Configuring Competing Classifier Chains in Distributed Stream Mining Systems

Fangwen Fu, Deepak S. Turaga, *Member, IEEE*, Olivier Verscheure, Mihaela van der Schaar, *Senior Member, IEEE*, and Lisa Amini

***Abstract*—Networks of classifiers are capturing the attention of system and algorithmic researchers because they offer improved accuracy over single model classifiers, can be distributed over a network of servers for improved scalability, and can be adapted to available system resources. In this paper, we develop algorithms to optimally configure networks (chains) of such classifiers given system processing resource constraints. We first formally define a global performance metric for classifier chains by trading off the end-to-end probabilities of detection and false alarm. We then design centralized and distributed algorithms to provide *efficient* and *fair* resource allocation among several classifier chains competing for system resources. We use the Nash Bargaining Solution from game theory to ensure this. We also extend our algorithms to consider arbitrary topologies of classifier chains (with shared classifiers among competing chains). We present results for both simulated and state-of-the-art classifier chains for speaker verification operating on real telephony data, discuss the convergence of our algorithms to the optimal solution, and present interesting directions for future research.**

***Index Terms*—Nash bargaining solutions, networked classifiers, resource management, stream mining.**

I. INTRODUCTION

RECENTLY, there has been an emergence of a variety of new applications (such as sensor networks, location-tracking services and networking monitors) that require operations such as classification, filtering, aggregation and correlation over high-volume, unbounded and continuous data streams. Streaming data has come to include documents, email, instant messages, transactional data, digital audio, video and images etc. Each application may be viewed as a processing pipeline that analyzes streaming data from a set of raw data sources to extract valuable information in real time [1]. Due to the naturally distributed set of data sources and jobs, as well as high computational burdens for the analytics, distributed stream mining systems have been recently developed [2], [3]. These systems leverage computational resources from a set of distributed processing nodes and provide the framework to deploy and run different stream mining applications on various resource topologies [4]. In such systems, complex jobs are decomposed into a network of operators performing feature extraction, classification, aggregation, and correlation. Decomposition into an ensemble of operators has merits that transcend the scalability, reliability, and performance objectives of large-scale, real-time stream mining systems. For instance, it has been shown that, using classifiers operating in series with the same model boosting [5]) or classifiers operating in parallel with multiple models (bagging [6]) has resulted in improved classification performance. In this paper, we focus on such chains of networked classifiers,[1] each processing data for a registered job/query in the stream processing system.

A key research challenge [7], [8] in distributed stream mining systems, is the management of limited system resources (e.g., CPU, memory, I/O bandwidth etc.) while providing desired application performance. A majority of current stream mining systems take a database-centric approach where relational operators are applied to streaming data. Since relational operators are well-understood, the system can statically determine the optimal operator order and placement over the available resource topology. Although some systems like Telegraph CQ [9] provide dynamic adaptation to available resources, they do not factor in application knowledge to achieve the best resource-to-accuracy tradeoffs. System S [4] makes an attempt in this direction by additionally providing hooks for applications to determine current resource usage so that they can adapt suitably to the available resources. As the number of stream processing applications is growing rapidly, this issue must be addressed systematically.

Prior research on resource constrained stream mining applications falls into two broad categories. The first set of approaches rely on load-shedding, where algorithms determine a discard policy given the observed data characteristics e.g., burst, and the desired quality of service (QoS) requirements [8], [10]. Several of these approaches are limited by their assumption that the impact of load shedding on performance is known *a priori*. Furthermore, they often only consider simple data management jobs such as aggregation, for which the quality of the job result depends only on the sample size. Recent work on intelligent load shedding [11] attempts instead to maximize certain quality of decision (QoD) measures based on the predicted distribution of feature values in future time units. The second

Manuscript received January 30, 2007; revised September 3, 2007. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Zhi-Quan (Tom) Luo.

F. Fu and M. van der Schaar are with the Electrical Engineering Department, University of California Los Angeles (UCLA), Los Angeles, CA 90095-1594 USA (e-mail: fwfu@ee.ucla.edu; mihaela@ee.ucla.edu).

D. S. Turaga, O. Verscheure, and L. Amini are with the IBM T. J. Watson Research Center, Hawthorne, NY, 10532 USA (e-mail: turaga@us.ibm.com; ov1@us.ibm.com; aminil@us.ibm.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JSTSP.2007.909368

---

[1] We focus on binary classifiers for simplicity. Note though that this does not limit the generality of our solutions, as any M-ary classifier may be decomposed into a chain of binary classifiers.

set of approaches [3], [12] formulate the resource managament in terms of a filtering problem and the designed solutions filter out unrelated data as early as possible, to conserve resources.

While the aforementioned approaches have advantages in terms of simplicity of design and flexibility of implementation, they also have several limitations. Firstly, most of them do not explicitly consider the impact of load-shedding on application performance. Those that do, e.g., the QoD-based load-shedding, impose a significant overhead for computing these metrics. Secondly, they often consider only locally available information and metrics (e.g., data characteristics or accuracy or QoD metric at a particular point in the system). This may lead to suboptimal end-to-end performance when the data discarded at one stage is essential for a downstream (later stage) classifier. Finally, these solutions are primarily designed per applications, and there is no consideration of their impact across different competing (for resources) applications.

In order to overcome some of these limitations, we use an alternate approach. Expressed simply, instead of deciding on *what fraction* of the data to process, as in load-shedding based approaches, we determine *how* the available data should be processed given the underlying resource allocation. Hence, we allow individual classifiers in the ensemble to operate at different *performance levels* given the resources allocated to them. The performance level of each classifier contributes to the end-to-end utility of the chain. We define this utility in terms of the accuracy, i.e., desired tradeoff between probability of detection and probability of false alarm (other metrics, such as QoD may also be used). Using this defined end-to-end utility, our resource management problem may be formulated as a network optimization problem (NOP) [13], [14]. Each job requires the *flow* of data from source to destination along a classifier chain and different jobs compete for computational resources on distributed nodes. However, unlike traditional NOPs, that determine a transmission rate per job, we configure the performance level, in terms of the operating point, of each classifier to meet system resource constraints while maximizing the end-to-end performance. This simultaneous configuration of multiple classifiers, makes our problem significantly more complex than traditional NOPs. An additional difficulty is caused due to the non-concavity of our utility function.

Due to the potential competition for resources among applications, there are two essential issues that we need to explicitly consider during this optimization: 1) efficient utilization of the computational resources and 2) *fairness* across different application performance requirements. Traditional NOPs design maximum sum of utility solutions (MSUS), i.e., mainly focus on efficient resource utilization. Instead, we use a cooperative game-theoretic framework to address fairness, and employ Nash bargaining solutions (NBS) [15], [16] to formulate the optimization. The NBS is Pareto optimal, which guarantees efficient resource utilization. The NBS also provides a precise mathematical characterization of fairness (defined in Section IV). Interestingly, the NBS may be re-interpreted as providing proportional fairness [17] for resource management after satisfying the minimum utility requirements of each user.

We first develop a centralized optimization using sequential quadratic programming (SQP) to solve this problem. Such an approach requires the knowledge of all resource information, as well as data characteristics, and classifier operating points at a central location. This makes it infeasible in real systems with dynamic arrival and departure of applications, classifiers, changing data characteristics etc. Hence, using duality theory [18], we decompose the centralized NBS optimization into two distributed subproblems, a per job classifier configuration and a per-processing node resource optimization. We then develop distributed algorithms to solve these problems, and provide insights into their impact on physical resource allocation. In summary, we make the following contributions in this paper:

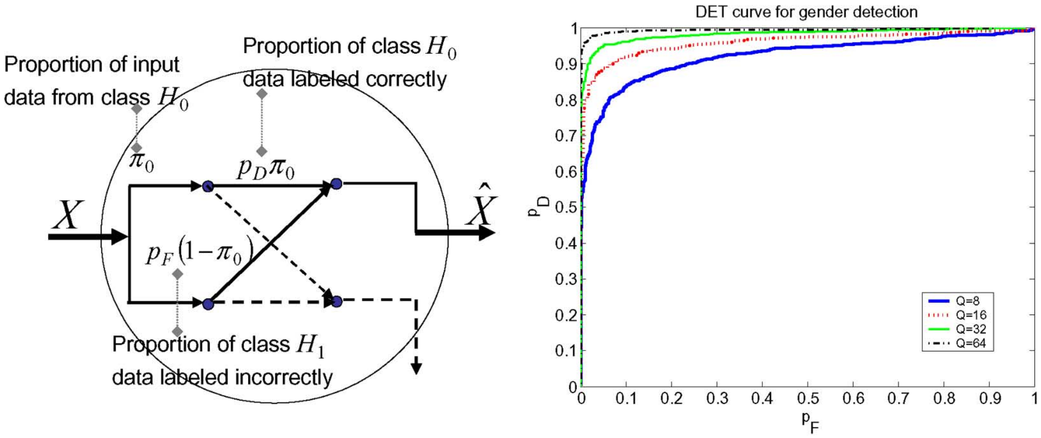
- we define utility associated with classifier chains by combining end-to-end detection and false alarm probabilities;
- we formulate the classifier configuration problem as a NOP and use NBS for efficient and fair solutions;
- we propose centralized and distributed solutions and provide insights into the system dynamics.

The paper is organized as follows. In Section II, we provide a background on classifiers, and define utility and resource consumption for individual classifiers. In Section III, we extend this to networked classifier chains. In Section IV, we formulate the resource-constrained configuration as a Nash Bargaining problem. We then provide the centralized solution in Section V and a distributed optimization in Section VI using duality theory. In Section VII we extend our solutions to consider arbitrary topologies of classifiers, i.e., with arbitrary sharing of classifiers across different chains. In Section VIII, we present simulation results and discuss the convergence of our algorithms. We conclude in Section IX with a summary of our findings, directions for future research, and a discussion on the application of these algorithms in conjunction with prior approaches such as load-shedding.

II. CLASSIFICATION BACKGROUND: BINARY FILTERING CLASSIFIERS

A binary filtering classifier classifies input data samples into belonging to either class $\mathcal{H}_0$ (class of interest) or class $\mathcal{H}_1$ and only forwards those samples that are classified as belonging to class $\mathcal{H}_0$ (discards samples classified as belonging to class $\mathcal{H}_1$). An example of such a filtering classifier is shown in Fig. 1.

The input data has *a priori* probability π_0 of belonging to class $\mathcal{H}_0$ and $(1-\pi_0)$ of belonging to class $\mathcal{H}_1$. This probability is assumed known or can be computed. The output of any classifier contains correctly as well as incorrectly classified data. Given X and $\hat{X}$, the input and output of the classifier, the proportion of correctly forwarded samples is captured by the probability of correct detection, $p_D = P(\hat{X} \in \mathcal{H}_0 | X \in \mathcal{H}_0)$, and the proportion of incorrectly forwarded samples is captured by the probability of false alarm $p_F = P(\hat{X} \in \mathcal{H}_0 | X \in \mathcal{H}_1)$. Classifier performance is characterized by a detection error tradeoff (DET) curve, i.e., the set of operating points (p_D, p_F) obtained by tuning its parameters. For example, in a Maximum Likelihood based classification scheme, different operating points are obtained by modifying the threshold. DET curves are concave and lie on or above the line $p_D = p_F$, between (0,0) and (1,1) [19]. A DET curve allows us to fully parameterize p_D as a function of p_F.


Proportion of input data from class H_0
Proportion of class H_0 data labeled correctly
π_0
$p_D\pi_0$
X
$\hat{X}$
$p_F(1-\pi_0)$
Proportion of class H_1 data labeled incorrectly
DET curve for gender detection
p_D
p_F
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Q=8
Q=16
Q=32
Q=64

(a) (b)

Fig. 1. Binary filtering classifier and example DET curves. (a) Binary filtering classifier. (b) Example DET curves.

The classifier performance is also heavily dependent on the underlying model complexity. For instance, a Gaussian mixture model (GMM) based parametric classifier's performance is likely to improve as Q, the number of Gaussians in the mixture, is increased. Similarly, the performance of a K-Nearest Neighbor algorithm is also likely to improve with K, the number of neighbors. This improved performance comes at the cost of increasing complexity, in terms of physical resources (CPU time, memory, and I/O bandwidth) required from the system, both for training and for online operation. In this paper, we assume a fixed underlying model complexity for each classifier.

The output rate of such a binary filtering classifier may be measured in terms of its throughput t and goodput g. The throughput captures the total data output by the classifiers (including both correctly and incorrectly classified samples), while the goodput captures the correctly classified data. When unit input data rate is processed by the classifier, the resulting t and g may be defined as

$$t = \pi_0 p_D + (1-\pi_0) p_F \tag{1}$$

$$g = \pi_0 p_D \tag{2}$$

where p_D, p_F, and π_0 are as defined earlier.

Utility u derived from a data mining classifier is computed by weighting p_D and p_F appropriately with the *cost of missed detection* and the *cost of false alarms* specific to any application. For instance, in the NIST evaluation of speaker verification [20], the cost of missed detection is set to be ten times as high as the cost of false alarms. Such a utility function may be written as $u = p_d - \theta p_F$, where $\theta > 0$ is the cost tradeoff ratio. Large θ reflects the preference for low p_F while small θ reflects the preference for high p_D. This performance metric has been used in one form or the other (e.g., as a confusion matrix, or as a receiver operating characteristic curve) in several diverse data mining and classification applications using several different types of classification algorithms. These diverse applications include the NIST speaker verification, credit card fraud detection, mining software for defects, intrusion detection systems, etc. [21]–[23].

A slightly modified version of this utility function may be written in terms of the actual rates t and g as

$$u = g - \theta(t-g) = (1+\theta)g - \theta t. \tag{3}$$

Given the DET curve, it is clear that this utility u is concave in p_F. At the same time, the resources R required by the classifier may be conveniently modeled as a linear function of the input rate, i.e., $R = \omega\rho$, where ω is the input rate, and ρ is a factor that captures the classifier model complexity, and also dependence on the underlying system architecture, implementation etc. For instance, for a GMM based classifier with Q Gaussians $\rho \propto Q$, as each data unit, requires $O(Q)$ units of computations.

III. CLASSIFIER CHAIN: UTILITY DEFINITION AND RESOURCE CONSUMPTION

A. *Binary Filtering Classifier Chain*

Consider a typical data mining problem that requires keyword (say "Hong Kong") spotting for a specific user (say "Mary") from a stream of speech samples collected from a networked microphone. We may partition this problem into a set of four exclusive questions (classifiers) with yes/no (binary) answers.

- Is the data segment speech or silence?
- Is the data segment from a female speaker?
- Is the data segment from speaker Mary?
- Does the data segment contain the keyword Hong Kong?

Note that the *exclusivity* comes from the fact that the desired set of data samples are only those with all four "yes" attributes. By partitioning the problem into this ensemble of four classifiers and filtering data (i.e., discarding data that is labeled "no" by any classifier) successively we can control the amount of resources consumed by each classifier in the ensemble. A chain of such N exclusive binary classifiers $(C^1 \cdots C^k \cdots C^N)$ is depicted in Fig. 2. The classifier configuration problem involves determining the operating point for each classifier given a set

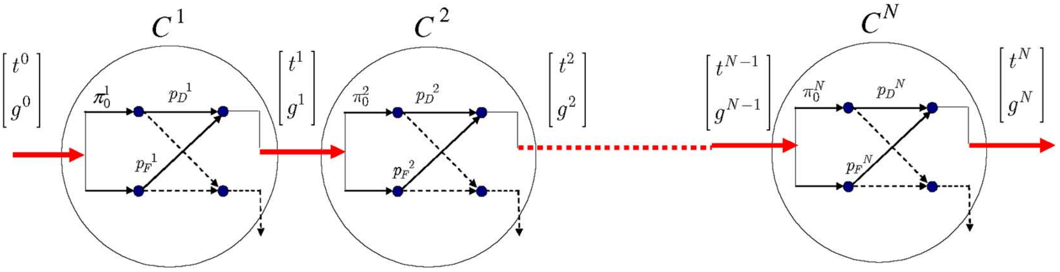

C^1
C^2
C^N
$\begin{bmatrix} t^0 \\ g^0 \end{bmatrix}$
$\begin{bmatrix} t^1 \\ g^1 \end{bmatrix}$
$\begin{bmatrix} t^2 \\ g^2 \end{bmatrix}$
$\begin{bmatrix} t^{N-1} \\ g^{N-1} \end{bmatrix}$
$\begin{bmatrix} t^N \\ g^N \end{bmatrix}$
π_0^1
$p_D{}^1$
$p_F{}^1$
π_0^2
$p_D{}^2$
$p_F{}^2$
π_0^N
$p_D{}^N$
$p_F{}^N$

Fig. 2. Filtering classifier chain with N classifiers.

of resource constraints, such that the end-to-end utility is maximized.

B. Utility Definition for Binary Filtering Classifier Chain

Consider the exclusive classifier chain shown in Fig. 2. The input for classifier C^k is the filtered output from classifier C^{k-1}. This filtering modifies the rate entering C^k, as well as the *a priori* probability of the data belonging to either class. Given the conditional probability $\phi^k = P(X \in \mathcal{H}_0^k | X \in \mathcal{H}_0^{k-1})$ relating the two classifiers, and the throughput t^{k-1} and goodput g^{k-1} for C^{k-1}, it is straightforward to show that we can compute t^k and g^k using a recursive relationship[2] as

$$g^k = \phi^k g^{k-1} p_D^k \tag{4}$$

and

$$t^k = t^{k-1} p_F^k + g^{k-1} \phi^k \left(p_D^k - p_F^k\right) \tag{5}$$

We may rewrite (4) and (5) using a matrix form as

$$\begin{bmatrix} t^k \\ g^k \end{bmatrix} = \underbrace{\begin{bmatrix} p_F^k & \phi^k \left(p_D^k - p_F^k\right) \\ 0 & \phi^k p_D^k \end{bmatrix}}_{\mathbf{T}^k\left(p_F^k\right)} \begin{bmatrix} t^{k-1} \\ g^{k-1} \end{bmatrix} \tag{6}$$

where the matrix $\mathbf{T}^k(p_F^k)$ is the transfer matrix that is a function of the operating point p_F^k. We ignore p_D^k in the above because, the DET curve uniquely determines p_D^k as a function of p_F^k. For further ease of notation, we also drop p_F^k and use $\mathbf{T}^k$ to denote the transfer matrix. Note that $t^0 = g^0 = 1$. By recursing from N to 1, we have

$$\begin{bmatrix} t^N \\ g^N \end{bmatrix} = \mathbf{T}^N \mathbf{T}^{N-1} \cdots \mathbf{T}^1 \begin{bmatrix} 1 \\ 1 \end{bmatrix} \tag{7}$$

where t^N and g^N represent the end-to-end throughput and goodput from this chain, and hence, as discussed earlier the end-to-end utility may be defined as

$$u(\mathbf{p_F}) = g^N - \Theta(t^N - g^N) = (1+\Theta)g^N - \Theta t^N \tag{8}$$

where $\mathbf{p_F}$ is the operating point vector, i.e., $\mathbf{p_F} = [p_F^1 \cdots p_F^N]^T$, and $\Theta \geq 0$, is a factor that trades off g^N with $(t^N - g^N)$. Hence, the utility can be rewritten as

$$u(\mathbf{p_F}) = [-\Theta \quad 1+\Theta] \mathbf{T}^N \mathbf{T}^{N-1} \cdots \mathbf{T}^1 \begin{bmatrix} 1 \\ 1 \end{bmatrix}. \tag{9}$$

C. Resource Consumption of Binary Filtering Classifier Chain

Each classifier C^k has a certain underlying model complexity and consumes the rate-dependent resource, ρ^k, per unit data. Given, input data rate ω, the data rate ω^k entering C^k becomes $\omega^k = \omega t^{k-1}$. Accordingly, the resource, R^k, consumed by classifier C^k is $R^k = \omega^k \rho^k = \omega t^{k-1} \rho^k$. The throughput t^{k-1}, entering C^k depends on the selected operating points (p_F^i, p_D^i) (or equivalently only p_F^i, given the DET curve) of all preceding classifiers $1 \leq i \leq k-1$.

D. Properties of the Utility Function

The considered utility function $u(\mathbf{p_F})$ has several *nice* properties that we wish to highlight

- The parameters p_F^k are continuous, and the function u is differentiable.
- The minimum utility a classifier chain can receive is 0 (when all data is discarded). Hence any algorithm to determine the optimal configuration $\mathbf{p_F^{\text{opt}}}$ to maximize the utility needs to search only the restricted subregion where $u \geq 0$.
- The utility u is concave along each individual axis (p_F^k) inside the subregion where $u \geq 0$, i.e., for a set of fixed $\{p_F^1, \ldots, p_F^{k-1}, p_F^{k+1}, \ldots, p_F^N\}$, the utility $u(p_F^k)$ is concave inside this region. We can show this by rewriting (9) as

$$u\left(p_F^k\right) = \underbrace{[-\Theta \quad 1+\Theta]\mathbf{T}^N \cdots \mathbf{T}^{k+1}}_{[x_1 \quad x_2]} \times \begin{bmatrix} p_F^k & \phi^k \left(p_D^k - p_F^k\right) \\ 0 & \phi^k p_D^k \end{bmatrix} \begin{bmatrix} t^{k-1} \\ g^{k-1} \end{bmatrix}. \tag{10}$$

Expanding these terms, we have

$$\begin{aligned} u\left(p_F^k\right) &= x_1 t^k + x_2 g^k \\ &= \phi^k (x_1 + x_2) g^{k-1} p_D^k + x_1 (t^{k-1} - \phi^k g^{k-1}) p_F^k \end{aligned} \tag{11}$$

Since p_D is a concave function of p_F and $\phi^k g^{k-1} \geq 0$, we only need to show that $x_1 + x_2 \geq 0$ to prove the concavity of $u(p_F^k)$. Given that $u \geq 0$, we have $x_1 t^k + x_2 g^k \geq 0$. Combining with the fact that $x_1 \leq 0$ and $t^k \geq g^k$, we can directly conclude that $x_1 + x_2 \geq 0$ in this subregion. Hence, $u(p_F^k)$ is concave inside this subregion. It is important to note that despite this concavity along each individual axis, this function is not necessarily concave in all

---

[2] Note that exclusivity implies $P(X \in \mathcal{H}_0^k | X \in \mathcal{H}_1^{k-1}) = 0$. Please see [24] for more details.

directions (especially those not along any axis), i.e., it is not multidimensional (MD) concave in general.

- We observe that the region $u \geq 0$ contains a subregion inside which u is MD concave. This concave subregion contains the origin, i.e., $\mathbf{p_F} = \mathbf{0}$.

IV. Problem Formulation: Configuring Resource Constrained Networked Classifiers

A. System Description

In distributed stream mining systems, there are often multiple classifier chains concurrently competing for limited system resources. By configuring (selecting the operating point parameterized by p_F) a classifier in the chain, we can control the resource consumption of all other classifiers in the chain downstream from it. At the same time, we also impact the end-to-end utility. Furthermore, given that classifiers from multiple chains may compete for resources (i.e., when they coexist on common processing nodes) this configuration also affects the resource consumption, and correspondingly utility, of other classifier chains. This makes resource management under shared infrastructure a challenging problem.

We consider a system with I binary filtering exclusive classifier chains. Chain $i (1 \leq i \leq I)$ has N_i classifiers with the k-th classifier denoted as C_i^k. These chains are instantiated on M processing nodes $(1 \cdots M)$, each of which has available resources $\mathcal{R}_m$ $(1 \leq m \leq M)$. We construct a location matrix, $\mathbf{A}_i = \{a_i(m,k)\}_{M \times N_i}$ for each chain, with $a_i(m,k) = 1$ if C_i^k is placed on processing node m and 0 otherwise. Additionally, we define the following notations:

- $\boldsymbol{\mathcal{R}} = [\mathcal{R}_1 \cdots \mathcal{R}_M]^T$: the M-dimensional vector[3] of available rate-dependent resources;
- $\boldsymbol{\rho}_i = [\rho_i^1 \cdots \rho_i^{N_i}]^T$: the N_i-dimensional vector of the resources consumed per unit data rate for chain i;
- $\mathbf{p}_{\mathbf{F}_i} = [p_{F_i}^1 \cdots p_{F_i}^{N_i}]^T$: the N_i-dimensional false alarm probability vector for chain i. This represents the operating points of each classifier in the chain;
- $\mathcal{M}(i)$: the set of processing nodes that chain i is located on;
- $\mathcal{C}(i,m)$: the set of classifiers of the chain i located on processing node m;
- $\mathbf{P_F} = [\mathbf{p}_{\mathbf{F}_1}^T \cdots \mathbf{p}_{\mathbf{F}_I}^T]^T$: the $(\sum_{i=1}^I N_i)$-dimensional false alarm probability vector aggregated across all the chains.
- $\omega = [\omega_1 \cdots \omega_I]$: the I-dimensional input data rate vector of all chains with ω_i being the input data rate to chain i;
- $\mathbf{h}_i(\mathbf{p}_{\mathbf{F}_i}) = [h_i^1 \cdots h_i^{N_i}]^T$: the N_i-dimensional resource consumption vector for chain i with

$$h_i^k = \omega_i t_i^{k-1} \rho_i^k \tag{12}$$

- $\mathbf{u}(\mathbf{P_F}) = [u_1(\mathbf{p}_{\mathbf{F}_1}) \cdots u_{N_i}(\mathbf{p}_{\mathbf{F}_I})]^T$: the I-dimensional utility vector of all the chains with the element $u_i(\mathbf{p}_{\mathbf{F}_i})$ being the utility function of chain i, computed as in (8).

In this paper, we assume that each classifier C_i^k operates with a certain *fixed* model complexity and the *corresponding DET curve is known a priori* (thereby allowing us to write all equations purely in terms of p_F^k, ignoring p_D^k). Our optimization involves configuring the operating points of individual classifiers to maximize the utilities derived by each chain, given the resource constraints. Note that the minimum utility for any chain is at least zero, obtained by setting $p_{F_i^k} = 0, \forall k \in \{1, \cdots, N_i\}$, i.e., no resource consumption. Hence, the feasible configuration (solution) set $\boldsymbol{\mathcal{S}}$ may be defined as follows.

Definition 1: Feasible Configuration Set: $\boldsymbol{\mathcal{S}}$ is the set of operating points for all the chains, $\mathbf{P_F}$, which results in no violation of the resource constraints with utilities of all chains being greater than zero, i.e.,

$$\boldsymbol{\mathcal{S}} = \Big\{ \mathbf{P_F} \Big| \sum_{i=1}^{I} \mathbf{A}_i \mathbf{h}_i\left(\mathbf{p}_{\mathbf{F}i}\right) \leq \boldsymbol{\mathcal{R}}, u_i\left(\mathbf{p}_{\mathbf{F}i}\right) > 0, \\ \mathbf{0} \leq \mathbf{p}_{\mathbf{F}i} \leq \mathbf{1}, \forall i \in \{1, \cdots, I\} \Big\}. \tag{13}$$

We assume that $\boldsymbol{\mathcal{S}}$ is nonempty (otherwise the problem has only a trivial solution). For each possible configuration $\mathbf{P_F} \in \boldsymbol{\mathcal{S}}$, we can derive the corresponding utility (defined as in (8)) vector $\mathbf{u}(\mathbf{P_F})$. The feasible set of utility vectors derived from $\boldsymbol{\mathcal{S}}$ may be defined as

$$\boldsymbol{\mathcal{U}} = \{\mathbf{u}(\mathbf{P_F}) | \mathbf{P_F} \in (\boldsymbol{\mathcal{S}})\}. \tag{14}$$

B. Nash Bargaining Solutions

While optimizing the utility vector, we need a notion of efficiency of any configuration. We thus need formal optimality criteria, and hence define Pareto optimality in the utility set as follows:

Definition 2: Pareto Optimality: The feasible utility vector $\mathbf{u} \in \boldsymbol{\mathcal{U}}$ is Pareto optimal if for each $\mathbf{v} \in \boldsymbol{\mathcal{U}}$ and $\mathbf{v} \geq \mathbf{u}$, then $\mathbf{v} = \mathbf{u}$.

Intuitively, a Pareto optimal configuration is one for which there are no other feasible utility vectors such that all chains have strictly greater utilities, simultaneously. Pareto optimality thus leads to an efficient utilization of the limited system resources. However, in general, there are multiple Pareto optimal points in the utility set. We thus need to select the *right*} Pareto optimal point for the system to operate at. This may be determined by further requiring *fairness* in solution. Fair resource allocation has been well understood in game-theoretic literature and hence we may use axioms from game theory in this context to formally define it. In this paper, we use the NBS [8] to achieve fairness (in the utility space) and efficient resource allocation.

In order to define the NBS, we first introduce the notion of a disagreement point. The disagreement point is defined as $\mathbf{d} = [d_1 \cdots d_I]$, with the i-th element representing the minimum utility agent (chain) i receives if it is involved in the resource allocation game. The bargaining problem consists of the feasible utility set $\boldsymbol{\mathcal{U}}$ [(14)], and the disagreement point $\mathbf{d}$, as well as be a function: $\mathbf{F} : (\boldsymbol{\mathcal{U}}, \mathbf{d}) \mapsto \mathbb{R}_+^I$ that maps them into a particular utility allocation with the following properties.

Definition 3: Nash Bargaining Solution: The utility point $\mathbf{u} = \mathbf{F}(\boldsymbol{\mathcal{U}}, \mathbf{d})$ is an NBS if the following axioms are satisfied [16].

1) Individual rationality: $\mathbf{u} \geq \mathbf{d}$.
2) Feasibility: $\mathbf{u} \in \boldsymbol{\mathcal{U}}$.

---

[3] We use the symbol ^{T} to represent transpose.

3) Pareto optimality: $\mathbf{u}$ is Pareto optimal.
4) Independency of irrelevant alternatives: if $\mathbf{u} \in \boldsymbol{\mathcal{U}}' \subset \boldsymbol{\mathcal{U}}$ and $\mathbf{u} = \mathbf{F}(\boldsymbol{\mathcal{U}}, \mathbf{d})$, then $\mathbf{u} = \mathbf{F}(\boldsymbol{\mathcal{U}}', \mathbf{d})$.
5) Independence of linear transformations: for any linear scale transform ψ: $\psi(\mathbf{F}(\boldsymbol{\mathcal{U}}, \mathbf{d})) = \mathbf{F}(\psi(\boldsymbol{\mathcal{U}}), \psi(\mathbf{d}))$.
6) Symmetry: if $\boldsymbol{\mathcal{U}}$ is invariant under all exchanges of agents, then $\mathbf{F}(\boldsymbol{\mathcal{U}}^{i\leftrightarrow j}, \mathbf{d}^{i\leftrightarrow j}) = \mathbf{F}(\boldsymbol{\mathcal{U}}, \mathbf{d})$ for all i, j, where $\boldsymbol{\mathcal{U}}^{i\leftrightarrow j}$ and $\mathbf{d}^{i\leftrightarrow j}$ are the utility set and disagreement point when i, j are interchanged.

Items 1, 2, and 3 ensure that the NBS satisfies the minimum utility requirements, is Pareto optimal, and is reachable. Items 4, 5, and 6 address fairness issues. The independency of irrelevant alternatives states that the NBS is not affected by enlarging the utility set if the solution is found in a smaller domain. The independence of linear transformations leads to scale invariance of the NBS. Finally, the symmetry of the NBS ensures that the solution does not depend on underlying labels.

It has been shown that there exists a unique NBS, $\mathbf{u}^{nbs}$, that satisfies the above six axioms and this NBS is the solution of the following optimization problem [16]:

$$\mathbf{u}^{nbs} = \arg\max_{\mathbf{u}\in\boldsymbol{\mathcal{U}}} \prod_{i=1}^{I} (u_i - d_i). \tag{15}$$

Our classifier configuration problem, involves solving the optimization problem in (15) for $\mathbf{P_F}$. As stated earlier, any chain can get zero utility without being involved in the configuration game. Hence, the disagreement point in this paper is assumed to be $\mathbf{d} = \mathbf{0}$. Additionally, since the function $\ln(x)$ is monotonically increasing over the range $(0, \infty)$, the optimal configuration $\mathbf{P}_\mathbf{F}^{\text{opt}}$ for the NBS may equivalently be obtained as the solution to the following optimization problem:

$$\mathbf{P}_\mathbf{F}^{\text{opt}} = \arg\max_{\mathbf{P_F}\in\boldsymbol{\mathcal{S}}} \sum_{i=1}^{I} \ln\left(u_i\left(\mathbf{p_F}_i\right)\right). \tag{16}$$

Note that the solution to the above optimization also leads to proportional fairness, i.e., resource allocation in a way that proportionally increases utilities of the different chains.[4] In summary, we may write out the constrained optimization problem we wish to solve as

$$\begin{aligned} &\max_{\mathbf{P_F}} \sum_{i=1}^{I} \ln\left(u_i\left(\mathbf{p_F}_i\right)\right) \\ &\text{subject to} \sum_{i=1}^{I} \mathbf{A}_i \mathbf{h}_i\left(\mathbf{p_F}_i\right) \le \boldsymbol{\mathcal{R}} \\ &\qquad u_i\left(\mathbf{p_F}_i\right) > 0 \\ &\qquad \mathbf{0} \le \mathbf{p_F}_i \le \mathbf{1}, \forall i. \end{aligned} \tag{17}$$

Note that while we explicitly mention the search space constraint $u_i(\mathbf{p}_{\mathbf{F}_i}) > 0$ in the above formulation, the use of $ln(u_i(\mathbf{p}_{\mathbf{F}_i}))$ automatically makes this constraint superfluous. Hence, we drop it in the rest of the paper.

V. Centralized Optimization for NBS

This optimization in (17) may be solved using SQP [25]. The basic idea behind SQP lies in modeling the nonlinear optimization problem at each iteration j by an approximate quadratic programming subproblem. The solution to the approximate quadratic subproblem $\mathbf{P}_\mathbf{F}^{(j)}$, is then used to construct a better approximation at the $j + 1$-th iteration, $\mathbf{p}_\mathbf{F}^{(j+1)}$. This procedure is repeated to create a sequence of approximations that converge to the optimal solution $\mathbf{p}_\mathbf{F}^{\text{opt}}$. It is important to note that the SQP algorithm may converge to a locally optimal solution. Although the SQP has been extensively discussed in literature [18], [25], we present the procedure for our problem at each iteration here, for completeness. We first define a new Lagrangian function $\boldsymbol{\mathcal{L}}^{SQP}$ as

$$\begin{aligned} \boldsymbol{\mathcal{L}}^{SQP}(\mathbf{P_F}, \eta, \boldsymbol{\xi}, \boldsymbol{\nu}) &= \sum_{i=1}^{I} \ln\left(u_i\left(\mathbf{p_F}_i\right)\right) \\ -\boldsymbol{\lambda}^T \left(\sum_{i=1}^{I} \mathbf{A}_i \mathbf{h}_i\left(\mathbf{p_F}_i\right) - \boldsymbol{\mathcal{R}}\right) &- \boldsymbol{\xi}^T(\mathbf{P_F} - 1) + \boldsymbol{\nu}^T \mathbf{P_F} \end{aligned} \tag{18}$$

where $\boldsymbol{\lambda}$ corresponds to the inequality $\sum_{i=1}^{I} \mathbf{A}_i \mathbf{h}_i(\mathbf{p}_{\mathbf{F}_i}) \le \boldsymbol{\mathcal{R}}$, $\boldsymbol{\xi} \in \mathbb{R}_+^{\sum_{i=1}^{I} N_i}$ corresponds to the inequality $\mathbf{P_F} \le \mathbf{1}$ and $\boldsymbol{\nu} \in \mathbb{R}_+^{\sum_{i=1}^{I} N_i}$ corresponds to the inequality $\mathbf{P_F} \ge \mathbf{0}$.

At iteration j, the objective function of (17) is approximated by a quadratic function and the constraints are linearized around the current approximation $\mathbf{p}_\mathbf{F}^{(j)}$. In particular, the quadratic subproblem has the following form:

$$\begin{aligned} &\max_{\mathbf{v}} \frac{1}{2} \mathbf{v}^T \boldsymbol{\Gamma}^{(j)} \mathbf{v} + \nabla\left(\sum_{i=1}^{I} \ln\left(u_i\left(\mathbf{p_F}_i^{(j)}\right)\right)\right)^T \mathbf{v} \\ &\text{subject to } \nabla\left(\sum_{i=1}^{I} \mathbf{A}_i \mathbf{h}_i\left(\mathbf{p_F}_i^{(j)}\right)\right)^T \mathbf{v} \\ &\qquad + \sum_{i=1}^{I} \mathbf{A}_i \mathbf{h}_i\left(\mathbf{p_F}_i^{(j)}\right) \le \boldsymbol{\mathcal{R}} \\ &\qquad \mathbf{P}_\mathbf{F}^{(j)} + \mathbf{v} \le \mathbf{1} \\ &\qquad \mathbf{P}_\mathbf{F}^{(j)} + \mathbf{v} \ge \mathbf{0} \end{aligned} \tag{19}$$

where $\mathbf{v}$ is the search direction of $\mathbf{P_F}$. The matrix $\boldsymbol{\Gamma}^{(j)}$ is a positive definite approximation of the Hessian matrix of the Lagrangian function in (18). $\boldsymbol{\Gamma}^{(j)}$ can be updated by any of the quasi-Newton methods, e.g., the BFGS method [26]. The $j+1$th approximation $\mathbf{P}_\mathbf{F}^{(j+1)}$ is then computed as

$$\mathbf{P}_\mathbf{F}^{(j+1)} = \mathbf{P}_\mathbf{F}^{(j)} + \gamma^{(j)} \mathbf{v} \tag{20}$$

where the step length parameter $\gamma^{(j)}$ is determined by an appropriate line search procedure so that a sufficient decrease in a merit function is obtained.

VI. Distributed Algorithm for NBS

We label our formulation in (17) as the *primal problem*, using optimization theory terminology. Instead of solving the primal problem directly, we instead use duality theory [18] to define the *dual problem*, present its properties, and design a distributed

---

[4] While in this paper we do not explicitly consider priorities or "weights" associated with different chains, they may easily be introduced into the formulation as multiplicative factors to $ln(u_i)$. Furthermore, our designed algorithms are directly applicable to scenarios with different priorities.

solution for it. As we will show, the dual problem has several advantages over the primal problem [27]. Firstly, unlike the primal problem, the dual may be solved using convex programming techniques. Secondly, since the objective function and part of the constraints use summations, the optimization problem is separable, enabling a distributed algorithm. Finally, the dual problem is computationally more feasible than the primal problem, as it deals with a smaller number of variables at one time, especially when the number of chains is greater than the number of processing nodes $(I > M)$.

A. Dual Problem

The constrained optimization in (17) may be rewritten as a Lagrangian as

$$\mathcal{L}(\mathbf{P_F}, \boldsymbol{\lambda}) = \sum_{i=1}^{I} \ln\left(u_i\left(\mathbf{p_F}_i\right)\right) - \boldsymbol{\lambda}^T \left(\sum_{i=1}^{I} \mathbf{A}_i \mathbf{h}_i\left(\mathbf{p_F}_i\right) - \boldsymbol{\mathcal{R}}\right) \tag{21}$$

where $\boldsymbol{\lambda} = [\lambda_1 \cdots \lambda_M]^T \in \mathbb{R}_+^M$ with λ_m being the Lagrange multiplier associated with the resource constraint on processing node m. Using duality theory, we may then write the dual problem as

$$\begin{aligned} &\min_{\boldsymbol{\lambda}} D(\boldsymbol{\lambda}), \\ &\text{subject to } \boldsymbol{\lambda} \geq 0 \end{aligned} \tag{22}$$

where $D(\boldsymbol{\lambda})$ is computed as

$$\begin{aligned} D(\boldsymbol{\lambda}) = &\max_{\mathbf{P_F}} \boldsymbol{\mathcal{L}}(\mathbf{P_F}, \boldsymbol{\lambda}), \\ &\text{subject to } \mathbf{0} \leq \mathbf{p_F}_i \leq \mathbf{1}, \forall i. \end{aligned} \tag{23}$$

It can be easily shown that $D(\boldsymbol{\lambda})$ is a convex function of $\boldsymbol{\lambda}$ [18], and the dual problem has a set of convex constraints on $\boldsymbol{\lambda}$. Hence, the dual problem can be solved efficiently if we can compute $D(\boldsymbol{\lambda})$.

It can be shown that when an appropriate Lagrange multiplier is selected, the optimal solution of the dual problem approaches the optimal solution of the primal problem. In general though, the dual solution overestimates the primal solution, by an amount referred to as the *duality gap*. We will discuss the duality gap for our problem in Section VIII-D.

B. Decomposition of Dual Problem

The Lagrangian in (21), may further be rewritten as

$$\begin{aligned} \mathcal{L}(\mathbf{P_F}, \boldsymbol{\lambda}) &= \sum_{i=1}^{I} \ln\left(u_i\left(\mathbf{p_F}_i\right)\right) - \boldsymbol{\lambda}^T \left(\sum_{i=1}^{I} \mathbf{A}_i \mathbf{h}_i\left(\mathbf{p_F}_i\right) - \boldsymbol{\mathcal{R}}\right) \\ &= \sum_{i=1}^{I} \Big[\ln\left(u_i\left(\mathbf{p_F}_i\right)\right) - \boldsymbol{\lambda}^T \mathbf{A}_i \mathbf{h}_i\left(\mathbf{p_F}_i\right)\Big] + \boldsymbol{\lambda}^T \boldsymbol{\mathcal{R}}. \end{aligned} \tag{24}$$

Hence, the dual problem in (22) and (23) can be modified as

$$\begin{aligned} &\min_{\boldsymbol{\lambda} \geq 0} \left\{ \max_{\mathbf{0} \leq \mathbf{p_F}_i \leq \mathbf{1}, \forall i} \boldsymbol{\mathcal{L}}(\mathbf{P_F}, \boldsymbol{\lambda}) \right\} \\ &= \min_{\boldsymbol{\lambda} \geq 0} \Bigg\{ \max_{\mathbf{0} \leq \mathbf{p_F}_i \leq \mathbf{1}, \forall i} \sum_{i=1}^{I} \Big[\ln\left(u_i\left(\mathbf{p_F}_i\right)\right) - \boldsymbol{\lambda}^T \mathbf{A}_i \mathbf{h}_i\left(\mathbf{p_F}_i\right)\Big] \\ &\qquad + \boldsymbol{\lambda}^T \boldsymbol{\mathcal{R}} \Bigg\} \\ &= \min_{\boldsymbol{\lambda} \geq 0} \Bigg\{ \sum_{i=1}^{I} \max_{\mathbf{0} \leq \mathbf{p_F}_i \leq \mathbf{1}, \forall i} \Big[\ln\left(u_i\left(\mathbf{p_F}_i\right)\right) - \boldsymbol{\lambda}^T \mathbf{A}_i \mathbf{h}_i\left(\mathbf{p_F}_i\right)\Big] \\ &\qquad + \boldsymbol{\lambda}^T \boldsymbol{\mathcal{R}} \Bigg\} \\ &= \min_{\boldsymbol{\lambda} \geq 0} \left\{ \sum_{i=1}^{I} \max_{\mathbf{0} \leq \mathbf{p_F}_i \leq \mathbf{1}, \forall i} \boldsymbol{\mathcal{L}}_i\left(\mathbf{p_F}_i, \boldsymbol{\lambda}\right) + \boldsymbol{\lambda}^T \boldsymbol{\mathcal{R}} \right\} \end{aligned} \tag{25}$$

where

$$\boldsymbol{\mathcal{L}}_i\left(\mathbf{p_F}_i, \boldsymbol{\lambda}\right) = \ln\left(u_i\left(\mathbf{p_F}_i\right)\right) - \boldsymbol{\lambda}^T \mathbf{A}_i \mathbf{h}_i\left(\mathbf{p_F}_i\right). \tag{26}$$

Note that, with this modification, the combined maximization for $\boldsymbol{\mathcal{L}}(\mathbf{p_F}, \boldsymbol{\lambda})$ is decomposed into I independent maximizations, each corresponding to a different classifier chain (application). *This is the key property that leads to the design of a distributed algorithm.* The dual problem can now be decomposed into a master problem and I subproblems. The master problem involves solving for the optimal Lagrange multiplier $\boldsymbol{\lambda}^{\text{opt}}$ and is defined as

$$\boldsymbol{\lambda}^{\mathbf{opt}} = \arg\min_{\boldsymbol{\lambda} \geq 0} \left\{ \sum_{i=1}^{I} D_i(\boldsymbol{\lambda}) + \boldsymbol{\lambda}^T \boldsymbol{\mathcal{R}} \right\}. \tag{27}$$

The I independent subproblems[5] involve solving for the optimal operating points $\mathbf{p_F}_i, \forall i$, given the multiplier $\boldsymbol{\lambda}$, and may be written as:

$$D_i(\boldsymbol{\lambda}) = \max_{\mathbf{0} \leq \mathbf{p_F}_i \leq \mathbf{1}, \forall i} \boldsymbol{\mathcal{L}}_i\left(\mathbf{p_F}_i, \boldsymbol{\lambda}\right) \tag{28}$$

In the following sections we describe the distributed algorithm to solve these problems.

C. Solution to the Master Problem

Although the master problem in (27) is convex, the objective function may not always be differentiable. Hence, we cannot use traditional gradient-based algorithms to solve the problem. In order to deal with this non-differentiability, we need to introduce the theory of subdifferentials [18].

Definition 4: Subgradient: A vector $\mathbf{y} \in \mathbb{R}^M$ is labeled a subgradient of the convex function $f : \mathbb{R}^M \mapsto \mathbb{R}$ at $\mathbf{x} \in \mathbb{R}^M$ if

$$f(\mathbf{z}) \geq f(\mathbf{x}) + (\mathbf{z} - \mathbf{x})^T \mathbf{y}, \quad \forall \mathbf{z} \in \mathbb{R}^M. \tag{29}$$

Definition 5: Subdifferential: The set of all subgradients of the convex function f at $\mathbf{x} \in \mathbb{R}^M$ is called the subdifferential of f at $\mathbf{x}$, and is denoted as $\partial f(\mathbf{x})$.

It has been proven that the subgradient (subdifferential) has the following properties.

1) If f is differentiable at $\mathbf{x}$ with the gradient $\nabla f(\mathbf{x})$, then $\nabla f(\mathbf{x})$ is the unique subgradient at $\mathbf{x}$.
2) If a sequence $\{\mathbf{x}^n\}$ converges to $\mathbf{x}$ and $\mathbf{y}^n \in \partial f(\mathbf{x}^n)$ for all n, then the sequence $\{\mathbf{y}^n\}$ is bounded and each of its limit points is a subgradient of f at $\mathbf{x}$.

---

[5] Recent work [15], [27] in communication networks has used similar approaches for network resource allocations.

3) $\mathbf{x}$ maximizes the convex function f over a convex set $\mathbf{X} \subset \mathbb{R}^M$ if and only if there exists a subgradient $\mathbf{y} \in \partial f(\mathbf{x})$ such that $\mathbf{y}^T(\mathbf{z} - \mathbf{x}) \geq 0, \forall \mathbf{z} \in \mathbf{X}$.

Using this theory of subdifferentials, we re-examine the dual problem of (27).

Proposition 1: For a given $\boldsymbol{\lambda} \in \mathbb{R}^M$, suppose that $\mathbf{p}_{\mathbf{F}_i}(\boldsymbol{\lambda})$, $1 \leq i \leq I$ minimizes the subproblems in (28), then one of the subgradients of the dual function $D(\boldsymbol{\lambda})$ is given by

$$\mathbf{y}\left(\mathbf{P}_{\mathbf{F}}(\boldsymbol{\lambda})\right) = \boldsymbol{\mathcal{R}} - \sum_{i=1}^{I} \mathbf{A}_i \mathbf{h}_i\left(\mathbf{p}_{\mathbf{F}_i}(\boldsymbol{\lambda})\right). \tag{30}$$

Proof: The proof is fairly standard, and we omit it here due to space limitations.

We may now design a subgradient method based solution to solve the master problem. The subgradient method [18] based solution is similar to a gradient descent iterative algorithm, except that we use the subgradient instead of the gradient. Hence, at the $(j+1)$-th iteration, we update the multiplier $\boldsymbol{\lambda}$ as

$$\boldsymbol{\lambda}^{(j+1)} = \left[\boldsymbol{\lambda}^{(j)} - \alpha^{(j)} \mathbf{y}^{(j)}\right]^+ \tag{31}$$

where $\mathbf{y}^{(j)} = \mathbf{y}(\mathbf{P}_{\mathbf{F}}(\boldsymbol{\lambda}^{(j)}))$, $[\cdot]^+ = \max(\cdot, 0)$ denotes the projection of the vector onto the set $\{\boldsymbol{\lambda} \geq \mathbf{0}\}$, and $\alpha^{(j)}$ is a positive scalar step size. In order to ensure convergence of this iterative algorithm, the step size $\alpha^{(j)}$ needs to be selected with the following properties [18]:

$$\alpha^{(j)} \rightarrow 0, as \quad j \rightarrow \infty \quad and \sum_{j=0}^{\infty} \alpha^{(j)} = \infty. \tag{32}$$

An example of $\alpha^{(j)}$ that satisfies these properties is

$$\alpha^{(j)} = 1/j. \tag{33}$$

We use this step size sequence as it is simple and has been shown to work for a many types of problems. We do not compare against other sequences, but there has been some work on designing step size sequences to accelerate the convergence of the subgradient method [28]. Note that the multiplier $\boldsymbol{\lambda}^{(j)}$ may be interpreted as a cost incurred for resources at iteration j, with $\lambda_m^{(j)}$ being the cost of a unit resource on processing node m. It is clear that the update of the cost $\lambda_m^{(j)}$ for the node m is determined only by the operating points of classifiers located on that node. Hence, the combined update may be decomposed into M independent cost updates (one per processing node)

$$\lambda_m^{(j+1)} = \left[\lambda_m^{(j)} - \alpha^{(j)} \left(\mathcal{R}_m - \sum_{k \in \cup_i \mathcal{C}(i,m)} h_i^k\left(t_i^{k-1}\right)\right)\right]^+, \quad m \in \{1, \cdots, M\} \tag{34}$$

where $\mathcal{C}(i,m)$ represents the set of classifiers of chain i lying in the processing node m. This multiplier update per processing node clearly reduces any message exchange overhead among the processing nodes.

D. Solution to the Subproblem

As mentioned earlier, the subproblem may be defined for chain i as

$$\mathbf{p}_{\mathbf{F}_i}^{\text{opt}} = \arg\max_{\mathbf{p}_{\mathbf{F}_i}} \left\{\ln\left(u_i\left(\mathbf{p}_{\mathbf{F}_i}\right)\right) - \boldsymbol{\lambda}^T \mathbf{A}_i \mathbf{h}_i\left(\mathbf{p}_{\mathbf{F}_i}\right)\right\} \quad \text{subject to} \quad \mathbf{0} \leq \mathbf{p}_{\mathbf{F}_i} \leq \mathbf{1}. \tag{35}$$

This optimization may then be solved independently per classifier chain, and may be physically interpreted as maximizing the utility for chain i after deducting the cost incurred to acquire resources at different processing nodes. Note that the optimization for chain i requires information on resource cost only from those nodes on which one or more of its classifiers are located, i.e., the set of nodes $\mathcal{M}(i)$.

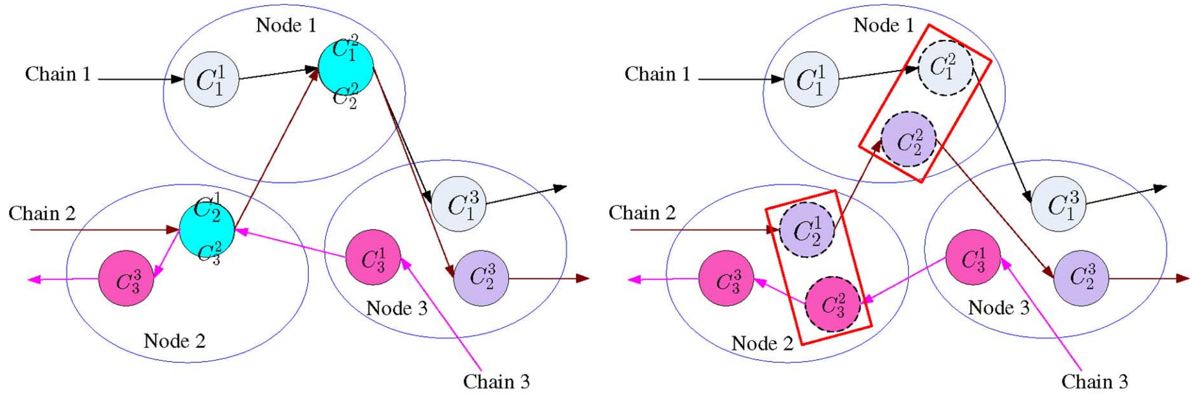
In order to solve each of these nonlinear differentiable optimization problems (defined in (35)), we again employ SQP [25]. We omit details here, given the discussion in Section V.

E. Combined Solution to the Dual Problem

We may combine the solution to the master problem and subproblems into one iterative solution that can be implemented in a distributed manner. In iteration j, each classifier chain first selects the operating point for all its classifiers simultaneously, by maximizing the net utility in (35). After operating point selection for all the chains, each processing node updates its cost per unit resource $\lambda_m^{(j)}$, as in (34). We summarize this iterative solution in Algorithm 1.

Algorithm 1 Combined Iterative Distributed Solution

```
Initialize $\lambda_m^{(0)}$ for $m = 1 \cdots M$
Set termination threshold $\epsilon$, and maximum number of iterations $J$
$\mathbf{u}^{(0)} \Leftarrow -\infty$
$j \Leftarrow 0$
repeat
    $j \Leftarrow j + 1$
    $i \Leftarrow 1$
    for $i = 1$ to $I$ do
        Identify resource price $\lambda_m^{(j)}$ for nodes on which chain $i$ classifiers are located.
        Solve the SQP to find the optimal operating point $\mathbf{p}_{\mathbf{F}_i}^{(j)}$.
        Determine throughput $t_i^{k-1}$ entering each classifier $C_i^k$.
    end for
    for $m = 1$ to $M$ do
        Update $\lambda_m^{(j)}$ using (34)
    end for
    $\mathbf{u}^{(j)} \Leftarrow \sum_{i=1}^{I} \ln(u_i(\mathbf{p}_{\mathbf{F}_i}^{(j)}))$
```



(a) (b)

Fig. 3. Decoupling chains with shared classifiers using virtual classifiers. (a) Chains with shared classifiers. (b) Shared → virtual classifiers.

```
until $(\mathbf{u}^{(j)} - \mathbf{u}^{(j-1)} \leq \boldsymbol{\epsilon}) \vee (j \geq J)$
return $\mathbf{p}_{\mathbf{F}_i}^{(j)}$, $j$, and $\mathbf{u}^{(j)} - \mathbf{u}^{(j-1)}$
```

VII. EXTENSION: NBS FOR CHAINS WITH SHARED CLASSIFIERS

In previous sections we assumed separate (disjoint) classifier chains for each application. However, this can be inefficient as it may lead to duplication of functionally identical classifiers across the system. Instead, it is much more efficient to reuse such classifiers across chains, e.g., if two different applications require data samples to be classified into speech and nonspeech, one single classifier may be used to handle data for both. In this section, we extend our designed classifier configuration algorithms to include arbitrary sharing of classifiers across chains. This allows us to consider much more general topologies of classifiers in a stream mining system.

In order to formulate the problem of configuring classifier chains with arbitrary classifier sharing, we define a sharing matrix $\mathbf{B}_i$ for each chain i. We first identify the number of *pairwise shared* classifiers, i.e., classifiers shared across each pair of chains. A classifier shared by G chains corresponds to ${}^{G}C_2$ pairwise shared virtual classifiers. Let the total number of pairwise shared classifiers in the system be E. We label the eth pairwise shared classifier as V^e. For each classifier, we also store the indices of the chains, across which the classifier is shared, using a vector $\mathbf{s}^e$. We determine E, V^e and $\mathbf{s}^e$ using Algorithm 2.

Algorithm 2 Sharing Matrix Construction

```
$E \Leftarrow 0$
for $i = 1$ to $I$ do
    for $k = 1$ to $N_i$ do
        for $i' = i + 1$ to $I$ do
            for $k' = 1$ to $N_{i'}$ do
                If $C_i^k = C_{i'}^{k'}$, $V^e \Leftarrow C_i^k$, $\mathbf{s}^e \Leftarrow [i\ i']$
                and $e \Leftarrow e + 1$
            end for
        end for
    end for
end for
Set $E = e$
return $E$, $V^e$ and $\mathbf{s}^e$.
```

Given these, the sharing matrix $\mathbf{B}_i$ with size $E \times N_i$ is defined as follows:

$$\mathbf{B}_i(e,k) = \begin{cases} 1, & \text{if } C_i^k = V^e \text{ and } i = \mathbf{s}^e(1) \\ -1, & \text{if} C_i^k = V^e \text{ and } i = \mathbf{s}^e(2) \\ 0, & \quad\text{otherwise.} \end{cases} \tag{36}$$

It is easy to verify that

$$[\mathbf{B}_1 \quad \cdots \quad \mathbf{B}_I]\mathbf{1} = \mathbf{0}. \tag{37}$$

An example of a topology with shared classifiers is illustrated in Fig. 3. For this chain, we have $E = 2$ and the sharing matrices $\mathbf{B}_1$, $\mathbf{B}_2$, and $\mathbf{B}_3$ are

$$\mathbf{B}_1 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$
$$\mathbf{B}_2 = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$
$$\mathbf{B}_3 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix}.$$

Using these defined sharing matrices, we can reformulate the optimization with an additional set of constraints on shared classifiers as

$$\begin{aligned} &\max_{\mathbf{p_F}} \sum_{i=1}^{I} \ln\left(u_i\left(\mathbf{p_F}_i\right)\right) \\ \text{subject to} \quad &\sum_{i=1}^{I} \mathbf{A}_i \mathbf{h}_i\left(\mathbf{p_F}_i\right) \leq \boldsymbol{\mathcal{R}} \\ &\sum_{i=1}^{I} \mathbf{B}_i \mathbf{p_F}_i = 0 \\ &0 \leq p_{F_i^k} \leq 1, \forall i, k. \end{aligned} \tag{38}$$

The key idea behind defining the formulation as above is the decomposition of each shared classifier into multiple *virtual* classifiers (each one corresponding to a separate chain that shares this classifier). Adding the constraint $\sum_{i=1}^{I}\mathbf{B}_i\mathbf{p}_{\mathbf{F}i}=0$ then ensures that all these *virtual* classifiers are configured to the same operating point (as they correspond to the same underlying classifier). We show an example of decomposing two shared classifier into four virtual classifiers in Fig. 3.

As before, we consider the dual problem for this optimization by introducing another Lagrange multiplier $\boldsymbol{\mu}=[\mu_1\ \cdots\ \mu_E]^T\in\mathbb{R}^E$, to capture the sharing constraint. The Lagrangian function is thus defined as

$$\boldsymbol{\mathcal{L}}(\mathbf{p_F},\boldsymbol{\lambda},\boldsymbol{\mu})=\sum_{i=1}^{I}\ln\left(u_i\left(\mathbf{p}_{\mathbf{F}i}\right)\right)-\boldsymbol{\lambda}^T\left(\sum_{i=1}^{I}\mathbf{A}_i\mathbf{h}_i\left(\mathbf{p}_{\mathbf{F}i}\right)-\boldsymbol{\mathcal{R}}\right)-\boldsymbol{\mu}^T\sum_{i=1}^{I}\mathbf{B}_i\mathbf{p}_{\mathbf{F}i} \quad (39)$$

and the dual problem may be written as

$$\min_{\boldsymbol{\lambda},\boldsymbol{\mu}} D(\boldsymbol{\lambda},\boldsymbol{\mu}), \quad \text{subject to } \boldsymbol{\lambda}\geq 0 \quad (40)$$

where $D(\boldsymbol{\lambda},\boldsymbol{\mu})$ is computed as

$$D(\boldsymbol{\lambda},\boldsymbol{\mu})=\max_{\mathbf{p_F}}\boldsymbol{\mathcal{L}}(\mathbf{p_F},\boldsymbol{\lambda},\boldsymbol{\mu}), \quad \text{subject to} \quad 0\leq {p_F}_i^k\leq 1,\forall i,k. \quad (41)$$

We can then decompose this into a master problem and a subproblem. The master problem involves determing the Lagrange multipliers and may be written as: $\boldsymbol{\lambda}^{\text{opt}},\boldsymbol{\mu}^{\text{opt}}$, which is given by

$$[\boldsymbol{\lambda}^{\text{opt}},\boldsymbol{\mu}^{\text{opt}}]=\arg\min_{\boldsymbol{\lambda}\geq 0,\boldsymbol{\mu}}\left\{\sum_{i=1}^{I}D_i(\boldsymbol{\lambda},\boldsymbol{\mu})+\boldsymbol{\lambda}^T\boldsymbol{\mathcal{R}}\right\}. \quad (42)$$

The subproblem involves configuring the classifiers given the multipliers, and may be written as

$$D_i(\boldsymbol{\lambda},\boldsymbol{\mu})=\max_{0\leq {p_F}_i^k\leq 1,\forall i,k}\boldsymbol{\mathcal{L}}_i\left(\mathbf{p}_{\mathbf{F}i},\boldsymbol{\lambda},\boldsymbol{\mu}\right) \quad (43)$$

where

$$\boldsymbol{\mathcal{L}}_i\left(\mathbf{p}_{\mathbf{F}i},\boldsymbol{\lambda},\boldsymbol{\mu}\right)=\ln\left(u_i\left(\mathbf{p}_{\mathbf{F}i}\right)\right)-\boldsymbol{\lambda}^T\mathbf{A}_i\mathbf{h}_i\left(\mathbf{p}_{\mathbf{F}i}\right)-\boldsymbol{\mu}^T\mathbf{B}_i\mathbf{p}_{\mathbf{F}i}. \quad (44)$$

Again, we can design an iterative distributed solution that may be summarized as follows. At iteration $j+1$, all chains first select their operating points for all their classifiers by maximizing (43) given a fixed $\boldsymbol{\lambda}^{(j)},\boldsymbol{\mu}^{(j)}$. After operating point selection, the servers update the cost per unit resource $\lambda_m^{(j)}(1\leq m\leq M)$ as

$$\lambda_m^{(j+1)}=\left[\lambda_m^{(j)}-\alpha^{(j)}\left(\mathcal{R}_m-\sum_{k\in\cup_i\mathcal{C}(i,m)}h_i^k\left({p_F}_i^k\right)\right)\right]^+ \quad (45)$$

where, $\mathcal{C}(i,m)$ and $\alpha^{(j)}$ are as defined earlier Each server then also updates the shared cost multiplier $\mu_e^{(j)}(1\leq e\leq E)$ as

$$\boldsymbol{\mu}_e^{(j+1)}=\left[\boldsymbol{\mu}_e^{(j)}-\beta^{(j)}\left(\sum_{i=1}^{I}\mathbf{B}_i\mathbf{p}_{\mathbf{F}i}\right)_e\right] \quad (46)$$

for all shared classifiers located on it.

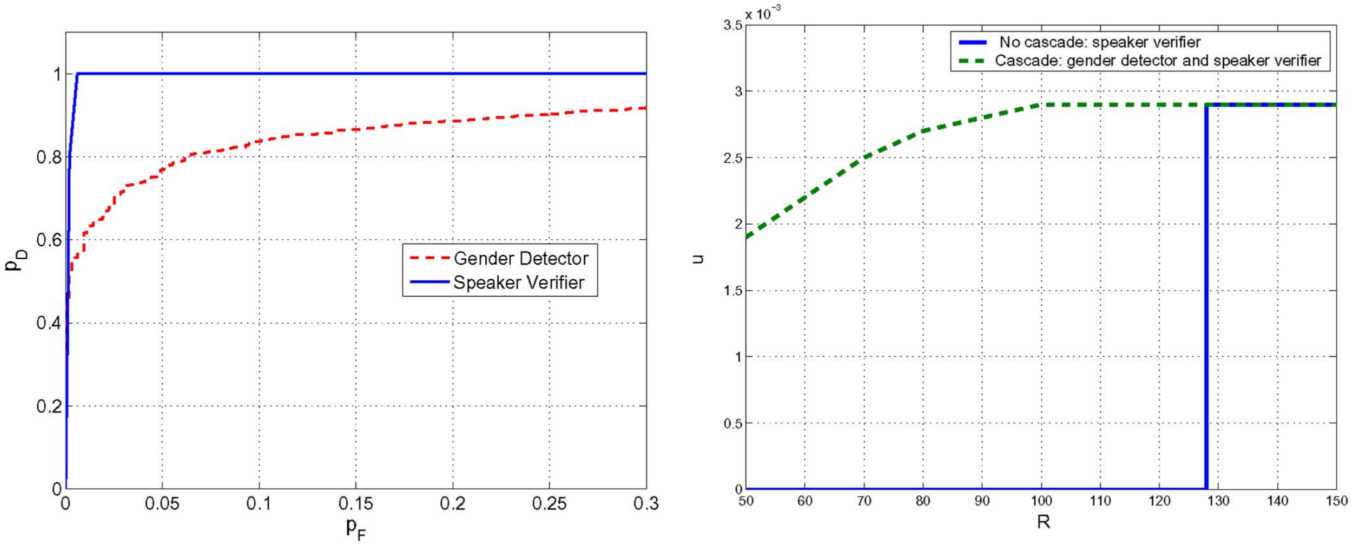
VIII. SIMULATION RESULTS

We present four different sets of results to highlight the performance of our classifier configuration algorithms given resource constraints. We first motivate the need for classifier cascades under resource constraints, for a real speaker verification application in Section VIII-A. We then examine the fairness achieved by NBS as compared with a MSUS. We present results for the centralized and distributed algorithms for an example topology with multiple competing classifier chains and discuss the convergence of these algorithms in Section VIII-C. Finally, we present results of the algorithms designed for topologies with arbitrary classifier sharing, in Section VIII-E. In all experiments, we use the SQP solver from Matlab [29] to generate results.

A. *Motivation for Classifier Cascade: Speaker Verification*

In this subsection, we demonstrate the use of a classifier cascade to benefit classification utility for a real speaker verification application, under resource constraints. We consider a speaker verification task, where we want to verify whether speech samples belong to a particular female speaker. We build a cascade of two classifiers for this task, with C^1 being a gender detector (i.e., forwards only female speech) and C^2 being the actual speaker verifier (tries to verify whether speech belongs to the specific speaker). These classifiers are trained on real Switchboard telephony data with 176 female speakers, and 106 male speakers using the IBM speaker verification system [30]. The complexities of the classifiers are fixed at $\rho^1=Q^1=8$ and $\rho^2=Q^2=128$ (equal to the number of Gaussians in the GMM assuming a unit resource consumption factor), and the corresponding DET curves are shown in Fig. 4.

We assume that the classifiers are located on one processing node, and we use $\Theta=0.1$ corresponding to the cost-tradeoff between misses and false alarms in NIST testing scenarios [20]. We vary the rate-dependent resource constraint $\mathcal{R}_1$ on the processing node from 50 to 150 and compare the performance of this cascade against using only the single classifier C^2 (speaker verifier) for this task. The utilities under different resource constraints for the cascade and the single classifier are shown in Fig. 4. As the resource constraint tightens, $(\mathcal{R}<128)$, the speaker verifier cannot process the incoming data by itself thereby leading to zero utility. As against this, the low-complexity gender detector in the cascade may be used to filter out irrelevant data samples early, thereby allowing the speaker verifier to process data even under tight resource constraints, thereby providing positive utility. Hence, under resource constraints, a classifier cascade (even with such independently trained classifiers), may be used to achieve better performance than a single classifier.



(a) (b)

Fig. 4. DET curves and comparison of classifier chain versus single classifier. (a) DET curves. (b) Single classifier versus cascade.

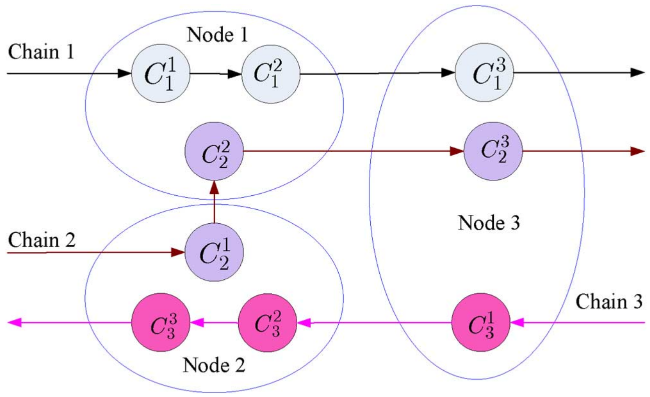


Fig. 5. Topology: three classifier chains, three processing nodes.

B. *Fairness Issues: Comparison of NBS With MSUS*

The MSUS is obtained as [6]

$$\mathbf{u}^{\mathrm{msus}} = \arg\max_{\mathbf{u}\in\boldsymbol{\mathcal{U}}} \sum_{i=1}^{I}(u_i). \tag{47}$$

We consider a topology with three classifier chains (each with three classifiers) located across three processing nodes (Fig. 5). The parameters for the chains, including the complexities of each classifier (ρ), the conditional probabilities (ϕ), the rate entering each chain (ω), and the desired end-to-end $p_D - p_F$ tradeoff (Θ) are listed in Table I. The DET curves for the classifiers are generated using a standard parametric form for a concave curve

$$(1-p_D)^{\alpha} + (p_F)^{\beta} = 1 \tag{48}$$

where α and β are the parameters of the curve that may be tuned to adjust its shape. The modeled DET curves for this example are shown in Fig. 6. The total rate-dependent resource constraint is set to $\mathcal{R} = 32$ on each of the three processing nodes. The NBS and MSUS are obtained using centralized SQP optimization, and the obtained utilities for all three chains are listed in Table II.

TABLE I
CLASSIFIER CHAIN PARAMETERS

	$\boldsymbol{\rho}$	$\boldsymbol{\phi}$	ω	Θ
Chain 1	$\begin{bmatrix} 8 & 32 & 64 \end{bmatrix}$	$\begin{bmatrix} 0.8 & 0.7 & 0.6 \end{bmatrix}$	1	0.5
Chain 2	$\begin{bmatrix} 8 & 32 & 64 \end{bmatrix}$	$\begin{bmatrix} 0.8 & 0.7 & 0.6 \end{bmatrix}$	1	1.0
Chain 3	$\begin{bmatrix} 8 & 32 & 64 \end{bmatrix}$	$\begin{bmatrix} 0.8 & 0.7 & 0.6 \end{bmatrix}$	1	5.0

TABLE II
UTILITY COMPARISON: NBS VERSUS MSUS

	Chain 1	Chain 2	Chain 3
NBS ($u_i\times1000$)	53.3	25.5	1
MSUS ($u_i\times1000$)	95.8	0.1	0

Also shown in Fig. 6 are the optimal operating points selected for each classifier by the two different optimizations.

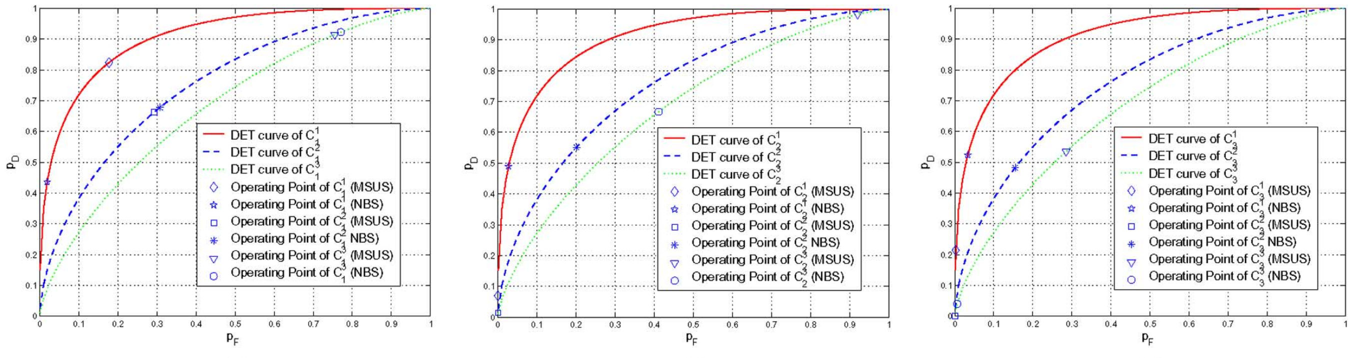
As is clear, with the MSUS, most of the utility is provided to Chain 1 (that has small Θ), with zero utility provided to Chain 3. Instead, with the NBS, all chains get nonzero utility, providing a greater level of fairness among the chains.

C. *Optimality and Convergence of the Distributed Algorithm for NBS*

In this section, we compare the obtained utilities after classifier configuration using the centralized optimization (solving the primal problem) with those obtained using the distributed optimization (solving the dual problem). We show that distributed algorithm does indeed converge to the centralized solution (both converge to the optimal solution), and we discuss the rate of convergence, as well as provide the intuition behind this convergence. For this simulation, we consider a more complicated topology of networked classifiers, shown in Fig. 7.

---

[6] As with the NBS, this discussion also holds for a weighted sum of utilities, given different priorities for the different chains.



(a) (b) (c)

Fig. 6. DET curves and classifier configuration NBS versus MSUS. (a) Chain 1. (b) Chain 2. (c) Chain 3.

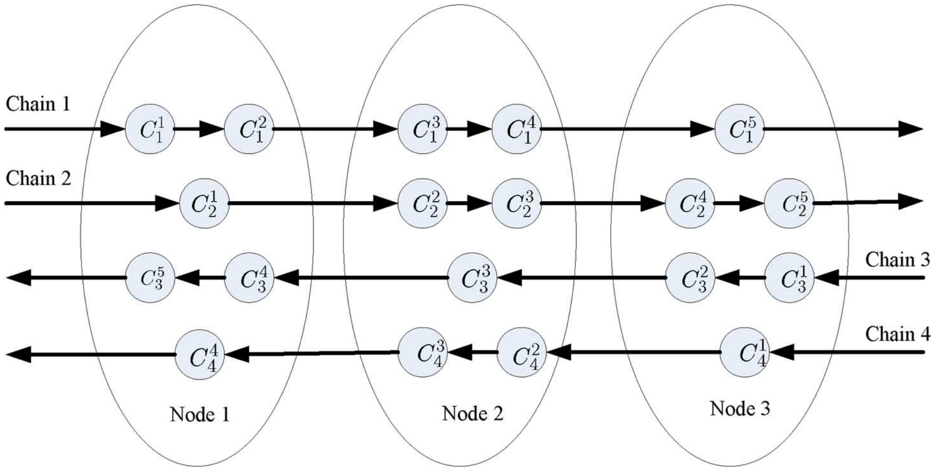
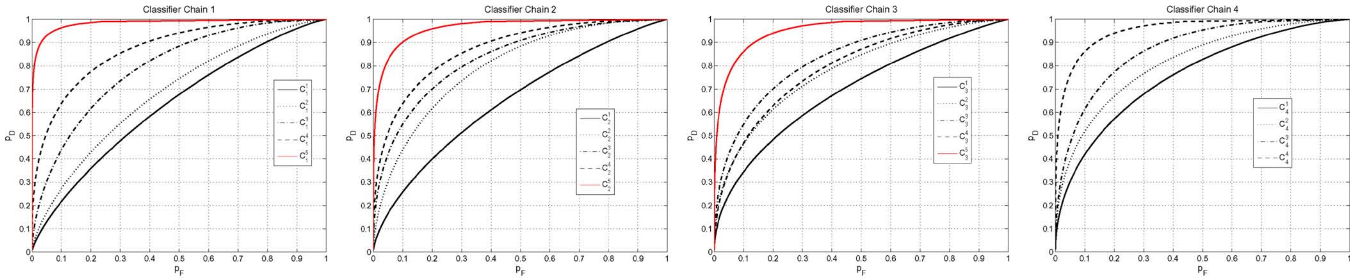


Fig. 7. Classifier topology: four chains, three processing nodes.

TABLE III
CLASSIFIER CHAIN PARAMETERS

	Chain 1	Chain 2	Chain 3	Chain 4
$\boldsymbol{\rho}$	$[8 \quad 10 \quad 15 \quad 20 \quad 32]$	$[10 \quad 15 \quad 18 \quad 28 \quad 36]$	$[12 \quad 16 \quad 18 \quad 30 \quad 32]$	$[15 \quad 20 \quad 24 \quad 30]$
$\boldsymbol{\phi}$	$[0.9 \quad 0.8 \quad 0.9 \quad 0.8 \quad 0.6]$	$[0.8 \quad 0.7 \quad 0.9 \quad 0.6 \quad 0.9]$	$[0.8 \quad 0.9 \quad 0.8 \quad 0.9 \quad 0.7]$	$[0.7 \quad 0.8 \quad 0.9 \quad 0.9]$



(a) (b) (c) (d)

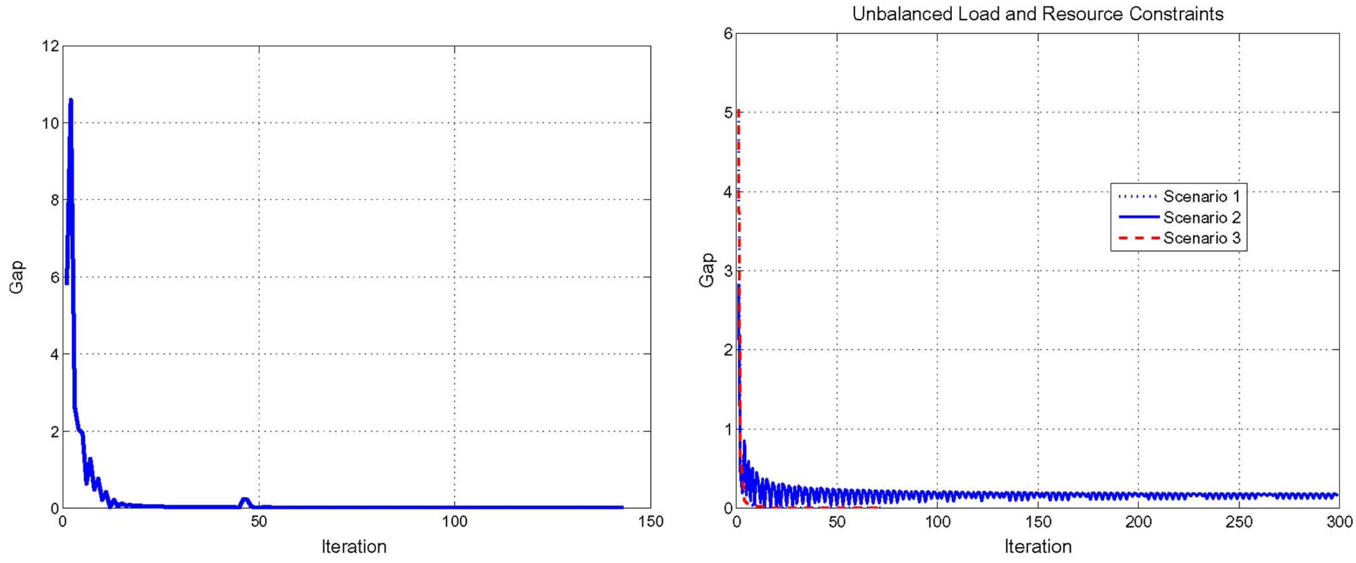
Fig. 8. DET curves for experiments. (a) Chain 1. (b) Chain 2. (c) Chain 3. (d) Chain 4.

The parameters associated with the classifiers (with $\Theta = 1$ for all chains) are shown in Table III and the DET curves are shown in Fig. 8.

The classifiers are distributed across three processing nodes with balanced available resources $\mathcal{R} = 36$ on each node, and balanced input rates $\boldsymbol{\omega} = [1\ 1\ 1\ 1]^T$ for all the chains. The combined log-utility (for the NBS) obtained by the centralized and distributed algorithms is the same, i.e., $ln(\mathbf{u}) = -11.5$, indicating that there is no duality gap. The utilities provided to the individual chains are also similar for both algorithms, and are listed in Table IV. We further verified that this is indeed the optimal solution.

TABLE IV
UTILITIES WITH SQP AND DISTRIBUTED ALGORITHM

	Chain 1	Chain 2	Chain 3	Chain 4
SQP ($u_i \times 1000$)	44.2	27.0	40.2	206.9
Distributed Algorithm ($u_i \times 1000$)	45.2	27.3	39.9	206.0



(a) (b)

Fig. 9. Utility gap between distributed and centralized algorithm: balanced and unbalanced scenarios. (a) Utility gap: balanced scenario. (b) Utility gap: unbalanced scenarios.

In order to highlight the convergence rate of the distributed algorithm, we also show the variation of the combined log-utility as a function of the iteration number in Fig. 9. In the figure, we plot $|ln(\mathbf{u}_P) - ln(\mathbf{u}_D^{(j)})|$, where $\mathbf{u}_P$ is the optimal utility (for the Primal problem) and $\mathbf{u}_D^{(j)}$ is the utility of the distributed algorithm (Dual problem) with j being the iteration number. It is clear that as $\lim_{j\to\infty} |ln(\mathbf{u}_D^{(j)}) - ln(\mathbf{u}_P)| = 0$. In fact, the convergence rate is rapid, with the gap approaching zero within the first 15 iterations.

We also repeated this experiment for unbalanced rates as well as unbalanced resource constraints under three scenarios, as shown in Table V. Scenario 1 has unbalanced input rates across the four chains, Scenario 2 has unbalanced resource constraints across the three processing nodes, and Scenario 3 has both unbalanced rates, as well as unbalanced resource constraints.

Again, we find that the distributed algorithm converges rapidly to the solution of the centralized algorithm, with the resulting gap as a function of the iteration number shown in Fig. 9. As we can see from the figure, for Scenario 1 and 3 the convergence happens within the first 15 iterations, while it is a little slower for Scenario 2. The resulting utilities for individual chains with the distributed and centralized solutions are also within 0.5% of each other.

D. Discussion: Duality Gap

As we can see from the results, the distributed and centralized algorithms converge to the same solution for a diverse sets of classifier parameters, unbalanced rates and constraints etc. In every such case, this identical solution necessarily implies convergence to the global optimum, and a duality gap of zero. Hence, for most practical scenarios that we encounter, we can easily verify the absence of this duality gap. In general, though, it is very involved to provide rigorous proofs on the convergence of the algorithms, and bounds on the duality gap, especially due to the non-concave utility function, and the non-convex set of constraints. While we do not provide a rigorous proof of the global convergence and the zero duality gap, we provide some intuition behind the observed experimental convergence of our algorithms. We believe that the absence of the duality gap is driven by the underlying geometry of the problems, and we refer to the discussion in [18] to explain this. Specifically, rewriting as the dual problem allows us to decompose our optimization into a master problem and a set of I separable subproblems (one per classifier chain). This separable form allows us to not only solve I simpler minimization problems, but also causes the duality gap to be relatively small, with the gap diminishing to zero relative to the optimal primal value, as I increases. This is because, in the case of separable problems the set of constraints-utility pair S may be written as a vector sum of I sets. Generally, a set S that is the vector sum of a large number of, possibly non-convex, roughly similar sets "tends to be convex", in the sense that any vector in the convex hull of S may be closely approximated by a vector in S. As a result, the duality gap tends to be relatively small and can often be shown to diminish to zero relative to the optimal primal value, as I increases. This discussion is based on a theorem by Shapley and Folkman, and it has been shown [31], that under some reasonable assumptions the duality gap is bounded by a non-negative scalar (depends on the structure of the underlying utility functions, and the constraint set). The above argument also indicates that these algorithms are likely to scale well, and may be used in large-scale real stream processing systems, where several such classifier chains compete for resources.

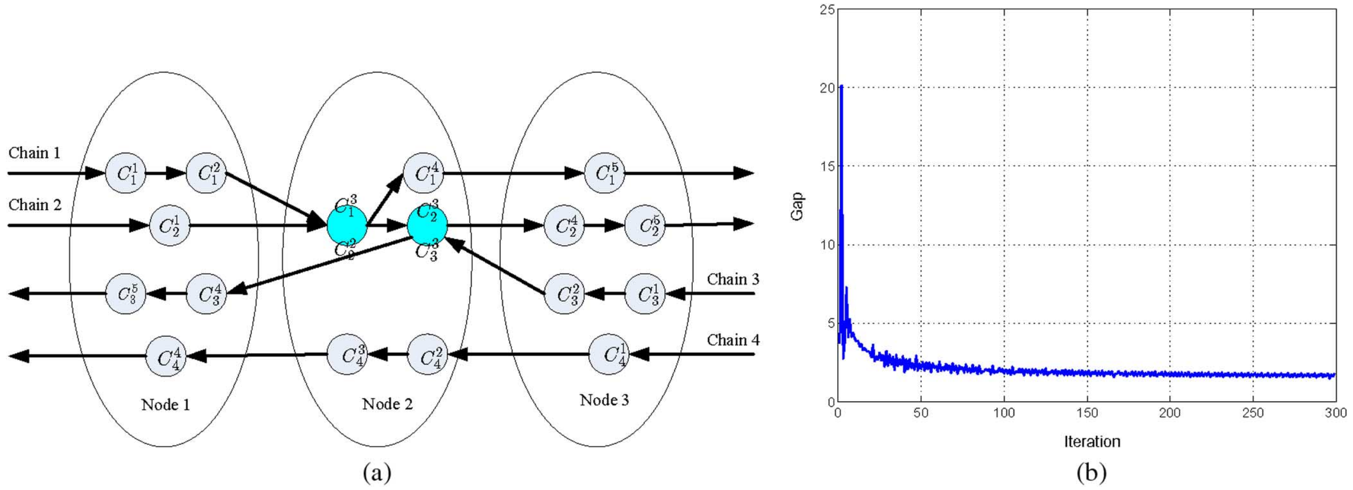
E. Results: Multiple Chains With Arbitrary Classifier Sharing

In this section, we present results of classifier configuration for a topology as shown in Fig. 10.

The DET curves and parameters for the classifiers in this case are selected as a subset of those in Fig. 8 and Table III. The resource constraints and the input rates are chosen as for the

TABLE V
SCENARIOS WITH UNBALANCED RATES AND RESOURCE CONSTRAINTS

	Scenario 1	Scenario 2	Scenario 3
Input Rates ω	$\begin{bmatrix} 1 & 0.5 & 0.5 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0.5 & 0.5 & 1 \end{bmatrix}$
Resource Constraint $\mathcal{R}$	$\begin{bmatrix} 36 & 36 & 36 \end{bmatrix}$	$\begin{bmatrix} 36 & 18 & 36 \end{bmatrix}$	$\begin{bmatrix} 36 & 18 & 36 \end{bmatrix}$



(a) (b)

Fig. 10. Topology and algorithm convergence: shared classifiers. (a) Topology: shared classifiers. (b) Algorithm convergence.

TABLE VI
UTILITIES WITH SQP AND DISTRIBUTED ALGORITHM: SHARED CLASSIFIERS

	Chain 1	Chain 2	Chain 3	Chain 4
SQP $(u_i \times 1000)$	24.3	41.2	9.9	227
Distributed Algorithm $(u_i \times 1000)$	43.2	26	40	205.6

Balanced Scenario in the previous Section. The utilities for the individual chains obtained using the centralized and distributed algorithms are shown in Table VI.

The convergence of the algorithm in terms of the gap $|ln(\mathbf{u}_P) - ln(\mathbf{u}_D^{(j)})|$, is shown in Fig. 10. While the distributed algorithm converges within 50 iterations, it does not converge to the optimal solution ($ln(\mathbf{u}_P) = -13$ and $ln(\mathbf{u}_D) = -11.58$). We believe that the additional sharing constraint causes the optimal solution to lie outside the MD-concave subregion, thereby not following our earlier intuition. A detailed investigation of these convergence issues for topologies with shared classifiers forms part of our future work.

IX. CONCLUSION

In this paper, design algorithms for the optimal configuration of competing chains of classifiers distributed across a resource constrained stream mining system. Specifically, we decompose queries into chains of binary *exclusive* classifiers, and define a unified performance metric for each chain by trading off the end-to-end probabilities of false alarm and detection (in terms of throughput and goodput). Furthermore, in order to design efficient as well as fair algorithms, we use NBS from game theory. We formulate the classifier configuration problem as a NOP, and design both centralized algorithms (using SQP), as well as distributed algorithms (using duality theory). We then consider arbitrary topologies of classifier chains, with classifier sharing across competing chains, and show that by decomposing each shared classifier into a set of virtual classifiers, we may directly extend our previous algorithms to design appropriate solutions (similar optimization with modified resource constraints).

We present several experimental results to highlight the performance of our algorithms. We first show the advantages of using a classifier cascade instead of a single classifier under resource constraints, for a speaker verification application with real telephony data. We then illustrate the fairness of the NBS as compared against a maximum sum of utility-based solution. We analyze the performance of our centralized and distributed algorithms on an application with four competing classifier chains, under both balanced as well as unbalanced rate and resource constraints. We show that the two algorithms converge to the same optimal solution, with no duality gap between them. While we do not provide a rigorous proof of the global convergence and the zero duality gap, we provide several intuitions behind the observed experimental convergence of our algorithms. Finally, we present results for scenarios with classifier sharing across chains. In this case, we observe a duality gap between the centralized and distributed algorithms (with overall utility difference of 12%), and we propose to investigate the causes for this as part of future work.

There are several other directions for future research. We would like to extend these ideas for arbitrary topologies of classifiers, going beyond topologies of classifier chains, and also consider the problem of constructing such topologies, i.e., ordering classifiers, distributing them across nodes, etc. Furthermore, we are interested in investigating the interaction of our approaches with load-shedding based approaches using a careful analysis of the underlying data characteristics, system resource constraints, and the nature of data queries. Specifically, the impact of shedding load at a point in the network (before classifier C^k) may be directly captured in terms of modifying the corresponding DET curve for this classifier, and as a result the throughput and goodput derived at the output of the classifier. The algorithms designed in this paper allow us to tune the classifier operating point (and also as a result the load-shedding strategy if incorporated into the classifier DET curve) in terms of maximizing the end-to-end utility. This goes significantly beyond the local measurements used when designing current load-shedding strategies. Finally, we are also examining the performance of these schemes in dynamically varying scenarios where the system resources, data characteristics, and classifier performance change with time.

REFERENCES

[1] M. Shah, J. Hellerstein, S. Chandrasekaran, and M. Franklin, "Flux: An adaptive partitioning operator for continuous query systems," in *Proc. IEEE Int. Conf. Data Engineering (ICDE)*, Bangalore, India, Mar. 2003.

[2] M. Cherniack, H. Balakrishnan, D. Carney, U. Cetintemel, Y. Xing, and S. Zdonik, "Scalable distributed stream processing," in *Proc. Conf. Innovative Data Systems Research (CIDR)*, Asilomar, CA, Jan. 2003.

[3] C. Olston, J. Jiang, and J. Widom, "Adaptive filters for continuous queries over distributed data streams," in *ACM SIGMOD 2003*, Jun. 2003, pp. 563–574.

[4] L. Amini, H. Andrade, F. Eskesen, R. King, Y. Park, P. Selo, and C. Venkatramani, The Stream Processing Core IBM T.J. Watson Research Center, Hawthorne, NY, Tech. Rep. RSC 23798, Nov. 2005.

[5] Y. R. Schapire, "A brief introduction to boosting," in *Proc. Int. Conf. Algorithmic Learning Theory*, Tokyo, Japan, Dec. 1999.

[6] A. Garg and V. Pavlovic, "Bayesian networks as ensemble of classifiers," in *Proc. IEEE Int. Conf. Pattern Recognition (ICPR)*, Quebec City, Canada, Aug. 2002.

[7] M. Gaber, A. Zaslavsky, and S. Krishnaswamy, "Resource-aware knowledge discovery in data streams," in *Proc. 1st Int. Workshop on Knowledge Discovery in Data Streams*, Sep. 2004.

[8] B. Babcock, S. Babu, M. Datar, and R. Motwani, "Chain: Operator scheduling for memory minimization in data stream systems," in *ACM SIGMOD*, San Diego, CA, Jun. 2003.

[9] S. Chandrasekaran, O. Cooper, A. Deshpande, M. Franklin, J. Hellerstein, W. Hong, S. Krishnamurthy, S. Madden, V. Raman, F. Reiss, and M. Shah, "TelegraphCQ: Continuous dataflow processing for an uncertain world," in *Proc. Conf. Innovative Data Systems Research (CIDR)*, Asilomar, CA, Jan. 2003.

[10] N. Tatbul, U. Cetintemel, S. Zdonik, M. Cherniack, and M. Stonebraker, "Load shedding in a data stream manager," in *Proc. 29th Int. Conf. Very Large Data Bases (VLDB)*, Berlin, Germany, Sep. 2003.

[11] Y. Chi, P. S. Yu, H. Wang, and R. R. Muntz, "Loadstar: A load shedding scheme for classifying data streams," in *Proc. IEEE Int. Conf. Data Mining (ICDM)*, Brighton, U.K., Nov. 2004.

[12] A. Jain, E. Y. Chang, and Y.-F. Wang, "Adaptive stream resource management using kalman filters," in *Proc. ACM SIGMOD Int. Conf. Management of Data*, Paris, France, 2004.

[13] S. H. Low and D. E. Lapsley, "Optimization flow control, I: Basic algorithm and convergence," *IEEE/ACM Trans. Netw.*, vol. 7, no. 6, pp. 861–874, Dec. 1999.

[14] D. Palomar and M. Chiang, "On alternative decompositions and distributed algorithms for network utility problems," in *Proc. IEEE Globecom*, Anaheim, CA, Mar. 2005.

[15] H. Yaiche, R. Mazumdar, and C. Rosenberg, "A game-theoretic framework for rate allocation and charging of elastic connections in broadband networks," *IEEE/ACM Trans. Netw.*, vol. 8, no. 5, pp. 667–678, Oct. 2000.

[16] K. Binmore, *Fun and Games, a Text on Game Theory*. Lexington, MA: D.C. Health, 1992.

[17] F. P. Kelly, A. K. Maullo, and D. K. H. Tan, "Rate control in communication networks: Shadow prices, proportional fairness, and stability," *J. Oper. Res. Soc.*, vol. 49, no. 3, pp. 237–252, Mar. 1998.

[18] D. P. Bertsekas, *Nonlinear Programming*, 2nd ed. Belmont, MA: Athena Scientific, 1999.

[19] R. Duda, P. Hart, and G. Stork, *Pattern Classification*. New York: Wiley, 2001.

[20] A. F. Martin and M. A. Przybocki, The NIST Speaker Recognition Evaluations: 1996–2001 [Online]. Available: http://www.nist.gov/speech/publications/papersrc/odyssey_paper1.pdf 2001

[21] T. Menzies, J. Greenwald, and A. Frank, "Data mining static code attributes to learn defect predictors," *IEEE Trans. Software Eng.*, vol. 33, no. 1, pp. 2–13, Jan. 2007.

[22] M. Campos and B. Milenova, "Creation and deployment of data mining-based intrusion detection systems in Oracle database 10 g," in *Proc. IEEE Int. Conf. Machine Learning (ICML)*, Vancouver, BC, Canada, Dec. 2005.

[23] R. Brause, T. Langsdorf, and M. Hepp, "Neural data mining for credit card fraud detection," in *IEEE Int. Conf. Tools with Artificial Intelligence (ICTAI)*, Hong Kong, China, Nov. 1999.

[24] D. S. Turaga, O. Verscheure, U. V. Chaudhari, and L. Amini, "Resource management for networked classifiers in distributed stream mining systems," in *Proc. IEEE Int. Conf. Data Mining (ICDM)*, Hong Kong, China, Dec. 2006.

[25] P. T. Boggs and J. W. Tolle, "Sequential quadratic programming," *Acta Numerica*, 1996.

[26] M. J. D. Powell, "A AST algorithm for nonlinearly constrained optimization calculations," in *Numerical Analysis*, G. A. Watson, Ed. Berlin, Germany: Springer Verlag, 1978, vol. 630, Lecture Notes in Mathematics.

[27] J. W. Lee, R. R. Mazumdar, and N. B. Shroff, "Non-convex optimization and rate control for multi-class services in the internet," *IEEE/ACM Trans. Netw.*, vol. 13, no. 4, pp. 827–840, Aug. 2005.

[28] X. Zhao, P. Luh, and J. Wang, "Surrogate gradient algorithm for Lagrangian relaxation," *J. Opt. Theory Applicat.*, vol. 100, no. 3, pp. 699–712, Mar. 1999.

[29] Matlab Optimization Toolbox [Online]. Available: http://www.mathworks.com/access/helpdesk/help/toolbox/optim/

[30] G. N. Ramaswamy, A. Navratil, U. V. Chaudhari, and R. D. Zilca, "The IBM system for the NIST 2002 cellular speaker verification evaluation," in *Proc. IEEE ICASSP*, Hong Kong, China, Apr. 2003.

[31] D. Bertsekas, *Constrained Optimization and Lagrange Multiplier Methods*. Belmont, MA: Athena Scientific, 1997.

Fangwen Fu received the Bachelor's and Master's degrees from Tsinghua University, Beijing, China, in 2002 and 2005, respectively. He is currently working toward the Ph.D. degree in the Department of Electrical Engineering, University of California, Los Angeles (UCLA).

During the summer of 2006, he was an intern at IBM T. J. Watson Research Center, Hawthorne, NY. His research interests include wireless multimedia streaming, resource management for networks and systems, applied game theory, and video processing and analysis.

Deepak S. Turaga (M'01) received the B.Tech. degree in electrical engineering from the Indian Institute of Technology, Bombay, in 1997 and the M.S. and Ph.D. degrees in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, PA, in 1999 and 2001, respectively.

He is currently a Research Staff Member in the Exploratory Stream Processing department at the IBM T. J. Watson Research Center, Hawthorne, NY. He was with Philips Research during 2001–2003 and with Sony Electronics in 2003–2004. His research interests lie primarily in statistical signal processing, multimedia processing, coding and streaming, machine learning, and computer vision applications. In these areas, he has published over 40 journal and conference papers and one book and one book chapter. He has filed over 20 invention disclosures and has participated actively in MPEG standardization activities.

Dr. Turaga received the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY 2006 Transactions Best Paper Award (with M. van der Schaar and B. Pesquet-Popescu), and is a coauthor for the 2006 IEEE ICASSP Best Student Paper (with H. Tseng, O. Verscheure, and U. Chaudhari). He is an Associate Editor of the IEEE TRANSACTIONS ON MULTIMEDIA, an Associate Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, and an Associate Editor of the *Hindawi Journal for the Advances in Multimedia*.

Olivier Verscheure received the B.S. degree from the Faculte Polytechnique de Mons, Belgium, in 1994, the M.Sc. degree from the Swiss Federal Institute of Technology, Lausanne (EPFL), both in electrical engineering, in 1995, and the

Ph.D. degree in computer science from the Institute for Computer Communications and Applications, EPFL, in June 1999.

He has been a member of the research staff at the IBM T. J. Watson Research Center, Hawthorne, NY, since July 1999. His research lies within the areas of scalable media servers, multimedia compression and streaming, data mining, and signal processing.

Dr. Versheure is a coauthor of the 2004 IEEE International Performance Computing and Communications Conference Best Paper Award (with L. Amini and G. Paleologo) and a coauthor for the 2006 IEEE ICASSP Best Student Paper (with H. Tseng, D. Turaga, and U. Chaudhari).

Mihaela van der Schaar (SM'04) received the M.S. and Ph.D. degrees from Eindhoven University of Technology, Eindhoven, The Netherlands, in 1996 and 2001, respectively.

Prior to joining the Electrical Engineering Department, University of California, Los Angeles (UCLA), in July 2005, she was a Senior Researcher at Philips Research, The Netherlands and U.S., between 1996 and June 2003, where she led a team of researchers working on multimedia coding, processing, networking, and streaming algorithms and architectures. From January to September 2003, she was also an Adjunct Assistant Professor at Columbia University, New York. From July 2003 until July 2005, she was an Assistant Professor in the Electrical and Computer Engineering Department, University of California at Davis. She has published extensively on multimedia communications, networking, architectures, systems, compression, and processing, and holds 30 granted U.S. patents and several more pending. Since 1999, she has been an active participant in the ISO Motion Picture Expert Group (MPEG) standard, to which she has made more than 50 contributions and for which she has received three ISO recognition awards.

Dr. van der Schaar has Chaired the *ad-hoc* group on MPEG-21 Scalable Video Coding for three years and Co-Chaired the MPEG *ad-hoc* group on Multimedia Test-beds. She was a Guest Editor of the *EURASIP Special Issue on Multimedia over IP and Wireless Networks* and General Chair of the Picture Coding Symposium 2004, the oldest conference on image/video coding. She was elected as a Member of the Technical Committee on Multimedia Signal Processing, as well as the Technical Committee on Image and Multiple Dimensional Signal Processing of the IEEE Signal Processing Society (SPS). She was an Associate Editor of the IEEE TRANSACTIONS ON MULTIMEDIA and the *SPIE Electronic Imaging Journal* from 2002 to 2005. Currently, she is an Associate Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, of the IEEE SIGNAL PROCESSING LETTERS, and of the newly founded IEEE SPS e-Newsletter.

Lisa Amini received the Ph.D. degree in computer science from Columbia University, New York, NY.

She manages the Exploratory Stream Processing Research Group at the IBM T. J. Watson Research Center, Hawthorne, NY, and has worked at IBM in the areas of distributed systems, networking, and multimedia for over 15 years. Her research interests include stream mining, scalable messaging, network overlays for intelligent content distribution, data stream management, and computing co-operatives. She has been and continues to be a member of the Program Committee for several top-tier conferences in these areas.