Foresighted Tree Configuration Games in Resource Constrained Distributed Stream Mining Sensors[☆]

Hyunggon Park[a,*], Deepak S. Turaga[b], Olivier Verscheure[b], Mihaela van der Schaar[c]

[a]*Department of Electronics Engineering, Ewha Womans University, Seoul, Korea*
[b]*IBM T. J. Watson Research Center, Hawthorne, NY, USA*
[c]*Electrical Engineering Department, University of California, Los Angeles, CA, USA*

Abstract

We consider the problem of optimizing stream mining applications that are constructed as tree topologies of classifiers and deployed on a set of resource constrained and distributed processing nodes (or sensors). The optimization involves selecting appropriate false-alarm detection tradeoffs (operating points) for each classifier to minimize an end-to-end misclassification penalty, while satisfying resource constraints. We design distributed solutions, by defining tree configuration games, where individual classifiers configure themselves to maximize an appropriate local utility. We define the local utility functions and determine the information that needs to be exchanged across classifiers in order to design the distributed solutions. We analytically show that there is a unique pure strategy Nash equilibrium in operating points, which guarantees convergence of the proposed approach. We develop both *myopic* strategy, where the utility is purely local to the current classifier, and *foresighted* strategy, where the utility includes impact of classifier's actions on successive classifiers. We analytically show that actions determined based on foresighted strategies improve the end-to-end performance of the classifier tree, by deriving an associated probability bound. We also investigate the impact of resource constraints on the classifier action selections for each strategy, and the corresponding application performance. We propose a learning-based approach, which enables each classifier to effectively adapt to the dynamic changes of resource constraints. We evaluate the performance of our solutions on an application for sports scene classification. We show that foresighted strategies result in better performance than myopic strategies in both resource unconstrained and resource constrained scenarios, and asymptotically approach the centralized optimal solution. We also show that the proposed distributed solutions outperform the centralized solution based on the Sequential Quadratic Programming on average in resource unconstrained scenarios.

Keywords: Resource constrained stream mining sensors, tree configuration games, myopic and foresighted strategies, coalitions for distributed classifiers, binary classifier trees.

[☆]This work was done while the first author was with IBM T. J. Watson Research Center, Hawthorne, NY, USA.

[*]Corresponding author

Email addresses: hyunggon.park@ewha.ac.kr (Hyunggon Park), deepak@us.ibm.com (Deepak S. Turaga), ov1@us.ibm.com (Olivier Verscheure), mihaela@ee.ucla.edu (Mihaela van der Schaar)

1. Introduction

There are an increasing number of applications that require processing and classification of continuous, high volume data streams. These include photo and video streaming services, online analysis of financial data streams, real-time manufacturing process control, search engines, spam filters, security, and medical services [1, 2, 3, 4, 5], etc. These applications often consists of processing topologies of distributed operators (or sensors) [6, 7, 8, 9, 10] deployed on large-scale stream mining systems in an ad hoc manner [9, 11, 12, 13, 14]. Specifically, many stream mining applications implement topologies (ensembles such as trees or cascades) of low-complexity binary classifier node sensors to hierarchically filter the data streams and jointly accomplish the task of complex classification [2, 15]. Using such tree topology of classifiers, data of disinterest as well as low confidence data can be filtered out (discarded), thereby potentially increasing end-to-end accuracy, and decreasing processing load on downstream classifiers.

Stream mining applications pose several interesting research challenges. These include the optimal construction and training of such topologies, as well as configuration and management of individual classifiers to maximize end-to-end performance – especially under dynamically varying resource constraints and data characteristics. In this paper, we focus on the classifier configuration problem for tree topologies, i.e., determining an operating point from Detection Error Tradeoff (DET) curves for each binary classifier in the tree, in order to maximize the end-to-end classification performance [16]. In the considered tree topologies with binary classifiers, individual classifiers can operate at different performance levels by selecting a tradeoff between probability of detection and probability of false alarm, i.e., operating point. For example, operating points can be selected by determining *decision thresholds* for likelihood ratio tests or for support vector machine (SVM) scores [3, 17], etc. Prior approaches model classifier tree configuration as an optimization problem and use centralized techniques such as Sequential Quadratic Programming (SQP) [16] to solve it. Such approaches require centralized control of all the classifiers, and the corresponding information and data. Hence, they suffer disadvantages in terms of having a single central point of control and associated failure, issues with scaling and adaptation as the topology grows, and not allowing large scale applications with capabilities distributed across multiple proprietary entities.

The main motivation of this work is to provide an analytical model to investigate the interaction among the classifiers in a stream mining application, and to design completely distributed solutions that can stabilize the application by enabling each classifier to adhere its selected operating point. Due to the informationally distributed nature, each classifier may not know how many classifiers are present or connected to it, or it may have only limited information about the operating points, objective functions (i.e., utility functions), available resources, etc., of the other classifiers. In this case, each classifier can decide an operating point that maximizes its local objective (i.e., local utility) determined based on limited available information. To capture this situation, we define a game-theoretic model, which is referred to as a *tree configuration game*. The tree configuration game is a class of non-cooperative sequential game [18], and it enables individual classifiers to find an equilibrium that achieves a desirable end-to-end application utility. We design an appropriate local utility function based on the end-to-end misclassification penalty, and minimal amount of local information exchange across classifiers. We then describe how individual classifiers can select an optimal action to maximize their local utilities. We analytically show that there is a unique pure strategy Nash equilibrium in selected actions for this game, and evaluate the application performance at this equilibrium.

In the case where additional information at one or more classifiers is available, we show that using the information also lead to another unique pure strategy Nash equilibrium, which can further improve the end-to-end application performance. Specifically, if a classifier has access to information about some of its successive classifiers (e.g., available actions, local utility functions, etc.), it can select an action that maximizes its *coalition* utility. The coalition of a classifier includes the successive classifiers for which information is available to the classifier. This action selection strategy is referred to as *foresighted*. This is unlike the *myopic* strategy, where a classifier selects actions to maximize its own local utility without considering the impact of its action selections on the other classifiers. We show that a foresighted strategy can outperform the myopic strategy, and analytically derive an associated probability bound on the performance.

We also consider the impact of resource constraints on the designed solutions, i.e., action selections for myopic and foresighted strategies. We show analytically that the optimal myopic actions do not change in the presence of resource constraints, although we may need to use load shedding to satisfy the constraints[1]. In contrast, the optimal foresighted actions are influenced by both the actions of other classifiers and system resource division policies. We propose a learning based approach, where a foresighted classifier iteratively estimates the impact of its actions on the coalition utility without requiring additional information exchange, and uses the learning to dynamically select its optimal action.

We present simulation results on an application for hierarchical semantic concept detection in sports images, and evaluate the performance against prior centralized approaches in [16]. We examine the impact of varying misclassification cost coefficients as well as granularity of the decision space (i.e., the available quantized DET curve per classifier) under both resource constrained and unconstrained scenarios. We demonstrate that the foresighted strategy outperforms the myopic strategy, and also show that the foresighted strategies lead to utility that approaches the centralized optimal solution, as the coalition size and the number of actions increase. An actual implementation of the proposed myopic strategy based on IBM System S processing core middleware [7] was demonstrated in [22].

This paper is organized as follows. In Section 2, we introduce the model for individual classifiers and classifier trees. In Section 3, we introduce the tree configuration game, including the utility function definition, available actions, etc. In Section 4, we describe the myopic and foresighted strategies for classifier configuration. We provide detailed analysis of the foresighted strategy and provide probabilistic bounds on performance improvements over the myopic strategy in Section 5. In Section 6, we study the impact of the resource constraints on the proposed tree configuration games, and the corresponding application utilities. Simulation results are presented in Section 7. We conclude with directions for future research in Section 8. The notation used in this paper is summarized in Table 1.

2. System Model – Hierarchical Binary Classifier Trees

We consider a stream mining system, which consists of several binary classifiers in a tree topology. An illustrative stream mining application [16] is depicted in Fig. 1. The topology of classifiers in this example is used to identify semantic concepts from sports image data using hierarchical filtering. Leaf classifiers in the application (e.g., classifier 4, 8, 9, etc.) represent

[1]In this paper, we use a naive load shedding approach, where stream data is randomly shed at the input of each classifier (i.e., due to buffer overflow). Advanced load-shedding approaches based on measures of burst, desired Quality of Service (QoS) requirements, data value or delay constraints [19, 20, 21] may also be used with our approaches.

Table 1: Summary of Notation

Notation	Description	Notation	Description
c_i ($\bar{c}_i$)	yes (no) classification unit (CU) in classifier i	π_i	strategy of c_i
C_i	classifier i, $C_i = \{c_i, \bar{c}_i\}$	U_i	utility of c_i
$\mathbf{G}_i$	coalition of c_i (set of CUs)	λ_i^F (λ_i^M)	cost coefficient for false alarm (missing) of c_i
a_i	action of c_i, $a_i = (p_i^F, p_i^D)$	ϕ_i	a-priori probability of stream to c_i
$\mathbf{A}_i$	action set of c_i	t_i (g_i)	throughput (goodput) of input stream to C_i
I_i	local information of c_i	$\mathbf{I}_{+i}$	information about successive CUs of c_i
$\mathbf{I}_i$	available set of information to c_i	$\mathbf{a}_{-i}$	set of actions for precedent CUs of c_i
$k \rightsquigarrow i$	c_k is a preceding CU of c_i	$i \rightsquigarrow^d k$	c_k is a successive CU with distance d from c_i
$\hat{x}$	x for c_i, or $\bar{x}$ for $\bar{c}_i$	$\Lambda_i \triangleq t_r\phi_i \cdot \Phi_i$	true fraction of stream data that belongs to c_i
$\mathbf{C}_L(\mathbf{G})$	set of leaf CUs in coalition $\mathbf{G}$		

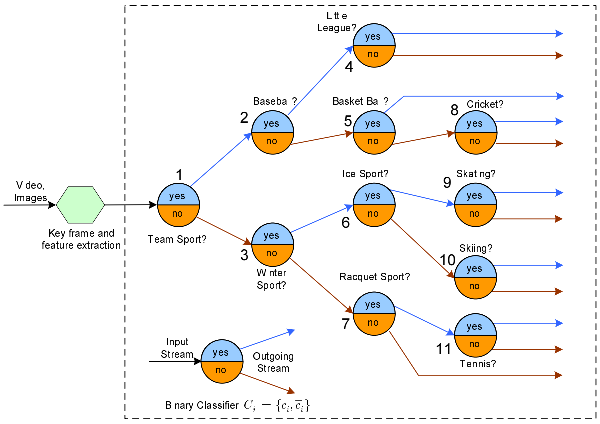


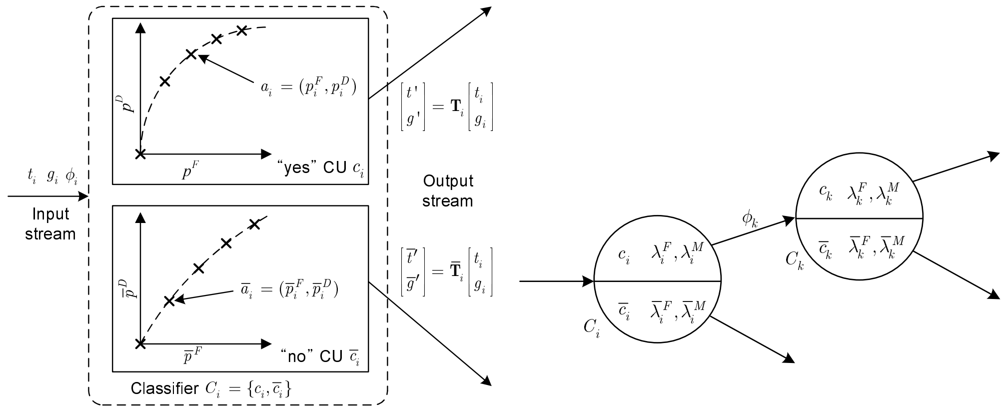
Figure 1: An illustrative stream mining application.

the actual classes of interest, while intermediate classifiers (e.g., classifier 2, 3, 6, etc.) assist in hierarchical filtering of data based on a semantic hierarchy of concepts. We now describe our model for classifiers, classifier trees, and end-to-end application utility.

2.1. Configuration of Binary Classifier

A binary classifier filters input data into two classes, a "yes" class (denoted by $\mathcal{H}_y$) and a "no" class (denoted by $\mathcal{H}_n$). Each classifier C_i may be modeled as two independent classification units (CUs)[2], corresponding to the "yes" CU c_i and "no" CU $\bar{c}_i$, i.e., $C_i = \{c_i, \bar{c}_i\}$. An illustration for C_i is depicted in Fig. 2(a). We use notation $i \rightsquigarrow k$ to denote that c_i (or $\bar{c}_i$) is a *preceding* CU for c_k (or $\bar{c}_k$). Equivalently, c_k (or $\bar{c}_k$) is a *successive* CU of c_i (or $\bar{c}_i$). The information about distance d, in number of hops, between two CUs is denoted by $i \rightsquigarrow^d k$. For example, in Fig. 1, c_1 is a preceding CU of c_2, which is denoted by $c_1 \rightsquigarrow c_2$ (or $c_1 \rightsquigarrow^1 c_2$). Note that the topology allows disambiguation between the two CUs per classifier.

[2]Independent CUs allow data to be labeled as *both* "yes" and "no", or neither.



(a) Input-output rates in a classifier.

(b) Misclassification cost coefficients.

Figure 2: Classifier configuration.

2.2. *Stream Characteristics*

The input stream for classifier C_i is characterized by *throughput* t_i and *goodput* g_i, which represent total data rate and correctly labeled data rate, respectively. The average fraction of the input stream data that represents true positive for "yes" CU c_i is denoted by ϕ_i (see Fig. 2(b)). Note that ϕ_i is pre-determined based on the classifier topology and stream data characteristics. $\bar{\phi}_i$ is similarly defined for "no" CU $\bar{c}_i$. Note that $\bar{\phi}_i = 1 - \phi_i$.

2.3. *Performance Measure of Classifier*

For a "yes" CU c_i, we label the probability of detection as p_i^D, the probability of miss as p_i^M ($p_i^M = 1 - p_i^D$), and the probability of false alarm as p_i^F. Similarly, for a "no" CU $\bar{c}_i$, we label the probability of detection as $\bar{p}_i^D$, the probability of miss as $\bar{p}_i^M$, and the probability of false as $\bar{p}_i^F$. As in [16], performance of c_i ($\bar{c}_i$) is controlled by its tradeoff between probability of false alarm p_i^F ($\bar{p}_i^F$) and probability of detection p_i^D ($\bar{p}_i^D$). The two CUs are independent and may have decoupled operating points, e.g., through the use of independent thresholds (one for "yes" and one for "no") for score-based classifiers. The set of operating points (p_i^F, p_i^D) (or $(\bar{p}_i^F, \bar{p}_i^D)$) represents the DET curve for c_i (or $\bar{c}_i$). The DET curve is a non-decreasing concave function that lies between $(0, 0)$ and $(1, 1)$.

2.4. *Cost Coefficients for Misclassification*

Misclassification cost coefficients λ_i^F ($\bar{\lambda}_i^F$) and λ_i^M ($\bar{\lambda}_i^M$) represent the cost/penalty per unit data rate of *false alarm* and *miss* for CU c_i ($\bar{c}_i$). These coefficients are specified by the application for leaf classifiers, since they represent the classes of interest. As will be discussed in Section 2.5 and Section 2.6, these coefficients are used to specify the end-to-end application utility (or equivalently, application cost). For the proposed approaches, we define the misclassification cost coefficients at intermediate classifiers, by decomposing the end-to-end utility into local utilities. We determine cost coefficients at intermediate classifiers, by explicitly considering the tree topology as well as the stream characteristics, and using backward propagation from the leaf coefficients. Specifically, as shown in Fig. 2(b), misclassification cost coefficients λ_i^F and λ_i^M of an intermediate CU c_i can be obtained by

$$\lambda_i^F = \phi_k \lambda_k^F + (1-\phi_k)\bar{\lambda}_k^F, \text{ and } \lambda_i^M = \phi_k \lambda_k^M + (1-\phi_k)\bar{\lambda}_k^M, \tag{1}$$

where C_k is an immediately successive classifier of c_i, i.e., $i \rightsquigarrow^1 k$.

2.5. Input and Output Rates

A CU's operating point controls output stream rates. For a stream with throughput t_i and goodput g_i entering c_i (and $\bar{c}_i$), the output rates t_i' and g_i' (correspondingly $\bar{t}_i'$ and $\bar{g}_i'$) may be derived as [16]

$$\begin{bmatrix} t_i' \\ g_i' \end{bmatrix} = \mathbf{T}_i \begin{bmatrix} t_i \\ g_i \end{bmatrix}, \text{and } \begin{bmatrix} \bar{t}_i' \\ \bar{g}_i' \end{bmatrix} = \bar{\mathbf{T}}_i \begin{bmatrix} t_i \\ g_i \end{bmatrix}, \tag{2}$$

where $\mathbf{T}_i$ and $\bar{\mathbf{T}}_i$ are given by

$$\mathbf{T}_i = \begin{bmatrix} p_i^F & \phi_i(p_i^D - p_i^F) \\ 0 & \phi_i p_i^D \end{bmatrix}, \text{ and } \bar{\mathbf{T}}_i = \begin{bmatrix} \bar{p}_i^D & \phi_i(\bar{p}_i^F - \bar{p}_i^D) \\ 0 & \bar{\phi}_i \bar{p}_i^D, \end{bmatrix}.$$

2.6. End-to-End Application Utility

Let $\mathbf{S}$ represent the set of all classifiers and $\mathbf{C}_L(\mathbf{S})$ be the set of leaf CUs in the application. Then, the end-to-end cost/penalty for misclassification incurred by leaf classifier $c_l \,(\in \mathbf{C}_L(\mathbf{S}))$ may be expressed as

$$(t_l' - g_l')\lambda_l^F + \left(\Lambda_l - g_l'\right)\lambda_l^M,$$

where Λ_l represents a true fraction of stream data that belongs to c_l for input stream rate t_r to the tree, defined as $\Lambda_l \triangleq t_r\phi_l \cdot \prod_{\forall k \in \{j | j \rightsquigarrow l\}} \hat{\phi}_k = t_r\phi_l \cdot \Phi_l$, with $\hat{\phi}_k = \phi_k$ for c_k and $\hat{\phi}_k = \bar{\phi}_k$ for $\bar{c}_k$. We define the utility achieved by c_l as the negative end-to-end cost, or

$$U_l = -\left[(t_l' - g_l')\lambda_l^F + \left(\Lambda_l - g_l'\right)\lambda_l^M\right]. \tag{3}$$

Similarly, $\bar{U}_l$ achieved by $\bar{c}_l$ can be expressed in terms of coefficients $\bar{\lambda}_l^F$ and $\bar{\lambda}_l^M$, i.e.,

$$\bar{U}_l = -\left[(\bar{t}_l' - \bar{g}_l')\bar{\lambda}_l^F + \left(\bar{\Lambda}_l - \bar{g}_l'\right)\bar{\lambda}_l^M\right], \tag{4}$$

where $\bar{\Lambda}_l$ may be defined similarly. Thus, the end-to-end application utility $U_{\mathbf{S}}$ may be expressed as

$$U_{\mathbf{S}} = \sum\nolimits_{c_l \in \mathbf{C}_L(\mathbf{S})} U_l + \sum\nolimits_{\bar{c}_l \in \mathbf{C}_L(\mathbf{S})} \bar{U}_l. \tag{5}$$

3. Tree Configuration Games

The goal of this paper is the optimized selection of operating points for individual CUs in order to maximize the end-to-end application utility (or equivalently, minimizing the end-to-end application cost). In this section, we define a tree configuration game, which is a class of non-cooperative sequential game [18]. The game is played by each classifier that selects an operating point to maximize its own local utility based on the available information. The game is assumed to progress from the root to the leaf, i.e., the root classifier configures itself before its children, and so on. We formally define the tree configuration game in the following sections.

3.1. *Definition of Tree Configuration Games*

For a stream mining application $\mathbf{S}$ with N classifiers, the tree configuration game consists of:

- a finite set of CUs (players): $\{C_i = (c_i, \bar{c}_i) | 1 \le i \le N\}$,
- for each CU $\hat{c}_i$, a nonempty set of actions: $\hat{\mathbf{A}}_i$,
- for each CU $\hat{c}_i$, a utility function: $\hat{U}_i(\cdot)$, and
- for each CU $\hat{c}_i$, a strategy: $\hat{\pi}_i(\cdot)$.

We use $\hat{x}$ to represent x and $\bar{x}$ as appropriate. For example, $\hat{c}_i$ represents c_i for "yes" CU and $\bar{c}_i$ for "no" CU, respectively. This notation is used in the rest of the paper. Each element of the tree configuration games is described in more detail.

3.1.1. *Action Set* $\hat{\mathbf{A}}_i$

The action set $\hat{\mathbf{A}}_i$ for CU $\hat{c}_i$ is defined as

$$\hat{\mathbf{A}}_i = \{\hat{a}_{i,k} | 1 \le k \le \hat{A}_i\}, \tag{6}$$

where $\hat{a}_{i,k} = (\hat{p}^F_{ik}, \hat{p}^D_{ik})$ is an action that involves selecting the kth operating point among $\hat{A}_i = |\hat{\mathbf{A}}_i|$ available operating points. This assumes a discrete set of operating points, which are typically obtained by empirical measurement. These operating points may be viewed as quantized versions of an underlying continuous DET curve – a differentiable concave function $\hat{f} : [0, 1] \to [0, 1]$, defined as [16]:

$$\hat{p}^D = \hat{f}(\hat{p}^F),$$

for $0 \le \hat{p}^F \le 1$. In this paper, we assume uniformly distributed (result of uniform quantization) operating points; alternative quantization schemes can also be used. Hence, action $\hat{a}_{i,k}$ corresponds to selecting operating point $\left(\frac{k-1}{\hat{A}_i - 1}, \hat{f}_i\left(\frac{k-1}{\hat{A}_i - 1}\right)\right)$, which is illustrated in Fig. 2(a).

3.1.2. *Local Utility Function* $\hat{U}_i(\cdot)$

Similar to (3), the local utility function for an intermediate CU c_i ($c_i \notin \mathbf{C}_L(\mathbf{S})$) can be expressed as

$$\hat{U}_i(\hat{a}_i) = -\left[(\hat{t}'_i - \hat{g}'_i)\hat{\lambda}^F_i + \left(\hat{\Lambda}_i - \hat{g}'_i\right)\hat{\lambda}^M_i\right], \tag{7}$$

where $\hat{\lambda}^F_i$ and $\hat{\lambda}^M_i$ are derived by the backward propagation scheme shown in (1). We assume that the backward propagation is performed at initialization, before the classifier configuration. Note that each CU needs to exchange information in order to identify cost coefficients and the local utility function in (7). The minimum information that needs to be exchanged across CUs is discussed in Section 4.

3.1.3. *Strategy* $\hat{\pi}_i$ *of CU* $\hat{c}_i$

A strategy of a CU determines its optimal action in the action set. This is determined as a function of the available information. Hence, the strategy may be expressed as

$$\hat{a}^*_i = \hat{\pi}_i(\hat{\mathbf{I}}_i) \in \hat{\mathbf{A}}_i, \tag{8}$$

where $\hat{\mathbf{I}}_i$ represents the available information to $\hat{c}_i$. Various types of available information will be discussed in Section 3.2, and their impact on the strategies will be studied in Section 4.

3.2. Available Information for CUs

In the tree configuration games, CU $\hat{c}_i$ can have various types of available information:

- *local* information $\hat{I}_i$: information specific to $\hat{c}_i$ such as observed throughput, available actions, misclassification cost coefficients, etc. Hence,

$$\hat{I}_i = \{\hat{t}_i, \hat{\phi}_i, \hat{\lambda}_i^F, \hat{\lambda}_i^M, \hat{\mathbf{A}}_i\}. \tag{9}$$

  The information about the available resources $\hat{R}_i$ can also be included. We assume that the local information is always available to each CU.

- information about *successive* classifiers $\hat{\mathbf{I}}_{+i}$: $\hat{\mathbf{I}}_{+i}$ includes local information of successive classifiers. Hence,

$$\hat{\mathbf{I}}_{+i} = \{\hat{I}_k | i \rightsquigarrow k, \forall k\}. \tag{10}$$

  Note that $\hat{\mathbf{I}}_{+i}$ excludes $\hat{I}_i$.

- information about *preceding* CU actions $\hat{\mathbf{a}}_{-i}$: The set of actions $\hat{\mathbf{a}}_{-i}$ taken by preceding CUs of $\hat{c}_i$ is expressed as

$$\hat{\mathbf{a}}_{-i} = \{\hat{a}_k | k \rightsquigarrow i, \forall k\}. \tag{11}$$

The impact of *what* information and *how much* information on the performance of the application is investigated in the next section. For simplicity, we only consider a "yes" CU c_i in the analysis, which also holds for a "no" CU $\bar{c}_i$. We explicitly discuss any differences for the two types of CUs.

4. Information Availability and Action Selection Strategies

In this section, we investigate the impact of the available information on the strategies, and the corresponding application utility. We assume first that there are unlimited resources, i.e. no resource constraints.

4.1. Myopic Strategy

A strategy π_i for CU c_i can determine an optimal action a_i^*, such that

$$a_i^* = \pi_i(\mathbf{I}_i) = \arg\max_{a_i \in \mathbf{A}_i} U_i(a_i) = \arg\min_{a_i \in \mathbf{A}_i} \left[(t_i' - g_i')\lambda_i^F + (\Lambda_i - g_i')\,\lambda_i^M \right]. \tag{12}$$

The strategy π_i in (12) determines an action that maximizes c_i's own local utility. However, it does *not* consider the impact of the selected action on the utilities of its successive classifiers, and the corresponding end-to-end application utility. Hence, π_i is referred to as *myopic* strategy.

Note that local information I_i is insufficient to solve the optimization problem shown in (12). Thus, information exchange across multiple classifiers is needed to determine $\mathbf{I}_i$ that enables c_i to find its optimal action. Proposition 1 identifies the minimum information needs to be additionally exchanged across CUs for the myopic strategies.

Proposition 1. For a CU c_i, g_i and $\Phi_i = \prod_{\forall k \in \{j | j \rightsquigarrow i\}} \hat{\phi}_k$ is the minimum additional information required to correctly determine an optimal action based on the myopic strategy.

Algorithm 1 Determining an Optimal Action of c_i based on Myopic Strategy

Given: Information $\mathbf{I}_i = \{I_i = \{t_i, \phi_i, \lambda_i^F, \lambda_i^M, \mathbf{A}_i\}, g_i, \Phi_i\}$, and DET characteristic function f_i.

1: **Compute an operating point** $x_i^* = p_i^F \in \mathbf{A}_i$

Compute x_i^* such that $\frac{\partial f_i(x_i^*)}{\partial x_i} = \frac{(t_i - g_i\phi_i)\lambda_i^F}{g_i\phi_i\lambda_i^M}$ `// or equivalently,` $\frac{\partial U_i(x_i)}{\partial x_i} = 0$

2: **Find the best action in** $\mathbf{A}_i$

If $x_i^* > 1$ or $x_i^* < 0$, **then**

$a_i^* = \arg\max_{a_i \in \{a_{i,1}, a_{i,A_i}\}} [U_i(a_i)]$ `// only need to consider two extreme actions`

else

$a_i^* = \arg\max_{a_i \in \{a_{i,(l_R-1)}, a_{i,l_R}\}} [U_i(a_i)]$ `// only need to consider two nearest actions`

Note a_{i,l_R} (includes $p_{il_R}^F$) is determined such that $p_{i(l_R-1)}^F < x_i^* \leq p_{il_R}^F$.

Proof. Without loss of generality, we consider a "yes" CU c_i, which has its local information $I_i = \{t_i, \phi_i, \lambda_i^F, \lambda_i^M, \mathbf{A}_i\}$. As shown in the input-output relationship of c_i in (2), t_i' and g_i' can be expressed as

$$\begin{bmatrix} t_i' \\ g_i' \end{bmatrix} = \begin{bmatrix} p_i^F & \phi_i(p_i^D - p_i^F) \\ 0 & \phi_i p_i^D \end{bmatrix} \begin{bmatrix} t_i \\ g_i \end{bmatrix}. \tag{13}$$

Thus, information g_i, in addition to I_i, is sufficient to specify t_i' and g_i'. Moreover, since Λ_i is defined as $\Lambda_i = t_r\phi_i \cdot \prod_{\forall k \in \{j | j \rightsquigarrow i\}} \hat{\phi}_k$, Φ_i is sufficient to specify Λ_i.

Based on Proposition 1, we can conclude that the information $\mathbf{I}_i$ required for myopic strategy in (12) is $\mathbf{I}_i = \{I_i, g_i, \Phi_i\}$. Hence, we require that information g_i and Φ_i are always forwarded with outgoing stream data. Note that Proposition 1 implies that both the size of messages and the number of messages that needs to be exchanged are constant.

Since the number of available actions is finite (i.e., $A_i < \infty$), a CU c_i can always determine a unique action based on strategy $\pi_i(\mathbf{I}_i)$ by comparing utilities achieved by different actions. This may require a high search complexity when the number of available actions is large. However, if the underlying continuous DET curve of the CU can be estimated (i.e., function f_i), then using the concavity of the local utility function (see Appendix), the optimal action can be efficiently determined with fixed computational complexity independent of the action set size[3]. An algorithm for finding the optimal action for c_i, assuming the underlying DET curve is available or can be estimated, is presented in Algorithm 1. This algorithm requires the comparison of only two candidate actions to determine the optimal action. Similarly, an optimal action of a "no" CU $\bar{c}_i$ can also be efficiently and uniquely determined.

We now show that the myopic strategy leads to a unique pure strategy Nash equilibrium.

Proposition 2. If each CU determines its action based on the myopic strategy, then there exists a unique pure strategy Nash equilibrium in operating points.

Proof. Let $c_i \in C_i$ be a "yes" CU, and it deploys the myopic strategy π_i given in (12) with available information $\mathbf{I}_i = \{I_i, g_i, \Phi_i\}$. Since the local utility of c_i is determined based on its own

[3]Note that the underlying continuous DET curve can be estimated as the convex hull of available actions [23] and the optimal solution can be found using Lagrangian optimization technique [24]. This approach is commonly used in several optimization problems such as rate-distortion optimization in video compression [25].

action $a_i \in \mathbf{A}_i$ as well as the set of actions $\mathbf{a}_{-i}$ of its preceding CUs, the local utility function defined in (7) can be explicitly expressed as $U_i(a_i, \mathbf{a}_{-i})$. Since $\mathbf{a}_{-i}$ is already determined and fixed, we have

$$U_i(a_i^*, \mathbf{a}_{-i}) \geq U_i(a_i, \mathbf{a}_{-i}), \tag{14}$$

for all actions $a_i(\neq a_i^*) \in \mathbf{A}_i$. Similarly, the same argument holds for a "no" CU $\bar{c}_i$, i.e.,

$$\bar{U}_i(\bar{a}_i^*, \bar{\mathbf{a}}_{-i}) \geq \bar{U}_i(\bar{a}_i, \bar{\mathbf{a}}_{-i}), \tag{15}$$

for all actions $\bar{a}_i(\neq \bar{a}_i^*) \in \bar{\mathbf{A}}_i$. Since each CU can uniquely determine its optimal action based on the myopic strategy, there exists one pure strategy Nash equilibrium in operating points.

The result drawn from Proposition 2 is consistent with the fact that a pure strategy Nash equilibrium always exists in finite extensive form games [26, 27]. Proposition 2 implies that the myopic strategy guarantees convergence. Importantly, the convergence time increases only *linearly with the tree depth*. Note that the performance achieved at this Nash equilibrium can be improved by utilizing additional information, as discussed next.

4.2. Foresighted Strategy for Coalitions

If c_i has information $\mathbf{I}_{+i}$ about its successive classifiers, then it can select an action while explicitly considering its impact on the utility of the successive classifiers. Specifically, CU c_i with $\mathbf{I}_{+i}$ can form a *coalition* with its successive classifiers, defined as follows.

Definition 1 (Coalition of a CU). *A coalition $\mathbf{G}_i$ of a CU c_i is a set of successive classifiers for which the information is available to c_i*[4].

Then, the CU can select an action that maximizes the coalition utility based on a *foresighted* strategy. Note that the classifiers in a coalition do *not* simultaneously determine their actions. Rather, the foresighted CU decides its action, while considering the actions that will be sequentially determined by its successive classifiers in its coalition, such that the jointly determined actions lead to the maximum coalition utility. Therefore, this is still a non-cooperative sequential game, where a local utility of a foresighted CU is its coalition utility. Specifically, the foresighted strategy $\pi_i(\mathbf{I}_i)$ of c_i with information $\mathbf{I}_i = \{I_i, \mathbf{I}_{+i}\}$ determines an action:

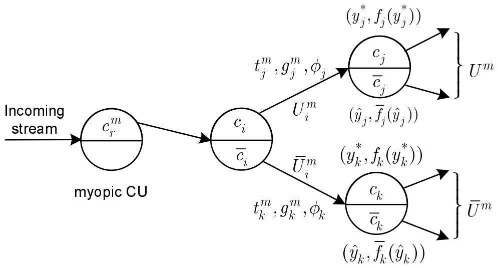
$$a_i^* = \pi_i(\mathbf{I}_i) = \arg\max_{a_i \in \mathbf{A}_i} U_{\mathbf{G}_i}(a_i, \{(a_k, \bar{a}_k) | C_k \in \mathbf{G}_i\}), \tag{16}$$

where $U_{\mathbf{G}_i}$ denotes the utility achieved by $\mathbf{G}_i$, i.e., the sum of utilities derived by the *leaf* CUs in coalition $\mathbf{G}_i$. A CU $\hat{c}_k \in \mathbf{G}_i$ is a leaf CU of coalition $\mathbf{G}_i$, if

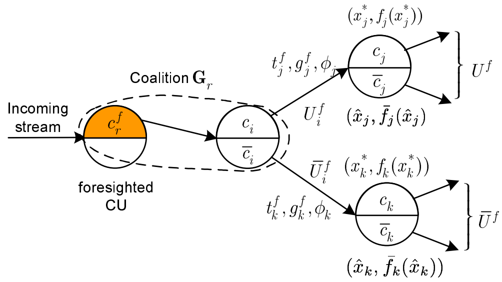
$$\left\{\hat{c}_h | k \rightsquigarrow^1 h \text{ for all } \hat{c}_h \in \mathbf{S}\right\} \notin \mathbf{G}_i,$$

i.e., none of its children are part of $\mathbf{G}_i$. We denote the set of leaf CUs in coalition $\mathbf{G}_i$ by $\mathbf{C}_L(\mathbf{G}_i)$. For example, in Fig. 3(b), leaf CUs in coalition $\mathbf{G}_r$ are c_i and $\bar{c}_i$, and thus, $\mathbf{C}_L(\mathbf{G}_r) = \{c_i, \bar{c}_i\}$. The coalition utility of $\mathbf{G}_r$ is given by $U_{\mathbf{G}_r} = U_i^f + \bar{U}_i^f$. As an extreme case, if $C_1 = \{c_1, \bar{c}_1\}$ is the root classifier of an application and it has the information about all of its successive classifiers, then the coalition utility represents the application utility defined in (5), i.e., $U_{\mathbf{G}_1} + U_{\bar{\mathbf{G}}_1} = U_{\mathbf{S}}$. By definition, if information about more successive classifiers is available to a CU, it can form a larger coalition. Note that $U_{\mathbf{G}_i}$ depends on the action a_i of c_i as well as the actions of coalition

[4] Similarly, a coalition of $\bar{c}_i$ is denoted by $\bar{\mathbf{G}}_i$.



(a) An illustration of myopic CU c_r^m



(b) An illustration of foresighted CU c_r^f

Figure 3: Illustrative myopic and foresighted CUs.

members (i.e., $\{(a_k, \bar{a}_k)|C_k \in \mathbf{G}_i\}$). However, c_i decides its action a_i, based on the assumption that $\{(a_k, \bar{a}_k)|C_k \in \mathbf{G}_i\}$ are myopically determined. This keeps the solution completely distributed, and allows the action a_i to be uniquely determined with low complexity. Hence, a foresighted action of a CU only *indirectly* controls the action selections of successive classifiers. Since the foresighted action can be uniquely determined by (16), and the corresponding actions of successive CUs are also uniquely determined, an application that consists of several foresighted CUs will also converge to a unique pure strategy Nash equilibrium.

5. Foresighted Strategies for Coalitions and Performance Improvement

In this section, we analytically show that foresighted strategies for coalitions formed by intermediate CUs may result in improved end-to-end application utility. As before, we focus on a scenario with no resource constraints. In the following analysis, we consider an elementary sub-tree shown in Fig. 3. In this example, the end-to-end application utility is determined by $U^m + \bar{U}^m$ for the myopic case in Fig. 3(a) and $U^f + \bar{U}^f$ for the foresighted case in Fig. 3(b).

Our approach is as follows. We first analytically express the utilities achieved by the myopic strategy $U^m + \bar{U}^m$ and the foresighted strategy $U^f + \bar{U}^f$ in terms of known parameters. Then, we derive a lower bound for the probability that $U^f + \bar{U}^f \geq U^m + \bar{U}^m$, i.e., when foresighted strategy outperforms myopic strategies. Finally, we show that the lower bound increases as the coalition utility based on the foresighted strategy improves.

Proposition 3. Foresighted strategies can improve the application utility with a higher probability bound.

Proof. Consider the elementary sub-tree shown in Fig. 3. In Fig. 3(a), c_r^m determines its action based on the myopic strategy, while c_r^f decides its action based on the foresighted strategy for its coalition $\mathbf{G}_r = \{c_r^f, C_i\} = \{c_r^f, c_i, \bar{c}_i\}$ in Fig. 3(b).

A foresighted action selected by c_r^f guarantees $U_i^f + \bar{U}_i^f \geq U_i^m + \bar{U}_i^m$, because the myopic strategy is a special case of the foresighted strategy. This may be decomposed as

$$U_i^f - U_i^m \geq \Delta u_i, \text{ and } \bar{U}_i^f - \bar{U}_i^m \geq -\Delta \bar{u}_i, \tag{17}$$

where $\Delta u_i - \Delta\bar{u}_i \geq 0$ for $\Delta u_i \geq 0$. Based on the input-output relationship given in (2), the conditions in (17) can be expressed as

$$\Delta t/\Delta g \leq (\lambda_i^F + \lambda_i^M)/\lambda_i^F - \Delta u_i/(\lambda_i^F \Delta g) \triangleq R_A, \tag{18}$$

$$\Delta\bar{t}/\Delta\bar{g} \leq (\bar{\lambda}_i^F + \bar{\lambda}_i^M)/\bar{\lambda}_i^F + \Delta\bar{u}_i/(\bar{\lambda}_i^F \Delta\bar{g}) \triangleq R_B, \tag{19}$$

where $\Delta t \triangleq t_j^f - t_j^m$, $\Delta g \triangleq g_j^f - g_j^m$, $\Delta\bar{t} \triangleq t_k^f - t_k^m$, and $\Delta\bar{g} \triangleq g_k^f - g_k^m$, respectively. Without loss of generality, we assume that $\Delta g > 0$ and $\Delta\bar{g} > 0$[5].

Now, we derive conditions on when the foresighted strategy of c_r^f leads to a higher end-to-end utility, i.e., when we have $U^f + \bar{U}^f > U^m + \bar{U}^m$. Consider the contradictory case, i.e., $U^m + \bar{U}^m \geq U^f + \bar{U}^f$. This condition can be decomposed as,

$$U^m - U^f \geq \Delta u, \tag{20}$$

$$\bar{U}^m - \bar{U}^f \geq -\Delta u. \tag{21}$$

Based on the input-output relationship in (2), the condition in (20) becomes

$$\begin{aligned}&\left[\{x_j^*(t_j^f - g_j^f\phi_j)\lambda_j^F - g_j^f\phi_j f_j(x_j^*)\lambda_j^M\} + \{(\bar{f}_j(\hat{x}_j)(t_j^f - g_j^f) + \phi_j\hat{x}_j g_j^f)\bar{\lambda}_j^F - g_j^f\bar{\phi}_j\bar{f}_j(\hat{x}_j)\bar{\lambda}_j^M\}\right] \\ &- \left[\{y_j^*(t_j^m - g_j^m\phi_j)\lambda_j^F - g_j^m\phi_j f_j(y_j^*)\lambda_j^M\} + \{(\bar{f}_j(\hat{y}_j)(t_j^m - g_j^m) + \phi_j\hat{y}_j g_j^m)\bar{\lambda}_j^F - g_j^m\bar{\phi}_j\bar{f}_j(\hat{y}_j)\bar{\lambda}_j^M\}\right] \geq \Delta u.\end{aligned} \tag{22}$$

Since action y_j^* (or $\hat{y}_j$) instead of x_j^* (or $\hat{x}_j$) incurs higher costs for c_j (or $\bar{c}_j$), the condition in (22) can be expressed as

$$y_j^*(\Delta t - \Delta g\phi_j)\lambda_j^F - f_j(y_j^*)\Delta g\phi_j\lambda_j^M + (\bar{f}_j(\hat{y}_j)(\Delta t - \Delta g) + \phi_j\hat{y}_j\Delta g)\bar{\lambda}_j^F - \Delta g\bar{\phi}_j\bar{f}_j(\hat{y}_j)\bar{\lambda}_j^M \geq \Delta u,$$

which leads to:

$$\Delta t/\Delta g \geq Q_j^N/Q_j^D + \Delta u/(\Delta g Q_j^D) \triangleq R_C, \tag{23}$$

where $Q_j^N = y_j^*\phi_j\lambda_j^F + f_j(y_j^*)\phi_j\lambda_j^M + \bar{f}_j(\hat{y}_j)(\bar{\lambda}_j^F + \bar{\phi}_j\bar{\lambda}_j^M) - \phi_j\hat{y}_j\bar{\lambda}_j^F$ and $Q_j^D = y_j^*\lambda_j^F + \bar{f}_j(\hat{y}_j)\bar{\lambda}_j^F$. Similarly, the condition in (21) can be expressed as

$$\Delta\bar{t}/\Delta\bar{g} \geq Q_k^N/Q_k^D - \Delta u/(\Delta\bar{g} Q_k^D) \triangleq R_D, \tag{24}$$

where $Q_k^N = y_k^*\phi_k\lambda_k^F + f_k(y_k^*)\phi_k\lambda_k^M + \bar{f}_k(\hat{y}_k)(\bar{\lambda}_k^F + \bar{\phi}_k\bar{\lambda}_k^M) - \phi_k\hat{y}_k\bar{\lambda}_k^F$ and $Q_k^D = y_k^*\lambda_k^F + \bar{f}_k(\hat{y}_k)\bar{\lambda}_k^F$. Based on (18), (19), (23) and (24), we can conclude that:

$$U^m + \bar{U}^m \geq U^f + \bar{U}^f \Rightarrow R_C \leq \Delta t/\Delta g \leq R_A, \text{ and } R_D \leq \Delta\bar{t}/\Delta\bar{g} \leq R_B. \tag{25}$$

Finally, (25) leads to a lower bound for the probability that $U^m + \bar{U}^m < U^f + \bar{U}^f$, i.e.,

$$\Pr\left\{U^m + \bar{U}^m < U^f + \bar{U}^f\right\} \geq 1 - \Pr\left\{R_C \leq \frac{\Delta t}{\Delta g} \leq R_A\right\} \cdot \Pr\left\{R_D \leq \frac{\Delta\bar{t}}{\Delta\bar{g}} \leq R_B\right\}. \tag{26}$$

Thus, as the difference in coalition utilities achieved by the foresighted and myopic strategies increases, i.e., Δu_i becomes large or $\Delta\bar{u}_i$ becomes small (thus, R_A or R_B have smaller values), the lower bound for $\Pr\left\{U^m + \bar{U}^m < U^f + \bar{U}^f\right\}$ increases.

[5]If $\Delta g < 0$, only the lower bound and the upper bound of the term $\{R_C \leq \Delta t/\Delta g \leq R_A\}$ in (26) are switched (i.e., it becomes $\{R_A \leq \Delta t/\Delta g \leq R_C\}$). Hence, the conclusions drawn from this proposition still hold. Similarly, if $\Delta\bar{g} < 0$, the term $\{R_D \leq \Delta\bar{t}/\Delta\bar{g} \leq R_B\}$ in (26) becomes $\{R_B \leq \Delta\bar{t}/\Delta\bar{g} \leq R_D\}$.

Since a general tree topology can be formed by concatenating such elementary sub-trees, Proposition 3 applies to any general tree topologies. Hence, if a foresighted strategy leads to significant improvements in local coalition utilities, it is also more likely to increase the end-to-end application utility.

Note that since the impact of a CU's foresighted decision propagates across its successive classifiers and the application utility is determined by the leaf classifiers, the actions of a foresighted CU near a root classifier with a large coalition may lead to higher improvements in utility. However, there are overheads associated with building large coalitions, specifically in terms of information exchange etc. In general, the size of a coalition may be determined by measuring the *marginal contribution* [27] of each classifier to the coalition utility. The marginal contribution for a classifier represents the coalition utility improvement when the classifier is added to the coalition. The coalition can then be formed by selecting only those classifiers with a high marginal contribution. This will be quantified using simulation results in Section 7.

6. Tree Configuration Games with Resource Constraints

In this section, we investigate the impact of resource constraints on the CU action selections for both myopic as well as foresighted strategies, and the corresponding application utilities.

6.1. Assumptions

6.1.1. Resource Constraints

A processing node with a finite amount of resources can be shared by several classifiers, and the resources need to be divided and allocated to each classifier[6]. The allocated resource to a classifier C_i is denoted by $R_i(<\infty)$. We model the computational resource requirements R_i^{req} for each individual classifier as being directly proportional to the total rate (throughput) of data entering it. Classifier C_i (i.e., c_i or $\bar{c}_i$) has a resource consumption factor α_i, which is the amount of resource required per unit throughput rate. Hence, $R_i^{req} = \alpha_i \cdot t_i$. The local information I_i of CU c_i thus also includes R_i and α_i, i.e., $I_i = \{t_i, \phi_i, \lambda_i^F, \lambda_i^D, \mathbf{A}_i, R_i, \alpha_i\}$.

6.1.2. Effective DET Curve

We assume that each CU uses random load-shedding if input stream data requires more resources than available resources, i.e. $R_i^{req} > R_i$. Random shedding leads to degradation in performance directly proportional to the amount of discarded data. Specifically, for a DET curve $p_i^D = f_i(p_i^F)$, $(0 \le p_i^F \le 1)$ for CU c_i, the effective DET curve with load shedding can be characterized by function $\Gamma : [0,1]^2 \to [0,1]^2$, defined as

$$\Gamma(x, f_i(x)) = (\gamma_i x, \gamma_i f_i(x)), \tag{27}$$

where $0 \le x = p_i^F \le 1$ and scaling factor γ_i, corresponding to the load shed fraction, is determined by

$$\gamma_i = \min\left\{1, \frac{R_i}{R_i^{req}}\right\}. \tag{28}$$

[6]Note that the node can use any resource division policy to achieve desired fairness criteria.

6.1.3. Resource Division Policy

In this paper, we consider two policies that nodes use to distribute resources among classifiers – equal resource division and proportional resource division.

6.2. Resource Constraints and Myopic Strategy

Under resource constraints, the myopic strategy needs to consider the effective DET curve to determine an action. For a "yes" CU c_i with its available information $\mathbf{I}_i = \{I_i = \{t_i, \phi_i, \lambda_i^F, \lambda_i^D, \mathbf{A}_i, R_i, \alpha_i\}, g_i, \Phi_i\}$, an action can be determined based on a strategy π_i such that the action maximizes its local utility. Thus,

$$\begin{aligned} a_i^* = \pi_i(\mathbf{I}_i) &= \underset{a_i \in \mathbf{A}_i}{\arg\max}\, U_i(a_i) \\ &= \underset{a_i=(p_i^F, p_i^D)\in\mathbf{A}_i}{\arg\min} \left[(t_i - g_i\phi_i)(\gamma_i p_i^F)\lambda_i^F + (\Lambda_i - (\gamma_i p_i^D)g_i\phi_i)\lambda_i^M\right], \end{aligned} \tag{29}$$

where γ_i denotes a scale factor that specifies the effective DET curve of c_i, as shown in (27). Since a non-negative constant does not change the convexity/concavity, a solution $a_i^* \in \mathbf{A}_i$ that maximizes $U_i(a_i)$ in (29) can be uniquely determined. In the following analysis, we show that the action a_i^* is the same as the optimal action determined by the myopic strategy in Section 4.1, which does not consider the resource constraints.

Lemma 4. Input throughput and goodput scaled by a non-negative constant do not change the optimal action of a myopic CU.

Proof. Let t_i and g_i be incoming throughput and goodput into a "yes" myopic CU c_i. An action $a_i = (x_i, f_i(x_i)) = (p_i^F, p_i^D)$ of c_i results in outgoing t_i' and g_i', expressed as

$$\begin{bmatrix} t_i' \\ g_i' \end{bmatrix} = \mathbf{T}_i \begin{bmatrix} t_i \\ g_i \end{bmatrix} = \begin{bmatrix} x_i & \phi_i(f_i(x_i) - x_i) \\ 0 & \phi_i f_i(x_i) \end{bmatrix} \begin{bmatrix} t_i \\ g_i \end{bmatrix}.$$

The local utility derived by the action a_i can be expressed as a function of x_i, i.e.,

$$\begin{aligned} U_i(a_i) &= -\left\{(t_i' - g_i')\lambda_i^F + (\Lambda_i - g_i')\lambda_i^M\right\} \\ &= -\left\{x_i(t_i - g_i\phi_i)\lambda_i^F + (\Lambda_i - \phi_i f_i(x_i)g_i)\lambda_i^M\right\} \\ &= -\left[x_i(t_i - g_i\phi_i)\lambda_i^F - \phi_i f_i(x_i)g_i\lambda_i^M\right] - \Lambda_i\lambda_M^i \\ &= U_i^E(x_i) - \Lambda_i\lambda_i^M, \end{aligned}$$

where

$$U_i^E(x_i) \triangleq -\left[x_i(t_i - g_i\phi_i)\lambda_i^F - \phi_i f_i(x_i)g_i\lambda_i^M\right]. \tag{30}$$

Since $U_i(a_i)$ is concave function of x_i, a unique optimal solution x_i^* can be determined, such that

$$\frac{\partial U_i(a_i^*)}{\partial x_i} = 0 \Leftrightarrow \frac{\partial f_i(x_i^*)}{\partial x_i} = \frac{(t_i - g_i\phi_i)\lambda_i^F}{\phi_i g_i \lambda_i^M}, \tag{31}$$

if $0 \le x_i^* \le 1$. If $x_i^* < 0$, $x_i^* = 0$, and if $x_i^* > 1$, $x_i^* = 1$. The corresponding maximum utility is thus $U_i^E(x_i^*) - \Lambda_i\lambda_i^M$. Note that $U_i^E(x_i^*) \ge 0$, since $U_i^E(\cdot)$ is a concave function with $U_i^E(0) = 0$.

Consider a throughput βt_i and a goodput βg_i, which are t_i and g_i scaled by a non-negative constant β. Then, an action $a_i = (x_i, f_i(x_i))$ of c_i results in outgoing t'_i and g'_i, expressed as

$$\begin{bmatrix} t'_i \\ g'_i \end{bmatrix} = \mathbf{T}_i \begin{bmatrix} \beta t_i \\ \beta g_i \end{bmatrix} = \begin{bmatrix} x_i & \phi_i(f_i(x_i) - x_i) \\ 0 & \phi_i f_i(x_i) \end{bmatrix} \begin{bmatrix} \beta t_i \\ \beta g_i \end{bmatrix}.$$

The local utility derived by the action a_i is expressed as

$$\begin{aligned} \tilde{U}_i(a_i) &= -\left\{(t'_i - g'_i)\lambda_i^F + (\Lambda_i - g'_i)\lambda_i^M\right\} \\ &= -\left\{\beta x_i(t_i - g_i\phi_i)\lambda_i^F + (\Lambda_i - \beta\phi_i f_i(x_i) g_i)\lambda_i^M\right\} \\ &= -\beta\left[x_i(t_i - g_i\phi_i\lambda_i^F - \phi_i f_i(x_i) g_i)\lambda_i^M\right] - \Lambda_i\lambda_i^M \\ &= \beta U_i^E(x_i) - \Lambda_i\lambda_i^M. \end{aligned}$$

Since $\tilde{U}_i(a_i)$ is concave function of x_i, a unique optimal solution $\tilde{x}_i$ is determined, such that

$$\frac{\partial U_i(\tilde{x}_i)}{\partial x_i} = 0 \Leftrightarrow \frac{\partial f_i(\tilde{x}_i)}{\partial x_i} = \frac{(t_i - g_i\phi_i)\lambda_i^F}{\phi_i g_i \lambda_i^M}. \tag{32}$$

Since x_i^* and $\tilde{x}_i$ are unique and they are determined by (31) and (32), we conclude that $x_i^* = \tilde{x}_i$. The corresponding maximum utility thus becomes $\beta U_i^E(x_i^*) - \Lambda_i\lambda_i^M$.

Since this property holds for a "no" CU (this can be shown using $\bar{\mathbf{T}}_i$ with the same argument), we conclude that an action determined based on the myopic strategy does not change by a throughput and a goodput scaled by a non-negative constant.

We can deduce the following corollary.

Corollary 5. If the optimal myopic action of CU c_i achieves a maximum local utility $U_i^E(x_i^*) - \Lambda_i\lambda_i^M$ for incoming throughput t_i and goodput g_i, then CU c_i can achieve its maximum local utility $\beta U_i^E(x_i^*) - \Lambda_i\lambda_i^M$ for β-scaled ($\beta \geq 0$) throughput βt_i and goodput βg_i using the same optimal action.

Based on the above two observations, we can state the impact of resource constraints on the myopic strategy as follows.

Proposition 6. An action determined by the myopic strategy is not altered by a random load-shedding scheme.

Proof. Let t_i and g_i be incoming throughput and goodput into a classifier $C_i = \{c_i, \bar{c}_i\}$. Let $a_i^* = (x_i^*, f_i(x_i^*))$ be an optimal action of c_i determined by the myopic strategy, and the corresponding utility be denoted by $U_i^E(x_i^*) - \Lambda_i\lambda_i^M$. A random load shedding scheme of c_i with allocated resources R_i results in an effective DET curve, determined by a mapping $\Gamma(x_i, f_i(x_i)) = (\gamma_i x_i, \gamma_i f_i(x_i))$, where $\gamma_i = \min\{1, R_i/R_i^{req}\}$. Then, an action $a_i = (x_i, f_i(x_i))$ leads to output stream rates t'_i and g'_i, expressed as

$$\begin{bmatrix} t'_i \\ g'_i \end{bmatrix} = \begin{bmatrix} \gamma_i x_i & \phi_i(\gamma_i f_i(x_i) - \gamma_i x_i) \\ 0 & \phi_i\gamma_i f_i(x_i) \end{bmatrix} \begin{bmatrix} t_i \\ g_i \end{bmatrix} = \begin{bmatrix} x_i & \phi_i(f_i(x_i) - x_i) \\ 0 & \phi_i f_i(x_i) \end{bmatrix} \begin{bmatrix} \gamma_i t_i \\ \gamma_i g_i \end{bmatrix}. \tag{33}$$

Note that (33) is equivalent to the case, where t_i and g_i are scaled by a constant γ_i. As shown in Lemma 4, constant-scaled throughput and goodput do not change an optimal action determined by the myopic strategy. Thus, we conclude that the random load-shedding scheme does not change the action determined by the myopic strategy, and thus, the optimal action under resource constraints is still $a_i^* = (x_i^*, f_i(x_i^*))$. Note that the corresponding local utility of c_i is $\gamma_i U_i^E(x_i^*) - \Lambda_i \lambda_i^M$ as shown in Corollary 5. The same argument holds for $\bar{c}_i$.

Therefore, Algorithm 1 can also be used to efficiently find the optimal action under resource constraints. Finally, since each CU can always determine its own actions, Proposition 2 still holds, implying that there is an equilibrium in operating points among the CUs.

6.3. Resource Constraints and Foresighted Strategy

Foresighted CUs need to determine their actions while considering the impact on resource allocations for successive classifiers in the coalition, as well as the corresponding coalition utilities. This is because the action of a classifier modifies resource requirements for all successive classifiers, thereby potentially modifying their available resources. We summarize the effect of resource constraints on the coalition utility in Lemma 7.

Lemma 7. If an action a_i of a foresighted CU c_i results in a coalition utility $U_k^E(x_k^*) - \Lambda_k \lambda_k^M$ at a leaf CU c_k in its coalition under no resource constraint, then the same action a_i results in the coalition utility

$$U_k(x_k^*) = \left(\prod_{\forall j, i \rightsquigarrow j, j \rightsquigarrow k} \gamma_j \right) U_k^E(x_k^*) - \Lambda_k \lambda_k^M, \tag{34}$$

under resource constraints, where γ_j represents the load shed fraction in CU $\hat{c}_j$ defined in (28).

Proof. As shown in Proposition 6, random load-shedding does not change the actions determined by the myopic strategy. However, due to the resource constraints, the outgoing stream rates of CU j are scaled by γ_j for all j, $i \rightsquigarrow j$ and $j \rightsquigarrow k$. Since a branch consists of cascaded CUs $\hat{c}_j$, the coalition utility becomes

$$\left(\prod_{\forall j, i \rightsquigarrow j, j \rightsquigarrow k} \gamma_j \right) U_k^E(x_k^*) - \Lambda_k \lambda_k^M,$$

using Corollary 5.

Lemma 7 clearly shows that the derived utility from each leaf CU $\hat{c}_k$ in coalition C_i depends on all γ_j such that $i \rightsquigarrow j$ and $j \rightsquigarrow k$. Since γ_j is determined based on CU $\hat{c}_j$'s resource requirements and the allocated resources from a node, it depends on the actions taken by other CUs sharing the same node as well as the resource division policy of the node. Thus, each action of the foresighted CU c_i may result in different distributions for γ_j. Note that in distributed systems, information about the resource division policy (which may be different per node) or classifier placement may not be available to each classifier. Hence, foresighted CUs need to observe the outcomes of their actions and iteratively learn the mapping between γ_j and their actions. This learning involves determining the probability mass function (pmf) of γ_j for each action. In this paper, we use a simple learning strategy that estimates the pmf of γ_j based on empirical frequency measurement. Different learning strategies [28, 29] may be used for this task, and may easily be incorporated into our designed foresighted strategy.

Based on the estimated pmf of γ_j for each action, a foresighted strategy π_i of c_i for its coalition $\mathbf{G}_i$ determines an action a_i^* that maximizes *expected* coalition utility, defined as

$$\begin{aligned} a_i^* = \pi_i(\mathbf{I}_i) &= \arg\max_{a_i \in \mathbf{A}_i} E\left[U_{\mathbf{G}_i}(a_i)\right] \\ &= \arg\min_{a_i \in \mathbf{A}_i} \sum_{\hat{c}_k \in \mathbf{C}_L(\mathbf{G}_i)} E\left[U_k(x_k^*)\right] \\ &= \arg\min_{a_i \in \mathbf{A}_i} \sum_{\hat{c}_k \in \mathbf{C}_L(\mathbf{G}_i)} \left[\left(\prod_{\forall j, i \rightsquigarrow j, j \rightsquigarrow k} E\{\gamma_j\}\right) U_k^E(x_k^*) - \Lambda_k \lambda_k^M\right], \end{aligned} \tag{35}$$

where utility $U_k(x_k^*)$ of leaf CU $\hat{c}_k$ in $\mathbf{G}_i$ is defined in (34). Our proposed algorithm is shown in Algorithm 2.

Algorithm 2 Foresighted Decision Process based on γ_j Estimation using Empirical Frequency

Given: granularity level n_i, $U_k^E(x_k^*)$ for all leaf CUs $\hat{c}_i$ in $\mathbf{G}_i$; $\hat{c}_k \in \mathbf{C}_L(\mathbf{G}_i)$, Λ_k, λ_k^M.

1: **Initialization**

$H_j^{a_i} = [H_j^{a_i}(1), \ldots, H_j^{a_i}(n_i)] \leftarrow \overbrace{[1, 0, \ldots, 0]}^{1 \times n_i}$ for all $\hat{c}_j \in \mathbf{G}_i$ and for all $a_i \in \mathbf{A}_i$.

$\Pr\left\{\gamma_j = (2r-1)/2n_i\right\} \leftarrow H_j^{a_i}(r) / \sum_{h=1}^{n_j} H_j^{a_i}(h)$ for $r = 1, \ldots, n_i$.

while stream data ingested **do**

2: **Find and take a foresighted action a_i^***

Find a_i^* using (35).

3: **Update $H_j^{a_i^*}$**

Observe γ_j that corresponds to a_i^* for all $\hat{c}_j \in \mathbf{G}_i$

if $(r-1)/n_i \leq \gamma_j < r/n_i$, **then**

$H_j^{a_i^*}(r) \leftarrow H_j^{a_i^*}(r) + 1$

4: **Update Probability Mass Function**

$\Pr\left\{\gamma_j = (2r-1)/2n_i\right\} \leftarrow H_j^{a_i}(r) / \sum_{h=1}^{n_j} H_j^{a_i}(h)$ for $r = 1, \ldots, n_i$.

7. Simulation Results

7.1. Simulation Set-up

To evaluate the performance of the proposed approaches, we consider a semantic concept detection application [16] for sports image in Fig. 1. This application is built using IBM's Multimedia Analysis and Retrieval Searching (IMARS) [30] classifiers. Incoming images of several different types are classified into six classes of interest: *Little League Baseball*, *Basketball*, *Cricket*, *Skating*, *Skiing* and *Tennis*, each of which corresponds to a specific type of sport. By introducing a set of additional intermediate concept detectors, we then construct a hierarchical topology of classifiers. The *Team Sports* classifier filters data relevant to the *Little League, Cricket* and *Basketball* classifiers. The *Winter Sports* classifier filters data relevant to the *Skating* and *Skiing* classifiers. Finally, the *Racquet Sports* classifier filters data relevant to the *Tennis* classifier. The mutually exclusive nature of concepts *Team Sports, Winter Sports* and *Racquet Sports* allows identifying them in series, i.e., passing only data that does not belong to a class to the next class. For simplicity of notation, each classifier is enumerated as shown in Fig. 1, and thus, our

Table 2: A-priori Probabilities in Application

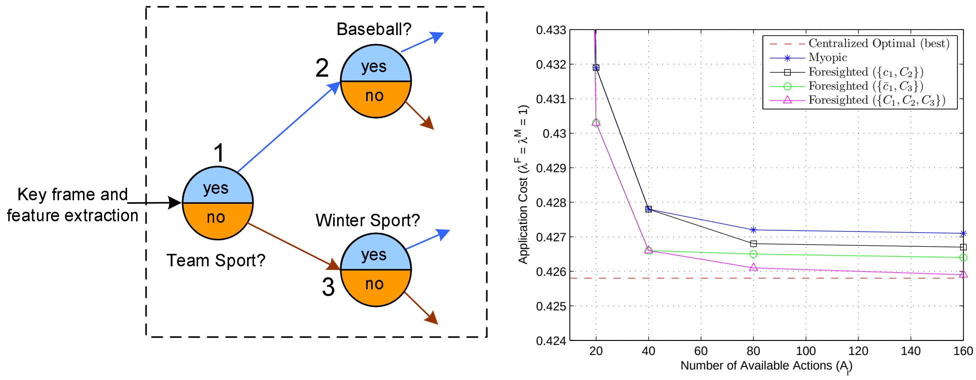
Classifier	Team Sports	Baseball	Little League	Basketball	Cricket	Winter Sports
ϕ_i	0.2371	0.2071	0.2132	0.2370	0.0358	0.3674
Classifier	Ice Sports	Skating	Skiing	Racquet Sports	Tennis	-
ϕ_i	0.0392	0.0358	0.6951	0.6374	0.4056	-

application is denoted by $\mathbf{S} = \{C_1, C_2, \ldots, C_{11}\}$. Misclassification cost coefficients are assigned to leaf CUs, $\mathbf{C}_L(\mathbf{S}) = \{c_4, \bar{c}_4, c_5, \bar{c}_7, c_8, \bar{c}_8, c_9, \bar{c}_9, c_{10}, \bar{c}_{10}, c_{11}, \bar{c}_{11}\}$. Misclassification cost coefficients for intermediate classifiers are assigned based on the proposed backward propagation scheme given in (1). Each classifier operates on low level image features such as color histograms, color correlograms, etc. and uses an SVM based classification strategy [31]. The DET curves, a-priori probabilities as well as computational requirements for individual classifiers were experimentally measured on a set of 20000 test images. The measured a-priori probabilities ϕ_i for each classifier in our application are shown in Table 2. We evaluate the myopic and foresighted strategies, and compare them against a centralized approach (based on SQP) previously proposed in [16]. In this section results are presented in terms of the application *cost* – negative of the utility defined in (3) and (4). By definition, maximizing the application utilities is equivalent to minimizing the application costs.

7.2. Impact of Action Granularity and Backward Propagation Scheme for Cost Coefficients

To highlight the impact of the number of available actions, the proposed backward propagation scheme for cost coefficients, and the foresighted strategies on the application cost, we consider an elementary sub-tree of our application with classifiers 1, 2, and 3, shown in Fig. 4(a). In this sub-tree, C_2 and C_3 are leaf classifiers, and we set $\lambda_l^F(\bar{\lambda}_l^F) = \lambda_l^M(\bar{\lambda}_l^M) = 1$ for $l = 2, 3$. Based on this, we derive $\lambda_1^F(\bar{\lambda}_1^F)$ and $\lambda_1^M(\bar{\lambda}_1^M)$ using the backward propagation scheme in (1). We consider different levels of quantization granularity, i.e., increased numbers of available actions $A_i = 20, 40, 80, 160$ for $i = 1, 2, 3$. We set the incoming stream rate $t_r = 1$. The resulting misclassification costs for different numbers of actions and different degrees of foresightedness (i.e., different coalitions) are shown in Fig. 4(b).

Fig. 4(b) shows that increasing the number of available actions leads to a lower application cost for the proposed approaches. On the graph, we also show the cost associated with the centralized optimal solution. We observe a slight performance gap between the myopic and the centralized optimal solution, as CUs with the myopic strategy determine their actions individually with no consideration of the impact on the end-to-end application cost. Moreover, unlike in the centralized approach, only finite number of (quantized) actions are available to each CU. However, the foresighted strategies lead the application costs to approach the centralized optimal solution, while always outperforming myopic strategies. This is because any actions selected by the myopic strategies are always included in the candidate actions for the foresighted strategies. The results also show that increasing the coalition size can lead to a lower application cost. Specifically, coalition $\{C_1, C_2, C_3\}$ achieves a lower cost than coalitions $\{c_1, C_2\}$ and $\{\bar{c}_1, C_3\}$, both of which outperform the myopic strategy. We also observe that the achieved costs (or utilities) from the same size coalitions, i.e., $\{c_1, C_2\}$ and $\{\bar{c}_1, C_3\}$, are different. This is because the derived application costs depend on not only the foresighted actions (a_1^* and $\bar{a}_1^*$) but also their DET curves



(a) (b)

Figure 4: (a) elementary sub-tree from our application, and (b) application costs for the elementary sub-tree.

Table 3: Achieved Application Costs (80 actions, Myopic Strategy)

Experiment Cases	$\lambda^F = \lambda^M$	$\lambda^F = 4\lambda^M$	$4\lambda^F = \lambda^M$
Centralized (average)	0.587 (0 %)	0.963 (0 %)	1.373 (0 %)
Distributed	0.536 (68.0 %)	0.806 (86.3 %)	1.187 (88.6 %)
Centralized (best)	0.512 (100 %)	0.781 (100 %)	1.163 (100 %)

and $\hat{\phi}_l$, $l = 2, 3$. Hence, forming coalition with CUs having better DET performance or higher $\hat{\phi}_l$ can result in lower coalition cost (i.e., higher coalition utility).

Finally, instead of the backward propagation scheme, if misclassification cost coefficients of C_1 are determined randomly or based on a heuristic average approach (e.g., $\lambda_1^F = (\lambda_2^F + \bar{\lambda}_2^F)/2$ and $\lambda_1^M = (\lambda_2^M + \bar{\lambda}_2^M)/2$), the resulting misclassification costs are 1.0108 and 0.8167 (for $A_i = 80$), respectively. These are 139.5% and 112.7% of the cost (0.7245) achieved by the proposed approach. The performance degradation for these schemes is caused because they do not consider stream characteristics.

7.3. Application Performance under No Resource Constraints

In this section, we quantify the performance of the entire application shown in Fig 1 achieved by the proposed tree configuration games. In this simulation, we use $A_i = 80$, $1 \leq i \leq 11$, and each CU deploys the myopic strategy. We compare against the centralized solution using SQP in [16]. As discussed in [16], since the gradient descent based SQP does not guarantee a global optimal solution, we use several different randomized starting points, and provide the minimum (best) as well as the average cost (average). The results are shown in Table 3.

It is clear that the proposed distributed approach based on myopic strategy always outperforms the *average* performance of centralized approach for different misclassification cost scenarios i.e., ($\lambda^F = \lambda^M = 1$), ($\lambda^F = 4, \lambda^M = 1$), and ($\lambda^F = 1, \lambda^M = 4$). These correspond to equal cost for false alarms and misses, high costs for false alarms, and high costs for misses respectively. Additionally, as the costs become unbalanced, the distributed algorithm perfor-

Table 4: Experimental Setup

Experiment Case	Coalitions in “yes” (c_1) branch	Coalitions in “no” ($\bar{c}_1$) branch
1	$\{c_1\}$ (self coalition)	$\{\bar{c}_1\}$ (self coalition)
2	$\mathbf{G}_1 = \{c_1, C_2\}$	$\bar{\mathbf{G}}_1 = \{\bar{c}_1, C_3\}$
3	$\mathbf{G}_1 = \{c_1, C_2, C_4\}$	$\bar{\mathbf{G}}_1 = \{\bar{c}_1, C_3, C_6\}$
4	$\mathbf{G}_1 = \{c_1, C_2, C_5\}$	$\bar{\mathbf{G}}_1 = \{\bar{c}_1, C_3, C_7\}$
5	$\mathbf{G}_1 = \{c_1, C_2, C_5\}$, $\bar{\mathbf{G}}_2 = \{\bar{c}_2, C_5, \}$	$\bar{\mathbf{G}}_1 = \{\bar{c}_1, C_3, C_6, C_7\}$

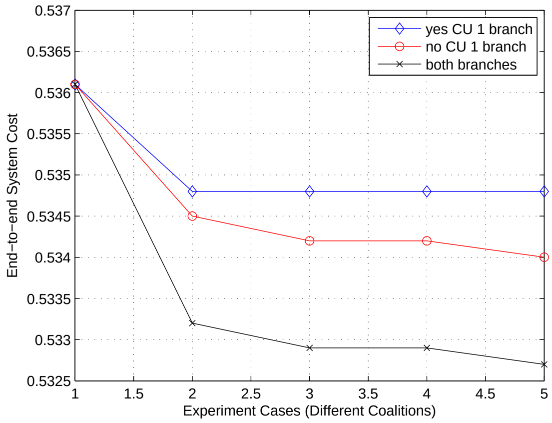


Figure 5: Application costs.

mance increasingly approaches the optimal performance of the centralized algorithms. These are reflected in the percentages in the table, computed as $\frac{(Cost - Cost^{cent}_{avg})}{(Cost^{cent}_{best} - Cost^{cent}_{avg})} \times 100\%$.

In order to investigate the impact of the foresighted strategies on application costs, we consider various coalition settings shown in Table 4. Again, we use $A_i = 80$, $1 \leq i \leq 11$, and set $\lambda^F_l(\bar{\lambda}^F_l) = \lambda^M_l(\bar{\lambda}^M_l) = 1$ for leaf CUs. The resulting application costs for various coalition settings are shown in Fig. 5.

These results show that the foresighted decisions can lead to a lower application cost (or equivalently, a higher application utility). In Fig. 5, the largest application cost (i.e., the minimum application utility) is incurred when CUs select their actions based on the myopic strategies (i.e., in Experiment Case 1, where no coalition is formed). Moreover, we observe that the application cost reduces (utility improves) as the coalition size increases. Interestingly, the cost reduction becomes significant when classifier 1 (i.e., the first classifier of this tree) starts to form its coalitions. For example, in both branches in Fig. 5, the largest cost reduction is achieved when classifier 1 forms coalitions with classifier 2 or classifier 3. However, the improvement gained for enlarging coalitions diminishes rapidly. These observations can be clearly quantified using *marginal contribution* [27] to the application utility. The marginal contribution $\Delta V_{\hat{c}_k}(\mathbf{G}_i)$ of $\hat{c}_k (\notin \mathbf{G}_i)$ to coalition utility $U_{\mathbf{G}_i}$ is defined as

$$\Delta V_{\hat{c}_k}(\mathbf{G}_i) = U_{\mathbf{G}_i^+} - U_{\mathbf{G}_i},$$

Table 5: Achieved Application Costs (80 actions)

Experiment Cases	Application Cost
Centralized (Average)	0.587 (0 %)
Myopic	0.536 (68.0 %)
Foresighted (c_1, C_2)	0.535 (69.3 %)
Foresighted $(\bar{c}_1, C_3)$	0.535 (69.3 %)
Foresighted (C_1, C_2, C_3)	0.533 (72.0 %)
Centralized (Best)	0.512 (100 %)

where $U_{\mathbf{G}_i^+}$ and $U_{\mathbf{G}_i}$ denote the application utilities achieved by coalition $\mathbf{G}_i^+$ and $\mathbf{G}_i$, respectively, and $\mathbf{G}_i^+$ denotes a enlarged coalition of $\mathbf{G}_i$ with additional CU $\hat{c}_k$, i.e., $\mathbf{G}_i^+ = \mathbf{G}_i \cup \{\hat{c}_k\}$. For example, the marginal contribution of C_3 to the application utility achieved by $\{\bar{c}_1\}$ (self coalition) is computed as

$$\Delta V_{C_3}(\{\bar{c}_1\}) = U_{\{\bar{c}_1, C_3\}} - U_{\{\bar{c}_1\}} = (-0.5345) - (-0.5361) = 0.0016.$$

However, the marginal contribution of C_6 to the application utility achieved by coalition $\{\bar{c}_1, C_3\}$ is given by

$$\Delta V_{C_6}(\{\bar{c}_1, C_3\}) = U_{\{\bar{c}_1, C_3, C_6\}} - U_{\{\bar{c}_1, C_3\}} = (-0.5342) - (-0.5345) = 0.0003,$$

which is approximately 20% of application utility improvement compared to $\Delta V_{C_3}(\{\bar{c}_1\})$. This confirms the observed application utility result trends. The marginal contribution can be used by foresighted CUs to determine their coalition size. For example, a foresighted CU c_i can include additional CU $\hat{c}_k$ in its coalition $\mathbf{G}_i$, only if its marginal contribution is more than a certain threshold θ_i, i.e., $\Delta V_{\{\hat{c}_k\}}(\mathbf{G}_i) > \theta_i$.

We also compare against the centralized solution in [16], and present results in Table 5. The percentages in the table are computed as $\frac{(Cost - Cost_{avg}^{cent})}{(Cost_{best}^{cent} - Cost_{avg}^{cent})} \times 100\%$. In the considered coalitions, the foresighted strategies for coalitions $\{c_1, C_2\}$ or $\{\bar{c}_1, C_3\}$ enable the application to achieve approximately 69% of centralized best solution. However, the application can achieve 72% of centralized best solution with the foresighted decisions for a larger coalition $\{C_1, C_2, C_3\}$, which is consistent to the result from Proposition 3. Note that a higher performance improvement can be obtained by enlarging the coalitions, as discussed in Section 7.2.

7.4. *Performance of Application with Resource Constraints*

We consider two different resource division policies at the nodes i) equal allocation, and ii) proportional allocation (proportional to each classifier's processing requirements). The processing complexity for our different classifiers (measured experimentally) is shown in Table 6. P is a normalization constant that normalizes the processor power (speed). The processing requirements for C_i are determined as $R_i^{req} = \alpha_i \cdot t_i$ and the corresponding γ is determined by (28).

7.4.1. *One Foresighted CU with Other Myopic CUs*

In this section, we consider two different cases, each with only one foresighted CU in the application. In the first case, we have foresighted CU c_1 that forms coalition $\{c_1, C_2\}$, while in

Table 6: Processing Complexity Per Image

Classifier	Team Sports	Baseball	Little League	Basketball	Cricket	Winter Sports
Complexity	$0.3884 \times P$	$0.1761 \times P$	$0.1307 \times P$	$0.0772 \times P$	$0.2006 \times P$	$0.3199 \times P$
Classifier	Ice Sports	Skating	Skiing	Racquet Sports	Tennis	-
Complexity	$0.2223 \times P$	$0.2403 \times P$	$0.2608 \times P$	$0.1276 \times P$	$0.1720 \times P$	-

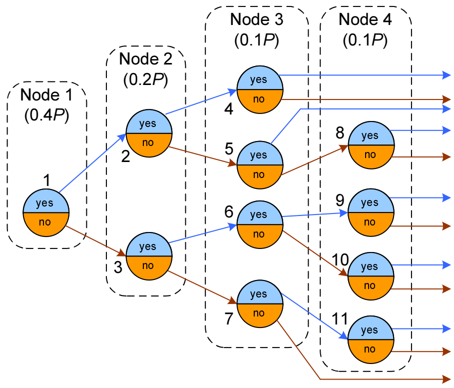


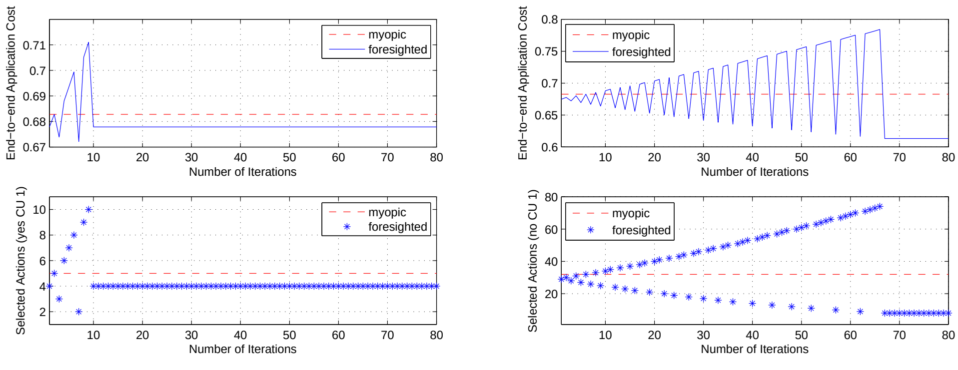
Figure 6: Application placement and available resources in each node.

the second case, we have foresighted CU $\bar{c}_1$ that forms coalition $\{\bar{c}_1, C_3\}$. Foresighted classifiers use empirical frequency based learning (Algorithm 2) to estimate the pmf of γ. The 11 classifiers in the considered application are distributed over 4 nodes as shown in Fig. 6: classifier 1 is located in node 1, classifiers 2 and 3 are located in node 2, classifiers 4, 5, 6, and 7 are located in node 3, and classifiers 8, 9, 10, and 11 are located in node 4. The available resources for node $\mathcal{N}$ ($\mathcal{N} = 1, 2, 3, 4$) are $R_{\mathcal{N}}^A = 0.4P, 0.2P, 0.1P, 0.1P$, respectively. We assume that each $A_i = 80$, $1 \le i \le 11$, and each node allocates its resources based on proportional resource division policy, which determines resource allocation R_i of C_i in a node $\mathcal{N}$ such that

$$R_i = \frac{R_i^{req}}{\sum_{h \in \mathcal{N}} R_h^{req}} \times R_{\mathcal{N}}^A, \tag{36}$$

where R_i and R_i^{req} denote the allocated and requested resources. The resulting application cost is presented in Fig. 7. We also compare against a completely myopic solution, where all classifiers use myopic strategies.

Fig. 7(a) and Fig. 7(b) show the selected actions and the corresponding application costs for foresighted CUs c_1 and $\bar{c}_1$, respectively. Action index $\hat{a}_{1,k}$ ($1 \le k \le A_1 = 80$) of CU $\hat{c}_1$ represents kth operating point $\left(\hat{p}_{ik}^F, \hat{p}_{ik}^D\right) = \left(\frac{k-1}{\hat{A}_i - 1}, \hat{f}_i\left(\frac{k-1}{\hat{A}_i - 1}\right)\right)$, as defined in (6). As discussed in Section 6.2, the actions determined by the completely myopic solution are not affected by resource constraints. Hence, in both Fig. 7(a) and Fig. 7(b), the selected actions based on the myopic strategy do not change over time, and the resulting application costs are constant. However, the actions selected by the foresighted CUs vary over time until the γ estimation based on Algorithm 2 is completed.



(a) Foresighted "yes" CU 1 (c_1) (b) Foresighted "no" CU 1 ($\bar{c}_1$)

Figure 7: Selected foresighted actions (over time) and the corresponding application costs.

In this case, the algorithm shown in Algorithm 2 guarantees a convergence of the foresighted action. The maximum number of iterations required for convergence is bounded by the number of available actions, i.e., $A_1 = 80$ in this example[7]. Note that γ for each CU sharing a node that deploys the proportional resource divisions scheme is the same, because γ_i for c_i (or $\bar{c}_i$) is determined by $\gamma_i = \min\{1, R_i/R_i^{req}\}$, and

$$\frac{R_i}{R_i^{req}} = \frac{\frac{R_i^{req}}{\sum_{h\in\mathcal{N}} R_h^{req}}}{R_i^{req}} = \frac{1}{\sum_{h\in\mathcal{N}} R_h^{req}}, \tag{37}$$

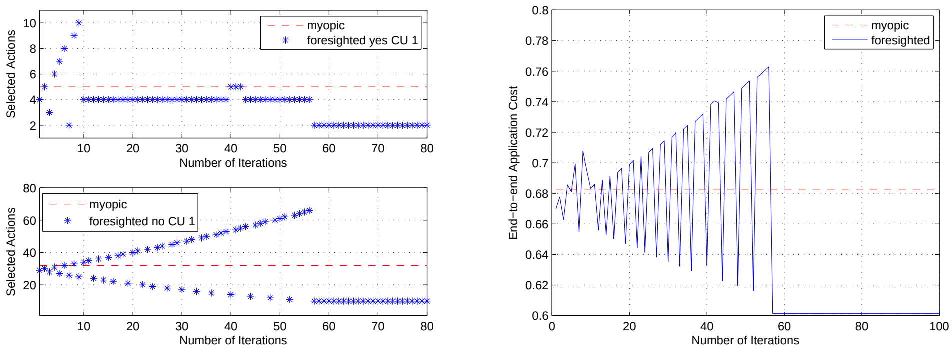
for all CUs that share node $\mathcal{N}$.

7.4.2. Foresighted CUs Competing for Resources

We now consider the case where two foresighted CUs compete for resources on a shared node. In this experiment, we assume that both c_1 and $\bar{c}_1$ are foresighted, with coalitions $\{c_1, C_2\}$ and $\{\bar{c}_1, C_3\}$, and both use empirical frequency based learning (Algorithm 2) to estimate the pmf of γ. Other simulation settings are the same as in Section 7.4.1, and the results are shown in Fig. 8.

Since both c_1 and $\bar{c}_1$ are foresighted, their actions jointly affect the resource allocations for classifiers C_2 and C_3, which leads to different resulting application costs, numbers of iterations for the convergence, and the converged optimal actions. Compared to the results in Fig. 7, we can observe that the numbers of iterations required to converge to their optimal actions are different: c_1 requires more iterations while $\bar{c}_1$ requires less iterations. Moreover, the converged optimal actions are also different. Specifically, the converged foresighted action index of c_1 in Fig. 7 is $a_1^* = a_{1,4}$, while $a_1^* = a_{1,2}$ in Fig. 8. Moreover, the converged foresighted action index of $\bar{c}_1$ in Fig. 7 is $\bar{a}_1^* = a_{1,8}$, while $\bar{a}_1^* = a_{1,10}$ in Fig. 8. Note that with increased number of foresighted CUs, the resulting cost decreases over the results in Fig. 7(b).

[7]In practice, the actual number of iterations before convergence depends on the DET curves and the first determined action.



(a) Selected actions of foresighted CUs

(b) Application costs

Figure 8: Selected actions and application costs.

Table 7: Achieved Application Costs (80 actions, Different Resource Division Policies)

	Equal Resource Division Policy	Proportional Resource Division Policy
Myopic Strategy	0.8243	0.6828
Coalition $\mathbf{G}_1 = \{c_1, C_2\}$	0.8175	0.6779
Coalition $\bar{\mathbf{G}}_1 = \{\bar{c}_1, C_3\}$	0.7885	0.6133
Coalitions $\{\mathbf{G}_1, \bar{\mathbf{G}}_1\}$	0.7817	0.6015

7.4.3. Impact of Resource Division Policies

Finally, we investigate the impact of different resource division policies on the convergence. We use the same settings as for previous experiments, however we allow node 2 (where classifiers 2 and 3 are placed) to use the two different policies – equal and proportional division. The equal resource division policy allocates resources equally to all classifiers placed on it, i.e.,

$$R_i = R^A_{\mathcal{N}}/|\mathcal{N}|, \text{ for all } C_i \text{ placed on } \mathcal{N}, \tag{38}$$

where $|\mathcal{N}|$ denotes the number of classifiers in node $\mathcal{N}$. The resulting application costs are shown in Table 7.

The results in Table 7 show that the proportional resource division policy always leads to better performance (lower application cost) than the equal resource division policy. This is because the equal resource division policy does not consider the resource requirements of the different classifiers (that depend on input data stream characteristics, processing costs etc.), while the proportional resource division policy does.

8. Conclusions

In this paper, we focus on the optimization of stream mining applications constructed as tree topologies of classifiers. We model the optimized configuration of these classifier tree topologies as a tree configuration game. We develop distributed approaches, where individual classifiers configure themselves by selecting actions (from a set of discrete available actions) that maximize their local utilities. We determine the minimum information that needs to be exchanged across classifiers and propose a novel scheme for determining the local utilities for intermediate CUs, which are required to successfully design the tree configuration games. We analytically show that the proposed approach guarantees a convergence to a unique pure strategy Nash equilibrium in operating points of each CU. If information about successive classifiers is available to a CU, then the CU can form a coalition for the successive classifiers, and select a foresighted action that maximizes the coalition utility. We analytically show that the foresighted strategies can eventually improve the application utility. Our simulation results, performed on a semantic concept detection application for sports image analysis, show that the performance of the proposed myopic approach is comparable to a centralized solution – outperforming the average performance. Specifically, the myopic solution improves 68% ~ 88.6% (depending on the misclassification costs) of the application utility compared to the centralized SQP-based approach on average, and a higher improvement can be achieved by deploying the foresighted solutions. Moreover, if more actions become available, a higher application utility is achieved. Foresighted strategies that maximize coalition utilities lead to a higher application utility than the myopic strategy. In addition, the performance incrementally improves as the coalition size increases. Future interesting research topics may include how to configure the topology of classifiers (or sensors) in tree or chain structures in an ad hoc manner (i.e., ordering classifiers) and what are the impact of the other resource division policies and load-shedding schemes on the system performance in the resource constrained scenarios.

Appendix A.

We show that the local utility function defined in (7) is concave if a DET curve is differentiable and non-decreasing concave. A function $U_i(x)$ is concave with respect to x if and only if $\frac{\partial^2 U_i(x)}{\partial x^2} \leq 0$.

Let f_i be a differentiable and non-decreasing concave function that characterizes the DET curve of CU c_i, i.e., $p_i^D = f_i(p_i^F)$ for $0 \leq p_i^F \leq 1$. Since function f_i is concave, we have

$$\frac{\partial^2 f_i(x_i)}{\partial x_i^2} \leq 0, \tag{A.1}$$

for $0 \leq x_i = p_i^F \leq 1$. Since p_i^D can be completely parameterized by p_i^F and f_i, the local utility function $U_i(a_i)$ for $a_i = (x_i, f_i(x_i))$ can be expressed as a function of x_i, i.e.,

$$U_i(x_i) = -[(t_i - g_i\phi_i)x_i\lambda_i^F + (\Lambda_i - g_i\phi_i f(x_i))\lambda_i^M].$$

The second derivative of $U_i(x_i)$ is given by

$$\frac{\partial^2 U_i(x_i)}{\partial x_i^2} = g_i\phi_i \frac{\partial^2 f_i(x_i)}{\partial x_i^2}\lambda_i^M \leq 0,$$

since parameters of g_i, ϕ_i, and λ_i^M are non-negative constants and $\frac{\partial^2 f_i(x_i)}{\partial x_i^2} \leq 0$ as shown in (A.1).

References

[1] M. A. Shah, J. M. Hellerstein, S. Chandrasekaran, M. J. Franklin, Flux: An adaptive partitioning operator for continuous query systems, in: Proc. Int. Conf. on Data Engineering (ICDE '03), 2003, pp. 25–36.

[2] R. Lienhart, L. Liang, A. Kuranov, A detector tree for boosted classifiers for real-time object detection and tracking, in: Proc. Int. Conf. on Multimedia and Expo (ICME '03), 2003, pp. 277–230.

[3] A. Ntoulas, M. Najork, M. Manasse, D. Fetterly, Detecting spam web pages through content analysis, in: Proc. Int. Conf. on World Wide Web (WWW '06), 2006, pp. 83–92.

[4] T. E. Senator, Multi-stage classification, in: Proc. Int. Conf. on Data Mining (ICDM '05), 2005, pp. 386–393.

[5] X. Gu, K. Nahrstedt, On composing stream applications in cooperative peer-to-peer environments, IEEE Trans. Parallel Distrib. Syst. 17 (8) (2006) 824–837.

[6] C. Olston, J. Jiang, J. Widom, Adaptive filters for continuous queries over distributed data streams, in: Proc. Int. Conf. on Management of Data (SIGMOD '03), 2003, pp. 563–574.

[7] L. Amini, H. Andrade, F. Eskesen, R. King, Y. Park, P. Selo, C. Venkatramani, The stream processing core, Tech. Rep. RSC 23798, IBM T.J. Watson Research Center (Nov. 2005).

[8] Y. Xing, S. B. Zdonik, J.-H. Hwang, Dynamic load distribution in the borealis stream processor, in: Proc. Int. Conf. on Data Engineering (ICDE '05), 2005, pp. 791–802.

[9] M. Cherniack, H. Balakrishnan, M. Balazinska, D. Carney, U. Çetintemel, Y. Xing, S. B. Zdonik, Scalable distributed stream processing, in: Proc. Conf. on Innovative Data Systems Research (CIDR), 2003.

[10] D. Kossmann, The state of the art in distributed query processing, ACM Computing Surveys 32 (4) (2000) 422–469.

[11] M. Balazinska, H. Balakrishnan, S. Madden, M. Stonebraker, Fault tolerance in the borealis distributed stream processing system, in: Proc. Int. Conf. on Management of Data (SIGMOD '05), 2005, pp. 13–24.

[12] A. Garg, V. Pavlovic, T. S. Huang, Bayesian networks as ensemble of classifiers, in: Proc. Int. Conf. on Pattern Recognition (ICPR '02), 2002, pp. 779–784.

[13] T. Repantis, X. Gu, V. Kalogeraki, QoS-aware shared component composition for distributed stream processing systems, IEEE Trans. Parallel Distrib. Syst.To appear.

[14] A. Gavrilovska, S. Kumar, S. Sundaragopalan, K. Schwan, Platform overlays: Enabling in-network stream processing in large-scale distributed applications, in: Proc. Int. Workshop on Netw. and Operating Systems Support for Digital Audio and Video (NOSSDAV '05), 2005.

[15] Y. Mao, X. Zhou, D. Pi, Y. Sun, S. T. C. Wong, Multiclass cancer classification by using fuzzy support vector machine and binary decision tree with gene selection., Journal of Biomedical Biotechnology 2005 (2) (2005) 160–171.

[16] D. S. Turaga, B. Foo, O. Verscheure, M. van der Schaar, Configuring topologies of distributed semantic concept classifiers for continuous multimedia stream processing, in: ACM Multimedia 2008, 2008, pp. 289–298.

[17] S. Pang, SVM classification tree algorithm with application to face membership authentication, in: Proc. Int. Joint Conf. on Neural Networks, Vol. 1, 2004, pp. 431–436.

[18] D. M. Kreps, R. Wilson, Sequential equilibrias, Econometrica 50 (4) (1982) 863–894.

[19] B. Babcock, S. Babu, M. Datar, R. Motwani, Chain: Operator scheduling for memory minimization in data stream systems, in: Proc. Int. Conf. on Management of Data (SIGMOD '03), 2003, pp. 253–264.

[20] N. Tatbul, U. Çetintemel, S. B. Zdonik, Staying fit: Efficient load shedding techniques for distributed stream processing, in: Proc. Int. Conf. on Very Large Databases (VLDB '07), 2007, pp. 159–170.

[21] S. Viglas, J. Naughton, Rate-based query optimization for streaming information sources, in: Proc. Int. Conf. on Management of Data (SIGMOD '02), 2002, pp. 37–48.

[22] D. S. Turaga, H. Park, R. Yan, O. Verscheurerh, Adaptive multimedia mining on distributed stream processing systems, in: Proc. IEEE Int. Conf. on Data Mining (ICDM 2009), 2009.

[23] F. Provost, T. Fawcett, Robust classification for imprecise environments, Machine Learning 42 (3) (2001) 203–231.

[24] S. Boyd, L. Vandenberghe, Convex Optimization, Cambridge University Press, New York, NY, 2004.

[25] A. Ortega, K. Ramchandran, Rate-distortion techniques in image and video compression, IEEE Signal Process. Mag. 15 (6) (1998) 23–50.

[26] J. Nash, Equilibrium points in *n*-person games, in: Proc. of the National Academy of Sciences, Vol. 36, 1950, pp. 48–49.

[27] M. J. Osborne, A. Rubinstein, A Course in Game Theory, The MIT Press, 1994.

[28] R. S. Sutton, A. G. Barto, Reinforcement Learning: An Introduction, The MIT Press, Cambridge, MA, 1998.

[29] D. Fudenberg, D. K. Levine, The Theory of Learning in Games, MIT Press, Cambridge, MA, 1998.

[30] J. R. Smith, IBM multimedia analysis and retrieval system (MARVEL), http://mp7.watson.ibm.com/marvel/.

[31] M. Campbell, A. Haubold, M. Liu, A. Natsev, J. R. Smith, J. Tesic, L. Xie, R. Yan, J. Yang, IBM Research TRECVID-2007 video retrieval system, in: TRECVID Workshop,, Gaithersburg, USA, 2007.