Perturbed Learning Automata in Coordination Games and Resource-Allocation Problems

Georgios C. Chasparis

Department of Data Analysis Systems
Software Competence Center Hagenberg GmbH, Austria

JKU

Linz, Austria
May 3rd, 2019

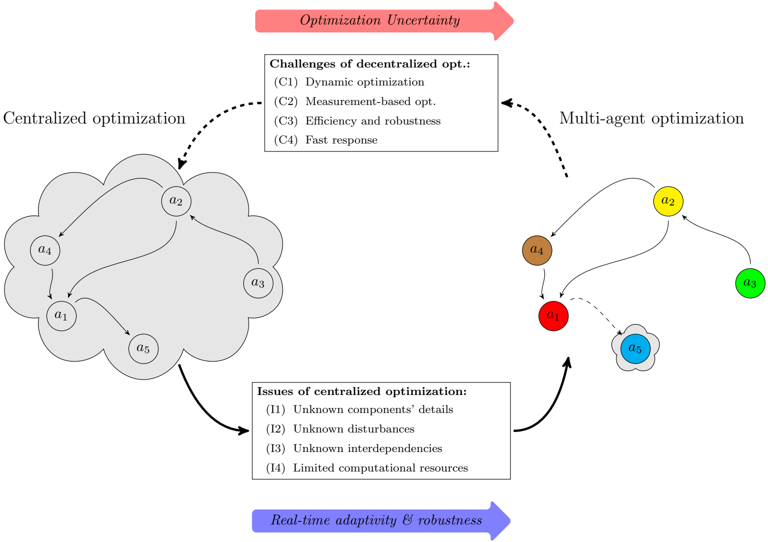
Outline

1. Centralized vs Decentralized Opt
2. Perturbed Learning Automata
3. Stochastic Stability
4. Scheduling Parallelized Applications

Outline

1. Centralized vs Decentralized Opt
2. Perturbed Learning Automata
3. Stochastic Stability
4. Scheduling Parallelized Applications

Centralized vs Decentralized Optimization

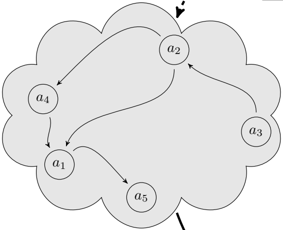


Centralized vs Decentralized Optimization

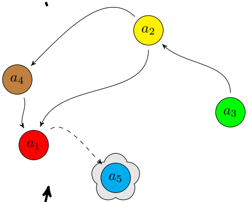
Optimization Uncertainty

Challenges of decentralized opt.:
- (C1) Dynamic optimization
- (C2) Measurement-based opt.
- (C3) Efficiency and robustness
- (C4) Fast response

Centralized optimization



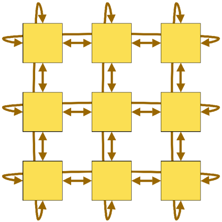
Multi-agent optimization



Issues of centralized optimization:
- (I1) Unknown components' details
- (I2) Unknown disturbances
- (I3) Unknown interdependencies
- (I4) Limited computational resources

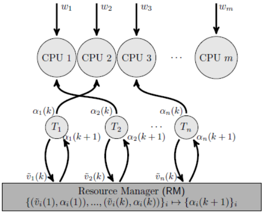
Real-time adaptivity & robustness

Challenges (*Resource-Aware Applications*)



Parallel deployment

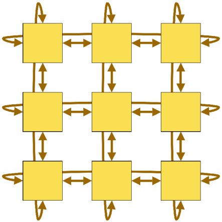
- *Why centralized opt. fails?*
    - Unknown application details
    - Unknown disturbances
    - Limited computational resources



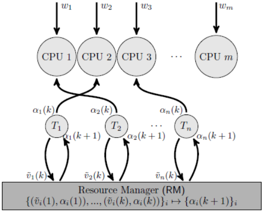
Adaptive Resource Allocation

Challenges (*Resource-Aware Applications*)

- *Why centralized opt. fails?*
  - Unknown application details
  - Unknown disturbances
  - Limited computational resources
- *Instead: Measurement-based opt.*
  - Performance indices may be unknown
  - Immediate reaction to performance drops
  - Reduced computational complexity



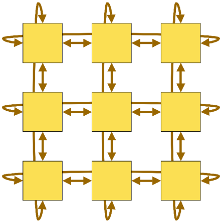
Parallel deployment



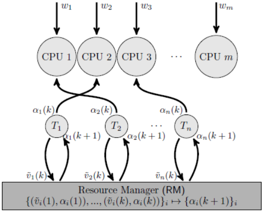
Adaptive Resource Allocation

Challenges (*Resource-Aware Applications*)

- *Why centralized opt. fails?*
  - Unknown application details
  - Unknown disturbances
  - Limited computational resources
- *Instead: Measurement-based opt.*
  - Performance indices may be unknown
  - Immediate reaction to performance drops
  - Reduced computational complexity
- $+$ *Distributed sensing/actuation*
  - Localized disturbance rejection
  - Reduced computational complexity
  - Reduced communication complexity



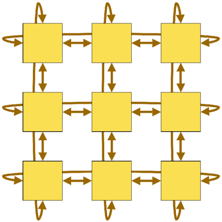
Parallel deployment



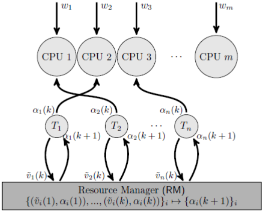
Adaptive Resource Allocation

Challenges (*Resource-Aware Applications*)

- *Why centralized opt. fails?*
  - Unknown application details
  - Unknown disturbances
  - Limited computational resources
- *Instead: Measurement-based opt.*
  - Performance indices may be unknown
  - Immediate reaction to performance drops
  - Reduced computational complexity
- + *Distributed sensing/actuation*
  - Localized disturbance rejection
  - Reduced computational complexity
  - Reduced communication complexity
- *Additional challenges:*
  - Optimization uncertainty
  - Convergence speed



Parallel deployment



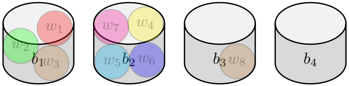
Adaptive Resource Allocation

Multiagent Coordination Problems

- *Additional (structural) assumptions:*
  - “Alignment” of interests
- *Challenges remain:*
  - distributed sensing/actuation
  - measurement-based learning
  - optimization uncertainty
  - convergence properties

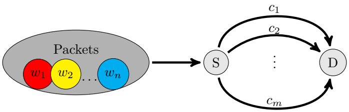
Multiagent Coordination Problems

- *Additional (structural) assumptions:*
  - “Alignment” of interests
- *Challenges remain:*
  - distributed sensing/actuation
  - measurement-based learning
  - optimization uncertainty
  - convergence properties



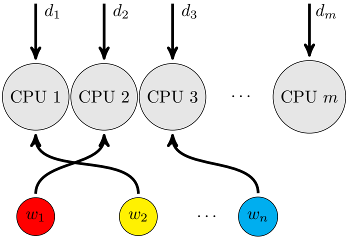
Multiagent Coordination Problems

- *Additional (structural) assumptions:*
  - "Alignment" of interests
- *Challenges remain:*
  - distributed sensing/actuation
  - measurement-based learning
  - optimization uncertainty
  - convergence properties



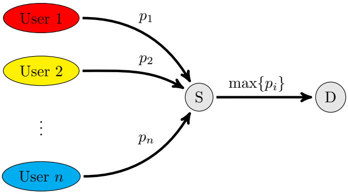
Multiagent Coordination Problems

- *Additional (structural) assumptions:*
  - “Alignment” of interests
- *Challenges remain:*
  - distributed sensing/actuation
  - measurement-based learning
  - optimization uncertainty
  - convergence properties



Multiagent Coordination Problems

- *Additional (structural) assumptions:*
  - "Alignment" of interests
- *Challenges remain:*
  - distributed sensing/actuation
  - measurement-based learning
  - optimization uncertainty
  - convergence properties

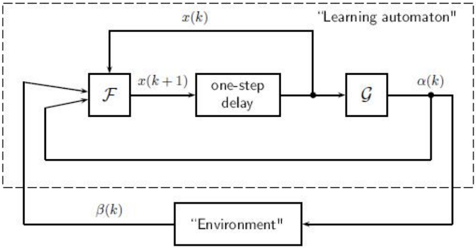


Outline

1. Centralized vs Decentralized Opt
2. Perturbed Learning Automata
3. Stochastic Stability
4. Scheduling Parallelized Applications

Learning Automata

- **Learning Automata:**
  - Agents revise their decisions *repeatedly*
  - Information is only *local*
    - Agents observe only their own utility
  - Agents reinforce an action through
    - repeated selection
    - reward size
  - Introduced/analyzed first by Tsetlin (1973)



Strategic-form Games: Basic Notation/Terminology

- Each agent i select actions based on the *strategy*

$$\sigma_i \triangleq \begin{pmatrix} \sigma_{i1} \\ \vdots \\ \sigma_{i|\mathcal{A}_i|} \end{pmatrix} \in \Delta\left(|\mathcal{A}_i|\right)$$

- Each agent i receives a *utility* (or *payoff*),

$$u_i : \mathcal{A} \to \mathbb{R}_+$$

Strategic-form Games: Basic Notation/Terminology

- Each agent i select actions based on the *strategy*

$$\sigma_i \triangleq \begin{pmatrix} \sigma_{i1} \\ \vdots \\ \sigma_{i|\mathcal{A}_i|} \end{pmatrix} \in \Delta\left(|\mathcal{A}_i|\right)$$

- Each agent i receives a *utility* (or *payoff*),

$$u_i : \mathcal{A} \to \mathbb{R}_+$$

- Example:
  - 2 players, 2 actions
  - strategy: e.g., $\sigma_i = (0.2, 0.8)$
  - utility: e.g., $u_i(A, A) = 2$.

	A	B
A	$2, 2$	$0, 0$
B	$0, 0$	$1, 1$

(Variable structure) Learning Automata

At each time period $k = 0, 1, 2, \ldots$, each agent i

1. **Action update:** Randomize using strategy $x_i(k)$,
$$\alpha_i(k) = \text{rand}_{x_i}[\mathcal{A}_i]$$
2. **Performance Observation:**
$$u_i = u_i(\alpha(k))$$
3. **Strategy update:**
$$x_i(k+1) = x_i(k) + \epsilon(k) \cdot u_i(\alpha(k)) \cdot (e_{\alpha_i(k)} - x_i(k))$$

(Variable structure) Learning Automata

At some time k, agent i

1. **Action update:** Selects $\alpha_i(k) = A$ based on strategy

$$x_i(k) = \begin{pmatrix} 0.2 \\ 0.8 \end{pmatrix}$$

2. **Performance Observation:**

$$u_i = u_i(A,A) = 2$$

3. **Strategy update:**

$$\begin{pmatrix} 0.2 + 1.6\epsilon \\ 0.8 - 1.6\epsilon \end{pmatrix} \leftarrow \begin{pmatrix} 0.2 \\ 0.8 \end{pmatrix} + \epsilon \cdot 2 \cdot \left[\begin{pmatrix} 1 \\ 0 \end{pmatrix} - \begin{pmatrix} 0.2 \\ 0.8 \end{pmatrix} \right]$$

Example:

	A	B
A	$2, 2$	$0, 0$
B	$0, 0$	$1, 1$

s c c h software competence center

(Variable structure) Learning Automata

At some time k, agent i

1. **Action update:** Selects $\alpha_i(k) = A$ based on strategy

$$x_i(k) = \begin{pmatrix} 0.2 \\ 0.8 \end{pmatrix}$$

2. **Performance Observation:**

$$u_i = u_i(A,A) = 2$$

3. **Strategy update:**

$$\begin{pmatrix} 0.2 + 1.6\epsilon \\ 0.8 - 1.6\epsilon \end{pmatrix} \leftarrow \begin{pmatrix} 0.2 \\ 0.8 \end{pmatrix} + \epsilon \cdot 2 \cdot \left[\begin{pmatrix} 1 \\ 0 \end{pmatrix} - \begin{pmatrix} 0.2 \\ 0.8 \end{pmatrix} \right]$$

Note:

- $x_i(k)$ increases *in the direction of* the selected action
- $x_i(k)$ increases *proportionally to* the observed performance

Prior Schemes: Erev-Roth type dynamics

Action update:

$$\alpha_i(t) = \mathrm{rand}_{x_i(k)}[\mathcal{A}_i]$$

Strategy update:

$$x_i(k+1) = x_i(k) + \epsilon_i(k) \cdot u_i(\alpha(k)) \cdot \left[e_{\alpha_i(k)} - x_i(k)\right]$$

- *Arthur (1993), Posch (1997) models:*

$$\epsilon_i(k) \triangleq \frac{1}{ck^{\nu} + u_i(\alpha(k))}$$

  - Excluding convergence to non-Nash equilibria.

Prior Schemes: Erev-Roth type dynamics

Action update:

$$\alpha_i(t) = \mathrm{rand}_{x_i(k)}[\mathcal{A}_i]$$

Strategy update:

$$x_i(k+1) = x_i(k) + \epsilon_i(k) \cdot u_i(\alpha(k)) \cdot \left[e_{\alpha_i(k)} - x_i(k)\right]$$

- *Arthur (1993), Posch (1997) models:*

$$\epsilon_i(k) \triangleq \frac{1}{ck^{\nu} + u_i(\alpha(k))}$$

  – Excluding convergence to non-Nash equilibria.

- *Urn Process:* [Hopkins & Posch (2005), Erev & Roth (1998)]

$$\epsilon_i(k) \triangleq \frac{1}{V_i(k) + u_i(\alpha(k))}$$

  + Excluding convergence to non-Nash equilibria.
  – Convergence to Nash equilibria only in 2-player partnership games

Prior Schemes: Learning automata

Action update:

$$\alpha_i(t) = \mathrm{rand}_{x_i(k)}[\mathcal{A}_i]$$

Strategy update:

$$x_i(k+1) = x_i(k) + \epsilon_i(k) \cdot u_i(\alpha(k)) \cdot \left[e_{\alpha_i(k)} - x_i(k)\right]$$

- *Narendra & Thathachar (1989):*

$$u_i(\alpha(k)) \in [0, 1]$$

  - Convergence to Nash equilibria only in *identical interest games*
  - Extension to large games requires an *absolute monotonocity* condition.

Prior Schemes: Learning automata

Action update:

$$\alpha_i(t) = \text{rand}_{x_i(k)}[\mathcal{A}_i]$$

Strategy update:

$$x_i(k+1) = x_i(k) + \epsilon_i(k) \cdot u_i(\alpha(k)) \cdot \left[e_{\alpha_i(k)} - x_i(k)\right]$$

- *Narendra & Thathachar (1989):*

$$u_i(\alpha(k)) \in [0, 1]$$

  – Convergence to Nash equilibria only in *identical interest games*
  – Extension to large games requires an *absolute monotonocity* condition.

- *Verbeeck et al (2007):*
  – Introduced a *coordinated exploration phase*
  + Convergence to efficient Nash equilibria

Prior Schemes: Learning automata

Action update:

$$\alpha_i(t) = \text{rand}_{x_i(k)}[\mathcal{A}_i]$$

Strategy update:

$$x_i(k+1) = x_i(k) + \epsilon_i(k) \cdot u_i(\alpha(k)) \cdot \left[e_{\alpha_i(k)} - x_i(k)\right]$$

- *Chasparis, Shamma & Rantzer (2014)*

$$\sigma_i(k) = (1-\lambda)x_i(k) + \lambda\mathbf{1}/n$$

  - \+ excludes convergence to non-Nash equilibria
  - \+ guarantees global convergence to pure Nash equilibria in potential games
  - − global convergence in generic coordination games is *not* shown

Why learning automata?

	A	B
A	$2,2$	$0,0$
B	$0,0$	$1,1$

- *equilibrium-selection* mechanism
  - We can get convergence to desirable outcomes
  - Modified selection rules may be required
- *measurement-based* dynamics
  - Agents only observe performance measurements
- "handles" *noisy observations*
  - noise is filtered out through the strategy-vector formulation
  - demonstrated in the analysis of Hopkins and Posch (2005)

Issues?

	A	B
A	$2, 2$	$0, 0$
B	$0, 0$	$1, 1$

- *Issues*
  - global convergence to efficient outcomes is difficult to show.
  - excluding convergence to mixed strategies.
  - Lyapunov-based techniques are not appropriate for large games

Issues?

	A	B
A	$2,2$	$0,0$
B	$0,0$	$1,1$

- *Issues*
  - global convergence to efficient outcomes is difficult to show.
  - excluding convergence to mixed strategies.
  - Lyapunov-based techniques are not appropriate for large games

- *Contributions*
  - a *stochastic stability* analysis for perturbed learning automata
  - global convergence guarantees (circumvents issues of Lyapunov-based analysis)
  - specialization to coordination games

Outline

1. Centralized vs Decentralized Opt
2. Perturbed Learning Automata
3. Stochastic Stability
4. Scheduling Parallelized Applications

Stochastic Stability

Strategy Update:

$$x_i(k+1) = x_i(k) + \epsilon_i(k) \cdot u_i(\alpha(k)) \cdot \left[e_{\alpha_i(k)} - x_i(k)\right]$$

Action selection:

$$\sigma_i(k) = (1-\lambda)x_i(k) + \lambda\mathbf{1}/n$$

Note:

- Defines an induced Markov chain in:

$$\mathcal{Z} \doteq \mathcal{A} \times \Delta(n)$$

- Infinite dimensional with t.p.f. P_λ

Assumption: $u_i(\alpha) > 0$ for all i and $\alpha \in \mathcal{A}$.

G. Chasparis, "*Stochastic Stability Analysis of Perturbed Learning Automata in Positive-Utility Games*," IEEE Transactions on Automatic Control, 2018.

Stochastic Stability

Strategy Update:

$$x_i(k+1) = x_i(k) + \epsilon_i(k) \cdot u_i(\alpha(k)) \cdot \left[e_{\alpha_i(k)} - x_i(k)\right]$$

Action selection:

$$\sigma_i(k) = (1-\lambda)x_i(k) + \lambda\mathbf{1}/n$$

Proposition

For $\lambda = 0$, the probability that eventually agents play the same action profile is 1

G. Chasparis, "*Stochastic Stability Analysis of Perturbed Learning Automata in Positive-Utility Games*," IEEE Transactions on Automatic Control, 2018.

Stochastic Stability

Strategy Update:

$$x_i(k+1) = x_i(k) + \epsilon_i(k) \cdot u_i(\alpha(k)) \cdot \left[e_{\alpha_i(k)} - x_i(k)\right]$$

Action selection:

$$\sigma_i(k) = (1-\lambda)x_i(k) + \lambda\mathbf{1}/n$$

Proposition

For $\lambda = 0$, the probability that eventually agents play the same action profile is 1

Remark

Reduce infinite dimensional P_λ to finite dimensional π (isomorphic with $\mathcal{A}$).

G. Chasparis, "*Stochastic Stability Analysis of Perturbed Learning Automata in Positive-Utility Games*," IEEE Transactions on Automatic Control, 2018.

Stochastic Stability

Strategy Update:

$$x_i(k+1) = x_i(k) + \epsilon_i(k) \cdot u_i(\alpha(k)) \cdot \left[e_{\alpha_i(k)} - x_i(k)\right]$$

Action selection:

$$\sigma_i(k) = (1-\lambda)x_i(k) + \lambda\mathbf{1}/n$$

Theorem

There exists a unique probability vector π *such that:*

1. $\mu_\lambda \Rightarrow \sum_{\alpha\in\mathcal{A}} \pi_\alpha \delta_\alpha(\cdot)$ *as* $\lambda \downarrow 0$,
2. π *is an invariant distribution of the (finite-state) Markov chain* $\hat{P}$

$$\hat{P}_{\alpha\alpha'} \doteq \lim_{t\to\infty} QP^t(\alpha, \mathcal{N}_\varepsilon(\alpha')),$$

for any $\varepsilon > 0$, *where* Q *is the t.p.f. of one player trembling.*

G. Chasparis, "*Stochastic Stability Analysis of Perturbed Learning Automata in Positive-Utility Games*," IEEE Transactions on Automatic Control, 2018.

Stochastic Stability

Strategy Update:

$$x_i(k+1) = x_i(k) + \epsilon_i(k) \cdot u_i(\alpha(k)) \cdot \left[e_{\alpha_i(k)} - x_i(k)\right]$$

Action selection:

$$\sigma_i(k) = (1-\lambda)x_i(k) + \lambda\mathbf{1}/n$$

Theorem

There exists a unique probability vector π such that:

1. $\mu_\lambda \Rightarrow \sum_{\alpha\in\mathcal{A}} \pi_\alpha \delta_\alpha(\cdot)$ *as* $\lambda \downarrow 0$,
2. π *is an invariant distribution of the (finite-state) Markov chain* $\hat{P}$

$$\hat{P}_{\alpha\alpha'} \doteq \lim_{t\to\infty} QP^t(\alpha, \mathcal{N}_\varepsilon(\alpha')),$$

for any $\varepsilon > 0$, *where* Q *is the t.p.f. of one player trembling.*

Infinite dimensional $\Rightarrow$ *Finite dimensional Markov chain*

G. Chasparis, "*Stochastic Stability Analysis of Perturbed Learning Automata in Positive-Utility Games*," IEEE Transactions on Automatic Control, 2018.

δ-resistance

Lemma

For sufficiently small $\epsilon > 0$, the one-step transition probabilities (of the finite approximation) satisfy:

$$\hat{P}_{\alpha\alpha'} = \gamma \lim_{\delta \downarrow 0} \exp\left(\frac{\eta(\delta)}{\epsilon u_j(\alpha')}\right)$$

for some negative constant $\eta(\delta)$.

δ-resistance

Lemma

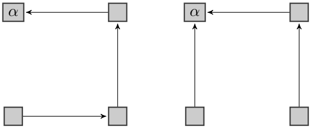
For sufficiently small $\epsilon > 0$, the one-step transition probabilities (of the finite approximation) satisfy:

$$\hat{P}_{\alpha\alpha'} = \gamma \lim_{\delta \downarrow 0} \exp\left(\frac{\eta(\delta)}{\epsilon u_j(\alpha')}\right)$$

for some negative constant $\eta(\delta)$.

- δ-resistance:

$$\varphi_\delta(\alpha|g) \doteq \sum_{(\alpha^{(k)} \to \alpha^{(\ell)})} \frac{1}{\epsilon u_j(\alpha^{(\ell)})}$$



($\mathcal{W}$-graphs)

δ-resistance

Lemma

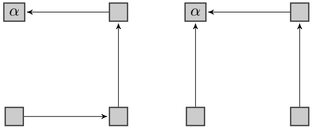
For sufficiently small $\epsilon > 0$, the one-step transition probabilities (of the finite approximation) satisfy:

$$\hat{P}_{\alpha\alpha'} = \gamma \lim_{\delta \downarrow 0} \exp\left(\frac{\eta(\delta)}{\epsilon u_j(\alpha')}\right)$$

for some negative constant $\eta(\delta)$.

- δ-resistance:

$$\varphi_\delta(\alpha|g) \doteq \sum_{(\alpha^{(k)} \to \alpha^{(\ell)})} \frac{1}{\epsilon u_j(\alpha^{(\ell)})}$$



($\mathcal{W}$-graphs)

Theorem

As $\epsilon \downarrow 0$, the set of stochastically stable action profiles $\mathcal{A}^*$ is such that, for any $\delta > 0$,

$$\max_{\alpha^* \in \mathcal{A}^*} \varphi_\delta^*(\alpha^*) < \min_{\alpha \in \mathcal{A} \setminus \mathcal{A}^*} \varphi_\delta^*(\alpha)$$

where φ_δ^* denotes minimum resistance over all g.

Specialization to Coordination Games

Definition (Coordination games)

A strategic-form game satisfying the positive-utility property 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 $\alpha_i' \in \mathrm{BR}_i(\alpha)$.

G. Chasparis, "*Stochastic Stability Analysis of Perturbed Learning Automata in Positive-Utility Games*," IEEE Transactions on Automatic Control, 2018.

Specialization to Coordination Games

Definition (Coordination games)

A strategic-form game satisfying the positive-utility property 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 $\alpha_i' \in \mathrm{BR}_i(\alpha)$.

Theorem

In any coordination game, as $\epsilon \downarrow 0$ and $\lambda \downarrow 0$,

$$\mathcal{S}^* \subseteq \mathcal{S}_{\mathrm{NE}}$$

G. Chasparis, "*Stochastic Stability Analysis of Perturbed Learning Automata in Positive-Utility Games*," IEEE Transactions on Automatic Control, 2018.

Specialization to Coordination Games

Definition (Coordination games)

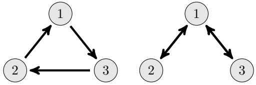
A strategic-form game satisfying the positive-utility property is a coordination game if, for every action profile α and player i, $u_j(\alpha_i', \alpha_{-i}) \geq u_j(\alpha_i, \alpha_{-i})$ for any $\alpha_i' \in \mathrm{BR}_i(\alpha)$.

Theorem

In any coordination game, as $\epsilon \downarrow 0$ and $\lambda \downarrow 0$,

$$\mathcal{S}^* \subseteq \mathcal{S}_{\mathrm{NE}}$$

- **Example:** *Network Formation Games.*



G. Chasparis, "*Stochastic Stability Analysis of Perturbed Learning Automata in Positive-Utility Games*," IEEE Transactions on Automatic Control, 2018.

Contribution Snapshot

Features/Conditions	*Strong* Convergence in Strategic-Form Games		
	Reinforcement-based learning	Q-learning	*Aspiration-based learning*
(Structural) Assumptions:			
2 players	✓	✓	✓
> 2 players	✓	○	✓
Potential games	✓	✓	✓
Coordination games	✓	○	✓
Weakly-acyclic games	○	○	✓
Convergence to:			
Nash equilibria	✓	✓	✓
(Pareto) Efficient Nash equil.	○	○	✓
(Pareto) Efficient outcomes	○	○	✓
Additional features:			
Noisy observations	✓	✓	○
Constant step-size	✓	○	✓

- **Aspiration-based learning:**
  - Benchmark-based learning (Marden, Young, Arslan, Shamma, 2009)
  - Trial-and-error learning (Young, 2011)
  - Mood-based learning (Marden, Young, Pao, 2014)
  - Aspiration learning (Chasparis, Arapostathis, Shamma, 2013)

Outline

1 Centralized vs Decentralized Opt

2 Perturbed Learning Automata

3 Stochastic Stability

4 Scheduling Parallelized Applications

Scheduling for Parallel Applications

- **Questions:** *how to?*
  - map components/threads and data to the available computing resources
  - dynamically reschedule and migrate components and data between resources
- **Prior work:**
  - *Static mapping approaches:*
    - make decisions *prior* to execution
    - involve *memory-aware* scheduling techniques [Markatos & LeBlanc '91]
    - involve *optimization* for near-optimal instantiation [Brown et al '14]
  - *Dynamic mapping approaches:*
    - make decisions *during runtime*
    - involve *exhaustive-search* type algorithms for best bindings [Klug et al '11]
    - involve *scheduling hints* about affinity issues [Broquedis et al '10, Olivier et al '11]
- **Criticism:**
  - *computational complexity*
  - *failure to consider irregular application behavior*
  - *failure to exploit feedback information from the application*

Framework

Setup

- n threads result from a parallelized application
- each thread needs to be executed on a NUMA/CPU node

Resource Manager: *Assumptions*

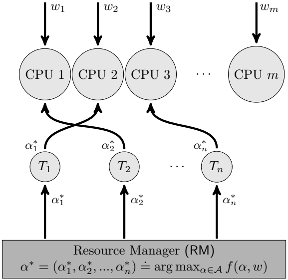
- application's details are *not* known
- threads may *not* be idled or postponed
- each thread may be assigned *only* to a single CPU

Static Optimization

- Centralized (*for all threads*) objective:

$$max_{\alpha \in \mathcal{A}} \quad f(\alpha, w)$$

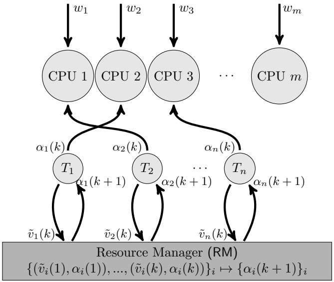
  - $\mathcal{A}$: set of allocations
  - w: external disturbances



- **Example:** *average processing speed*
- **Issues:**
  - only *measurements* of the processing speed are available
  - the exogenous disturbances w are *unknown*

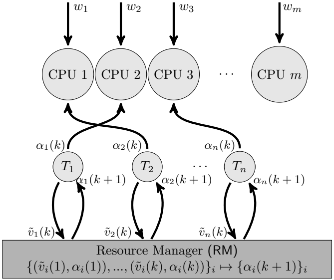
Measurement- or *learning-based* optimization

- At regular time instances k
  1. *measure* processing speeds
  2. *evaluate* current assignments
  3. *assign* next allocation



- **Example:**
  1. *measure* $\tilde{v}_i$
  2. *compute* $\tilde{f} = \sum_i \tilde{v}_i/n$
  3. *pick* α^* that provided the maximum $\tilde{f}$ so far.

Measurement- or *learning-based* optimization

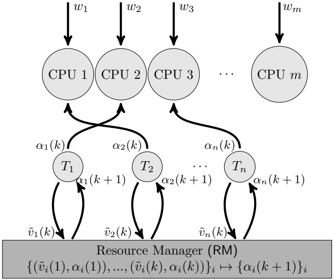


- At regular time instances k
  1. *measure* processing speeds
  2. *evaluate* current assignments
  3. *assign* next allocation

- **Issues:**
  - computation complexity (m^n allocations)
  - a *testing period* is necessary

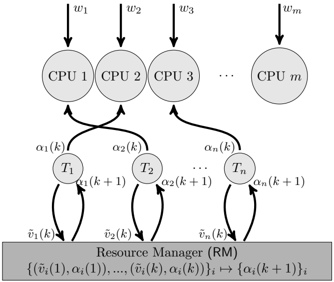
Distributed learning optimization

- At regular time instances k, *each thread*
  1. *measures* processing speed
  2. *evaluates* current assignment
  3. *assigns* next allocation



Distributed learning optimization

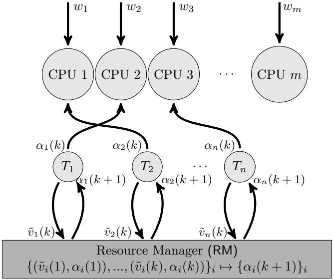
- At regular time instances k, *each thread*
  1. *measures* processing speed
  2. *evaluates* current assignment
  3. *assigns* next allocation



- **Advantages:**
  - reduces computational complexity
  - allows for immediate response to performance variations
  - allows for more direct exploration
  - may guarantee global optimality (*subject to design*)

Distributed learning optimization

- At regular time instances k, *each thread*
  1. *measures* processing speed
  2. *evaluates* current assignment
  3. *assigns* next allocation

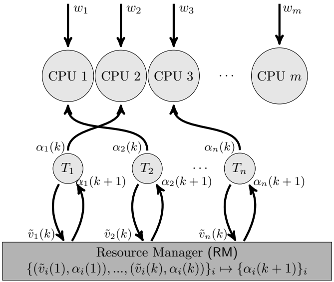


- **Goal:** *design, for each thread,*
  1. the performance function
  2. the selection criterion

  so that, *our original (global)* objective $\tilde{f}$ is maximized.

Distributed Learning Automata for CPU pinning (*PaRL-Sched*)

- At regular time instances k, *each thread*
  1. *measures* processing speed
  2. *creates* strategies over CPU ass.
  3. *decides* over the next assignment



- **Game structure:**
  - Load balancing game (in principle, *potential game*)

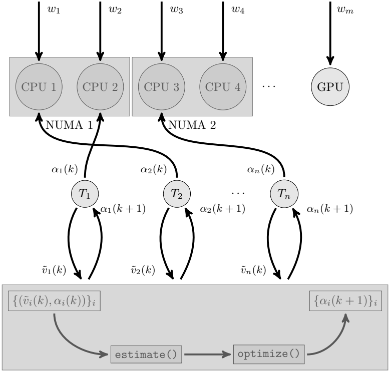
- **Implement:** Perturbed Learning Automata
  - Currently, local stability analysis

G. C. Chasparis and M. Rossbory, "Efficient Dynamic Pinning of Parallelized Applications by Distributed Reinforcement Learning," Int. J. Parallel Program., pp. 1–15, 2017.

Distributed Learning for NUMA architectures

- **Issues/challenges:**
  1. *nested/multi-layer* resources
     (e.g., NUMA-CPU pairs)
  2. *additional degrees of optimization*
     (e.g., due to memory affinities)

- **Contributions:**
  1. *multi-layer* resource allocation
     (i.e., distinguish NUMA from CPU placement)
  2. *multi-time-scale* resource allocation
     (i.e., slower NUMA switching than CPU switching)
  3. *novel aspiration-based learning*
     (for NUMA/memory placement)



Dynamic Scheduler

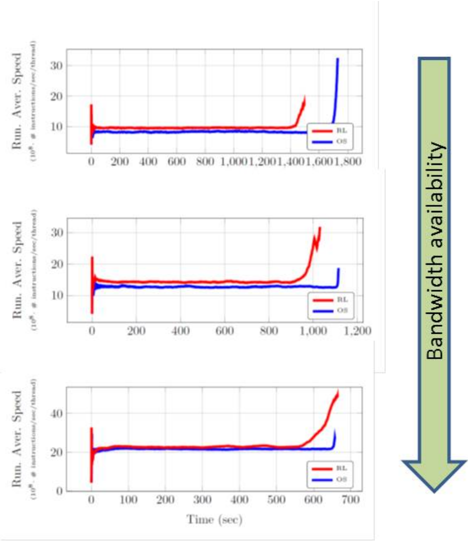
G. C. Chasparis et al., "Learning-based Dynamic Pinning of Parallelized Applications in Many-Core Systems," Euromicro Conf. (PDP), 2019

Setup

- **Setup:**
  - Linux platform (2 NUMA nodes x 14 CPU cores each)
  - C++ POSIX thread library for parallelization
  - PAPI.h for measurement collection
  - numa.h policy library
- **Dynamic Scheduler:** PaRL-sched
  - Aspiration-based learning for NUMA placements (*slow response*)
  - Reinforcement-based learning for CPU placements (*fast response*)
  - Utility of each thread = *processing speed*
- **Parallelized Application:**
  - Ant-Colony Optimization

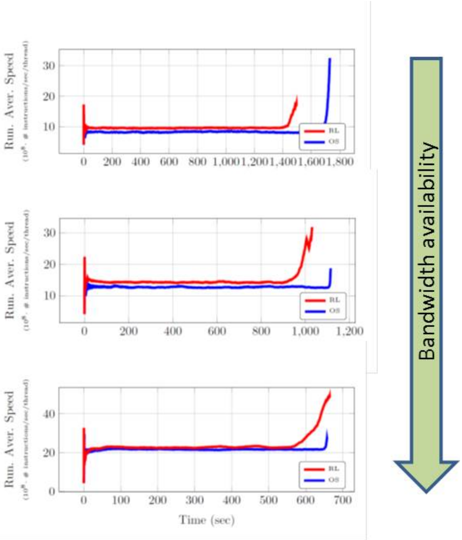
Non-uniform CPU availability

- **Learning-based Scheduler**
  - *Under large interferences:* completes up to 10% faster
  - *Under small interferences:* matches OS completion time
  - *Always:* achieves larger average speed / thread



Non-uniform CPU availability

- **Learning-based Scheduler**
  - *Under large interferences:* completes up to 10% faster
  - *Under small interferences:* matches OS completion time
  - *Always:* achieves larger average speed / thread
- **Note:**
  - Larger Average Speed / Thread $\not\Rightarrow$ Smaller Completion Time



Summary

Perturbed Learning Automata:

- perturbed learning automata for measurement-based optimization
- stochastic stability analysis in positive-utility games
- specialization in coordination games

Scheduling of Parallelized Applications:

- distributed learning framework for resource management in NUMA architectures
- increased average processing speed per thread in all experiments
- implies shorter completion times under large interferences