THE PENNSYLVANIA STATE UNIVERSITY
SCHREYER HONORS COLLEGE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

IMPACT OF FEEDBACK DELAY ON PREDICTING HIT RATE

ZHAOHONG LYU
FALL 2019

A thesis
submitted in partial fulfillment
of the requirements
for baccalaureate degrees
in Computer Science and Mathematics
with honors in Computer Science

Reviewed and approved* by the following:

Thomas F La Porta
Professor of Computer Science and Engineering
Thesis Supervisor

Jesse Louis Barlow
Professor of Computer Science and Engineering
Honors Adviser

*Signatures are on file in the Schreyer Honors College.

Abstract

Feedback is used in many network algorithms and networked applications. Yet, in most cases, feedback is not received immediately; instead, feedback delay is very common due to the nature of network communication. In the worst-case, the network can be congested, resulting in a huge feedback delay, which is likely to bring negative impacts to the provided service. In this thesis, we mainly focus on a dual-path Network Utility Maximization (NUM) framework, which targets the optimization of resource allocation for image crowd-processing. As part of this system, an estimate of hit rate is used to allocate resources. We consider the impact of feedback delay on predicting hit rate, which can be a threat to the NUM performance.

Table of Contents

List of Figures **iii**

List of Tables **iv**

Acknowledgements **v**

1 Introduction **1**

2 Background and Related Work **4**

2.1 TCP Feedback 5

2.2 NUM Feedback 6

2.3 Wheatman's Dual-path NUM 8

3 Data Collection and Algorithm **13**

3.1 Data Analysis 14

3.2 Counter Algorithm 16

3.3 Feedback with Delays 20

4 Results with Dual Path **26**

5 Conclusion **32**

Bibliography **34**

List of Figures

2.1 The System Model 9

3.1 Trace of Real Cases 15
3.2 Prediction results of images in trace1 without delay 17
3.3 Prediction results of images in trace2 without delay 18
3.4 Prediction results of images in trace3 without delay 19
3.5 Prediction results of images in trace1 with delay 21
3.6 Prediction results of images in trace2 with delay 22
3.7 Prediction results of images in trace3 with delay 23
3.8 Behaviors of three counters in scenario of trace2 with delay=2 24
3.9 Behaviors of Counter3 [0,1] in scenario of trace2 with delay=1/2/4/10 25

4.1 Prediction results of images in trace1 with different cloud delay 28
4.2 Prediction results of images in trace2 with different cloud delay 29
4.3 Prediction results of images in trace3 with different cloud delay 30

List of Tables

2.1 Nomenclature and Notation ’1 6

2.2 Nomenclature and Notation ’2 10

Acknowledgements

I thank Thomas F La Porta, Mark Mahon, Kristina Sorensen Wheatman and Fidan Mehmeti for supporting me to participate in this research.

Chapter 1

Introduction

Feedback responses are essential in many applications for several purposes, e.g., confirmation and reinforcement training. Yet in the network algorithms, feedback might experience congestion issues, resource allocations issues, and rate control issues that degrade network performance.

Congestion control is the type of algorithm that requires feedback in a network. The Transmission Control Protocol (TCP) [1], first described by Vint Cerf and Bob Kahn in 1974, is one of the protocols in the transport layer of the Internet protocol suite that uses feedback for congestion control and other functions. TCP guarantees that all the packets are transmitted in order and error-free between hosts. We will talk more about TCP in the next chapter.

Resource allocation algorithms also use feedback for making decisions. In a traditional Internet model, there are billions of connected computing devices. They are hosts (end systems) running network apps, such as PCs, smartphones and even servers owned by institutions. However, network resources are limited; we have a fixed amount of fibers, and a fixed amount of base stations, etc. How to allocate the network resources properly is a challenge. To help optimize resource allocation, algorithms in networks often rely on feedback.

As we have limited network resources, e.g., bandwidth, routers and switches, people have developed many models to control transmission rates based on allocated resources. One of these models is the network utility maximization (NUM) model, first proposed by Kelly in his seminal works [2] and [3]. In the NUM framework, we are trying to maximize the aggregated utility of all users. The assumption on the utility function is that each U_i is an increasing concave function of the transmission rate. There are several constraints associated with the utility function. The model introduces "shadow prices" for every resource, e.g. wireless links. Each resource has a maximum capacity and current capacity in use. Intuitively, the idea is that, as the current capacity gets closer to the maximum, the shadow price of the corresponding resource increases so that users need to "pay" more if they continue using this resource. On the other hand, when the resource is relatively idle, the shadow price decreases, so as to attract users and to fully utilize as many capacities as possible. For information updates between system and users, feedback is required to communicate the prices to the user.

Feedback may also be used at the application level. In an application we consider in the thesis, the objective is to collect every image containing an object of interest from a set of mobile devices as fast as possible. We use the estimated hit rate of a device for improving our resource allocation of limited network and processor resources. Intuitively, there might be different profiles of users that are offloading images at the same time. As the network resource is limited, we can allocate more resources to users we predict to generate a high amount of hits. This requires receiving feedback to determine if an image that has been processed is a hit or miss. If the image is processed remotely on a server, this feedback may be delayed. Without this feedback information, the system might have difficulties in allocating resources for processing the next image. If a client is marked as having a lower hit rate, it might receive fewer resources and experience a long processing time. This illustrates the need for fast accurate feedback of hit rate in our system.

In this thesis, we consider issues with feedback in the related system. For example, network communications have physical limitations. The feedback delay issue is a potential threat in our model since we cannot get the latest feedback to make the prediction, which affects the performance of algorithms negatively. In consideration of late feedback, some algorithms may be sub-optimal or others may be harmful. We do not consider cases of no feedback and incorrect feedback, which occur when packets are lost or failed in error-checking during communication.

This thesis is outlined as follows. The second chapter is the background and related work. We show an overview of the most relevant work. In the third chapter, we present our data analysis and algorithms for predicting hit rates based on that analysis. The fourth chapter shows the result of algorithms when delay is introduced. The fifth chapter is the conclusion.

Chapter 2

Background and Related Work

In this chapter, we research some algorithms that require feedback to improve performance. Specifically, we research TCP congestion control and the Network Utility Maximization algorithm. We then design the system we used for our example for image processing.

2.1 TCP Feedback

TCP, known as a reliable data transfer protocol, requires feedback to confirm if the packet chunk is correct and in order [4]. One mechanism of TCP is to encapsulate the sequence number (SEQ) and the acknowledgment number (ACK) in the TCP segment, where both numbers are 32 bits. The existence of the sequence number ensures the reconstruction of packets in ascending order. And the acknowledgment number notifies the receiver that the sender is expecting the data with the sequence number equal to the acknowledgment number. For example, if a host, A, sends a packet with SEQ=100, A expects to receive a feedback acknowledgment with ACK=101 in a certain time window. However, if A receives feedback with ACK=100 or A does not hear back from the receiver within the time window, this information implies that A needs to retransmit the packet with SEQ=100. These two circumstances are also known as retransmission timeout (RTO) and duplicate cumulative acknowledgments (DupAcks).

Every time the sender sends out the packet, it sets up an automatic timer waiting for feedback acknowledgments [5]. When the network resource is limited and the network gets congested, feedback is delayed and can not be sent back within the timer. Due to the RTO mechanism, the sender has to send the packet again, which costs more resources. In other words, it is bad and harmful when feedback delay is severe, resulting in retransmission.

To avoid feedback congestion and to achieve high performance, TCP uses additional mechanisms. For example, TCP limits the total number of unacknowledged packets by maintaining a congestion window for every connection [6]. At first, the size of the congestion window is relatively small. Then TCP utilizes a technique called “slow start” to increase the congestion window either after a new connection is initialized or after a timeout. By deploying the congestion window

with the slow start strategy, TCP prevents the link between senders and receivers from becoming overloaded.

Meanwhile, the self-clocking TCP [7] mechanism guarantees that new transmissions are paced by returning ACKs. In order words, the transmission rate is equal to the available bandwidth of the bottleneck link. There are many other mechanisms and algorithms, such as fast retransmit and calculations on round-trip time (RTT), to ensure TCP is a robust protocol for high performance. Hence, TCP relies on feedback for reliable data transfer, coupled with congestion control mechanisms to avoid high feedback delay.

2.2 NUM Feedback

The NUM framework is another way to perform rate control. Here we describe the high order logic for a general network utility maximization (NUM) framework [2]. First, each user has a utility function where the utility is a function of rate. The network sets a price to transmit as a function of rate. The prices are higher as the network becomes congested. The user sets its rate to maximize its utility. Users need feedback updates of prices, and the network needs feedback updates of user rates.

Table 2.1 summarizes the notation used in this section. [8]

i	user i
x_i	rate of individual user i
U_i	utility of individual user i
U	total utilities of all users
ℓ	link ℓ
R_i^l	indicator variable of link ℓ occupied by user i or not
c_l	maximum capacity of link ℓ
α	step-size value
t	iteration index

Table 2.1: Nomenclature and Notation

We formulate the basic NUM [2] by setting the objective function 2.1, where $i = 1, \ldots, I$ users, $\ell = 1, \ldots, L$ links and the individual user rates are given as $\vec{\mathbf{x}} = \{x_1, \ldots, x_I\}$, (from a

convex set).

SYSTEM $(\mathbf{U}, \vec{\mathbf{x}}; \mathbf{R}, \vec{\mathbf{c}})$ [9]:

$$\begin{aligned} &\underset{\vec{\mathbf{x}} \geq 0}{\text{maximize}}\ \mathbf{U}(\vec{\mathbf{x}}) = \sum_{i=1}^{I} U_i(x_i) \\ &\text{subject to } \sum_{i=1}^{I} R_i^{\ell} x_i \leq c_{\ell},\ \ \forall\, \ell, \end{aligned} \tag{2.1}$$

$$R_i^{\ell} = \begin{cases} 1 & \text{user } i \text{ uses resource } \ell \\ 0 & \text{otherwise} \end{cases},$$

Note that the utility function takes the form $U_i(x_i) = \log(x_i + 1)$, where $(x_i + 1)$ ensures $\log$ gives positive utility. Depending on the different demands, the utility function might vary. If the utility functions are strictly concave functions, the problem has a unique centralized solution [10].

Result 1 *According to Vo's work [10], the optimal solution at each iteration is [9]*

$$x_i^* = \frac{1}{\sum_{\ell=1}^{L} R_i^{\ell} \lambda_{\ell}^*} \quad \forall\, i, \tag{2.2}$$

where the shadow prices λ_{ℓ}^* *are found from the slackness conditions* $\lambda_{\ell}^*(c_{\ell} - \sum_i R_i^{\ell} x_i^*) = 0$*,* $\forall\, \ell = 1, \ldots, L$*, and* $\vec{\lambda^*} \geq 0$*.*

However, this centralized solution can be inefficient, especially in a large-scale environment, i.e. too many users and requests. Practically, we can take an iterative process to solve the shadow price $\vec{\lambda}$ according to the gradient descent method [8].

$$\lambda_{\ell}(t+1) = \left[\lambda_{\ell}(t) - \alpha \left(c_{\ell} - \sum_{i=1}^{I} R_i^{\ell} x_i(t) \right) \right]^{+}, \forall\, \ell, \tag{2.3}$$

where $[\cdot]^+$ denotes the absolute value, t denotes the iteration index, and $\alpha > 0$ denotes a sufficiently

small positive step-size parameter.

Please refer to Kelly's journal paper [2] and Palomar's journal paper [8] for detailed proofs on the centralized and decentralized solutions to such kind of problems. By solving these equations, the system will achieve the maximum sum utility.

In the equation 2.2, we observe that each user requires the feedback of the shadow price to update its rate. Equation 2.3 shows that NUM also requires feedback of user rates to update the shadow price of network resources. Ideally, we treat t as the iteration index, so for each iteration, the user gets the updated information of shadow prices from the previous iteration and the NUM system gets the updated information of user rates. However, if feedback delay is present, the user might receive no feedback on shadow prices, which leads to a sub-optimal decision of rates. If the system does not get information about the updated user rates on time, it may result in sub-optimal pricing.

2.3 Wheatman's Dual-path NUM

The image processing system we used in this thesis allocates rates based on NUM. The system is shown in Figure 2.1 [9]. In this system, a requester sends a query to a set of mobile devices to collect images with an object of interest. Then the system determines the resource allocation, based on the shadow prices of resources and hit rates of users, to achieve the highest aggregated utility.

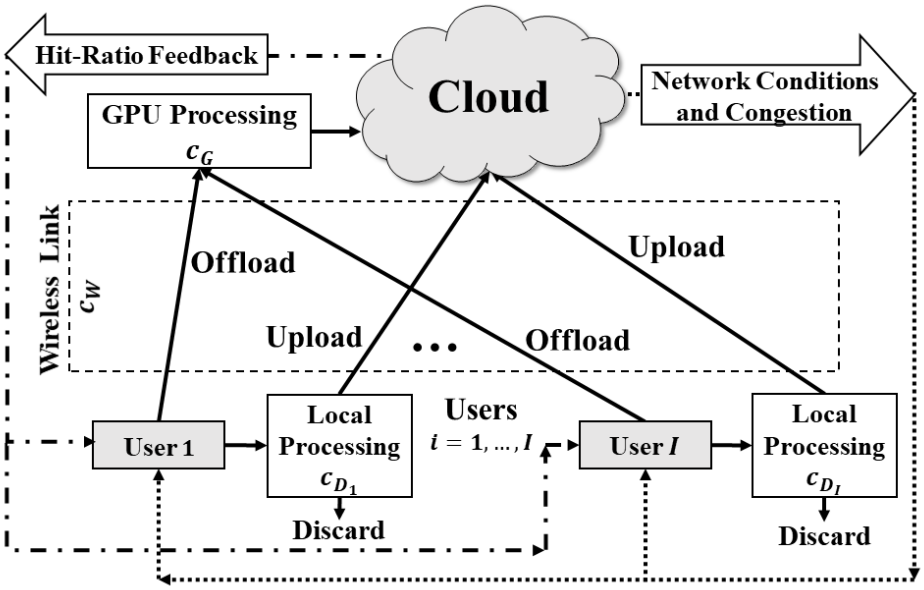

Figure 2.1: The system model for mobile crowdsourcing using a dual-path approach incorporating proposed hit rates. [9]

The system can be applied in many scenarios, such as law enforcement and emergency response. For example, a request may be broadcasted to the public to collect an image with an object of interest so that the requester can gather as much information as possible on the details of the interested object, e.g. the last location, the last seen timestamp. Our objective is to collect the images that contain the object of interest as fast as possible from the perspective of the cloud server.

Users have images stored to be processing. For each image, the device will decide either to offload it to the cloud for processing or process it locally. The decisions are based on the transmission rate, the congestion level of wireless links and network processors, etc. Networks use NUM to set prices of wireless links and the cloud. Users have utility functions that consider the expected hit rate. The system can only predict the hit rate from previous results from feedback.

The system uses a dual-path version of NUM [9]. One path is to upload the image to the cloud for processing, and the other is to process the images locally and then upload those with a hit. We present the formulation of Dual-path NUM with hit rate consideration, which is extended from basic NUM. Table 2.2 is an extension table of Table 2.1 for additional variables.

h_i	hit rate of user i
θ_{ij}	percentage of rate of link j out of the total rate of user i
j	path j
J_i	paths of user i
R_{ij}^l	indicator variable of path j for user i passes over link l

Table 2.2: Nomenclature and Notation

According to Wheatman's paper [9], we extend the basic NUM to this NUM formulation for the Multi-Path case.

SYSTEM^M $(\mathbf{U}, \mathbf{X}; \mathbf{R}, \vec{\mathbf{c}})$:

$$\begin{aligned} &\underset{\vec{\mathbf{x}} \geq 0}{\text{maximize}}\ \mathbf{U}(\mathbf{X}) = \sum_{i=1}^{I} U_i \left(\sum_{j=1}^{J(i)} x_{ij} \right) \\ &\text{subject to} \sum_{i=1}^{I} \sum_{j=1}^{J(i)} R_{ij}^{\ell} x_{ij} \leq c_\ell, \ \ \forall\, \ell. \end{aligned} \tag{2.4}$$

$$R_{ij}^{\ell} = \begin{cases} 1 & \text{route of path } j \text{ for user } i \text{ passes over link } \ell \\ 0 & \text{otherwise} \end{cases}.$$

Note that the matrix values $(\mathbf{X})_{ij} = x_{ij}$ refer to all the rates given to user i over path j. However, since the objective function 2.4 is a non-strictly concave function [10], we need to approximate the optimization problem $SYSTEM^M$, similar to what has been done in [10].

The new utility function becomes $U_i \left(\sum_{j=1}^{J} x_{ij} \right) = \sum_{j=1}^{J} \theta_{ij} \tilde{U}_i \left(\frac{x_{ij}}{\theta_{ij}} \right)$, where $\theta_{ij} = \frac{x_{ij}}{\sum_{k=1}^{J(i)} x_{ik}} \quad \forall\, i, j$. If $\sum_{k=1}^{J(i)} x_{ik} = 0$, we define $\theta_{ij} = 1$, so that $\frac{x_{ij}}{\theta_{ij}}$ is valid. And the new weighted utility function for each user i becomes $\tilde{U}_i \left(\frac{x_{ij}}{\theta_{ij}} \right) = \theta_{ij} \log \left(\frac{x_{ij}}{\theta_{ij}} \right)$.

Consequently, the parametrized approximation problem becomes [9]

$\widetilde{\textbf{SYSTEM}}^{M}\ (\tilde{\mathbf{U}}, \mathbf{X}, \boldsymbol{\Theta}; \mathbf{R}, \vec{\mathbf{c}})$:

$$\begin{aligned} &\underset{\vec{\mathbf{x}} \geq 0}{\text{maximize}}\ \tilde{\mathbf{U}}(\mathbf{X}, \boldsymbol{\Theta}) = \sum_{j=1}^{J(i)} \theta_{ij} \log\left(\frac{x_{ij}}{\theta_{ij}}\right) \\ &\text{subject to} \sum_{i=1}^{I} \sum_{j=1}^{J(i)} R_{ij}^{\ell} x_{ij} \leq c_{\ell}, \quad \forall\, \ell. \end{aligned} \tag{2.5}$$

Result 2 *Optimal values of rates x_i^* becomes [10]*

$$x_{ij}^*(t) = \frac{h_i \theta_{ij}^*}{\sum_{\ell=1}^{L} R_{ij}^{\ell} \lambda_{\ell}^*} \quad \forall\, i, j, \tag{2.6}$$

The full proof of these processes is introduced in [10] and [11].

In practice, the iterative decentralized approach to update the θ_{ij}

$$\theta_{ij}(t+1) = \frac{x_{ij}(t)}{\sum_{k=1}^{J(i)} x_{ik}(t)} \quad \forall\, i, j. \tag{2.7}$$

Finally, the new formula of shadow prices is

$$\lambda_{\ell}(t+1) = \left[\lambda_{\ell}(t) - \alpha \left(c_{\ell} - \sum_{i=1}^{I} \sum_{j=1}^{J(i)} R_{ij}^{\ell} x_{ij}(t)\right)\right]^{+}, \forall\, \ell, \tag{2.8}$$

We summarize Wheatman's algorithm [9] for readers' interests on how the system works. Note that we modify the algorithm by discarding the energy consideration.

Algorithm 1: Dual-path Hit-rate NUM Considerations (DH-NUM)

Data: Initialize at a feasible point within the domain $\mathcal{S}'$, such that $0 < \mathbf{X}(0) \ll \max(c_\ell)$ and $\lambda_\ell(0) = \frac{1}{\sum_i \sum_j R_{ij}^\ell}$. Start at $t = 1$ with default termination at $t = T$.

while $t \leq T$, **do**

  for $i = 1, \ldots, I$, **do**

    for $j = 1, \ldots, J(i)$, **do**

      User i updates rate x_{ij} for path j as Eq.(2.6);

      Obtain $\theta_{ij}(t+1)$ from Eq. (2.7);

    end

  end

  for $\ell = 1, \ldots, L$, **do**

    Resource ℓ updates price λ_ℓ according to Eq.(2.8);

  end

  Set $t \leftarrow t + 1$;

end

Wheatman's experiment [9] shows that this dual-path system always provides the highest aggregate utility, compared to the system in which all images are processed locally on the mobile devices (Always Local) and the system in which all images are offloaded to the GPU for processing (Always Offload).

Result 2 shows the formula 2.6 is a function of hit rate. If a user has a high hit rate, we allocate more resources to that user since it is more likely to collect the images with the object of interest, and vice versa. Even though users can get feedback from the local processor immediately, users need the feedback from the remote processor as well, so that users have full knowledge of the set of images. Similar to the feedback on shadow prices and user rates, feedback on the hit rate is another important factor of the performance for dual-path NUM. If hit rate feedback is congested and delayed, users fail to get the latest updates, which may degrade the aggregated NUM utility. In the next chapter, we focus on hit rate feedback with real-life traces.

Chapter 3

Data Collection and Algorithm

3.1 Data Analysis

Because we use the expected hit rate to help determine resource allocation in our system, the accurate prediction of hit rate is critical for the operation of the algorithms. When looking at real-world image data sources, we noticed several images in a cluster will likely have an object of interest, and then a subsequent cluster will not. We will call these clusters runs with streaks of "hits" and "misses", where the presence of the object indicates a "hit" while the lack indicates a "miss." This phenomenon occurs because of either the object of interest moves or the camera moves. For example, if we are looking for a red truck, the truck can be captured by one station camera in a few seconds, then it runs out of the scene. If we are taking pictures during travel, we can take photos of one spot, then we move on to the next spot.

To test this conjecture, we captured image data from several stationary cameras that record video of sidewalks and roads at a major research university [12]. Images were simultaneously extracted at ten-second intervals from 6 camera positions for approximately 45 minutes. Hence, we obtained six sets of 273 images that we call traces. We then analyzed the images gathered for multiple objects of interest by hand. If the image contains the object of interest, we mark 1 (i.e. hit) in the trace; otherwise, we use 0 (i.e. miss). The results confirm our conjecture. Note that we use the first 100 images instead of the entire set of 273 images in our experiment for simplicity.

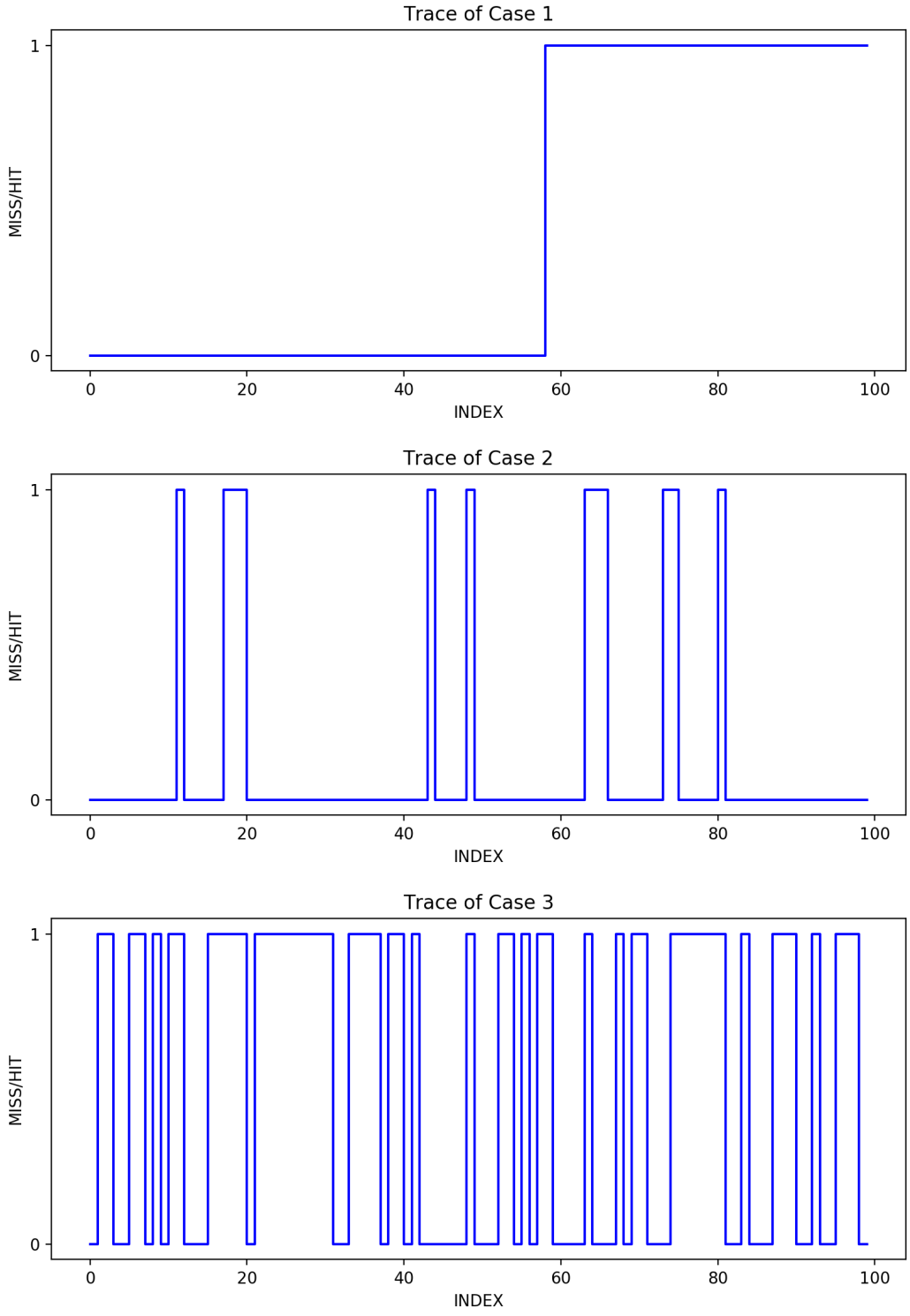

Figure 3.1: Three types of traces from the real world

In Figure 3.1, we show three representatives of traces with different characteristics. The first trace (top) shows a long run of misses, followed by a long run of hits, which can be some static

objects in interest. In this case, the trace looks simple and easy to make a prediction. The second trace (middle) shows a trace with mostly misses, but with a number of short runs of hits, which can be some moving objects. In this case, it is easy to predict misses but challenging to predict hits since the sequence gets misses most of the time. The third trace (bottom) shows misses and hits are flipping, which can be some dynamic objects. If the sequence follows this characteristic, it is hard to predict correctly without a priori knowledge of classification.

3.2 Counter Algorithm

Based on the analysis above, we developed three classes of image predictor. We used straight counting methods in which we incremented the counter for a hit and decremented the counter for a miss, coupled with a threshold. Every time the system predicts whether the next image is hit or miss, it checks the current counter. If the counter is positive, we predict the next image as a hit; otherwise, we predict the next image as a miss. Note that the counter never exceeds the threshold. For instance, if the counter is -2 and the upcoming feedback result is miss, the counter will stay at -2.

We call these three classes of image predictor as Counter1, Counter2 and Counter3. Counter1 [-1, 2] uses the upper bound 2 and the lower bound -1, which is least reactive. Counter2 [0, 2] uses the upper bound 2 and the lower bound 0, which is more reactive than the first. Counter3 [0, 1] uses the upper bound 1 and the lower bound 0, which should be the most reactive. Intuitively, Counter3 is just making the prediction based on the result of the previous image.

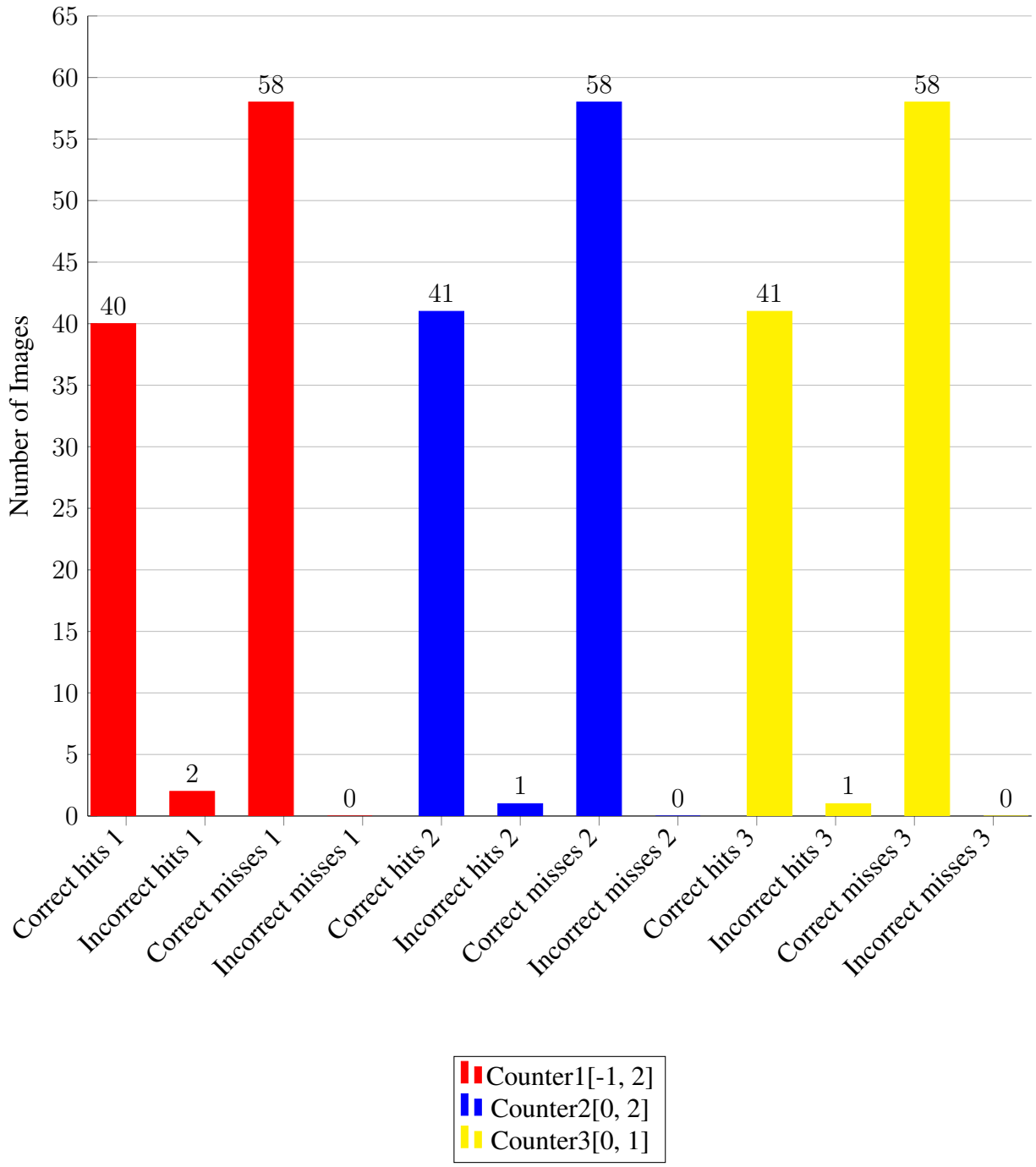

Figure 3.2: Prediction results of images in trace1 without delay

In Figure 3.2, we show the performance of the three counters on trace1. The three counters behave similarly. Counter1 with threshold [-1, 2] iterates from -1 to 0, then from 0 to 1, so it misses the first two hits during the transition. The other two counters miss only the first hit. This shows that counting methods work well when encountering traces with similar characteristics as

trace1. We will look at the comparison results for trace2 and trace3.

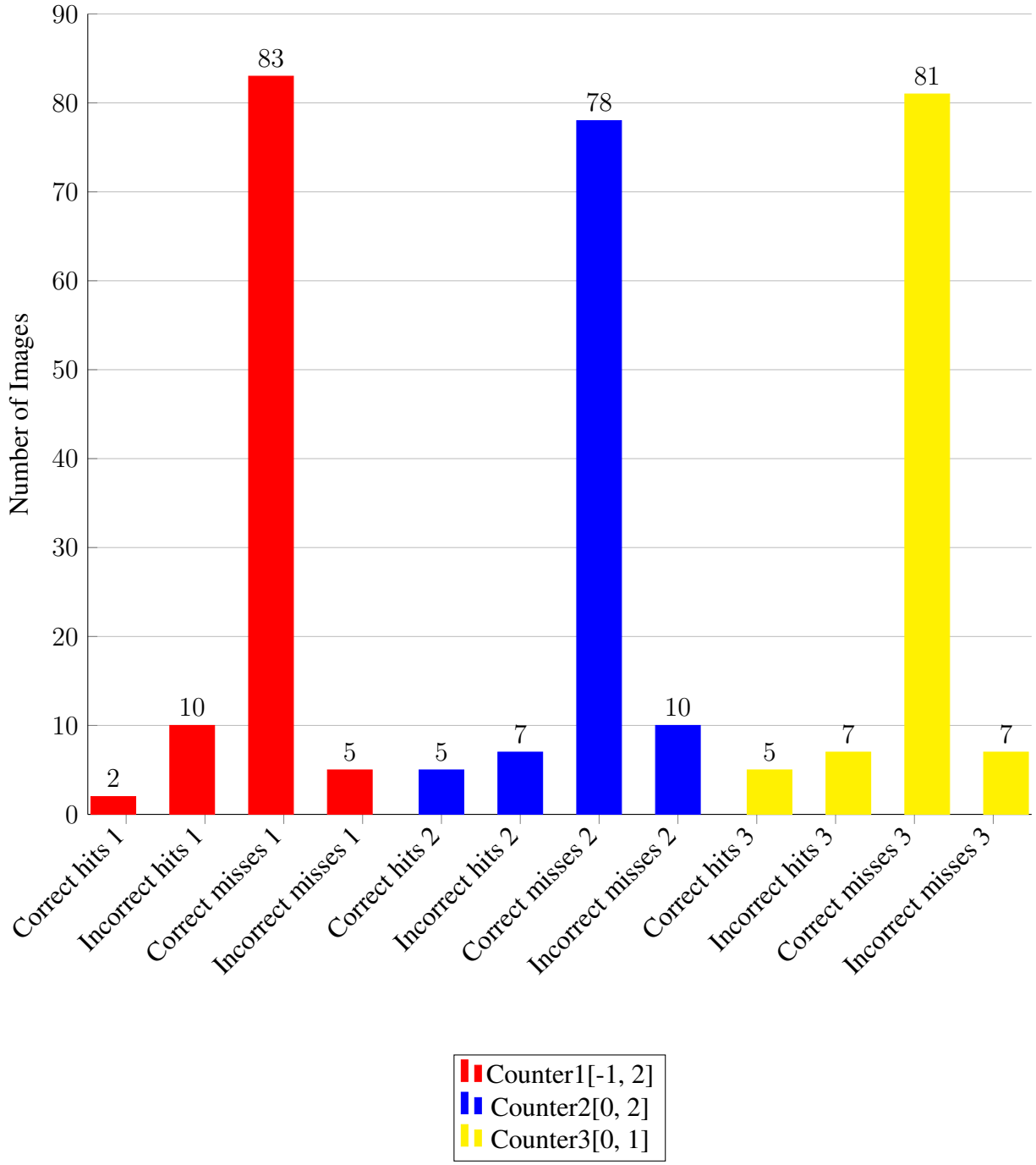

Figure 3.3: Prediction results of images in trace2 without delay

In Figure 3.3, we show the results for trace2. The three counters show different behaviors, but still, they achieve about 85% correctness overall. Counter1 with threshold [-1, 2] makes more

incorrect predictions on hits. In other words, it usually gives “miss” predictions when the truth is “hit”. On the other hand, Counter2 with threshold [0, 2] is less accurate on predictions on misses, because it is not as reactive when runs change from hits to misses. Counter3 with threshold [0, 1] makes mistakes but without obvious biases on incorrect hits or incorrect misses.

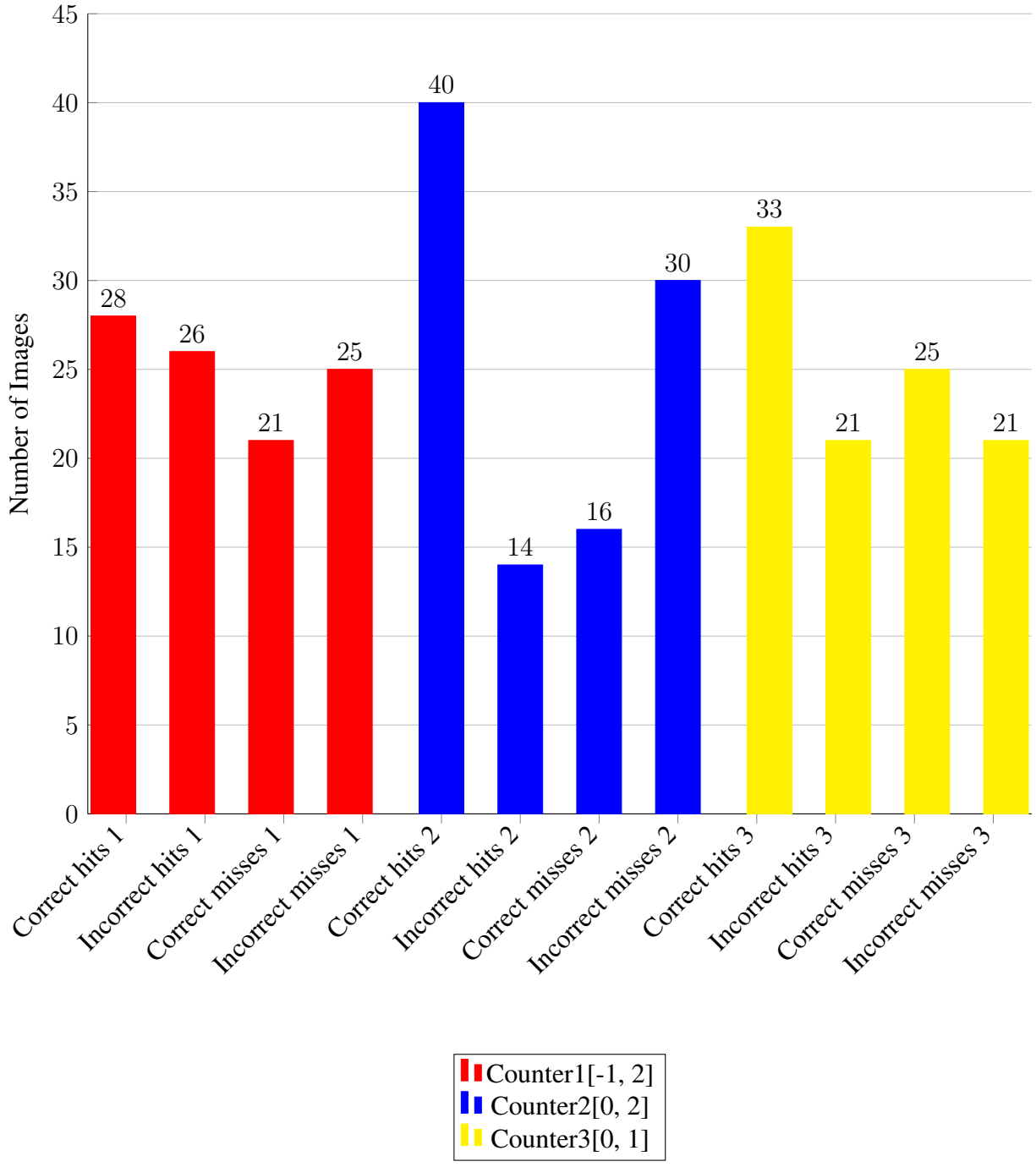

Figure 3.4: Prediction results of images in trace3 without delay

In Figure 3.4, we show the results with trace3. This is a difficult trace to predict because the runs of hits and misses are short. Counter1 with threshold [-1, 2] predicts mistakenly about half of the time, which is not different from guessing. Counter2 with threshold [0, 2], again, shows similar characteristics on prediction bias. The reason is that it is more likely to make a "hit" prediction as when the counter is 1 or 2, it predicts the next image as a hit, but only when the counter is 0, it predicts the next image as a miss. The results show it predicts 74% of hits correctly, but only 35% of misses. Counter3 with threshold [0, 1] achieves 58% of overall correctness, which is the highest among these 3 counters. Counter3 with threshold [0, 1] is the only one that achieves 50% on both "hit" and "miss" prediction. This is because it is highly reactive and catches transition during short runs.

To conclude, we can see that all three sets of threshold work well, depending on the length of the runs of hits and misses. Next, we see how three algorithms will work if the feedback is delayed, as many be the case of images as processed remotely.

3.3 Feedback with Delays

We will look into the scenario while varying the feedback delay from 2 to 4 to 10. For example, when the delay is 2, if we are predicting image 10, we only have results of image 1 to 8, but we have no information on image 9 due to the delayed feedback. Intuitively, when the delay is 1, we have all previous feedback results, i.e. without delay.

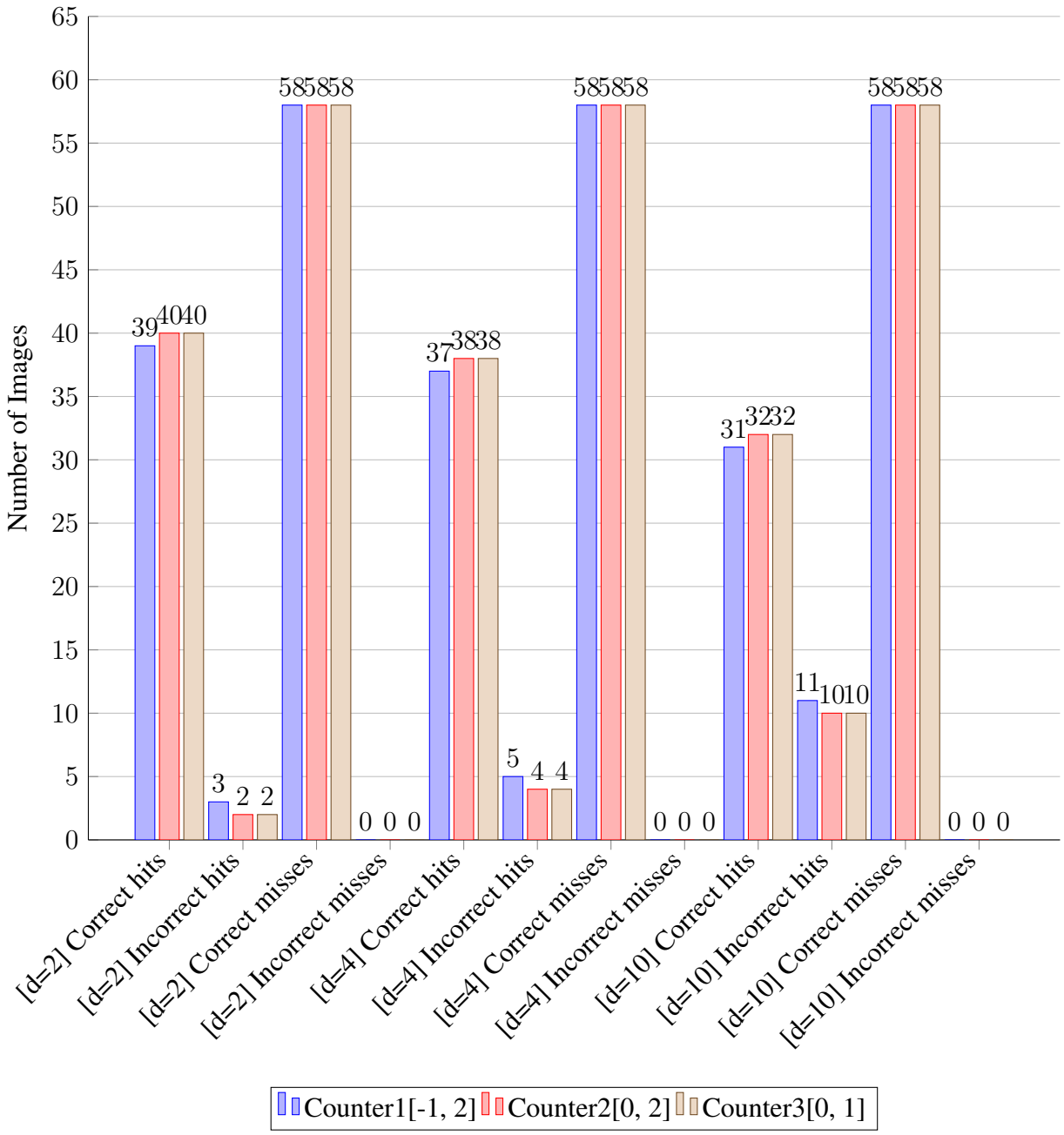

Figure 3.5: Prediction results of images in trace1 with delay

In Figure 3.5, we show the results including all three counting methods on trace1 with the delay from 2 to 4 to 10. Delay increases from left to right, shown in the figure. When the delay increases, counters make more “bad” predictions as shown in the increasing incorrect hits. But overall, the counting methods work well in this case.

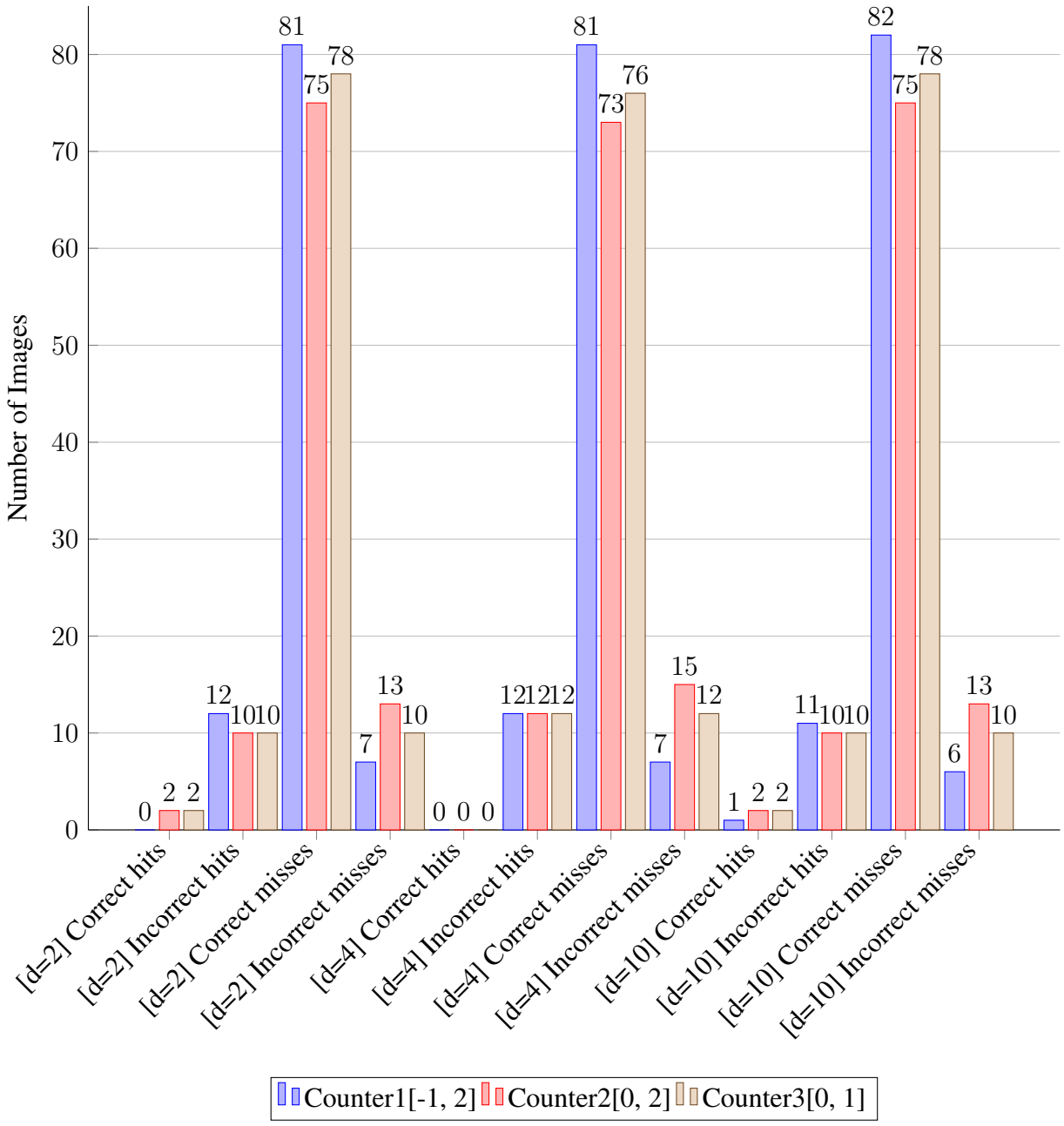

Figure 3.6: Prediction results of images in trace2 with delay

Figure 3.6 shows the results including all three counting methods on trace2 with the delay from 2 to 4 to 10. We can see that the counting methods perform poorly for predicting hits because the length of runs for the hits is shorter than the delay feedback. Comparing to the performance of prediction without delay, it is clear that the performance of the counting methods is degraded in the presence of delay. But it is surprising to see that the number of correct hits in d=10 (i.e.

feedback delay is 10) is larger than the one in d=4 (i.e. feedback delay is 4). As we investigate this phenomenon by looking at the detailed logs of the experiment, we find out that the counter makes correct hits prediction by luck. The delayed cluster of hits prediction coincides with the current cluster of hits.

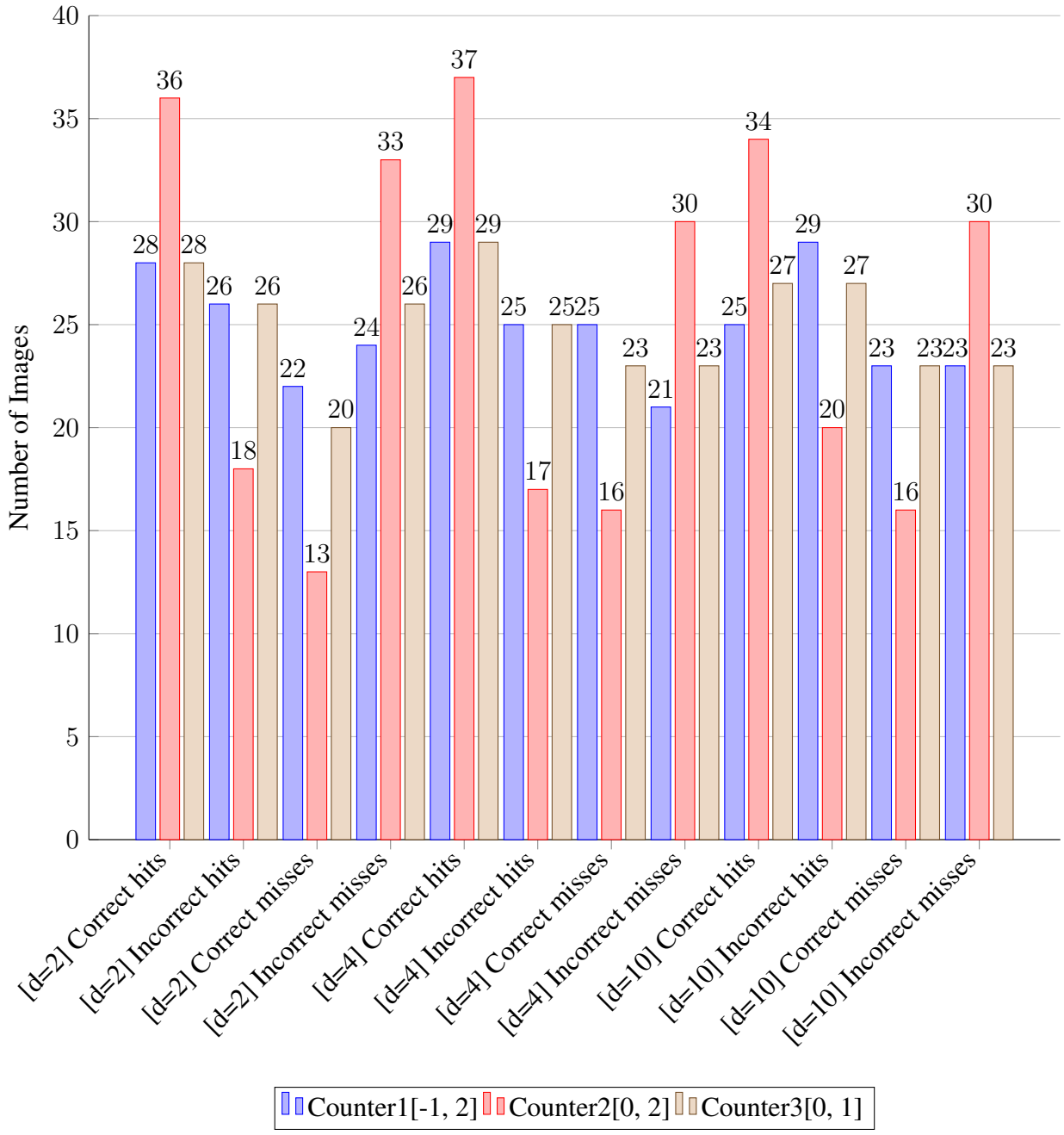

Figure 3.7: Prediction results of images in trace3 with delay

Figure 3.7 shows the results including all three counting methods on trace3 with the delay from

2 to 4 to 10. Considering three different delays together, we observe that Counter1 with threshold [-1, 2] has 50.6% correctness on “hit”, and 50.7% correctness on “miss”; Counter2 with threshold [0, 2] has 66.0% correctness on “hit”, and 32.6% correctness on “miss”; Counter3 with threshold [0, 1] has 51.9% correctness on “hit”, and 47.8% correctness on “miss”. We can see that the counting methods perform badly in trace3 with delay, so we might need to consider some other methods (e.g. weighted average method).

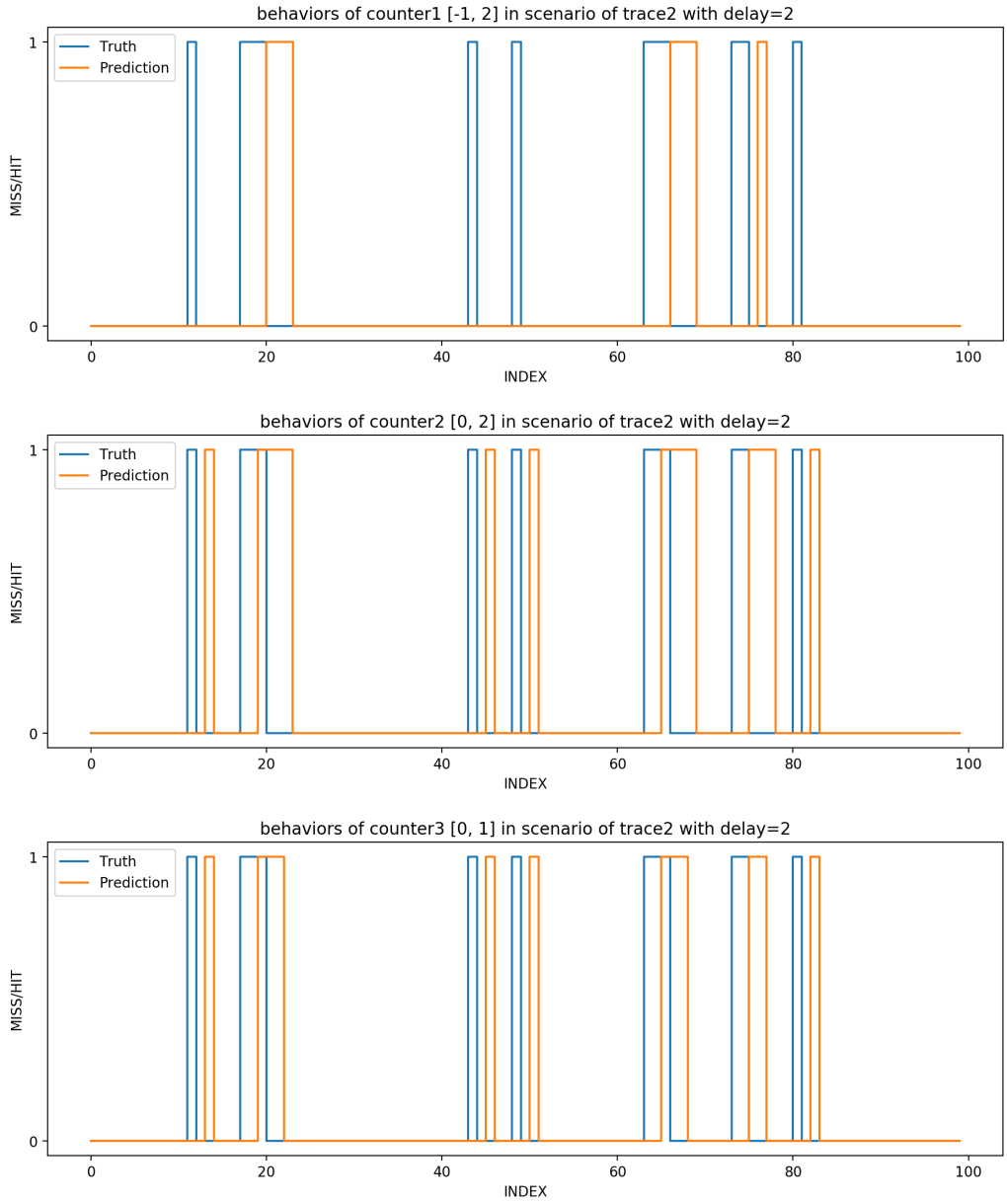

Figure 3.8: Behaviors of three counters in scenario of trace2 with delay=2

Figure 3.8 reveals different reactiveness between three counters. As we reduce the size of the threshold from [-1, 2] to [0, 2], we can see that Counter2 starts responding to shorter spikes of hits (2 continuing sequence of hits.) As we further reduce the threshold from [0, 2] to [0, 1], we can see that Counter3 does better and is the most reactive counter, regardless of the presence of feedback delay.

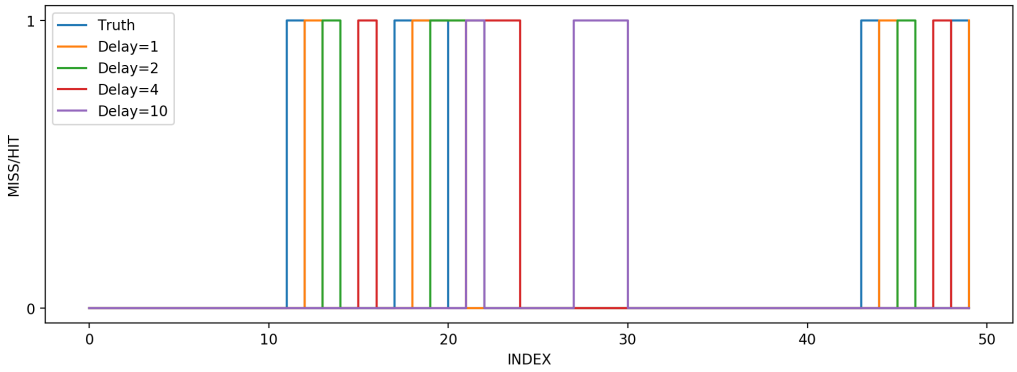

Figure 3.9: Behaviors of Counter3 [0,1] in scenario of trace2 with delay=1/2/4/10

In Figure 3.9, we show how delay influences the predictions of counting methods. The waves are held back for certain iterations due to the effect of feedback delay. Because of the shift of waves, the performance of prediction is greatly degraded.

Based on these results, it is clear that feedback delay has negative impacts on prediction. We will choose Counter3 with threshold [0, 1] for the following analysis on the dual-path experiment since Counter3 with threshold [0, 1] presents the most robust results in the experiments with and without delay.

Chapter 4

Results with Dual Path

In this chapter, we will perform some tests to show how our counting method works in a dual-path system. Note that we isolate the predictor from the real NUM framework here, but we simulate different scenarios that would happen in the dual-path NUM framework.

To simulate the dual-path system tests, we need to introduce one more variable, the period, for which we send images on the remote path and the local path. The remote path will lead to higher feedback delays. When the period is 2, we send every other image to the local processor. When the period is 3, we send every 3rd image to the local processor. In order words, we send 2 images to the cloud, then send 1 image to local for processing. Note that now we have two delay parameters since we have two paths: remote and local. We assume the local delay is 1 (i.e. no delay for local processing) since processing images locally would not produce any feedback delay. On the other hand, we vary the remote delay for simulating different levels of network congestion.

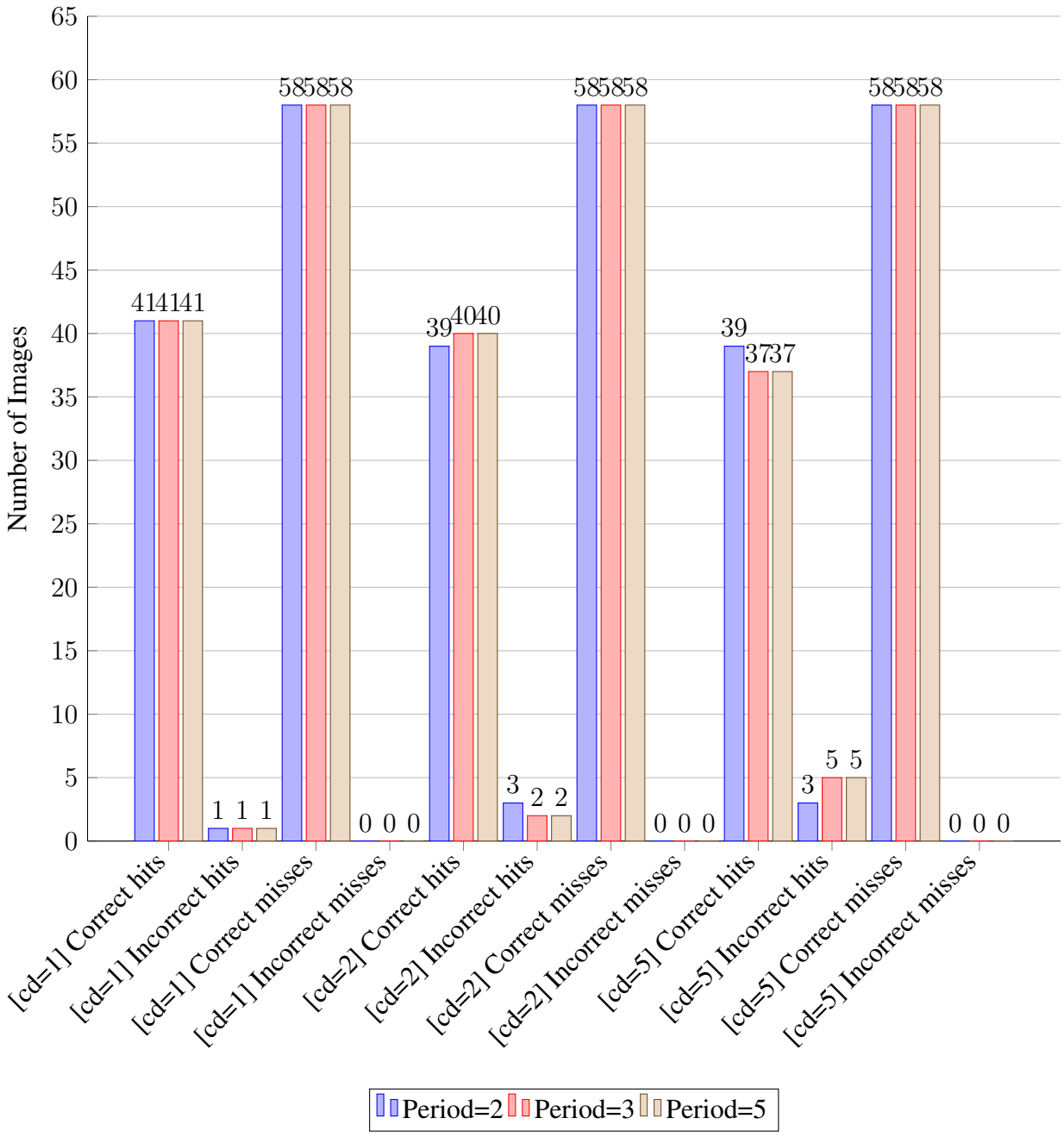

Figure 4.1: Prediction results of images in trace1 with different cloud delay

In Figure 4.1, we show the results including all three periods on trace1 with cloud delay from 1 to 2 to 5. For traces with similar characteristics of trace1, our counting method is robust among different periods, even when the cloud delay is large.

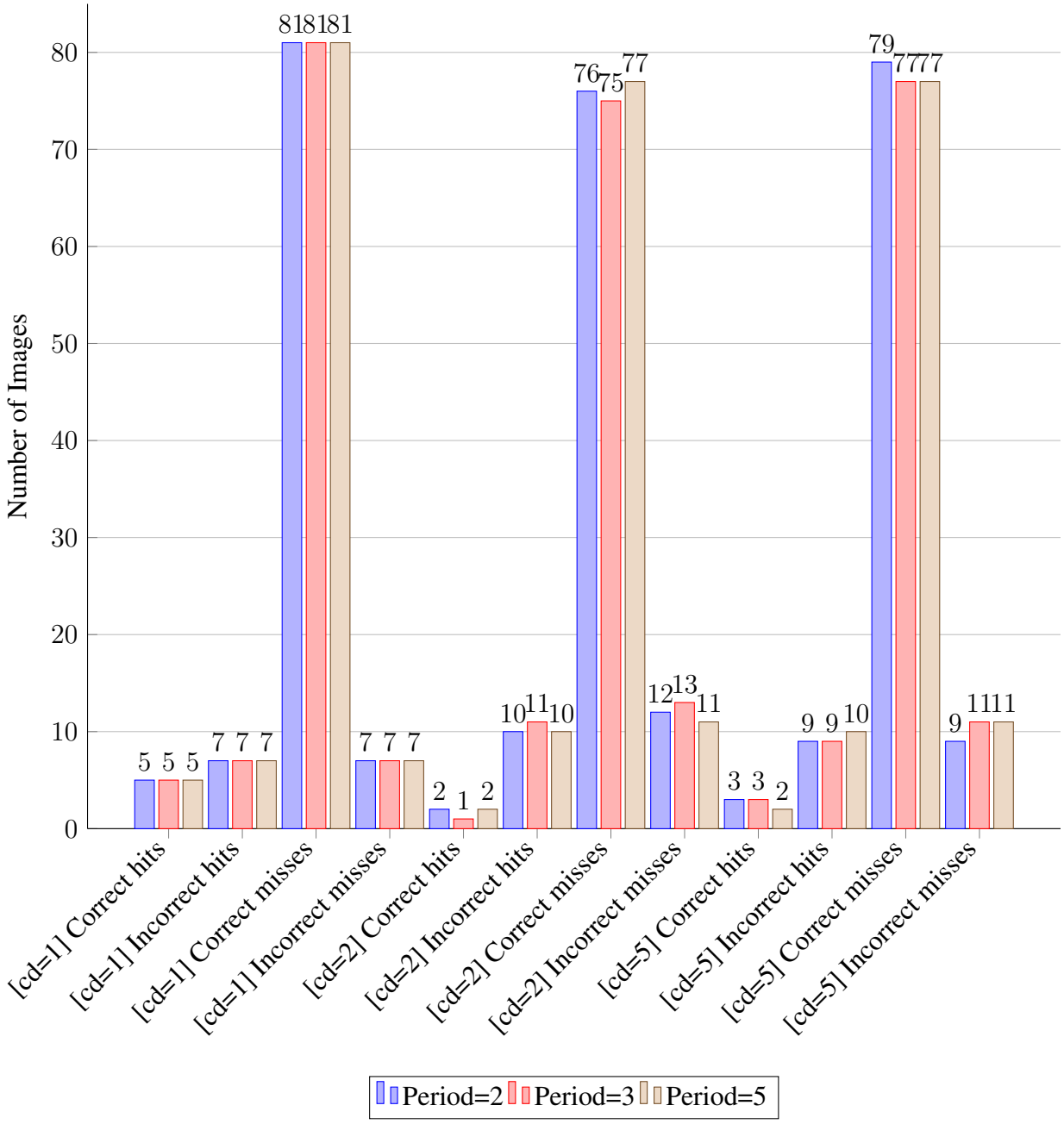

Figure 4.2: Prediction results of images in trace2 with different cloud delay

In Figure 4.2, we show the results including all three periods on trace2 with cloud delay from 1 to 2 to 5. As the cloud delay increases, we observe that the counting methods make more incorrect predictions, even though the counting methods maintain about 80% of the total correctness, regardless of various periods. However, the poor prediction on hits can be a threat that degrades the performance. The hits are predicted correctly only 17-25% of the time. The result shows that

the optimal value of rate allocation in equation 2.6 of Chapter 2 will be significantly lower than if the hits were correctly predicted.

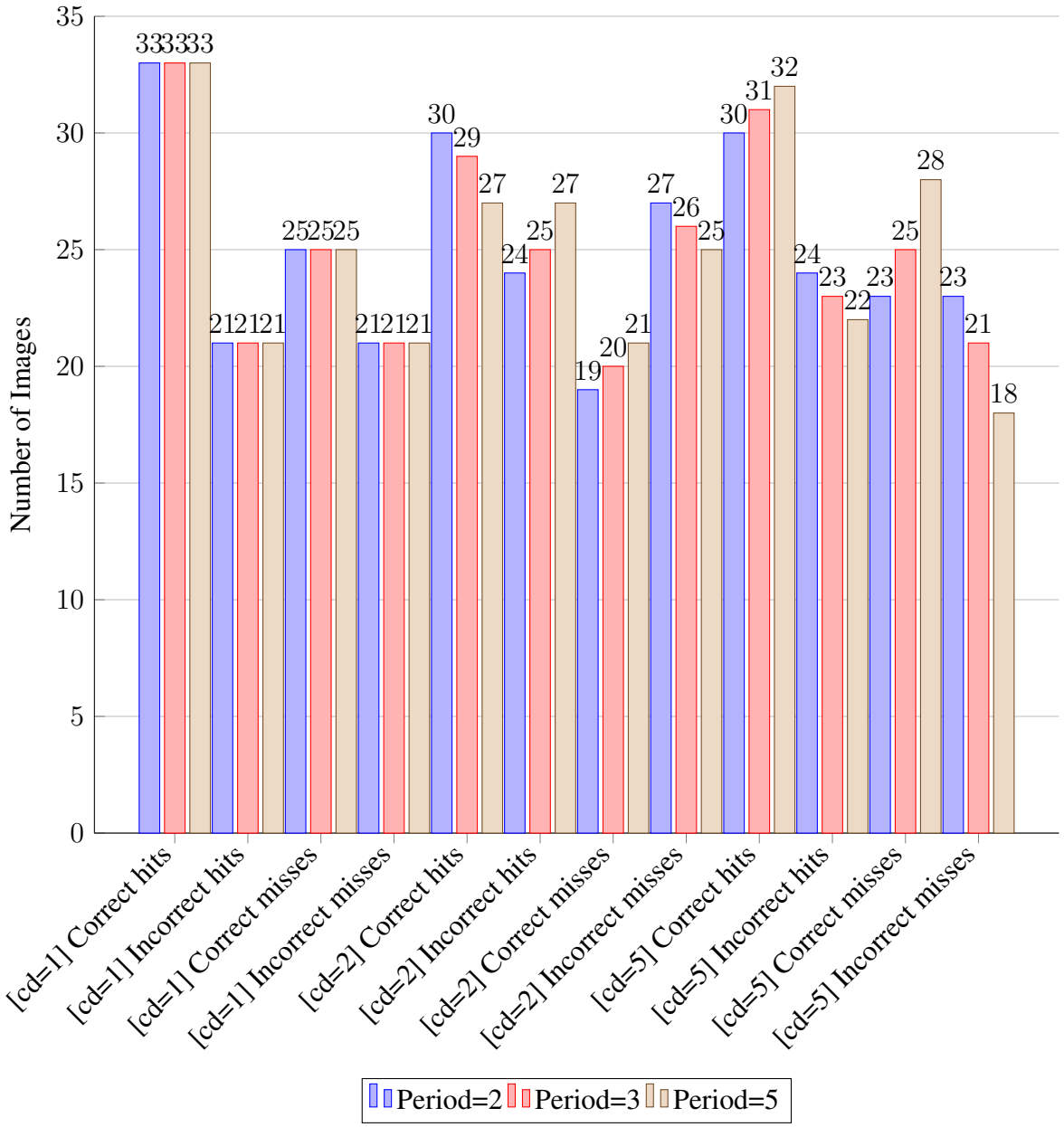

Figure 4.3: Prediction results of images in trace3 with different cloud delay

In Figure 4.3, we show the results including all three periods on trace3 with cloud delay from 1 to 2 to 5. The total correctness reduces to about 55%. As cloud delay increases, predictions are affected negatively regardless of different levels of period. Considering three different delays together, we observe that Period2 has 57.4% correctness on “hit”, and 48.6% correctness on “miss”;

Period3 has 57.4% correctness on “hit”, and 50.7% correctness on “miss”; Period5 has 56.8% correctness on “hit”, and 53.6% correctness on “miss”.

Three types of period have similar performance. Yet, the performance for trace3 has room to improve. For example, we can utilize the weighted average method, which might be more stable in the situation of flipping hits and misses. While these results are only accurate with 50%, the algorithm will cause performance degradation to the resource allocation algorithm.

In conclusion, we observe that the counting methods perform well with the traces that have long runs of hits or misses with relatively short delays. However, for the traces that only have short runs or runs shorter than delay, the counting methods perform poorly. For the second case, we need a better solution other than the counting methods. To predict better with short runs, one possible approach is to put more weights on the most recent feedback (e.g. weighted moving average). The more recent the feedback is, the more it is relative to the next image. We may also consider tracking all the statistics of images as well as network delay. By learning the characteristics of the runs and delay, we utilize some adaptive methods to switch through different prediction rules in order to fit the learned characteristics of the image set and to achieve better performance.

Chapter 5

Conclusion

In this thesis, we consider the impact of delay on the performance of feedback required for network algorithms.

In Chapter 3, we perform the analysis of real-world image/video data to show characteristics of hit and miss distribution. Then we propose counting methods and test their performance on the representatives of traces with and without delay. The Counter3 with threshold [0, 1] provides the most robust results during the tests.

In Chapter 4, we isolate the hit rate predictor from the Dual-path NUM. Then we perform several tests to show how our counting method works in a dual-path system. We show the counting method inherits its good performance from single-path to dual-path, depending on the distribution of hits and misses.

For future work, we can do experiments on the weighted moving average method with real-world traces and figure out in which scenario the weighted moving average method performs well. We may figure out some adaptive predicting methods that switch between counting methods and weighted moving average. Furthermore, we may combine the dual-path NUM framework with the hit rate predictor and perform tests on it, as to quantify the impact of incorrect predictions on the NUM system due to feedback delay.

Bibliography

[1] Specification of Internet Transmission Control Program. RFC 675, December 1974.

[2] A. K. Maulloo F. P. Kelly and D. K. H. Tan. Rate control for communication networks: Shadow prices, proportional fairness and stability. *Journal of the Operational Research Society*, 49(3), 1998.

[3] F. Kelly. Charging and rate control for elastic traffic. *European Transactions on Telecommunications*, 8, 1997.

[4] D. Comer. *Internetworking with TCP/IP: Principles, protocols, and architecture*. Internetworking with TCP/IP. Pearson Prentice Hall, 2006.

[5] J. Chu V. Paxson, M. Allman and M. Sargent. Computing TCP's Retransmission Time, June 2011.

[6] V. Jacobson. Congestion avoidance and control. *SIGCOMM Comput. Commun. Rev.*, 18(4):314–329, August 1988.

[7] P. L. Dordal. An introduction to computer networks, 2019.

[8] D. P. Palomar and M. Chiang. A tutorial on decomposition methods for network utility maximization. *IEEE Journal on Selected Areas in Communications*, 24(8), 2006.

[9] M. Mahon H. Qiu K. Chan K. S. Wheatman, F. Mehmeti and T. La Porta. Optimal resource allocation for crowdsourced image processing. In *Proc. of IEEE INFOCOM 2020*. IEEE, 2020 in review.

[10] A. Le P. Vo and C. Hong. The successive approximation approach for multi-path utility maximization problem. In *Proc. of IEEE ICC*, 2012.

[11] X. Lin and N. Shroff. Utility maximization for communication networks with multipath routing. *IEEE Transactions on Automatic Control*, 51(5), 2006.

[12] S. Rallapalli A. J. Bency K. Chan R. Urgaonkar B. S. Manjunath H. Qiu, X. Liu and R. Govindan. Kestrel: Video analytics for augmented multi-camera vehicle tracking. In *Proc. of IEEE/ACM Third IoTDI*, pages 48–59. IEEE, 2018.

Academic Vita

Zhaohong Lyu

State College, PA | lvzh.louis2013@gmail.com

https://www.linkedin.com/in/v1siuol/ | https://www.v1siuol.com | https://github.com/v1siuol

PROFILE

- Developed a mobile app that provides campus events ticketing services and secondhand market platform online, attracting over 4000 users up-to-date
- Had plenty of research experiences and active on GitHub, being able to learn trending technologies quickly via official documents and paper online
- Familiar with Python and React, proficient in written and verbal English & Mandarin

EDUCATION

The Pennsylvania State University **June 2016-Present**

Schreyer Honors College

Bachelor of Science in Computer Science & Bachelor of Science in Mathematics (Systems Analysis Option)
Awards: Dean's List (2016 & 2017 & 2018 & 2019)

EXPERIENCE

Vice President

Chinese Undergraduate Student Association, Penn State, PA **March 2017-Present**

- Led our team to plan lots of campus events, attracting over thousands of students in total
- Maintained our webpage (https://www.psucusa.com/), which has 400PV and 80UV daily

Research Assistant

D.A.T.A. Lab, Penn State, PA **February 2018-August 2019**

- Evaluated and validated the results of aircraft models generated from our self-designed GAN model
- Scraped Tweets posts, filtered and pre-processed the dataset, then built the model to train the dataset, open source code: https://github.com/DataLabPSU/Tweets-Database-Manager
- Tested the data for accuracy of authorship attribution for social media forensics

Teaching Assistant

College of Engineering, Penn State, PA **January 2017-Present**

- Helped professor solve students' problems on Artificial Intelligence course in Fall 2019
- Reviewed students' codes and resolved their concerns on Object Oriented Programming with Web-Based Applications course in Spring 2018
- Assisted recitation sessions and answering students' questions on Intro to Python Programming course in Fall 2018

PROJECTS

Personal Website

https://www.v1siuol.com/ **July 2018-Present**

- Created a personal blog via React, Flask, nginx, uWSGI, MySQL and deployed on Ubuntu

Honors Thesis **March 2019-Present**

- Researched on the impact of feedback delay on predicting hit rate and on the performance of dual-path Network Utility Maximization (NUM) framework

Group Research Project

https://github.com/jed326/EasyCCGTrees **August 2018-November 2018**

- Categorized automatically wh-questions by combinatory categorial grammar (CCG), increasing the response rates and precisions of searching queries

https://github.com/Oponn-1/3D-Point-Cloud-Denoising **August 2018-December 2018**

- Implemented the tests and evaluations for 3D point clouds de-noising project