Exploiting Channel Memory in Multi-User Wireless Scheduling without Channel Measurement: Capacity Regions and Algorithms

Chih-ping Li (http://www-scf.usc.edu/∼chihpinl)
Michael J. Neely (http://www-rcf.usc.edu/∼mjneely)

University of Southern California
Los Angeles, CA, USA

WiOpt presentation, Avignon, France, June 1, 2010

Tech report ∼ arXiv:1003.2675v2

Supported by DARPA IT-MANET grant W911NF-07-0028, NSF Career grant CCF-0747525, and U.S. Army Research Laboratory.

Motivation

Wireless channels $\sim$ i.i.d. over time

- simple
- wlog if instantaneous states known for free

Motivation

Wireless channels $\sim$ i.i.d. over time

- simple
- wlog if instantaneous states known for free

Channels have memory

- fading channels $\sim$ FSMC [Zorzi'96]
- cognitive radio networks $\sim$ secondary users see Markovian channels [Zhao'07]

Motivation

Wireless channels $\sim$ i.i.d. over time

- simple
- wlog if instantaneous states known for free

Channels have memory

- fading channels $\sim$ FSMC [Zorzi'96]
- cognitive radio networks $\sim$ secondary users see Markovian channels [Zhao'07]

Channel memory could help

- probing overhead
- channel-blind transmission regime (e.g. wireless CSMA)

Motivation

Wireless channels $\sim$ i.i.d. over time

- simple
- wlog if instantaneous states known for free

Channels have memory

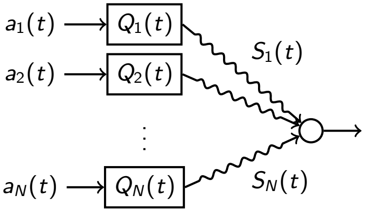
- fading channels $\sim$ FSMC [Zorzi'96]
- cognitive radio networks $\sim$ secondary users see Markovian channels [Zhao'07]

Channel memory could help

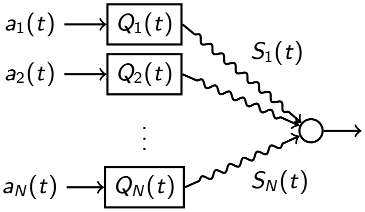
- probing overhead
- channel-blind transmission regime (e.g. wireless CSMA)

Q: How to exploit channel memory?

A Downlink Problem

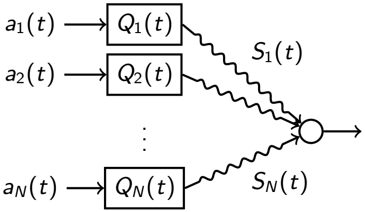


A Downlink Problem



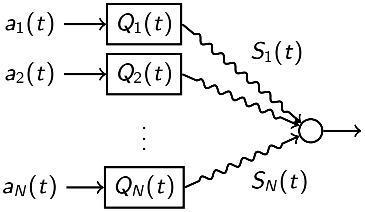
- slotted time $t \in \{0, 1, 2, \ldots\}$

A Downlink Problem

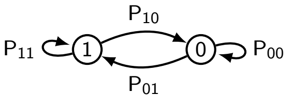


- slotted time $t \in \{0, 1, 2, \ldots\}$
- $a_n(t) \sim$ i.i.d.

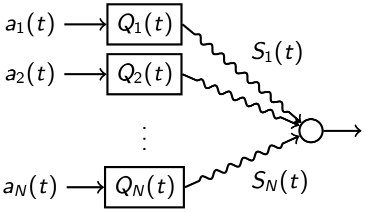
A Downlink Problem



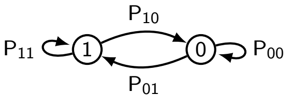
- slotted time $t \in \{0, 1, 2, \ldots\}$
- $a_n(t) \sim$ i.i.d.
- $S_n(t) \sim$ symmetric positively correlated Markov ON/OFF



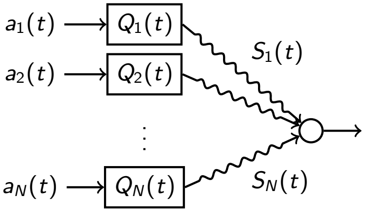
A Downlink Problem



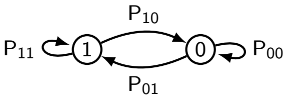
- slotted time $t \in \{0, 1, 2, \ldots\}$
- $a_n(t) \sim$ i.i.d.
- $S_n(t) \sim$ symmetric positively correlated Markov ON/OFF
- no channel probing



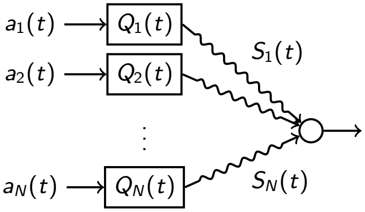
A Downlink Problem



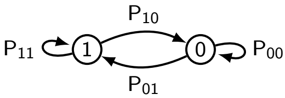
- slotted time $t \in \{0, 1, 2, \ldots\}$
- $a_n(t) \sim$ i.i.d.
- $S_n(t) \sim$ symmetric positively correlated Markov ON/OFF
- no channel probing
- serve one user / one packet per slot



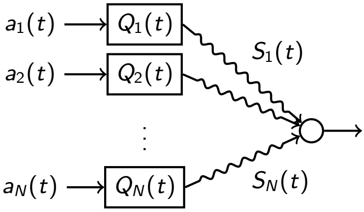
A Downlink Problem



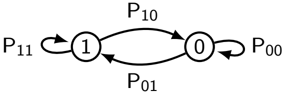
- slotted time $t \in \{0, 1, 2, \ldots\}$
- $a_n(t) \sim$ i.i.d.
- $S_n(t) \sim$ symmetric positively correlated Markov ON/OFF
- no channel probing
- serve one user / one packet per slot
- ACK/NACK feedback from the served user



A Downlink Problem



- slotted time $t \in \{0, 1, 2, \ldots\}$
- $a_n(t) \sim$ i.i.d.
- $S_n(t) \sim$ symmetric positively correlated Markov ON/OFF
- no channel probing
- serve one user / one packet per slot
- ACK/NACK feedback from the served user

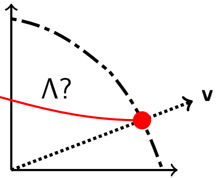


Goal: Network capacity region? Throughput-optimal policy?

Difficulty

- capacity region $\Lambda = N$-dimensional POMDP
- information state:

$$\omega_n(t) \triangleq \Pr\{S_n(t) = 1 \mid \text{observation history}\}$$

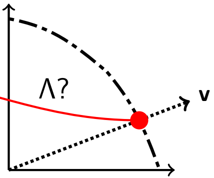


Difficulty

Difficulty

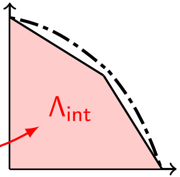
- capacity region Λ = N-dimensional POMDP
- information state:

$$\omega_n(t) \triangleq \Pr\{S_n(t) = 1 \mid \text{observation history}\}$$

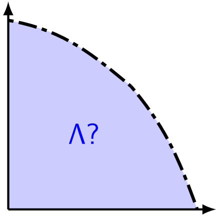


Idea

- forget POMDP
- Construct a good inner bound Λ_{int}



Road Map



network capacity region
$\sim$ POMDP

Road Map

network capacity region
$\sim$ POMDP

Road Map

network capacity region
$\sim$ POMDP

asymptotically tight when
(1) # of users is large
(2) arrivals are symmetric

Road Map

For any known $\boldsymbol{\lambda}$ interior to Λ_{int} $\sim$ a polynomial-time algorithm

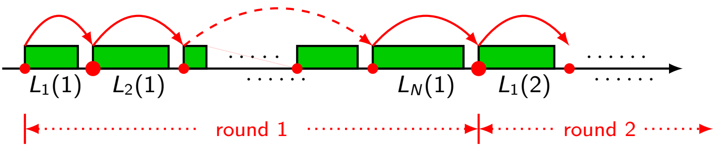
network capacity region $\sim$ POMDP

asymptotically tight when
(1) # of users is large
(2) arrivals are symmetric

Round Robin (a building block)

Round robin v1:

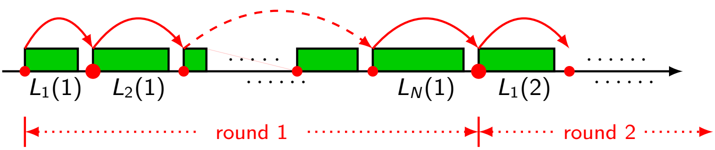
- Serve users in circular order ($1 \to 2 \to \cdots \to N$); stay with each one until receiving a NACK.



Round Robin (a building block)

Round robin v1:

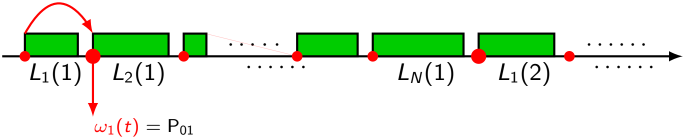
- Serve users in circular order ($1 \rightarrow 2 \rightarrow \cdots \rightarrow N$); stay with each one until receiving a NACK.
- Maximize sum throughput [Ahmad et al'09]



Round Robin (a building block)

Round robin v1:

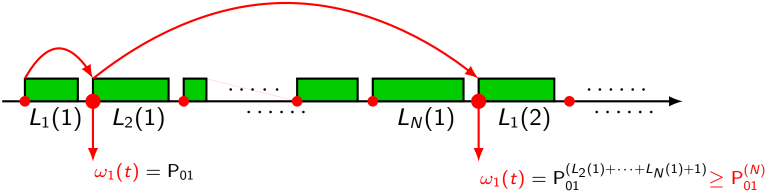
- Serve users in circular order ($1 \to 2 \to \cdots \to N$); stay with each one until receiving a NACK.
- Maximize sum throughput [Ahmad et al'09]
- Hard to analyze for $N > 2$



Round Robin (a building block)

Round robin v1:

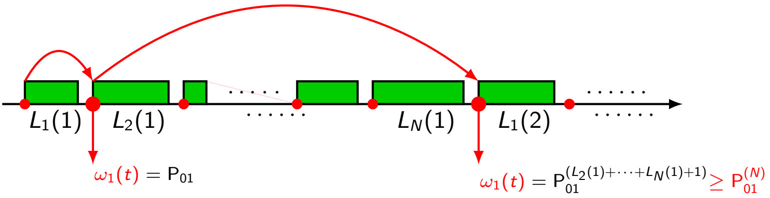
- Serve users in circular order ($1 \to 2 \to \cdots \to N$); stay with each one until receiving a NACK.
- Maximize sum throughput [Ahmad et al'09]
- Hard to analyze for $N > 2$



Round Robin (a building block)

Round robin v1:

- Serve users in circular order ($1 \to 2 \to \cdots \to N$); stay with each one until receiving a NACK.
- Maximize sum throughput [Ahmad et al'09]
- Hard to analyze for $N > 2$



Round robin v2:

- When switch to a new channel, set $\omega_n(t) = \mathrm{P}_{01}^{(N)}$.

Throughput of RRv2

Theorem

Sum throughput of RRv2 is

$$c_N \triangleq \frac{\mathrm{P}_{01}(1-(1-x)^N)}{x\mathrm{P}_{10}+\mathrm{P}_{01}(1-(1-x)^N)}, \quad x \triangleq \mathrm{P}_{01}+\mathrm{P}_{10}<1,$$

and each user shares c_N/N.

Throughput of RRv2

Theorem

Sum throughput of RRv2 is

$$c_N \triangleq \frac{\mathrm{P}_{01}(1-(1-x)^N)}{x\mathrm{P}_{10}+\mathrm{P}_{01}(1-(1-x)^N)}, \quad x \triangleq \mathrm{P}_{01}+\mathrm{P}_{10}<1,$$

and each user shares c_N/N.

Lemma

- *Sum throughput never exceeds* $c_\infty = \dfrac{\mathrm{P}_{01}}{x\mathrm{P}_{10}+\mathrm{P}_{01}}$.

Throughput of RRv2

Theorem

Sum throughput of RRv2 is

$$c_N \triangleq \frac{\mathrm{P}_{01}(1-(1-x)^N)}{x\mathrm{P}_{10}+\mathrm{P}_{01}(1-(1-x)^N)}, \quad x \triangleq \mathrm{P}_{01}+\mathrm{P}_{10} < 1,$$

and each user shares c_N/N.

Lemma

- *Sum throughput never exceeds* $c_\infty = \dfrac{\mathrm{P}_{01}}{x\mathrm{P}_{10}+\mathrm{P}_{01}}$.
- *Throughput loss of RRv2* $\leq c_\infty - c_N \leq c_\infty(1-x)^N$.

Throughput of RRv2

Theorem

Sum throughput of RRv2 is

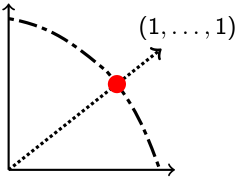
$$c_N \triangleq \frac{\mathsf{P}_{01}(1-(1-x)^N)}{x\mathsf{P}_{10}+\mathsf{P}_{01}(1-(1-x)^N)}, \quad x \triangleq \mathsf{P}_{01}+\mathsf{P}_{10} < 1,$$

and each user shares c_N/N.

Lemma

- *Sum throughput never exceeds* $c_\infty = \dfrac{\mathsf{P}_{01}}{x\mathsf{P}_{10}+\mathsf{P}_{01}}$.
- *Throughput loss of RRv2* $\le c_\infty - c_N \le c_\infty(1-x)^N$.

"asymptotically optimal for symmetry traffic"



Inner Capacity Bound

Randomized RRv2:

1. Randomly pick a subset of active users.
2. Run RRv2 for one round on active users with the order least-recently-used-first.

Inner Capacity Bound

Randomized RRv2:

1. Randomly pick a subset of active users.
2. Run RRv2 for one round on active users with the order least-recently-used-first.

Intuitive Performance

- $\phi \sim N$-dim binary vector; user n is active if nth entry is 1.
- $M(\phi) \sim$ number of active users in ϕ.

$$\underbrace{\frac{c_{M(\phi)}}{M(\phi)}}_{\text{per-user throughput}} \times \phi$$

Inner Capacity Bound

Randomized RRv2:

1. Randomly pick a subset of active users.
2. Run RRv2 for one round on active users with the order least-recently-used-first.

Intuitive Performance

- $\phi \sim N$-dim binary vector; user n is active if nth entry is 1.
- $M(\phi) \sim$ number of active users in ϕ.

$$\underbrace{\frac{c_{M(\phi)}}{M(\phi)}}_{\text{per-user throughput}} \times \phi$$

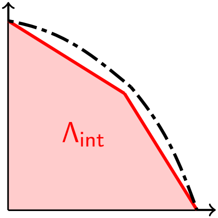
- Random mixing $\sim$ time sharing

Inner Capacity Bound

Theorem

The capacity region under randomized RRv2 is

$$\Lambda_{int} = \left\{ \boldsymbol{\lambda} \mid \boldsymbol{\lambda} \leq \boldsymbol{\mu},\ \boldsymbol{\mu} \in \mathrm{conv}\left(\left\{ \frac{c_{M(\phi)}}{M(\phi)} \phi \right\} \right) \right\}.$$

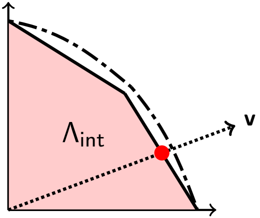


Inner Capacity Bound

Theorem

The capacity region under randomized RRv2 is

$$\Lambda_{int} = \left\{ \boldsymbol{\lambda} \mid \boldsymbol{\lambda} \leq \boldsymbol{\mu},\ \boldsymbol{\mu} \in \text{conv}\left(\left\{ \frac{c_{M(\phi)}}{M(\phi)} \phi \right\} \right) \right\}.$$

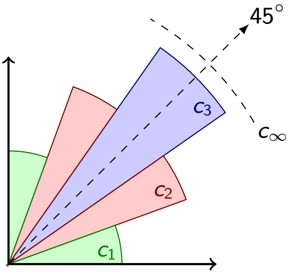


Lemma

If a directional vector $\mathbf{v} \sim$ *positive combination of binary vectors having* n *ones, then in that direction:*

max sum throughput $\geq c_n$

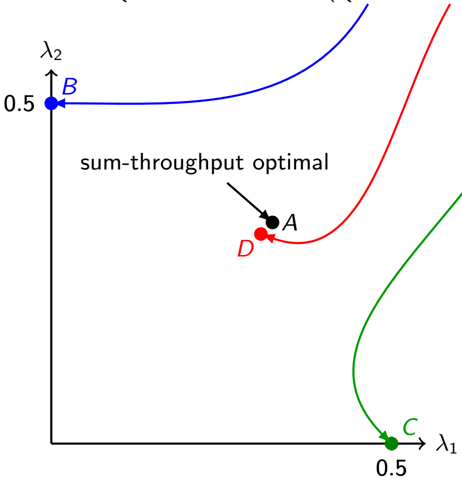
sum throughput loss $\leq c_\infty (1-x)^n$



Two-User Example

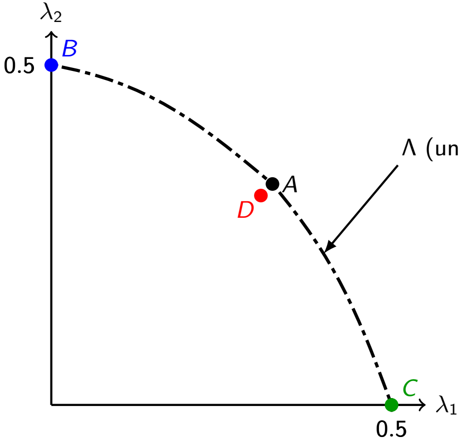
Inner Bound:

$$\Lambda_{\text{int}} = \left\{ \boldsymbol{\lambda} \mid \boldsymbol{\lambda} \leq \boldsymbol{\mu} \in \text{conv}\left(\left\{ \begin{bmatrix} 0 \\ c_1 \end{bmatrix}, \begin{bmatrix} c_2/2 \\ c_2/2 \end{bmatrix}, \begin{bmatrix} c_1 \\ 0 \end{bmatrix} \right\}\right) \right\}. \qquad (\mathsf{P}_{01} = \mathsf{P}_{10} = 0.2)$$



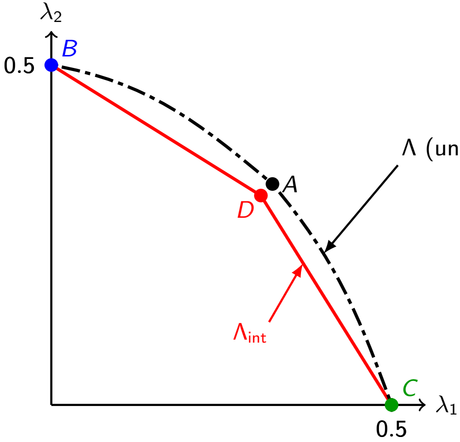
Inner Bound:

$$\Lambda_{\text{int}} = \left\{ \boldsymbol{\lambda} \mid \boldsymbol{\lambda} \leq \boldsymbol{\mu} \in \text{conv}\left(\left\{ \begin{bmatrix} 0 \\ c_1 \end{bmatrix}, \begin{bmatrix} c_2/2 \\ c_2/2 \end{bmatrix}, \begin{bmatrix} c_1 \\ 0 \end{bmatrix} \right\}\right) \right\}. \qquad (P_{01} = P_{10} = 0.2)$$



Inner Bound:

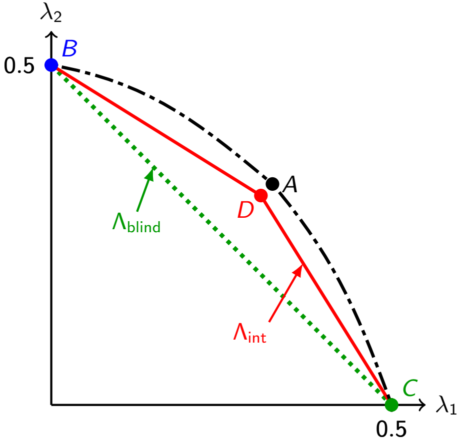
$$\Lambda_{\text{int}} = \left\{ \boldsymbol{\lambda} \mid \boldsymbol{\lambda} \leq \boldsymbol{\mu} \in \text{conv}\left(\left\{ \begin{bmatrix} 0 \\ c_1 \end{bmatrix}, \begin{bmatrix} c_2/2 \\ c_2/2 \end{bmatrix}, \begin{bmatrix} c_1 \\ 0 \end{bmatrix} \right\}\right) \right\}. \qquad (P_{01} = P_{10} = 0.2)$$



Two-User Example

Inner Bound:

$$\Lambda_{\text{int}} = \left\{ \boldsymbol{\lambda} \mid \boldsymbol{\lambda} \leq \boldsymbol{\mu} \in \text{conv}\left(\left\{ \begin{bmatrix} 0 \\ c_1 \end{bmatrix}, \begin{bmatrix} c_2/2 \\ c_2/2 \end{bmatrix}, \begin{bmatrix} c_1 \\ 0 \end{bmatrix} \right\} \right) \right\}. \qquad (\mathsf{P}_{01} = \mathsf{P}_{10} = 0.2)$$



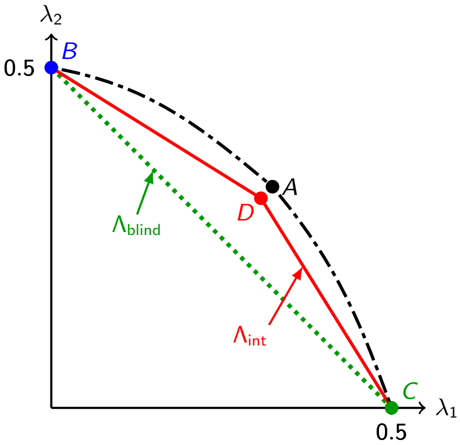
- If treating channels $\sim$ i.i.d.

sum throughput $\leq \pi_{\text{ON}}$

Two-User Example

Inner Bound:

$$\Lambda_{\text{int}} = \left\{ \boldsymbol{\lambda} \mid \boldsymbol{\lambda} \le \boldsymbol{\mu} \in \text{conv}\left(\left\{ \begin{bmatrix} 0 \\ c_1 \end{bmatrix}, \begin{bmatrix} c_2/2 \\ c_2/2 \end{bmatrix}, \begin{bmatrix} c_1 \\ 0 \end{bmatrix} \right\} \right) \right\}. \qquad (P_{01} = P_{10} = 0.2)$$



- If treating channels $\sim$ i.i.d.

  $$\text{sum throughput} \le \pi_{\text{ON}}$$

- Max throughput gain from channel memory:

  $$c_N - \pi_{\text{ON}} \xrightarrow{N \to \infty} \frac{P_{01}}{x\, P_{10} + P_{01}} - \frac{P_{01}}{P_{10} + P_{01}}$$

  $$x \triangleq P_{01} + P_{10}$$

Simple MaxWeight policy

Stabilize any $\boldsymbol{\lambda} \in \Lambda_{\text{int}}$:

- find a proper random mixture of $(2^N - 1)$ RRv2 policies so that $\boldsymbol{\mu} > \boldsymbol{\lambda}$
- impractical

Simple MaxWeight policy

Stabilize any $\boldsymbol{\lambda} \in \Lambda_{\text{int}}$:

- find a proper random mixture of $(2^N - 1)$ RRv2 policies so that $\boldsymbol{\mu} > \boldsymbol{\lambda}$
- impractical

Q-dependent dynamic round robin (QRR)

1. Observe $Q_n(t)$, and find ϕ^* that maximizes

$$f(\overrightarrow{Q}(t)) \triangleq \sum_{n:\text{active}} \left[\frac{Q_n(t) \mathsf{P}_{01}^{(M(\phi))}}{\mathsf{P}_{10}} - \frac{\mathsf{P}_{10} + \mathsf{P}_{01}^{(M(\phi))}}{\mathsf{P}_{10}} \sum_{n=1}^{N} Q_n(t) \lambda_n \right]$$

2. Run RRv2 on active users of ϕ^* for one round by least-recently-used-first.

Simple MaxWeight policy

Stabilize any $\boldsymbol{\lambda} \in \Lambda_{\text{int}}$:

- find a proper random mixture of $(2^N - 1)$ RRv2 policies so that $\boldsymbol{\mu} > \boldsymbol{\lambda}$
- impractical

Q-dependent dynamic round robin (QRR)

1. Observe $Q_n(t)$, and find ϕ^* that maximizes

$$f(\overrightarrow{Q}(t)) \triangleq \sum_{n:\text{active}} \left[\frac{Q_n(t)\mathsf{P}_{01}^{(M(\phi))}}{\mathsf{P}_{10}} - \frac{\mathsf{P}_{10} + \mathsf{P}_{01}^{(M(\phi))}}{\mathsf{P}_{10}} \sum_{n=1}^{N} Q_n(t)\lambda_n \right]$$

2. Run RRv2 on active users of ϕ^* for one round by least-recently-used-first.

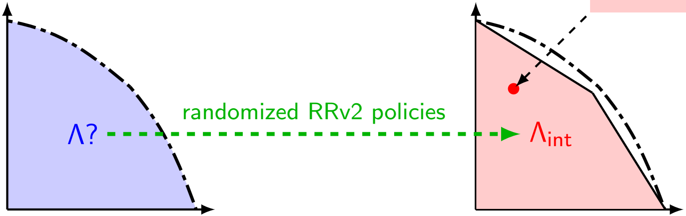
Theorem

For any $\boldsymbol{\lambda}$ interior to Λ_{int}, policy QRR *stabilizes the network.*

- proved by a frame-based variable-length Lyapunov drift argument
- QRR $\sim$ polynomial time algorithm

Wrap Up

QRR policy stabilizes the network for all $\boldsymbol{\lambda}$ interior to Λ_{int}.



network capacity region $\sim$ POMDP

asymptotically tight when
(1) # of users is large
(2) arrivals are symmetric

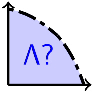
$$\text{max channel memory gain} = \frac{P_{01}}{x\,P_{10} + P_{01}} - \frac{P_{01}}{P_{10} + P_{01}} \quad (x = P_{01} + P_{10} < 1)$$

Applications / Future Work

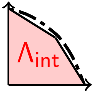
- Channel measurement and delayed information in multi-user wireless scheduling
  - Limited channel probing
  - Other QoS metrics
- Opportunistic spectrum access in cognitive radio networks
  - State of the art is POMDP for single-user case. Lots of potentials here!
- How channel memory can help in modern network protocols
  - Random access (e.g. CSMA) in wireless networks?
- Transform POMDP and restless bandit into stochastic network control problems

A New Methodology for Restless Bandit

In this paper:



all feasible solutions
to a restless bandit

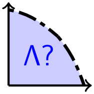


feasible solutions
of practical interest

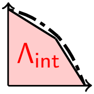
Let $\boldsymbol{\lambda}^*$ be the optimal solution to a restless bandit problem over Λ_{int}.
If $\boldsymbol{\lambda}^*$ is known, QRR achieves it in the network.

A New Methodology for Restless Bandit

In this paper:



all feasible solutions to a restless bandit



feasible solutions of practical interest

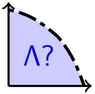
Let $\boldsymbol{\lambda}^*$ be the optimal solution to a restless bandit problem over Λ_{int}.
If $\boldsymbol{\lambda}^*$ is known, QRR achieves it in the network.

What if....

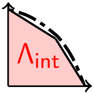
.... a QRR-like policy achieves $\boldsymbol{\lambda}^*$ without knowing it in advance, then we are actually solving (approximately) a high-dimensional restless bandit!

A New Methodology for Restless Bandit

In this paper:



all feasible solutions to a restless bandit

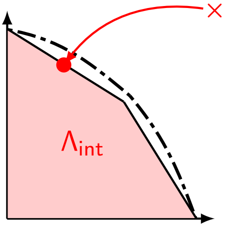


feasible solutions of practical interest

Let $\boldsymbol{\lambda}^*$ be the optimal solution to a restless bandit problem over Λ_{int}.
If $\boldsymbol{\lambda}^*$ is known, QRR achieves it in the network.

What if....

.... a QRR-like policy achieves $\boldsymbol{\lambda}^*$ without knowing it in advance, then we are actually solving (approximately) a high-dimensional restless bandit!



- pour unlimited traffic into the network
- perform admission control and utility maximization [Neely, Modiano, Li, ToN'08] to let the network learn the optimal solution