Multimedia Communications and Systems Laboratory

http://medianetlab.ee.ucla.edu

Towards a Systematic Approach for Modeling and Optimizing Distributed and Dynamic Multimedia Systems

Presenter: Brian Foo

Advisor: Mihaela van der Schaar

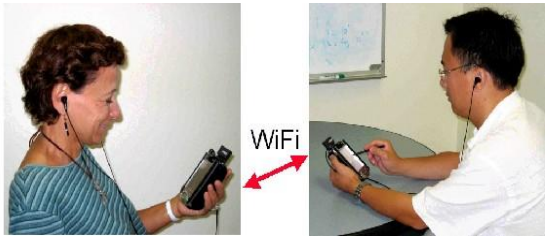
UCLA

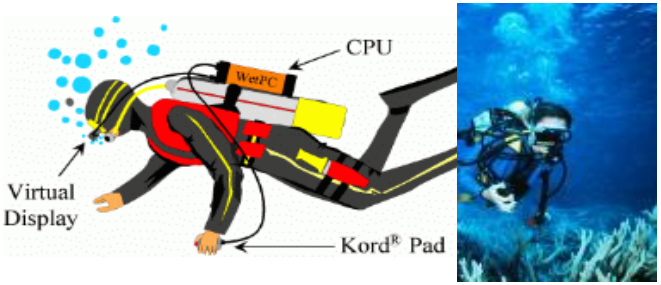
Proliferation of multimedia applications

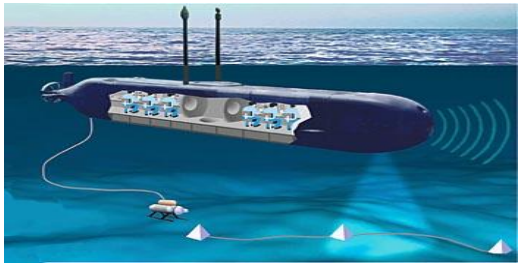
Video compression
Postprocessing

Multimedia stream mining
Image/video retrieval

Online gaming
Virtual reality and 3D



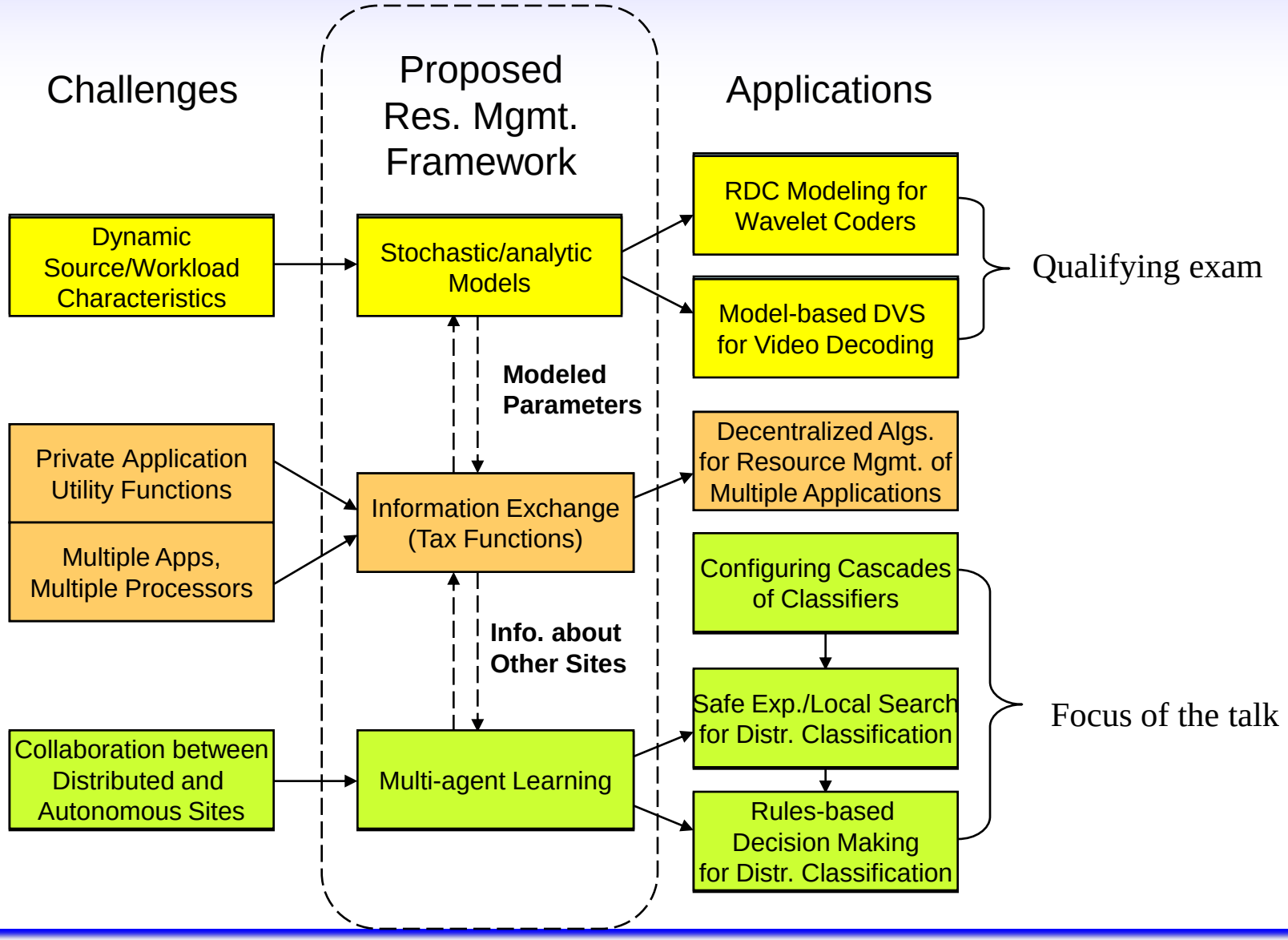




Challenges for designing and optimizing multimedia systems

- Multimedia data and applications are highly dynamic!
  - Real-time system resource adaptation required
- Support for multiple concurrent applications
  - Dividing resources efficiently and fairly among applications
  - Regard for applications' autonomy
- Distributed computing resources
  - Collaboration required to jointly process application
  - Information-decentralization (delay, high communications cost, proprietary or legal restrictions)

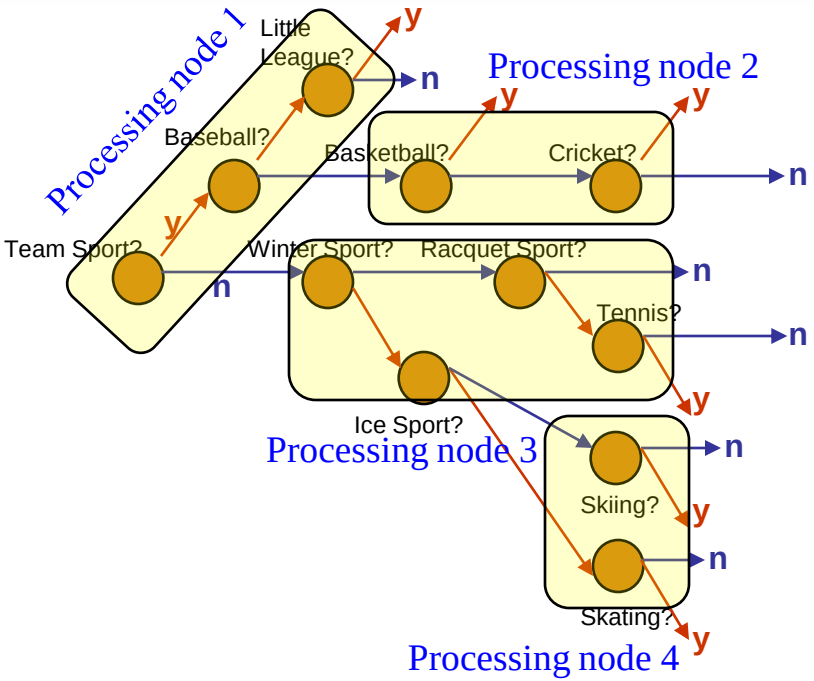
Overview of Thesis Topics



Outline of Presentation

- New area emerging: Resource-constrained stream mining
  - Static stream, same site
  - Static stream, different autonomous sites
  - Dynamic stream, different autonomous sites
- Decentralized Resource Allocation for Multiple Multimedia Tasks
  - Tax functions
- Modeling multimedia data and application dynamics
  - Applications to Dynamic Voltage Scaling
- Conclusions and future directions

Cascaded Topologies of Classifiers on Distributed Stream Mining Systems: Same Site



Borealis, Aurora, TelegraphCQ

In cooperation with Marvel and the System S Stream Processing Core group at IBM T. J. Watson, Hawthorne, NY

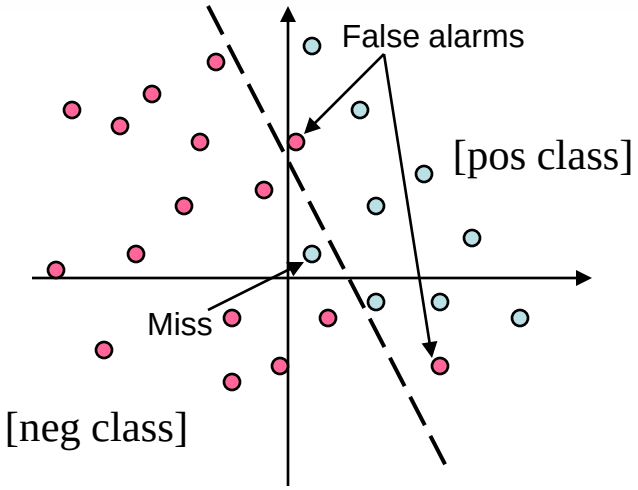
[Foo, Turaga, Verscheure, vdSchaar, Amini, Signal Processing Letters, 2008.]

- Complex classifiers can be decomposed into cascaded topologies of binary classifiers [Schapire, 1999]
- Application operators can be instantiated on distributed processing devices with individual resource constraints.
- Issues: placement, fault tolerance, load shedding, etc.

Prior Approaches to Load Shedding for Stream Mining Systems

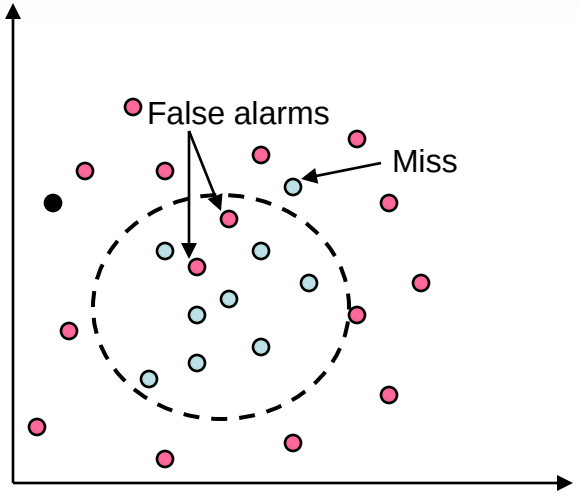
- Probabilistic load shedding
  - Reduce network delay [Tatbul 2002]
  - Reduce memory consumption [Babcock 2003]
- Quality-aware load shedding for data mining
  - Windowed load shedding for aggregation queries [Tatbul 2004, 2006]
- Load shedding for classification?
  - Very little work in this area! [Muntz 2005] – Single classifier
- Limitations
  - Suboptimal classification performance/application utility!
- Our approach
  - First to formalize load shedding as an application optimization problem: maximize joint classification quality subject to resource constraints, delay, and dynamics

Configuring Classifier Operating Points



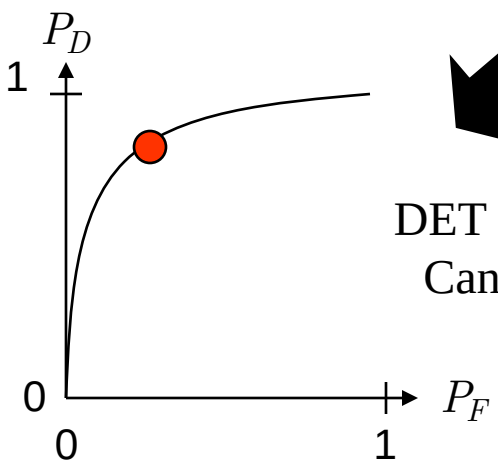
SVM: Linear Kernel Function

$$k_{\text{th}} \leq k = \mathbf{u}^T \mathbf{v}$$



SVM: Radial Basis Kernel Function

$$k_{\text{th}} \leq k = e^{-\gamma |\mathbf{u}-\mathbf{v}|^2}$$

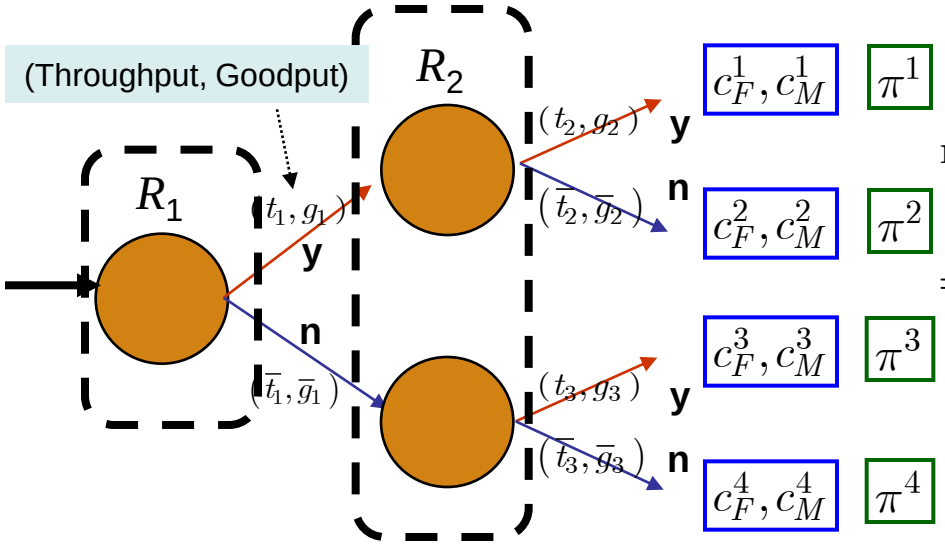


DET curve relates misses and false alarms.
Can parameterize operating point by pf.

Affects throughout (output rate)
and goodput (output rate of detected data)

Problem Formulation

- Given
  - Costs of misclassification (c_M, c_F) per data object per class
  - True volume of data in each class π^k
  - Placement and Resource constraints
  - Throughput and Goodput
- Objective
  - Minimize end-to-end misclassification cost
  - Satisfy resource constraints



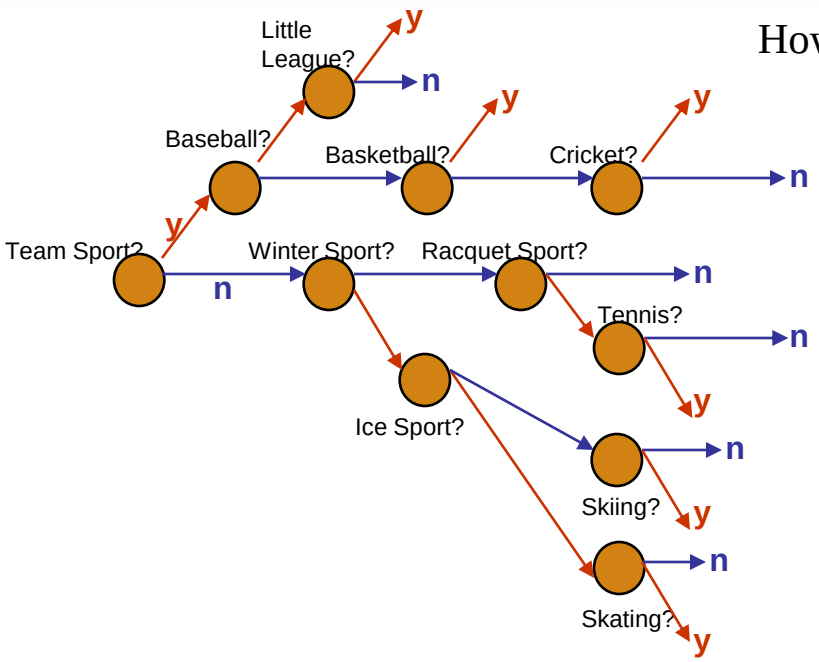
$$\text{minimize } c = \sum_{k=1}^{K} \left[c_F^k \cdot \text{false\_alarms} + c_M^k \cdot \text{misses} \right]$$

$$= \sum_{k=1}^{K} \left[c_F^k \cdot \left(t^k(\mathbf{p}_F) - g^k(\mathbf{p}_F) \right) + c_M^k \cdot \left(\pi^k - g^k(\mathbf{p}_F) \right) \right]$$

$$\text{s.t. } \mathbf{A}\mathbf{h}(\mathbf{p}_F) \le \mathbf{R}$$

$$0 \le \mathbf{p}_F \le 1$$

Computing Goodput and Throughput for a Semantic Tree Topology

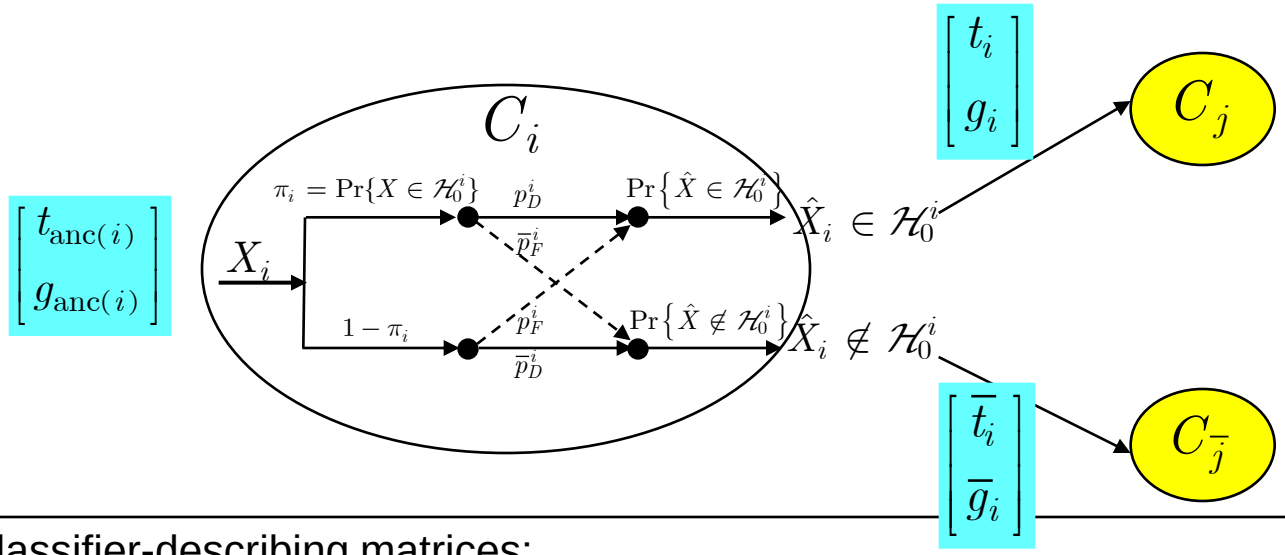


How can goodput/throughput be computed?

When each classifier in a branch filters a subclass, this property is referred to as *exclusivity*.

Goodput and throughput for each class can be computed recursively.

Calculation of throughput and goodput



Classifier-describing matrices:

$$T_i^k = \begin{bmatrix} p_F^i & \pi_i \left(p_D^i - p_F^i \right) \\ 0 & \pi_i p_D^i \end{bmatrix} \quad \text{or} \quad \overline{T}_i^k = \begin{bmatrix} \overline{p}_D^i & \pi_i \left(\overline{p}_F^i - \overline{p}_D^i \right) \\ 0 & \overline{\pi}_i \overline{p}_D^i \end{bmatrix}$$

Throughput and goodput out of classifier Ci

$$\begin{bmatrix} t_i \\ g_i \end{bmatrix} = \mathbf{T}_i^k \begin{bmatrix} t_{\mathrm{anc}(i)} \\ g_{\mathrm{anc}(i)} \end{bmatrix} = \ldots = \mathbf{T}_i^k \mathbf{T}_{\mathrm{anc}(i)}^k \ldots \mathbf{T}_1^k \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

Calculation of resource constraints

- Resource h_i consumed by classifier C_i is proportional to the input rate $t_{\mathrm{anc}(i)}$, i.e.

$$h_i = \alpha_i t_{\mathrm{anc}(i)}$$

  - Coefficient α_i is the processing complexity per data object

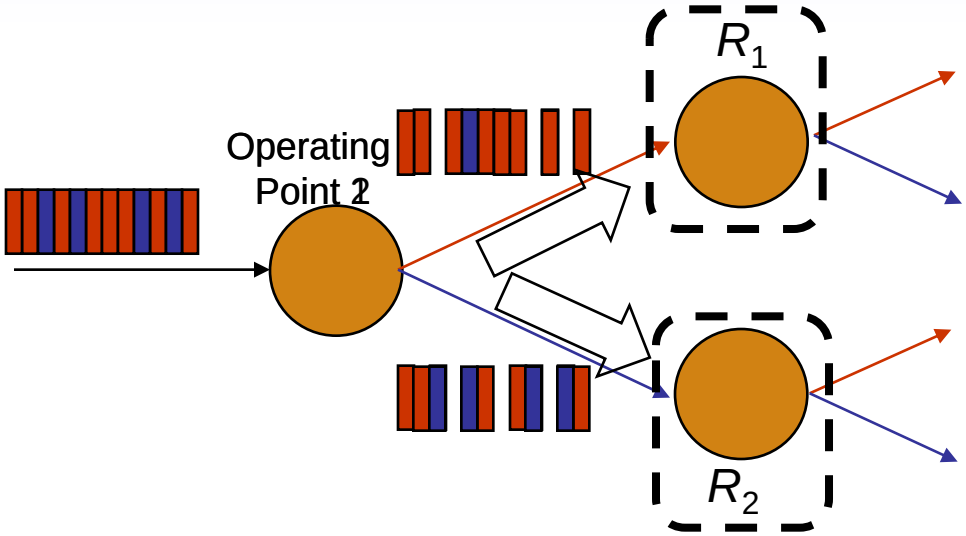
- Placement: described by matrix A, where:

$$A_{mi} = 1 \text{ if } C_i \in \mathrm{node}(m)$$

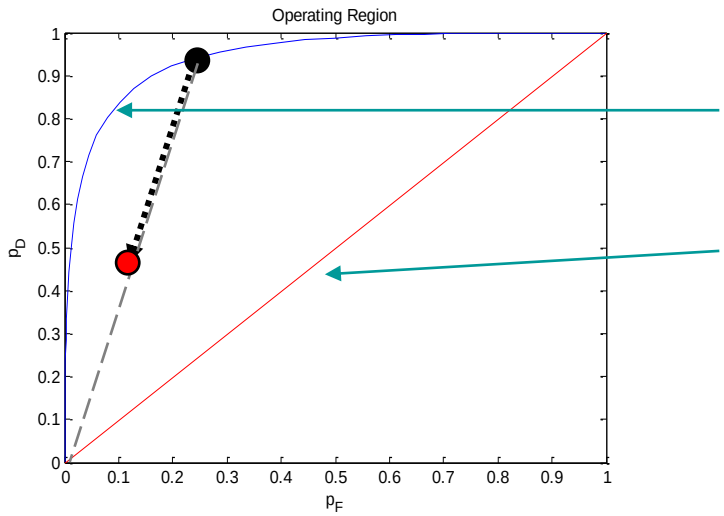
$$A_{mi} = 0 \text{ otherwise}$$

- Node resource availability: $\mathbf{R} = (R_1, \ldots, R_M)^T$
- Resource constraint inequality: $\mathbf{Ah} \leq \mathbf{R}$

Prior Approaches to Load Shedding for Stream Mining Systems



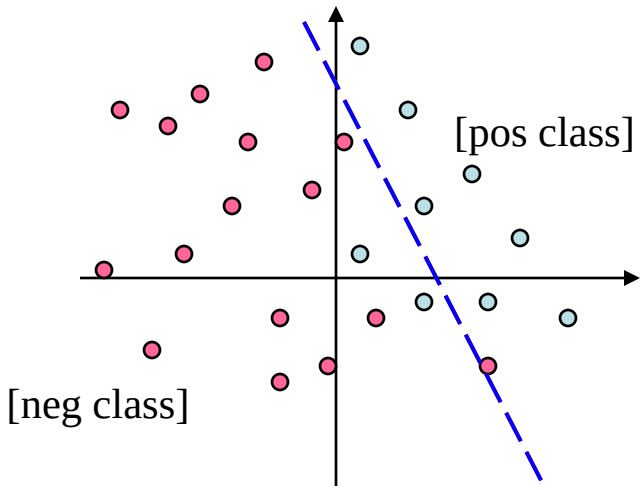
To reduce delay when not be feasible to meet tight resource constraints ⇒ **Arbitrary Load Shedding at next classifier** [Babcock, 2003; Tatbul, Zdonik, 2006]



DET curve

Random data forwarding curve

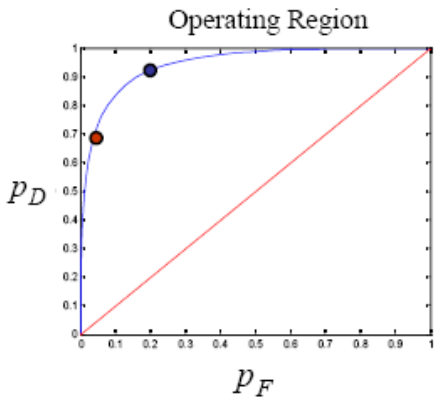
A Proposed Approach: Multiple Operating Points



Positive threshold
Negative threshold

Shedding of low confidence data at current classifier ⇒ **Intelligent Load Shedding**

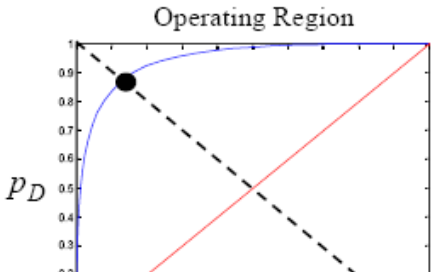
Also support **Replication** of low confidence data when resources are available **(significant difference from current literature)**



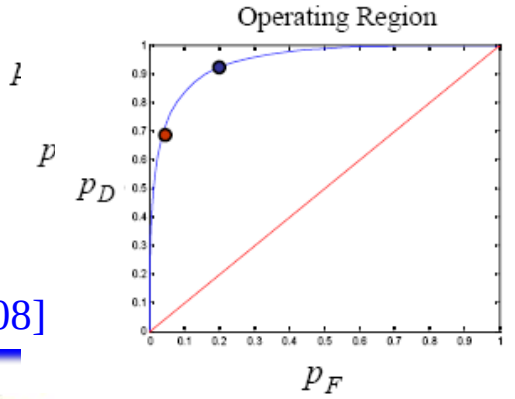
Centralized Solutions

- Use Sequential Quadratic Programming (SQP).
  - Running SQP several times with different starting points gives higher probability of finding global optimum.

- Considered Algorithms
  - A) Equal Error Rate (EER) configuration (e.g. [Muntz, 2005])
  - B) Single operating point, no resource constraint consideration.
    - Let system take care of load shedding
    - (e.g. [Schapire, 1999] + [Babcock, 2003; Zdonik, 20
  - C) Single operating point, jointly optimized by shedding [Foo, Turaga, Verscheure, vdSchaar, Amini, SPL, 2008.]
    - Algorithm considers resource constraints down
      point and load shedding jointly.



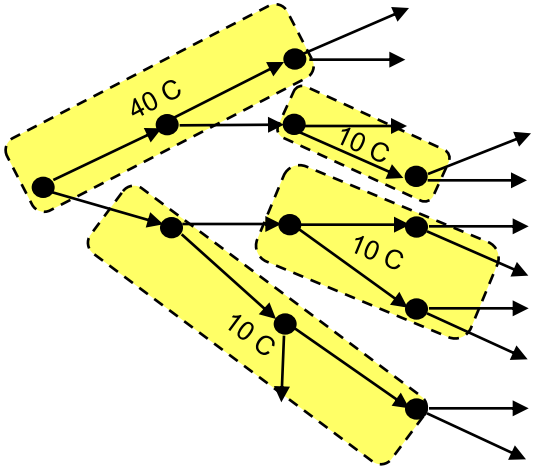
  - D) Proposed: Multiple operating points! [Foo, Turaga, Verscheure, vdSchaar, Amini, SPL, 2(
    - Use a separate threshold for yes and no output e
    - Intelligent load shedding and replication of data



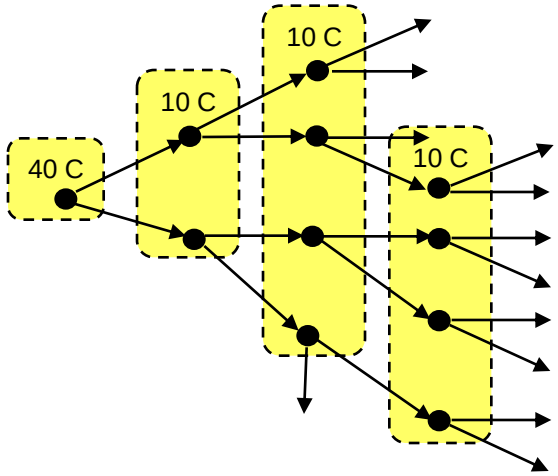
Distributed algorithm?
[Foo, Turaga, Verscheure, vdSchaar, Amini, TCSVT, submitted 2008]

Experimental Results with Sports Image Classification

Placement 1-along each branch
(cross-talk minimizing)



Placement 2-across different branches
(failure-resilient)



Resulting Costs per data object for Cf = Cm = 1

Algorithm	No Resource Constraints	Cross-talk Minimizing Placement	Failure-resilient Placement
EER	1.9563	1.2971	1.3604
LS at Input	0.7742	0.9226	0.9442
LS at Output	0.7907	0.9158	0.8964
Mult. Op. Pts.	0.6959	0.8640	0.8419

Experimental Results with Sports Image Classification

Resulting costs for different cost functions

Algorithm	High Cost of False Alarms (4x)	High Cost of Misses (4x)
EER	3.8906	3.8906
LS at Input	1.9356	1.9355
LS at Output	0.9655	1.9365
Mult. Op. Pts.	0.8703	1.5438

Load shedding: When error rate is too high, the best solution is to prevent false alarms by shedding the entire output load.

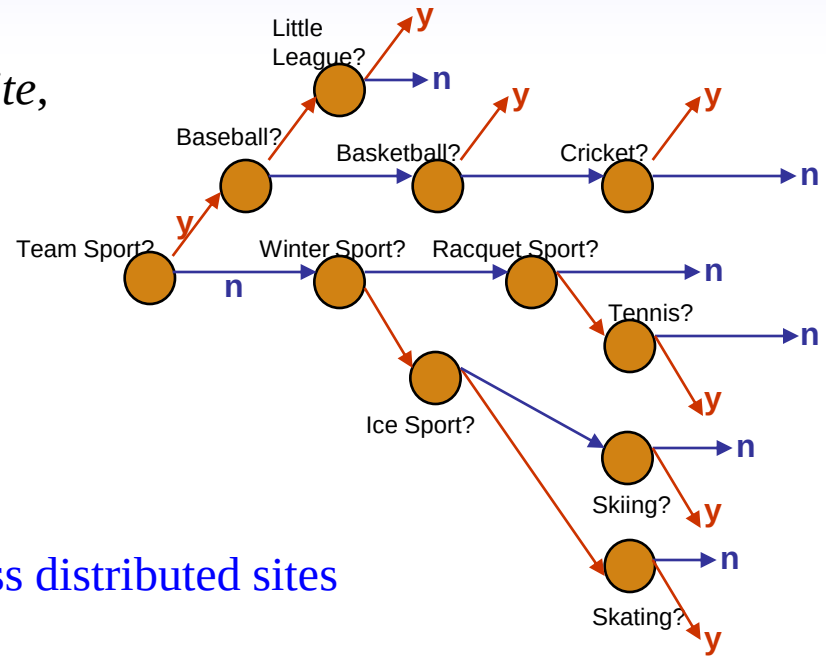
Replication: When cost of misses are high, it is better to replicate such that the probability of missing data in each class is minimized.

Outline of Presentation

- New area emerging: Resource-constrained stream mining
  - Static stream, same site
  - Static stream, different autonomous sites
  - Dynamic stream, different autonomous sites
- Decentralized Resource Allocation for Multiple Multimedia Tasks
  - Tax functions
- Modeling multimedia data and application dynamics
  - Applications to Dynamic Voltage Scaling
- Conclusions and future directions

Challenge of Distributed Analytics

When the classifiers are *jointly trained at one site*, features such as *exclusivity can be guaranteed*



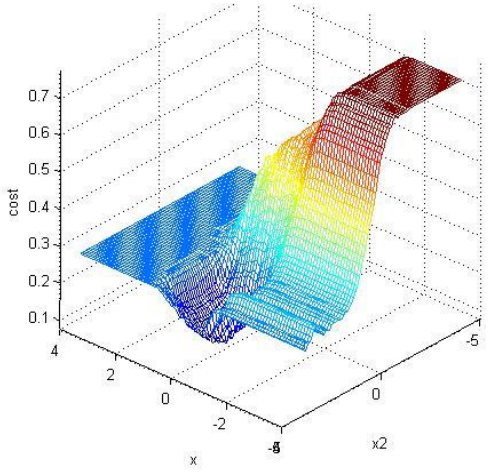
But what about semantic classifiers trained across distributed sites that don't obey simple relationships?

(e.g. distributed data sets –
classifiers for "basketball", "outdoor", and "children" images)

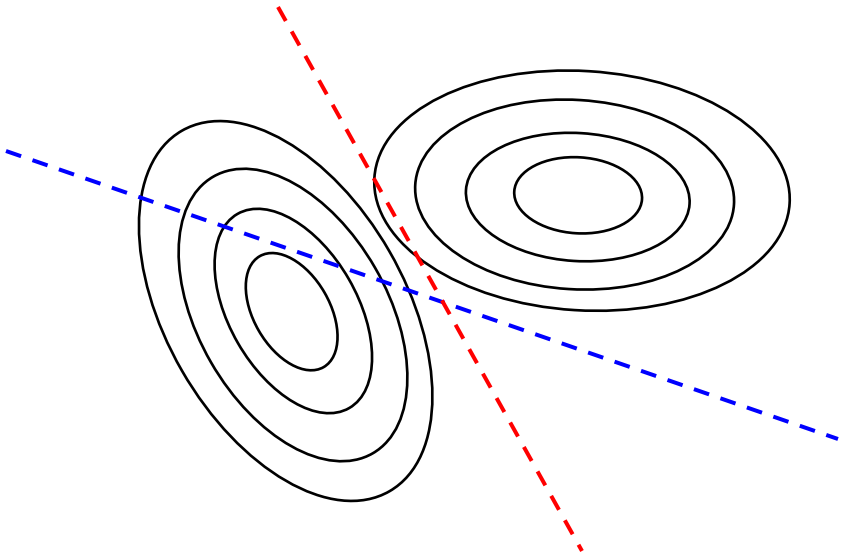
Limitations of Analytical Joint Classifier Configuration

Problem: Unless the joint probability distribution is known, it is impossible to determine the joint performance.

Correlations in the classifier functions on the filtered data must be known to determine π_i, which affects both performance and delay!



Cost (e.g. 1 - quality) for joint thresholding of two classifiers in speaker detection.



If analytics are not shared between distributed nodes, we cannot determine end-to-end cost!

Related Works in Distributed Classification

- P. Varshney, *Distributed Detection and Data Fusion*, Springer, 1997, ISBN: 978-0-387-94712-9.
- J. Vaidya, C. Clifton, "Privacy-preserving k-means clustering over vertically partitioned data," *ACM SIGKDD*, 2003.
- S. Merugu, J. Ghosh, "Privacy-preserving distributed clustering using generative models," *ICDM*, 2003.

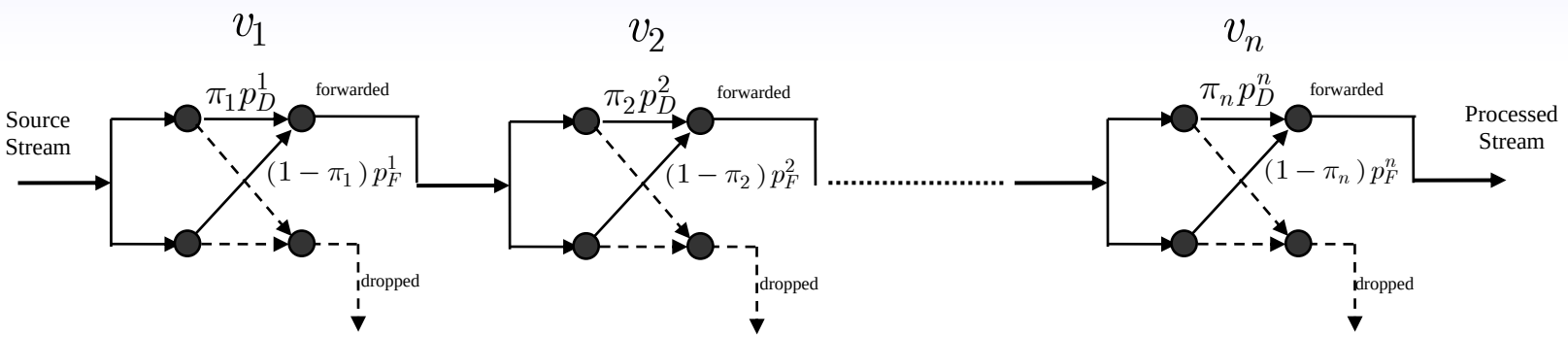
Limitations

- Constructing centralized classification models requires high complexity and communications overhead on an already overloaded system!
- Also requires systems to share information about datasets/analytics.
  - Not possible if datasets have proprietary/legal restrictions.

Proposed solution for estimating classification utility:
[Foo, vdSchaar, SPIE 2008]

Generate a *model* delay-sensitive stream processing utility,
and estimating with a low-overhead *information exchange mechanism*.

Modeling Classifier Chain Performance and Delay



π_i (conditional a priori probability of positive data)

p_D^i (detection probability)

p_F^i (false alarm probability)

$t_i = \pi_i p_D^i + (1-\pi_i)\,p_F^i$ (ratio of forwarded data from classifier i)

$g_i = \pi_i p_D^i$ (ratio of correctly forwarded data from classifier i)

$D_i \sim (\mu_i - \lambda_i)\,e^{-(\mu_i-\lambda_i)t}$ (delay at classifier i using M/M/1 model)

Average service rate μ_i is fixed, but arrival rate depends on ℓ_j for all upstream classifiers v_j, i.e. $\lambda_i = \lambda_{i-1} t_{i-1} = \ldots = \lambda_0 \prod_{j=1}^{i-1} t_j$.

Stream Processing Utility Model

Estimated classification quality for v_i

$$F_i = g_i - \theta_i (t_i - g_i)$$
$$= \pi_i p_D^i - \theta_i (1 - \pi_i) p_F^i$$

θ_i denotes the relative importance of false alarms to misses.

Delay penalty function [Horvitz, 1991] :

$$G(D_i) = E\left[e^{-\varphi D_i}\right]$$

(φ denotes the delay-sensitivity of stream mining application.)

End-to-end Quality of classifier chain:

$$F = \prod_{i=1}^{n} F_i = \prod_{i=1}^{n} (g_i - \theta_i (t_i - g_i))$$

End-to-end Delay penalty based on M/M/1:

$$G(D) = E\left[e^{-\varphi D}\right] = \prod_{i=1}^{n} \left[\frac{\mu_i - \lambda_i}{\mu_i - \lambda_i + \varphi}\right]^+$$

End-to-end Delay-sensitive Stream Processing Utility:

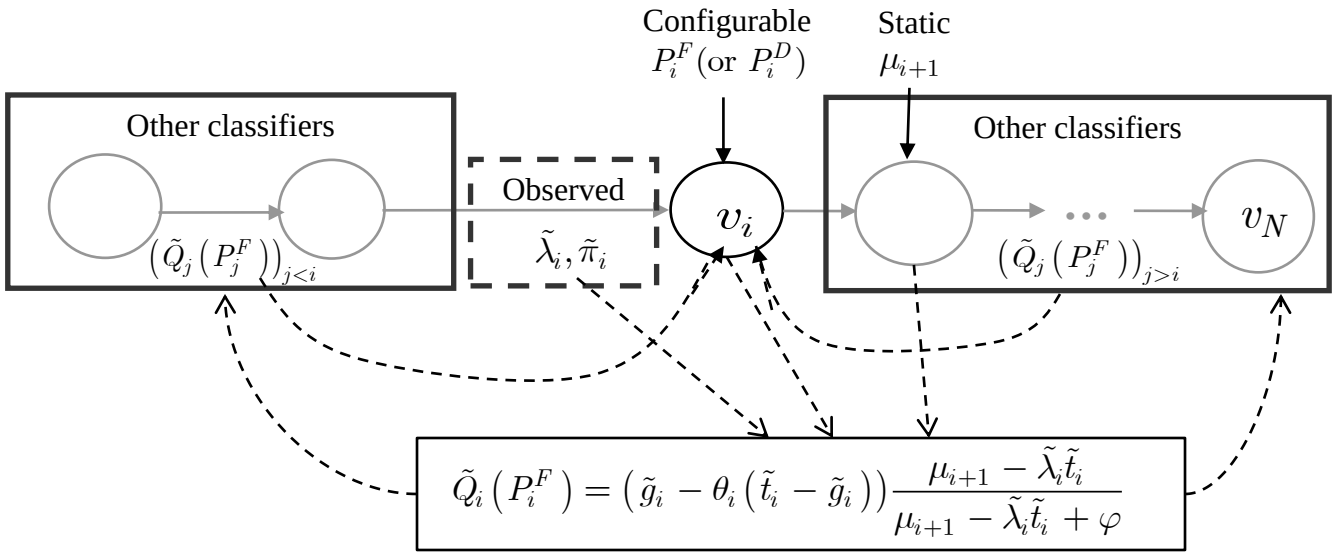
$$Q\left(\mathbf{P}^F\right) = F \cdot \Phi_D(-\varphi) = \prod_{i=1}^{n} [g_i - \theta_i (t_i - g_i)]^+ \left[\frac{\mu_i - \lambda_i}{\mu_i - \lambda_i + \varphi}\right]^+$$

Distributed Information Exchange Mechanism

Decomposition of Utility function:

$$Q(\mathbf{P}^F) = \prod_{i=1}^{n}(g_i - \theta_i(t_i - g_i))\prod_{i=1}^{n}\left(\frac{\mu_i - \lambda_i}{\mu_i - \lambda_i + \varphi}\right)$$

$$\approx \underbrace{\left(\frac{\mu_1 - \lambda_1}{\mu_1 - \lambda_1 + \varphi}\right)}_{\text{constant}}\prod_{i=1}^{n-1}\underbrace{(\tilde{g}_i - \theta_i(\tilde{t}_i - \tilde{g}_i))\left(\frac{\mu_{i+1} - \tilde{\lambda}_i\tilde{t}_i}{\mu_{i+1} - \tilde{\lambda}_i\tilde{t}_i + \varphi}\right)}_{\text{known at } v_i}\underbrace{(\tilde{g}_n - \theta_n(\tilde{t}_n - \tilde{g}_n))}_{\text{known at } v_n}$$



Each classifier can obtain the global utility estimate $\tilde{Q}(\mathbf{P}^F)$ after info exchange.

Autonomous sites, stream dynamics, private parameters → multi-agent solutions.

A Multi-agent Learning algorithm, and Our Proposed Solution

Safe experimentation [Marden, Shamma, 2007]: Experiment with different *discrete* action space.

Limitations: 1) Long convergence time for large action spaces!

2) Classifiers can adjust thresholds *continuously* (infinite actions)!

Proposed Safe-experimentation and local search algorithm:

[Foo, vdSchaar, SPIE 2008]

1) Randomly select initial configuration: $P_i^F(0)$

2) Set baseline utility after information exchange: $u^b(1) = Q\left(\mathbf{P}^F(0)\right)$

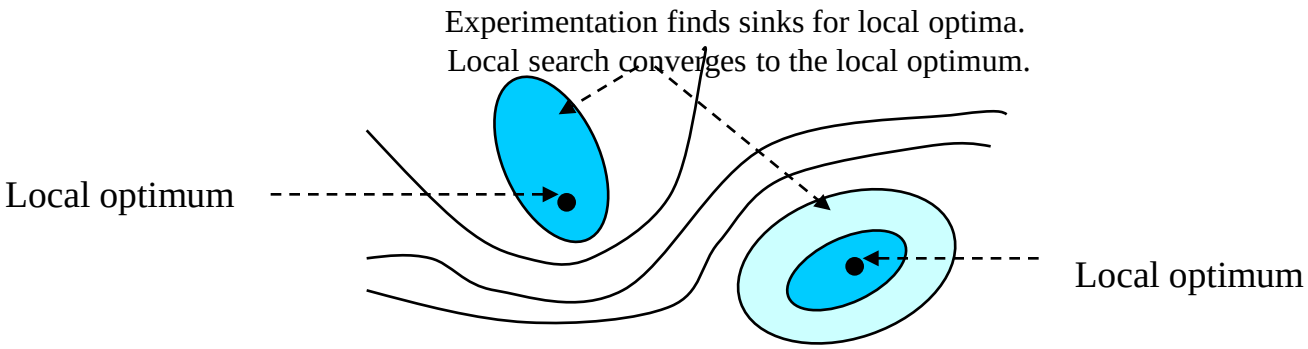
3) At each time of reconfiguration,
choose a new random action with probability ε_t,
or perturb the original action with probability $1-\varepsilon_t$,
using a random variable $Z_i(t)$ with $\lim_{t\to\infty} Z_i(t) = 0$

Update the baseline action and utility to the maximum utility observed.

4) Go to step 2 and repeat.

Optimal convergence of the proposed algorithm for static streams

Main result: the proposed safe experimentation and local search algorithm converges to the globally optimal solution with probability 1 subject to appropriate exploration rates.



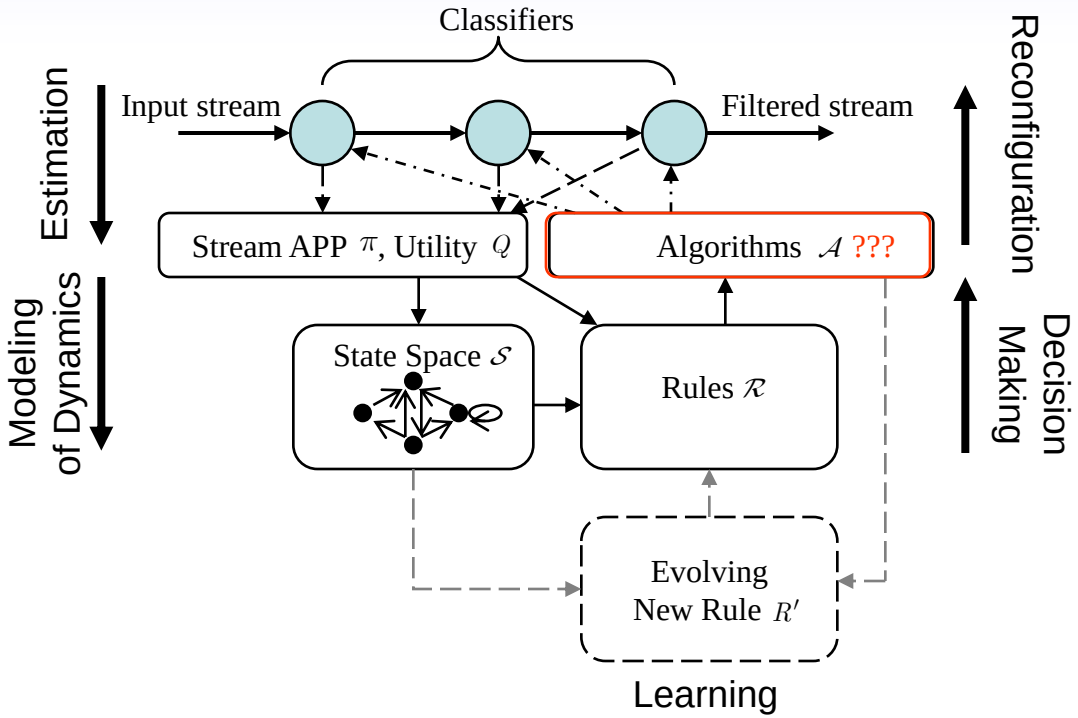
Limitation of Experimentation for *Dynamic* Streams

- Safe experimentation/local search works well in static environments, but does poorly in dynamic environments
  - Requires frequent random experimentation to rediscover high utility configs.

- Confusion matrix and delay for a medium loaded system, where APPs change by 3-20% for each speaker class during each iteration:

	Labeled Spkr of Interest	*Other Speakers*
True Spkr of Interest	4.21	13.54
Other Speakers	11.06	153.66

Proposed approach for reconfiguring distributed chains of classifiers in dynamic environments



Learning solutions required to obtain to optimal joint configuration in both static and dynamic environments!

Proposed Rules-based Approach for Choosing Reconfiguration Algorithms

- *States* $\mathcal{S} = \{S_1, \ldots, S_M\}$ based on quantized system metrics (e.g. global utility, APP, local utilities)

- *Multiple algorithms* $\mathcal{A} = \{A_1, \ldots, A_K\}$ that can be chosen for reconfiguring classifiers
  - e.g. safe experimentation and local search can be split into two different algorithms

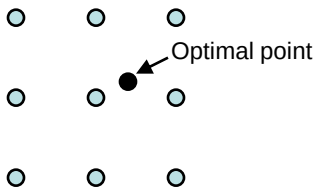
- Model state transitions as a function of the previous state and algorithm (i.e. $s_{t+1} \sim p(s \mid s_t, a_t)$)

- *Rules* $\mathcal{R} = \{R_1, \ldots, R_H\}$ that map each state to an algorithm.
  - Similar to policies in Markov Decision Processes (MDP)
    See: M. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, John Wiley & Sons, 1994.
  - But there is a key difference!

Difference between Algorithms and Actions

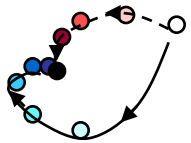
$$\mathbf{P}_t^F = A_k\left(\mathbf{P}_{t-1}^F,\ldots,\mathbf{P}_{t-\tau}^F\right)$$

Actions $A_k\left(\mathbf{P}_{t-1}^F,\ldots,\mathbf{P}_{t-\tau}^F\right) = \mathbf{c}_k$



Quantized configurations, cannot approach the optimal point.

Algorithms



Optimized based on analytical modeling and previous configurations/results.

The rules-based approach takes advantage of both:
1) MDP for steady-state/best response optimization in dynamic environments, and
2) optimal algorithms for configuration in static (or less dynamic) environments.

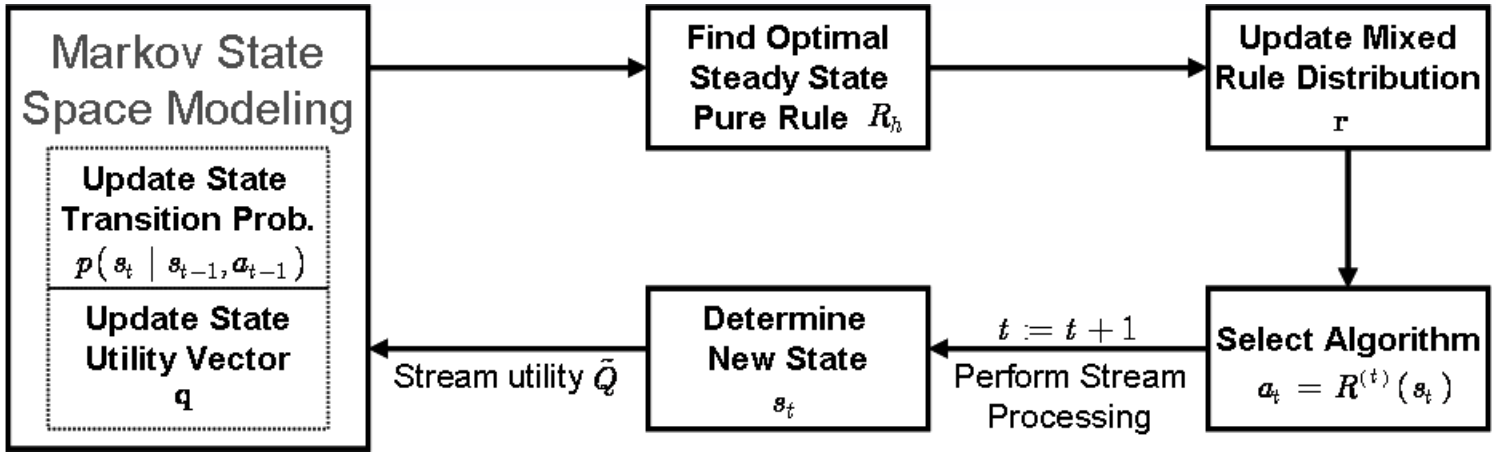
Why not just increase action space? Convergence time increases quadratically.

Learning the optimal rule

- An optimal rule exists in our proposed framework

- How to find the optimal rule when stream dynamics are initially unknown?
  - i) Randomly select all algorithms in all states, and estimate the parameters (utilities and transition probabilities).
    - Poor performance during random experimentation!
    - How long to experiment?
  - ii) Reinforce the estimated optimal rule.
    - Can be highly suboptimal!
  - **iii) Solution that provides both perfect estimation, and converges to the optimal rule.**

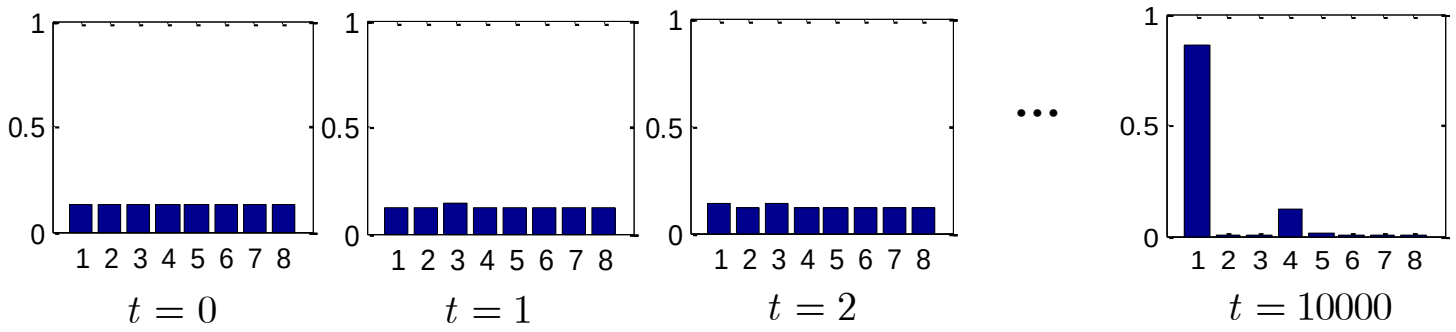
Solution: Learning the Optimal Rule



1) Initialize "tendency" of playing each rule R_h to $c_h = 1$.
2) Play each rule R_h with probability $p_{R_h} = \sqrt[M]{c_h} / \sum_{g=1}^{H} \sqrt[M]{c_g}$
3) Apply the chosen algorithm to reconfigure classifiers.
4) Process the stream, estimate utility $\tilde{Q}$, and determine new state.
5) Update transition probability matrix $p(s_t \mid s_{t-1}, a_{t-1})$ and state utilities $\mathbf{q}(\mathcal{S})$.
6) Find the projected optimal steady state pure rule R^*, and increment its frequency by 1.
7) Return to step 2.

Evolution of Rule Distributions

8 different rules (for the speaker classification application).
Note the convergence toward one optimal rule.



Sometimes a small probability exists for a secondary rule if dynamics aren't completely Markov.

Estimation accuracy and performance bounds

Proposition:

Suppose that $\left|Q(S_m) - \hat{Q}(S_m)\right| \le \sigma$ and $\left|\mathbf{P}_{ij}(R_h) - \tilde{\mathbf{P}}_{ij}(R_h)\right| \le \delta$

Then the steady state utility of the convergent rule deviates
from the utility of the optimal rule by no more than approximately

$$2M\delta(U_Q + 2M\sigma)$$

where U_Q is the average system utility of the highest utility state.

Corollary:

In the worst case, the expected number of iterations required
for the solution to determine a pure rule that has average utility within
$M\delta(U_Q + 2M\sigma)$ of the optimal pure rule with probability at least
$(1-\varepsilon)(1-\eta)$ is $O\left(\max_{m=1,\ldots,M}\left(1/\left(4n\delta^2\right), v_m^2/\left(\varepsilon\sigma^2\right)\right)\right)$, where
v_m^2 is the utility variance within each state S_m, and σ^2 is the average variance
of the utility estimation error.

Distributed approach for learning rules

- Set of rules played can only be given in the form: $\mathcal{R} = \mathcal{R}_1 \times \mathcal{R}_2 \times ... \times \mathcal{R}_n$ where $\mathcal{R}_i$ corresponds to the rule at site

- Each site updates rules based on local states $\mathcal{S}_i \times \mathcal{S}'$ and local algorithms $\mathcal{A}_i$, thus simulating a global state and algorithm space: $\mathcal{S} = \mathcal{S}_1 \times \mathcal{S}_2 \times ... \times \mathcal{S}_n \times \mathcal{S}'$

$$\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2 \times ... \times \mathcal{A}_n$$

- $\mathcal{S}'$ is a shared state space, which captures exchanged information across the sites (i.e. partial information).

- Can prove convergence to a Nash equilibrium (local optimum), but not global optimum.

Evolving a New Rule out of Existing Rules

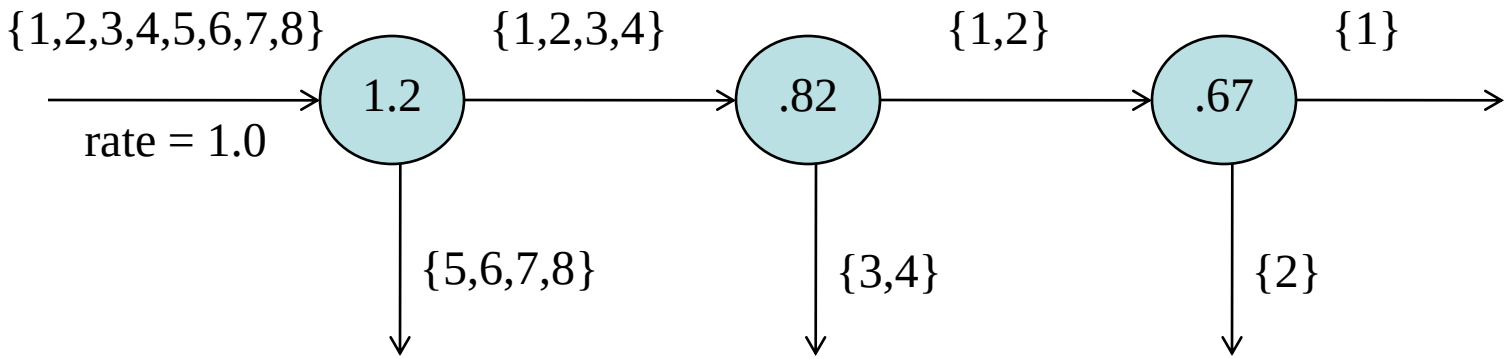
- Main idea: Instead of choosing a distribution of rules, choose the optimal best response algorithm for each state individually to derive a new rule.
- Initialize $c(S_m, A_k) := \sum_{h=1}^{H} \mathrm{I}(R_h(S_m) = A)$ based on existing rules.
- Update state transition probabilities $p(s' \mid s, a)$ and state utilities $Q(S_h)$ after processing stream.
- Determine optimal best response algorithm:

$$k^* := \arg\max_{k|A_k \in \mathcal{R}} \sum_{k=1}^{K} \sum_{s' \in \mathcal{S}} p(s' \mid s, A_k) Q(s')$$

- What's the benefit?
  - Might discover better rules that were missed in the existing set.
  - Makes better minimum performance guarantees (shown in simulations)

Experimental Setup for Static Streams

Setup: chain of 3 successive speech filtering classifiers
for identifying a speaker out of 8 people
from the UCIKDD archive [Kudo et al, 1999]



Different safe experiment parameters: $\varepsilon_t = 1/t, 1/\sqrt[3]{t}, t^{-t/50}$

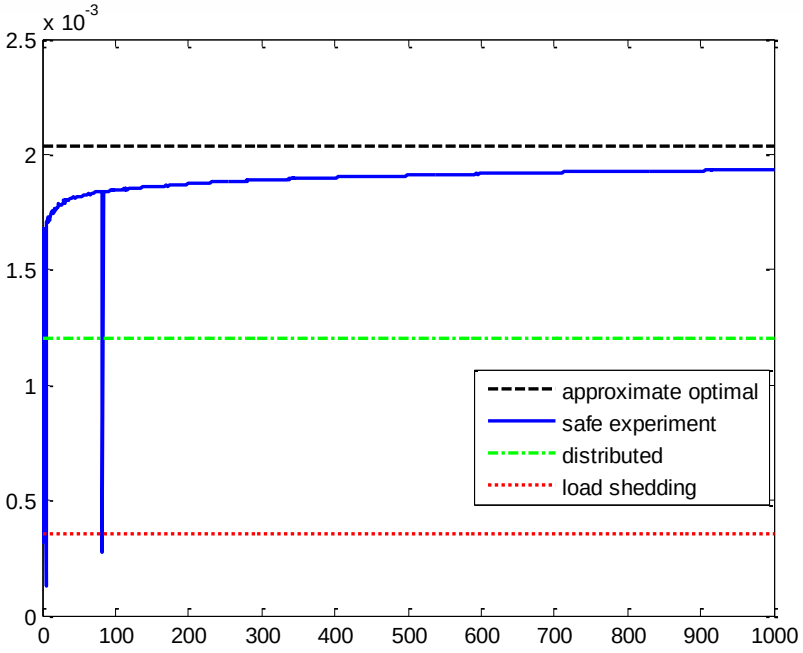
Perturbation: $Z_i(t) = N(0, \alpha/t^2)$

Other algorithms for comparison:

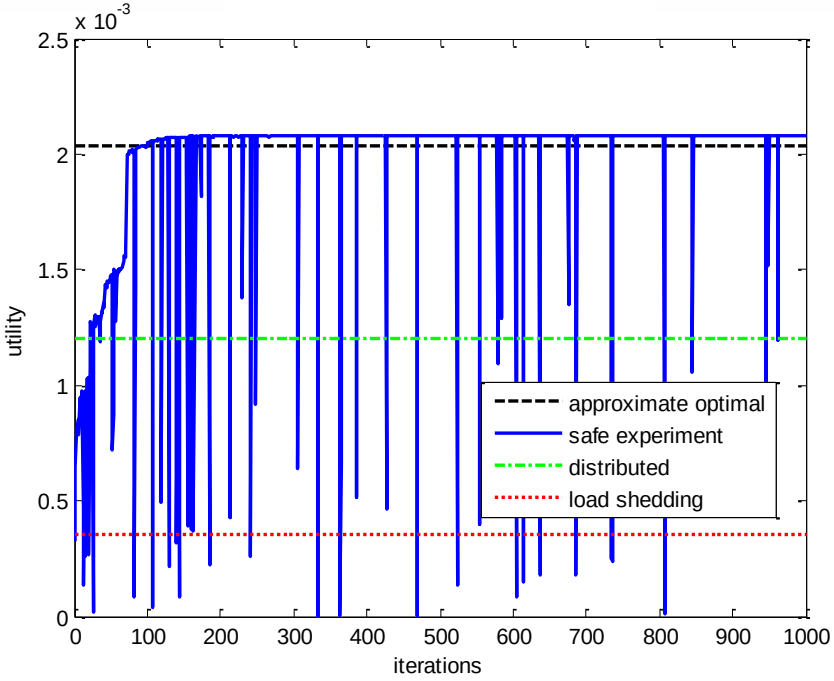
Random load shedding to decrease load to 0.7, i.e. prior work [Tatbul, Babcock, etc]

Distributed algorithm without information exchange: $\max \tilde{Q}_i(P_i^F) = (\tilde{\wp}_i - \theta_i(\tilde{\ell}_i - \tilde{\wp}_i)) \dfrac{\mu_{i+1} - \tilde{\lambda}_i \tilde{\ell}_i}{\mu_{i+1} - \tilde{\lambda}_i \tilde{\ell}_i + \varphi}$

Experimental Results



$\varepsilon_t = 1/t$



$\varepsilon_t = 1/\sqrt[3]{t}$

Experimental Setup for Dynamic Streams

Same application as before (3 chained classifiers for speaker recognition)

Stream APP changes randomly between 3-20% during each interval

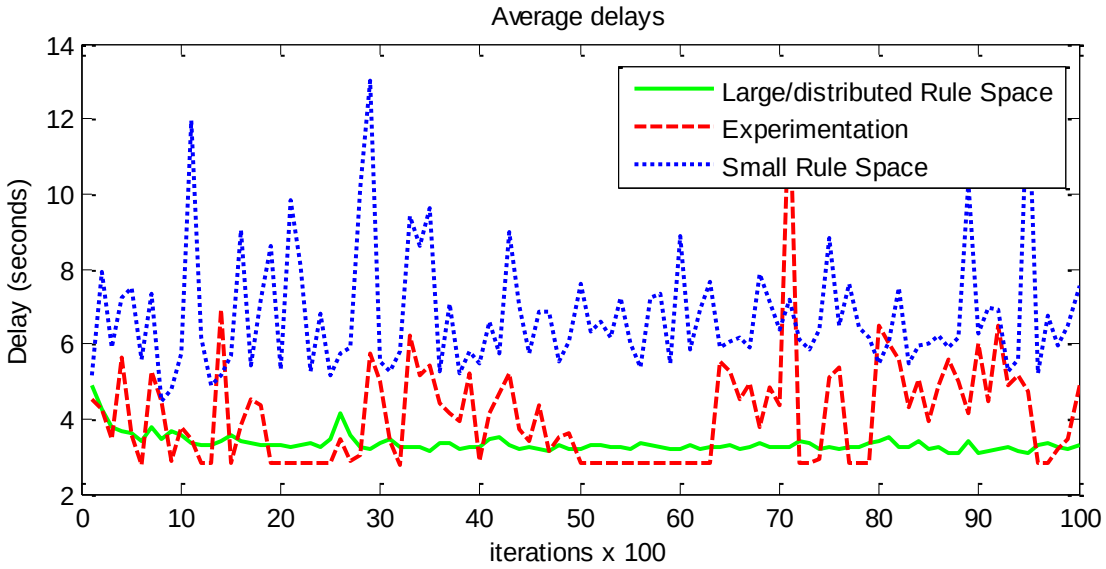
Comparison of 3 different rules-based approaches:

1) Single rule (safe experimentation/local search)
2) Small rules space (8 rules, 4 states, 4 algorithms)
3) Large distributed rules space (8 local rules, 8 local states, 4 local algorithms)

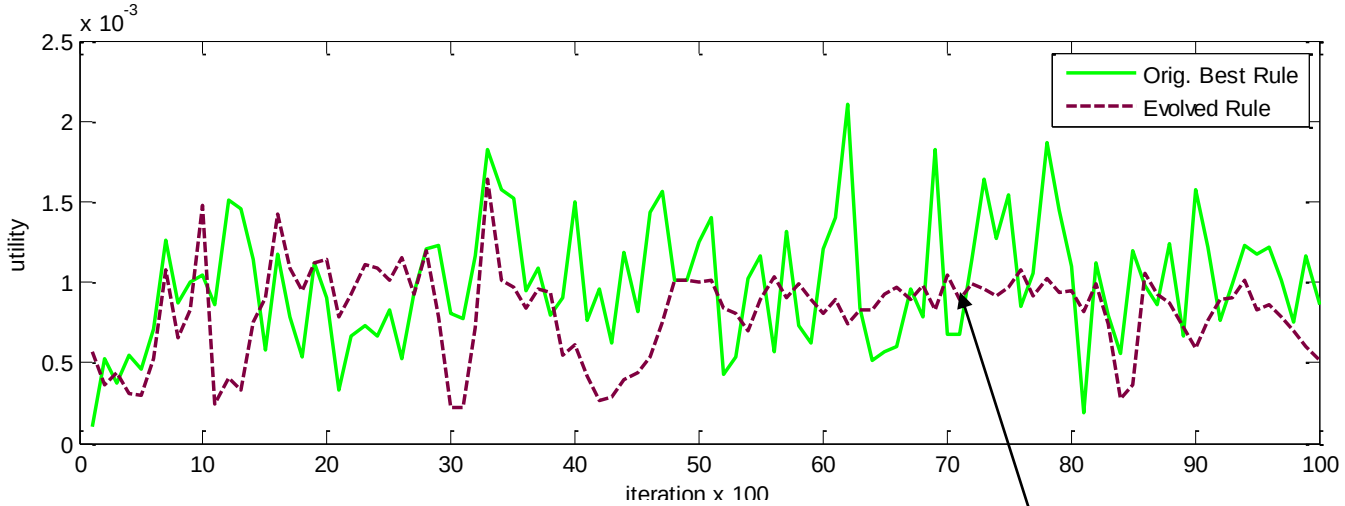
Experimental Results for Dynamic Streams

Stream APP changed between 5-20% per iteration

Approach	***Experimentation***		***Small Rule Space***		***Large Rule Space***	
	Lbled Spkr of Interest	*Lbled Other Speakers*	*Lbled Spkr of Interest*	*Lbled Other Speakers*	*Lbled Spkr of Interest*	*Lbled Other Speakers*
True Spkr of Interest	4.21	13.54	10.15	7.93	11.95	6.12
Other Speakers	11.06	153.66	29.97	126.21	9.58	146.61
Average Delay	3.96 secs.		6.51 secs.		3.42 secs.	



Results of Evolved Rule



Lower average perf, but usually better minimum perf.
This is a result of best-response play!

Summary of Main Contributions

- Stochastic Modeling of Multimedia Application Workload
  - RDC modeling for receiver-aware bitstream adaptation
    [Foo, Andreopoulos, vdSchaar, TSP 2008]
  - Forecasting workload requirements → Near-optimal DVS algorithms
    [Foo, vdSchaar, TSP 2008], [Cao, Foo, He, vdSchaar, DAC 2008]
  - Quality-complexity models → Fair/efficient resource allocation solutions for multiple tasks
    [Foo, vdSchaar, SPIE MCN 2008]

- Information Exchange Mechanisms
  - Enable distributed system to determine application performance
    [Foo, vdSchaar, SPIE MCA 2008]
  - Decentralized resource management
- Learning solutions
  - Collaboration between autonomous sites
    [Foo, vdSchaar, SPIE MCA 2008]
  - Distributed processing of dynamic multimedia applications and streams

Possible directions for future research

- Economics-motivated systems
  - System resource brokers
  - Auctioning of distributed services

- Joint optimization of our framework
  - Challenge: can optimal *joint* modeling, information exchange, and learning scheme be proposed for future systems?
  - Some expected benefits:
    - More efficient solutions for optimal resource allocation
    - Improved convergence/adaptation rate for dynamic streams

List of Accepted and Submitted Papers

Journal Papers (Accepted)

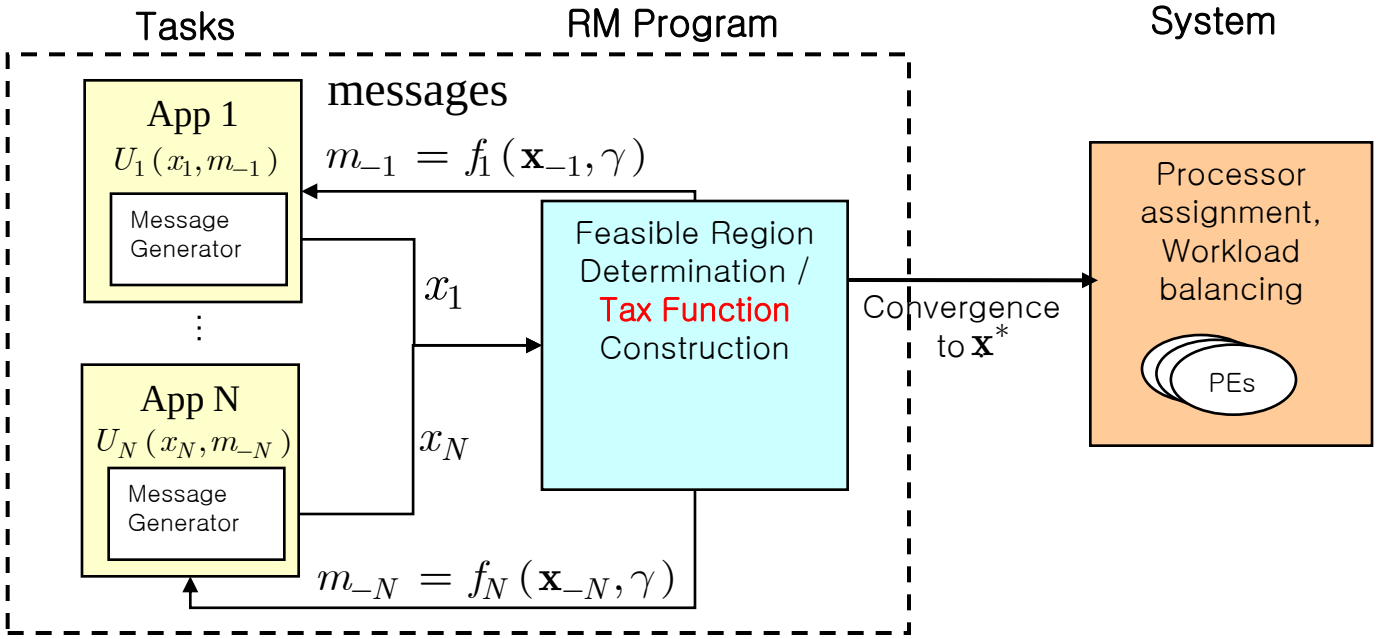
- **B. Foo**, Y. Andreopoulos, M. van der Schaar. "Analytical Rate-Distortion-Complexity Modeling of Wavelet-based Video Coders." *IEEE Trans. on Signal Processing*, Vol. 56, No. 2, Feb. 2008.
- **B. Foo**, M. van der Schaar. "A Queuing Theoretic Approach to Processor Power Adaptation for Video Decoding Systems." *IEEE Trans. on Signal Processing*, Vol 56, No. 1, Jan. 2008.
- **B. Foo,** D. Turaga, O. Verscheure, M. van der Schaar, L. Amini. "Resource Constrained Stream Mining with Classifier Tree Topologies," *IEEE Signal Processing Letters,* May. 2008.

Conference Papers

- **B. Foo**, M. van der Schaar, "Distributed optimization for real-time multimedia stream mining systems," *SPIE Multimedia Content Access,* Jan. 2008. (Invited paper)
- Z. Cao, **B. Foo,** L. He, M. van der Schaar, "Optimality and Improvement of Dynamic Voltage Scaling Algorithms for Multimedia Applications," *Design Automation Conference (DAC),* 2008. (Nominated for best paper award)
- **B. Foo**, M. van der Schaar, "Joint Scheduling and Resource Allocation for Multiple Video Decoding Tasks," *SPIE Multimedia Communications and Networking,* 2008.
- **B. Foo,** D. Turaga, O. Verscheure, M. van der Schaar, L. Amini, "Configuring Trees of Classifiers in Distributed Stream Mining Systems," *IBM Austin Center for Advanced Studies,* 2008.
- **B. Foo**, Y. Andreopoulos, M. van der Schaar. "Analytical Complexity Modeling of Wavelet-based Video Coders." *ICASSP* 2007.
- **B. Foo**, M. van der Schaar, "Queuing-theoretic Processor Power Adaptation for Video Decoding Systems." *ICIP* 2007.
- D. Turaga, **B. Foo,** O. Verscheure, R. Yan, "Configuring Topologies of Distributed Semantic Concept Classifiers for Continuous Multimedia Stream Processing," *submitted to ACM Multimedia,* 2008.

Decentralized Resource Management for Multiple Multimedia Tasks

- Streaming Applications –> Networked Devices
- Task information is *decentralized*
  - Autonomous tasks/users may not reveal their private information
- Cope with different system objectives
  - Workload balancing, energy minimization, utility maximization, etc.
- Proposed Solution: **Message Exchange Protocol** between tasks and RM



Tax functions for Decentralized Resource Allocation

- The system constructs a "tax function" to penalize each task.
  - Tax can reflect system energy cost, processor utilization, memory allocation, etc.
  - Penalties can affect "tokens" for future use of system resources.
- Each task submits its resource requirements to the RM, based on its achieved utility and the system tax.
  - App. utility can be calculated [Foo, vdSchaar, TSP, Feb. 2008]
- The system can construct different types of tax functions to achieve different results! Some examples:
  - Maximizing the sum of application utilities
  - Perform workload balancing across multiple processors
  - Dynamic voltage scaling for multiple tasks
- Must satisfy certain properties: e.g. KKT conditions for optimality in centralized objective

Tax functions for Decentralized Resource Allocation

- Maximizing Sum of Quality while Minimizing Energy Consumption
  - Centralized objective function (x is computational resources, e.g. cycles):

$$\max_{x_i} \sum_{i=1}^{I} Q_i(x_i) - \lambda E^* \left(\sum_{i=1}^{I} x_i \right)$$
$$\text{s.t.} \quad x_i \geq 0$$

  - Quality-complexity modeling [Foo, Andreopoulos, vdSchaar, TSP 2008]

  - Tax function assigned to each user (Excess-energy-minimization tax, or EEM):

$$t_i(x_i) = \lambda E^*(x_i + d_i) - \lambda E^*(d_i)$$

  - Application resource demand: $\arg\max_{x_i} U_i(x_i) = Q_i(x_i) - t_i(x_i)$
- Perceived computational resource demand from other users (updated based on prior resource demands and current resource demands):

$$d_i = \sum_{l \neq i}(x_l') + \frac{1}{\gamma}\sum_{l \neq i}(x_l - x_l')$$

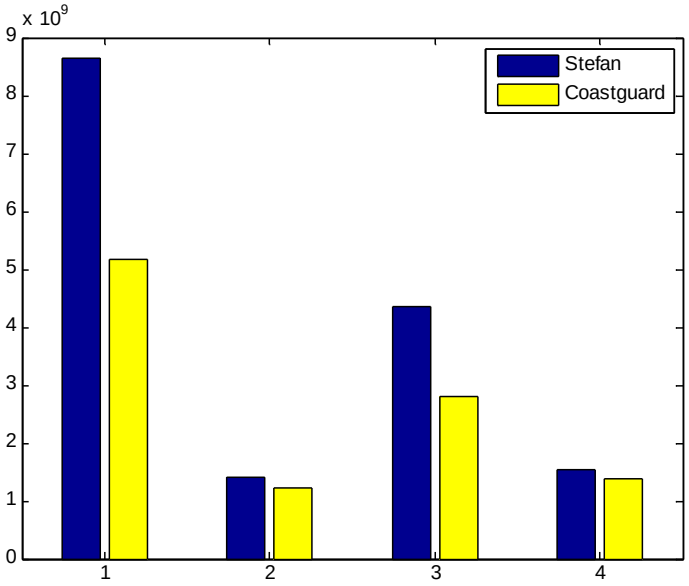
where $\gamma \geq 1$

is a dampening factor to prevent non-convergent dynamics

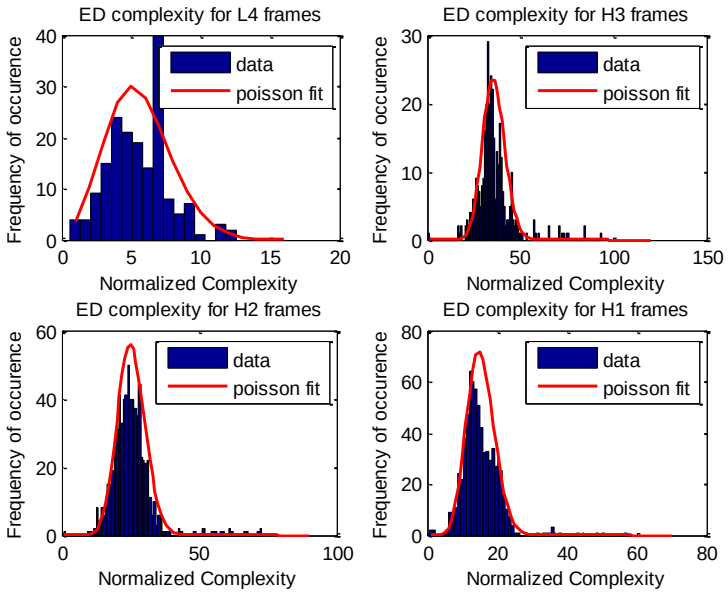
Modeling Multimedia Application Workload for Dynamic Voltage Scaling Algorithms

Enables *foresighted* decision-making and planning based on expected behavior of applications and system resource availabilities

Different classes of jobs, different stochastic models of the workloads (Mean, variance, delay distribution) [Foo, vdSchaar, TSP Jan. 2008]



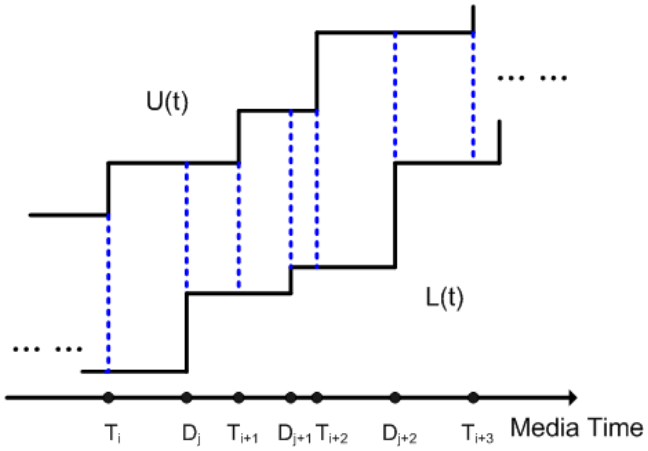
Comparison of various decoding jobs for video sequences *Stefan* and *Coastguard*.

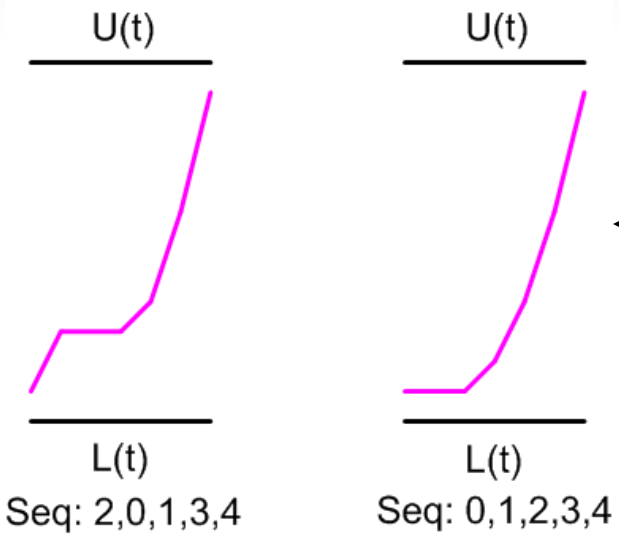


The workload variance within the same types of decoding jobs.

Application: Robust LP DVS algorithm

[Cao, Foo, He, vdSchaar, 2008]





Switching order incurs negligible overhead compared to processing multimedia tasks!

Independence of scheduling order enables us to formulate as a LP, NOT *integer* LP problem. Computationally tractable.

Fraction of time in adaptation interval i for using voltage level j.

$$\min \; E = \sum_{i=1}^{N} \sum_{j=0}^{K} (A_{ij} \bullet P_j \bullet (I_i - I_{i-1}))$$

Real-valued

$$0 \le A_{ij} \le 1, \; for \; 0 \le j \le K \; and \; \sum_{j=0}^{K} A_{ij} = 1$$

Workload based on Stochastic model. Robust LP.

$$L(I_n) \le \sum_{i=1}^{n} \sum_{j=0}^{K} (F_j \bullet A_{ij} \bullet (I_i - I_{i-1})) \le U(I_n), \forall \; 1 \le n \le L$$

Energy Savings Result