An Experts Learning Approach to Mobile Service Offloading

Cem Tekin, Mihaela van der Schaar
Electrical Engineering Department, University of California, Los Angeles
Email: cmtkn@ucla.edu, mihaela@ee.ucla.edu

***Abstract*—Mobile devices are more and more often called on to perform services which require too much computation power and battery energy. If delay is an important consideration, offloading to the cloud may be too slow and a better approach is to offload to a resource-rich machine in the proximity of the device. This paper develops a new approach to this problem in which the machines are viewed as a collection of experts - but experts that are coupled in space and in time: the current action at a given machine affects the future state of the given machine and of other machines to which the given machine is connected. At any time, given the state and unknown dynamics of the system, the experts available at that time should cooperatively pick the best available actions. Within this framework, we propose online learning algorithms that results in substantial savings in energy consumption.**

***Index Terms*—Online learning, coupled experts, exploration-exploitation tradeoff, distributed learning.**

I. Introduction

Mobile devices are more and more often called on to perform services (or tasks) - video analysis, gaming, online navigation - for which they are ill-suited, sometimes because the service requires complex computations but more often simply because the task demands too much energy from the battery of the mobile device. One approach to such problems is to offload the service to the cloud. However, if - as is often the case – delay is an important consideration, offloading to the cloud may be too slow and a better approach is to offload to a resource-rich machine in the proximity of the device. For example, the wireless LAN (802.11n) connection to a resource-rich machine will have a 400 Mbps nominal bandwidth, while the 3G connection to the cloud has a nominal bandwidth of about 15Mbps [1]. Moreover, even if the bandwidth is adequate, latency can be an issue when directly connecting to the cloud [2].

Because many devices will compete for the services of the same collection of machines (even if there are many machines, the number of devices will typically be many times the number of machines), a given machine will often find it necessary to offload its service requests to yet another machine. This creates an immensely complicated scheduling problem - made even more complicated by the fact that offloading needs will change over time because of changing mobile user activity. Moreover, the distribution of the mobile users' service requests is unknown a priori to the machines. They need to learn this online, based on the observed service requests, while simultaneously learning how to make optimal scheduling decisions to maximize the number of completed requests. We would like to emphasize that formalizing the considered mobile offloading problem as a multi-user optimization problem would require complete knowledge of the mobile users' service requests, as well as how these are spatially and temporally correlated at every moment in time. However, in practice, such information is not known a priori and, moreover, it changes over time. Hence, novel approaches are needed to effectively address the considered problem which go beyond conventional optimization solutions such as network utility maximization [3].

This paper develops a first approach to solve this challenging problem in which each machine is viewed as a collection of experts - but experts that are coupled in space and in time: For each machine, there is a set of experts, such that there is an expert for each state of the machine. At each time step, each expert chooses a set of actions for its machine. The actions chosen at a given machine (requests by mobile devices and by other machines) affects the future state of the given machine and of other machines to which the given machine is connected. This affects which experts will be available in the future. Within this framework, we propose two online learning algorithms and compare their performance with respect to the optimal scheduling policy computed based on the exact knowledge of the time-varying mobile users' service request distribution. We call this performance gap the regret. We establish analytic regret bounds for general time-varying Markovian mobile users' service requests. However, the resulting algorithm achieving this regret bound is centralized, requires global information about the new service requests, and has exponential complexity in the number of machines. These drawbacks make the algorithm impractical for settings with large number of machines - but it provides a benchmark for what is possible. The second algorithm avoids these drawbacks by forming clusters of cooperative machines that acts independently of each other: it is distributed, requires only local information about the new service requests, and its complexity only depends on the number of machines located in each cluster. We show that when the mobile user service request rate is low, this algorithm performs almost as good as the centralized one.

To the best of our knowledge, our paper is the first work that considers mobile service offloading as a learning prob-

lem with multiple mobiles and multiple machines distributed over a geographic region.

The remainder of this paper is as follows. The problem formulation is given in Section II, along with the definition of regret. In Section III, a centralized online learning algorithm and its regret is given. In Section IV, a distributed online learning algorithm is given. Numerical results are given in Section V. Related work is given in Section VI, and concluding remarks are made in Section VII.

II. Problem Formulation

We consider a system given in Fig. 1 that consist of K cooperative machines distributed over a bounded geographic region $\mathcal{R}$ indexed by the set $\mathcal{K} = \{1, 2, \ldots, K\}$. At each discrete time step $t = 1, 2, \ldots$, a random number of mobile users from different locations in $\mathcal{R}$ request services from one of these machines based on the proximity of the machine. In doing so, the goal of a mobile user is to reduce its battery cost of performing the service while incurring a small amount of communication cost due to data transmission to and from the machine. The region $\mathcal{R}$ is partitioned into K regions given by $\mathcal{R}_K = (R_1, \ldots, R_K)$, where R_i is the service region of machine i. Any mobile user in region R_i can only request service from machine i. We assume that a machine processes the tasks of mobile users connected to it in FIFO (first in first out) way. For any delay constraint λ, each machine has a computational capacity L_λ, which is the maximum number of tasks in its queue that can be processed with at most λ delay. At any instance of time, the machine will deny the request of a user for which it cannot meet the delay requirement. Then, that user will process its own data, which eventually consumes more energy for the user. It is possible that a user can send its task to be processed by a machine and that machine can forward the task to one of the machines that is k hops away. This can either be done by directly communicating with the machine that is k hops away, or by routing the request hop-by-hop to the machine that is k hops away. Let $\delta_{k,\lambda}$ be the number of services that can be completed in the time it takes for a machine to send its mobile's service request to a machine that is k hops away and receive back the completed service. We assume that $\delta_{k,\lambda}$ increases in k.

We assume that the number of mobile users that requests service in the area covered by each machine evolves according to an unknown, time varying Markov process. This assumption covers both the temporal and spatial aspects of the mobile user activity. A mobile user which is in the coverage area of a machine may move into the coverage area of another machine, or a mobile user may request a service long after it has been in the coverage area of a machine i (For example, a person comes to a cafe, eats her lunch, starts mobile services, then walks to a nearby bookstore without turning off the mobile services). A time varying Markovian mobility model is proposed based on real measurements in [4]. Similarly, a time varying community mobility model which exhibits Markovian dynamics is proposed in [5].

Let $N_i(t)$ be the random variable which denotes the number of users that requests service from machine i at

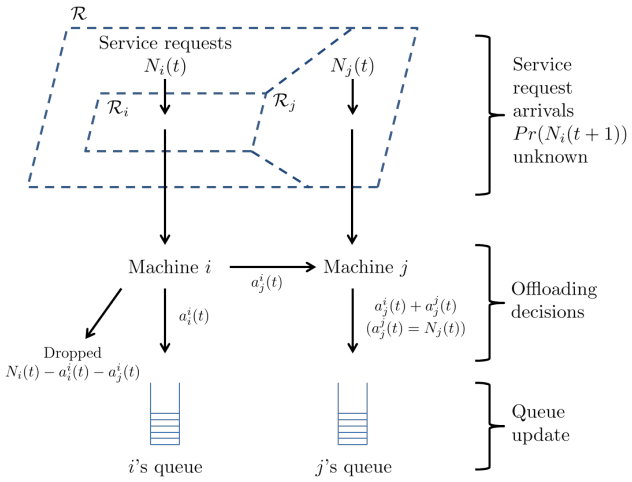

Fig. 1. System model.

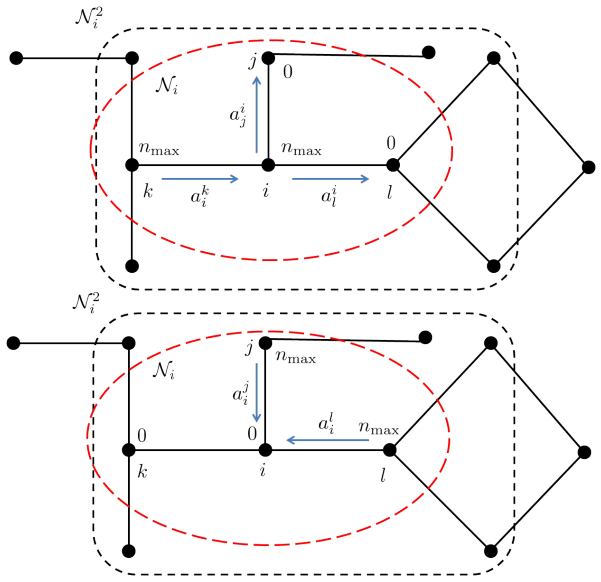

Fig. 2. Illustration of how i's forwarding decision is affected by a neighbor of j.

time slot t. Let $n_{\max}$ denote the maximum possible number of users in the coverage area of any machine at any time slot t. The mobile user activity in the system can be modeled as a discrete time Markov chain with state space $\mathcal{M} = \{\boldsymbol{n} = (n_1, \ldots, n_K) : 0 \leq n_i \leq n_{\max}, i \in \mathcal{K}\}$, and *mobile* state transition probabilities

$$p^t_{\boldsymbol{n}',\boldsymbol{n}} := Pr\left(N_1(t+1) = n_1, \ldots, N_K(t+1) = n_K \middle| N_1(t) = n'_1, \ldots, N_K(t) = n'_K\right), \quad (1)$$

which are unknown. Let P^t be the transition probability matrix at time t, whose elements are given by (1). We assume that P^t is irreducible for any t, and the state space $\mathcal{M}$ is known. For the simplicity of presentation we assume that users are homogeneous which means that the delay requirement of all users are fixed and the same, but our results can easily be extended to the more general case.

Let $x_i(t)$ be the length of the processing queue of machine i at the beginning of time slot t. We assume that machine i has processing capacity $c_i = c$ tasks at each time slot.

Let $\mathcal{N}_i$ be the set of machines which are neighbors of machine i including itself. Let $\mathcal{N}_i^k$ be the set of machines that are at most k hops away from i. The sets $\mathcal{N}_i$ and $\mathcal{N}_i^k$ are shown in Fig. 2. Let $a_i^l(t)$ be the number of new mobile user requests in the region of machine l which are admitted to the processing queue of machine i at time t. Let $\boldsymbol{a}_i(t) = (a_i^l(t))_{l\in\mathcal{N}_i^k}$, and $\boldsymbol{a}(t) = (\boldsymbol{a}_1(t), \ldots, \boldsymbol{a}_K(t))$. Using this information, we can calculate the length of the processing queue for machine i at the beginning of slot t as $x_i(t) = x_i(t-1) + \sum_{l\in\mathcal{N}_i^k} a_i^l(t-1) - c$.[1] Due to the delay constraint, the task offloading actions at time t should be such that $x_i(t) + a_i^i(t) + \sum_{l\in\mathcal{N}_i^k - \{i\}} a_i^l(t) \leq L_\lambda$, $x_i(t) + \sum_{l\in\mathcal{N}_i^k - \mathcal{N}_i^{z-1}} a_i^l(t) \leq L_\lambda - \delta_{z,\lambda}$, for all $i \in \mathcal{K}$, $z = 2, \ldots, k-1$. Also, each service request offloaded from l to i, which is z hops away, should be assigned to a slot in the queue that is greater than $\delta_{z,\lambda}/2$, since it will arrive to queue of i only after $\delta_{z,\lambda}/2$ services are completed by i. It is easy to see that the machines are both spatially and temporally coupled. For example, consider the configuration given in Fig. 2. Depending on the service requests, in the upper figure, machines i and k experience high number of new requests. Since the neighbors j and l of i have no new requests, the congestion on i and k is alleviated by forwarding services to j and l. Here i's forwarding actions affect both the actions and the queue lengths of j and l. Differently, in the lower figure j and l experience high number of service requests, therefore, they forward the service requests to i. But if large number of service requests arrive to i in the next time slot, then i may not be able to serve all its service requests due to serving j and l's requests. Let $e_i(t)$ be the number of new service requests in region R_i that are not served by any machine. The total loss of the system by time T is given by $\sum_{t=1}^T \sum_{i=1}^K e_i(t)$. Our goal is to design an online learning algorithm that minimizes $E\left[(\sum_{t=1}^T \sum_{i=1}^K e_i(t))/T\right]$. Let $\mathcal{S} := \{(\boldsymbol{x}, \boldsymbol{n}) : x_i \leq L_\lambda, n_i \leq n_{\max}, i \in \mathcal{K}\}$ denote the state space of the system. We call $\boldsymbol{x}$ the *queue state*, $\mathcal{X}$ the queue space and $\boldsymbol{n}$ the *new service requests state*. Given the state $\boldsymbol{s}$, the set of available actions such that each machine can offload to a machine within k hops away is

$$A_k(\boldsymbol{s}) := \Bigg\{ \boldsymbol{a} : x_i + \sum_{l\in\mathcal{N}_i} a_i^l \leq L_\lambda, x_i + \sum_{l\in\mathcal{N}_i^k - \mathcal{N}_i^{z-1}} a_i^l \leq L_\lambda - \delta_{z,\lambda}, \forall z = 2, \ldots, k-1, \sum_{l\in\mathcal{N}_i^k} a_l^i \leq n_i, \forall i \in \mathcal{K} \Bigg\}.$$

Our goal is to learn the best action given state $\boldsymbol{s}$. For each state $\boldsymbol{s}$, we can define an expert that recommends an action $\boldsymbol{a}_i$ for machine i. The goal of each expert is to learn the best action $\boldsymbol{a}_i^*$ given $\boldsymbol{s}$ (or a subset of $\boldsymbol{s}$ as we will show in Section IV) for machine i such that the joint action profile $\boldsymbol{a}$ is optimal. Let $A_k = \cup_{\boldsymbol{s}\in\mathcal{S}} A_k(\boldsymbol{s})$, be the union of the set of possible actions over all queue states. The queue length is a deterministic function of the actions and the new service requests are independent of the actions taken. Therefore, state transition probability from $\boldsymbol{s}$ to $\boldsymbol{s}'$ given action $\boldsymbol{a}$ is taken at time t is given by

$$Q^t(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}') := p^t_{\boldsymbol{n},\boldsymbol{n}'} \Pi_{i=1}^K I\left(x_i(t+1) = x_i(t) + \sum_{l\in\mathcal{N}_i^k} a_i^l(t) - c \right), \tag{2}$$

where $I(.)$ is the indicator function. Note that $Q^t(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}')$ is either zero or is equal to $p^t_{\boldsymbol{n},\boldsymbol{n}'}$ for some $\boldsymbol{n}, \boldsymbol{n}' \in \mathcal{M}$.

The problem we consider is much harder than the optimization version of the same problem. In the optimization version P^t is known for all t up to T. Then, the problem can be formulated as a Markov decision process (MDP) and solved by dynamic programming.

A. Assumptions on State Transition Probabilities

As we mentioned before, transition probabilities are learned online. If they change too fast, then no algorithm can learn them accurately. Therefore, we assume that the mobile user activity transition probabilities are slowly changing which is given in the following assumption.

Assumption 1: $||P^{t'} - P^t||_\infty \leq L|t'-t|^\alpha$, for some known $L > 0$ and $\alpha > 0$, where $||.||_\infty$ is the maximum absolute row sum norm.

We also have the following ergodicity assumption on the transition probabilities.

Assumption 2: For every t, P^t is ergodic with expected mixing time at most τ and expected cover time at most τ_{cov}.

This assumption says that every state is reachable from every other state. Indeed, the transition probability matrix in [4] derived from real-world data is ergodic. The following assumption is used to analyze the dependence of the regret on the variability of the transition probabilities.

Assumption 3: Let $Z^t := [I - P^t - P^t_\infty]^{-1}$, where $P^t_\infty = \lim_{l\to\infty} \frac{1}{l} \sum_{k=1}^l (P^t)^k$. Z^t is called the fundamental matrix associated with P^t. Assume there exists $z < \infty$ such that $||Z^t||_\infty \leq z$ for all t.

These assumptions together imply that, after enough observations, all the states are reached so that the estimates of transition probabilities can be formed, while these estimates will not be very different from the true values at the current time.

B. Definition of Regret

In this subsection we define the benchmark policy that we compare against. Let Π_d be the set of deterministic, stationary policies such that for any $\pi \in \Pi_d$, $\pi(\boldsymbol{s}) \in A_k(\boldsymbol{s})$. Let Π_d^m be the set of deterministic policies that can change at times $t = \zeta m + 1$, $\zeta = 0, 1, 2, \ldots$. These policies are not stationary, but they are stationary for m time slots, thus we call them m-stationary policies. The family of policies $\{\Pi_d^m\}_{m=1,2,\ldots}$ includes Π_d. Then, the total expected loss of the best m-stationary policy by time T is given by $J_T^m = (1/T) \min_{\pi\in\Pi_d^m} E[\sum_{t=1}^T e(\boldsymbol{n}(t), \pi(\boldsymbol{s}(t)))]$, where

[1] When a new request is accepted, its position in the queue is reserved. Hence, even if the service offloaded to j has not arrived to i, i reserves a slot in its queue for that service.

$e(\boldsymbol{n}, \pi(\boldsymbol{s}))$ is the number of service requests that are denied at state $\boldsymbol{s}$ by policy π. The regret of a learning algorithm α up to time T, whose action at time t is $\boldsymbol{a}(t)$, is given by

$$R_T^m = E\left[\sum_{t=1}^{T} e(\boldsymbol{n}(t), \boldsymbol{a}(t))\right]/T - J_T^m. \tag{3}$$

Our goal is to design an online learning algorithm whose regret is small.

III. Online Learning of the Optimal Offloading Policy

If the state transition probabilities were known, then the optimal offloading policy can be computed by dynamic programming. However since the mobile user activity is unknown and varying in time, it is not possible to know the transition probabilities beforehand. In this section we assume that there is a controller which learns about the mobile user activity over time, and makes service offloading decisions based on the state of the system. We propose an online learning algorithm for this controller called *centralized service offloading* (CSO).

Centralized Service Offloading (CSO):
1: Input: T, m_T
2: Initialize: $t = 1$, $m = 1$, $(\boldsymbol{x}(0), \boldsymbol{n}(0)) = (\boldsymbol{0}, \boldsymbol{0})$
3: **while** $t \geq 1$ **do**
4: Observe $(\boldsymbol{x}(t), \boldsymbol{n}(t))$.
5: **if** $m = 1$ **then**
6: Select an action myopically (machine i's goal is to serve the direct service requests to itself only) based on $(\boldsymbol{x}(t), \boldsymbol{n}(t))$.
7: **else**
8: Select an action based on the policy computed according to (5).
9: **end if**
10: **if** $t = mT/m_T + 1$ **then**
11: $m = m + 1$
12: Compute the state transition probabilities $\hat{p}_{m-1}(\boldsymbol{n}, \boldsymbol{n}')$ (hence $Q_{m-1}(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}')$) according to (4).
13: Compute the optimal policy for round m according to (5).
14: **end if**
15: $t = t + 1$.
16: **end while**

Fig. 3. Pseudocode for the CSO algorithm.

The pseudocode of CSO is given in Fig. 3. It is similar to the ORDP algorithm given in [6], which uses robust dynamic programming [7] to find the optimal policy given an uncertainty region for the transition dynamics. The difference is that CSO does not need to randomize and follow the perturbed leader as in ORDP since the reward is a fixed deterministic function of the action, and CSO needs to use only a window of transition probability estimates, rather than exploiting the full sequence of estimates, since the difference between transition probabilities at two time slots t and t' that are far from each other can be large. Given the final time T, CSO partitions $\{1, 2, \ldots, T\}$ to m_T rounds. We let $T_R := T/m_T$ to be the length of a single round, and assume that it is an integer. Therefore, the first round will contain times $\{1, \ldots, T/m_T\}$, the second round will contain times $\{T/m_T + 1, \ldots, 2T/m_T\}$, and so on. Let $\hat{p}_m(\boldsymbol{n}, \boldsymbol{n}')$ be the transition probability estimate from state $\boldsymbol{n}$ to $\boldsymbol{n}'$ calculated in the mth round. We have

$$\hat{p}_m(\boldsymbol{n}, \boldsymbol{n}') = \frac{1 + \sum_{t=(m-1)T/m_T+1}^{mT/m_T-1} I(\boldsymbol{n}(t+1) = \boldsymbol{n}',\ \boldsymbol{n}(t) = \boldsymbol{n})}{|\mathcal{M}| + \sum_{t=(m-1)T/m_T+1}^{mT/m_T-1} I(\boldsymbol{n}(t) = \boldsymbol{n})}. \tag{4}$$

In the first round the algorithm acts myopically since the transition probability estimates to solve the robust dynamic program is not obtained yet. If possible, it assigns each service arriving to each machine to the machine itself. If not, it denies the mobile's service request. This way it is guaranteed that the number of service denials in the first round is at most $K(n_{\max} - c)T/m_T$. Since this bound is trivial, we neglect the regret in the first round and focus on the regret incurred in rounds after the first round. At the beginning of the mth round, CSO solves the following robust dynamic program using the estimated transition probabilities in round $m - 1$.

$$V_m(\boldsymbol{s}) = \arg\max_{\boldsymbol{a} \in A_k(\boldsymbol{s})} \left(-e(\boldsymbol{n}, \boldsymbol{a}) + \inf_{Q \in \mathcal{D}_m(\tau, \tau_{\text{cov}})} \left(\sum_{\boldsymbol{s}'} V_m(\boldsymbol{s}') Q(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}')\right) - V_m(\boldsymbol{s}^*)\right), \tag{5}$$

where $\boldsymbol{s}^*$ is a fixed state, $V_m(\boldsymbol{s}^*)$ is the normalization term and $\mathcal{D}_m(\tau, \tau_{\text{cov}})$ is the uncertainty set, i.e., the set that is believed to have the actual state transition probabilities for round m with a high probability. The solution of (5) produces an estimated optimal policy for each expert. Specifically, the action that the expert of machine i recommends in state $\boldsymbol{s}$ is denoted by $\gamma_{m,i}(\boldsymbol{s})$. Let $\gamma_m = (\gamma_{m,1}, \ldots, \gamma_{m,K})$. Then, based on the state at time t in round m, action $\gamma_m(\boldsymbol{s}(t))$ is taken by the controller. In order to bound the regret of the online learning algorithm, we will first bound the regret of any transition probability matrix lying in the uncertainty set $\mathcal{D}_m(\tau, \tau_{\text{cov}})$. Then, we will bound the probability that the true transition probabilities lies in this uncertainty set. Specifically, we let the uncertainty set for mobile state transition probabilities to be

$$(\mathcal{C}_m)_{\boldsymbol{n}, \boldsymbol{n}'} := \Big[\left(\hat{p}_{m-1}(\boldsymbol{n}, \boldsymbol{n}') - 2L\left(T_R\right)^{\alpha}\right)^+, \min\left\{\hat{p}_{m-1}(\boldsymbol{n}, \boldsymbol{n}') + 2L\left(T_R\right)^{\alpha}, 1\right\}\Big],$$

for all $\boldsymbol{n}, \boldsymbol{n} \in \mathcal{M}$, where $(x)^+ = \max\{0, x\}$. This directly defines an uncertainty set $\mathcal{D}_m$, for which we have $(\mathcal{D}_m)_{\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}'}$ is equal to zero or $(\mathcal{C}_m)_{\boldsymbol{n}, \boldsymbol{n}'}$ for some $\boldsymbol{n}, \boldsymbol{n} \in \mathcal{M}$. Note that $\mathcal{D}_m$ can contain transition probability matrices that are periodic or that have mixing and cover time greater than τ and τ_{cov}. The regret for these transition probability matrices can be large. By Assumption 2, it is known that the actual transition probability matrices are aperiodic and their mixing and cover times are at most τ and τ_{cov}. Therefore we let $\mathcal{D}_m(\tau, \tau_{\text{cov}}) \subset \mathcal{D}_m$ be the set of transition probability matrices whose mixing and cover times are at most τ and τ_{cov}. We have the following lemma.

Lemma 1: Consider any $\epsilon > 0$. If CSO is run with $m_T = T/T_R$, where $T_R = \left\lceil \left(\frac{(0.5 n_{\max}^{2K} \tau_{\text{cov}} \log\left(2 n_{\max}^{K} |A_k|/\epsilon\right)}{L^2 \epsilon} \right)^{\frac{1}{2\alpha+1}} \right\rceil$,[2] we have for every round m with probability

$$P(\mathcal{D}_m) \geq (1-\epsilon)^2,$$

the true set of transition probabilities in round m will lie in $\mathcal{D}_m$.

Proof:

Let X_1 be the covering time of the Markov chain P^t, $t = (m-1)T/m_T + 1, \ldots, mT/m_T$, i.e., the time all states are visited starting from the beginning of round $m-1$. Similarly let X_i be the ith covering time, that is the time it takes to visit all states after visiting all states for the $i-1$th time. Consider any $a > 0$. Let Y be the time when all the states of the Markov chain is observed at least $D := (\log(a) m_T^{2\alpha} (n_{\max})^{2K})/(L^2 T^{2\alpha})$ times. For simplicity, we assume that this number is an integer. If this is not an integer, then it can be rounded up to the smallest integer that is greater than it. Then we have $Y \leq X_1 + \ldots + X_D$. Using Markov inequality, we get $P(Y > T/m_T) \leq P(X_1 + \ldots + X_D > T/m_T) \leq (D \tau_{\text{cov}} m_T)/T$. Therefore, the probability that all the states are observed at least D times in a round is greater than or equal to $1 - D\tau_{\text{cov}} m_T/T$. Using a Chernoff bound, it can be shown that given all states are observed at least D times, the probability that the true transition probability matrix will lie in $\mathcal{D}_m$ is $(1 - 2(n_{\max})^K |A_k|/a^2)$. Combining these we have

$$P(\mathcal{D}_m) := \left(1 - \frac{(m_T)^{2\alpha+1} (n_{\max})^{2K} \tau_{\text{cov}} \log a}{L^2 T^{2\alpha+1}}\right) \times \left(1 - \frac{2(n_{\max})^K |A_k|}{a^2}\right).$$

We get the final result by setting $a = \sqrt{\frac{2 n_{\max}^K |A_k|}{\epsilon}}$, and T_R to the specified value. ∎

Even when the true transition probabilities lies in $\mathcal{D}_m$, the solution of the robust dynamic program is suboptimal. In the next theorem we bound the regret by bounding the suboptimality due to uncertainty and the suboptimality due to the correct transition probabilities not being in the uncertainty region $\mathcal{D}_m$.

Theorem 1: Consider any $\delta > 0$. Let $m_T = T/T_R$, where $T_R = \left\lceil \left(\frac{(2K(n_{\max}-c) n_{\max}^{2K} \tau_{\text{cov}} \log\left(8K(n_{\max}-c) n_{\max}^K |A_k|/\delta\right)}{L^2 \delta} \right)^{\frac{1}{2\alpha+1}} \right\rceil$. Under Assumptions 1, 2 and 3, if $(T_R)^{1+\alpha} \geq (4e\tau)/(L 2^\alpha)$ and $L \leq \frac{2\delta^{3\alpha+1}/(4K(n_{\max}-c))^{3\alpha+1}}{(2 n_{\max}^{2K} \tau_{\text{cov}} \log(8K(n_{\max}-c) n_{\max}^K |A_k|/\delta))^\alpha (z+1)^{2\alpha+1}}$, the regret of CSO (given in (3)) with respect to the optimal T_R stationary policy is upper bounded by $R_T^{T_R} \leq \delta$, where z is the number given in Assumption 3.

Proof: The proof is similar to the proof of Theorem IV.1 in [6]. The difference is that we use m_T to control the variation in uncertainty in a round. When we have more rounds, this means that the variation in the transition probabilities will be small. However, the shorter each round, less explorations will be performed, so the sample mean transition probability estimates may not be accurate. We can balance this tradeoff by a careful choice of m_T. Whenever the true transition probabilities lies in the region of uncertainty $\mathcal{D}_m(\tau, \tau_{\text{cov}})$, using a similar analysis to [6], we can bound the regret by $K(n_{\max} - c)\left((z+1)L\left(2T_R\right)^\alpha\right)$. Whenever the true transition probabilities are not in $\mathcal{D}_m(\tau, \tau_{\text{cov}})$, since in the worst case each machine can process c tasks at each time slot, the regret is bounded by $K(n_{\max} - c)$. Combining these two we get

$$\begin{aligned} R_T^m \leq & P(\mathcal{D}_m) K(n_{\max} - c)\left((z+1)L\left(2T_R\right)^\alpha\right) \\ & + (1 - P(\mathcal{D}_m)) K(n_{\max} - c), \end{aligned}$$

The result follows form substituting the values for T_R and L given in the statement of the theorem, and calculating $P(\mathcal{D}_m)$ according to Lemma 1. ∎

The robust dynamic program can be solved via linear programming. However the state space is exponential in the number of machines which makes it computationally prohibitive for large number of machines. Estimating the state transition probabilities is same as estimating the mobile transition probabilities because of the relation given in (2). Note that this bound holds given that L is sufficiently small as given in Theorem 1. Indeed, we require $L = o(\delta^{3\alpha+1})$ to achieve δ regret.

Since centralized learning is computationally expensive and also requires mobile user activity information from all machines, in the next section we will propose a decentralized offloading algorithm in which every machine decides which mobile users to offload based on the mobile user activity in its neighborhood.

IV. Online Decentralized Service Offloading

In this section, we propose an online decentralized service offloading algorithm where machines form clusters. Each cluster has a cluster head machine which tells the other machines how to jointly offload the mobile user service requests arriving to that cluster. The clusters are formed in the initialization phase. A k-partition is a partition of the machines such that for each cluster in the partition, each machine is no more than $2k$ hops away from another machine in that cluster. We call a k-partition with minimal number of sets the minimal k-partition. Let $\mathcal{P}_k$ denote the set of cluster head machines in the minimal k-partition. In our analysis we neglect all machines with no other neighbors than itself, since their best decision is to always use their processing capacity for the new service requests in their own regions. We assume that the minimum degree of the graph is $v > 0$, i.e., each machines has at least v neighbors.

For simplicity of analysis, in this section we consider a binary, independent mobile users' service request model, where there are $n_{\max}$ new requests to the region of each machine i with probability $p_i(t)$ and 0 new requests with

[2] $\lceil r \rceil$ is the smallest integer that is larger than or equal to r.

probability $1-p_i(t)$ at time t. Then, Assumption 1 becomes

$$|p_i(t) - p_i(t')| \leq L|t-t'|^{\alpha}, \tag{6}$$

for all $i \in \mathcal{K}$, for some $L > 0$ and $\alpha > 0$. We also assume that $p_i(t) \leq p^*$ for all t and $i \in \mathcal{K}$ for some known $p^* < 1$. We will show that when p^* is small enough, the offloading policy computed only using the state information of each cluster independently from the other clusters will be close to optimal.

Let $\mathcal{N}_i^k$ be the set of machines in a k-cluster where i is the cluster head machine. Recall that in the previous section we assumed that $\mathcal{N}_i^k$ is the set of neighbors of i that are at most k hops away from i. In this section, we slightly generalize $\mathcal{N}_i^k$ so i does not have to be the center machine that is at most k hops away from any other machine, but it can be any machine in the cluster that is at most $2k$ hops away from any other machine in the cluster. The per time step processing capacity of the machines in $\mathcal{N}_i^k$ is equal to $|\mathcal{N}_i^k|c$, while per time step expected number of new service requests to $\mathcal{N}_i^k$ is upper bounded by $p^*|\mathcal{N}_i^k|n_{\max}$.

First, we will compute the performance of a myopic cluster policy which only requires knowledge about p^* and the new service requests to $\mathcal{N}_i^k$ for each $i \in \mathcal{P}_k$. According to the myopic cluster policy, at every time slot t, every machine $j \in \mathcal{N}_i^k$ reports $n_j(t)$ to the head machine i. Then i makes the following scheduling. All the new requests to $\mathcal{N}_i^k$ is arbitrarily distributed to the machines in $\mathcal{N}_i^k$, with each machine being assigned at most c tasks. This way, at each time at most $|\mathcal{N}_i^k|c$ tasks will be processed by cluster i, while all the excess tasks will be denied by the machines. With this scheduling scheme, the queue length at any machine at any time slot will be at most c. We assume that k is such that $L_\lambda - \delta_{k,\lambda} \geq c$, so that the delay requirement is satisfied. This policy does not take into account the estimation of time varying mobile users' service request probabilities. Nor does it exploit the queue capacities of each machine efficiently. Due to these reasons, performance of the myopic cluster policy forms a lower bound for the performance of our distributed online learning algorithm that we will describe in the following section.

The following theorem provides a bound on the expected average loss of the myopic cluster policy based on the value of p^* and k. Unlike the previous section for which the regret is defined with respect to the best m-stationary policy, here the regret $R_T := \max_{m=1,2,\ldots} R_T^m$ is defined with respect to the optimal policy in $\{\Pi_d^m\}_{m=1,2,\ldots}$.

Theorem 2: Given $p^* n_{\max} < c$, the regret R_T of the myopic cluster policy γ_{MC} given in (3) is upper bounded by $(n_{\max}-c)\sum_{i\in\mathcal{P}_k} e^{-2|\mathcal{N}_i^k|(c/n_{\max}-p^*)^2}|\mathcal{N}_i^k|$.

Proof: Let X_j denote the number of new service requests to machine j. The event $E_i = \left\{\sum_{j\in\mathcal{N}_i^k} X_j \leq c|\mathcal{N}_i^k|\right\}$ implies that all the tasks will be processed in $\mathcal{N}_i^k$, and the queue states will be $x_j = 0$ for $j \in \mathcal{N}_i^k$ after the tasks are processed. Since there is no increase in queue states and since all the packets are served, under the event E_i the myopic policy is optimal for $\mathcal{N}_i^k$. Myopic policy is only suboptimal for $\mathcal{N}_i^k$ on event E_i^c. But whenever E_i^c happens the contribution to the regret can be at most $(n_{\max}-c)|\mathcal{N}_i^k|$. Since the new requests to each machine is independent of each other, by using a Chernoff bound we get $P(E_i^c) \leq e^{-2|\mathcal{N}_i^k|(c/n_{\max}-p^*)^2}$. It can be shown that $R_T \leq E[\sum_{i\in\mathcal{P}_k}|\mathcal{N}_i^k|(n_{\max}-c)I(E_i^c)]$, which gives the main regret bound. ∎

The condition $p^* n_{\max} < c$ implies that expected number of new service requests to each machine is less than its processing capacity. Although $n_{\max}$ new requests can happen at each machine, the probability of having such requests is small, which implies that with a high probability not all the machines are *congested* at the same time. Although this regret bound depends on parameters like $|\mathcal{N}_i^k|$ and $\mathcal{P}_k$, whose exact values depend on the structure of the network graph, we can get bounds on these values for special types of graphs. The following corollary gives the regret bounds when $k = 1$, and the graph is such that every head machine in the partition $\mathcal{P}_1$ has at least v neighbors are connected to it.

Corollary 1: Consider the case when $k = 1$, i.e., each cluster consists of a center node i and its neighbors. Then, for $p^* n_{\max} < c$, the regret of the myopic cluster policy is upper bounded by $Ke^{-2(v+1)(c/n_{\max}-p^*)^2}(n_{\max}-c)$.

Proof: Since each $i \in \mathcal{P}_1$ has at least v neighbors, we have $\mathcal{P}_1 \leq K/(v+1)$. The result follows from Theorem 2. ∎

A. The Distributed Service Offloading Algorithm (DSO)

In this subsection, we propose the distributed service offloading algorithm (DSO), whose pseudocode is given in Fig. 4. DSO is similar to CSO, but the difference is that there is no central controller. Instead, given the parameter $k > 0$, at the initialization phase the machines form the minimal k-cluster, $\mathcal{P}_k$ by exchange of their location and neighbor information. The head machine of each cluster acts as a controller.

Distributed Service Offloading (DSO):
1: Input: T, m_T, k, p^*
2: Initialize: $t = 1$, $m = 1$, $(\boldsymbol{x}(0), \boldsymbol{n}(0)) = (\mathbf{0}, \mathbf{0})$
3: Cluster formation: form $\mathcal{P}_k$, each machine knows to which cluster it belongs.
4: **while** $t \geq 1$ **do**
5: **for** $i \in \mathcal{P}_k$ **do**
6: Observe $(\boldsymbol{x}_i(t), \boldsymbol{n}_i(t))$.
7: **if** $m = 1$ **then**
8: Select an action according to the myopic cluster policy.
9: **else**
10: Select an action based on the policy computed according to (8).
11: **end if**
12: **if** $t = mT/m_T + 1$ **then**
13: $m = m + 1$
14: Compute the state transition probabilities $\hat{p}_{m-1,i}(\boldsymbol{n}_i, \boldsymbol{n}_i')$ (hence $\hat{Q}_{m-1,i}(\boldsymbol{s}_i, \boldsymbol{a}_i, \boldsymbol{s}_i')$) according to (7).
15: Compute the optimal policy for round m according to (8).
16: **end if**
17: **end for**
18: $t = t + 1$.
19: **end while**

Fig. 4. Pseudocode for the DSO algorithm.

Given the final time T, DSO partitions $\{1, 2, \ldots, T\}$ to

m_T rounds. For simplicity we assume that $R_T = T/m_T$ is an integer, which is the length of each round in time slots. Since each cluster head acts based on the observations and actions within the cluster, we use subscript i to denote parts of the state, queue state, mobile service requests and transition probability estimates restricted to the machines in $\mathcal{N}_i^k$. Similar to CSO, each $i \in \mathcal{P}_k$ forms transition probability estimates $\hat{p}_{m,i}(\boldsymbol{n}_i, \boldsymbol{n}_i')$ based on its observations from $\mathcal{N}_i^k$ in the mth round, where $\boldsymbol{n}_i = (n_j)_{j \in \mathcal{N}_i^k}$. Due to the binary, independent new mobile users' service request assumption that we have for this section, the mobile state transition probabilities can be written as

$$\hat{p}_{m,i}(\boldsymbol{n}_i, \boldsymbol{n}_i') = \Pi_{j \in \mathcal{N}_i^k} \hat{p}_m(j, n_j'), \tag{7}$$

where[3] $\hat{p}_m(j, n_j') = (m_T/T) \sum_{t=(m-1)T/m_T+1}^{mT/m_T-1} I(n_j(t) = n_j')$.

This follows from the fact that the present state does not depend on the past state and it is the product of the probabilities of new service requests to each machine in set $\mathcal{N}_i^k$. In the first round the algorithm runs the myopic cluster policy since nothing is known about the actual service request probabilities. At the beginning of the mth round, $i \in \mathcal{P}_k$ solves the following robust dynamic program using the estimated transition probabilities in round $m-1$.

$$V_{m,i}(\boldsymbol{s}_i) = \underset{\boldsymbol{a}_i \in A_i(\boldsymbol{s}_i)}{\arg\max} \left(-e_i(\boldsymbol{n}_i, \boldsymbol{a}_i) + \inf_{Q \in \mathcal{D}_{m,i}} \left(\sum_{\boldsymbol{s}_i'} V_{m,i}(\boldsymbol{s}_i') Q(\boldsymbol{s}_i, \boldsymbol{a}_i, \boldsymbol{s}_i') \right) - V_{m,i}(\boldsymbol{s}_i^*) \right), \tag{8}$$

where $\boldsymbol{s}_i^*$ is a fixed state, $V_{m,i}(\boldsymbol{s}_i^*)$ is the normalization term, $A_i(\boldsymbol{s}_i)$ is the set of actions for the cluster of machine i given the state is $\boldsymbol{s}_i$, and $\mathcal{D}_{m,i}$ is the uncertainty set whose elements are given as

$$(\mathcal{D}_{m,i})_{j,0} = \left[\max\left\{ 1-p^*, \left(\hat{p}_{m-1}(j,0) - 2L\left(\frac{T}{m_T}\right)^{\alpha} \right)^+ \right\}, \min\left\{ 1, \hat{p}_{m-1}(j,0) + 2L\left(\frac{T}{m_T}\right)^{\alpha} \right\} \right],$$

and

$$(\mathcal{D}_{m,i})_{j,n_{\max}} = \left[\left(\hat{p}_{m-1}(j,0) - 2L\left(\frac{T}{m_T}\right)^{\alpha} \right)^+, \max\left\{ \hat{p}_{m-1}(j,n_{\max}) + 2L\left(\frac{T}{m_T}\right)^{\alpha}, p^* \right\} \right],$$

for $j \in \mathcal{N}_i^k$. Let $\gamma_{m,i}$ denote the stationary deterministic policy for $i \in \mathcal{P}_k$ given by the solution of the above. Then, based on the state at time t in round m, action $\gamma_{m,i}(\boldsymbol{s}_i(t))$ is taken by machine i and announced to the machines in $\mathcal{N}_i^k$. The following theorem gives an upper bound on the regret of DSO.

Theorem 3: For $p^* n_{\max} < c$, the regret of DSO is upper bounded by $(n_{\max} - c) \sum_{i \in \mathcal{P}_k} e^{-2|\mathcal{N}_i^k|(c/n_{\max} - p^*)} |\mathcal{N}_i^k|$.

[3]If $\sum_{\boldsymbol{n}_i'} \hat{p}_{m,i}(\boldsymbol{n}_i, \boldsymbol{n}_i') \neq 1$, then normalization is done by dividing right hand side of (7) by this sum.

policy	DSO	myopic	myopic cluster
average number of denials	0.328	0.449	0.602

TABLE I
AVERAGE NUMBER OF SERVICE DENIALS FOR DSO, MYOPIC AND MYOPIC CLUSTER POLICIES

Proof: The robust dynamic program in (8) is solved using the uncertainty sets $(\mathcal{D}_{m,i})_{j,0} \subset [1-p^*, 1]$ and $(\mathcal{D}_{m,i})_{j,n_{\max}} \subset [0, p^*]$. The myopic cluster policy is in the set of admissible policies for each $i \in \mathcal{P}_k$. Therefore if it is optimal, it would be chosen given the uncertainty sets. If it is not optimal then a better policy will be chosen by robust dynamic programming. Therefore the regret of the learning algorithm is never worse than the regret of the myopic cluster policy. ■

Although we bound the performance of DSO using the myopic cluster policy, in general DSO's performance can be much better. That is because of the fact that the myopic cluster policy only lets the queue length to reach c, while the queue length of any machine can reach L_λ in the policy calculated by DSO. Intuitively the excess buffer capacity $L_\lambda - c$ can be used to queue up services when large number of new service requests happen, since the probability of large number of requests happening is small because p^* is small. The state space of each cluster i increases exponentially in $|\mathcal{N}_i^k|$, compared to CSO whose state space increases exponentially in K. When the cluster sizes are small, DSO's computational complexity is significantly smaller than CSO's.

V. NUMERICAL RESULTS

In this section we compare the performance of DSO with two different myopic policies that do not learn the transition probabilities in an online way: myopic policy and myopic cluster policy. In the myopic policy, every machine serves the requests in its own region, while in the myopic cluster policy, each cluster serves c times cluster size number of service requests at each time slot, without queueing them.

A. Simulation of Average Number of Service Denials

We run our simulation with final time $T = 5000$, and round length $T_R = 1000$. There are three machines such that only machines 1 and 2, and 2 and 3 are one hop neighbors. Each machine either has 0 or $n_{\max}$ arrivals at each time step, and the (unknown) new service request probabilities $p_i(t) = Pr(n_i(t) = n_{\max})$, $i = 1, 2, 3$ are such that when $p_1(t)$ is increasing $p_3(t)$ is decreasing. In this section, instead of solving the robust dynamic program given in (8), DSO solves the dynamic program $V_{m,i}(\boldsymbol{s}_i) = \max_{\boldsymbol{a}_i \in A_i(\boldsymbol{s}_i)}(-e_i(\boldsymbol{n}_i, \boldsymbol{a}_i) + \sum_{\boldsymbol{s}_i'} V_{m,i}(\boldsymbol{s}_i') \hat{Q}_{m-1,i}(\boldsymbol{s}_i, \boldsymbol{a}_i, \boldsymbol{s}_i') - V_{m,i}(\boldsymbol{s}_i^*))$, for $i = 2$, at the beginning of each round m, based on its estimated transition probabilities in the previous round. Since DSO uses the myopic cluster policy in its first round, we neglect the service denials of all policies in the first round, i.e., first 1000 slots, and take the average of the denials for the remaining slots. In Table I, we give the average number of service denials of myopic policies and DSO. In this setting, DSO provides about 27% and 45% improvement over the myopic and myopic cluster policies, respectively. The reason that myopic cluster policy is worse than the myopic policy is that it does not exploit the queueing ability of the machines.

VI. Related Work

Prior work on mobile service offloading usually considers it as an optimization problem rather than a learning problem. Moreover, the focus is usually on the optimal scheduling of a single machine with heterogeneous service requests instead of a network of machines. In [8], a protocol to decide which mobile services to offload based on the mobile's throughput-delay tradeoffs is proposed as the solution of a single-user static optimization problem. In contrast, we consider offloading to a locally available virtual machine network and assume that the statistics are unknown and time-varying, and provide theoretical and numerical results for both our centralized and distributed algorithms. In [9], a computation offloading algorithm that minimizes mobile energy consumption given a delay constraint is proposed. This is done by dividing the task to be offloaded into several components and deciding which components should be offloaded. In contrast, we look into the problem from a more general perspective, where each task is offloaded or not as a whole, but the question is how the offloading decision affects the future decisions of the neighboring machines. An experimental setup to study the feasibility of computation offloading and data backups in the cloud is given in [10]. In [11], the authors formulate an optimization problem for joint virtual machine placing and routing.

Our problem formulation is related to the multi-armed bandits, learning with expert advice and reinforcement learning in unknown Markov decision processes (MDPs). The goal in these problems is to choose the best possible action given incomplete or noisy information about the state transition dynamics of the system. In [12] and [13], optimal learning policies are derived for an unknown MDP with stochastic but fixed rewards and fixed transition probabilities. It is shown that the average reward of these policies converges to the average reward of the best deterministic policy. An experts learning approach is taken in [14], in which there is an expert associated with each state, and the experts' goal is to choose the best action given the state. In [15] a contextual bandit problem is considered, where the goal is to choose the best arm given the context information. In [6], the authors consider online learning in an unknown MDP in which the rewards and state transition probabilities can change arbitrarily. Convergence to the optimal policy in terms of the average reward is not possible in this case because the uncertainty about the state transition probabilities does not vanish. However, given some restrictions on how the rewards and transition probabilities can change, the authors are able to analytically bound the loss in performance in terms of the average reward. A fast online learning algorithm that exploits a special structure in an unknown MDP is proposed in [16], and a bound on its performance in the limit $T \to \infty$ is given.

Table II provides a comparison of our work with related work in online learning.

y= yes, n=no	[6]	[12], [13], [16]	[15]	our work
time-inhomogeneous	y	n	y	y
Markovian (M)/ iid (i)	M	M	i	M
distributed	n	n	n	y

TABLE II
Comparison with related work in online learning

VII. Conclusion

In this paper we developed an experts learning framework for mobile service offloading. We proposed both centralized and decentralized online learning algorithms and gave analytic regret bounds for both. Importantly, the coupled experts approach we developed in this paper can also be applied to many other deployment scenarios and applications: online routing problems where the link delays change over time, online channel selection algorithms for cognitive radios, etc.

References

[1] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for VM-based cloudlets in mobile computing," *IEEE Pervasive Computing*, vol. 8, no. 4, pp. 14–23, 2009.

[2] H. A. Lagar-Cavilla, N. Tolia, E. De Lara, M. Satyanarayanan, and D. OHallaron, "Interactive resource-intensive applications made easy," in *Middleware 2007*. Springer, 2007, pp. 143–163.

[3] D. P. Palomar and M. Chiang, "A tutorial on decomposition methods for network utility maximization," *IEEE Journal on Selected Areas in Communications (JSAC)*, vol. 24, no. 8, pp. 1439–1451, 2006.

[4] W. Hsu, K. Merchant, H. Shu, C. Hsu, and A. Helmy, "Weighted waypoint mobility model and its impact on ad hoc networks," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 9, no. 1, pp. 59–63, 2005.

[5] W. Hsu, T. Spyropoulos, K. Psounis, and A. Helmy, "Modeling time-variant user mobility in wireless mobile networks," in *Proc. of the 26th IEEE International Conference on Computer Communications (INFOCOM)*. IEEE, 2007, pp. 758–766.

[6] J. Y. Yu and S. Mannor, "Online learning in Markov decision processes with arbitrarily changing rewards and transitions," in *Proc. of the International Conference on Game Theory for Networks (GAMENETS)*, 2009, pp. 314–322.

[7] A. Nilim and L. El Ghaoui, "Robust control of Markov decision processes with uncertain transition matrices," *Operations Research*, vol. 53, no. 5, pp. 780–798, 2005.

[8] Y. Im, C. Joe-Wong, S. Ha, S. Sen, T. Kwon, and M. Chiang, "AMUSE: Empowering users for cost-aware offloading with throughput-delay tradeoffs," in *Proc. of the 32nd IEEE International Conference on Computer Communications (INFOCOM)*, 2013.

[9] S. Barbarossa, S. Sardellitti, and P. D. Lorenzo, "Computation offloading for mobile cloud computing based on wide cross-layer optimization," in *Future Network and Mobile Summit*, 2013.

[10] M. V. Barbera, S. Kosta, A. Mei, and J. Stefa, "To offload or not to offload? The bandwidth and energy costs of mobile cloud computing," in *Proc. of the 32nd IEEE International Conference on Computer Communications (INFOCOM)*, 2013.

[11] W. Jiang, M. Chen, S. Ha, T. Lan, and M. Chiang, "Joint VM placement and routing for data center traffic engineering," in *Proc. of the 31st IEEE International Conference on Computer Communications (INFOCOM)*, March 2012, pp. 758–766.

[12] P. Ortner, "Logarithmic online regret bounds for undiscounted reinforcement learning," in *Proc. of the Conference on Advances in Neural Information Processing Systems (NIPS 2006)*, vol. 19. The MIT Press, 2007, p. 49.

[13] A. Tewari and P. Bartlett, "Optimistic linear programming gives logarithmic regret for irreducible MDPs," *Advances in Neural Information Processing Systems*, vol. 20, pp. 1505–1512, 2008.

[14] E. Even-Dar, S. M. Kakade, and Y. Mansour, "Experts in a Markov decision process," *Advances in Neural Information Processing Systems*, vol. 17, pp. 401–408, 2005.

[15] A. Slivkins, "Contextual bandits with similarity information," *JMLR: Workshop and Conference Proceedings*, vol. 19, pp. 679–701.

[16] F. Fu and M. van der Schaar, "Structure-aware stochastic control for transmission scheduling," *IEEE Transactions on Vehicular Technology*, vol. 61.