Optimal Resource Allocation for Crowdsourced Image Processing

Kristina Sorensen Wheatman
The Pennsylvania State University
kms674@psu.edu

Fidan Mehmeti
The Pennsylvania State University
fzm82@psu.edu

Mark Mahon
The Pennsylvania State University
mpm114@psu.edu

Hang Qiu
University of Southern California
hangqiu@usc.edu

Kevin Chan
US CCDC Army Research Laboratory
kevin.s.chan.civ@mail.mil

Thomas La Porta
The Pennsylvania State University
tfl12@psu.edu

***Abstract*—Crowdsourced image processing has the potential to vastly impact response timeliness in various emergency situations. Because images can provide extremely important information regarding an event of interest, sending the right images to an analyzer as soon as possible is of crucial importance. In this paper, we consider the problem of optimally assigning resources, both local (CPUs in phones) and remote (network-based GPUs) to mobile devices for processing images, ultimately sending those of interest to a centralized entity while also accounting for the energy consumption. To that end, we use the Network Utility Maximization (NUM) framework, coupled with a hit-ratio estimator and energy costs, to enable a distributed implementation of the system. Our results are validated using both synthetic simulations and real-life traces.**

***Index Terms*—Crowdsourcing, Optimization, Resource allocation.**

I. Introduction

Crowdsourced image processing [1], [2] is an important tool in gathering information rapidly for emergency response, law enforcement and investigative applications. Images often contain a rich set of information concerning an event. Extracting this information is frequently time-critical. In this paper, we propose a crowdsourced image processing system that collects images containing an object of interest from mobile devices distributed across a network. The system leverages the processors in mobile devices that are designed to capture images and videos, as well as relatively powerful processors located at the edge of the wireless network. These network-based processors form an *edge cloud*, a notion that is becoming popular to support distributed analytics services [3]. A challenge with such a system is determining how to optimally allocate resources among the devices containing images. There is contention between the mobile devices for the wireless link to the edge cloud and for the processing capabilities within the edge cloud. In an attempt to avoid unnecessary wireless congestion, mobile devices may choose to perform processing locally to extract image information. However, local processing consumes significant energy and can be orders of magnitude slower than processing on a powerful GPU. Furthermore, images found to contain the object of interest must be uploaded to the cloud for collection anyway.

Our objective is to design a system and set of algorithms to optimally assign resources to mobile devices for processing and collecting images so that images of interest are prioritized. Whereas prior work [1], [4] has focused on minimizing the processing time for *all images*, we are instead interested in maximizing the rate at which we gather *important images* to obtain useful information as quickly as possible. This implies that mobile devices with images unlikely to contain information of interest receive lower priority when allocating resources.

To solve this problem we resort to Network Utility Maximization (NUM). The NUM framework lends itself to a distributed implementation that can be run on mobile devices and edge cloud nodes, allowing for proportional fairness that can be used to give higher priority to devices with more useful images. Specifically, our contributions are:

- We formulate an optimization problem where the objective is to maximize the total utility, while capturing weighted proportional fairness, given the shared wireless

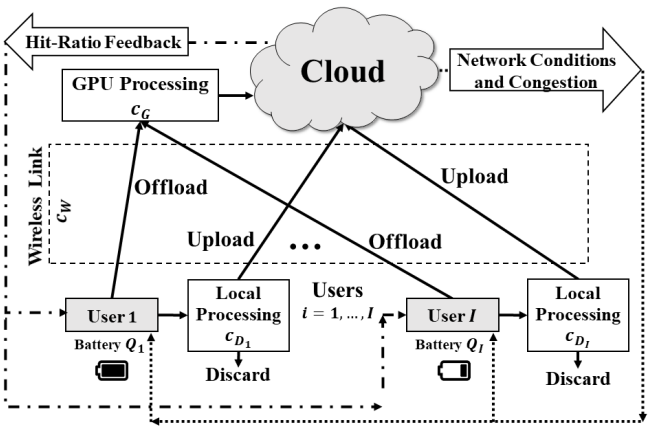

Fig. 1. The system model for mobile crowdsourcing using a dual-path approach incorporating proposed *hit-ratios* and *energy conservation* measures.

Research was sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-09-2-0053 (the ARL Network Science CTA) and NSF Award CNS 1815465. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

978-1-7281-6630-8/20/$31.00 2020 © IEEE

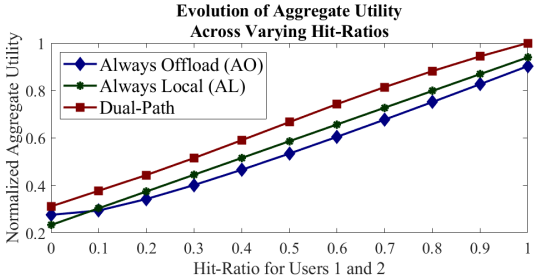

Fig. 2. Utility aggregated across four users for Always Offload (AO), Always Local (AL), and Dual-Path varying hit-ratio of Users 1 and 2.

link and GPU capacities, by using expected *hit-ratios* as a component in the utility. We define *hit-ratio* as the estimated likelihood of images containing the object of interest.

- We propose a dual-path approach where a user can either immediately *offload* images for processing, or first process images locally and then *upload*. We show that this approach can significantly improve performance.
- We incorporate an energy cost by allowing a user to indicate a usable energy budget. We introduce a tunable parameter that lets the system trade-off the importance of instantaneous utility against system lifetime according to the energy budget of each user.
- We present a hit-ratio estimator and provide extensive validations of our results using both synthetic data and data from real image traces.

Our results show that our dual-path approach of using either the local CPU or offloading to the network based GPU provides better performance than relying solely on local processing or always offloading. We also obtain images with the desired object of interest faster than algorithms focused on minimizing completion time [4]. Further, we show that our energy parameter allows the system operator to tune the system to achieve higher instantaneous utility in the short term, or to use energy more judiciously to allow more images of interest to be found over a longer time period. Tuning this parameter can increase the number of images of interest gathered by 75% while increasing the number of images processed by only 6%, thus showing that the algorithm is intelligent in allocating resources to the most productive users.

This paper is organized as follows. In Section II we present the motivation for this work, followed by the problem formulation and the corresponding algorithms in Section III. We introduce a *hit-ratio* estimator in Section IV, providing numerical results and further insights in Sections V and VI, including our analysis of *hit-ratio* estimators. In Section VII we discuss some related work. Finally, we conclude our work in Section VIII.

II. Motivation

We consider a crowdsourcing system (Fig. 1) in which mobile devices have stored images which may or may not include an object of interest. A query is issued to mobile devices wishing to assist in the search for the object. The objective is to collect images that possess the object of interest into the cloud as fast as possible.

The system includes mobile device CPUs with a processing capacity of C_D for performing object classification on images, and batteries of capacity Q. The mobile devices communicate with the network over a shared wireless link of capacity C_W. The edge network contains a GPU shared by the mobile devices and has an available capacity of C_G to process images that are offloaded to it. The mobile devices have two choices when processing images: they can (i) *offload* them to the shared GPU for processing, or (ii) process them locally on their own CPU. In the latter case, images containing the object of interest are *uploaded* over the shared wireless link.

If the mobile device decides to process an image locally, the load on the shared GPU is automatically reduced, and the load on the shared wireless link is reduced if the image does not contain the object of interest. However, processing on the local CPU is slower than on the network GPU (assuming the network GPU is not congested), and requires precious energy from the battery of the local device. Intuitively, local processing is desirable for devices with low hit-ratios.

If the mobile device decides to offload images to the shared GPU for processing, experimental data confirms that the mobile device will expend less energy than with local processing. Nevertheless, this may cause more congestion at the shared GPU and on the wireless link. Intuitively, offloading images to the GPU is desirable for devices with high hit-ratios.

If there are no energy costs in the system, i.e., all devices have persistent power sources, our solution provides optimal resource allocation to maximize the aggregate utility of the system where utility is a weighted function of the rate of gathered images with the object of interest. If energy sources are not persistent, there is a trade-off between instantaneous utility and lifetime of the system. We provide a tunable parameter that allows for the system operator to tune this trade-off. In this case our solution provides optimal resource allocation to maximize the utility of the system, given the chosen energy-sensitive parameter setting.

Fig. 2 shows the aggregated utility of four devices as the hit-ratio increases for three algorithms. We compare the utility for the optimal allocation of resources (as derived in this paper, called *dual-path* in Fig. 2), a system in which all images are processed locally on the mobile devices (*always local (AL)*), and a system in which all images are offloaded to the GPU for processing (*always offload (AO)*). As can be seen, the dual-path approach always provides the highest utility. In this example, the GPU capacity is $C_G = 1.6$ Gbps, the wireless link capacity is $C_W = 25$ Mbps, and the local CPU of the mobile devices has a capacity of $C_D = 8$ Mbps. Users 1 and 2 have equivalent variable hit-ratios, while users 3 and 4 have a stable hit-ratio of $h_i = 0.33$. We do not consider energy costs in this motivating example.

III. Problem Formulation

In this section, we present the optimization formulation for our setting, where the hit-ratios and the energy consumption are considered.

A. NUM Background

Our work builds upon the basic principles of Network Utility Maximization (NUM) [5], [6], [7], as well as multi-path approaches [8], [9]. The NUM framework is a well known method for solving convex optimizations and lends itself to a distributed implementation that can be proven as optimal. NUM consists of maximizing the aggregate utility over all

users to obtain optimal resource allocation. To do this in a decentralized setting, network resources set prices for their usage, where the prices are a function of congestion. These prices are sent to the users. The users set their optimal rates using the received prices and their willingness to pay given their individual utility functions, where utility is a function of their rate. In our solution we adapt a dual-path version of NUM where the first path is offloading images to the network GPU for processing, and the second path is local processing on the mobile device followed by uploading images that contain the object of interest.

B. Dual-Path NUM with Hit-Ratio Considering Energy Constraints

The NUM formulation for the dual-path case, with users $i = 1, \ldots, I$, paths $j = 1, 2$ per each user i, and resources $\ell = 1, \ldots, L$ consists in solving the following optimization [8], [9]:

$$\textbf{SYSTEM } (\mathbf{U}, \mathbf{X}; \mathbf{R}, \vec{\mathbf{c}}):$$
$$\underset{\vec{\mathbf{x}} \geq 0}{\text{maximize}} \ \mathbf{U}(\mathbf{X}) = \sum_{i=1}^{I} U_i \left(x_{i1} + x_{i2} \right)$$
$$\text{subject to } \sum_{i=1}^{I} (R_{i1}^{\ell} x_{i1} + R_{i2}^{\ell} x_{i2}) \leq c_{\ell}, \quad \forall \ \ell. \tag{1}$$

The utility U_i is a function of the sum of matrix values $(\mathbf{X})_{ij} = x_{ij}$, denoting rates given to user i over path j. We assume that all the rates belong to the convex set $\mathcal{S}' = \{x_{ij} \geq 0 \ \forall \ i, j, \ (x_{i1} + x_{i2}) \subset \mathbb{R}\}$. The capacity of resource ℓ is denoted by c_{ℓ}. Indicator variable R_{ij}^{ℓ} is 1 if the route of path j for user i uses resource ℓ, 0 otherwise. For path 1, the resources are the shared wireless link and the network-based GPU which are used for all images. For path 2, the resources are the CPU on the mobile device, and the shared wireless link for images that are uploaded. Since the objective function in Eq.(1) is non-strictly concave, we approximate the optimization function as in [9], defining a new parameter $\theta_{ij} = \frac{x_{ij}}{x_{i1} + x_{i2}}$, $\forall \ i, j$. The new utility function becomes $U_i \left(x_{i1} + x_{i2} \right) = \theta_{i1} \tilde{U}_i \left(\frac{x_{i1}}{\theta_{i1}} \right) + \theta_{i2} \tilde{U}_i \left(\frac{x_{i2}}{\theta_{i2}} \right)$.

1) Incorporating Hit-Ratio: The hit-ratio of a specific user i is the percentage of images possessing an object of interest. Each user i has a unique hit-ratio $h_i \in [0, 1]$. For proportionally fair resource allocation with hit-ratio dual-path NUM, the utility function for each path of user i is

$$\tilde{U}_i \left(\frac{x_{ij}}{\theta_{ij}}, h_i \right) = h_i \log \left(\frac{x_{ij}}{\theta_{ij}} \right), \quad \forall \ i, j. \tag{2}$$

Because utility is only earned for images that have a hit, the utility function is multiplied by h_i.

2) Energy Considerations: Each mobile device has finite battery life, represented by the percentage of energy remaining in the system, Q_i. We limit the energy devoted for crowdsourcing by the mobile device by setting a threshold, below which the participation of the mobile user in the process will stop. Denote this threshold as η_i^*, for which it holds $0 \leq \eta_i^* \leq Q_i(t) \leq 1$ for all t. Define $E_{ij}^{\ell} = E_{ij}^{\ell}(Q_i(t))$ to be the energy cost scaling of path j for user i when using resource ℓ at time instant t. This scaling factor allows us to incorporate energy as well as congestion into the prices of the resources as more fully described below.

C. HED-NUM Optimization Formulation

We propose the optimization for hit-ratio, energy-conscious, dual-path NUM (HED-NUM) as

$$\tilde{\textbf{SYSTEM}}^{HED} \ (\tilde{\mathbf{U}}, \mathbf{X}, \boldsymbol{\Theta}, \vec{\mathbf{h}}; \mathbf{R}, \mathbf{E}, \vec{\mathbf{c}}):$$
$$\underset{\vec{\mathbf{x}} \geq 0}{\text{maximize}} \ \tilde{\mathbf{U}}(\mathbf{X}, \boldsymbol{\Theta}, \vec{\mathbf{h}}) = \sum_{i=1}^{I} \sum_{j=1}^{2} h_i \theta_{ij} \log \left(\frac{x_{ij}}{\theta_{ij}} \right)$$
$$\text{subject to } \sum_{i=1}^{I} \sum_{j=1}^{2} E_{ij}^{\ell} R_{ij}^{\ell} x_{ij} \leq c_{\ell}, \quad \forall \ \ell. \tag{3}$$

Let C_{G_z} represent the resource constraint on the z^{th} GPU, C_{W_m} denote the bandwidth restrictions for m^{th} wireless link, and C_{D_i} designate the maximum capacity of local processing device for user i. We consider our scenario where every user uses two possible paths to send the images; one path forwards the images directly through the wireless link to a GPU (*offloading*) with rate x_{i1}, and the other option is through the local processor first and then to be uploaded directly via the wireless channel (*uploading*) with rate x_{i2}. The constraints for Eq.(3) are now

$$\begin{aligned} &\textstyle\sum_{i=1}^{I} E_{i1}^{G_z} R_{i1}^{G_z} x_{i1} \leq C_{G_z}, && z = 1, \ldots, Z, \\ &\textstyle\sum_{i=1}^{I} E_{i1}^{W_m} R_{i}^{W_m} \left(x_{i1} + h_i x_{i2} \right) \leq C_{W_m}, && m = 1, \ldots, M, \\ &E_{i}^{D_i} x_{i2} \leq C_{D_i}, && i = 1, \ldots, I. \end{aligned} \tag{4}$$

The energy cost scaling for this specific dual-path case, given a chosen *energy exponent* $b > 0$, is[1]

$$\begin{aligned} E_{ij}^{G_z} &= 1, & E_{i2}^{W_m} &= h_i E_{i1}^{W_m}, \\ E_{i1}^{W_m} &= \tfrac{1}{(Q_i(t) - \eta_i^*)^b}, & E_{ij}^{D_i} &= \left(\gamma_i^{(2)} / \gamma_i^{(1)} \right) E_{i1}^{W_m}, \end{aligned} \tag{5}$$

across all variables i, j, m, z. Setting b allows us to tune the impact of energy on the resource allocations. The energy cost scaling for the GPUs is unity because they have a persistent power supply. The energy scaling factors $E_{ij}^{\ell} = E_{ij}^{\ell}(Q_i(t), b)$ for wireless link W_m and local processing D_i resources are functions of the remaining battery $Q_i(t)$ per user i at time instant t, as well as the energy consumption rate of the resource (wireless transmission or CPU) and the energy exponent b included in $(Q_i(t) - \eta_i^*)^b$. Let the wireless channel energy usage per second be denoted by $\gamma_i^{(1)}$ and the local processing energy usage per second be denoted by $\gamma_i^{(2)}$ for each user i. The ratio $\gamma_i^{(2)} / \gamma_i^{(1)}$ determines the relative energy cost of local processing compared to wireless transmission.[2]

D. Dynamic Operation

We decompose this optimization into two problems, each solved iteratively. The first is solved by the network resources, namely the GPU and base stations controlling the wireless links, which derive prices that are sent to the mobile devices. The second is solved by the mobile devices that set their optimal rates for the two available paths based on the sum of the costs for each of those paths and their utility functions.

[1] In the case of $b = 0$, set $E_{i2}^{W_m} = h_i$ and $E_{ij}^{\ell} = 1$ for all other i, j, ℓ.

[2] Experimental results show that $\gamma_i^{(2)} / \gamma_i^{(1)} \geq 1$ [4]. A slightly modified formulation of E_{ij}^{ℓ} emerges for the case when $\gamma_i^{(2)} / \gamma_i^{(1)} < 1$.

When in operation, the values of θ_{ij}, resource prices λ_ℓ, and rates x_{ij} are updated iteratively.

To update θ_{ij}, the mobile devices use their previous optimal rates, calculated in *Result 1* below, according to:[3]

$$\theta_{ij}(t+1) = \frac{x^*_{ij}(t)}{x^*_{i1}(t) + x^*_{i2}(t)}, \quad \forall\, i, j. \tag{6}$$

We introduce *shadow prices* $\vec{\lambda} = \{\lambda_1, \ldots, \lambda_L\} \geq 0$ corresponding to each resource ℓ. These *shadow prices* emerge mathematically as the Lagrange multipliers. Using the gradient descent method, shadow pricing updates for each resource ℓ evolve as

$$\lambda_\ell(t+1) = \left[\lambda_\ell(t) - \alpha\left(c_\ell - \textstyle\sum_{i=1}^{I}\sum_{j=1}^{2} E^\ell_{ij} R^\ell_{ij} x_{ij}(t)\right)\right]^+, \tag{7}$$

where $[\cdot]^+$ denotes the projection onto the non-negative orthant, and $\alpha > 0$ is a sufficiently small positive step-size value [10]. Eq. (7) is solved by the network GPU and base station of the wireless link to set their prices in each iteration.

To account for energy usage over time, the values of the energy scaling factors must evolve by updating $Q_i(t)$. Define Δt as the time duration throughout time step $(t-1) \to t$ and the energy capacity of each mobile device as Ω_i. Then $Q_i(t)$ evolves as

$$Q_i(t+1) = Q_i(t) - \frac{\Delta t}{\Omega_i}\left[\gamma_i^{(1)}\big(x_{i1}(t) + h_i x_{i2}(t)\big) + \gamma_i^{(2)} x_{i2}(t)\right], \tag{8}$$

where $Q_i(0) = 100\%$ for each user i with full initial battery. Given the natural battery constraint $0 \leq \eta^*_i \leq Q_i \leq 1$, the thresh-holding constraint of η^*_i requires user i to conserve battery by terminating processing once $\eta^*_i \cdot 100\%$ remaining battery has been reached.[4]

Given the prices from the network resources, the users then set their optimal rates.

Result 1. *The optimal solution at each iteration is [9]*

$$x^*_{ij}(t) = \frac{h_i \theta^*_{ij}}{\sum_{\ell=1}^{L} E^\ell_{ij} R^\ell_{ij} \lambda^*_\ell} \quad \forall\, i, j, \tag{9}$$

where shadow prices λ^*_ℓ are found from slackness conditions.

The proof can be found in Appendix. Given a selected b, the optimal rate values at each time instant t for user i for each path can be expanded to:

$$x^*_{i1}(t) = \frac{h_i \theta^*_{i1}(t)}{\lambda_G + E^{W_m}_{i1}\lambda_{W_m}},\ x^*_{i2}(t) = \frac{h_i \theta^*_{i2}(t)}{E^{D_i}_i \lambda_{D_i} + h_i E^{W_m}_{i1}\lambda_{W_m}}. \tag{10}$$

The mobile devices calculate these rates based on the prices received from the network and their own calculation of θ_{ij}.

IV. Hit-Ratio Estimator

A key part of our system is the use of expected hit-ratio to help determine resource allocation, thus the proper estimation of hit-ratio is critical for the operation of the algorithms.

We postulate that when using real-world image data sources when looking for specific objects, there will be several images in a cluster, which we call *runs*, that will have the object with a high likelihood (*hits*), followed by a cluster of images that do not (*misses*). To test this conjecture we analyze image data from several deployed cameras to determine realistic distributions of object hits and misses. We then propose a hit-ratio estimator.

A. Data Analysis

We used image data from a camera network deployed at a major research university [11]. These stationary cameras record video of sidewalks and roads. Images were simultaneously collected at ten second intervals from 6 camera positions for approximately 45 minutes, resulting in six sets of 273 images which we call *traces*. We analyzed, by hand, the images gathered for several objects of interest. The data confirm our conjecture. There are runs during which the large majority of images have the object of interest, followed by runs where very few, if any, images have the object of interest.

We found that the length of the runs for which there was a high hit-ratio ranged from 1 image to all 273 images in a trace, with the average hit-ratio during these runs being 81.7%. The length of the runs for the low hit-ratio periods ranged from 1 image to 231 images with an average hit-ratio of 4.8%.

We expect the results taken from cameras that move to have similar characteristics in that there will be states during which the hit-ratios of objects of interest are high, and then states during which the hit-ratio is low.

B. Estimator

We developed two classes of hit-ratio estimators. In the first, we use the moving average of hit-ratios. We considered and evaluated a straight average, and two-point and ten-point weighted moving averages with both linear and quadratic weighting. In the second, we use counting methods in which we incremented a counter for a hit and decremented a counter for a miss, coupled with a threshold. We considered several thresholds but found the best performing to be when the counter is initialized at 0 and incremented or decremented for a hit or miss, respectively, within a range of -1 to +2. If the counter value is +1 or +2, the estimator predicts the next image to be a hit, otherwise it predicts a miss. We evaluate these approaches in the next section.

V. Results: Utility Comparison and Energy Parameters across Homogeneous Cells

In this section we highlight the effects of our algorithm on resource allocation and the sensitivity of the performance to different energy settings. To do this, we consider a homogeneous setting. We first discuss the simulation setting, including the system configuration, the image processing used and our assumptions regarding energy usage. We present an evaluation of the hit-ratio estimator. We then present results for variable settings of the energy usage parameter, b. Individual user results are presented with the average values of allocated resources to provide clarity behind aggregate behaviors.

A. Simulation Setup

We consider a system with 80 equivalent cells with independent wireless channels. For convenience, all of the capacities and resource usages in this paper are presented in terms of bits per second (bps), a conversion from processing rates using image sizes. All of the cells share a network-based GPU with a capacity of $C_G = 1.6$ Gbps (about 4K images/sec). The wireless link in each cell has a capacity of $C_W = 25$ Mbps

[3] So long as $x_{i1} + x_{i2} \neq 0$ such that $\frac{x_{ij}}{\theta_{ij}}$ is defined, else $\theta_{ij} = 1$.

[4] In Section V we assume that the battery threshold is 20%.

(about 6 images/sec) shared among all users in that cell. Each device has a CPU with a maximum local processing rate of $C_D = 16$ Mbps (about 4 images/sec). Observe that the maximum rate or speed through the local CPU is less than the throughput of the wireless link. We set the energy threshold of each user to 20% of its total battery capacity.

Each user i has the same log-based utility function, $U_i = h_i \cdot log(1 + x_{i1} + x_{i2})$, where h_i is the estimated hit-ratio for the user, x_{i1} is the rate in bps for which images are offloaded to the GPU, and x_{i2} is the rate in bps at which images are processing locally[5]. In each cell two users (users 1 and 2) have an average hit-ratio of $h_i = 0.5$, and eight users (users 3-10) have an average hit-ratio of $h_i = 0.05$. We refer to these as high hit-ratio and low hit-ratio users, respectively. We describe how the hits and misses are generated for each result obtained in Sections V-VI. We pick processing values assuming that an existing complex, pre-trained CNN (i.e., GoogLeNet [12]) is used for image processing on both the local CPU and GPU. In the hit-ratio dual-path NUM model outlined in Section III, the image analysis through the wireless-to-GPU path of user i occurs once the image reaches the GPU. In the second local-to-wireless path of user i, the image processing is completed locally and only images defined as *hits* are sent through the wireless link to decrease bandwidth congestion. GoogLeNet requires image data inputs of size 224x224x3, yielding $401,408$ bits per image at 8-bit color [12].

We assume an average user battery capacity of $\Omega_i = 9.25$ Wh for each user i. The energy consumption to send for the wireless channels is set to $\gamma_i^{(1)} = 0.125$ mJ/sec [13]. Processing locally requires an estimated average consumption rate of $\gamma_i^{(2)} = 0.375$ mJ/sec, as prior studies suggest that local processing is thrice more energy expensive than offloading [4]. GPU processing is considered "free" in terms of energy costs.

Specific NUM parameter values for these simulations include a gradient descent updating parameter value of $\alpha = 5 \cdot 10^{-5}$ ($\alpha = 3 \cdot 10^{-6}$ for $b = 2$ to ensure convergence), and the total number of iterations is $T = 240k$. Each iteration proceeds after an interval of approximately 16 ms.

B. Hit-Ratio Estimator Evaluation

To evaluate the hit-ratio estimator, we used the simulation setup described above. We generated synthetic sets of images with *hit* and *miss* distributions that mimic the trace data we analyzed. We present the hit and miss distribution generator used for testing and the hit-ratio estimator evaluation below.

1) Distribution Generator: To approximate the hit and miss distribution of the trace data we developed synthetic two-state generator. The system transitions between a high state and a low state, as shown in Fig. 3(c). While in the high state, the hit-ratio is p_h, and while in the low state the hit-ratio is p_l. The residence time in the two states depends on the transition probabilities p and q. Using this model we can closely approximate the trace data from the camera network. We use this model in our simulations.

Specifically, high hit-ratio users with an average hit-ratio of $h_i = 0.5$ have parameters $p = 0.10$, $q = 0.10$, $p_h = 0.95$ and $p_l = 0.05$. This results in an average residence time in the high hit-ratio state of 9 images and the low hit-ratio state of 9 images. Low hit-ratio users with an average hit-ratio of $h_i = 0.05$ include parameters $p = 1/3$, $q = 1/39$, $p_h = 0.50$ and $p_l = 0.026$, resulting in an average residence time in the high hit-ratio state of 2 images and the low hit-ratio state of 38 images. For simplicity of analysis, we present the results of a single cell.

[5]We are using the modified function $log(1 + x)$ in this section to avoid having negative values for the utility functions.

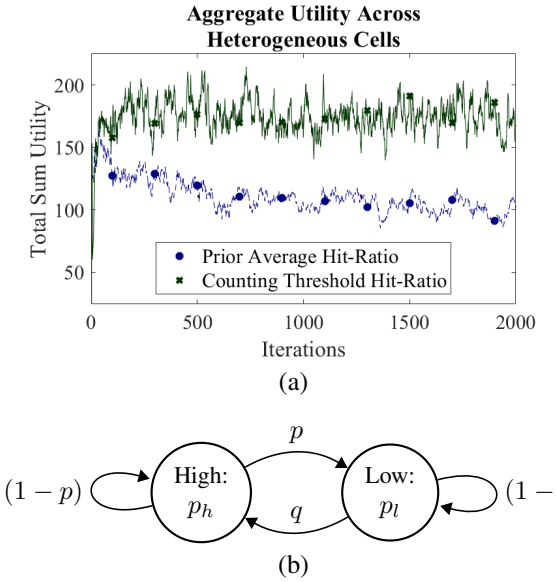

Fig. 3. (a) Comparative aggregate utility across ninety users in ten cells. (b) Generator Markov chain for data simulation of *hits* and *misses*.

2) Hit-Ratio Estimator Evaluation: In Fig. 3(b) we show the sum utility achieved by our system with a a counter-based and average-based hit-ratio estimator. As can be seen in Fig. 3, the counter-based hit-ratio estimator outperforms the average-based hit-ratio estimator. Therefore, for the rest of this paper we will use the counter-based hit-ratio estimator.

C. Comparison with PicSys

We compare HED-NUM to PicSys [4] which is a dual-path crowdsourced image processing algorithm that has the objective of minimizing the *completion time* required to process all images, unlike HED-NUM which has the objective of maximizing the rate of *gathering images* with the object of interest. Similar to HED-NUM, PicSys allows for images to be processed locally or offloaded to an external GPU for analysis, and must upload all images that possess the object of interest. PicSys does not perform wireless resource allocation. We ignore energy in this comparison.

All parameters are the same as described in Section V-A. We use the two-stage version of PicSys which includes local

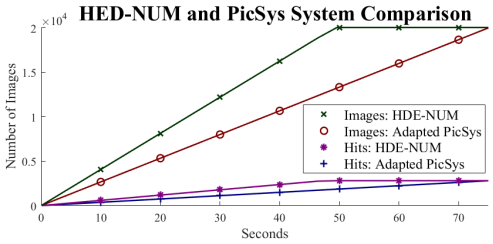

Fig. 4. Comparison of HED-NUM and PicSys crowdsourced image processing systems.

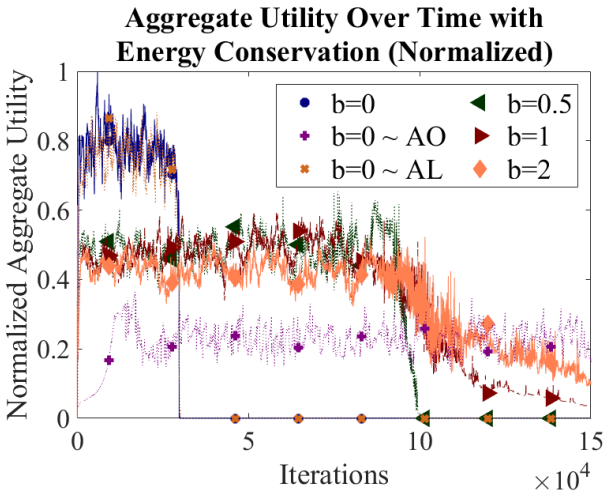

Fig. 5. Comparison of normalized aggregate utility across different energy parameter scenarios: $b = 0$, $b = 0 \sim$ Always Offload, $b = 0 \sim$ Always Local, $b = 0.5$, $b = 1$, and $b = 2$, for $T = 150k$ iterations (40 minutes).

processing and image offloading. Since PicSys does include dynamic resource allocation, we set their total wireless resource to $C_W = 100$ Mbps, where PicSys dedicates equivalent bandwidth to each user. Even with this strong advantage for PicSys, we show in Fig. 4 that HED-NUM performs almost 50% faster in both gathering images and, consequently, gathering images possessing the object of interest.

D. Overview of Results

Fig. 5 shows the main result of the simulations, comparing the achieved utility for the dual-path approach with different values of energy parameter b over the number of iterations. The utility values are normalized to that of the case $b = 0$, the energy protocol achieving the highest instantaneous utility. Also included on the graph are the results for the case in which images are always offloaded (always offload (AO)) to the GPU and the case in which all images are processed locally (always local (AL)) and only those with a hit are uploaded. For these results, we generate hits and misses for the images using the synthetic distribution generator described in Section IV with parameters set to achieve the desired average hit-ratio. The dual-path approach outperforms the AO and AL approaches by making optimal decisions on the allocation of the GPU and wireless links. In all cases the aggregate utility drops to zero when the remaining battery in the mobile devices reaches its threshold. The results also show that as energy usage is considered more important by virtue of increasing the value of b, the system achieves a lower instantaneous utility, but executes more iterations because of the more intelligent energy-aware resource allocation. This illustrates the usefulness of the parameter b.

The AO and AL results are not included moving forward. To more clearly isolate the effects of hit-ratio and energy constraints on resource allocation, we resort to using an average hit-ratio that is uniformly distributed for the results in Figs. 6 and 7.

E. No Energy Consideration, $b = 0$

The first case we examine is when energy is not considered when making resource allocation decisions, requiring $b = 0$. This setting achieves the highest instantaneous utility because resource allocation is optimized solely to maximize immediate utility. Thus, $b = 0$ is the typical setting when the system is not expected to operate for a sustained period.

In Fig. 5, we observe that in this configuration the system does achieve the highest aggregate utility for about 30k iterations before the battery threshold in the devices is reached and no more images are processed. Fig. 6 shows more details of this case for two representative users, user 2 which has a high hit-ratio ($h_2 = 0.5$) and user 10 which has a low hit-ratio ($h_{10} = 0.05$). The results for all other high and low hit-ratio users are the same.

Fig. 6(a) shows that high hit-ratio user *uploads* images that it has processed locally that have hits at more than four times the rate that it *offloads* to the GPU for processing. The low hit-ratio user performs all local processing, only using the wireless capacity to upload hits. The entire 25 Mbps wireless link is used within the cell, and both high and low hit-ratio users use all 16 Mbps of their local CPUs (Fig. 6(b)). Yet very little GPU capacity is used. As shown, the batteries of the high hit-ratio users are depleted shortly before those of the low hit-ratio users.

F. Energy Consideration: $b = 1$

In this case the resource allocation is made with consideration of the energy utilization. For these results we set $b = 1$, i.e., the "price" each user pays for a specific collection of resources is multiplied by a factor inversely proportional to the remaining battery percentage. As seen in Fig. 5, the instantaneous utility for $b = 1$ is lower than in the case of $b = 0$ initially, but the system processes images and produces utility for significantly more iterations. After about 100k iterations, the utility drops after the high hit-ratio users have depleted their allotted energy. The results are shown in more detail in Fig. 7.

Fig. 7(a) shows the residual energy in users 2 and 10 versus the number of iterations. The high hit-ratio user, user 2, is allocated more resources because the system is optimizing the rate at which images with hits are uploaded to the network. As a result, the energy in the high hit-ratio user is depleted more quickly than the low hit-ratio user, user 10. The energy depletion drives the cost of the CPU of user 2 to increase as shown in Fig. 7(b). User 2 has a high hit-ratio of 0.5, meaning that half of all images processed locally need to be uploaded directly to the cloud. As shown in Fig. 7(c), user 2 *offloads* images to be processed at the GPU at a rate 2-3 times higher than it *uploads* images with hits that it processes locally.

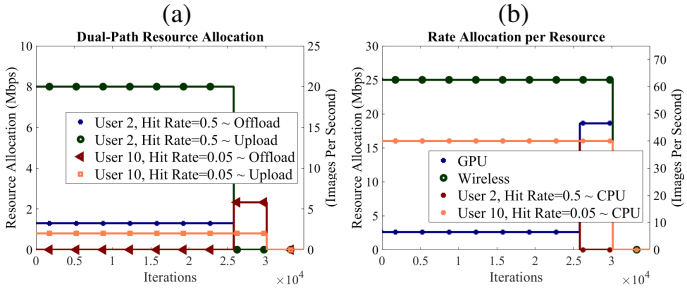

Fig. 6. Dual-path NUM analysis over $T = 35k$ iterations (about 10 minutes) for energy exponent parameter value $b = 0$. (a) Total summation of user rates. (b) Resource capacity utilization.

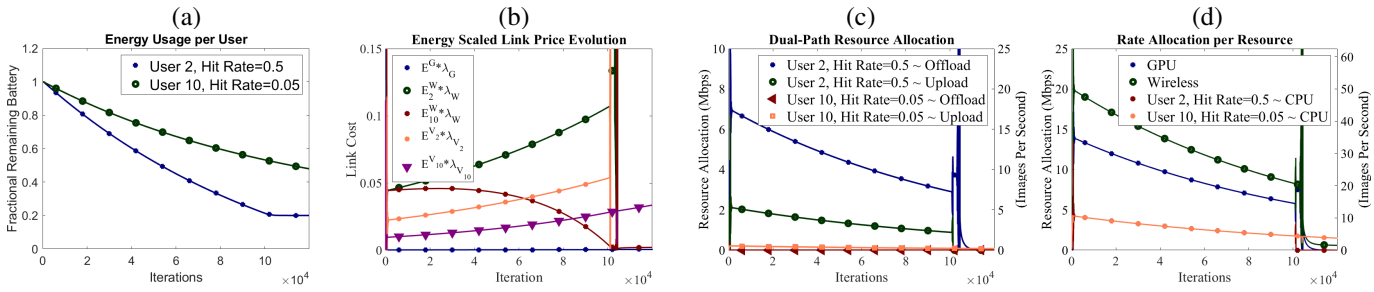

Fig. 7. Dual-path NUM analysis over $T = 120k$ iterations (32 minutes) for energy exponent parameter value $b = 1$. (a) Energy consumption over time. (b) Shadow price updates. (c) User rate allocation. (d) Resource capacity utilization.

This is the reverse of the case in which energy was not a factor, i.e., $b = 0$. This result emerges since the energy cost for local processing and then uploading half of the images is now considered higher than directly offloading and processing the images on the GPU. User 10 has a hit-ratio of 0.05, meaning that only a small percentage of locally processed images need to be uploaded to the cloud. To maximize the number of hits sent over the wireless resource, the low hit-ratio user performs all local processing. As battery level drops over time, user 10 decreases its rate of local image processing.[6]

Fig. 7(d) shows that the system throttles itself due to the sensitivity towards energy. The wireless link is never fully utilized. Likewise, neither the high or low hit-ratio user highly tax their local CPU, each using less than 5 Mbps of the 16 Mbps available. The GPU in this case is much more heavily used than in the case when $b = 0$ because of the increased offloading of the high hit-ratio users. In fact, when considering all 80 cells in the system, the GPU is used at 1.2 Gbps, which approaches its limit. The usage of all resources declines as the energy costs rise over time.

G. Comparison of Results

In this subsection we compare the performance of the system with different settings of b. Recall that energy costs are a function of remaining battery percentages Q_i for each user, and are used in the cost scaling factor $(Q_i - \eta_i^*)^b$ from the energy NUM formulation in Section III.

The best selection of b is heavily dependent upon the specific objectives of the user in terms of the trade-off between instantaneous utility and the time duration that the system has sufficient energy to continue processing images. While Fig. 5 shows the instantaneous utility achieved for different values of b versus the number of iterations, Fig. 8 illustrates the *accumulated* utility versus iterations for values of b of $0, 0.5, 1,$ and 2. This shows that b can be set to maximize the utility given a time window during which the query must complete.

Another way to set b is to use it to tune how many stored images may be processed. Because the system gives priority to devices with a higher hit-ratio, these devices tend to expend their battery faster and therefore process fewer images. By setting b to a larger value, we expect the high hit-ratio devices to have more of a chance to process images before exhausting their battery. Table I summarizes the number of images processed across the system for different values of b until the instantaneous utility is reduced to 1% of its peak value. As can be seen, perhaps surprisingly, as b increases, the growth in the number of images processed is relatively small, and the number of images processed actually decreases when b is increased from 1 to 2. This is because the system throttles itself more aggressively to save energy. However, the number of *hits* gathered by the system increases by more than 75% as b is increased from 0 to 2 despite the number of images processed increasing by only about 6%. There is a corresponding increase in effective hit-ratio, meaning that of the extra images processed, and images processed in general, more are from the high hit-ratio devices. This shows that the extra sensitivity to energy benefits high hit-ratio devices.

Based on these results, it is clear that b can be used to tune the system based on different objectives. It is for further study to develop a general framework for setting b based on specific numeric requirements, given a system configuration.

[6]Diminished wireless usage and minimal congestion account for the energy cost of the wireless resource for user 10 to drop as shown in Fig. 7(b).

TABLE I
COMPARISON OF UTILITY PERFORMANCE, GIVEN ENERGY EXPONENTS

ENERGY	*b=0*	*b=0.5*	*b=1*	*b=2*
No. Images	193,175	204,634	218,587	205,597
Image Hits	25,924	31,657	38,624	45,601
Est. Hit-Ratio	0.1342	0.1547	0.1767	0.2218
Battery (Min)	8.08	31.28	111.60	258.29

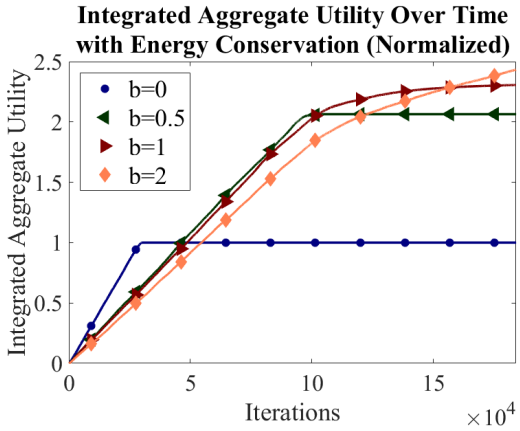

Fig. 8. Comparison of integrated aggregate utility for $T = 184k$ iterations (about 50 minutes) across different energy parameter scenarios.

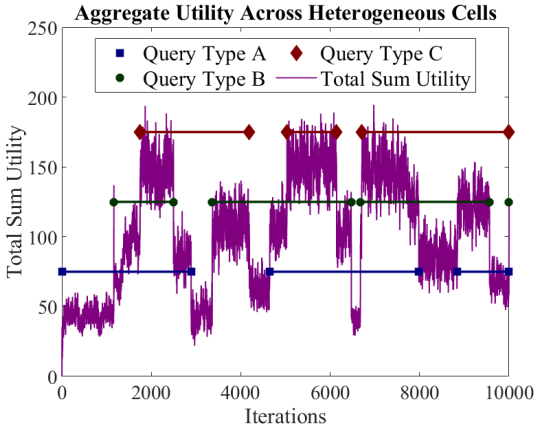

Fig. 9. (Synthetic Data) Plot of aggregate utility with query duration of types A, B, and C overlaid.

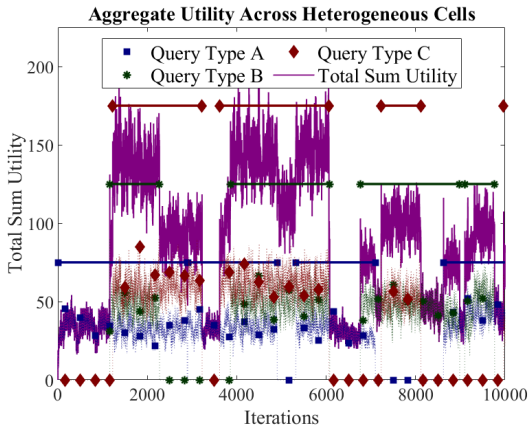

Fig. 10. (Trace Data) Utility comparison between queries using camera data for each user.

VI. Results: Synthetic Simulation and Trace Validation with Heterogeneous Cells

In this section we evaluate our system in more complex, heterogeneous settings. We consider multiple types of overlapping queries with different numbers of users in each cell responding, with users having different hit-ratios. We first consider synthetically generated hits and misses, and then evaluate the system with trace data from the camera network.

A. Heterogeneous Case: Synthetic Data

In this section we evaluate our system with synthetic data. We consider a reduced version of the system described above with nine cells, each with 21 users. All of the users have access to a single network-based GPU via the shared wireless link in their cell. Due to the reduced number of cells in the network in this simulation, we reduce the capacity of the GPU to $C_G = 160$ Mbps so that it may experience high loads. The wireless link in each cell has a capacity of $C_W = 25$ Mbps. Each mobile device has a CPU with a maximum local processing rate of $C_D = 16$ Mbps.

We define three query types in the system (A, B, C). A third of the users in each cell are assigned to each query type. When a query of a type matching that of a user is received by the user, a user has a 50% chance of accepting the query which results in a heterogeneous number of users responding in each cell. To define high, medium and low hit-ratio users, we analyzed our trace data and found that given traces from all six cameras, across all the objects of interest the hit-ratios for different objects in the traces were evenly split into those with $h_i \in [0.50, 1.0]$, $h_i \in (0.2, 0.5)$, and $h_i \in [0, 0.2]$. We defined these hit-ratio ranges as high, medium, and low, respectively, and divided assignments of high, medium, and low hit-ratio users equally across each cell. Users are randomly assigned a hit-ratio within their respective range to give us a random distribution of hit-ratios among users. For example, a high hit-ratio user is given a hit-ratio randomly between 0.5 and 1. The hits and misses are generated using the hit-ratio generator presented in Section IV by tuning p, q, p_h and p_l.

Query Types A, B, and C have lengths randomly assigned from a uniform distributed between 1000 and 4000 iterations, with pauses between individual queries chosen from a uniform distribution between 50 and 1000 iterations. The number of photos stored on each phone is obtained from a uniform distribution between 100 and 1000 images. A query may end when the designated length of the specified query has been spent, or all photos from responsive users have been processed. For simplicity of illustration in the following results, we consider the case with persistent energy supplies and $b = 0$. Conclusions from Section V regarding the battery parameter b may be extended and applied to cases of heterogeneous cells and queries. Fig. 9 illustrates the combined utility over time across the three query types, with start time and duration of each query overlaid for intuitive analysis.

While drastic increases in utility occur amid simultaneous queries, wireless link congestion compounded with competition for the GPU yields a collective utility equalling less than the sum of individual queries amid little to no resource scarcity. However, hit-ratio dual-path NUM provides the appropriate distribution of resources to achieve the maximum utility across users for varying queries when the estimated hit-ratio is included in the formulation.

B. Heterogeneous Case: Experimental Trace Validation

In this section, we define a scenario with the same number of users and cells as in the previous experiment using trace data from the camera network. We define three query types, but unlike our previous experiment, each of these query types correspond to an average hit-ratio. To realize this, every user is given a single 273 time-series image trace from the camera network. A third of the users are assigned trace data that will result in each of the hit-ratio ranges, so that one-third will be assigned as high, medium and low hit-ratio. These users are then mapped to the corresponding query type.

Queries are generated with the same frequency and duration as in the previous setting. The queries end when all images have been processed or the query time expires. To prevent users from being synchronized in their hits and misses, each user has a random starting point in the sequence of 273 images. The stream of images continues chronologically and wraps back to the beginning of the trace if necessary.

The separate and combined utilities of each query type are shown in Fig. 10. As expected, periods of simultaneous queries produce regions of greatly increased overall utility.

Our hit-ratio dual-path NUM framework quickly adjusts to the environment to reallocate congested resources appropriately.

VII. Related Work

A considerable amount of research exists regarding crowdsourcing for video processing where users are coupled to the cloud to assist in processing. In [1], [2], the primary objective is to minimize the time required to process all videos or images across all the mobile devices. These works are focused on labeling all images as fast as possible, regardless of the fact whether images contain the object of interest or not. While these algorithms may process more images in a shorter time frame, our optimization focuses on maximizing utility based on uploading positive hits as quickly as possible.

Network utility maximization (NUM) was first proposed by Kelly in his seminal works [5], [6], where shadow prices were introduced to provide fairness in resource allocation, which hold for static user settings and single-path source-destinations. Extensions including dynamic NUM [14] and multi-path NUM were proposed in [8] and [9], respectively, which provide theoretical foundations for our current work.

The approach proposed in [15] incorporates both energy costs and multi-path routing using NUM. The benefit of their work lies in faster convergence using the gradient descent method. However, the energy efficiency involves maximizing a single variable - summation of rates over transmitted power. The innovation of our work includes the adjustable energy variable b, based upon mission objectives.

In [16], a multi-path routing algorithm for wireless ad-hoc networks, called MP-DSR, considers network conditions and congestion through error rates to improve end-to-end reliability. In search for improvements in overall QoS, path selection depends only on accuracy. On the other hand, in our work, the selection of the transmission medium depends on the energy consumption, since the limited battery life of mobile devices is a very important factor to be taken into account.

Summarizing, the novelty of our work is along the following dimensions: (i) we propose the incorporation of a hit-ratio parameter for each individual user within the dual-path NUM framework; (ii) we design an energy conservation method for improved utility and sustained battery allotment; (iii) our model maximizes the rate at which the images with hits arrive at the edge cloud.

VIII. Conclusions

In this paper, we have considered the problem of optimally allocating resources to mobile users that participate in crowdsourced processing of images. The objective was to maximize the rate at which the images that contain the object of interest are transmitted to a central entity. The problem formulation incorporates the energy consumption and differentiates between the users depending on the hit-ratios of the objects of interest in every user's mobile device. We have also proposed ways to estimate the hit-ratio of every user. The dual-path approach that we propose is shown to provide significant performance improvements.

Appendix

Proof. Define the Lagrangian function following constrained optimization problem in Eq.(3), where the objective functions are given by Eq.(2), as

$$\begin{aligned}\tilde{\mathcal{L}}(\vec{\mathbf{x}},\vec{\lambda}) \;=\; & \textstyle\sum_{i=1}^{I}\sum_{j=1}^{2} h_i\theta_{ij}\log\left(\frac{x_{ij}}{\theta_{ij}}\right) \\ & + \textstyle\sum_{\ell=1}^{L}\lambda_\ell\left(c_l - \sum_{i=1}^{I}\sum_{j=1}^{2} E_{ij}^{\ell}R_{ij}^{\ell}x_{ij}\right),\end{aligned}$$

where $\vec{\lambda} = \{\lambda_1,\ldots,\lambda_L\}$ are the shadow prices for every resource, necessitating $\lambda_\ell \geq 0$ for all ℓ. Optimal solutions x_{ij}^* must satisfy $\left.\frac{\partial\tilde{\mathcal{L}}}{\partial x_{ij}}\right|_{\vec{\mathbf{x}}_i=\vec{\mathbf{x}}_i^*} = 0,\ \forall\ i,j,$ as well as $\lambda_\ell^*\left(c_\ell - \sum_{i=1}^{I}\sum_{j=1}^{2} E_{ij}^{\ell}R_{ij}^{\ell}x_{ij}^*\right) = 0,\ \forall\ \ell.$ Differentiating the Lagrangian yields $h_i\theta_{ij}^*\left(\frac{1/\theta_{ij}^*}{x_{ij}^*/\theta_{ij}^*}\right) - \sum_{\ell=1}^{L}E_{ij}^{\ell}R_{ij}^{\ell}\lambda_\ell^* = 0,\ \forall\ i,j,$ resulting in Eq.(9). Optimal shadow prices λ_ℓ^* can be found from the slackness conditions $\lambda_\ell^*\left(c_\ell - \sum_{i=1}^{I}\sum_{j=1}^{2} E_{ij}^{\ell}R_{ij}^{\ell}x_{ij}^*\right) = 0$ for every link ℓ. Consider all possible combinations of $\lambda_\ell^* = 0$, and $c_\ell - \sum_{i=1}^{I}\sum_{j=1}^{2} E_{ij}^{\ell}R_{ij}^{\ell}x_{ij}^* = 0$ with $\lambda_\ell^* > 0$. With the objective of maximization, assume as many resources as possible are fully utilized. Hence, we expect $\lambda_\ell^* > 0$, and $c_\ell - \sum_{i=1}^{I}\sum_{j=1}^{2} E_{ij}^{\ell}R_{ij}^{\ell}x_{ij}^* = 0$ for all depleted resources ℓ; otherwise, $\lambda_\ell^* = 0$ for slackness. These last conditions together with Eq.(9) provide the sufficient number of equations to determine all optimal x_{ij}^* and λ_ℓ^* values. The final solution depends on actual routing configuration. □

References

[1] Z. Lu, K. Chan, and T. La Porta, "A computing platform for video crowdprocessing using deep learning," in *Proc. of IEEE INFOCOM*, 2018.

[2] Z. Lu, K. Chan, R. Urgaonkar, and T. La Porta, "On-demand video processing in wireless networks," in *Proc. of IEEE ICNP*, 2016.

[3] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: A survey of the emerging 5G network edge cloud architecture and orchestration," *IEEE Communications Surveys Tutorials*, vol. 19, no. 3, 2017.

[4] N. Felemban, F. Mehmeti, H. Khamfroush, Z. Lu, S. Rallapalli, K. S. Chan, , and T. F. L. Porta, "Picsys: Energy-efficient fast image search on distributed mobile networks," *IEEE Transactions on Mobile Computing*.

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

[6] F. P. Kelly, A. K. Maulloo, and D. K. H. Tan, "Rate control for communication networks: Shadow prices, proportional fairness and stability," *Journal of the Oper. Research Society*, vol. 49, no. 3, 1998.

[7] D. P. Palomar and M. Chiang, "A tutorial on decomposition methods for network utility maximization," *IEEE JSAC*, vol. 24, no. 8, 2006.

[8] X. Lin and N. Shroff, "Utility maximization for communication networks with multipath routing," *IEEE Transactions on Automatic Control*, vol. 51, no. 5, 2006.

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

[10] R. Srikant and L. Ying, *Communication Networks: An Optimization, Control and Stochastic Networks Perspective*. Cambridge University Press, 2014.

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

[12] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. of IEEE CVPR*, 2015.

[13] P. Joshi, D. Colombi, B. Thors, L. Larsson, and C. Tornevik, "Output power levels of 4G user equipment and implications on realistic RF EMF exposure assessments," *IEEE Access*, vol. 5, 2017.

[14] N. Trichakis, A. Zymnis, and S. Boyd, "Dynamic network utility maximization with delivery contracts," in *Proc. of IFAC*, 2008.

[15] J. Zhang and H. Lee, "Energy-efficient utility maximization for wireless networks with/without multipath routing," *AEU - International Journal of Electronics and Communications*, vol. 64, 2010.

[16] R. Leung, J. Liu, E. Poon, A. Chan, and B. Li, "MP-DSR: A QoS-aware multi-path dynamic source routing protocol for wireless ad-hoc networks," in *Proc. of IEEE LCN*, 2001.