s c c h
software competence center
hagenberg

FUZZY.LOGIC.LAB.LINZ

Advances in Knowledge-Based Technologies

Proceedings of the Master and PhD Seminar Summer term 2019, part 1

Johannes Kepler University Linz
Kopfgebäude, KG 519
3 May 2019

Software Competence Center Hagenberg
Softwarepark 21
A-4232 Hagenberg
Tel. +43 7236 3343 800
Fax +43 7236 3343 888
www.scch.at

Fuzzy Logic Laboratorium Linz
Softwarepark 21
A-4232 Hagenberg
Tel. +43 7236 3343 431
Fax +43 7236 3343 434
www.flll.jku.at

Program

Session 1 — Chair: Susanne Saminger-Platz

09:00 Ulrike Anlauf:
The Steiner Tree Problem considering Obstacles

09:30 Laura Peham:
Deep learning for advanced OCT image classification

Coffee Break: 10:00—10:15

Session 2 — Chair: Bernhard Moser

10:15 Katrin Treitinger:
Transfer Learning with Fuzzy Systems

10:45 Georgios C. Chasparis:
Perturbed Learning Automata in Coordination Games and Resource Allocation Problems

The Steiner Tree Problem considering Obstacles

This talk deals with the exact solution of industrial shortest path problems, i.e. the so-called Ascent Assembly Problem. The company partners provided us with models of harbour cranes and asked for an optimal assembly of an ascent to reach all points of interest for safety or repair purposes. Not all areas of such a crane are fit to attach such staircases. Those prohibited areas will be considered as obstacles and the solution should not cross any of them. This real-life problem leads to the Steiner Tree Problem considering also obstacles. Beside this crane scenario there exists a lot of other applications especially in Network Design. Those contain a wide range of industrial problems like the planning of roads, telephone line installations and electric circuits. The Steiner Tree Problem itself was most prominently discussed by Jakob Steiner in the 18th century, based on the early work of Fermat. Most of the progress in recent history can be credited to Winter who studied this problem from different perspectives (heuristics and exact solutions). The Steiner Tree Problem is difficult to solve and is NP hard, therefore most solutions are provided by meta-heuristic-solvers. The best heuristic solutions for the Steiner Tree Problem are within 4% from the exact solution

This talk is based on an investigation and adoption of results by Winter et al. for finding an optimal solution and comparison to known results.

Deep learning for advanced OCT image classification

Laura Peham

Department of Knowledge-Based Mathematical Systems - Johannes Kepler University Linz
and
RECENDT - Research Center for Non Destructive Testing GmbH
Altenberger Straße 69, 4040 Linz, Austria

April 30, 2019

Abstract

Optical coherence tomography (OCT) is a non destructive and non-contacting imaging technology and can visualize the internal structures of various materials. It works similar to ultrasound but it uses light waves instead of sound waves and has the great advantage over other imaging modalities that state-of-the-art systems can reach an axial resolution of 1-10μm. But a disadvantage is that because of the scattered light the imaging depth limits to about 2mm. OCT-imaging is used primary for biomedical issues e.g. in ophthalmology to view the retina layers or in dermatology for the detection of skin diseases. In recent years it also became popular for industrial applications for the non destructive testing of materials which is needed in quality assurance and for the development of new materials.
Deep Learning models are able to learn from very complex datasets and make predictions for new data based on this knowledge without the need of user-defined mathematical rules describing the task to solve. It is a machine learning method which deals with deep artificial neural networks. To get output values for each input value neurons (= units) are arranged in layers and each layer takes the output of the former layer as input and passes it forward to the next layer. Different kinds of architectures can be used to handle different input structures e.g. for the classification of images convolutional layers are used in addition to fully connected layers.
Medical specialist literature has already covered the successful application of deep learning on OCT-images for biomedical issues but the application of deep learning for OCT-image classification in an industrial context has not been covered by specialist literature so far. As the analysis and categorization of different materials is an important issue in many industrial processes I will examine the use of deep learning on OCT-images for material classification in two different tasks: The first task is to categorize 3D-printed objects with respect to their material (expressed by different color pigments) with only their (greyscale) OCT-image as input. Given are objects in the colors green, grey, red and transparent with two different objects in each color where in most of the cases the OCT-recordings of these four colors are well distinguishable. Therefore I recorded 12800 images (3200 per color) of different areas on the objects including surface defects, interior defects, plain surfaces and different slopes. Evaluating different model architectures obtained at most 99,5% accuracy on the test set until now, but the hyper-parameter optimization is still running.
The second task will be the categorization of OCT-images of different materials with respect to their coating-thickness.

Transfer Learning with Fuzzy Systems

Katrin Treitinger
LCM - Linz Center of Mechatronics
Linz, Austria
katrin.treitinger@lcm.at

April 29, 2019

Abstract

Before beginning with the transfer learning, Takagi-Sugeno Fuzzy Models have to be defined and the corresponding membership functions. In this regard the differences between the "Conventional Gaussian Membership Functions", which lead to axis-parallel ellipsoids and the "Generalized Gaussian Membership Functions", which lead to arbitrary ellipsoids, will be discussed. The resulting generalized fuzzy rules exhibit a better accuracy, but the rules become less interpretable. In order to find the principle rule structure, i.e. the adequate number of rules for the (transfer) learning problem at hand, a clustering of the data samples is carried out in the Source Task. Thus, in a first step a similar rule structure is assumed, consequently a fixed number of rules over both tasks. Four kinds of clustering methods are mentioned and also how an adequate number of clusters can be estimated: The "Fuzzy c-Means" algorithm and the "Gustafson-Kessel" algorithm are representing the clustering methods with fixed numbers of rules, where a second evaluation step is needed to find the best number of clusters. The "evolving Vector Quantization" and the "evolving Vector Quantization for arbitrary ellipsoids" are representing the clustering methods, where the numbers of clusters are determined on the fly. After the numbers of clusters are calculated, consequent parameters (for the Source Tasks) can be determined with the "Weighted Least Squares" WLS method for every rule individually (local learning) or the "Least Squares" LS method for one solution over all rules (global learning). The definitions of these optimization problems serve as starting point for a joint optimization (over Source and Target Task) for transfer learning. Three variants for a joint optimization are presented to establish transfer learning in fuzzy systems, all of which are relying on the concept of feature space representation learning through distribution matching of rule activation levels:

Variant 1 will match the distributions only in the consequent space, local for each rule separately, while the WLS Part will minimize the error on source task.

Variant 2 will match the distribution in the consequent space and in the antecedent space, local for each rule, while the WLS part will minimize the error on source task.

Variant 3 will match the distribution in the consequent space and in the antecedent space, global over all C rules, while the LS part will minimize the error on source task.

The local variants are expected to run more stable and quickly, because they have to deal with smaller optimization problems, but they can be "blind" for the global performance and perhaps won't find the global optimum. Matching only in the consequent space is expected to produce more precise result, because of a linear optimization problem, but only if the rules are close and only the functional tendency is different. The global variant will probably need more processing power because it has to deal with high-dimensional distributions.

Abstract: Perturbed Learning Automata in Coordination Games and Resource Allocation Problems

Georgios C. Chasparis

Recently, multi-agent formulations have been utilized to tackle distributed optimization problems, due to the increased communication and computational complexity of centralized schemes. In such multi-agent formulations, decisions are usually taken in a repeated fashion, where agents select their next actions based on their own prior experience (i.e., performance measurements). In this seminar, I will present a class of reinforcement-based learning (namely, perturbed learning automata) for convergence to efficient outcomes in (multi-agent) coordination games. Prior work in this class of learning dynamics primarily analyzes asymptotic convergence through stochastic approximations, where convergence can be associated with the limit points of an ordinary-differential equation (ODE). However, analyzing global convergence through an ODE-approximation requires the existence of a potential function, which naturally restricts the analysis to a fine class of games. To overcome these limitations, an alternative framework is proposed for analyzing asymptotic convergence that is based upon an explicit characterization of the invariant probability measure of the induced Markov chain (i.e., stochastic stability). We further describe a methodology for computing the invariant probability measure in positive-utility games, together with an illustration in the context of coordination games.

In the second part of this seminar, I will briefly present an experimental study of this class of dynamics in the context of resource allocation for massively parallel applications in many-core computing platforms. Comparison is performed with the standard Linux scheduler.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAC.2019.2895300, IEEE Transactions on Automatic Control

Stochastic Stability of Perturbed Learning Automata in Positive-Utility Games

Georgios C. Chasparis

***Abstract*—This paper considers a class of reinforcement-based learning (namely, *perturbed learning automata*) and provides a stochastic-stability analysis in repeatedly-played, positive-utility, finite strategic-form games. Prior work in this class of learning dynamics primarily analyzes asymptotic convergence through stochastic approximations, where convergence can be associated with the limit points of an ordinary-differential equation (ODE). However, analyzing global convergence through an ODE-approximation requires the existence of a Lyapunov or a potential function, which naturally restricts the analysis to a fine class of games. To overcome these limitations, this paper introduces an alternative framework for analyzing asymptotic convergence that is based upon an explicit characterization of the invariant probability measure of the induced Markov chain. We further provide a methodology for computing the invariant probability measure in positive-utility games, together with an illustration in the context of coordination games.**

I. Introduction

Recently, multi-agent formulations have been utilized to tackle distributed optimization problems, since communication and computational complexity might be an issue under centralized schemes. In such formulations, decisions are usually taken in a *repeated* fashion, where agents select their next actions based on their *own* prior experience. In the case of finite number of actions for each agent, such multi-agent interactions can be designed as strategic-form games, where agents are repeatedly involved in a strategic interaction with a fixed *payoff* or *utility* function. Such framework finds numerous applications, including, for example, the problem of distributed overlay routing [2], distributed topology control [3] and distributed resource allocation [4].

Given the repeated fashion of the involved strategic interactions in such formulations, several questions naturally emerge, including: a) *Can agents "learn" to asymptotically select optimal actions?*, b) *What information should agents share with each other?*, and c) *What is the computational complexity of the learning process?* Under the scope of engineering applications, it is usually desirable that each agent shares minimum amount of information with other agents, while the computational complexity of the learning process is small. Naturally, *payoff-based learning* has drawn significant attention. Under such class of learning dynamics, each agent receives *only* measurements of its own utility function, while the details of this function (i.e., its mathematical formula) are unknown. Furthermore, each agent cannot access the actions selected or utilities received by other agents.

In such repeatedly-played strategic-form games, a popular objective for payoff-based learning is to guarantee convergence (in some sense) to Nash equilibria. Convergence to Nash equilibria may be desirable, especially when the set of optimal centralized solutions belongs to the set of Nash equilibria.

Reinforcement-based learning has been utilized in strategic-form games in order for agents to gradually learn to play Nash equilibria. It may appear under alternative forms, including discrete-time replicator dynamics [5], learning automata [6], [7] and Q-learning [8]. In all these classes of learning dynamics, deriving conditions under which convergence to Nash equilibria is achieved may not be a trivial task especially in the case of large number of agents (as it will be discussed in detail in the forthcoming Section II).

In the present paper, we consider a class of reinforcement-based learning introduced in [9] that is closely related to both discrete-time replicator dynamics and learning automata. We will refer to this class of dynamics as *perturbed learning automata*. The main difference with prior reinforcement-based learning schemes lies in a) the step-size sequence, and b) the perturbation (or *mutations*) term. The step-size sequence is assumed constant, thus introducing a fading-memory effect of past experiences in each agent's strategy. On the other hand, the perturbation term introduces errors in the selection process of each agent. Both these two features can be used for designing a desirable asymptotic behavior.

We provide an analytical framework for deriving conclusions over the asymptotic behavior of the dynamics that is based upon an explicit characterization of the invariant probability measure of the induced Markov chain. In particular, we show that in all finite strategic-form games satisfying the *Positive-Utility Property* (i.e., games with strictly positive utilities), the support of the invariant probability measure coincides with the set of pure strategy profiles. Furthermore, we provide a methodology for computing the set of stochastically stable states in all positive-utility games. We illustrate

An earlier version of parts of this paper appeared in [1]. This work has been partially 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.

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

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAC.2019.2895300, IEEE Transactions on Automatic Control

this methodology in the context of coordination games and provide a simulation study in distributed network formation. This illustration is also of independent interest since it extends prior work in coordination games under reinforcement-based learning, where convergence to mixed strategy profiles may only be excluded under strong conditions in the utility function (e.g., existence of a potential function).

In the remainder of the paper, Section II presents the investigated class of learning dynamics, related work and the main contributions. Section III provides a simplification in the characterization of stochastic stability, while Section IV presents its technical derivation. This result is utilized for computing the stochastically stable states in positive-utility games in Section V. In Section VI, we present an illustration of the proposed methodology in the context of coordination games, together with a simulation study in distributed network formation. Finally, Section VII presents concluding remarks.

Notation:

- For a Euclidean topological space $\mathcal{Z} \subset \mathbb{R}^n$, let $\mathcal{N}_\delta(x)$ denote the δ-neighborhood of $x \in \mathcal{Z}$, i.e., $\mathcal{N}_\delta(x) \doteq \{y \in \mathcal{Z} : |x-y| < \delta\}$, where $|\cdot|$ denotes the Euclidean distance.
- e_j denotes the *unit vector* in $\mathbb{R}^n$ where its jth entry is equal to 1 and all other entries are equal to 0.
- $\Delta(n)$ denotes the *probability simplex* of dimension n, i.e., $\Delta(n) \doteq \left\{x \in \mathbb{R}^n : x \geq 0, \mathbf{1}^{\mathrm{T}} x = 1\right\}$.
- For some set A in a topological space $\mathcal{Z}$, let $\mathbb{I}_A : \mathcal{Z} \to \{0,1\}$ denote the index function, i.e.,

$$\mathbb{I}_A(x) \doteq \begin{cases} 1 & \text{if } x \in A, \\ 0 & \text{else.} \end{cases}$$

- For a finite set A, $|A|$ denotes its cardinality.
- For a finite set A and any probability distribution $\sigma \in \Delta(|A|)$, the random selection of an element of A will be denoted by $\mathrm{rand}_\sigma[A]$. If $\sigma = (1/|A|, ..., 1/|A|)$, the random selection will be denoted by $\mathrm{rand}_{\mathrm{unif}}[A]$.
- $\boldsymbol{\delta}_x$ denotes the Dirac measure at x.
- $\log(\cdot)$ denotes the natural logarithm.

II. Perturbed Learning Automata

A. Terminology

We consider the standard setup of *finite strategic-form games*. Consider a finite set of *agents* (or *players*) $\mathcal{I} = \{1, ..., n\}$, and let each agent i have a finite set of actions $\mathcal{A}_i$. Let $\alpha_i \in \mathcal{A}_i$ denote any such action of agent i. The set of *action profiles* is the Cartesian product $\mathcal{A} \doteq \mathcal{A}_1 \times \cdots \times \mathcal{A}_n$ and let $\alpha = (\alpha_1, ..., \alpha_n)$ be a representative element of this set. We will denote $-i$ to be the complementary set $\mathcal{I} \backslash i$ and often decompose an action profile as follows $\alpha = (\alpha_i, \alpha_{-i})$. The *payoff/utility function* of agent i is a mapping $u_i(\cdot) : \mathcal{A} \to \mathbb{R}$. A *finite strategic-form game* is defined by the triple $\langle \mathcal{I}, \mathcal{A}, \{u_i(\cdot)\}_i \rangle$.

TABLE I
Perturbed Learning Automata

At fixed time instances $t = 1, 2, ...$, and for each agent $i \in \mathcal{I}$, the following steps are executed recursively. Let $\alpha_i(t)$ and $x_i(t)$ denote the current action and strategy of agent i, respectively.

1) (***action update***) Agent i selects a new action $\alpha_i(t+1)$ as follows:

$$\alpha_i(t+1) = \begin{cases} \mathrm{rand}_{x_i(t)}[\mathcal{A}_i], & \text{with probability } 1-\lambda, \\ \mathrm{rand}_{\mathrm{unif}}[\mathcal{A}_i], & \text{with probability } \lambda, \end{cases} \quad (1)$$

for some small perturbation factor $\lambda > 0$.

2) (***evaluation***) Agent i applies its new action $\alpha_i(t+1)$ and receives a measurement of its utility $u_i(\alpha(t+1)) > 0$.
3) (***strategy update***) Agent i revises its strategy $x_i \in \Delta(|\mathcal{A}_i|)$ as follows:

$$\begin{aligned} & x_i(t+1) \\ & \quad = \quad x_i(t) + \epsilon \cdot u_i(\alpha(t+1)) \cdot [e_{\alpha_i(t+1)} - x_i(t)] \\ & \quad \doteq \quad \mathcal{R}_i(\alpha(t+1), x_i(t)), \end{aligned} \quad (2)$$

for some constant step-size $\epsilon > 0$.

For the remainder of the paper, we will be concerned with finite strategic-form games that satisfy the ***Positive-Utility Property***.

Property 2.1 (Positive-Utility Property): For any agent $i \in \mathcal{I}$ and any action profile $\alpha \in \mathcal{A}$, $u_i(\alpha) > 0$.

This property is rather generic and applies to a large family of games. For example, games at which some form of alignment of interests exists between agents (e.g., *coordination games* [10] or *weakly-acyclic games* [11]), can be designed to satisfy this property, since agents' utilities/preferences are rather close to each other at any given action profile. However, in the forthcoming analysis, we do not impose any structural constraint but Property 2.1.

B. Perturbed Learning Automata

We consider a form of reinforcement-based learning that belongs to the general class of *learning automata* [7]. In learning automata, each agent updates a finite probability distribution $x_i \in \mathcal{X}_i \doteq \Delta(|\mathcal{A}_i|)$ representing its beliefs about the most profitable action.

The proposed learning model is described in Table I. At the first step, each agent i updates its action given its current strategy vector $x_i(t)$. Its selection is slightly perturbed by a perturbation (or *mutations*) factor $\lambda > 0$, such that, with a small probability λ agent i follows a uniform strategy (or, it *trembles*). At the second step, agent i evaluates its new selection by collecting a utility measurement, while in the last step, agent i updates its strategy vector.

Here, we identify actions $\mathcal{A}_i$ with vertices of the simplex, $\{e_1, ..., e_{|\mathcal{A}_i|}\}$. For example, if agent i selects its jth action at time t, then $e_{\alpha_i(t)} \equiv e_j$. To better see how the strategies evolve, let us consider the following toy example. Let the current strategy of agent i be $x_i(t) = \begin{pmatrix} 1/2 & 1/2 \end{pmatrix}^{\mathrm{T}}$, i.e.,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAC.2019.2895300, IEEE Transactions on Automatic Control

agent i has two actions, each assigned probability $1/2$. Let also $\alpha_i(t+1)=1$, i.e., agent i selects the first action according to rule (1). Then, the new strategy vector for agent i is:

$$x_i(t+1)=1/2\begin{pmatrix}1+\epsilon u_i(\alpha(t+1))\\ 1-\epsilon u_i(\alpha(t+1))\end{pmatrix}.$$

Note that the strategy of the selected action increased by an amount that is proportional to the reward received. In other words, the dynamics reinforce repeated selection, and the reinforcement size, $\epsilon u_i(\alpha(t+1))$, depends on the reward received.

By playing a strategic-form game repeatedly over time, players do not always experience the same reward when selecting the same action, since other players may also change their actions. This dynamic element of the reinforcement size is the factor that complicates its convergence analysis, as it will become clear in the forthcoming Section II-C.

Note that by letting the step-size ϵ to be sufficiently small and since the utility function $u_i(\cdot)$ is uniformly bounded in $\mathcal{A}$, $x_i(t)\in\Delta(|\mathcal{A}_i|)$ for all t.

In case $\lambda=0$, the above update recursion will be referred to as the *unperturbed learning automata*.

C. Related work

In this section, we provide a short overview of alternative payoff-based learning schemes specifically designed for repeatedly-played strategic-form games with a *finite* set of actions and a *fixed* utility function for each player. We have identified four main classes of payoff-based dynamics under such structural assumptions, namely *discrete-time replicator dynamics*, *learning automata*, *Q-learning*, and *aspiration-based learning*. Note that payoff-based learning has also been applied to static games with continuous action sets, e.g., extremum-seeking control [12], [13] or actor-critic reinforcement learning [14]. The focus here instead is only on *finite* action sets.

Discrete-time replicator dynamics: A type of learning dynamics which is quite closely related to the dynamics of Table I is the discrete-time version of *replicator dynamics* (cf., [15]). There have been several variations with respect to the selection of the step-size sequence. For example, Arthur [5] considered a similar rule, with $\lambda=0$ and step-size $\epsilon_i(t)=1/(ct^{\nu}+u_i(\alpha(t+1))$, for some positive constant c and for $\nu\in(0,1)$ (in the place of the constant step-size ϵ of (2)). A comparative model is also used by Hopkins and Posch in [16], with $\epsilon_i(t)=1/(V_i(t)+u_i(\alpha(t+1)))$, where $V_i(t)$ is the accumulated benefits of agent i up to time t, which gives rise to the urn process of Erev-Roth [17]. Some similarities are also shared with the Cross' learning model of [18], where $\epsilon_i(t)=1$ and $u_i(\alpha(t))\leq 1$, and its modification presented by Leslie in [19], where $\epsilon(t)$, instead, is decreasing with time.

The main difference of the proposed dynamics of Table I lies in the perturbation parameter $\lambda>0$ which was first introduced and analyzed in [9]. A state-dependent perturbation term has also been investigated in [20]. The perturbation parameter may serve as an equilibrium selection mechanism, since it may exclude convergence to *action profiles that are not Nash equilibria* (briefly, *non-Nash action profiles*) [9]. It resolved one of the main issues of discrete-time replicator dynamics, that is the positive probability of convergence to non-Nash action profiles.

Although excluding convergence to non-Nash action profiles can be guaranteed by sufficiently small $\lambda>0$, establishing convergence to action profiles that are Nash equilibria (*pure Nash equilibria*) may still be an issue. This is desirable in the context of coordination games [21], where Pareto-efficient outcomes are usually pure Nash equilibria (see, e.g., the definition of a coordination game in [10]). As shown in [20], convergence to pure Nash equilibria can be guaranteed only under strong conditions in the utility function. For example, as shown in [20, Proposition 8], and under the ODE-method for stochastic approximations, it requires a) the existence of a potential function, and b) conditions over the Jacobian matrix of the potential function. Even if a potential function does exist, verifying conditions (b) is practically infeasible for games of more than 2 players [20].

On the other hand, an important side-benefit of using this class of dynamics is the indirect "filtering" of the utility-function measurements (through the formulation of the strategy vectors in (2)). This is demonstrated, for example, in [16] for the Erev-Roth model [17], where the robustness of convergence/non-convergence asymptotic results is presented under the presence of noise in the utility measurements.

Learning automata: Learning automata, as first introduced by [6], have been used to the control of complex systems due to their simple structure and low computational complexity (cf., [7, Chapter 1]). *Variable-structure stochastic automata* may incorporate a form of reinforcement of favorable actions, similarly to the replicator dynamics discussed above. An example is the *linear reward-inaction scheme* [7, Chapter 4]. Comparing it with the reinforcement rule of (2), the linear reward-inaction scheme accepts a utility of the form $u_i(\alpha)\in\{0,1\}$, where 0 corresponds to an unfavorable response and 1 corresponds to a favorable one. More general forms can also be used when the utility function may accept discrete or continuous values in the unit interval $[0,1]$.

Analysis of learning automata in games has been restricted to zero-sum and identical-interest games [7], [22]. In identical interest games, convergence analysis has been derived for small number of players and actions, due to the difficulty in deriving conditions for *absolute monotonicity*, which corresponds to the property that *the expected utility received by each player increases monotonically in time* (cf., [7, Definition 8.1]). Similar are the results presented in [22].

The property of *absolute monotonicity* is closely related to the existence of a *potential function*, as in the case of potential

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAC.2019.2895300, IEEE Transactions on Automatic Control

games [23]. Similarly to the discrete-time replicator dynamics, convergence to non-Nash action profiles cannot be excluded when the step-size sequence is constant, even if the utility function satisfies $u_i(\alpha) \in [0, 1]$. (The behavior under decreasing step-size is different as [20, Proposition 2] has shown.) Furthermore, deriving conditions for excluding convergence to mixed strategy profiles in coordination games continues to be an issue, as in discrete-time replicator dynamics.

Recognizing these issues, reference [24] introduced a class of linear reward-inaction schemes in combination with a coordinated exploration phase so that convergence to the efficient (pure) Nash equilibrium is achieved. However, coordination of the exploration phase requires communication between the players, an approach that does not fit to the distributed nature of dynamics pursued here.

Q-learning: Similar questions of convergence to Nash equilibria also appear in alternative reinforcement-based learning formulations, such as approximate dynamic programming and Q-learning. Usually, under Q-learning, players keep track of the discounted running average reward received by each action, based on which optimal decisions are made (see, e.g., [25]). Convergence to Nash equilibria can be accomplished under a stronger set of assumptions, which increases the computational complexity of the dynamics. For example, in the Nash-Q learning algorithm of [8], it is indirectly assumed that agents need to have full access to the joint action space and the rewards received by other agents.

More recently, reference [26] introduced a Q-learning scheme in combination with either adaptive play or better-reply dynamics in order to attain convergence to Nash equilibria in potential games [23] or weakly-acyclic games. However, this form of dynamics requires that each player observes the actions selected by the other players, since a Q-value needs to be assigned to each joint action.

When the evaluation of the Q-values is totally independent, as in the individual Q-learning in [25], then convergence to Nash equilibria has been shown only for 2-player zero-sum games and 2-player partnership games with countably many Nash equilibria. Currently, there exist no convergence results in multi-player games. To overcome this deficiency of Q-learning, in the context of stochastic dynamic games, reference [27] employs an additional feature (motivated by [11]), namely *exploration phases*. In any such *exploration phase*, *all* agents use constant policies, something that allows for an accurate computation of the optimal Q-factors. We may argue that the introduction of common exploration phases for all agents partially destroys the distributed nature of the dynamics, since it requires synchronization between agents.

Aspiration-based learning: Recently, there have been several attempts to establish convergence to Nash equilibria through alternative payoff-based learning dynamics, e.g., the *benchmark-based dynamics* of [11] for convergence to Nash equilibria in weakly-acyclic games, the *trial-and-error learning* [28] for convergence to Nash equilibria in generic games, the *mood-based dynamics* of [29] for maximizing welfare in generic games and the *aspiration learning* in [10] for convergence to efficient outcomes in coordination games. We will refer to such approaches as *aspiration-based learning*. For these types of dynamics, convergence to Nash equilibria or efficient outcomes can be established without requiring any strong monotonicity properties (as in the multi-player weakly-acyclic games in [11]).

The case of noisy utility measurements, which are present in many engineering applications, has not currently been addressed through aspiration-based learning. The only exception is reference [11], under benchmark-based dynamics, where (synchronized) *exploration phases* are introduced through which each agent plays a fixed action for the duration of the exploration phase. If such exploration phases are large in duration (as required by the results in [11]), this may reduce the robustness of the dynamics to dynamic changes in the environment (e.g., changes in the utility function). One reason that such robustness analysis is currently not possible in this class of dynamics is the fact that decisions are taken directly based on the measured performances (e.g., by comparing the currently measured performance with the benchmark performance in [11]).

D. Contributions

The aforementioned literature in payoff-based learning dynamics in finite strategic-form games can be grouped into two main categories, namely *reinforcement-based learning* (including discrete-time replicator dynamics, learning automata and Q-learning) and *aspiration-based learning*. Summarizing their main advantages/disadvantages, we may argue the following high-level observations.

(O1) *Strong asymptotic convergence guarantees* for large number of players, even for generic games, are currently possible under aspiration-based learning. Similar results in reinforcement-based learning are currently restricted to games of small number of players and under strong structural assumptions (e.g., the existence of a potential function). See, for example, the discussion on discrete-time replicator dynamics or learning automata in [20], or the discussion on Q-learning in [27].

(O2) *Noisy observations* can be "handled" through reinforcement-based learning due to the indirect *filtering* of the observation signals (e.g., through the formulation of the strategy-vector in the dynamics of Table I or through the formulation of the Q factors in Q-learning). This is demonstrated, for example, in the convergence/non-convergence asymptotic results presented in [16] for a variation of the proposed learning dynamics of Table I (with $\lambda = 0$ and decreasing ϵ) and under the presence of noise. Similar effects in

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAC.2019.2895300, IEEE Transactions on Automatic Control

aspiration-based learning can currently be achieved only through the introduction of *synchronized exploration phases*, as discussed in Section II-C.

Motivated by these two observations (O1)–(O2), and the obvious inability of reinforcement-based learning to provide strong asymptotic convergence guarantees in large games, this paper advances the asymptotic convergence guarantees for a class of reinforcement-based learning described in Table I. Our goal is to go beyond common restrictions of small number of players and strong assumptions in the game structure (such as the existence of a potential function).

The proposed dynamics (also *perturbed learning automata*) were first introduced in [9] to resolve stability issues in the boundary of the domain appearing in prior schemes [5], [16]. This was achieved through the introduction of the perturbation factor λ of Table I. However, strong convergence guarantees (e.g., w.p.1 convergence to Nash equilibria or efficient outcomes) is currently limited to small number of players and under strict structural assumptions, e.g., the existence of a potential function and conditions on its Jacobian matrix [20].

In this paper, we drop the assumption of a decreasing step-size sequence, and instead we consider the case of a *constant* step-size $\epsilon > 0$. Such selection increases the adaptivity of the dynamics to varying conditions (e.g., the number of agents or the utility function). Furthermore, we provide a stochastic-stability analysis that provides a detailed characterization of the invariant probability measure of the induced Markov chain. In particular, our contributions are as follows:

(C1) We provide an equivalent finite-dimensional characterization of the infinite-dimensional induced Markov chain of the dynamics, that simplifies significantly the computation of its invariant probability measure. This simplification is based upon a weak-convergence result and it applies to any finite strategic-form game with the Positive-Utility Property 2.1 (*Theorem 3.1*).

(C2) We capitalize on this simplification and provide a methodology for computing stochastically stable states in positive-utility finite strategic-form games (*Theorem 5.1*).

(C3) We illustrate the utility of this methodology in establishing stochastic stability in a class of coordination games with no restriction on the number of players or actions (*Theorem 6.1*).

These contributions significantly extend the utility of reinforcement-based learning given observation (O1). Note that (C2) does not impose any structural assumptions other than the positive-utility property. Furthermore, (C3) is of independent interest. To the best of our knowledge, (C3) is the first convergence result in the context of reinforcement-based learning in repeatedly-played finite strategic-form games with the following features: a) a completely distributed setup (i.e., without any information exchange between players), b) more than two players, and c) a weakly-acyclicity condition that does not require the existence of a potential function.

The derived convergence results may not be as strong as the ones currently derived under aspiration-based learning, as discussed in Section II-C. However, reinforcement-based learning may better incorporate noisy observations (as discussed in observation (O2)). Moreover, additional features may allow for stronger convergence guarantees, even to Pareto-efficient outcomes, as presented in [9]. Given the simplified analytical framework presented here, the prospects of even stronger convergence guarantees are promising.

This paper is an extension over an earlier version appeared in [1], which only focused on contribution (C1) above.

III. Stochastic Stability

In this section, we provide a characterization of the *invariant probability measure* μ_λ of the induced Markov chain P_λ of the dynamics of Table I. The importance lies in an equivalence relation (established through a weak-convergence argument) of μ_λ with an invariant distribution of a finite-state Markov chain. Characterization of the stochastic stability of the dynamics will follow directly due to the Birkhoff's individual ergodic theorem. This simplification in the characterization of μ_λ will be the first important step for providing specialized results for stochastic stability in strategic-form games.

A. Terminology and notation

Let $\mathcal{Z} \doteq \mathcal{A} \times \mathcal{X}$, where $\mathcal{X} \doteq \mathcal{X}_1 \times \ldots \times \mathcal{X}_n$, i.e., pairs of joint actions α and strategy profiles x. We will denote the elements of the state space $\mathcal{Z}$ by z.

The set $\mathcal{A}$ is endowed with the discrete topology, $\mathcal{X}$ with its usual Euclidean topology, and $\mathcal{Z}$ with the corresponding product topology. We also let $\mathfrak{B}(\mathcal{Z})$ denote the Borel σ-field of $\mathcal{Z}$, and $\mathfrak{P}(\mathcal{Z})$ the set of *probability measures* (p.m.) on $\mathfrak{B}(\mathcal{Z})$ endowed with the Prohorov topology, i.e., the topology of weak convergence. The learning algorithm of Table I defines an $\mathcal{Z}$-valued Markov chain. Let $P_\lambda : \mathcal{Z} \times \mathfrak{B}(\mathcal{Z}) \to [0,1]$ denote its transition probability function (t.p.f.), parameterized by $\lambda > 0$. We refer to the process with $\lambda > 0$ as the *perturbed process*. Let also $P : \mathcal{Z} \times \mathfrak{B}(\mathcal{Z}) \to [0,1]$ denote the t.p.f. of the *unperturbed process*, i.e., when $\lambda = 0$. We also define the t-step t.p.f. $P^t : \mathcal{Z} \times \mathfrak{B}(\mathcal{Z}) \to [0,1]$ recursively as:

$$P^t(z, D) = \int_{\mathcal{Z}} P(z, dy) P^{t-1}(y, D).$$

We let $\mathcal{C}_b(\mathcal{Z})$ denote the Banach space of real-valued continuous functions on $\mathcal{Z}$ under the sup-norm (denoted by $\|\cdot\|_\infty$) topology. For $f \in \mathcal{C}_b(\mathcal{Z})$, define

$$P_\lambda f(z) \doteq \int_{\mathcal{Z}} P_\lambda(z, dy) f(y),$$

and

$$\mu[f] \doteq \int_{\mathcal{Z}} \mu(dz) f(z), \text{ for } \mu \in \mathfrak{P}(\mathcal{Z}).$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAC.2019.2895300, IEEE Transactions on Automatic Control

The unperturbed process governed by the t.p.f. P will be denoted by $Z \doteq \{Z_t : t \geq 0\}$. Let $\Omega \doteq \mathcal{Z}^\infty$ denote the canonical path space, i.e., an element $\omega \in \Omega$ is a sequence $\{\omega(0), \omega(1), \dots\}$, with $\omega(t) = (\alpha(t), x(t)) \in \mathcal{Z}$. We use the same notation for the elements (α, x) of the space $\mathcal{Z}$ and for the coordinates of the process $Z_t = (\alpha(t), x(t))$. Let also $\mathbb{P}_z[\cdot]$ denote the unique p.m. induced by the unperturbed process P on the product σ-field of $\mathcal{Z}^\infty$, initialized at $z = (\alpha, x)$, and $\mathbb{E}_z[\cdot]$ the corresponding expectation operator. Let also $\theta : \Omega \to \Omega$ denote the *shift operator*, such that $(Z \circ \theta_t)(\omega) \doteq Z(\theta_t(\omega)) = \{Z_t, Z_{t+1}, ...\}$. Furthermore, for $D \in \mathfrak{B}(\mathcal{Z})$, let $\tau(D)$ be the first hitting time of the unperturbed process to D, i.e., $\tau(D) \doteq \inf\{t \geq 0 : Z_t \in D\}$.

B. Stochastic stability

First, we note that both P and P_λ ($\lambda > 0$) satisfy the *weak Feller property* (cf., [30, Definition 4.4.2]).

Proposition 3.1: Both the unperturbed process P ($\lambda = 0$) and the perturbed process P_λ ($\lambda > 0$) have the weak Feller property.

Proof. See Appendix A. □

The measure $\mu_\lambda \in \mathfrak{P}(\mathcal{Z})$ is called an *invariant probability measure* (i.p.m.) for P_λ if

$$(\mu_\lambda P_\lambda)(A) \doteq \int_{\mathcal{Z}} \mu_\lambda(dz) P_\lambda(z, A) = \mu_\lambda(A), \qquad A \in \mathfrak{B}(\mathcal{Z}).$$

Since $\mathcal{Z}$ defines a locally compact separable metric space and P, P_λ have the weak Feller property, they both admit an i.p.m., denoted μ and μ_λ, respectively [30, Theorem 7.2.3].

We would like to characterize the *stochastically stable states* $z \in \mathcal{Z}$ of P_λ, that is any state $z \in \mathcal{Z}$ for which any collection of i.p.m.'s $\{\mu_\lambda \in \mathfrak{P}(\mathcal{Z}) : \mu_\lambda P_\lambda = \mu_\lambda, \lambda > 0\}$ satisfies $\liminf_{\lambda \to 0} \mu_\lambda(z) > 0$. As the forthcoming analysis will show, the stochastically stable states will be a subset of the set of *pure strategy states* (p.s.s.) defined as follows:

Definition 3.1 (Pure Strategy State): A pure strategy state is a state $s = (\alpha, x) \in \mathcal{Z}$ such that for all $i \in \mathcal{I}$, $x_i = e_{\alpha_i}$, i.e., x_i coincides with the vertex of the probability simplex $\Delta(|\mathcal{A}_i|)$ which assigns probability 1 to action α_i.

We will denote the set of pure strategy states by $\mathcal{S}$.

Theorem 3.1 (Stochastic Stability): There exists a unique probability vector $\pi = (\pi_1, ..., \pi_{|\mathcal{S}|})$ such that for any collection of i.p.m.'s $\{\mu_\lambda \in \mathfrak{P}(\mathcal{Z}) : \mu_\lambda P_\lambda = \mu_\lambda, \lambda > 0\}$, the following hold:

(a) $\lim_{\lambda \to 0} \mu_\lambda(\cdot) = \hat{\mu}(\cdot) \doteq \sum_{s \in \mathcal{S}} \pi_s \boldsymbol{\delta}_s(\cdot)$, where convergence is in the weak sense.

(b) The probability vector π is an invariant distribution of the (finite-state) Markov process $\hat{P}$, such that, for any $s, s' \in \mathcal{S}$,

$$\hat{P}_{ss'} \doteq \lim_{t \to \infty} Q P^t(s, \mathcal{N}_\delta(s')), \tag{3}$$

for some $\delta > 0$ sufficiently small, where Q is the t.p.f. corresponding to *only one agent trembling* (i.e., following the uniform distribution of (1)).

The proof of Theorem 3.1 requires a series of propositions and it will be presented in detail in Section IV.

Theorem 3.1 implicitly provides a stochastically stability argument. In fact, the expected asymptotic behavior of the dynamics can be characterized by $\hat{\mu}$ and, therefore, π. In particular, by Birkhoff's individual ergodic theorem [30, Theorem 2.3.4], the weak convergence of μ_λ to $\hat{\mu}$, and the fact that μ_λ is ergodic, we have that the expected percentage of time that the process spends in any $O \in \mathcal{B}(\mathcal{Z})$ such that $\partial O \cap \mathcal{S} \neq \varnothing$ is given by $\hat{\mu}(O)$ as the experimentation probability λ approaches zero and time increases, i.e.,

$$\lim_{\lambda \downarrow 0} \left(\lim_{t \to \infty} \frac{1}{t} \sum_{k=0}^{t-1} P_\lambda^k(x, O) \right) = \hat{\mu}(O)\,.$$

C. Discussion

Theorem 3.1 establishes "*equivalence*" (in a weak convergence sense) of the original (perturbed) learning process with a simplified process, where *only one agent trembles at the first iteration and then no agent trembles thereafter*. This simplification in the analysis has originally been capitalized to analyze *aspiration learning* dynamics in [31], [10], and it is based upon the observation that *under the unperturbed process, agents' strategies will converge to a pure strategy state*, as it will be shown in the forthcoming Section IV.

The limiting behavior of the original (perturbed) dynamics is characterized by the (*unique*) invariant distribution of the finite-state Markov chain $\{P_{ss'}\}$, whose states correspond to the pure strategy states $\mathcal{S}$. In other words, *we should expect that as the perturbation parameter λ approaches zero, the algorithm spends the majority of the time on pure strategy states*. The importance of this result lies on the fact that no constraints have been imposed in the payoff matrix/function of the game other than the Positive-Utility Property 2.1.

In the forthcoming Section V, we will use this result to provide a methodology for computing the set of stochastically stable states. Then, in Section VI, this methodology will be illustrated in the context of coordination games.

IV. Technical Derivation

In this section, we provide the main steps for the proof of Theorem 3.1. We begin by investigating the asymptotic behavior of the unperturbed process P, and then we characterize the i.p.m. of the perturbed process with respect to the p.s.s.'s $\mathcal{S}$.

A. Unperturbed Process

For $t \geq 0$ define the sets

$$A_t \doteq \{\omega \in \Omega : \alpha(\tau) = \alpha(t)\,, \text{ for all } \tau \geq t\}\,,$$

$$B_t \doteq \{\omega \in \Omega : \alpha(\tau) = \alpha(0)\,, \text{ for all } 0 \leq \tau \leq t\}\,.$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAC.2019.2895300, IEEE Transactions on Automatic Control

Note that $\{B_t : t \geq 0\}$ is a non-increasing sequence, i.e., $B_{t+1} \subseteq B_t$, while $\{A_t : t \geq 0\}$ is non-decreasing, i.e., $A_{t+1} \supseteq A_t$. Let $A_\infty \doteq \bigcup_{t=0}^{\infty} A_t$ and $B_\infty \doteq \bigcap_{t=1}^{\infty} B_t$. In other words, A_∞ *corresponds to the event that agents eventually play the same action profile, while* B_∞ *corresponds to the event that agents never change their actions*. The following proposition discusses convergence of the unperturbed process to the set of p.s.s.'s, $\mathcal{S}$.

Proposition 4.1 (Convergence to p.s.s.): Let us assume that the step-size $\epsilon > 0$ is sufficiently small such that $0 < \epsilon u_i(\alpha) < 1$ for all $\alpha \in \mathcal{A}$ and $i \in \mathcal{I}$. Then, the following hold:

(a) $\inf_{z\in\mathcal{Z}} \mathbb{P}_z[B_\infty] > 0$,
(b) $\inf_{z\in\mathcal{Z}} \mathbb{P}_z[A_\infty] = 1$.

Proof. See Appendix B. □

Statement (a) of Proposition 4.1 states that *the probability that agents never change their actions is bounded away from zero*, while statement (b) states that *the probability that eventually agents play the same action profile is one*. This also indicates that any invariant measure of the unperturbed process can be characterized with respect to the pure strategy states $\mathcal{S}$, which is established by the following proposition.

Proposition 4.2 (Limiting t.p.f. of unperturbed process): Let μ denote an i.p.m. of P. Then, there exists a t.p.f. Π on $\mathcal{Z} \times \mathfrak{B}(\mathcal{Z})$ with the following properties:

(a) for μ-a.e. $z \in \mathcal{Z}$, $\Pi(z, \cdot)$ is an i.p.m. for P;
(b) for all $f \in \mathcal{C}_b(\mathcal{Z})$, $\lim_{t\to\infty} \|P^t f - \Pi f\|_\infty = 0$;
(c) μ is an i.p.m. for Π;
(d) the support[1] of Π is on $\mathcal{S}$ for all $z \in \mathcal{Z}$.

Proof. The state space $\mathcal{Z}$ is a locally compact separable metric space and the t.p.f. of the unperturbed process P admits an i.p.m. due to Proposition 3.1. Thus, statements (a), (b) and (c) follow directly from [30, Theorem 5.2.2 (a), (b), (e)].

(d) Let us assume that the support of Π includes points in $\mathcal{Z}$ other than the pure strategy states in $\mathcal{S}$. Then, there exists an open set $O \in \mathfrak{B}(\mathcal{Z})$ such that $O \cap \mathcal{S} = \varnothing$ and $\Pi(z^*, O) > 0$ for some $z^* \in \mathcal{Z}$. According to (b), P^t converges weakly to Π. Thus, from Portmanteau theorem (cf., [30, Theorem 1.4.16]), we have that $\liminf_{t\to\infty} P^t(z^*, O) \geq \Pi(z^*, O) > 0$. This is a contradiction of Proposition 4.1(b), which concludes the proof. □

Proposition 4.2 states that the limiting unperturbed t.p.f. converges weakly to a t.p.f. Π which accepts the same i.p.m. as P. Furthermore, *the support of* Π *is the set of pure strategy states* $\mathcal{S}$. This is a rather important observation, since the limiting perturbed process can also be "related" (in a weak-convergence sense) to the t.p.f. Π, as it will be shown in the following section.

[1]The *support* of a measure μ on $\mathcal{Z}$ is the unique closed set $F \subset \mathfrak{B}(\mathcal{Z})$ such that $\mu(\mathcal{Z}\backslash F) = 0$ and $\mu(F \cap O) > 0$ for every open set $O \subset \mathcal{Z}$ such that $F \cap O \neq \varnothing$.

B. Perturbed process

According to the definition of perturbed learning automata of Table I, when an agent updates its action, there is a small probability $\lambda > 0$ that it "*trembles*," i.e., it selects a new action according to a uniform distribution. Thus, we can decompose the t.p.f. induced by the one-step update as follows:

$$P_\lambda = (1 - \varphi(\lambda))P + \varphi(\lambda)Q_\lambda$$

where $\varphi(\lambda) = 1 - (1 - \lambda)^n$ is the probability that at least one agent trembles (since $(1 - \lambda)^n$ is the probability that no agent trembles), and Q_λ corresponds to the t.p.f. when at least one agent trembles. Note that $\varphi(\lambda) \to 0$ as $\lambda \downarrow 0$.

Define also Q as the t.p.f. where *only one* agent trembles, and Q^* as the t.p.f. where *at least two agents tremble*. Then, we may write:

$$Q_\lambda = (1 - \psi(\lambda))Q + \psi(\lambda)Q^*, \tag{4}$$

where $\psi(\lambda) \doteq 1 - \frac{n\lambda(1-\lambda)^{n-1}}{1-(1-\lambda)^n}$ corresponds to the probability that at least two agents tremble given that at least one agent trembles. It also satisfies $\psi(\lambda) \to 0$ as $\lambda \downarrow 0$, which establishes an approximation of Q_λ by Q as the perturbation factor λ approaches zero.

Let us also define the infinite-step t.p.f. when trembling only at the first step (briefly, *lifted* t.p.f.) as follows:

$$P_\lambda^L \doteq \varphi(\lambda) \sum_{t=0}^{\infty} (1 - \varphi(\lambda))^t Q_\lambda P^t = Q_\lambda R_\lambda \tag{5}$$

where $R_\lambda \doteq \varphi(\lambda) \sum_{t=0}^{\infty} (1 - \varphi(\lambda))^t P^t$, i.e., R_λ corresponds to the *resolvent* t.p.f.

In the following proposition, we establish weak-convergence of the lifted t.p.f. P_λ^L to $Q\Pi$ as $\lambda \downarrow 0$, which will further allow for an explicit characterization of the weak limit points of the i.p.m. of P_λ.

Proposition 4.3 (i.p.m. of perturbed process): The following hold:

(a) For $f \in \mathcal{C}_b(\mathcal{Z})$, $\lim_{\lambda\to 0} \|R_\lambda f - \Pi f\|_\infty = 0$.
(b) For $f \in \mathcal{C}_b(\mathcal{Z})$, $\lim_{\lambda\to 0} \|P_\lambda^L f - Q\Pi f\|_\infty = 0$.
(c) Any i.p.m. μ_λ of P_λ is also an i.p.m. of P_λ^L.
(d) Any weak limit point in $\mathfrak{P}(\mathcal{Z})$ of μ_λ, as $\lambda \to 0$, is an i.p.m. of $Q\Pi$.

Proof. (a) For any $f \in \mathcal{C}_b(\mathcal{Z})$, we have

$$\begin{aligned} &\|R_\lambda f - \Pi f\|_\infty \\ &= \|\varphi(\lambda) \sum_{t=0}^{\infty} (1 - \varphi(\lambda))^t P^t f - \Pi f\|_\infty \\ &= \|\varphi(\lambda) \sum_{t=0}^{\infty} (1 - \varphi(\lambda))^t (P^t f - \Pi f)\|_\infty, \end{aligned}$$

where we have used $\varphi(\lambda) \sum_{t=0}^{\infty} (1 - \varphi(\lambda))^t = 1$. Note that

$$\begin{aligned} &\varphi(\lambda) \sum_{t=T}^{\infty} (1 - \varphi(\lambda))^t \|P^t f - \Pi f\|_\infty \\ &\leq (1 - \varphi(\lambda))^T \sup_{t\geq T} \|P^t f - \Pi f\|_\infty. \end{aligned}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAC.2019.2895300, IEEE Transactions on Automatic Control

From Proposition 4.2(b), we have that for any $\delta > 0$, there exists $T = T(\delta) > 0$ such that the r.h.s. is uniformly bounded by δ for all $t \geq T$. Thus, the sequence

$$\Theta_T \doteq \varphi(\lambda) \sum_{t=0}^{T} (1 - \varphi(\lambda))^t (P^t f - \Pi f)$$

is Cauchy and therefore convergent (under the sup-norm). In other words, there exists $\Theta \in \mathbb{R}$ such that $\lim_{T\to\infty} \|\Theta_T - \Theta\|_\infty = 0$. For every $T > 0$, we have

$$\|R_\lambda f - \Pi f\|_\infty \leq \|\Theta_T\|_\infty + \|\Theta - \Theta_T\|_\infty.$$

Note that

$$\|\Theta_T\|_\infty \leq \varphi(\lambda) \sum_{t=0}^{T} (1 - \varphi(\lambda))^t \|P^t f - \Pi f\|_\infty.$$

If we take $\lambda \downarrow 0$, then the r.h.s. converges to zero. Thus,

$$\lim_{\lambda \downarrow 0} \|R_\lambda f - \Pi f\|_\infty \leq \|\Theta - \Theta_T\|_\infty, \text{ for all } T > 0,$$

which concludes the proof.

(b) For any $f \in \mathcal{C}_b(\mathcal{Z})$, we have

$$\begin{aligned} &\|P_\lambda^L f - Q\Pi f\|_\infty \\ &\quad \leq \|Q_\lambda (R_\lambda f - \Pi f)\|_\infty + \|Q_\lambda \Pi f - Q\Pi f\|_\infty \\ &\quad \leq \|R_\lambda f - \Pi f\|_\infty + \|Q_\lambda \Pi f - Q\Pi f\|_\infty. \end{aligned}$$

The first term of the r.h.s. approaches 0 as $\lambda \downarrow 0$ according to (a). The second term of the r.h.s. also approaches 0 as $\lambda \downarrow 0$ since $Q_\lambda \to Q$ as $\lambda \downarrow 0$.

(c) By definition of the perturbed t.p.f. P_λ, we have

$$P_\lambda R_\lambda = (1 - \varphi(\lambda)) P R_\lambda + \varphi(\lambda) Q_\lambda R_\lambda.$$

Note that $Q_\lambda R_\lambda = P_\lambda^L$ and $(1 - \varphi(\lambda)) P R_\lambda = R_\lambda - \varphi(\lambda) I$, where I corresponds to the identity operator. Thus,

$$P_\lambda R_\lambda = R_\lambda - \varphi(\lambda) I + \varphi(\lambda) P_\lambda^L.$$

For any i.p.m. of P_λ, μ_λ, we have

$$\mu_\lambda P_\lambda R_\lambda = \mu_\lambda R_\lambda - \varphi(\lambda) \mu_\lambda + \varphi(\lambda) \mu_\lambda P_\lambda^L,$$

which equivalently implies that $\mu_\lambda = \mu_\lambda P_\lambda^L$, since $\mu_\lambda P_\lambda = \mu_\lambda$. We conclude that μ_λ is also an i.p.m. of P_λ^L.

(d) Let $\hat{\mu}$ denote a weak limit point of μ_λ as $\lambda \downarrow 0$. To see that such a limit exists, take $\hat{\mu}$ to be an i.p.m. of P. Then,

$$\begin{aligned} &\|P_\lambda f - P f\|_\infty \\ &\quad \geq \|\mu_\lambda (P_\lambda f - P f)\|_\infty \\ &\quad = \|(\mu_\lambda - \hat{\mu})(I - P)[f]\|_\infty. \end{aligned}$$

Thus, the weak convergence of P_λ to P implies that $\mu_\lambda \Rightarrow \hat{\mu}$. Note further that

$$\begin{aligned} &\hat{\mu}[f] - \hat{\mu}[Q\Pi f] \\ &\quad = (\hat{\mu}[f] - \mu_\lambda[f]) + \mu_\lambda[P_\lambda^L f - Q\Pi f] + \\ &\quad\quad (\mu_\lambda[Q\Pi f] - \hat{\mu}[Q\Pi f]). \end{aligned}$$

The first and the third term of the r.h.s. approaches 0 as $\lambda \downarrow 0$ due to the fact that $\mu_\lambda \Rightarrow \hat{\mu}$. The same holds for the second term of the r.h.s. due to part (b). Thus, we conclude that any weak limit point of μ_λ as $\lambda \downarrow 0$ is an i.p.m. of $Q\Pi$. □

Proposition 4.3 establishes convergence (in a weak sense) of the i.p.m. μ_λ of the perturbed process to an i.p.m. of $Q\Pi$. In the following section, this convergence result will allow for a more explicit characterization of μ_λ as $\lambda \downarrow 0$.

C. Equivalent finite-state Markov process

Define the finite-state Markov process $\hat{P}$ as in (3).

Proposition 4.4 (Unique i.p.m. of $Q\Pi$): There exists a unique i.p.m. $\hat{\mu}$ of $Q\Pi$. It satisfies

$$\hat{\mu}(\cdot) = \sum_{s \in \mathcal{S}} \pi_s \boldsymbol{\delta}_s(\cdot) \tag{6}$$

for some constants $\pi_s \geq 0$, $s \in \mathcal{S}$. Moreover, $\pi = (\pi_1, ..., \pi_{|\mathcal{S}|})$ is an invariant distribution of $\hat{P}$, i.e., $\pi = \pi \hat{P}$.

Proof. From Proposition 4.2(d), we know that the support of Π is the set of pure strategy states $\mathcal{S}$. Thus, the support of $Q\Pi$ is also on $\mathcal{S}$. From Proposition 4.3, we know that $Q\Pi$ admits an i.p.m., say $\hat{\mu}$, whose support is also $\mathcal{S}$. Thus, $\hat{\mu}$ admits the form of (6), for some constants $\pi_s \geq 0$, $s \in \mathcal{S}$.

For any two distinct $s, s' \in \mathcal{S}$, note that $\mathcal{N}_\delta(s')$, $\delta > 0$, is a continuity set of $Q\Pi(s, \cdot)$, i.e., $Q\Pi(s, \partial\mathcal{N}_\delta(s')) = 0$. Thus, from Portmanteau theorem, given that $QP^t \Rightarrow Q\Pi$,

$$Q\Pi(s, \mathcal{N}_\delta(s')) = \lim_{t\to\infty} QP^t(s, \mathcal{N}_\delta(s')) = \hat{P}_{ss'}.$$

If we also define $\pi_s \doteq \hat{\mu}(\mathcal{N}_\delta(s))$, then

$$\pi_{s'} = \hat{\mu}(\mathcal{N}_\delta(s')) = \sum_{s \in \mathcal{S}} \pi_s Q\Pi(s, \mathcal{N}_\delta(s')) = \sum_{s \in \mathcal{S}} \pi_s \hat{P}_{ss'},$$

which shows that π is an invariant distribution of $\hat{P}$.

It remains to establish uniqueness of the invariant distribution of $Q\Pi$. Note that the set $\mathcal{S}$ of pure strategy states is isomorphic with the set $\mathcal{A}$ of action profiles. If agent i trembles (as t.p.f. Q dictates), then all actions in $\mathcal{A}_i$ have positive probability of being selected, i.e., $Q(\alpha, (\alpha_i', \alpha_{-i})) > 0$ for all $\alpha_i' \in \mathcal{A}_i$ and $i \in \mathcal{I}$. It follows by Proposition 4.1 that $Q\Pi(\alpha, (\alpha_i', \alpha_{-i})) > 0$ for all $\alpha_i' \in \mathcal{A}_i$ and $i \in \mathcal{I}$. Finite induction then shows that $(Q\Pi)^n(\alpha, \alpha') > 0$ for all $\alpha, \alpha' \in \mathcal{A}$. It follows that if we restrict the domain of $Q\Pi$ to $\mathcal{S}$, it defines an irreducible stochastic matrix. Therefore, $Q\Pi$ has a unique i.p.m. □

D. Proof of Theorem 3.1

Theorem 3.1(a)–(b) is a direct implication of Propositions 4.3–4.4.

V. Stochastically Stable States

In this section, we capitalize on Theorem 3.1 and we further simplify the computation of the stochastically stable states in strategic-form games satisfying Property 2.1.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAC.2019.2895300, IEEE Transactions on Automatic Control

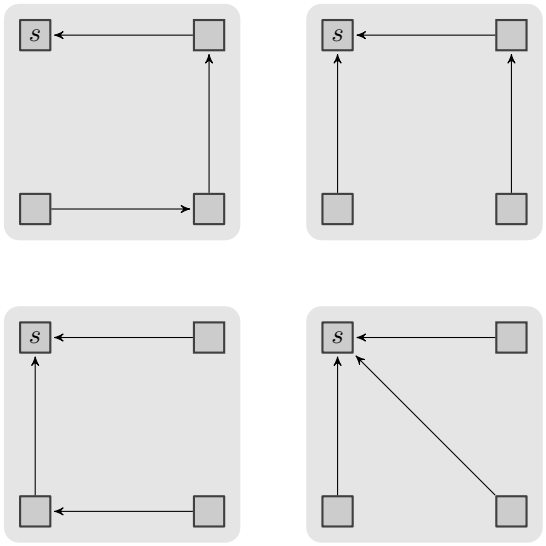

Fig. 1. Examples of s-graphs in case $\mathcal{S}$ contains four states.

A. Background on finite Markov chains

In order to compute the invariant distribution of a finite-state, irreducible and aperiodic Markov chain, we are going to consider a characterization introduced by [32]. In particular, for finite Markov chains an invariant measure can be expressed as the ratio of sums of products consisting of transition probabilities. These products can be described conveniently by means of graphs on the set of states of the chain. In particular, let $\mathcal{S}$ be a finite set of states, whose elements will be denoted by s_k, s_ℓ, etc., and let a subset $\mathcal{W}$ of $\mathcal{S}$.

Definition 5.1: ($\mathcal{W}$-graph) A graph consisting of arrows $s_k \to s_\ell$ ($s_k \in \mathcal{S}\backslash\mathcal{W}, s_\ell \in \mathcal{S}, s_\ell \neq s_k$) is called a $\mathcal{W}$-graph if it satisfies the following conditions:

1) every point $k \in \mathcal{S}\backslash\mathcal{W}$ is the initial point of exactly one arrow;
2) there are no closed cycles in the graph; or, equivalently, for any point $s_k \in \mathcal{S}\backslash\mathcal{W}$ there exists a sequence of arrows leading from it to some point $s_\ell \in \mathcal{W}$.

Fig. 1 provides examples of $\{s\}$-graphs for some state $s \in \mathcal{S}$ when $\mathcal{S}$ contains four states. We will denote by $\mathcal{G}\{\mathcal{W}\}$ the set of $\mathcal{W}$-graphs and we shall use the letter g to denote graphs. If $\hat{P}_{s_k s_\ell}$ are nonnegative numbers, where $s_k, s_\ell \in \mathcal{S}$, define also the transition probability along path g as

$$\varpi(g) \doteq \prod_{(s_k \to s_\ell) \in g} \hat{P}_{s_k s_\ell}.$$

The following Lemma holds:

Lemma 5.1 (Lemma 6.3.1 in [32]): Let us consider a Markov chain with a finite set of states $\mathcal{S}$ and transition probabilities $\{\hat{P}_{s_k s_\ell}\}$ and assume that every state can be reached from any other state in a finite number of steps. Then, the stationary distribution of the chain is $\pi = [\pi_s]$, where

$$\pi_s = \frac{R_s}{\sum_{s_i \in \mathcal{S}} R_{s_i}}, s \in \mathcal{S} \tag{7}$$

where $R_s \doteq \sum_{g \in \mathcal{G}\{s\}} \varpi(g)$.

In other words, in order to compute the weight that the stationary distribution assigns to a state $s \in \mathcal{S}$, it suffices to compute the ratio of the transition probabilities of all $\{s\}$-graphs over the transition probabilities of all graphs.

B. Approximation of one-step transition probability

We wish to provide an approximation in the computation of the transition probabilities under $\hat{P}$, in order to explicitly compute the stationary distribution π of Theorem 3.1. Based on the definition of the t.p.f. $Q\Pi$, and as $\lambda \downarrow 0$, a transition from s to s' influences the stationary distribution only if s differs from s' in the action of a *single* agent. This observation will be capitalized by the forthcoming Lemmas 5.2–5.3, to approximate the transition probability from s to s' under $\hat{P}$.

Let us define $\tau_s^*(D)$ to be the minimum number of steps that the process $Q\Pi$ needs in order to reach D when starting from $s \in \mathcal{S}$ (i.e., the minimum possible first hitting time to D).

Lemma 5.2 (One-step transition probability): Consider any two action profiles $\alpha, \alpha' \in \mathcal{A}$ which differ in the action of a single agent j, and let $s, s' \in \mathcal{S}$ be the p.s.s.'s associated with α and α', respectively. Set $z' = (\alpha', x')$, where $x'_j \doteq e_{\alpha_j} + \epsilon u_j(\alpha')(e_{\alpha'_j} - e_{\alpha_j})$, which corresponds to the state after agent j perturbed once starting from s and played α'_j. Define also

$$\breve{P}_{ss'}(\delta) \doteq \mathbb{P}_{z'}[\tau(\mathcal{N}_\delta(s')) \leq \infty]$$

which corresponds to the probability that the process eventually reaches $\mathcal{N}_\delta(s')$ starting from z'. For sufficiently small ϵ such that $0 < \epsilon u_j(\alpha') < 1$, the following hold:

(a) The transition probability from s to s' under $Q\Pi$ can be approximated as follows:

$$\hat{P}_{ss'} = \gamma_j \cdot \lim_{\delta \downarrow 0} \breve{P}_{ss'}(\delta), \tag{8}$$

where $\gamma_j \doteq 1/(n\,|\mathcal{A}_j|)$ corresponds to the probability that agent j trembled and selected action α'_j under Q.

(b) $\breve{P}_{ss'}(\delta)$ coincides with the probability of the *shortest path*, i.e.,

$$\breve{P}_{ss'}(\delta) = \mathbb{P}_{z'}\left[\alpha(t+1) = \alpha', \forall t < \tau_s^*(\mathcal{N}_\delta(s'))\right].$$

(c) There exists negative constant $\eta(\delta)$, such that for any transition step $s \to s'$ (with the above properties) and for sufficiently small $\epsilon > 0$,

$$\breve{P}_{ss'}(\delta) \approx \exp\left(\frac{\eta(\delta)}{\epsilon u_j(\alpha')}\right). \tag{9}$$

Proof. See Appendix C. □

Note that for sufficiently small ϵ, *the larger the destination utility $u_j(\alpha')$, the larger the transition probability to s'*, since $\eta(\delta) < 0$. In a way, the inverse of the destination utility at s' represents a measure of "resistance" of the process to transit to

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAC.2019.2895300, IEEE Transactions on Automatic Control

s'. Lemma 5.2 provides a tool for simplifying the computation of stochastically stable pure strategy states as it will become apparent in the following section.

C. Approximation of stationary distribution

In this section, using Lemma 5.2 that approximates one-step transition probabilities, we provide an approximation of the i.p.m. of the t.p.f. $Q\Pi$. By definition of $Q\Pi$, this approximation is based upon the observation that for the computation of the quantities R_s of Lemma 5.1, it suffices to consider only those paths in $\mathcal{G}\{s\}$ which involve *one-step* transitions as defined in Lemma 5.2.

Define $\mathcal{G}^{(1)}\{s\} \subseteq \mathcal{G}\{s\}$ to be the set of s-graphs consisting solely of one-step transitions, i.e., for any $g \in \mathcal{G}^{(1)}\{s\}$ and any arrow $(s_k \to s_\ell) \in g$, the associated action profiles, say $\alpha^{(k)}, \alpha^{(\ell)}$, respectively, differ in a single action of a single agent. It is straightforward to check that $\mathcal{G}^{(1)}\{s\} \neq \varnothing$ for any $s \in \mathcal{S}$.

Lemma 5.3 (Approximation of stationary distribution): The stationary distribution of the finite Markov chain $\{\hat{P}_{s_k s_\ell}\}$, $\pi = [\pi_s]$, satisfies

$$\pi_s = \lim_{\delta \downarrow 0} \frac{\breve{R}_s(\delta)}{\sum_{s_i \in \mathcal{S}} \breve{R}_{s_i}(\delta)}, \qquad s \in \mathcal{S}, \tag{10}$$

where $\breve{R}_s(\delta) \doteq \sum_{g \in \mathcal{G}^{(1)}\{s\}} \breve{\varpi}(g;\delta)$, and

$$\breve{\varpi}(g;\delta) \doteq \bar{\gamma}_g \prod_{(s_k \to s_\ell) \in g} \breve{P}_{s_k s_\ell}(\delta), \tag{11}$$

for some constant $\bar{\gamma}_g \in (0,1)$.

Proof. According to Lemma 5.1, for any $s \in \mathcal{S}$, we have $\pi_s = R_s / \sum_{s_i \in \mathcal{S}} R_{s_i}$. Given the definition of the t.p.f. Q, where only one agent trembles, we should only consider one-step transition probabilities (as defined in Lemma 5.2). Thus,

$$R_s = \sum_{g \in \mathcal{G}^{(1)}\{s\}} \varpi(g) = \sum_{g \in \mathcal{G}^{(1)}\{s\}} \prod_{(s_k \to s_\ell) \in g} \hat{P}_{s_k s_\ell}.$$

According to Lemma 5.2 and Equation (8), we have

$$\begin{aligned} R_s &= \lim_{\delta \downarrow 0} \sum_{g \in \mathcal{G}^{(1)}\{s\}} \prod_{(s_k \to s_\ell) \in g} \gamma_{j(s_k,s_\ell)} \breve{P}_{s_k s_\ell}(\delta) \\ &= \lim_{\delta \downarrow 0} \sum_{g \in \mathcal{G}^{(1)}\{s\}} \bar{\gamma}_g \prod_{(s_k \to s_\ell) \in g} \breve{P}_{s_k s_\ell}(\delta) \end{aligned}$$

where $j(s_k, s_\ell)$ denotes the single agent whose action changes from s_k to s_ℓ, and $\bar{\gamma}_g \doteq \prod_{(s_k \to s_\ell) \in g} \gamma_{j(s_k,s_\ell)} \in (0,1)$. Thus, the conclusion follows. □

Note that Lemma 5.3 provides a simplification to Theorem 3.1, since it suffices to compute the transition probabilities of the $\mathcal{W}$-graphs consisting solely of one step transitions. Furthermore, the transition probability of any such graph, $\breve{\varpi}(g;\delta)$, can be computed by Lemma 5.2, which provides an explicit formula for one-step transitions. In the following section, the computation of the stationary distribution will further be simplified and related to the resistance of one-step transitions.

D. δ-resistance

We have shown in Lemma 5.2, that the one-step transition probability $\breve{P}_{ss'}(\delta)$ increases as the destination utility increases. Informally, *the inverse destination utility at s' represents a form of "resistance" to approaching state s'*. In this section, we will formalize this notion.

Definition 5.2 (δ-resistance): For a pure strategy state $s \in \mathcal{S}$, let us consider any graph $g \in \mathcal{G}^{(1)}\{s\}$. For any $\delta > 0$, the δ-resistance associated with $s \in \mathcal{S}$ in graph g, is defined as follows:

$$\varphi_\delta(s|g) \doteq \sum_{(s_k \to s_\ell) \in g} \frac{1}{\epsilon u_j(\alpha^{(\ell)})}. \tag{12}$$

In other words, the δ-resistance of a state s along a graph g corresponds to the sum of the inverse utilities of the destination states along that graph, scaled by $1/\epsilon$. We further denote by $\varphi^*_\delta(s)$ the minimum δ-resistance, i.e., $\varphi^*_\delta(s) \doteq \min_{g \in \mathcal{G}^{(1)}\{s\}} \varphi_\delta(s|g)$ and by $g^*(s)$ the $\{s\}$-graph that attains this minimum resistance.

E. Stochastically stable states

The stochastically stable states can be identified as the states of minimum resistance.

Theorem 5.1 (Stochastically stable states): As $\epsilon \downarrow 0$, the set of stochastically stable p.s.s.'s $\mathcal{S}^*$ is such that, for any $\delta > 0$

$$\overline{\Phi}_\delta(\mathcal{S}^*) \doteq \max_{s^* \in \mathcal{S}^*} \varphi^*_\delta(s^*) < \min_{s \in \mathcal{S} \backslash \mathcal{S}^*} \varphi^*_\delta(s) \doteq \underline{\Phi}_\delta(\mathcal{S} \backslash \mathcal{S}^*). \tag{13}$$

Proof. By Lemmas 5.2–5.3, for any state $s \in \mathcal{S}$ and for any graph $g \in \mathcal{G}^{(1)}\{s\}$, we have that, as $\epsilon \downarrow 0$,

$$\breve{\varpi}(g;\delta) = \bar{\gamma}_g \prod_{(s_k \to s_\ell) \in g} \breve{P}_{s_k s_\ell}(\delta) \approx \bar{\gamma}_g e^{\eta(\delta)\varphi_\delta(s|g)},$$

and

$$\breve{R}_s(\delta) = \sum_{g \in \mathcal{G}^{(1)}\{s\}} \bar{\gamma}_g e^{\eta(\delta)\varphi_\delta(s|g)}.$$

Thus, for the states in $\mathcal{S} \backslash \mathcal{S}^*$, and for sufficiently small $\epsilon \downarrow 0$, we have

$$\sum_{s \in \mathcal{S} \backslash \mathcal{S}^*} \breve{R}_s(\delta) = e^{\eta(\delta)\underline{\Phi}_\delta(\mathcal{S} \backslash \mathcal{S}^*)} \cdot \sum_{s \in \mathcal{S} \backslash \mathcal{S}^*} \sum_{g \in \mathcal{G}^{(1)}\{s\}} \bar{\gamma}_g e^{\eta(\delta)(\varphi_\delta(s|g) - \underline{\Phi}_\delta(\mathcal{S} \backslash \mathcal{S}^*))}.$$

Note that the second part of the r.h.s. approaches a finite value as $\epsilon \downarrow 0$, since $\varphi_\delta(s|g) \geq \underline{\Phi}_\delta(\mathcal{S} \backslash \mathcal{S}^*)$ for each $s \in \mathcal{S} \backslash \mathcal{S}^*$. Analogously, for the states in $\mathcal{S}^*$, we have

$$\sum_{s \in \mathcal{S}^*} \breve{R}_s(\delta) = e^{\eta(\delta)\overline{\Phi}_\delta(\mathcal{S}^*)} \cdot \sum_{s \in \mathcal{S}^*} \sum_{g \in \mathcal{G}^{(1)}\{s\}} \bar{\gamma}_g e^{\eta(\delta)(\varphi_\delta(s|g) - \overline{\Phi}_\delta(\mathcal{S}^*))}.$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAC.2019.2895300, IEEE Transactions on Automatic Control

Thus, for sufficiently small ϵ,

$$\frac{\sum_{s\in\mathcal{S}\backslash\mathcal{S}^*}\breve{R}_s(\delta)}{\sum_{s\in\mathcal{S}^*}\breve{R}_s(\delta)} = e^{\eta(\delta)(\underline{\Phi}_\delta(\mathcal{S}\backslash\mathcal{S}^*)-\overline{\Phi}_\delta(\mathcal{S}^*))} \frac{\sum_{s\in\mathcal{S}\backslash\mathcal{S}^*}\sum_{g\in\mathcal{G}^{(1)}\{s\}}\bar{\gamma}_g e^{\eta(\delta)(\varphi_\delta(s|g)-\underline{\Phi}_\delta(\mathcal{S}\backslash\mathcal{S}^*))}}{\sum_{s\in\mathcal{S}^*}\sum_{g\in\mathcal{G}^{(1)}\{s\}}\bar{\gamma}_g e^{\eta(\delta)(\varphi_\delta(s|g)-\overline{\Phi}_\delta(\mathcal{S}^*))}}.$$

Given that $\underline{\Phi}_\delta(\mathcal{S}\backslash\mathcal{S}^*)-\overline{\Phi}_\delta(\mathcal{S}^*)>0$, the first part of the r.h.s. approaches 0 as $\epsilon\downarrow 0$. Also, the numerator of the ratio of the r.h.s. approaches a finite value, due to the definition of $\underline{\Phi}_\delta(\mathcal{S}^*)$. On the other hand, each term of the denominator approaches either a finite value or ∞ as $\epsilon\downarrow 0$. Thus,

$$\frac{\sum_{s\in\mathcal{S}\backslash\mathcal{S}^*}\breve{R}_s(\delta)}{\sum_{s\in\mathcal{S}^*}\breve{R}_s(\delta)} \xrightarrow{\epsilon\downarrow 0} 0. \tag{14}$$

Denote by $\pi_{\mathcal{S}^*}$ the probability assigned by the stationary distribution π to $\mathcal{S}^*$. Then, according to (10), we have:

$$\begin{aligned}\lim_{\epsilon\downarrow 0}\pi_{\mathcal{S}^*} &= \lim_{\epsilon\downarrow 0}\lim_{\delta\downarrow 0}\frac{\sum_{s^*\in\mathcal{S}^*}\breve{R}_{s^*}(\delta)}{\sum_{s\in\mathcal{S}}\breve{R}_s(\delta)} = \lim_{\delta\downarrow 0}\lim_{\epsilon\downarrow 0}\frac{\sum_{s^*\in\mathcal{S}^*}\breve{R}_{s^*}(\delta)}{\sum_{s\in\mathcal{S}}\breve{R}_s(\delta)}\\ &= \lim_{\delta\downarrow 0}\lim_{\epsilon\downarrow 0}\frac{1}{1+\sum_{s\in\mathcal{S}\backslash\mathcal{S}^*}\breve{R}_s(\delta)/\sum_{s^*\in\mathcal{S}^*}\breve{R}_{s^*}(\delta)}.\end{aligned}$$

Note that the interchange of limits in the second equality is valid due to the finiteness of the limits of the transition probabilities (according to Lemma 5.3). Given (14), we conclude that $\lim_{\epsilon\downarrow 0}\pi_{\mathcal{S}^*}=1$. Conversely, $\lim_{\epsilon\downarrow 0}\pi_{\mathcal{S}\backslash\mathcal{S}^*}=0$. Thus, the stochastically stable states may only be contained in $\mathcal{S}^*$. □

In other words, Theorem 5.1 says that, in order for a p.s.s. set $\mathcal{S}^*$ to be stochastically stable, it suffices to show that for any $s\in\mathcal{S}^*$ there exists a $\{s\}$-graph with strictly smaller δ-resistance from any other state $s'\in\mathcal{S}\backslash\mathcal{S}^*$. Note that this theorem applies to any game that satisfies the positive-utility property. In the following section, we illustrate the utility of Theorem 5.1 in computing the stochastically stable states in coordination games.

VI. Illustration in Coordination Games

A. Stochastic stability

In this section, we will be using the notion of *best response* of a agent i into an action profile $\alpha=(\alpha_i,\alpha_{-i})$, as well as the notion of *Nash equilibrium*. In particular, we define:

Definition 6.1 (Best response): The best response of a player i to an action profile $\alpha=(\alpha_i,\alpha_{-i})$ is defined as the following set of actions: $\mathrm{BR}_i(\alpha)\doteq\arg\max_{a\in\mathcal{A}_i}u_i(a,\alpha_{-i})$.

Definition 6.2 (Nash equilibrium): An action profile $\alpha^*=(\alpha_i^*,\alpha_{-i}^*)$ is a Nash equilibrium, if for every player i, $\alpha_i^*\in\mathrm{BR}_i(\alpha^*)$.

A best-response of a player i to an action profile will often be denoted by α_i^*. Note that, according to the above definition, the best response of a player is never empty. We also introduce the following notion of a coordination game.

Definition 6.3 (Coordination game): A strategic-form game satisfying the positive-utility property (Property 2.1) is a coordination game if, for every action profile α and player i, $u_j(\alpha_i',\alpha_{-i})\geq u_j(\alpha_i,\alpha_{-i})$ for any $\alpha_i'\in\mathrm{BR}_i(\alpha)$.

In other words, a coordination game is such that at any action profile, if a player plays a best response, then no other player gets worse-off. This is satisfied by default when the current action profile is a Nash equilibrium, since a player's best response is to play the same action.

In order to address stochastic stability, we will further need to introduce the notion of the best-BR (briefly, BBR).

Definition 6.4 (Best-BR): Let $i^*:\mathcal{A}\to\mathcal{I}$ be defined as:

$$i^*(\alpha)\doteq\arg\max_{i\in\mathcal{I}}\left\{u_i(\alpha_i,\alpha_{-i}):\alpha_i\in\mathrm{BR}_i(\alpha)\right\}.$$

The one-step transition $\alpha=(\alpha_{i^*},\alpha_{-i^*})\to(\alpha_{i^*}^*,\alpha_{-i^*})$, where $\alpha_{i^*}^*\in\mathrm{BR}_{i^*}(\alpha)$, is the best-BR to the current action profile α and will briefly be denoted by $\mathrm{BBR}(\alpha)$.

In other words, $\mathrm{BBR}(\alpha)$ corresponds to the one-step transition, where the player which changes its action receives the largest utility among all possible one-step transitions from α.

Lemma 6.1: Let $\mathcal{S}_{\mathrm{NE}}$ be the set of p.s.s.'s which correspond to the set of pure Nash equilibria. In any coordination game, the $\{\mathcal{S}_{\mathrm{NE}}\}$-graph that attains the minimum δ-resistance is: $g^*(\mathcal{S}_{\mathrm{NE}})=\left\{(s_k\to s_\ell):\alpha^{(\ell)}\in\mathrm{BBR}(\alpha)\right\}$.

Proof. Under the coordination property, and starting from any state $s\notin\mathcal{S}_{\mathrm{NE}}$, we can construct a path starting from s and leading to $\mathcal{S}_{\mathrm{NE}}$ that consists only of one-step best-BR's. Such a path will include no cycles (since the utility of all players may not decrease along such path). Furthermore, such path of best-BR's may only terminate at a Nash equilibrium.

By Definition 5.1 of a $\{\mathcal{S}_{\mathrm{NE}}\}$-graph, a state $s\notin\mathcal{S}_{\mathrm{NE}}$ is the source of exactly one arrow. Among the possible arrows with source s, the one that corresponds to a best-BR is the one with the minimum δ-resistance (since it provides the maximum possible destination utility). We conclude that the $\{\mathcal{S}_{\mathrm{NE}}\}$-graph(s) consisting only of best-BR's provide the minimum δ-resistance. □

Lemma 6.1 shows that the $\{\mathcal{S}_{\mathrm{NE}}\}$-graph of minimum δ-resistance is the graph consisting of the one-step best-BR's starting from any non-Nash action profile. Using this property, we can show that the set of Nash equilibria are the stochastically stable states in any coordination game.

Theorem 6.1 (Stochastic stability in coordination games): In any coordination game of Definition 6.3, as $\epsilon\downarrow 0$ and $\lambda\downarrow 0$, the stochastically stable pure strategy states satisfy $\mathcal{S}^*\subseteq\mathcal{S}_{\mathrm{NE}}$.

Proof. It suffices to show that all p.s.s.'s outside $\mathcal{S}_{\mathrm{NE}}$ provide a δ-resistance which is strictly higher than the δ-resistance of any Nash equilibrium in $\mathcal{S}_{\mathrm{NE}}$ (as Theorem 5.1 dictates).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAC.2019.2895300, IEEE Transactions on Automatic Control

Consider an action profile α which is not a Nash equilibrium and the corresponding p.s.s. s. Consider the part of the optimal $\{\mathcal{S}_{\rm NE}\}$-graph which leads to s, i.e.,

$$g^*(s|\mathcal{S}_{\rm NE}) \doteq \{(s_k \to s_\ell) \in g^*(\mathcal{S}_{\rm NE}) : \exists \text{ path from } s_\ell \text{ to } s\}.$$

In other words, $g^*(s|\mathcal{S}_{\rm NE})$ corresponds to the part of the minimum-resistance graph $g^*(\mathcal{S}_{\rm NE})$ whose arrows lead to s. This graph might be empty if s is not a recipient of any arrow in $g^*(\mathcal{S}_{\rm NE})$. For the remainder of the proof, define the graphs: $g_1 \doteq g^*(\mathcal{S}_{\rm NE}) \backslash g^*(s|\mathcal{S}_{\rm NE})$, $g_2 \doteq g^*(s) \backslash g^*(s|\mathcal{S}_{\rm NE})$. Note that, $g^*(s|\mathcal{S}_{\rm NE}) \subset g^*(s)$, i.e., the graph that leads to s through the minimum resistance graph of $\mathcal{S}_{\rm NE}$ is also part of the minimum resistance graph of s. By construction, we also have $g^*(s|\mathcal{S}_{\rm NE}) \subset g^*(\mathcal{S}_{\rm NE})$. Thus, the exact same arrows (i.e., the ones in $g^*(s|\mathcal{S}_{\rm NE})$) are subtracted from $g^*(\mathcal{S}_{\rm NE})$ and $g^*(s)$ to define the graphs g_1 and g_2, respectively.

By definition of the $\{\mathcal{S}_{\rm NE}\}$-graphs, a node within the set $\{\mathcal{S}_{\rm NE}\}$ cannot be the source of any arrow in g_1. Similarly, node s may not be the source of any arrow in g_2. Since $|\mathcal{S}_{\rm NE}| \geq 1$, and the fact that only a single arrow may stem from any given node, we conclude that $|g_1| \leq |g_2|$, i.e., g_2 contains at least as many arrows as g_1.

Furthermore, by construction of graphs g_1 and g_2, there exists at least one node $s' \notin \mathcal{S}_{\rm NE}$ with the following property: $(s' \to s'') \in g_1$ such that $\alpha'' \in {\rm BBR}(\alpha')$, and $(s' \to s''') \in g_2$ such that $\alpha''' \notin {\rm BBR}(\alpha')$. This is due to the fact that any path in g_2 should eventually lead to $s \notin \mathcal{S}_{\rm NE}$.

Thus, we conclude that g_2 contains at least as many arrows as g_1, and g_2 contains arrows which are not best-BR steps. Since only best-BR transition steps achieve the minimum resistance, we conclude that $\varphi(s|g_2) > \varphi(s|g_1)$, which implies that any $\{s\}$-graph may only have larger δ-resistance as compared to the minimum δ-resistance of $g^*(\mathcal{S}_{\rm NE})$. □

B. Simulation study in distributed network formation

In this section, we perform a simulation study of the proposed learning dynamics in a class of network formation games [33]. We consider n nodes deployed on the plane and assume that the set of actions of each node or agent i, $\mathcal{A}_i$, contains all possible combinations of neighbors of i, denoted N_i, with which a link can be established, i.e., $\mathcal{A}_i = 2^{N_i}$, including the empty set. Links are considered unidirectional, and a link established by node i with node j, denoted (j, i), starts at j with the arrowhead pointing to i. A *graph* G is defined as a collection of nodes and directed links. Define also a *path* from j to i as a sequence of nodes and directed links that starts at j and ends to i following the orientation of the graph, i.e.,

$$(j \to i) = \big\{j = j_0, (j_0, j_1), j_1, \ldots, (j_{m-1}, j_m), j_m = i\big\}$$

for some positive integer m. In a *connected* graph, there is a path from any node to any other node.

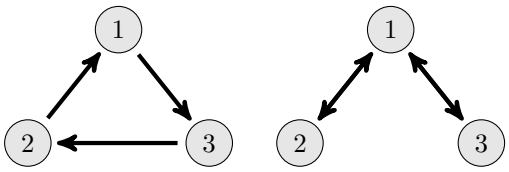

Fig. 2. Nash networks in case of $n = 3$ agents and $0 < \nu < 1$.

Let us consider a utility function $u_i : \mathcal{A} \to \mathbb{R}$, such that

$$u_i(\alpha) \doteq \sum_{j \in \mathcal{I} \backslash \{i\}} \chi_\alpha(j \to i) - \kappa\, |\alpha_i|\,, \tag{15}$$

$i \in \mathcal{I}$, where $|\alpha_i|$ denotes the number of links corresponding to α_i and κ is a constant in $(0, 1)$. Also,

$$\chi_\alpha(j \to i) \doteq \begin{cases} 1 & \text{if } (j \to i) \subseteq G_\alpha\,, \\ 0 & \text{otherwise,} \end{cases}$$

where G_α denotes the graph induced by joint action α. As it was shown in Proposition 4.2 in [33], *a network G^* is a Nash equilibrium if and only if it is* critically connected*, i.e., i) it is connected, and ii) for any $(s, i) \in G$, $(s \to i)$ is the unique path from s to i*. For example, the Nash equilibria for $n = 3$ agents and unconstrained neighborhoods are shown in Fig. 2.

Proposition 6.1: The network formation game defined by (15) is a coordination game.

Proof. First, note that any network formation game with the utility of (15) satisfies the positive-utility property. This is due to the fact that for any single link of cost $\kappa \in (0, 1)$, an agent receives utility of at least 1. For any joint action $\alpha \notin \mathcal{A}^*$ assume that a node i picks its best response. Then no other agent becomes worse off, since a best response of any node i always retains connectivity. Note that this is not necessarily true for any other change in actions. Thus, the coordination property of Definition 6.3 is satisfied. □

Fig. 3 depicts the response of the learning dynamics in the network formation game. We consider 6 nodes deployed on the plane, where the neighbors of each node are defined as the two immediate nodes (e.g., the neighbors of node 1 are $N_1 = \{2, 6\}$). According to Theorem 6.1, in order for the average behavior to be observed, λ and ϵ need to be sufficiently small. We choose: $\epsilon = \lambda = 0.005$, and $\kappa = 1/2$.

Given the large number of actions, we do not plot the strategy vector for each node. Instead, we plot the inverse total distance from each node to its neighboring nodes. In a wheel structure (and only under this structure), the inverse total distance to the neighboring nodes is equal to $1/1+5 = 1/6 \approx 0.167$. The wheel structure is among the Nash equilibria of this game (as shown in [33]) and the unique payoff-dominant equilibrium (i.e., every node receives its maximum utility). The wheel structure is the emergent structure as shown in Fig. 3.

The simulation of Fig. 3 verifies Theorem 6.1, since convergence (in a weak sense) is attained to the set of Nash equilibria.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAC.2019.2895300, IEEE Transactions on Automatic Control

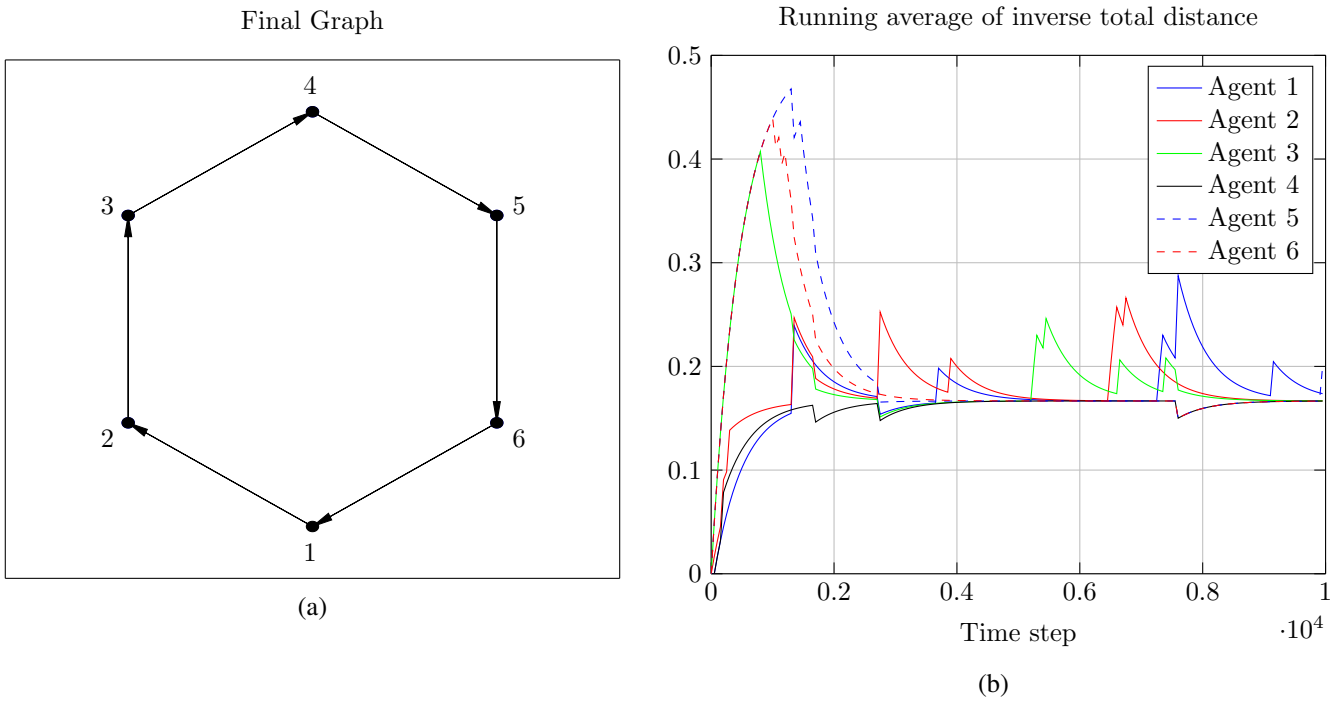

Fig. 3. (a) Final graph and (b) running-average inverse total distance with time under the perturbed learning automata dynamics of Table I when applied to the network formation game.

However, it also demonstrates the potential of this class of dynamics for stronger convergence results, since the emergent Nash equilibrium is also payoff-dominant.

VII. Conclusions & Future Work

In this paper, we considered a class of reinforcement-based learning dynamics that belongs to the family of discrete-time replicator dynamics and learning automata, and we provided an explicit characterization of the invariant probability measure of the induced Markov chain. Through this analysis, we demonstrated convergence (in a weak sense) to the set of pure strategy states, overcoming prior limitations of the ODE-method for stochastic approximations, such as the existence of a potential function. Furthermore, we provided a simplified methodology for computing the set of stochastically stable states, and we demonstrated its utility in the context of coordination games. This is the first result in this class of dynamics that demonstrates global convergence properties with no restrictions in the number of players and without requiring the existence of a potential function. Thus, it opens up new possibilities for the use of reinforcement-based learning in distributed control of multi-agent systems.

Appendix A
Proof of Proposition 3.1

Let us first consider the perturbed process P_λ. Let us also consider any sequence $\{z^{(k)} = (\alpha^{(k)}, x^{(k)})\}$ such that $z^{(k)} \to z = (\alpha, x) \in \mathcal{Z}$. For any open set $O \in \mathfrak{B}(\mathcal{Z})$,

$$\begin{aligned} &P_\lambda(z^{(k)} = (\alpha^{(k)}, x^{(k)}), O) \\ &\quad = \sum_{\alpha \in \mathcal{P}_{\mathcal{A}}(O)} \Big\{ \prod_{i=1}^{n} \tilde{x}_{i\alpha_i}^{(k)} \cdot \prod_{i=1}^{n} \mathbb{P}_{z^{(k)}}[\mathcal{R}_i(\alpha, x_i^{(k)}) \in \mathcal{P}_{\mathcal{X}_i}(O)] \Big\} \\ &\quad = \sum_{\alpha \in \mathcal{P}_{\mathcal{A}}(O)} \Big\{ \prod_{i=1}^{n} \mathbb{I}_{\mathcal{P}_{\mathcal{X}_i}(O)}(\mathcal{R}_i(\alpha, x_i^{(k)})) \tilde{x}_{i\alpha_i}^{(k)} \Big\}, \end{aligned}$$

where $\mathcal{P}_{\mathcal{X}_i}(O)$ and $\mathcal{P}_{\mathcal{A}}(O)$ are the *canonical projections* defined by the product topology, and $\tilde{x}_{i\alpha_i}^{(k)} \doteq (1-\lambda)x_{i\alpha_i}^{(k)} + \lambda/|\mathcal{A}_i|$. Similarly, we have:

$$P_\lambda(z, O) = \sum_{\alpha \in \mathcal{P}_{\mathcal{A}}(O)} \Big\{ \prod_{i=1}^{n} \mathbb{I}_{\mathcal{P}_{\mathcal{X}_i}(O)} \left(\mathcal{R}_i\left(\alpha, x_i\right)\right) \tilde{x}_{i\alpha_i} \Big\}.$$

To investigate the limit of $P_\lambda(z^{(k)}, O)$ as $k \to \infty$, it suffices to investigate the behavior of the sequence $\zeta_i^{(k)} \doteq \mathbb{I}_{\mathcal{P}_{\mathcal{X}_i}(O)}(\mathcal{R}_i(\alpha, x_i^{(k)}))$. We distinguish the following (complementary) cases:

(a) $\mathcal{R}_i(\alpha, x_i) \notin \mathcal{P}_{\mathcal{X}_i}(O)$ and $\mathcal{R}_i(\alpha, x_i) \notin \partial\mathcal{P}_{\mathcal{X}_i}(O)$: In this case, there exists an open ball about the next strategy vector that does not share any common points with $\mathcal{P}_{\mathcal{X}_i}(O)$. Due to the continuity of the function $\mathcal{R}_i(\alpha, \cdot)$, we have that $\zeta_i^{(k)} \to \zeta_i \doteq \mathbb{I}_{\mathcal{P}_{\mathcal{X}_i}(O)}(\mathcal{R}_i(\alpha, x_i)) \equiv 0$.

(b) $\mathcal{R}_i(\alpha, x_i) \in \mathcal{P}_{\mathcal{X}_i}(O)$: In this case, there exists an open ball about the next strategy vector that belongs to $\mathcal{P}_{\mathcal{X}_i}(O)$, since $O \in \mathfrak{B}(\mathcal{Z})$. Due to the continuity of the function $\mathcal{R}_i(\alpha, \cdot)$, we have that $\zeta_i^{(k)} \to \zeta_i = 1$.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAC.2019.2895300, IEEE Transactions on Automatic Control

(c) $\mathcal{R}_i(\alpha, x_i) \notin \mathcal{P}_{\mathcal{X}_i}(O)$ and $\mathcal{R}_i(\alpha, x_i) \in \partial\mathcal{P}_{\mathcal{X}_i}(O)$: In this case, $\zeta_i \equiv 0$. We conclude that $\liminf_{k\to\infty} \zeta_i^{(k)} \geq \zeta_i = 0$, since $\zeta_i^{(k)} \in \{0, 1\}$.

In either one of the above (complementary) cases, we have that $\liminf_{k\to\infty} \zeta_i^{(k)} \geq \zeta_i$. Finally, due to the continuity of the perturbed strategy vector $\tilde{x}_{i\alpha_i}$ with respect to $x_{i\alpha_i}$, we conclude that for any sequence $z^{(k)} \to z$, $\liminf_{k\to\infty} P_\lambda(z^{(k)}, O) \geq P_\lambda(z, O)$. Thus, by [30, Proposition 7.2.1], P_λ satisfies the weak Feller property.

The above derivation can be generalized to any selection probability function $f(x_{i\alpha_i})$ in the place of $\tilde{x}_{i\alpha_i}$, provided that it is a continuous function. Thus, the proof for the unperturbed process P follows the exact same reasoning by simply setting $f(x_{i\alpha_i}) = x_{i\alpha_i}$.

APPENDIX B
PROOF OF PROPOSITION 4.1

(a) Let us consider an action profile $\alpha = (\alpha_1, ..., \alpha_n) \in \mathcal{A}$, and an initial strategy profile $x(0) = (x_1(0), ..., x_n(0))$ such that $x_{i\alpha_i}(0) > 0$ for all $i \in \mathcal{I}$. Note that if the same action profile α is selected consecutively up to time t, then the strategy of agent i satisfies:

$$x_i(t) = e_{\alpha_i} - (1 - \epsilon u_i(\alpha))^t (e_{\alpha_i} - x_i(0)). \tag{16}$$

Given that B_t is non-increasing, from continuity from above we have

$$\mathbb{P}_z[B_\infty] = \lim_{t\to\infty} \mathbb{P}_z[B_t] = \lim_{t\to\infty} \prod_{k=0}^{t} \prod_{i=1}^{n} x_{i\alpha_i}(k). \tag{17}$$

Note that $\mathbb{P}_z[B_\infty] > 0$ if and only if

$$\sum_{t=0}^{\infty} \log(x_{i\alpha_i}(t)) > -\infty, \text{ for all } i \in \mathcal{I}. \tag{18}$$

Let us introduce the variable $y_i(t) \doteq 1 - x_{i\alpha_i}(t)$, which corresponds to the probability of agent i selecting any action other than α_i. Condition (18) is equivalent to

$$-\sum_{t=0}^{\infty} \log(1 - y_i(t)) < \infty, \text{ for all } i \in \mathcal{I}. \tag{19}$$

Note that $y_i(t+1)/y_i(t) = 1 - \epsilon u_i(\alpha) < 1$, which (by the Ratio test, cf., [34, Theorem 6.2.4]) implies that the series of positive terms $\sum_{t=1}^{\infty} y_i(t)$ is convergent. Hence, $\lim_{t\to\infty} y_i(t) = 0$. Thus, from L'Hospital's rule (cf., [35, Theorem 5.13]),

$$\lim_{t\to\infty} \frac{-\log(1 - y_i(t))}{y_i(t)} = \lim_{t\to\infty} \frac{1}{1 - y_i(t)} = 1 > 0. \tag{20}$$

From the Limit Comparison Test (cf., [34, Theorem 6.2.2]), we conclude that condition (19) holds, which equivalently implies that $\mathbb{P}_z[B_\infty] > 0$. Lastly, due to (16), $\mathbb{P}_z[B_\infty]$ is continuous with respect to $x(0)$ which takes values in a bounded and closed set $\mathcal{X}$. Thus, by [34, Theorem 3.2.2], we conclude that $\inf_{z\in\mathcal{Z}} \mathbb{P}_z[B_\infty] > 0$.

(b) Define the set $C_\ell \doteq \left\{z \in \mathcal{Z} : |x_i|_\infty > 1 - \epsilon^\ell, \forall i \in \mathcal{I}\right\}$, where $|x_i|_\infty \doteq \max\{x_{i\alpha_i}, \alpha_i \in \mathcal{A}_i\}$, i.e., C_ℓ corresponds to a strategy being close to a vertex of $\mathcal{X}$. For $\ell > 0$,

$$\begin{aligned}
\mathbb{P}_z[A_t] &\geq \sum_{k=1}^{t} \mathbb{P}_z[\tau(C_\ell) = k\,, Z \circ \theta_k \in B_\infty] \\
&= \sum_{k=1}^{t} \mathbb{P}_z[Z \circ \theta_k \in B_\infty | \tau(C_\ell) = k] \cdot \mathbb{P}_z[\tau(C_\ell) = k] \\
&\geq \inf_{z\in C_\ell} \mathbb{P}_z[B_\infty] \cdot \sum_{k=1}^{t} \mathbb{P}_z[\tau(C_\ell) = k] \\
&\geq \inf_{z\in C_\ell} \mathbb{P}_z[B_\infty] \cdot \inf_{z\in C_\ell^c} \mathbb{P}_z[\tau(C_\ell) \leq t],
\end{aligned} \tag{21}$$

where the second inequality is due to the Markov property. Consider the subsequence $t_k = k\ell^m$, for some $m = m(\ell) > 0$ such that, the time block of ℓ^m iterations is sufficiently large so that C_ℓ can be reachable from any state in C_ℓ^c. Then,

$$\mathbb{P}_z[\tau(C_\ell) \leq t_k | \tau(C_\ell) > t_{k-1}] \geq \inf_{z\in C_\ell^c} \mathbb{P}_z[B_{\ell^m}] \geq \inf_{z\in C_\ell^c} \mathbb{P}_z[B_\infty],$$

where the last inequality is due to (17). Given (a), and for any $\ell > 0$, the r.h.s. of the above inequality is bounded away from zero. Hence, from the counterpart of the Borel-Cantelli Lemma (cf., [36, Section 3.3]) and the fact that $\{\tau(C_\ell) \leq t_k\} \subseteq \{\tau(C_\ell) \leq t_{k+1}\}$, we have that, for any $\ell > 0$,

$$\lim_{k\to\infty} \inf_{z\in C_\ell^c} \mathbb{P}_z[\tau(C_\ell) \leq t_k] = 1. \tag{22}$$

Finally, set $k = \ell$. Then, $t_k = t_\ell = \ell^{m+1}$. Given (21)–(22) and from continuity from below, we have

$$\mathbb{P}_z[A_\infty] = \lim_{\ell\to\infty} \mathbb{P}[A_{t_\ell}] \geq \lim_{\ell\to\infty} \inf_{z\in C_\ell} \mathbb{P}_z[B_\infty] = 1,$$

where the last equality is due to the definition of C_ℓ and (17).

APPENDIX C
PROOF OF LEMMA 5.3

(a) The state z', realized after agent j trembled and played α'_j starting from s, is uniquely defined as $z' \doteq (\alpha', e_{\alpha_j} + \epsilon u_j(\alpha')(e_{\alpha'_j} - e_{\alpha_j}))$. Thus, we can write:

$$\begin{aligned}
&QP^t(s, \mathcal{N}_\delta(s')) \\
&\quad = \int_{\mathcal{Z}} \gamma_j \boldsymbol{\delta}_{z'}(dy) P^t(y, \mathcal{N}_\delta(s')) = \gamma_j P^t(z', \mathcal{N}_\delta(s')).
\end{aligned}$$

Given that $\mathcal{N}_\delta(s')$ is a continuity set of $Q\Pi(s, \cdot)$, from Portmanteau theorem we have that, for any $\delta > 0$,

$$\hat{P}_{ss'} = Q\Pi(s, \mathcal{N}_\delta(s')) = \gamma_j \lim_{t\to\infty} P^t(z', \mathcal{N}_\delta(s')).$$

Note also that $P^t(z', \mathcal{N}_\delta(s')) \leq \mathbb{P}_{z'}[\tau(\mathcal{N}_\delta(s')) \leq t]$. Since the sequence of events $\{\tau(\mathcal{N}_\delta(s')) \leq t\}_t$ is non-decreasing, then from continuity from below, we have that, for any $\delta > 0$,

$$\lim_{t\to\infty} P^t(z', \mathcal{N}_\delta(s')) \leq \mathbb{P}_{z'}[\tau(\mathcal{N}_\delta(s')) \leq \infty]. \tag{23}$$

On the other hand, we have

$$P^t(z', \mathcal{N}_\delta(s')) \geq \sum_{k=1}^{t} \mathbb{P}_{z'}[\tau(\mathcal{N}_\delta(s')) = k, Z \circ \theta_k \in B_\infty]$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAC.2019.2895300, IEEE Transactions on Automatic Control

$$\geq \inf_{z\in\mathcal{N}_\delta(s')} \mathbb{P}_z[B_\infty]\cdot\mathbb{P}_{z'}[\tau(\mathcal{N}_\delta(s'))\leq t],$$

where in the second inequality we have used the Markov property. Given that $\lim_{\delta\downarrow 0}\inf_{z\in\mathcal{N}_\delta(s')}\mathbb{P}_z[B_\infty]=1$, we get

$$\lim_{\delta\downarrow 0}\lim_{t\to\infty} P^t(z',\mathcal{N}_\delta(s'))\geq\lim_{\delta\downarrow 0}\mathbb{P}_{z'}[\tau(\mathcal{N}_\delta(s'))\leq\infty]. \quad (24)$$

The conclusion follows directly from (23)–(24).

(b) Consider the unperturbed process initiated at state z', i.e., $Z_0=z'$. Let us also define the set

$$D_{j,\ell}(\alpha')\doteq\left\{(\alpha,x)\in\mathcal{Z}: x_{j\alpha'_j}>1-H_j(\alpha')^\ell\right\},$$

where $H_j(\alpha')\doteq 1-\epsilon u_j(\alpha')$. The set $D_{j,\ell}(\alpha')$ is the unreachable set in the strategy space of agent j when starting from $x_{j\alpha'_j}=0$ and playing action α'_j for ℓ consecutive steps. Define also the set

$$E_{j,\ell}(\alpha')\doteq D_{j,\ell+1}(\alpha')^c\cap D_{j,\ell}(\alpha').$$

One possibility for realizing a transition from z' to $\mathcal{N}_\delta(s')$ is to follow the shortest path, that is, the path of playing action α' consecutively. Thus,

$$\breve{P}_{ss'}(\delta)\geq\mathbb{P}_{z'}\left[\alpha(t+1)=\alpha',\forall t<\tau_s^*(\mathcal{N}_\delta(s'))\right]. \quad (25)$$

When the process reaches $\mathcal{N}_\delta(s')$ for the first time, action profile α' has been played for at least $\tau_s^*(\mathcal{N}_\delta(s'))$ times.[2] Define an iteration subsequence $\{t_k\}$, with $Z_{t_0}=Z_0=z'$, such that, at any time t_k action α' is selected for the next iteration (i.e., $\alpha(t_0+1)=...=\alpha(t_k+1)=\alpha'$). Due to the Markov property, in order for the unperturbed process to reach $\mathcal{N}_\delta(s')$, there exists time t_k such that $\alpha(t_k+1)=\alpha'$ while $Z_{t_k}\in E_{j,k}(\alpha')$. Thus,

$$\begin{aligned}\breve{P}_{ss'}(\delta) \quad\leq\quad & \mathbb{P}_{z'}\left[\exists\{t_k\}:\alpha(t_k+1)=\alpha', Z_{t_k}\in E_{j,k}(\alpha'),\right.\\ & \left.\text{for all } k<\tau_s^*(\mathcal{N}_\delta(s'))\right].\end{aligned}$$

Using the properties of the conditional probability, we also have:

$$\breve{P}_{ss'}(\delta) \quad\leq\quad \mathbb{P}_{z'}\left[\exists\{t_k\}:\alpha(t_k+1)=\alpha',\forall k<\tau_s^*(\mathcal{N}_\delta(s'))\right|$$
$$\left.Z_{t_k}\in E_{j,k}(\alpha')\right].$$

Using again the Markov property,

$$\begin{aligned}&\breve{P}_{ss'}(\delta)\\ &\leq\prod_{t<\tau_s^*(\mathcal{N}_\delta(s'))}\sup_{z\in E_{j,t}(\alpha')}\mathbb{P}_{z'}\left[\alpha(t+1)=\alpha'|Z_t=z\right]\\ &=\mathbb{P}_{z'}\left[\alpha(t+1)=\alpha',t<\tau_s^*(\mathcal{N}_\delta(s'))\right].\end{aligned}$$

Given also (25), the conclusion follows.

(c) The minimum first hitting time to the set $\mathcal{N}_\delta(s')$ satisfies:

$$\tau_s^*(\mathcal{N}_\delta(s'))=\left\lceil\frac{\log(\delta)}{\log(H_j(\alpha'))}\right\rceil\doteq T(\epsilon),$$

where $H_j(\alpha')\doteq 1-\epsilon u_j(\alpha')$. There exists correction factor $c=c(\epsilon,\delta)\in[0,1)$, such that

$$T(\epsilon)=\frac{\log(\delta)}{\log(H_j(\alpha'))}+c(\epsilon,\delta).$$

Due to statement (b), and for sufficiently small $\epsilon>0$, we have:

$$\log\left(\breve{P}_{ss'}(\delta)\right)\approx\sum_{t=1}^{T(\epsilon)}\log\left(1-H_j(\alpha')^t\right). \quad (26)$$

In the remainder of the proof, we will approximate the r.h.s. of (26).

To simplify notation, denote $H\doteq H_j(\alpha')$. Note that

$$\begin{aligned}&\lim_{\epsilon\downarrow 0}\log\left(H^{T(\epsilon)}\right)\\ &=\lim_{\epsilon\downarrow 0}\left\{\left(\frac{\log(\delta)}{\log(H)}+c(\epsilon,\delta)\right)\log(H)\right\}=\log(\delta),\end{aligned}$$

and due to the continuity of the natural logarithm,

$$\lim_{\epsilon\downarrow 0}H^{T(\epsilon)}=\delta. \quad (27)$$

As a result, for any $\ell\in\mathbb{N}$,

$$\lim_{\epsilon\downarrow 0}H^{\ell T(\epsilon)}=\delta^\ell.$$

By Taylor series expansion of the natural logarithm (for small argument values), we have:

$$\log\left(1-H^t\right)\approx-\sum_{\ell=1}^{\infty}\frac{H^{\ell t}}{\ell}.$$

Thus,

$$\begin{aligned}&(1-H)\sum_{t=1}^{T(\epsilon)}\log\left(1-H^t\right)\\ &\approx-\sum_{\ell=1}^{\infty}\frac{1}{\ell}\Big[(1-H)\sum_{t=1}^{T(\epsilon)}H^{\ell t}\Big]\\ &=-\sum_{\ell=1}^{\infty}\frac{1}{\ell}\Big[(1-H)\frac{1-H^{\ell(T(\epsilon)+1)}}{1-H^\ell}-(1-H)\Big]\\ &=-\sum_{\ell=1}^{\infty}\frac{1}{\ell}\Big[\frac{1-H^{\ell(T(\epsilon)+1)}}{1+H+\cdots+H^{\ell-1}}-(1-H)\Big]\end{aligned}$$

[2] Let us assume that along a sample path from z' to $\mathcal{N}_\delta(s')$ and at iteration t_k, the strategy of agent j with respect to action α'_j is $x_{j\alpha'_j}(t_k)=\rho>0$. If agent j selects action α'_j at time t_k+1, it's next strategy will be:

$$x_{j\alpha'_j}(t_k+1)=\rho+\epsilon u_j(\alpha')(1-\rho)=\epsilon u_j(\alpha')+H_j(\alpha')\rho\doteq x^*_{j\alpha'_j}.$$

If, instead, agent j selects action $\alpha_j\neq\alpha'_j$ at time t_k+1 and then α'_j at time t_k+2, i.e., it deviates from playing action α'_j, then the strategy evolves as:

$$\begin{aligned}x_{j\alpha'_j}(t_k+1) &= \rho+\epsilon u_j(\alpha)(-\rho)\\ &= H_j(\alpha)\rho,\\ x_{j\alpha'_j}(t_k+2) &= H_j(\alpha)\rho+\epsilon u_j(\alpha')(1-H_j(\alpha)\rho)\\ &= (H_j(\alpha')\rho)H_j(\alpha)+\epsilon u_j(\alpha')\\ &< x^*_{j\alpha'_j},\end{aligned}$$

since $0<\epsilon u_j(\alpha)<1$. Informally, any single deviation from the shortest path to s' cannot recover the drop in the strategy at the next iteration. Thus, along any path from z' to $\mathcal{N}_\delta(s')$, when the process reaches $\mathcal{N}_\delta(s')$ for the first time, action α' has been played for at least $\tau_s^*(\mathcal{N}_\delta(s'))$ times, which is the number of iterations required for reaching $\mathcal{N}_\delta(s')$ along the shortest path.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAC.2019.2895300, IEEE Transactions on Automatic Control

Note that, for any $\ell \in \mathbb{N}$, $H^\ell \to 1$ as $\epsilon \downarrow 0$. Thus, we have

$$\begin{aligned}\lim_{\epsilon\downarrow 0}(1-H)\sum_{t=1}^{T(\epsilon)}\log\left(1-H^t\right)\\ \approx \quad -\sum_{\ell=1}^{\infty}\frac{1}{\ell^2}(1-\delta^\ell) \doteq \eta(\delta),\end{aligned}$$

which corresponds to a negative finite constant. Hence, using the fact that $1-H = \epsilon u_j(\alpha')$, we conclude that, for sufficiently small $\epsilon > 0$,

$$\log(\breve{P}_{ss'}(\delta)) \approx \frac{\eta(\delta)}{\epsilon u_j(\alpha')}.$$

REFERENCES

[1] G. C. Chasparis, "Stochastic stability analysis of perturbed learning automata with constant step-size in strategic-form games," in *American Control Conference*, Seattle, USA, 2017, pp. 4607–4612.

[2] B. G. Chun, R. Fonseca, I. Stoica, and J. Kubiatowicz, "Characterizing selfishly constructed overlay routing networks," in *Proc. of IEEE INFOCOM 04*, Hong-Kong, 2004.

[3] R. Komali, A. B. MacKenzie, and R. P. Gilles, "Effect of selfish node behavior on efficient topology design," *IEEE Transactions on Mobile Computing*, vol. 7, no. 9, pp. 1057–1070, 2008.

[4] G. Wei, A. V. Vasilakos, Y. Zheng, and N. Xiong, "A game-theoretic method of fair resource allocation for cloud computing services," *The Journal of Supercomputing*, vol. 54, no. 2, pp. 252–269, Nov. 2010.

[5] W. B. Arthur, "On designing economic agents that behave like human agents," *J. Evolutionary Econ.*, vol. 3, pp. 1–22, 1993.

[6] M. Tsetlin, *Automaton Theory and Modeling of Biological Systems*. Academic Press, 1973.

[7] K. Narendra and M. Thathachar, *Learning Automata: An introduction*. Prentice-Hall, 1989.

[8] J. Hu and M. P. Wellman, "Nash Q-learning for general-sum stochastic games," *J. Machine Learning Research*, vol. 4, no. Nov, pp. 1039–1069, 2003.

[9] G. Chasparis and J. Shamma, "Distributed dynamic reinforcement of efficient outcomes in multiagent coordination and network formation," *Dynamic Games and Applications*, vol. 2, no. 1, pp. 18–50, 2012.

[10] G. Chasparis, A. Arapostathis, and J. Shamma, "Aspiration learning in coordination games," *SIAM J. Control and Optim.*, vol. 51, no. 1, 2013.

[11] J. R. Marden, H. P. Young, G. Arslan, and J. S. Shamma, "Payoff based dynamics for multi-player weakly acyclic games," *SIAM J. Control Optim.*, vol. 48, no. 1, pp. 373–396, 2009.

[12] P. Frihauf, M. Krstic, and T. Basar, "Nash Equilibrium Seeking in Noncooperative Games," *IEEE Transactions on Automatic Control*, vol. 57, no. 5, pp. 1192–1207, May 2012.

[13] M. Ye and G. Hu, "Distributed Seeking of Time-Varying Nash Equilibrium for Non-Cooperative Games," *IEEE Transactions on Automatic Control*, vol. 60, no. 11, pp. 3000–3005, Nov. 2015.

[14] S. Perkins, P. Mertikopoulos, and D. S. Leslie, "Mixed-Strategy Learning With Continuous Action Sets," *IEEE Transactions on Automatic Control*, vol. 62, no. 1, pp. 379–384, Jan. 2017.

[15] J. Hofbauer and K. Sigmund, *Evolutionary Games and Population Dynamics*. Cambridge: Cambridge University Press, 1998.

[16] E. Hopkins and M. Posch, "Attainability of boundary points under reinforcement learning," *Games Econ. Behav.*, vol. 53, pp. 110–125, 2005.

[17] I. Erev and A. Roth, "Predicting how people play games: reinforcement learning in experimental games with unique, mixed strategy equilibria," *Amer. Econ. Rev.*, vol. 88, pp. 848–881, 1998.

[18] T. Börgers and R. Sarin, "Learning through reinforcement and replicator dynamics," *J. Econ. Theory*, vol. 77, no. 1, pp. 1–14, 1997.

[19] D. Leslie, "Reinforcement learning in games," Ph.D. dissertation, School of Mathematics, University of Bristol, 2004.

[20] G. C. Chasparis, J. S. Shamma, and A. Rantzer, "Nonconvergence to saddle boundary points under perturbed reinforcement learning," *Int. J. Game Theory*, vol. 44, no. 3, pp. 667–699, 2015.

[21] D. Lewis, *Convention: A Philosophical Study*. Blackwell Publishing, 2002.

[22] P. Sastry, V. Phansalkar, and M. Thathachar, "Decentralized learning of Nash equilibria in multi-person stochastic games with incomplete information," *IEEE Trans. Syst. Man Cybern.*, vol. 24, no. 5, pp. 769–777, 1994.

[23] D. Monderer and L. Shapley, "Potential games," *Games Econ. Behav.*, vol. 14, pp. 124–143, 1996.

[24] K. Verbeeck, A. Now, J. Parent, and K. Tuyls, "Exploring selfish reinforcement learning in repeated games with stochastic rewards," *Autonomous Agents and Multi-Agent Systems*, vol. 14, no. 3, pp. 239–269, Apr. 2007.

[25] D. Leslie and E. Collins, "Individual Q-Learning in Normal Form Games," *SIAM J. Control Optim.*, vol. 44, no. 2, pp. 495–514, Jan. 2005.

[26] A. C. Chapman, D. S. Leslie, A. Rogers, and N. R. Jennings, "Convergent Learning Algorithms for Unknown Reward Games," *SIAM J. Control Optim.*, vol. 51, no. 4, pp. 3154–3180, Jan. 2013.

[27] G. Arslan and S. Yuksel, "Decentralized Q-Learning for Stochastic Teams and Games," *IEEE Transactions on Automatic Control*, vol. PP, no. 99, pp. 1–1, 2016.

[28] H. P. Young, "Learning by trial and error," *Games and Economic Behavior*, vol. 65, no. 2, pp. 626–643, Mar. 2009.

[29] J. R. Marden, H. P. Young, and L. Pao, "Achieving Pareto optimality through distributed learning," *SIAM J. Control Optim.*, vol. 52, no. 5, pp. 2753–2770, 2014.

[30] O. Hernandez-Lerma and J. B. Lasserre, *Markov Chains and Invariant Probabilities*. Birkhauser Verlag, 2003.

[31] R. Karandikar, D. Mookherjee, and D. Ray, "Evolving aspirations and cooperation," *J. Econ. Theory*, vol. 80, pp. 292–331, 1998.

[32] M. I. Freidlin and A. D. Wentzell, *Random perturbations of dynamical systems*. New York, NY: Springer-Verlag, 1984.

[33] G. C. Chasparis and J. S. Shamma, "Network Formation: Neighborhood Structures, Establishment Costs, and Distributed Learning," *IEEE Transactions on Cybernetics*, vol. 43, no. 6, pp. 1950–1962, Dec. 2013.

[34] M. Reed, *Fundamental Ideas of Analysis*. John Wiley & Sons, Inc., 1998.

[35] W. Rudin, *Principles of Mathematical Analysis*. McGraw-Hill Book Company, 1964.

[36] L. Breiman, *Probability*. Philadelphia: SIAM, 1992.

Georgios C. Chasparis Georgios Chasparis (S'04-M'08) was born in Athens in 1978. He received the mechanical engineering degree from the National Technical University of Athens, Greece, in 2001, and the M.Sc. and Ph.D. degrees from the University of California Los Angeles, CA, in 2004 and 2008, respectively.

From 2008 to 2010, he was a Postdoctoral Fellow in the Department of Electrical and Computer Engineering at the Georgia Institute of Technology, GA, and from 2010 to 2012, he was a Postdoctoral Fellow in the Department of Automatic Control at Lund University, Sweden. Since 2012, he has been with the Department of Data Analysis Systems at the Software Competence Center Hagenberg, GmbH, Austria, where he is currently a Research Team Leader in Prognosis, Control and Optimization. His research interests include evolutionary learning in games, distributed control and optimization, and operations research.

Efficient Dynamic Pinning of Parallelized Applications by Distributed Reinforcement Learning

Georgios C. Chasparis & Michael Rossbory

International Journal of Parallel Programming

ISSN 0885-7458

Int J Parallel Prog
DOI 10.1007/s10766-017-0541-y

Springer

Int J Parallel Prog
https://doi.org/10.1007/s10766-017-0541-y

Efficient Dynamic Pinning of Parallelized Applications by Distributed Reinforcement Learning

Georgios C. Chasparis[1] · **Michael Rossbory**[1]

Received: 31 May 2017 / Accepted: 18 November 2017

Abstract This paper introduces a resource allocation framework specifically tailored for addressing the problem of dynamic placement (or pinning) of parallelized applications to processing units. Under the proposed setup each thread of the parallelized application constitutes an independent decision maker (or agent), which (based on its own prior performance measurements and its own prior CPU-affinities) decides on which processing unit to run next. Decisions are updated recursively for each thread by a resource manager/scheduler which runs in parallel to the application's threads and periodically records their performances and assigns to them new CPU affinities. For updating the CPU-affinities, the scheduler uses a distributed reinforcement-learning algorithm, each branch of which is responsible for assigning a new placement strategy to each thread. The proposed framework is flexible enough to address alternative optimization criteria, such as maximum average processing speed and minimum speed variance among threads. We demonstrate analytically that convergence to locally-optimal placements is achieved asymptotically. Finally, we validate these results through experiments in Linux platforms.

Keywords Dynamic pinning · Reinforcement learning · Parallel applications

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.

✉ Georgios C. Chasparis
georgios.chasparis@scch.at

Michael Rossbory
michael.rossbory@scch.at

[1] Software Competence Center Hagenberg GmbH, Softwarepark 21, 4232 Hagenberg, Austria

Published online: 24 November 2017

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 [7], access bandwidth and battery life in wireless communications [8], computing bandwidth under certain QoS requirements [1], computing bandwidth for time-sensitive applications [4], computing bandwidth and memory in parallelized applications [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. However, guaranteeing efficiency through the adjustment of resources requires the formulation of a centralized optimization problem (e.g., mixed-integer linear programming formulations [1]), which further requires information about the specifics of each task (i.e., application details). 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 in resource allocation, not to mention their computational complexity, feedback from the running tasks in the form of performance 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) [3], or in the case of allocating virtual processors to time-sensitive applications [4].

To this end, this paper proposes a distributed learning scheme specifically tailored for addressing the problem of dynamically assigning/pinning threads of a parallelized application to the available processing units. The proposed scheme is flexible enough to incorporate alternative optimization criteria. In particular, we demonstrate its utility in maximizing the average processing speed of the overall application. The proposed scheme also reduces computational complexity usually encountered in centralized optimization problems, while it provides an adaptive response to the variability of the provided resources.

The paper is organized as follows. Section 2 discusses the related work and contribution of this paper. Section 3 describes the overall problem formulation and objective. Section 4 introduces the concept of multi-agent formulations and discusses their advantages. Section 5 presents the proposed reinforcement-learning algorithm for dynamic placement of threads and Sect. 6 presents its convergence analysis. Section 7 presents experiments of the proposed resource manager in a Linux platform and comparison tests with the operating system's response. Finally, Sect. 8 presents concluding remarks.

Notation:

- $|x|$ denotes the Euclidean norm of a vector $x \in \mathbb{R}^n$.
- $\mathsf{dist}(x, A)$ denotes the minimum distance from a vector $x \in \mathbb{R}^n$ to a set $A \subset \mathbb{R}^n$, i.e., $\mathsf{dist}(x, A) \doteq \inf_{y \in A} |x - y|$.

- $\mathcal{B}_\delta(A)$ denotes the δ-neighborhood of a set $A \subset \mathbb{R}^n$, i.e., $\mathcal{B}_\delta(A) \doteq \{x \in \mathbb{R}^n : \mathsf{dist}(x, A) < \delta\}$.
- For some finite set A, $|A|$ denotes the cardinality of A.
- The probability simplex of dimension n is denoted by $\Delta(n)$ and defined as $\Delta(n) \doteq \{x = (x_1, \ldots, x_n) \in [0, 1]^n : \sum_{i=1}^n x_i = 1\}$.
- $e_j \in \mathbb{R}^n$ denotes the unit vector whose jth entry is equal to 1 while all other entries are zero;
- For a vector $\sigma \in \Delta(n)$, let $\mathsf{rand}_\sigma [a_1, \ldots, a_n]$ denote the random selection of an element of the set $\{a_1, \ldots, a_n\}$ according to the distribution σ;

2 Related Work and Contribution

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 to address a centralized (*global*) objective for resource allocation through agent-based (*local*) objectives, where, for instance, agents may represent the tasks to be allocated. Examples include the cooperative game formulation for allocating bandwidth in grid computing [14], the non-cooperative game formulation in the problem of medium access protocols in communications [15] or for allocating resources in cloud computing [17]. The main advantage of distributing the decision-making process is the considerable reduction in computational complexity (a group of n tasks can be allocated to m resources with m^n possible ways, while a single task may be allocated with only m possible ways). This further allows for the development of online selection rules where tasks/agents make decisions often using current observations of their *own* performance.

Prior work has demonstrated the importance of thread-to-core bindings in the overall performance of a parallelized application. For example, [9] describes a tool that checks the performance of each of the available thread-to-core bindings and searches an optimal placement. Unfortunately, the *exhaustive-search* type of optimization that is implemented may prohibit runtime implementation. Reference [3] combines the problem of thread scheduling with scheduling hints related to thread-memory affinity issues. These hints are able to accommodate load distribution given information for the application structure and the hardware topology. The HWLOC library is used to perform the topology discovery which builds a hierarchical architecture consisting of hardware objects (NUMA nodes, sockets, caches, cores, etc.), and the BubbleSched library [16] is used to implement scheduling policies. A similar scheduling policy is also implemented by [13].

Contrary to this line of research, this paper proposes a dynamic (learning-based) scheme for optimally allocating threads of a parallelized application into a set of available CPU cores. The proposed methodology implements a distributed reinforcement learning algorithm (executed in parallel by a resource manager/scheduler), according to which each thread is considered as an independent agent making decisions over its own CPU-affinities. The proposed algorithm requires minimum information exchange, that is only the performance measurements collected from each running thread. In the current setup, we provide a design that maximizes average processing

speed, however it can be easily modified to accommodate alternative criteria. Furthermore, it exhibits adaptivity and robustness to possible irregularities in the behavior of a thread or to possible changes in the availability of resources. We analytically demonstrate that the reinforcement-learning scheme asymptotically learns a locally-optimal allocation, while it is flexible enough to accommodate several optimization criteria. We also demonstrate through experiments in a Linux platform that the proposed algorithm outperforms the scheduling strategies of the operating system with respect to its average processing speed and completion time.

3 Problem Formulation and Objective

3.1 Framework

We consider a resource allocation framework for addressing the problem of dynamic pinning of parallelized applications. In particular, we consider a number of threads $\mathcal{I} = \{1, 2, \ldots, n\}$ resulting from a parallelized application. These threads need to be pinned for processing into a set of available CPU cores $\mathcal{J} = \{1, 2, \ldots, m\}$ (not necessarily homogeneous).

We denote the *assignment* of a thread i to the set of available CPU cores by $\alpha_i \in \mathcal{A}_i \equiv \mathcal{J}$, i.e., α_i designates the number of the CPU where this thread is being assigned to. Let also $\alpha = \{\alpha_i\}_i$ denote the *assignment profile*.

Responsible for the assignment of CPU cores into the threads is the Resource Manager (RM), which periodically checks the prior performance of each thread and makes a decision over their next CPU placements so that a (user-specified) objective is maximized. *For the remainder of the paper*, we will assume that:

(a) The internal properties and details of the threads are not known to the RM. Instead, the RM may only have access to measurements related to their performance (e.g., their processing speed).
(b) Threads may not be idled or postponed. Instead, the goal of the RM is to assign the *currently* available resources to the *currently* running threads.
(c) Each thread may only be assigned to a single CPU core.

3.2 Static Optimization and Issues

Let $v_i = v_i(\alpha, w)$ denote the processing speed of thread i which depends on both the assignment profile α, as well as exogenous parameters aggregated within w. The exogenous parameters w summarize, for example, the impact of other applications running on the same platform (*disturbances*). Then, the previously mentioned centralized objectives may take on the following form:

$$\max_{\alpha \in \mathcal{A}} f(\alpha, w). \tag{1}$$

We consider two objectives:

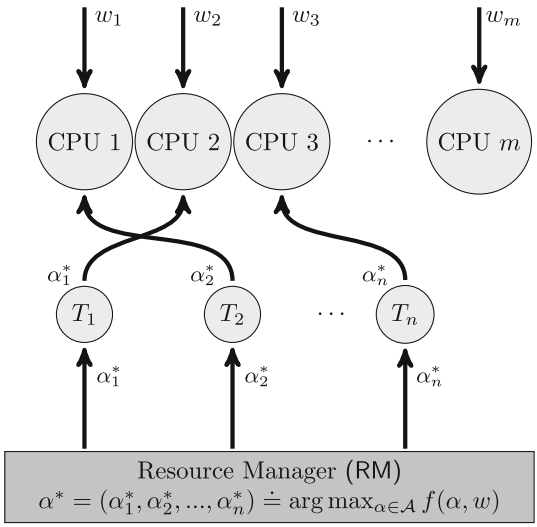

Fig. 1 Schematic of *static* resource allocation framework

(O1) $f(\alpha, w) \doteq \sum_{i=1}^{n} v_i/n$, corresponds to the average processing speed of all threads;

(O2) $f(\alpha, w) \doteq \sum_{i=1}^{n}[v_i - \gamma(v_i - \sum_{j\in\mathcal{I}} v_j/n)^2]/n$, for some $\gamma > 0$, corresponds to the average processing speed minus a penalty that is proportional to the speed variance among threads.

Any solution to the optimization problem (1) will correspond to an *efficient assignment*. Figure 1 presents a schematic of a *static* resource allocation framework sequence of actions where the centralized objective (1) is solved by the RM once and then it communicates the optimal assignment to the threads.

However, there are two practical issues when posing an optimization problem in the form of (1). In particular,

1. the function $v_i(\alpha, w)$ is unknown and it may only be evaluated through measurements of the processing speed, denoted $\tilde{v}_i$;
2. the exogenous disturbances $w = (w_1, \ldots, w_m)$ are unknown and may vary with time, thus the optimal assignment may not be fixed with time.

3.3 Measurement- or Learning-Based Optimization

We wish to address a *static* objective of the form (1) through a *measurement-* or *learning-based* optimization approach. According to such approach, the RM reacts to measurements of the objective function $f(\alpha, w)$, periodically collected at time instances $k = 1, 2, \ldots$ and denoted $\tilde{f}(k)$. For example, in the case of objective (O1), $\tilde{f}(k) \doteq \sum_{i=1}^{n} \tilde{v}_i(k)/n$. Given these measurements and the current assignment $\alpha(k)$ of resources, the RM selects the next assignment of resources $\alpha(k+1)$ so that the measured objective approaches the true optimum of the unknown performance function $f(\alpha, w)$. In other words, the RM employs an update rule of the form:

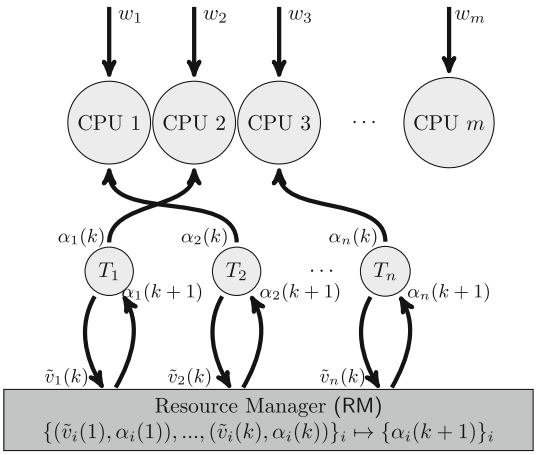

Fig. 2 Schematic of *dynamic* resource allocation framework

$$\{(\tilde{v}_i(1), \alpha_i(1)), \ldots, (\tilde{v}_i(k), \alpha_i(k))\}_i \mapsto \{\alpha_i(k+1)\}_i \tag{2}$$

according to which prior pairs of measurements and assignments for each thread i are mapped into a new assignment $\alpha_i(k+1)$ that will be employed during the next evaluation interval.

The overall framework is illustrated in Fig. 2 describing the flow of information and steps executed. In particular, at any given time instance $k = 1, 2, \ldots$, each thread i communicates to the RM its current processing speed $\tilde{v}_i(k)$. Then the RM updates the assignments for each thread i, $\alpha_i(k+1)$, and communicates this assignment to them.

3.4 Objective

The objective in this paper is to address the problem of adaptive or dynamic pinning through a distributed learning framework. Each thread will constitute an independent decision maker or agent, thus naturally introducing a multi-agent formulation. Each thread selects its own CPU assignment independently using its own preference criterion (although the necessary computations for such selection are executed by the RM). The advantages are two-folded: (a) it reduces computational complexity, since each thread has only m available choices (instead of m^n available group choices), and (b) it allows for a faster response to changes in resource availability.

The goal is to design a preference criterion and a selection rule for each thread, so that when each thread tries to maximize its own (*local*) criterion then certain guarantees can be achieved regarding the overall (*global*) performance of the parallelized application.

In the following sections, we will go through the design for such a distributed scheme, and we will provide guarantees with respect to its asymptotic behavior and robustness.

4 Multi-agent (or Game) Formulation

The first step towards a distributed learning scheme is the decomposition of the decision making process into multiple decision makers (or agents). Naturally, in the problem of placing threads of a parallelized application into a set of available processing units, *each thread may be considered as an independent decision maker*.

4.1 Strategy

Since each agent (or thread) selects actions independently, we generally assume that each agent's action is a realization of an independent discrete random variable. Let $\sigma_{ij} \in [0, 1]$, $j \in \mathcal{A}_i$, denote the probability that agent i selects its jth action in $\mathcal{A}_i$. If $\sum_{j=1}^{|\mathcal{A}_i|} \sigma_{ij} = 1$, then $\sigma_i \doteq (\sigma_{i1}, \ldots, \sigma_{i|\mathcal{A}_i|})$ is a probability distribution over the set of actions $\mathcal{A}_i$ (or *strategy* of agent i). Then $\sigma_i \in \Delta(|\mathcal{A}_i|)$. To provide an example, consider the case of 3 available CPU cores, i.e., $\mathcal{A}_i = \{1, 2, 3\}$. In this case, the strategy $\sigma_i \in \Delta(3)$ of thread i may take the following form: $\sigma_i = (0.2, 0.5, 0.3)$, such that 0.2 corresponds to the probability of assigning itself to CPU core 1, 0.5 corresponds to the probability of assigning itself to CPU core 2 and 0.3 corresponds to the probability of assigning itself to CPU core 3. Briefly, the assignment selection will be denoted by $\alpha_i = \mathsf{rand}_{\sigma_i}[\mathcal{A}_i]$.

We will also use the term *strategy profile* to denote the combination of strategies of all agents $\sigma = (\sigma_1, \ldots, \sigma_n) \in \mathbf{\Delta}$ where $\mathbf{\Delta} \doteq \Delta(|\mathcal{A}_1|) \times \cdots \times \Delta(|\mathcal{A}_n|)$ is the set of strategy profiles.

Note that if σ_i is a unit vector (or a vertex of $\Delta(|\mathcal{A}_i|)$), say e_j, then agent i selects its jth action with probability one. Such a strategy will be called *pure strategy*. Likewise, a *pure strategy profile* is a profile of pure strategies.

4.2 Utility Function

A cornerstone in the design of any measurement-based algorithm is the *preference criterion* or *utility function* u_i for each thread $i \in \mathcal{A}$. The utility function captures the benefit of a decision maker (thread) resulting from the assignment profile α selected by all threads, i.e., it represents a function of the form $u_i : \mathcal{A} \to \mathbb{R}_+$ (where we restrict it to be a positive number). Often, we may decompose the argument of the utility function as follows $u_i(\alpha) = u_i(\alpha_i, \alpha_{-i})$, where $-i \doteq \mathcal{I} \backslash i$. The utility function introduces a preference relation for each decision maker where $u_i(\alpha_i, \alpha_{-i}) \geq u_i(\alpha'_i, \alpha_{-i})$ translates to α_i being more desirable/preferable than α'_i.

It is important to note that the utility function u_i of each agent/thread i is subject to *design* and it is introduced in order to guide the preferences of each agent. Thus, u_i may not necessarily correspond to a measured quantity, but it could be a function of available performance counters.

For example, a natural choice for the utility of each thread is its own execution speed v_i. Other options may include more egalitarian criteria, where the utility function of each thread corresponds to the overall global objective $f(\alpha, w)$. The definition of a utility function is open-ended.

5 Reinforcement Learning (RL)

We employ a distributed learning framework (namely, *perturbed learning automata*) that is based on the reinforcement learning algorithm introduced in [5,6]. It belongs to the general class of *learning automata* [12].

The basic idea behind reinforcement learning is rather simple. If agent i selects action j at instance k and a favorable payoff results, $u_i(\alpha)$, the action probability $\sigma_{ij}(k)$ is increased and all other entries of $\sigma_i(k)$ are decreased.

According to the *perturbed learning automata* [5,6], the strategy of each thread at any time instance $k = 1, 2, \ldots$ is as follows:

$$\sigma_i(k) = (1 - \lambda)x_i(k) + \frac{\lambda}{|\mathcal{A}_i|}\mathbf{1} \tag{3}$$

where $\lambda > 0$ corresponds to a perturbation term (or *mutation*), $x_i(k)$ corresponds to the *nominal strategy* of agent i and $\mathbf{1}$ is a vector of ones of appropriate size. The nominal strategy is updated according to the following update recursion:

$$x_i(k+1) = x_i(k) + \epsilon \cdot u_i(\alpha(k)) \cdot [e_{\alpha_i(k)} - x_i(k)], \tag{4}$$

for some constant step size $\epsilon > 0$. Note that according to this recursion, the new nominal strategy will increase in the direction of the action $\alpha_i(k)$ which is currently selected and it will increase proportionally to the utility received. Finally, each agent updates its action by randomizing over the strategy σ_i, i.e.,

$$\alpha_i(k+1) = \mathsf{rand}_{\sigma_i}[\mathcal{A}_i].$$

In comparison to [5,6], the difference lies in the use of the constant step size $\epsilon > 0$ (instead of a decreasing step-size sequence). This selection increases the adaptivity and robustness of the algorithm to possible changes in the environment. This is because a constant step size provides a fast transition of the nominal strategy from one pure strategy to another.

Furthermore, the reason for introducing the perturbation term λ is to provide the possibility for the nominal strategy to escape from pure strategy profiles, that is profiles at which all agents assign probability one in one of the actions. Setting $\lambda > 0$ is essential for providing an adaptive response of the algorithm to changes in the environment.

6 Convergence Analysis

In this section, we establish a connection between the asymptotic behavior of the nominal strategy profile $x(k)$ with the *Nash equilibria*[1] of the induced assignment game, that is the set of locally stable strategy profiles.

[1] A strategy profile $\sigma^* = (\sigma_1^*, \ldots, \sigma_n^*) \in \mathbf{\Delta}$ is a Nash equilibrium if, no agent has incentive to change unilaterally its own strategy, i.e., no agent can increase its expected utility by altering its own strategy.

Let the utility function u_i for each thread i correspond to the global objective (1), i.e., $u_i(\alpha) = f(\alpha, w)$ defined by either (O1) or (O2). Let us denote $\mathcal{S}^\lambda$ to be the set of *stationary points* of the mean-field dynamics (cf., [10]) of the recursion (4), defined as follows

$$\mathcal{S}^\lambda \doteq \left\{ x \in \mathbf{\Delta} : g_i^\lambda(x) \doteq \mathbb{E}\left[u_i(\alpha(k))[e_{\alpha_i(k)} - x_i(k)] | x(k) = x \right] = 0, \forall i \in \mathcal{I} \right\}.$$

The expectation operator $\mathbb{E}[\cdot]$ is defined appropriately over the canonical path space $\Omega = \mathbf{\Delta}^\infty$ with an element ω being a sequence $\{x(0), x(1), \ldots\}$ with $x(k) = (x_1(k), \ldots, x_n(k)) \in \mathbf{\Delta}$ generated by the reinforcement learning process. Similarly we define the probability operator $\mathbb{P}[\cdot]$. In other words, the set of stationary points corresponds to the strategy profiles at which the expected change in the strategy profile is zero.

According to [5,6], a connection can be established between the set of stationary points $\mathcal{S}^\lambda$ and the set of Nash equilibria of the induced assignment game. In particular, for sufficiently small $\lambda > 0$, *the set of* $\mathcal{S}^\lambda$ *includes only* λ*-perturbations of Nash-equilibrium strategies* [5,6].

The following proposition is a straightforward extension of [5, Theorem 1] to the case of constant step size.

Proposition 1 *Let the* RM *employ the strategy update rule* (4) *and placement selection* (3) *for each thread* i*. Updates are performed periodically with a fixed period such that* $\tilde{v}_i(k) > 0$ *for all* i *and* k*. Let the utility function for each thread* i *satisfy* $u_i(\alpha) = f(\alpha, w)$*, under either objective (O1) or (O2), where* $\gamma \geq 0$ *is small enough such that* $u_i(\alpha(k)) > 0$ *for all* k*. Then, for some* $\lambda > 0$ *sufficiently small, there exists* $\delta = \delta(\lambda)$*, with* $\delta(\lambda) \downarrow 0$ *as* $\lambda \downarrow 0$*, such that*

$$\mathbb{P}\left[\liminf_{k \to \infty} \mathsf{dist}(x(k), \mathcal{B}_\delta(\mathcal{S}^\lambda)) = 0 \right] = 1. \tag{5}$$

Proof The proof follows the exact same steps of the first part of [5, Theorem 1], where the decreasing step-size sequence is being replaced by a constant $\epsilon > 0$. □

Proposition 1 states that when we select λ sufficiently small, the nominal strategy trajectory will be approaching the set $\mathcal{B}_\delta(\mathcal{S}^\lambda)$ infinitely often with probability one, that is a small neighborhood of the Nash equilibria. We require that the update period is large enough so that each thread is using resources within each evaluation period. Of course, if a thread stops executing then the same result holds but for the updated set of threads.

The following proposition provides a characterization of the stochastically stable outcomes.

Proposition 2 (Weak convergence to Nash equilibria) *Under the hypotheses of Proposition 1, the fraction of time that the nominal strategy profile* $x(k)$ *spends in* $\mathcal{B}_\delta(\mathcal{S}^\lambda)$ *goes to one (in probability) as* $\epsilon \to 0$ *and* $k \to \infty$*.*

Proof The proof follows directly from [10, Theorem 8.4.1] and Proposition 1. □

Proposition 2 states that if we take a small step size $\epsilon > 0$, then as the time index k increases, we should expect that the nominal strategy spends the majority of the time within a small neighborhood of the Nash equilibrium strategies. Given that the utility function satisfies $u_i(\alpha) = f(\alpha, w)$, for each i, then *the set of Nash equilibria includes the set of efficient assignments*, i.e., the solutions of (1). Thus, due to Proposition 2, it is guaranteed that *the nominal strategies* $x_i(k)$, $i \in \mathcal{I}$, *will spend the majority of the time in a small neighborhood of locally-optimal assignments, which provides a minimum performance guarantee throughout the running time of the parallelized application.*

Note that due to varying exogenous factors (w), the Nash-equilibrium assignments may not stay fixed for all future times. The above proposition states that the process will spend the majority of the time within the set of the Nash-equilibrium assignments for as long as this set is fixed. If, at some point in time, this set changes (due to, e.g., other applications start running on the same platform), then the above result continues to hold but for the new set of Nash equilibria. Hence, the process is adaptive to possible performance variations.

7 Experiments

In this section, we present an experimental study of the proposed reinforcement learning scheme for dynamic pinning of parallelized applications. Experiments were conducted on `20×Intel©Xeon©CPU E5-2650 v3 2.30 GHz` running Linux Kernel 64bit 3.13.0-43-generic. The machine divides the physical cores into two NUMA nodes (Node 1: 0-9 CPU cores, Node 2: 10-19 CPU cores).

7.1 Experimental Setup

We consider a computationally intensive routine that executes a fixed number of computations (corresponding to the combinations of M out of a set of $N > M$ numbers). The routine is being parallelized using the `pthread.h` (C++ POSIX thread library), where each thread is executing a replicate of the above set of computations. The nature of these computations does not play any role and in fact it may vary between threads (as we shall see in the forthcoming experiments). Intentionally, the considered application does not make extensive memory use, since the main focus is the investigation of the utility of reinforcement learning in maximizing the overall processing speed.

Throughout the execution, and with a fixed period of 0.2 s, the RM collects measurements of the total instructions per sec (using the PAPI library [11]) for each one of the threads separately. Given the provided measurements, the update rule of Eq. (4) with the utility function $u_i(\alpha) = f(\alpha, w)$ under (O2) is executed by the RM. Placement of the threads to the available CPU cores is achieved through the `sched.h` library (in particular, the `pthread_setaffinity_np` function). In the following, we demonstrate the response of the RL scheme in comparison to the operating system (OS) response (i.e., when placement of the threads is not controlled by the RM). We compare them for different values of $\gamma \geq 0$ in order to investigate the influence of more balanced speeds to the overall running time. In all the forthcoming

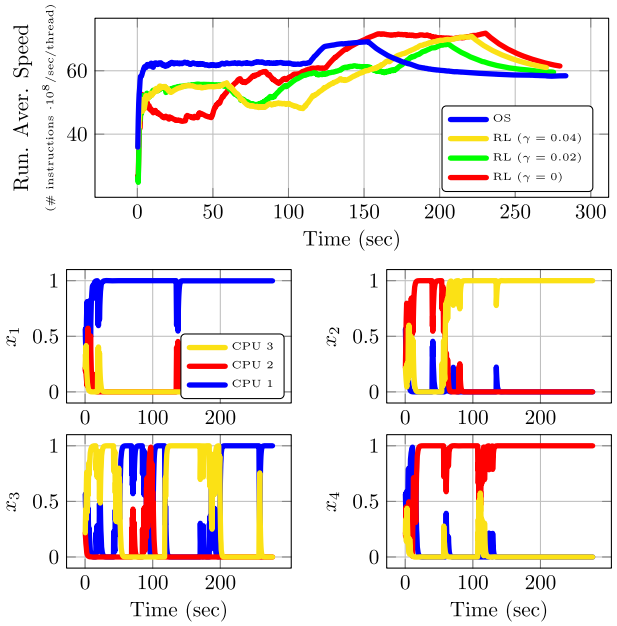

Fig. 3 *Experiment 1.* Running average execution speed when 4 threads run on 3 identical CPU cores. Thread 3 requires half as much computing bandwidth as the rest of the threads, which are identical. The strategies of the threads (x_1, x_2, x_3 and x_4) capture the nominal selection probabilities and are generated by the RL scheme with $\gamma = 0.04$. The RL schemes of all threads run with $\epsilon = 0.005$ and $\lambda = 0.005$

experiments, the RM is executed within the master thread which is always running in the first available CPU (CPU 1).

7.2 Experiment 1: Small Number of Threads and Uniform Availability of Resources

In this experiment, we consider the case of small number of threads and CPU cores. Threads are non-identical with respect to the requested bandwidth, while CPU cores are identical in the amount of provided bandwidth (*uniform availability*). In particular, one of the threads requires smaller CPU bandwidth than the rest. We should expect that in an optimal setup, threads that require smaller CPU bandwidth should be the ones sharing a CPU core with other threads. On the other hand, threads that require large bandwidth, they should be placed alone.

In particular, in this experiment, Thread 3 requires only half as much computing bandwidth as the rest of the threads (i.e., Threads 1, 2 and 4). The resulting performance is depicted in Fig. 3.

We observe indeed that Thread 1, 2 and 4 (which require larger computing bandwidth) are allocated to different CPU cores (CPU 1, 3 and 2, respectively). On the other hand, Thread 3 is switching between CPU 1 and CPU 3, since both provide almost equal processing bandwidth to Thread 3. In other words, the less demanding thread

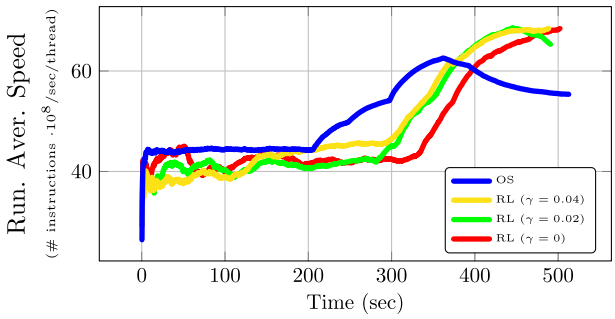

Fig. 4 *Experiment 2.* Running average execution speed when 7 non-identical threads run on 3 CPU cores. Thread 1 and 2 require half as much computing bandwidth as the rest of the threads, which are identical. Thread 3 is joining after the 120 s. The RL schemes of all threads run with $\epsilon = 0.005$ and $\lambda = 0.005$

is sharing a CPU core with one of the most demanding threads. Note that this assignment corresponds to a Nash equilibrium (as Proposition 2 states), since there is no thread that can benefit by *unilaterally* changing its strategy. It is also straightforward to check that this assignment is also efficient, in the sense that no other assignment could increase the overall average speed.

Note, finally, that the difference with the processing speed of the OS scheme is small, although a more balanced processing speed ($\gamma = 0.04$) improved slightly the overall completion time. However, given that the performance index used corresponds to the objective (O2), we may only provide a guarantee with respect to the average processing speed.

7.3 Experiment 2: Small Number of Threads and Non-uniform Availability of Resources

In this experiment, we demonstrate the robustness of the algorithm in a dynamic environment. We consider 7 threads and 3 available CPU cores. The first two threads (Thread 1 and 2) require about half as much computing bandwidth as the rest of the threads. The rest of the threads (Thread 3, 4, 5, 6 and 7) are identical. However, Thread 3 starts running later in time (in particular, after 120 s).

Figure 4 illustrates the evolution of the RL-based scheduling scheme under different values of γ. Again, a faster response of the overall application can be achieved when higher values of γ are selected. The difference should be attributed to the fact that the OS fails to distinguish between threads with different bandwidth requirements. Table 1 presents a statistical analysis of these schemes where the speed difference between the RL ($\gamma = 0.04$) and the OS reaches approximately 5% on average.

Note that, in general, a maximization of the running average speed of the threads should not necessarily imply a reduction in the completion time. However, in the current setup of almost identical threads, we should expect that increasing the running average speed increases the chances of improving the completion time.

In both Experiment 1 and 2, we do not utilize any initial smart placement of the threads, rather the initial strategies of the threads correspond to the uniform distribu-

Table 1 *Experiment 2.* Comparison between the OS performance and RL schemes when $\epsilon = 0.005$ and $\lambda = 0.005$ for different values of γ

Run #	OS (s)	RL ($\gamma = 0$) (s)	RL ($\gamma = 0.02$) (s)	RL ($\gamma = 0.04$) (s)
1	513	505	492	489
2	530	506	489	494
3	536	517	518	515
4	533	507	515	509
5	523	502	491	496
6	513	523	501	492
7	520	514	497	492
8	530	518	499	497
9	520	532	500	497
10	528	517	493	492
Aver.	524.6	514.1	499.5	497.3
SD	8.06	9.29	9.85	8.27

tion. This is the main reason that it takes some time for the RM to increase the average speed of the threads. However, we observe that it is eventually able to reach higher speed levels compared to the OS, which explains the shorter completion time.

7.4 Experiment 3: Large Number of Threads in a Dynamic Environment with Multiple NUMA Nodes

In this experiment, we would like to see how the proposed learning scheme scales when we increase the number of threads and available CPU cores. Although the considered application does not make heavy memory use, we allow placement in both available NUMA nodes, thus indirectly considering cache-memory related disturbances. Finally, we utilize an initial round-robin placement of the threads, that intends on enhancing the initial adjustment phase of the RM.

In particular, in this experiment, we consider 25 identical threads that can be placed on 12 CPU cores. The resource availability in these cores is not uniform, given that in the first 6 cores other applications are also running.

In Fig. 5, we demonstrate the evolution of the running average processing speed under the OS scheduler and the learning-based RM. First, we observe that the initial placement of CPU cores makes a significant difference with respect to the overall completion time (compared to the previous experiments). Furthermore, we observe that minimizing variance can again decrease the completion time, which is indeed reasonable in the case of identical threads.

Lastly, it is important to point out that even though we have an increased number of threads and CPU cores, the algorithm scales well. As expected, issues with respect to cache-memory use should not be so apparent in this simulation example, due to the nature of this application.

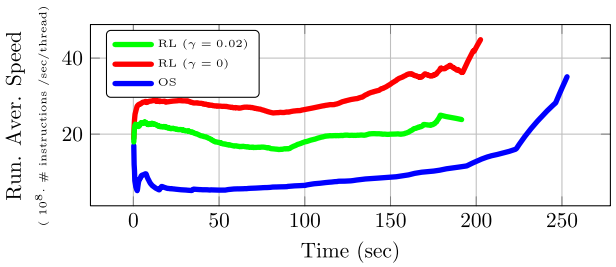

Fig. 5 *Experiment 3.* Running average execution speed when 25 identical threads run on 12 CPU cores. The first 6 available CPU cores are also occupied by other applications. The RL schemes of all threads run with $\epsilon = 0.005$ and $\lambda = 0.005$

Table 2 *Experiment 3.* Comparison between the OS performance and RL schemes when $\epsilon = 0.005$ and $\lambda = 0.005$ for different values of γ

Run #	OS (s)	RL ($\gamma = 0$) (s)	RL ($\gamma = 0.02$) (s)
1	244	201	196
2	243	191	196
3	263	203	192
4	255	195	198
5	254	194	193
6	252	199	200
7	259	190	191
8	276	192	195
9	250	197	197
10	248	200	190
Aver.	254.44	196.78	195.44
SD	9.85	4.50	3.37

Table 2 also presents a statistical analysis of these schemes under the current setup. We observe that the completion time difference between the RL schemes and the OS reaches approximately 20% on average.

8 Conclusions

We proposed a measurement-based learning scheme for addressing the problem of efficient dynamic pinning of parallelized applications into processing units. According to this scheme, a centralized objective is decomposed into thread-based objectives, where each thread is assigned its own utility function. A RM updates a strategy for each one of the threads corresponding to its beliefs over the most beneficial CPU placement for this thread. Updates are based on a reinforcement learning rule, where prior actions are reinforced proportionally to the resulting utility. It was shown that, when we appropriately design the threads' utilities, then convergence to the set of locally optimal assignments is achieved. Besides its reduced computational complexity, the proposed scheme is adaptive and robust to possible changes in the environment.

We demonstrated the utility of the proposed framework in the maximization of the running average processing speed of the threads, which in the case of almost identical threads led to a significant reduction in the completion time. Alternative objectives may also be defined, as long as they consist of measured performance indicators. An interesting future direction would involve the possibility of criteria that also optimize memory placement, something that would enlarge the application domain of the proposed learning scheme.

References

1. Bini, E., Buttazzo, G.C., Eker, J., Schorr, S., Guerra, R., Fohler, G., Årzén, K.E., Vanessa, R., Scordino, C.: Resource management on multicore systems: The ACTORS approach. IEEE Micro **31**(3), 72–81 (2011)
2. Brecht, T.: On the importance of parallel application placement in NUMA multiprocessors. In: Proceedings of the Symposium on Experiences with Distributed and Multiprocessor Systems (SEDMS IV). pp. 1–18. San Deigo, CA (1993)
3. Broquedis, F., Furmento, N., Goglin, B., Wacrenier, P.A., Namyst, R.: ForestGOMP: an efficient OpenMP environment for NUMA architectures. Int. J. Parallel Program. **38**, 418–439 (2010)
4. Chasparis, G.C., Maggio, M., Bini, E., Årzén, K.E.: Design and implementation of distributed resource management for time-sensitive applications. Automatica **64**, 44–53 (2016)
5. Chasparis, G.C., Shamma, J.S., Rantzer, A.: Nonconvergence to saddle boundary points under perturbed reinforcement learning. Int. J. Game Theory **44**(3), 667–699 (2015)
6. Chasparis, G., Shamma, J.: Distributed dynamic reinforcement of efficient outcomes in multiagent coordination and network formation. Dyn. Games Appl. **2**(1), 18–50 (2012)
7. De Angelis, F., Boaro, M., Fuselli, D., Squartini, S., Piazza, F., Wei, Q.: Optimal home energy management under dynamic electrical and thermal constraints. IEEE Trans. Ind. Inform. **9**(3), 1518–1527 (2013)
8. Inaltekin, H., Wicker, S.: A one-shot random access game for wireless networks. In: International Conference on Wireless Networks, Communications and Mobile Computing (2005)
9. Klug, T., Ott, M., Weidendorfer, J., Trinitis, C.: `autopin`: automated optimization of thread-to-core pinning on multicore systems. In: Stenstrom, P. (ed.) Transactions on High-Performance Embedded Architectures and Compilers III. Lecture Notes in Computer Science, vol. 6590, pp. 219–235. Springer, Berlin (2011)
10. Kushner, H.J., Yin, G.G.: Stochastic Approximation and Recursive Algorithms and Applications, 2nd edn. Springer, New York (2003)
11. Mucci, P.J., Browne, S., Deane, C., Ho, G.: PAPI: a portable interface to hardware performance counters. In: Proceedings of the Department of Defense HPCMP Users Group Conference. pp. 7–10 (1999)
12. Narendra, K., Thathachar, M.: Learning Automata: An Introduction. Prentice-Hall, Upper Saddle River (1989)
13. Olivier, S., Porterfield, A., Wheeler, K.: Scheduling task parallelism on multi-socket multicore systems. In: ROSS'11. pp. 49–56. Tuscon, Arizona, USA (2011)
14. Subrata, R., Zomaya, A.Y., Landfeldt, B.: A cooperative game framework for QoS guided job allocation schemes in grids. IEEE Trans. Comput. **57**(10), 1413–1422 (2008)
15. Tembine, H., Altman, E., ElAzouri, R., Hayel, Y.: Correlated evolutionary stable strategies in random medium access control. In: International Conference on Game Theory for Networks. pp. 212–221 (2009)
16. Thibault, S., Namyst, R., Wacrenier, P.: Building portable thread schedulers for hierarchical multiprocessors: the bubblesched framework. In: Euro-Par. ACM. Rennes, France (2007)
17. Wei, G., Vasilakos, A.V., Zheng, Y., Xiong, N.: A game-theoretic method of fair resource allocation for cloud computing services. J. Supercomput. **54**(2), 252–269 (2010)