Design for Low-Power IoT Systems: Factored MDPs and Application Examples

Adrian E. Sapio and Shuvra S. Bhattacharyya

Maryland DSPCAD Research Group
http://www.ece.umd.edu/DSPCAD/home/dspcad.htm

Department of Electrical and Computer Engineering, and
Institute for Advanced Computer Studies
University of Maryland, College Park, 20742, USA

May. 27, 2018

ISCAS 2018 Tutorial, Florence, Italy, May 27, 2018

Design for Low-Power IoT Systems

Module 1: Transport Triggering Approach
Jarmo Takala, Tampere University of Technology, Finland

Module 2: Coarse-Grained Reconfigurable Acceleration Units
Francesca Palumbo, Università degli Studi di Sassari, Italy

Module 3: Markov Decision Processes and Power/Performance/Thermal (PPT)
Marilyn Wolf, Georgia Tech., USA

Module 4: Factored MDPs and Application Examples
Shuvra Bhattacharyya, U. Maryland, USA and Tampere U. Technology, Finland

Module Outline: Factored MDPs and Application Examples

- Factored Markov Decision Processes (MDPs)
- MDP-based System Design Example: Reconfigurable Digital Channelizer Design
- Modeling Reconfiguration Delays
- Comparison with Manually Generated Policies
- Comparison with Prior Work On Adaptive Reconfiguration
- Summary

Factored MDPs — Motivation

MDPs involve “enumeration of state space”

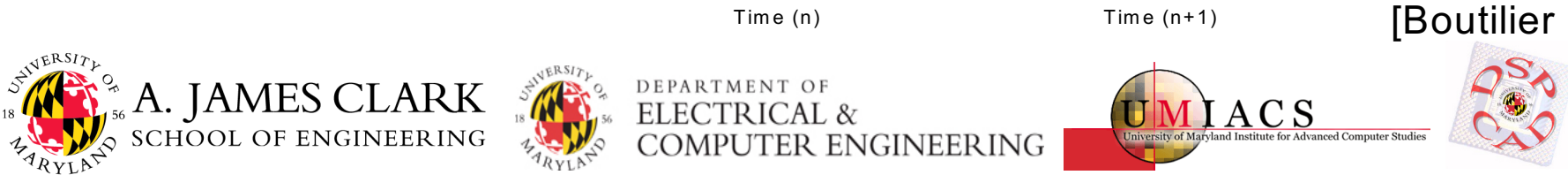
- MDP state space is generally made up of multiple variables
- Exponential increase with each new variable or value
- Channelizer example state space:
  - 3,328 states in MDP state space
  - 11 actions in MDP action space
  - MDP requires one State Transition Matrix (STM) of size 3328x3328
    ~ 11M entries per action
  - 11M * 11 ~ **121MB Model Size**
- Adding other subsystems quickly makes model size and solver runtime infeasible
- With factored model: **121MB → 66KB**

Factored MDPs

[Boutilier ’95]: Introduces concept of Factored MDPs to AI community

- Addresses “curse of dimensionality” in MDPs
- Motivation
  - Some parts of an MDP state space generally don’t depend on each other
  - This independence can be exploited to represent the global state more compactly
- Can lead to significant reduction in model size and solver runtime
- Several published algorithms in this area, e.g.:

[Guestrin ‘03], [Szita ‘08]



[Boutilier ‘95]

Factored STMs (1 of 2)

- In MDPs, we have discrete states that are elements of a discrete state space
- … and discrete actions that are elements of a discrete action space.
- It is necessary to define the probabilities of transition from each state to every other state, *for each action*.
- When stored in a computer, we refer to this as the State Transition Matrices (STMs).
- The STMs are NA separate matrices, each of size NS x NS.
- Example for NS=3, NA=2:

$$s \in S = \{s_1, s_2, \ldots s_{NS}\}$$
$$a \in A = \{a_1, a_2, \ldots a_{NA}\}$$

$$\Pr[s'|s, a]$$

STM for Action 1

	s_1	s_2	s_3
s_1	.1	.2	.7
s_2	.3	.3	.4
s_3	0	1	0

STM for Action 2

	s_1	s_2	s_3
s_1	.2	.7	.1
s_2	1	0	0
s_3	0	.5	.5

Each row must sum to 1.

Collectively, the STMs are a lookup table of size (NS x NS x NA) elements.

Probability of transition from State 3 to State 2 when taking action 2 = 0.5

Factored STMs (2 of 2)

- If the state-space is composed of two or more state variables, and can be decomposed into two or more subsets, it can be possible to ”factor" the STMs.
- This can lead to large reductions in storage requirements for the STMs.
- Whether this is possible or not depends on the causal relationships and dependencies between the state variables that make up the state space.

Example:

- The state is made up of two state variables x and y. NX = 10, NY=5, A=3.
- x must be in the state space, but is independent of both y and a.
- In this case, the STMs can be written as a *product of terms*.
- We call this a **Factored MDP**.
- The size required for the STMs has been reduced considerably:
  - From: NXNY x NXNY x NA = 7500 elements
  - To: (NX x NX) + (NY x NY x NA) = 100 + 75 = 175 elements

Product Formulation of STMs

$$x \in X, y \in Y$$

$$s = (x, y) \in S = \{\{X\} \times \{Y\}\}$$

$$s \in S = \{(x_1, y_1), (x_1, y_2), \dots (x_2, y_1) \dots\}$$

$$\Pr[s'|s, a] = \Pr[(x', y')|(x, y), a)]$$

$$\Pr[s'|s, a] = \Pr[x'|x] \cdot \Pr[y'|y, a]$$

Due to the independence of *x* with respect to *y* and *a*

Factored MDPs — Example (1)

Consider a simplified decision framework for an autonomous robot:

- Use MDP "State" to capture current state of environment and robot.
- Use MDP "Action" to list possible decisions the robot can make.
- This is a simplified/adapted version of an example from [Boutilier '95] .

Define state space as the combination of Boolean state variables W,R,U,O:

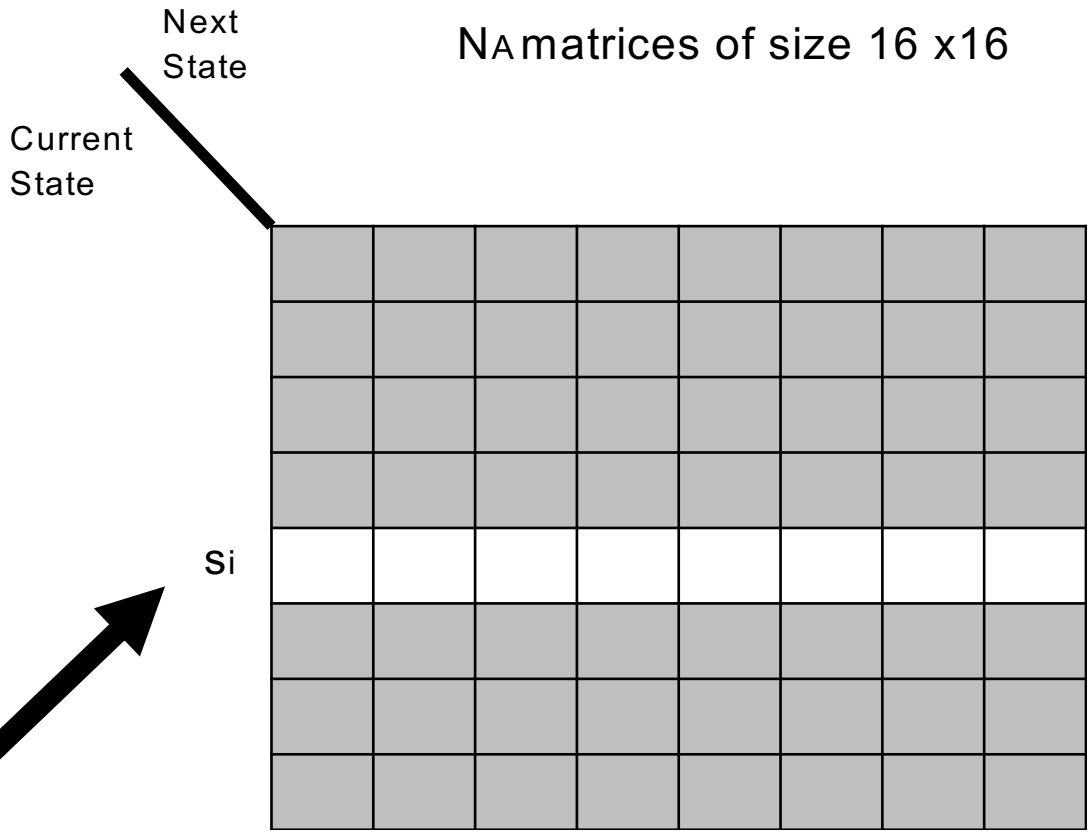
- W: 1=Robot is Wet, 0=Robot is not wet
- R: 1=It is raining outside, 0=It is not raining outside
- U: 1=Robot has an umbrella, 0=Robot does not have an umbrella
- O: 1=Robot is outside, 0=Robot is not outside

Define the action space as the decisions that the robot can make (in pursuit of some long-term goal):

- Action 1: Robot goes outside
- Action 2: Robot picks up an object
- …
- Action NA

Factored MDPs — Example (2)

Example STM definition:



N_A matrices of size 16 x16

- Each row is a discrete probability distribution.
- Contains the probability of transition from one state to another state.
- Transition from all states to all other states must be specified.
- One matrix must be created for each possible action.
- All combinations of all state variables must be enumerated to create the rows and columns.

Example starting state s_i:
W=0, R=0, U=0, O=0
(Robot is not wet, it is not raining, Robot has no umbrella, Robot is not outside)

Factored MDPs — Example (3)

Create probability mass function (PMF) for this state, use it to populate a single row of one STM:

Action: Robot goes outside

How does this action affect each of the state variables?
Informally…

R (raining):
- There is a 30% chance of rain today. (Not affected by robot)

O (outside):
- If the robot decides to go outside, then the robot will be outside.

W (wet):
- If the robot was already wet, the robot will continue to be wet.
- If the robot goes outside with no umbrella, and it is raining, the robot will get wet.

U (umbrella):
- The robot will continue to have (or not have) an umbrella without change, since the action does not involve acquiring an umbrella.

Factored MDPs — Example (4)

Compute STM row (PMF) for State s_i, Action "Go Outside":

Must specify all combinations of state variables

O	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
U	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
R	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
W	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
	0	0	0	0	0	0	0	0	-	-	-	-	-	-	-	-
	0	0	0	0	0	0	0	0	-	-	-	-	0	0	0	0
	0	0	0	0	0	0	0	0	-	-	0	-	0	0	0	0
	0	0	0	0	0	0	0	0	-	0	0	-	0	0	0	0
	0	0	0	0	0	0	0	0	.7	0	0	.3	0	0	0	0

- Pr[Not Outside] = 0
- Pr[Having Umbrella] = 0
- Pr[Being dry in the rain] = 0
- Pr[Getting wet in sun] = 0
- Pr[Rain] = 0.3

STM row for this state, action: [0 0 0 0 0 0 0 0 .7 0 0 .3 0 0 0 0]

Factored MDPs — Example (5)

To compute the STMs, we must repeat this process for all states and all actions.

- Curse of dimensionality:
  - Adding another state variable increases size of the state space:
    e.g., increases from 16 → 32 states in our case
  - Real world state variables are not necessarily Boolean

- STMs are very large and must be stored, and then consumed by an MDP solver to generate control policies.

- Conceptually, there is a structure embedded within the STMs that can be exploited for more compact representations and processing efficiency.

- Factorization of current example:
  - An STM row is a tabular format of: $\Pr[W', R', U', O' | W, R, U, O, a]$
  - Using basic probability theorems (Bayes theorem, conditional independence, etc.), and knowledge of MDP environment, we can re-write this in a more compact format …

Factored MDPs — Example (6)

$$\Pr[W', R', U', O' | W, R, U, O, a] \quad = 256 \text{ elements}$$

- Probability of rain is not dependent on other state variables or action.
- Probability of being outside is only affected by action and previous state of *O* (not previous state of *W, R*, *U*).
- Probability of having an umbrella is only affected by action and previous state of *U* (not previous state of *W*, *R*, *O*).
- Thus, we can compress a 16 x 16 = 256 element matrix into a 42 element matrix (for a given action):

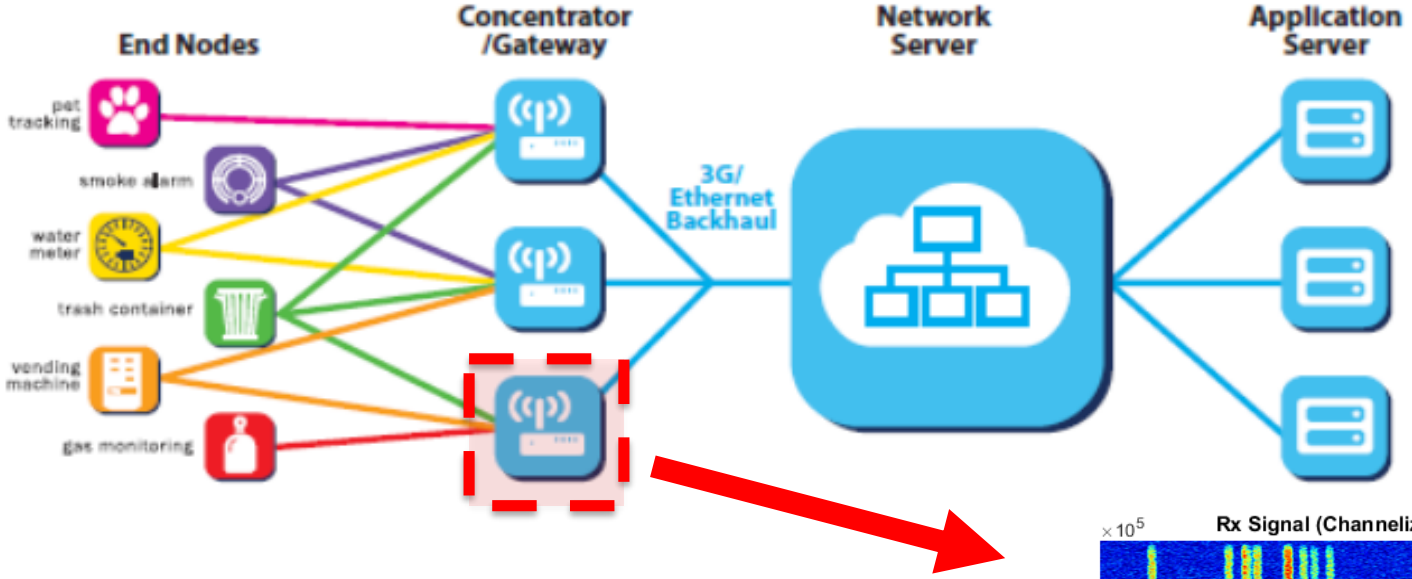
$$\underbrace{\Pr[R']}_{2} \; \underbrace{\Pr[O' | O, a]}_{4} \; \underbrace{\Pr[U' | U, a]}_{4} \; \underbrace{\Pr[W' | W, R, U, O, a]}_{32}$$

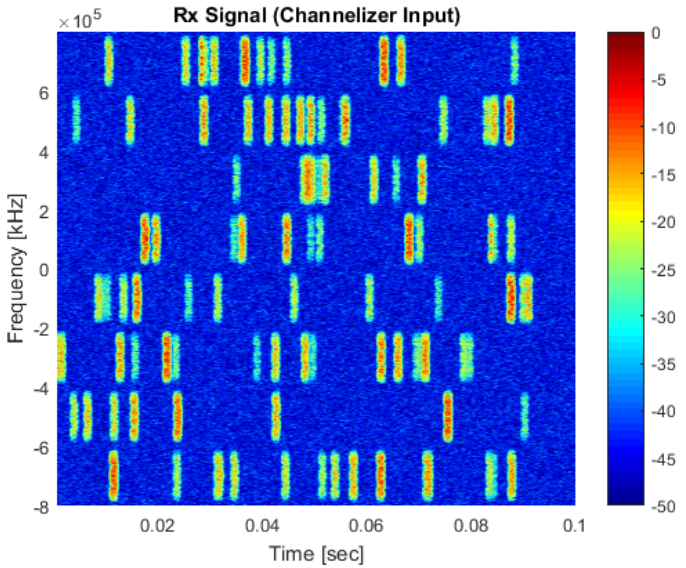
$$2 + 4 + 4 + 32 = 42 \text{ elements}$$

Module Outline: Factored MDPs and Application Examples

- Factored Markov Decision Processes (MDPs)
- MDP-based System Design Example: Reconfigurable Digital Channelizer Design
- Modeling Reconfiguration Delays
- Comparison with Manually Generated Policies
- Comparison with Prior Work On Adaptive Reconfiguration
- Summary

Channelizer Application:
Smart Cities Base Station

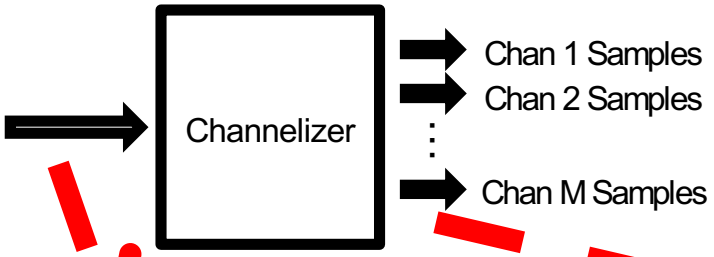




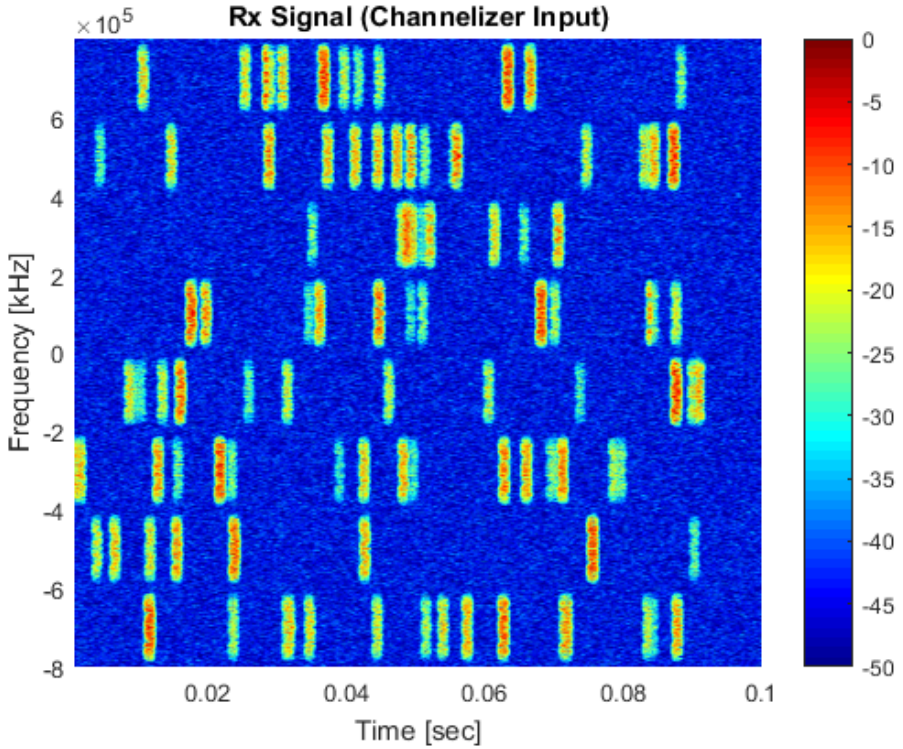
Front End Channelization needed for Frequency Division Multiplexing

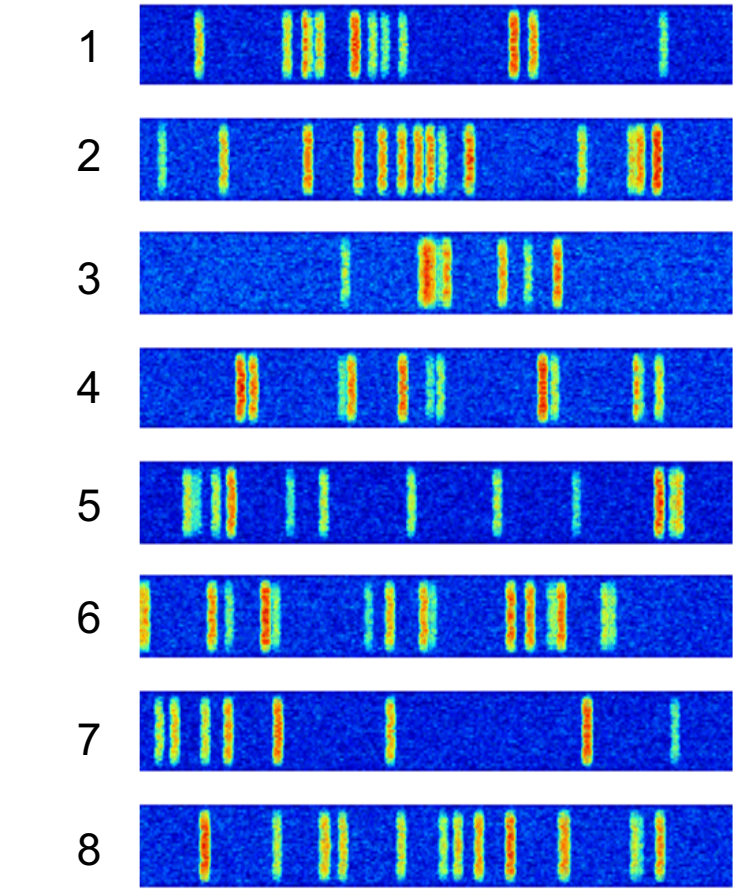
- 72 channels used in US 915 MHz Band by some IoT base stations

Receiver Processing



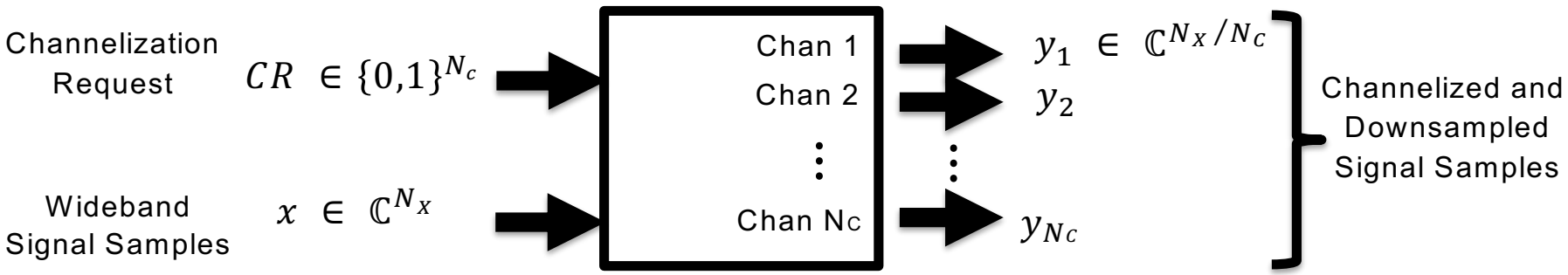
What is the most strategic algorithm for separating the signals?





Digital Channelizer

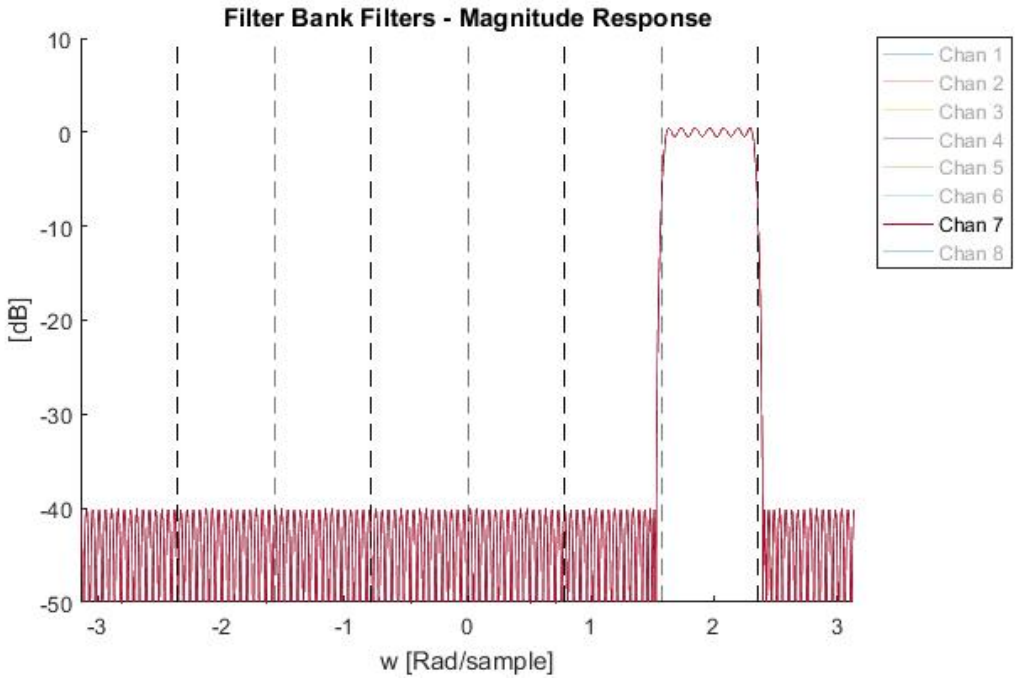
Nc = # of sub-channels
Nx = # of signal samples per frame



Example:

Number of Channels: Nc = 8

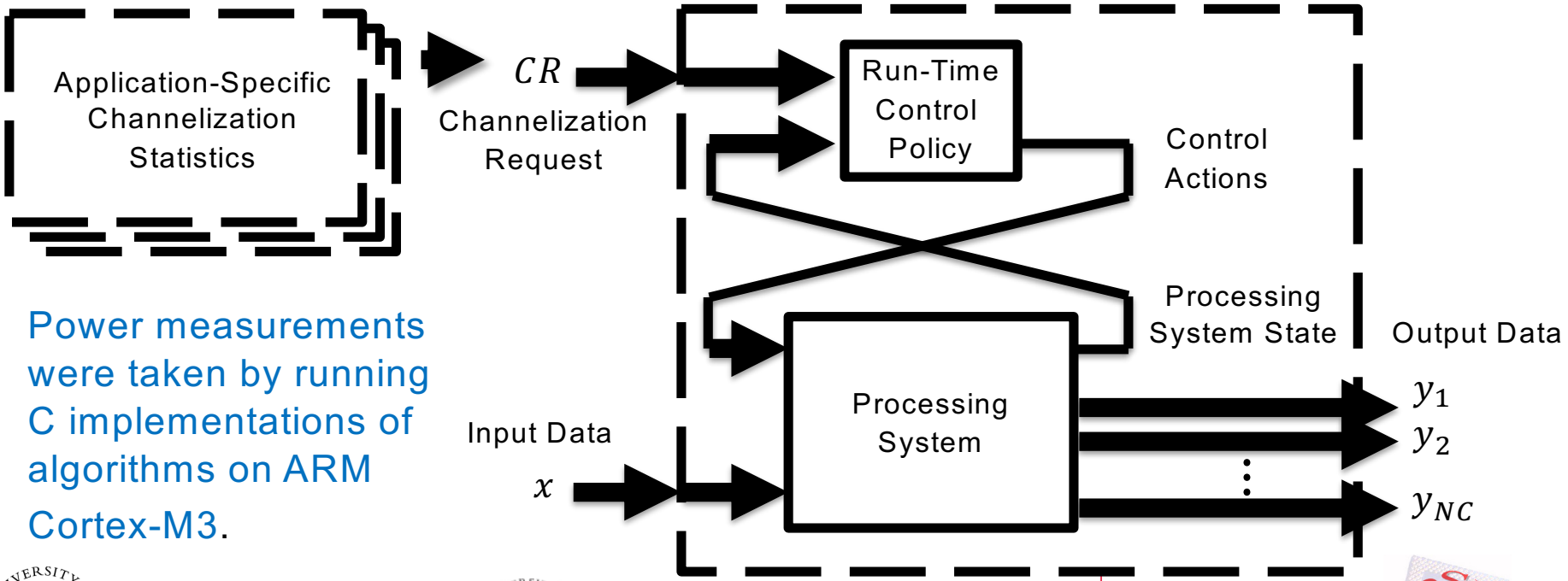
$$CR = \{0,0,0,0,0,0,1,0\}$$



- Model CR time series as a stochastic process
- Can we model its statistics?

Reconfigurable Channelizer System

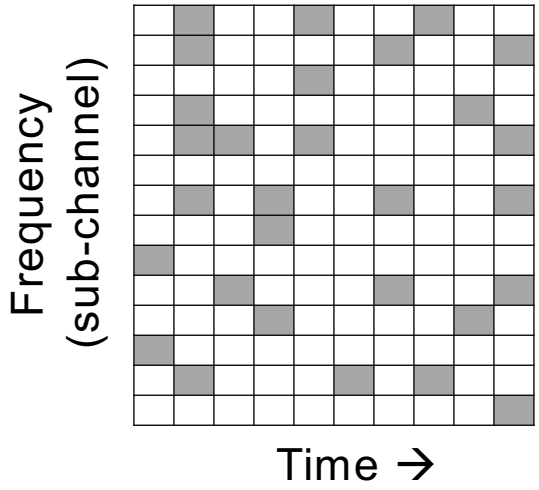
- A simulation for a dynamic channelizer was developed with 3 top-level processing states [Sapio '16]:
  - (a) DCM: tunable decimation filter (8 subconfigurations);
  - (b) DFTFB: DFT filter bank;
  - (c) Sleep.
- A 2-frame delay was assumed to switch between algorithms.



Power measurements were taken by running C implementations of algorithms on ARM Cortex-M3.

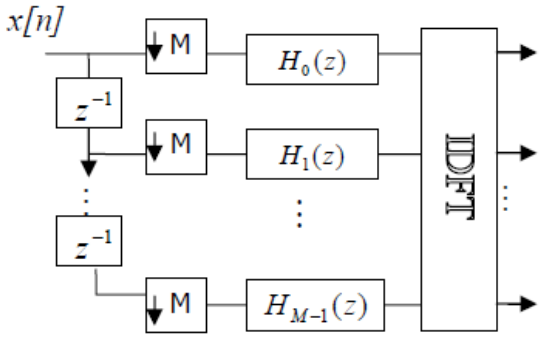
Use Case *A*: Dynamic Spectrum Access

Channelization Requests



Requests are modeled as IID Bernoulli across both time and subchannel dimensions

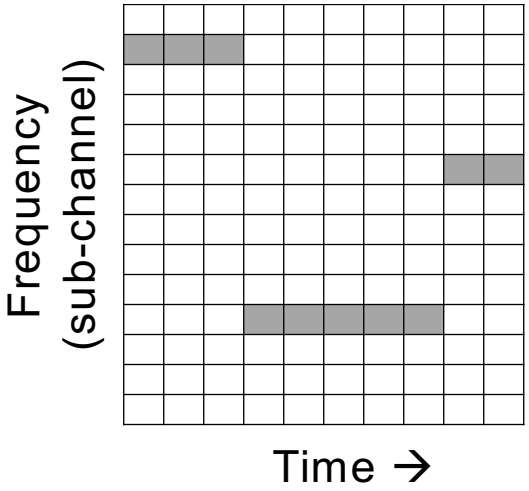
Selected DSP Algorithm



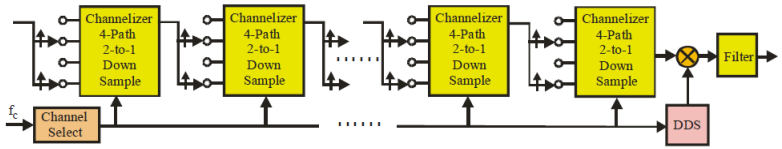
Polyphase DFT Filter Bank ("DFTFB") → outputs all subchannels at all times

Use Case *B*: Sequential Sensing

Channelization Requests



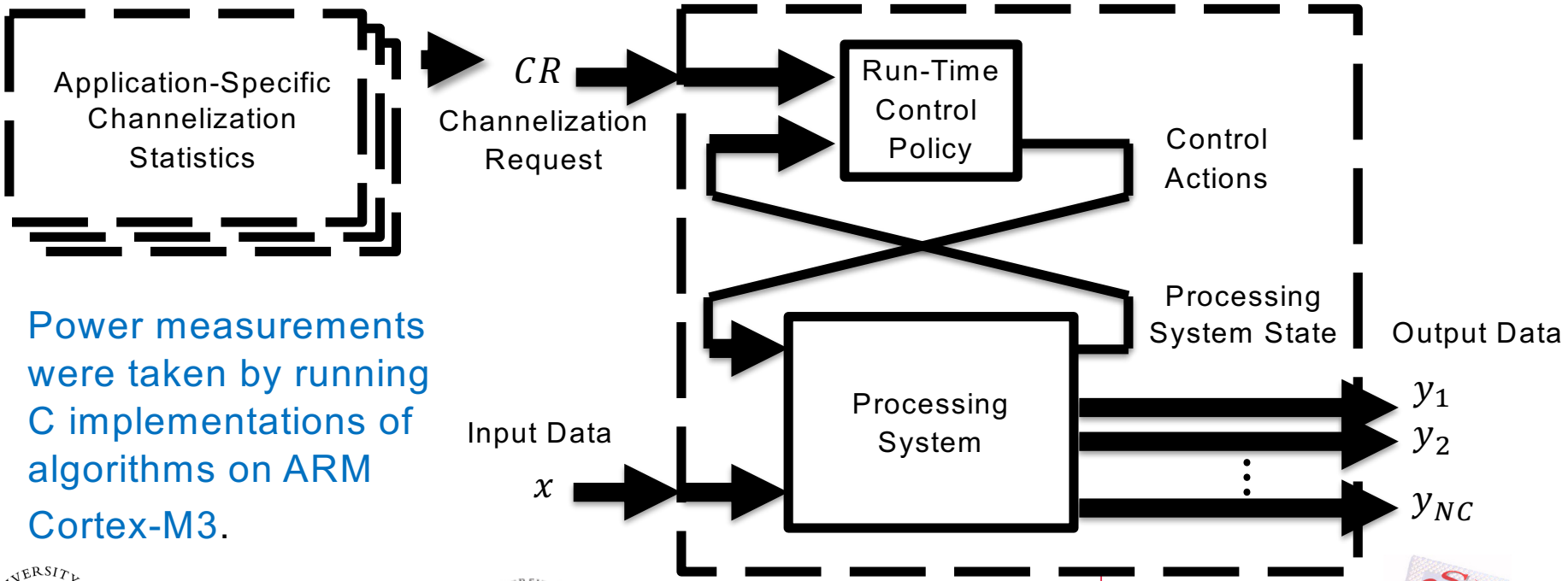
Selected DSP Algorithm



Tunable polyphase decimation filter ("DCM") → extracts a single subchannel

Reconfigurable Channelizer System (Revisited)

- A simulation for a dynamic channelizer was developed with 3 top-level processing states [Sapio '16]:
  - (a) DCM: tunable decimation filter (8 subconfigurations);
  - (b) DFTFB: DFT filter bank;
  - (c) Sleep.
- A 2-frame delay was assumed to switch between algorithms.



Power measurements were taken by running C implementations of algorithms on ARM Cortex-M3.

Digital Channelizer State Space

We define the state space as all the possible states of the channelization requests, together with all the possible states of the system configurations:

$$S = \{CR\} \times \{CF\}$$

"CR" = Channelization Requests:
Number of channels: Nc = 8

$$\{CR\} = \{[0,0,0,0,0,0,0,0], [0,0,0,0,0,0,0,1], [0,0,0,0,0,0,1,0], \ldots [1,1,1,1,1,1,1,1]\}$$

256 CR States Total

"CF" = System Configurations :
Number of channels: Nc = 8
2 algorithms: DCM, DFTFB
2 sleep modes: DCM, DFTFB
8 Subconfigurations in DCM
2 transition states

13 CF States Total

Table 1 Categories of processing states and their properties.

State Category	Num States	Num Channels Provided	Average Power
SLEEP	2	0	5.36 μW
DCM	8	1	7.61 mW
DFTFB	1	8	17.92 mW
Trans. DFTFB	1	0	10.25 mW
Trans. DCM	1	0	10.25 mW

Digital Channelizer Action Space

We define the action space as one each action for each system configuration (not including transition states)

$$A = \left\{ \begin{array}{c} Sleep\ (DCM), \\ Sleep\ (DFTFB), \\ DCM\colon Chan1, \\ DCM\colon Chan2, \\ DCM\colon Chan3, \\ DCM\colon Chan4, \\ DCM\colon Chan5, \\ DCM\colon Chan6, \\ DCM\colon Chan7, \\ DCM\colon Chan8, \\ DFTFB \end{array} \right\}$$

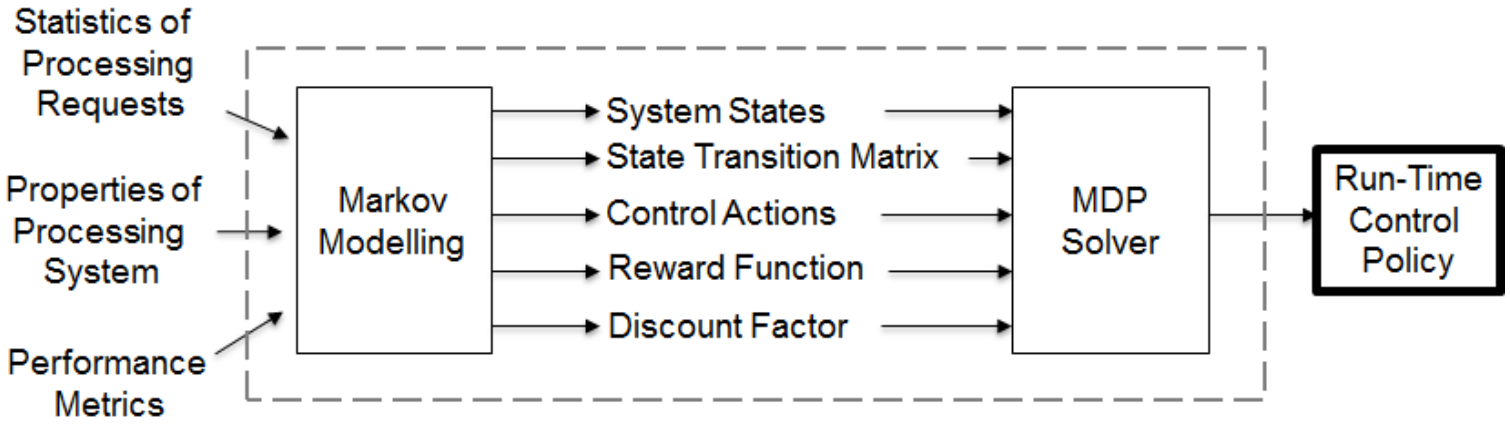
Table 1 Categories of processing states and their properties.

State Category	Num States	Num Channels Provided	Average Power
SLEEP	2	0	5.36 μW
DCM	8	1	7.61 mW
DFTFB	1	8	17.92 mW
Trans. DFTFB	1	0	10.25 mW
Trans. DCM	1	0	10.25 mW

11 Action States in Total

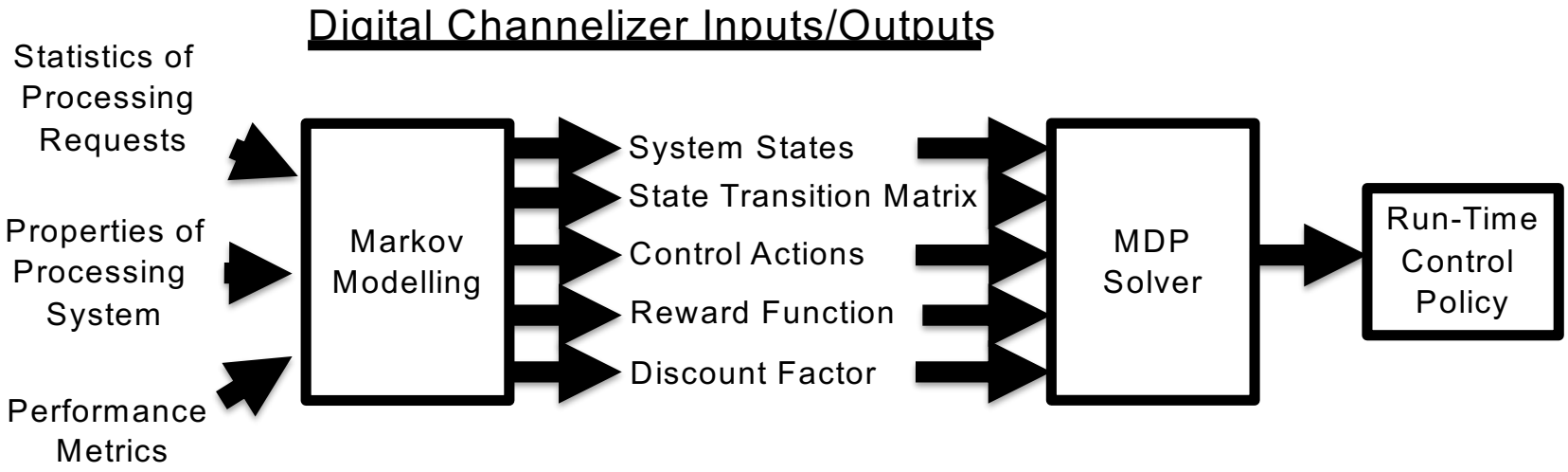
Markov Modeling (1)

- Algorithmic method to convert measurable quantities into control policies
- Enables autonomous adaptation of system level configurations



- Processing request can be implicit or explicit
- Properties can be physical constraints
  - Programmable logic size
  - Transition time between states

DSP Design Problem:
Digital Channelizer for N sub-channel FDM signal



States of Service Requester (chann. requests): $sr \in SR = \{1, 2, \ldots, N_{SR}\}$

States of Service Provider (processing config.): $sp \in SP = \{1, 2, \ldots, N_{SP}\}$

MDP State Space : $s \in S = \{SR \times SP\}$

Factorization :

$$STM = \Pr[\, s^{(n+1)} |\, s^{(n)}, a^{(n)}]$$
$$= \Pr[\, sr^{(n+1)}, sp^{(n+1)} |\, sr^{(n)}, sp^{(n)}, a^{(n)}]$$
$$= \Pr[\, sr^{(n+1)} |\, sr^{(n)}]\Pr[\, sp^{(n+1)} |\, sp^{(n)}, a^{(n)}]$$

Rewards: Multiobjective Optimization

- In an MDP framework, a Reward Function is a mapping:
- We use a *scalarization* approach to steer the MDP solver with multiple performance metrics:
  - i=1 “Service Rate” = Number of output channels produced.
  - i=2 “Power Consumption” = Average power consumed by processing platform.
- Metrics must adhere to convention of 1=most rewarded, 0=least rewarded.
- Metrics can be known at design time or measured at runtime.
- At design time, the application engineer “instructs” the system in terms of r .
- → Relative importance of each metric, instead of static rules for reconfiguration.

$$R(s,a,s'): \mathcal{S}\times\mathcal{A}\times\mathcal{S} \rightarrow \mathbb{R}$$

$$f_i(s,a,s'): \mathcal{S}\times\mathcal{A}\times\mathcal{S} \rightarrow [0,1]$$

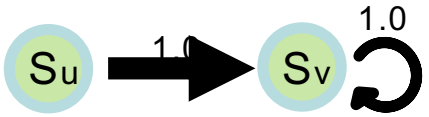
$$R(s,a,s') = \sum_i r_i \, f_i(s,a,s')$$

Module Outline: Factored MDPs and Application Examples

- Factored Markov Decision Processes (MDPs)
- MDP-based System Design Example: Reconfigurable Digital Channelizer Design
- Modeling Reconfiguration Delays
- Comparison with Manually Generated Policies
- Comparison with Prior Work On Adaptive Reconfiguration
- Summary

Modeling Reconfiguration Delays

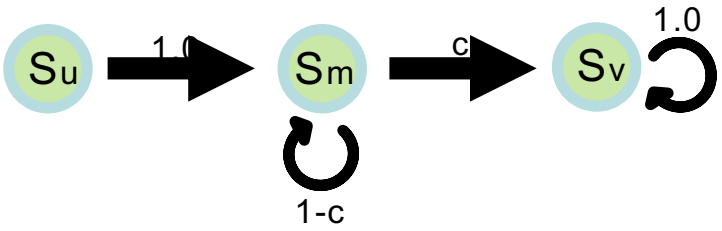
- MDPs have been successfully used in applications requiring multi-step decisions, e.g., motion planning.
- Can be used to model transitions in reconfigurable platforms.
- Simple example:
  - Platform Configuration 1 : State u;
  - Platform Configuration 2 : State v.



- Consider taking action “a”, which puts the platform in Configuration #2.
- Modeled behavior:
  - At timestep n: Platform is in State u.
  - At timestep n+1: Platform is in State v.
- What happens if reconfiguration delays take longer than 1 time step?
  - System dynamics will not match modeled dynamics.
  - MDP-generated policies will be optimal for the model, but not optimal for control of the actual system.

Modeling Reconfiguration Delays

- Strategy: use ***Transition States***.
- Create a state *m* in the MDP state space that does *not* correspond to a specific platform configuration.
- Rather, the state can *m* be defined as being "*in transition from state u to state v*".
- Models effect of taking action "a", which triggers a deterministic (multi-time-step) transition in the actual system. (In general, we can relax this requirement of determinacy).
- MDPs require all transitions (even deterministic ones) to be modeled as stochastic events.



- Modeled as a stochastic transition:
  1. Transition (with probability 1) to the transition state m.
  2. Transition (with probability c) to the final state v.
  3. Stay (with probability 1) in the final state v (until another action is selected to leave state v).

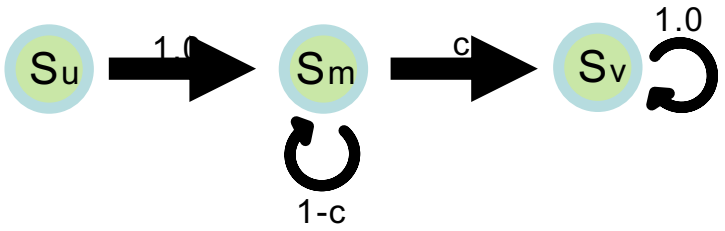
Transition States (1)

- Added to model reconfigurations within a processing resource with duration longer than one frame (MDP time step).
- Represents transition through undesirable temporary state to more desirable final state.
- STM computation for a transition with large delay.

(Conditional) State Transition Matrix

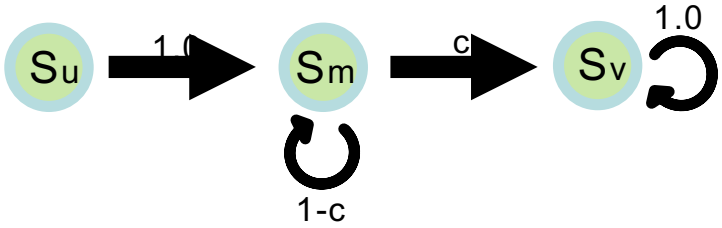
Current State \ Next State	Su	Sm	Sv
Su	0	1	0
Sm	0	1-c	c
Sv	0	0	1

(Conditional) Markov Chain Diagram



$$c = \left[floor\left(\frac{T_{u,v|w}}{T_F} \right) \right]^{-1}$$

Transition States (2)



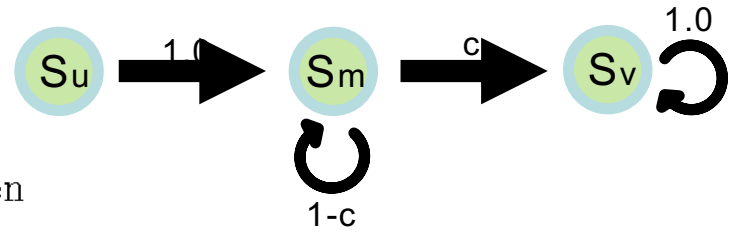
- s_u, s_m, s_v are 3 distinct states in the MDP state-space.
- Only states u and v correspond to actual states in the system.
- State m exists *only in the model.*
- State m is created to model a multi-step (deterministic) transition using a more compact form.
- The edge weights in the graph show the transition probabilities in the transformed system model (when taking the action that causes the modeled transition).

Deriving the Trans. State Parameter *c*

- We can model observations in the transition state m as a Bernoulli random variable.
- The two possible outcomes are: remaining in the transition, and completing the transition.
- The Maximum Likelihood Estimator of the parameter c in this context can be derived as:

$$c = \left[floor\left(\frac{T_{u,v|w}}{T_F}\right) \right]^{-1}.$$

- Here, $T_{u,v|w}$ is the transition time from state u to v when action w is received in state u,
- and T_F is the fixed, constant time step duration of the discrete-time iteration of the MDP.
- T_F corresponds to how often the controller determines the state, and uses that to look up the action to take, from a given policy.



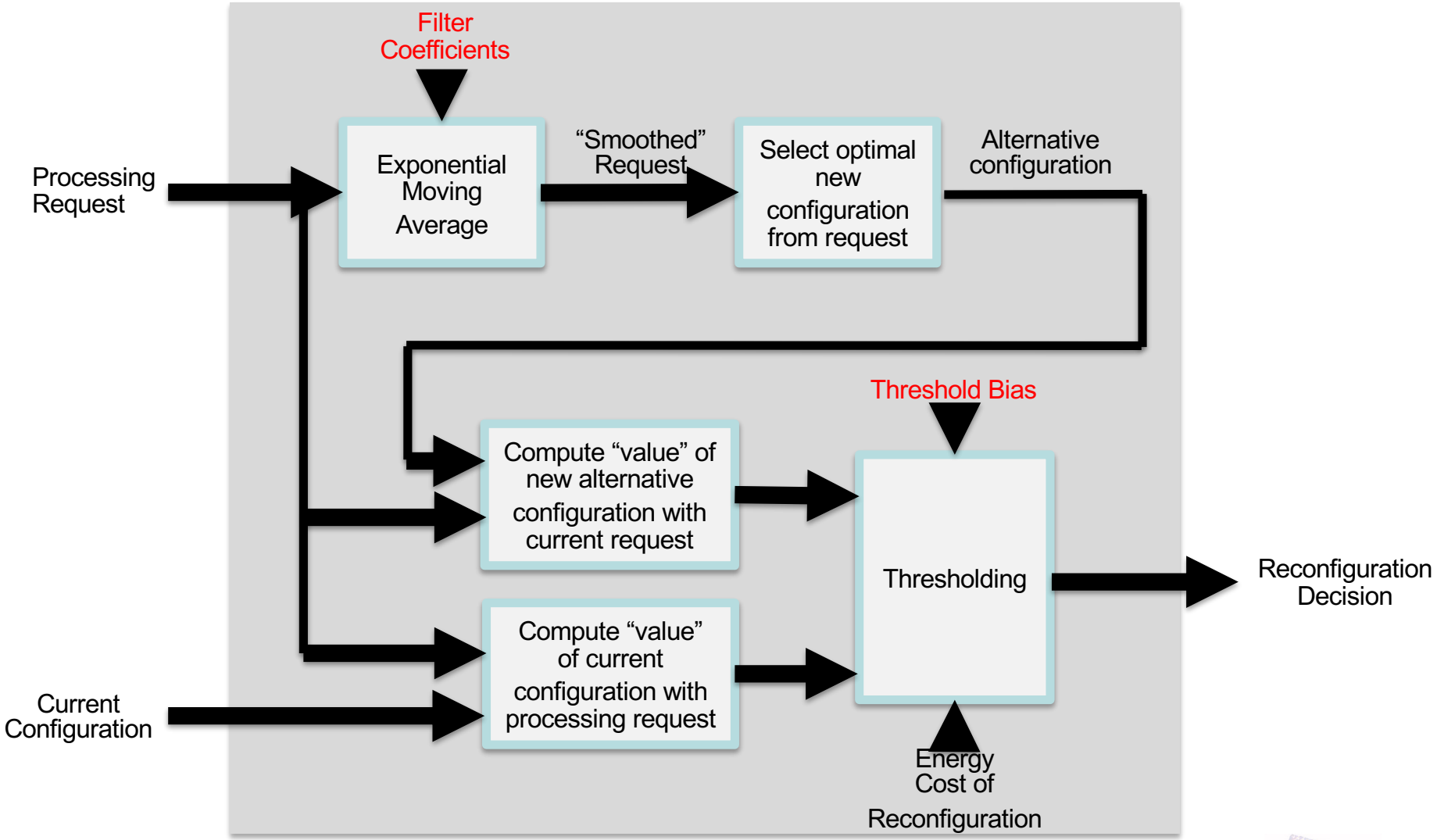
Example:
Transition time is 4.67 frames
→ c = 1/4

Module Outline: Factored MDPs and Application Examples

- Factored Markov Decision Processes (MDPs)
- MDP-based System Design Example: Reconfigurable Digital Channelizer Design
- Modeling Reconfiguration Delays
- Comparison with Manually Generated Policies
- Comparison with Prior Work On Adaptive Reconfiguration
- Summary

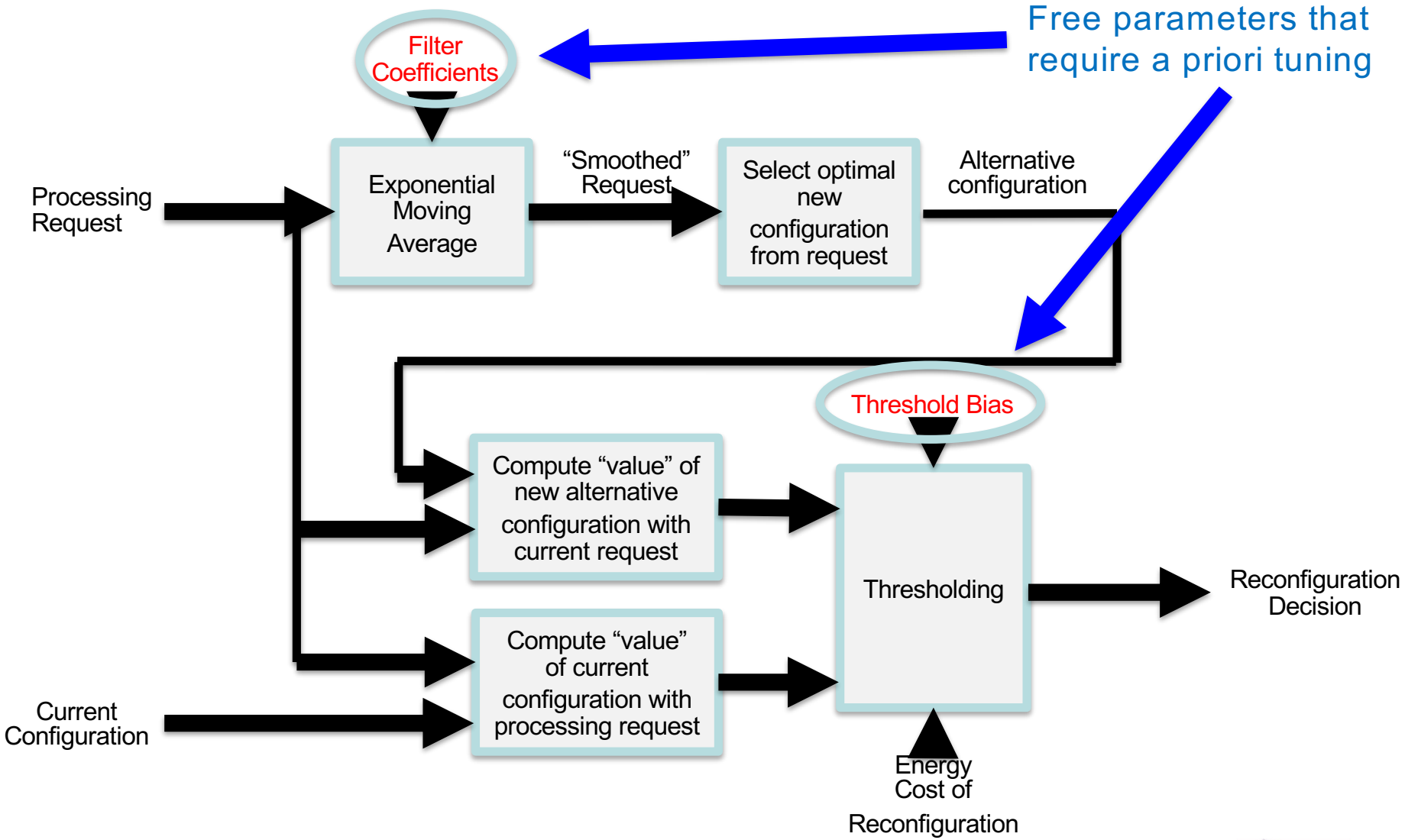
Representative Alternative: [Hsieh'14] (1)

Highly Adaptive Reconfiguration Platform (HARP)



Representative Alternative: [Hsieh'14] (2)

Highly Adaptive Reconfiguration Platform (HARP)



Key Differences

[Hsieh'14]:

- Time average filtering and thresholding used to compute reconfiguration decisions.
- Only metric used in "value" is the energy consumption of the wireless sensor network platform.

MDP-Based Control:

- Decision making under uncertainty — models external environment as a random process.
- Enables *prediction* of future requests using statistics of process.
- Multi-objective optimization (e.g., energy, accuracy, latency, …)
- No tuning parameters — allows for more robustness in dynamic adaptation to changing external environments.

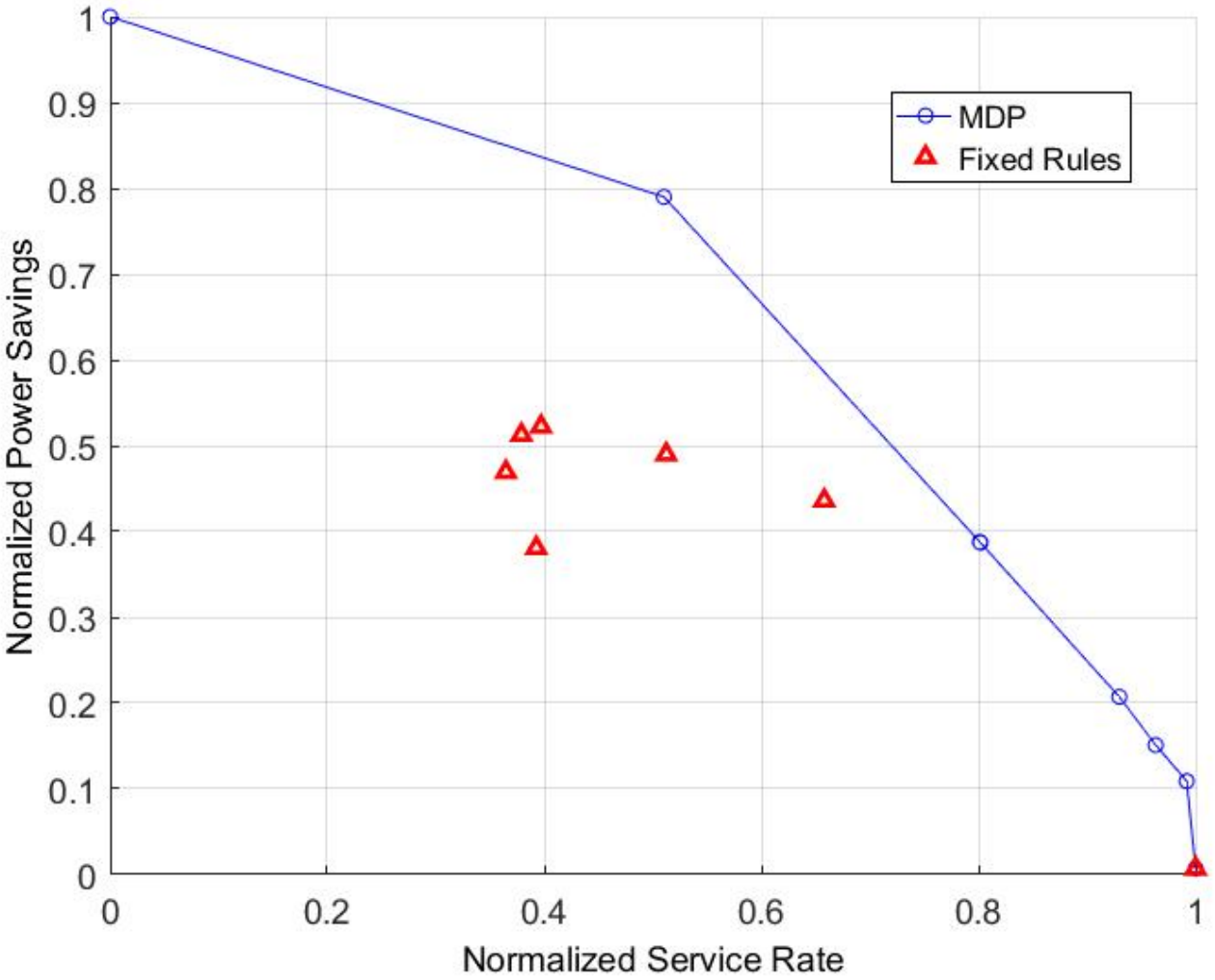
Fixed Reconfiguration Rules (1)

- DFTFB — This policy keeps the DFTFB algorithm on the chip at all times, and invokes it in all frames regardless of the external requests.
- DFTFB+Sleep—This policy also keeps the DFTFB algorithm on the chip at all times. However, if the number of requested channels is 0, the DFTFB is put into sleep mode. Otherwise, the DFTFB is kept on.
- DCM+Sleep — This policy keeps the DCM algorithm on the chip at all times. If the number of requested channels is 0, the DCM is put into sleep mode. Otherwise, the DCM is kept on and applied to produce one of the requested channels.
- DFTFB+DCM+Sleep — This is a set of policies that use both the DFTFB and DCM algorithms.

Fixed Reconfiguration Rules (2)

- DFTFB+DCM+Sleep — This is a set of policies that use both the DFTFB and DCM algorithms.
  - The reconfiguration decision occurs based on how many channels are requested in the upcoming frame.
  - If less than DFT THRESH channels are requested, the DCM algorithm is used.
  - If more than this threshold are requested, the DFTFB algorithm is used.
  - Additionally, if the number of requested channels is 0, the algorithm that is currently is loaded is put into sleep mode.
  - If a reconfiguration is in progress, it is allowed to finish regardless of incoming requests.
  - The DFT THRESH parameter is varied from 2 to 6, resulting in 5 different control policies.

Results 1: MDP vs. Fixed Rules

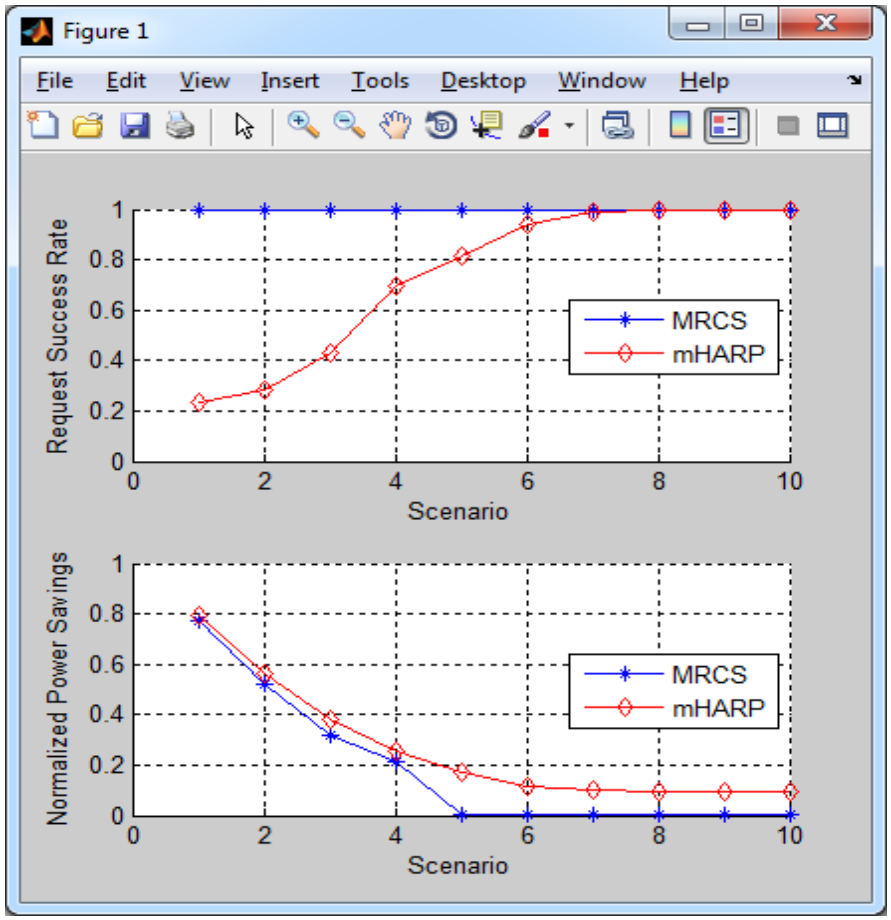


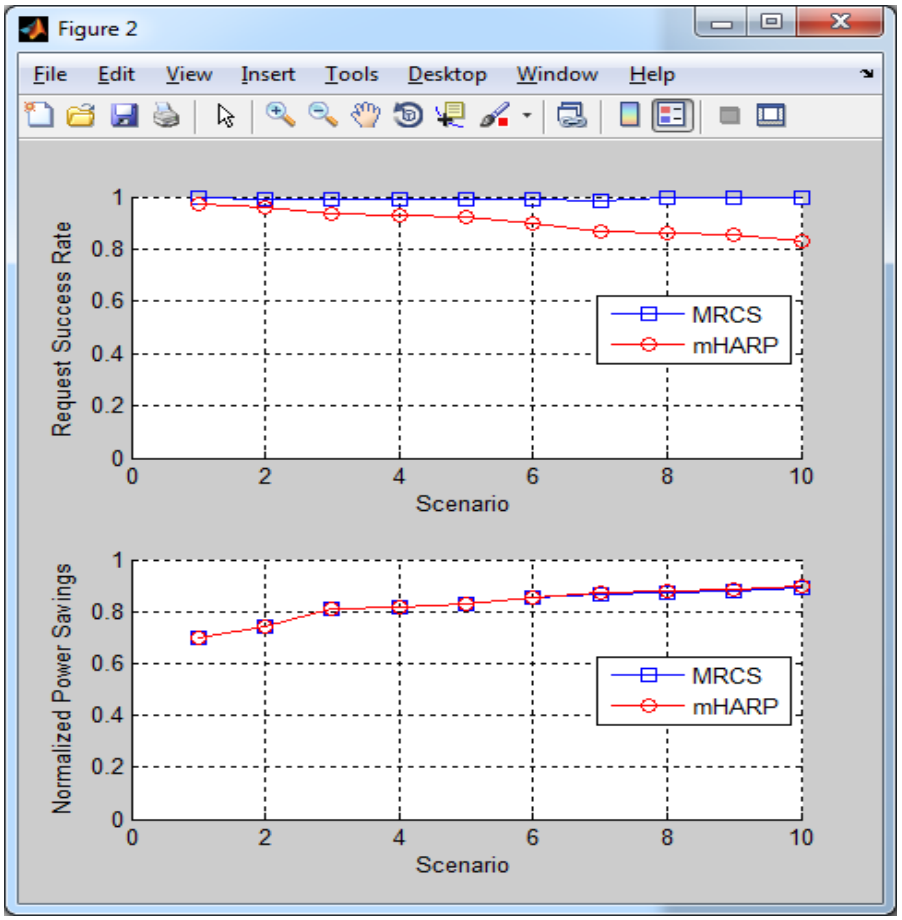
Input characteristics: The request stream switches randomly between the two use cases.

Results 2: MDP vs. mHARP [Hsieh '14]

Application 1:
Dynamic Spectrum Access
[Lee '14]

Application 2:
Sequential Sensing
[Xu '11]





Module Outline: Factored MDPs and Application Examples

- Factored Markov Decision Processes (MDPs)
- MDP-based System Design Example: Reconfigurable Digital Channelizer Design
- Modeling Reconfiguration Delays
- Comparison with Manually Generated Policies
- Comparison with Prior Work On Adaptive Reconfiguration
- Summary

Summary

- In this module of the Tutorial, we have covered Factored MDPs and Application Examples.
- Factorization of state transition matrices (STMs).
- Reconfigurable channelizer example.
- Modeling reconfiguration delays.
  - Transition states.
- Experimental comparison with alternative reconfiguration strategies.
  - Dynamic spectrum access.
  - Sequential sensing.

References (1)

[Benini '99] L. Benini, A. Bogliolo, A. Paleologo, and G. De Micheli, "Policy optimization for dynamic power management," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol. 18, no. 6, pp. 813–833, June 1999.

[Boutilier '95] C. Boutilier, R. Dearden, and M. Goldszmidt. Exploiting structure in policy construction. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1104-1111, 1995.

[Gue '03] C. Geustrin D. Koller, R. Parr, et. Al. "Efficient Solution Algorithms for Factored MDPs", *Journal of Artificial Intelligence Research*, vol. 19, p. 399–468, 2003.

[Hsieh '14] C.-M. Hsieh, F. Samie, M. S. Srouji, M. Wang, Z. Wang, and J. Henkel. Hardware/software co-design for a wireless sensor network platform. In *Proceedings of the International Conference on Hardware/Software Codesign and System Synthesis*, pages 1-10, 2014.

[Lee '14] C.-S. Lee, W.-C. Chen, S. S. Bhattacharyya, and T.-S. Lee. Dynamic, data-driven spectrum management in cognitive small cell networks. In *Proceedings of the International Conference on Signal Processing and Communication Systems*, pages 1-5, December 2014.

References (2)

[Sapio '18] A. Sapio, L. Li, J. Wu, M. Wolf, and S. S. Bhattacharyya. Reconfigurable digital channelizer design using factored Markov decision processes. *Journal of Signal Processing Systems,* 2018. To appear; preprint available at url http://arxiv.org/abs/1712.08340.

[Sapio '16] A. Sapio, M. Wolf, and S. S. Bhattacharyya. Compact modeling and management of reconfiguration in digital channelizer implementation. In *Proceedings of the IEEE Global Conference on Signal and Information Processing*, pages 595-599, December 2016.

[Szita '08] I. Szita, A. Lorincz, "Factored value iteration converges", *Acta Cybernetica*, vol. 18, num. 4, p. 615–635, 2008.

[Xu '11] M. Xu, H. Li, and X. Gan. Energy efficient sequential sensing for wideband multi-channel cognitive network. In *IEEE International Conference on Communications*, pages 1-5, 2011.