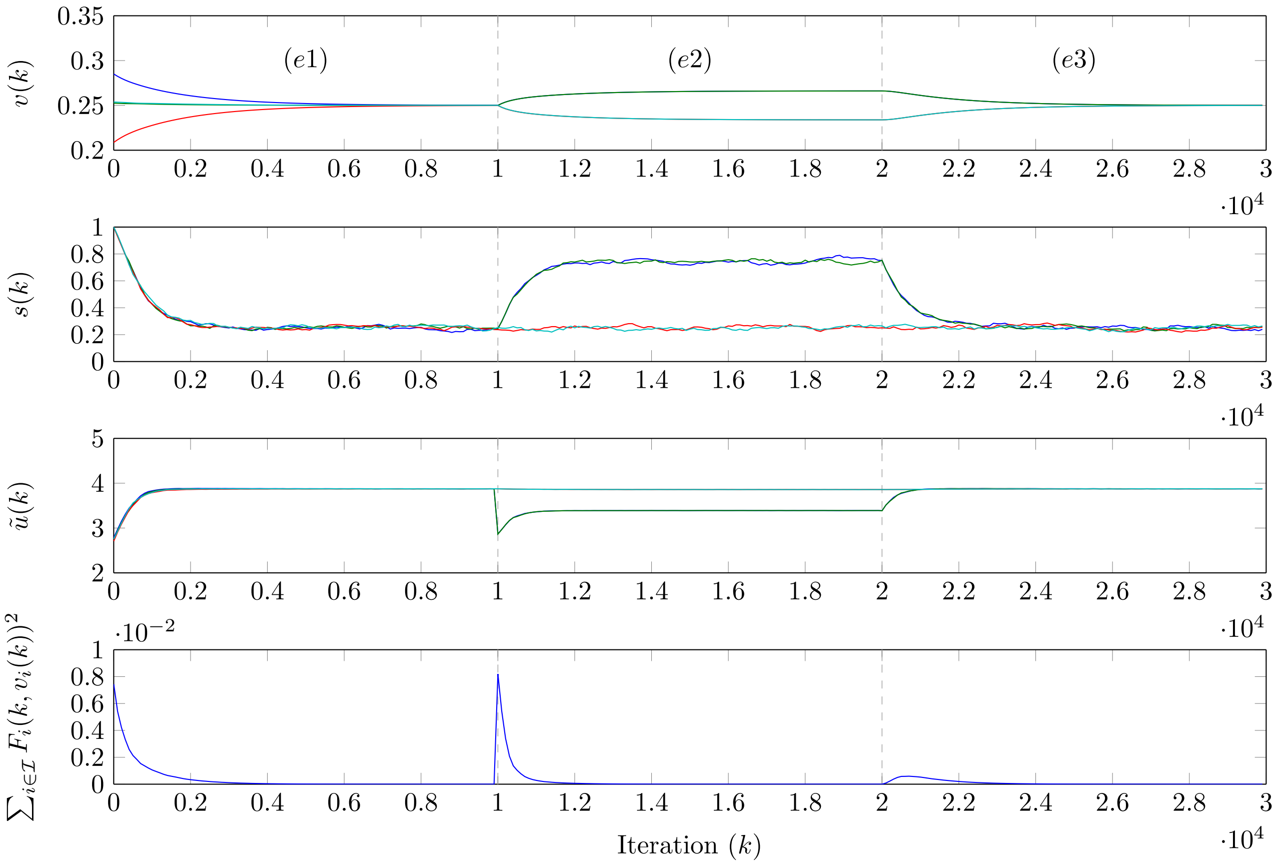

Figure 4: Resource- and operation-level adaptation for 4 identical tasks.

All terms of the above ODE are functions of an artificial continuous-time index t, which is skipped here to avoid confusion.

The following proposition associates the above system of ODE's (16) with the potential attractors of the recursions (10).

Proposition 5.1 (ODE approximation) *Consider the overall update recursion (10) with a fixed demand $d(k) = d$ and with a step size ϵ satisfying condition (8) and*

$$\epsilon\mu(\epsilon) < \frac{1}{\gamma}.$$

Let $\mathcal{L}$ denote the limit points[2] *of the system of ODE's*

$$\dot{\bar{v}}_i(t) = \Phi_i\left(\bar{s}^*(\bar{v}, \bar{d}), \bar{v}, \bar{d}\right), \quad i \in \mathcal{I}, \tag{17}$$

where $\bar{s}^* = (\bar{s}_1^*, ..., \bar{s}_n^*)$, and

$$\bar{s}_i^*(\bar{v}_i, \bar{d}_i) \doteq \arg\max_{\bar{s}_i\in[0,1]} u_i(\bar{s}_i, \bar{v}_i, \bar{d}_i), \quad i \in \mathcal{I}.$$

[2]The set of *limit points* $\mathcal{L}$ of an ODE $\dot{x} = g(x)$ with domain A is defined as $\mathcal{L} \doteq \lim_{t\to\infty} \bigcup_{x\in A}\{x(s), s \geq t : x(0) = x\}$, i.e., it is the set of all points in A to which the solution of the ODE converges.

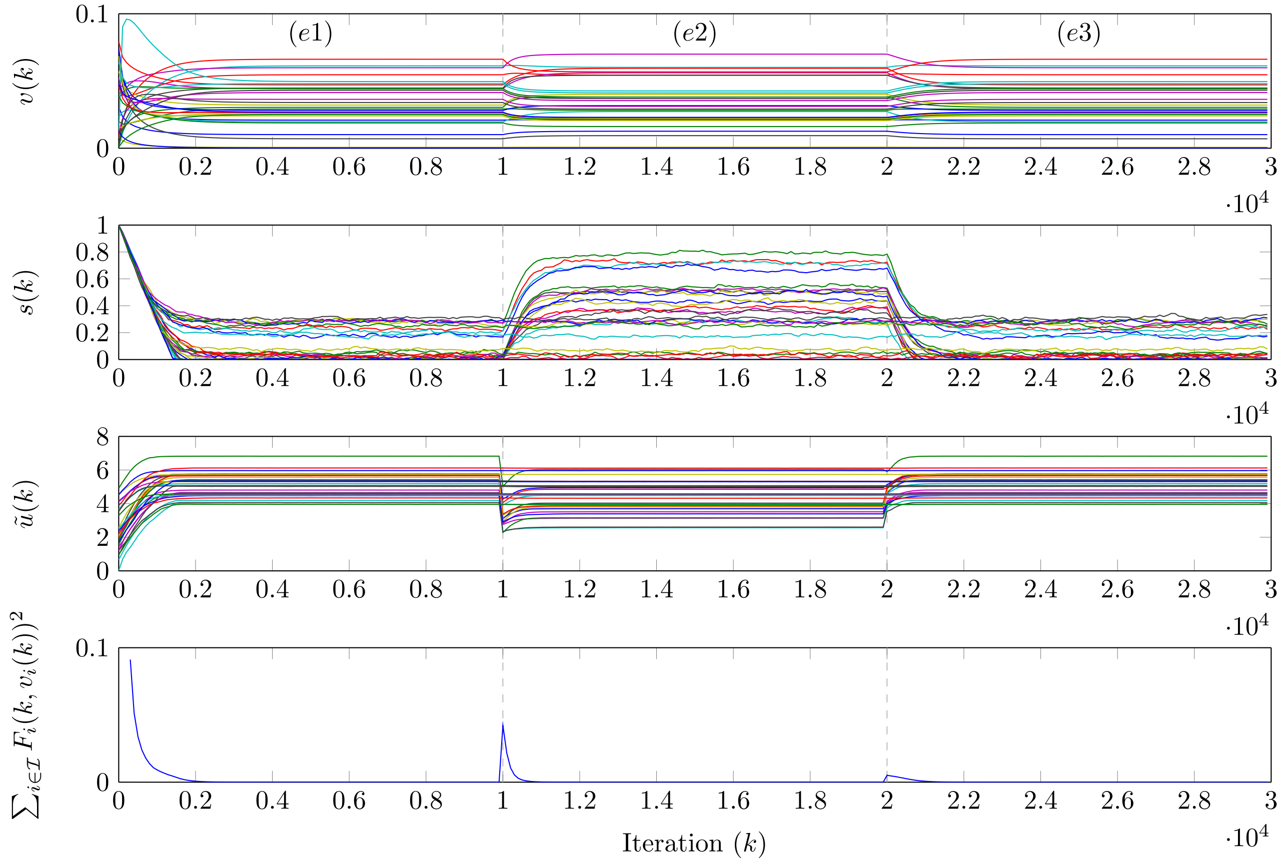

Figure 5: Resource- and operation-level adaptation for 30 randomly generated tasks.

As $\overline{\sigma} \downarrow 0$ and $\gamma \to \infty$, the following hold: for any $\delta > 0$, the fraction of time that the discrete-time process $\{s(k), v(k)\}$ spends in the δ-neighborhood of $\mathcal{L}$, $\mathcal{B}_\delta(\mathcal{L})$, goes to one (in probability) as $k \to \infty$.

Proof. See Appendix 8.3. □

The importance of Proposition 5.1 lies in the observation that through the two time-scale dynamics, we guarantee that: *independently of the resource-level adjustments $v_i(k)$, the operation level is always located at the maximizer of the utility function u_i.* Note though that convergence is established *in distribution* and states that if we consider a sufficiently large number of iterations $k \to \infty$, the recursion spends most of its time in a small neighborhood of the limit points of the ODE (16). In fact, the fraction of time that this occurs approaches one as the step size approaches zero and the number of iterations increases.

5.2 Global convergence

In general, the limit points of the limiting ODE (17) may include stationary points (i.e., $\dot{\tilde{v}}_i(t) = 0$, $i \in \mathcal{I}$) but not necessarily only stationary points. However, we have shown the following:

- Proposition 4.6 shows that the allocation of resources will be in a δ-neighborhood of the set of the efficient allocations $\mathcal{B}_\delta(\mathcal{E})$ infinitely often with probability one.

- Proposition 5.1 shows that the fraction of time that the discrete-time process spends in a δ-neighborhood of the limit points of the ODE (17), $\mathcal{B}_\delta(\mathcal{L})$, goes to one (in probability) as time increases.

The combination of these two results may only be valid if and only if the set of limit points $\mathcal{L}$ in Proposition 5.1 is replaced by the set of efficient allocations $\mathcal{E}^*$. This leads to the following theorem.

Theorem 5.1 (Global convergence) *Consider the hypotheses of Proposition 5.1. As $\overline{\sigma} \downarrow 0$ and $\gamma \to \infty$, we have: for any $\delta > 0$, the fraction of time that the discrete-time process $\{s(k), v(k)\}$ spends in $\mathcal{B}_\delta(\mathcal{E}^*)$, goes to one (in probability) as $k \to \infty$.*

5.3 Discussion

Theorem 5.1 establishes convergence (in a weak sense) of the discrete-time recursion of Equation (10) to the efficient allocations $\mathcal{E}^*$ as defined in Definition 2.1. We should expect that the fraction of time that these allocations are observed is close to one as we increase γ (or, equivalently, decrease ϵ).

The result applies for a fixed demand $d = (d_1, ..., d_n)$ set by the user. However, if the demand changes, the algorithm will automatically adapt to the new condition, due to the use of a constant step size ϵ. That is, the algorithm is adaptive to changes in the demand of the user. The time needed for the algorithm to converge to the efficient allocations depend on the selected step size ϵ. However, given the linear complexity of the recursions (10) with the number of tasks, as well as its distributed nature, the algorithm is computationally efficient.

The distributed nature of the dynamics lies in the observation that the overall update recursion (10) of task i can also be updated by each one of the tasks independently. In this case, the RM is simply responsible for communicating the measured quantities F_i to each one of the tasks.

6 Simulations

In this section, we provide a simulation study to demonstrate the convergence properties of the proposed dynamics. In all considered simulation studies, we introduce 3 time-zones (depicted by (e1), (e2) and (e3)), where we alter the demand-level requested by the tasks. In particular, in time-zone (e2) the demand-level of half the considered tasks increases by a factor of 2, while in time-zone (e3) the demand-level of the same tasks returns to its initial level (i.e., the one at time-zone (e1)). With this variation in the demand level of the task, we wish to demonstrate the adaptation of the proposed learning framework in varying user requests.

In the first simulation study of Figure 4, we consider 4 *identical* tasks with a utility function of the form (1), i.e., all parameters of the utility function are identical in each task. Furthermore, the initial requested demands are identical, as well as the weights of the tasks, i.e., $\lambda_i = \lambda = 1$ for each i. In particular, the considered parameters of the tasks simulated are: $a_i = 2$, $b_i = 1$, $c_i = 2$, $\epsilon = 0.0005$, $\mu(\epsilon) = \epsilon^{-1/20}$, and $\overline{\sigma} = \overline{\zeta} = 0.001$. In this case, and according to Definition 2.1, we should expect a unique efficient allocation corresponding to $\lambda_i / \sum_j \lambda_j = 1/n = 1/4 = 0.25$. This is indeed the emergent behavior in time-zone (e1). When the demand increases for two of the tasks, the emergent allocation accommodates this request (time-zone (e2)), and when the updated requests return to their original values, the initial allocation emerges again. This adaptive response of the dynamics to the demand variations should be attributed to the selection of constant step size ϵ, and agrees with our remark in Section 5.3. Note, finally, that throughout the simulation study, the efficiency criterion of F_i approaching zero is maintained, something that verifies our global convergence result of Theorem 5.1.

In the second simulation study of Figure 5, we consider 30 tasks with a utility function of the form (1). The parameters of the utility function, the weights and the demands of the tasks are randomly generated. Note that the main conclusions noted for the simple case of 4 tasks continue to hold even for this large number of tasks. In particular, the dynamics adapt rather fast to the variation in the demands, while the efficiency criterion is maintained throughout the simulations.

7 Conclusions and Future Work

We proposed a measurement- or utility-based learning scheme for addressing a large class of resource allocation problems. An initially formulated centralized objective was translated into resource- and operation-level-adjustment dynamics which exhibit desirable properties, such as starvation avoidance, balance and efficiency. Furthermore, global convergence guarantees to efficient outcomes were demonstrated under generic assumptions in the design of the utility functions.

The importance of the proposed methodology lies in the fact that no a-priori knowledge of the details of the utility function is required, something that it is relevant to several practical scenarios, such as the allocation of CPU bandwidth in computing applications. The proposed dynamics can also be distributed in a natural way, a property that is rather attractive when we consider an extremely large number of applications. The investigation of the convergence properties of the proposed update dynamics under this context (which may lead to asynchronous updates) should also be investigated.

8 Appendix

8.1 Proof of Proposition 4.1 (Bounded inverse utility)

By Taylor-series expansion of the inverse measurement function about its nominal value $u_i(s_i, v_i, d_i)$, we have that

$$[\tilde{u}_i]^{-1} = \sum_{m \geq 0} \frac{(-1)^m}{[u_i(s_i, v_i, d_i)]^{m+1}} \sigma_i^m.$$

This approximation is convergent by the ratio test, since $\overline{\sigma} < 1$ and $u_i(s_i, v_i, d_i) \geq 1$. We conclude that

$$\begin{aligned} &[\tilde{u}_i]^{-1} \\ &= \sum_{m \geq 0} (-1)^m \frac{\sigma_i^m}{[u_i(s_i, v_i, d_i)]^{m+1}} \\ &\approx [u_i(s_i, v_i, d_i)]^{-1} + \mathcal{O}\left(\frac{\sigma_i^2}{[u_i(s_i, v_i, d_i)]^3}\right). \end{aligned}$$

The second part of the above approximation is nonnegative. Thus, given that $1 \leq u_i(s_i, v_i, d_i) \leq c_i \leq \overline{c}$, we conclude that

$$[\tilde{u}_i]^{-1} \geq 1/c_i \geq 1/\overline{c}.$$

Furthermore, given that $[u_i(s_i, v_i, d_i)]^{-1}$ is uniformly bounded from below, and the fact that $\sigma_i^2 \leq \overline{\sigma}^2$, we may write equivalently that

$$[\tilde{u}_i]^{-1}$$

$$
\begin{aligned}
&\approx \ [u_i(s_i,v_i,d_i)]^{-1} + \mathcal{O}\left(\sigma_i^2\right) \\
&\leq \ [u_i(s_i,v_i,d_i)]^{-1} + \mathcal{O}\left(\overline{\sigma}^2\right) \\
&\leq \ 1 + \mathcal{O}\left(\overline{\sigma}^2\right),
\end{aligned}
$$

which establishes the desired upper bound.

8.2 Proof of Proposition 4.2 (Bounded fairness)

Given Proposition 4.1, and as $\overline{\sigma} \downarrow 0$, we have

$$
\begin{aligned}
&F_i(k, v_i(k)) \\
&\quad = \ \lambda_i[\tilde{u}_i(k)]^{-1} - v_i \sum_{j\in\mathcal{I}} \lambda_j [\tilde{u}_j(k)]^{-1} \\
&\quad \leq \ \lambda_i\left(1 + \mathcal{O}\left(\overline{\sigma}^2\right)\right) - v_i \sum_{j\in\mathcal{I}} \lambda_j / c_j \\
&\quad \leq \ 1 + \mathcal{O}\left(\overline{\sigma}^2\right) - v_i n \underline{\lambda}/\overline{c},
\end{aligned}
$$

where the last inequality results from the fact that $\underline{\lambda} \leq \lambda_i \leq 1$ and $\underline{c} \leq c_i \leq \overline{c}$. Accordingly, we get

$$
\begin{aligned}
&F_i(k, v_i(k)) \\
&\quad \geq \ {}^{\underline{\lambda}}\!/\!_{\overline{c}} - v_i n \left(1 + \mathcal{O}\left(\overline{\sigma}^2\right)\right),
\end{aligned}
$$

which concludes the proof.

8.3 Proof of Proposition 5.1 (ODE approximation)

Note the following:

- The utility function $u_i(\cdot,\cdot,\bar{d}_i)$ is continuous with respect to the operation level s_i and the resource level v_i, and therefore $\Phi(\cdot,\cdot,\bar{d})$ is also continuous with respect to s and v.

- The observation terms in (10) are uniformly bounded for all k in the domain, and therefore uniformly integrable. To show this, first note that $\mathbb{E}_x\left[|F_i(k, v_i(k))|\right] < \infty$ uniformly for all k, according to Proposition 4.2. Secondly, note that

$$
|\rho_i(k) - \rho_i(0)| \leq \sum_{\ell=0}^{k} \kappa(1-\kappa)^k \left|\tilde{u}_i(k-\ell) - \rho_i(0)\right|,
$$

  where $\kappa \doteq \epsilon\mu(\epsilon)\gamma$. Given Proposition 4.1, if $\kappa < 1$ (which will be the case when we take $\epsilon \downarrow 0$), $|\rho_i(k) - \rho_i(0)| < \infty$ uniformly for all k as long as $\rho_i(0)$ is bounded. The same conclusion can also be derived for the auxiliary state variable $\xi_i(k)$, given that $s_i(k) \in [0,1]$ for all k.

- Let $\mathcal{L}'$ denote the limit points of the ODE (16). The uniform integrability of the observation terms in (10) establishes (according to Theorem 8.2.1 in [10]) the following weak-convergence: As $\overline{\sigma} \downarrow 0$ and for any $\delta > 0$, the fraction of time that $(s_i(k), v_i(k), \rho_i(k), \xi_i(k))_i$ spends in the δ-neighborhood of $\mathcal{L}'$, $\mathcal{B}_\delta(\mathcal{L}')$, goes to one (in probability) as $\epsilon \downarrow 0$ and $k \to \infty$.

- As we increase the value of γ, we may refine the set of limit points $\mathcal{L}'$. Note that the auxiliary state variables $\bar{\rho}_i$ and $\bar{\xi}_i$ satisfy:

$$
\begin{aligned}
\dot{\bar{\rho}}_i(t)/\gamma &= \mu\left(u_i(\bar{s}_i(t), \bar{v}_i(t), \bar{d}_i(t)) - \bar{\rho}_i(t)\right) \\
\dot{\bar{\xi}}_i(t)/\gamma &= \mu\left(\bar{s}_i(t) - \bar{\xi}_i(t)\right).
\end{aligned}
$$

Given that both $\dot{\bar{\rho}}_i(t)$ and $\dot{\bar{\xi}}_i(t)$ are uniformly bounded, as we take $\gamma \to \infty$, we establish the following identities:

$$\begin{cases} u_i(\bar{s}_i(t), \bar{v}_i(t), \bar{d}_i(t)) = \bar{\rho}_i(t) \\ \bar{s}_i(t) = \bar{\xi}_i(t) \end{cases}$$

and therefore, by taking the time derivatives,

$$\begin{cases} \dot{u}_i(\bar{s}_i(t), \bar{v}_i(t), \bar{d}_i(t)) = \dot{\bar{\rho}}_i(t) \\ \dot{\bar{s}}_i(t) = \dot{\bar{\xi}}_i(t). \end{cases}$$

– Consider the (unprojected) faster response ODE, defined by

$$\begin{pmatrix} \dot{\bar{s}}_i \\ \dot{\bar{\rho}}_i \\ \dot{\bar{\xi}}_i \end{pmatrix} = \begin{pmatrix} \mu \tanh\left(\frac{u_i(\bar{s}_i, \bar{v}_i, \bar{d}_i) - \bar{\rho}_i}{\bar{s}_i - \bar{\xi}_i}\right) \\ \mu\gamma\left(u_i(\bar{s}_i, \bar{v}_i, \bar{d}_i) - \bar{\rho}_i\right) \\ \mu\gamma\left(\bar{s}_i - \bar{\xi}_i\right) \end{pmatrix}, \tag{18}$$

for some given $\bar{v}$ and $\bar{d}$. Furthermore, consider the nonnegative function

$$W(s) \doteq \sum_{i \in \mathcal{I}} \left\{ \max_{\bar{s}_i \in [0,1]} u_i(\bar{s}_i, \bar{v}_i, \bar{d}_i) - u_i(\bar{s}_i, \bar{v}_i, \bar{d}_i) \right\}.$$

Its time derivative satisfies:

$$\dot{W}(s) = -\mu \sum_{i \in \mathcal{I}} \nabla_{\bar{s}_i} u_i(\bar{s}_i, \bar{v}_i, \bar{d}_i) \cdot \dot{\bar{s}}_i(t),$$

where $\dot{\bar{s}}_i(t)$ is evaluated along the trajectories of the ODE (18). Thus, we may write:

$$\begin{aligned} \dot{\bar{s}}_i(t) \quad &= \quad \mu \tanh\left(\frac{\dot{\bar{\rho}}_i(t)}{\dot{\bar{\xi}}_i(t)}\right) \\ &\xrightarrow{\gamma \to \infty} \quad \mu \tanh\left(\frac{\dot{u}_i(\bar{s}_i(t), \bar{v}_i, \bar{d}_i(t))}{\dot{\bar{s}}_i(t)}\right). \end{aligned} \tag{19}$$

Since the involved time derivatives are evaluated along the trajectories of the ODE (18), note that

$$\begin{aligned} &\frac{\dot{u}_i(\bar{s}_i(t), \bar{v}_i, \bar{d}_i)}{\dot{\bar{s}}_i(t)} \\ &= \lim_{\Delta t \to 0} \frac{[u_i(\bar{s}_i(t+\Delta t)) - u_i(\bar{s}_i(t))]/\Delta t}{[\bar{s}_i(t+\Delta t) - \bar{s}_i(t)]/\Delta t} \\ &= \lim_{\Delta t \to 0} \frac{u_i(\bar{s}_i(t+\Delta t)) - u_i(\bar{s}_i(t))}{\bar{s}_i(t+\Delta t) - \bar{s}_i(t)} \\ &= \nabla_{\bar{s}_i} u_i(\bar{s}_i(t), \bar{v}_i, \bar{d}_i) \end{aligned}$$

where the last equality is due to the continuity of the solution $\bar{s}_i(t)$. Thus, we conclude that,

$$\dot{W}(\bar{s}) \xrightarrow{\gamma \to \infty}$$

$$-\mu \sum_{i\in\mathcal{I}} \nabla_{\bar{s}_i} u_i(\bar{s}_i, \bar{v}_i, \bar{d}_i) \tanh\big(\nabla_{\bar{s}_i} u_i(\bar{s}_i, \bar{v}_i, \bar{d}_i)\big) \leq 0.$$

The time derivative of the nonnegative function $W(\cdot)$ accepts a unique zero, satisfying

$$\begin{aligned} \dot{W}(\bar{s}) = 0 \quad &\Leftrightarrow \quad \nabla_{\bar{s}_i} u_i(\bar{s}_i, \bar{v}_i, \bar{d}_i) = 0 \\ &\Leftrightarrow \quad \bar{s}_i = \bar{s}_i^*(\bar{v}_i, \bar{d}_i) \doteq \arg\max_{\bar{s}_i\in[0,1]} u_i(\bar{s}_i, \bar{v}_i, \bar{d}_i), \end{aligned}$$

which according to [9, Theorem 3.1], shows that $\bar{s}_i^*$ is a globally asymptotically stable equilibrium point of the fast response dynamics (18). Given also Assumption 2.2, $\bar{s}_i^*(\bar{v}_i, \bar{d}_i)$ is the unique globally asymptotically stable equilibrium point of the ODE (18).

– Given that the globally asymptotically stable equilibrium of the unprojected dynamics lies within the domain $\mathcal{S}_i \doteq [0,1]$, $\bar{s}_i^*$ is also the unique globally asymptotically stable point of the projected ODE.

Thus, the conclusion is a direct implication of [10, Theorem 8.6.1].

References

[1] F. De Angelis, M. Boaro, D. Fuselli, S. Squartini, F. Piazza, and Q. Wei. Optimal home energy management under dynamic electrical and thermal constraints. *IEEE Transactions on Industrial Informatics*, 9(3):1518–1527, Aug 2013.

[2] Enrico Bini, Giorgio C. Buttazzo, Johan Eker, Stefan Schorr, Raphael Guerra, Gerhard Fohler, Karl-Erik Årzén, Romero Vanessa, and Claudio Scordino. Resource management on multicore systems: The ACTORS approach. *IEEE Micro*, 31(3):72–81, 2011.

[3] T. Brecht. On the importance of parallel application placement in NUMA multiprocessors. In *Proceedings of the Symposium on Experiences with Distributed and Multiprocessor Systems (SEDMS IV)*, pages 1–18, San Deigo, CA, July 1993.

[4] F. Broquedis, N. Furmento, B. Goglin, P.-A. Wecrenier, and R. Namyst. Forestgomp: An efficient openmp environment for NUMA architectures. *Int J Parallel Prog*, 38:418–439, 2010.

[5] Georgios C. Chasparis. Reinforcement-learning-based efficient resource allocation with demand-side adjustments. In *European Control Conference (ECC)*, pages 3066–3072, Linz, Austria, 2015.

[6] Georgios C. Chasparis, Martina Maggio, Enrico Bini, and Karl-Erik Årzén. Design and implementation of distributed resource management for time-sensitive applications. *Automatica*, 64:44–53, February 2016.

[7] Georgios C. Chasparis and Thomas Natschlaeger. Regression Models for Output Prediction of Thermal Dynamics in Buildings. *Journal of Dynamic Systems, Measurement, and Control*, 139(2):021006–021006, November 2016.

[8] H. Inaltekin and S. Wicker. A one-shot random access game for wireless networks. In *International Conference on Wireless Networks, Communications and Mobile Computing*, 2005.

[9] H.K. Khalil. *Nonlinear Systems.* Prentice-Hall, 1992.

[10] Harold J. Kushner and G. George Yin. *Stochastic Approximation and Recursive Algorithms and Applications.* Springer-Verlag New York, Inc., 2nd edition, 2003.

[11] M. B. Nevelson and R. Z. Hasminskii. *Stochastic Approximation and Recursive Estimation.* American Mathematical Society, Providence, RI, 1976.

[12] David C. Steere, Ashvin Goel, Joshua Gruenberg, Dylan McNamee, Calton Pu, and Jonathan Walpole. A feedback-driven proportion allocator for real-rate scheduling. In *Proc. of the* 3^{rd} *Symposium on Operating Systems Design and Implementation*, February 1999.

[13] Riky Subrata, Albert Y. Zomaya, and Björn Landfeldt. A cooperative game framework for QoS guided job allocation schemes in grids. *IEEE Transactions on Computers*, 57(10):1413–1422, October 2008.

[14] H. Tembine, E. Altman, R. ElAzouri, and Y. Hayel. Correlated evolutionary stable strategies in random medium access control. In *International Conference on Game Theory for Networks*, pages 212–221, 2009.

[15] Guiyi Wei, Athanasios V. Vasilakos, Yao Zheng, and Naixue Xiong. A game-theoretic method of fair resource allocation for cloud computing services. *The Journal of Supercomputing*, 54(2):252–269, November 2010.

[16] J. Weibull. *Evolutionary Game Theory.* MIT Press, Cambridge, MA, 1997.