A Theoretical Framework for Online Information Search

Rohit Negi *
Electrical and Computer Engineering
Carnegie Mellon University

Abstract

A significant part of human activity today consists of searching for a piece of information online, utilizing knowledge repositories. This endeavor may be time-consuming if the individual searching for the information is unfamiliar with the subject matter of that information. However, experts can aid individuals find relevant information by searching online. This paper describes a theoretical framework to model the dynamic process by which requests for information come to a system of experts, who then answer the requests by searching for those pieces of information.

1 Introduction

The Internet today has been transformed from a network providing connectivity, to a massive repository of human (and machine) knowledge, with information relevant to nearly every aspect of human life stored in some corner. Search engines allow keyword-based search of this knowledge, and while natural language queries are increasingly useful, searching for complex information requires human thinking (augmented with the capabilities of search engines) to obtain useful search results. While there are canonical 'big problems' in different fields that require specialized experts, a large part of human life deals with a vast number of small problems, each affecting a different individual in its own unique manner. These problems require the individual to search the Internet for ideas relevant to solving that problem, an activity that may receive mixed results, depending on the expertise of that individual. But given pervasive online connectivity, there are potentially a large number of 'experts' available online that an individual can consult, who can contribute their knowledge to problems related to their expertise [1].

Given the growing importance of such online requests for information, this paper envisions a large number of requests for information being made, but also a large number of potential experts available to answer those requests. Since the requests must be responded to in a timely manner, we propose a dynamic framework, where requests arrive stochastically, are handled by expert(s) who search for relevant information, and depart when the expert provides a response. Preliminary results on scheduling requests, and on the resulting capacity of the system are presented.

*This work was supported by the National Science Foundation under award 2008570.

2 Theoretical Framework

The problem setting in the paper assumes that requests for information come into a social network stochastically. Each request is handled by an expert (or experts), which searches for information to answer that request, and succeeds in providing information answering that request after a random amount of time, based on the complexity of the search. This requires describing a quantitative model for information search and also describing a model for scheduling these requests, so that experts can answer them.

1. **Model of Information Search**

   Time is assumed to be discretized finely, so that it is measured as $t = 1, 2, 3, \ldots$ time slots. Let $\mathcal{M}$ be a large set of information facts. A *topic* $x \subset \mathcal{M}$ is a large subset of facts - examples being 'Windows 10 debugging' or 'Seventeenth century poetry'. The set of topics $\mathcal{X}$ is assumed to be large but finite to avoid technical clutter. An *expert* is a *research time* function $T : \mathcal{X} \to [1, \infty)$, where $T(x) \geq 1$ is the mean time that expert takes to answer a request concerning topic x; this average time is assumed to be known to the expert. This time is required because the expert will typically need to search for information relevant to the specific request before being able to answer it. We assume that the time to answer a specific request is a geometrically distributed random variable (with mean value $T(x)$). A typical request may be 'Why does my Windows 10 laptop become hot and shut down?', which concerns the topic 'Windows 10 debugging'. For conciseness, we will simply call a request concerning topic x as *request* x.

2. **Model of Dynamic Scheduling**

   It is assumed that there is a social network of n experts, represented as a graph $G = (V, E)$, where the vertices V represent experts and the edges E represented coordination opportunity between pairs of experts. By *coordination*, we mean that a scheduler (described below) can assign a request in expert i's queue to expert j, as long as $(i, j) \in E$ in the graph. For example, if experts exclusively use a Knowledge market (or an Internet Q&A Forum) like `Quora` [2], they can all coordinate with each other, and so G is a complete graph. On the other hand, if a social network like `Twitter` or `Facebook` is used, the graph may have a complex structure, precluding arbitrary coordination. This paper only considers a complete graph linking the experts.

   We adopt a dynamic stochastic model of information searching. In each time slot t, at each expert $i \in V$, each request $x \in \mathcal{X}$ may newly arrive with probability $\lambda p_i(x)$, and so, we need a multi-class queuing model. Denote as $a_{x,i}(t) = 0, 1$ the non-arrival or arrival of request x at expert i, respectively. Its arrival is independent of arrival of requests in other topics, arrivals at other experts, and arrivals in other time slots. Here, $0 < \lambda < 1$ and $p_i(x) > 0$ is a probability mass function (p.m.f.) over topics x (so, $\sum_{x \in \mathcal{X}} p_i(x) = 1$). λ can be interpreted as the *request load* on the network, while $p_i(x)$ causes requests for certain topics to appear more frequently. Due to independence, we allow multiple different topics x to arrive at any expert, and also multiple experts to see requests from the same topic x. Each expert i puts request x into its own *virtual queue* and increases the length $Q_{x,i}(t)$ of that topic's queue by one request (all requests will actually be written in random access

memory, so the virtual queue is a book-keeping artifact). In practice, the requests may be given to the expert by users she knows in her social circle, or may be selected by the expert from a knowledge market like `Quora`.

A scheduler then assigns different requests to different experts, subject to the social network graph, allowing the experts to *coordinate* in handling the requests. Since this paper assumes a complete graph model, the scheduler can assign any request to any expert.

Expert i works on its assigned request x by searching for information (equivalently, called 'researching x'), and answers it successfully in that time slot with probability $q_i(x) \doteq \frac{1}{T_i(x)} \leq 1$. Experts with larger $q_i(x)$ presumably have deeper knowledge that allows them to quickly research problems, and so, a crude measure of expertise of an expert is $R_i = \sum_x q_i(x)$. $d_{x,i}(t) = 0, 1$ indicates failure or success of i finding the answer for x during time slot t, respectively. If the request is not answered successfully, it goes back in its queue. Future scheduling of that request does not utilize the past history of handling that request. Thus, the number of (potentially non-consecutive) time slots needed to answer a request is a geometric random variable with average time $T_i(x)$. Clearly, the queue lengths update as $Q_{x,i}(t+1) = Q_{x,i}(t) + a_{x,i}(t) - d_{x,i}(t)$.

The maximum request load λ that can be researched by this system, while keeping the request queues stable is called *capacity*. Queue stability can be defined either as *stability-in-the-mean* [3], i.e.,

$$\limsup_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \sum_{x} E[Q_{x,i}(t)] < \infty, \qquad \forall i, \tag{1}$$

or as positive recurrence of the queue Markov chain [4]. Given the large number of topics (large $\mathcal{X}$), we may be willing to reject requests that do not match the expertise available to research them, i.e., $e_{x,i}(t) \in \{0, 1\}$, if a new arriving request x at expert i is kept or rejected, respectively, at time t. So, we will also wish to characterize capacity under $\varepsilon-$loss constraint. i.e., the maximum load that a system can handle while keeping queues stable, with losses bounded as below.

$$\limsup_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \sum_{x} E[e_{x,i}(t)] \leq \varepsilon, \qquad \forall i. \tag{2}$$

3 Results

Based on the theoretical framework of information search presented in Section 2, we present preliminary results on the performance of the system.

3.1 Single Expert

Consider a simple setting with only a single expert '1', as shown in Figure 1(a). At discrete time t, requests arrive and are placed in their respective queues. A scheduler assigns a request x from one of the queues to the expert, who searches for information to answer it and succeeds in answering it with probability $q(x)$, which depends on the expert's average search time $T(x)$ for that request.

Lemma 1 *The capacity is* $\lambda^* = \left(\sum_x \frac{p(x)}{q(x)}\right)^{-1}$. *Further, any* $\lambda < \lambda^*$ *can be achieved using any work conserving scheduler (such as one that assigns an arbitrary request in the queue to the expert.)*

Evidently, capacity is high if the expertise of the expert matches closely with the population of requests coming in, so that none of the ratios $\frac{p(x)}{q(x)}$ is too large. In light of this elementary result, we can call $\lambda_j(p) \doteq \left(\sum_x \frac{p(x)}{q_j(x)}\right)^{-1}$ as the capacity of the expert j with respect to p.m.f. $p(x)$.

We can also characterize the capacity under loss constraint (2).

Lemma 2 *If we are willing to accept average loss rate* ε, *the capacity is no less than the* λ^* *specified by the Linear program below.*

$$\lambda^* = \max_{\mu(x)} \left(\sum_x \mu(x)\frac{p(x)}{q(x)}\right)^{-1} \quad \textit{where} \tag{3}$$

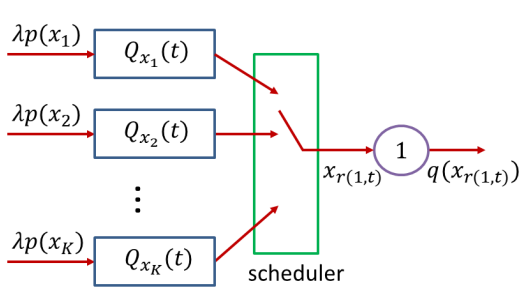
$$\sum_x \mu(x)p(x)\frac{q(x)+\varepsilon}{q(x)} = 1, \tag{4}$$

$$0 \le \mu(x) \le 1, \forall x. \tag{5}$$

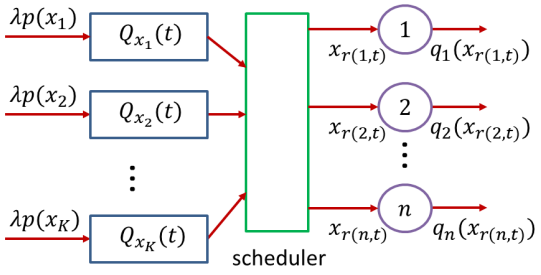
Any $\lambda < \lambda^*$ *can be achieved by an offline scheduler; one that first solves this optimization problem assuming known* $p(x), q(x)$.

The offline scheduler first calculates the probabilities $\mu(x)$ by solving the optimization problem (3) before considering requests. After that, when request $a_x(t)$ comes in, the scheduler drops it (so $e_x(t) = 1$) independently with probability $1 - \mu(x)$. Otherwise, it gets inserted into its topic queue. The optimal solution $\mu(x)$ is intuitive. Define $\nu(x) = \mu(x)\frac{p(x)}{q(x)}(q(x)+\varepsilon)$ as the variables to optimize, instead of $\mu(x)$. Then, the problem (3) is to minimize over $\nu(x)$ the function $\sum_x \frac{\nu(x)}{q(x)+\varepsilon}$ subject to $\sum_x \nu(x) = 1$ and $0 \le \nu(x) \le \frac{p(x)}{q(x)}(q(x)+\varepsilon)$. This minimization can be done as follows. Arrange the topics in the sequence of decreasing $q(x)$, as $q(x_1) \ge q(x_2) \cdots \ge q(x_{|\mathcal{X}|})$. Find the largest m such that $\sum_{i\le m} \frac{p(x)}{q(x)}(q(x)+\varepsilon) \le 1$. All the $x_i, i \le m$ are assigned $\mu(x_i) = 1$ so that these requests are never dropped. Note that these are the requests that have the largest $q(x)$ (i.e., within the expertise of the expert.) Finally, choose $\nu(x_{m+1}) = 1 - \sum_{i\le m}\nu(x_i)$, so that x_{m+1} is occasionally dropped with probability $\nu(x_{m+1})\frac{q(x_{m+1})}{p(x_{m+1})(q(x_{m+1})+\varepsilon)}$. For the remaining requests $i > m+1$, we set $\mu(x_i) = \nu(x_i) = 0$, so that these requests (which are far from the expertise of the expert) are always dropped. Then, the capacity is $\left(\sum_{i\le m}\frac{p(x_i)}{q(x_i)} + \mu(x_{m+1})\frac{p(x_{m+1})}{q(x_{m+1})}\right)^{-1}$.

For $\varepsilon = 0$, the solution to (3) is the same as Lemma 1, because the equality (4) reduces to $\sum_x \mu(x)p(x) = 1$, and so, can only be satisfied by $\mu(x) \equiv 1$ (since $p(x) > 0$ is a p.m.f.) Lemma 2 is especially useful when there is a gross mismatch between the requests and the expert. For example, if the expert has $q(x) = 0$ iff $x \in \mathcal{X}_0$, the lossless capacity is $\lambda^* = 0$. But if we allow loss, we can set $\mu(x) = 0, \forall x \in \mathcal{X}_0$ and $\mu(x) = 1$ otherwise, to achieve a load $\lambda = \left(\sum_{x\notin\mathcal{X}_0}\frac{p(x)}{q(x)}\right)^{-1} > 0$, while accepting a loss of $\varepsilon = \lambda\sum_{x\in\mathcal{X}_0} p(x)$.



(a) Single expert



(b) Multiple coordinating experts in complete graph

Figure 1: Expert scheduling scenarios.

Suppose that the expert has an erroneous estimate $\hat{T}(x)$ of her average searching time $T(x)$. For example, the expert may have an intuitive approximation of these times based on her past experience answering questions about these topics. Since the scheduler uses $\hat{T}(x)$ to schedule while the true search time is $T(x)$, the capacity λ^* calculated in Lemma 1 may be an over-estimation, resulting in queue instability. However, an achievable load can be guaranteed if we assume that the estimation error has a known bound, i.e., if we assume $\hat{T}(x) \geq \gamma T(x), \forall x$, for some constant $\gamma \leq 1$.

Corollary 1 *Let λ^* be the capacity in Lemma 1 calculated using the erroneous search times $\hat{T}(x)$ that have bounded errors. Then, any work conserving scheduler using $\hat{T}(x)$ can achieve any load less than $\lambda < \gamma\lambda^*$ with stable queues.*

3.2 Multiple Coordinating Experts

Now consider n experts on a social network with a complete graph (so that they can all see requests in each others' queues). Since the theoretical framework allows the scheduler to schedule requests from a neighbor's queue, in the complete graph case, we can equivalently assume that the queues of all the experts are merged together for each topic x; i.e., $Q_x(t) = \sum_i Q_{x,i}(t)$. Define $p(x) = \sum_i p_i(x)$ as the merged p.m.f. See Figure 1b. This models experts that each monitor a single knowledge market like `Quora`. In this case, we have the following result.

Lemma 3 *The capacity with multiple coordinating experts is at least*

$$\lambda^* = \left(\max_{(\alpha_i)} \sum_x \min_i \left(\alpha_i \frac{p(x)}{q_i(x)} \right) \right)^{-1} \quad \text{where} \quad \sum_i \alpha_i = 1, \quad \alpha_i \geq 0 \tag{6}$$

Further, any $\lambda < \lambda^*$ can be achieved using an offline scheduler.

The offline scheduler is assumed to know $p(x), q_i(x)$. It maintains separate topic queues $Q_{x,i}(t)$ for each expert i. Before considering requests, it first calculates the solution to the convex dual problem [5] of the maximization problem over α_i stated in (6). (For brevity, we

will simply call this maximization problem as the problem (6).) The dual problem is the Linear program below (see Lemma 5).

$$\min_{\mu,\ s_{i,x}} \quad \mu \quad \text{s.t.} \tag{7}$$

$$\sum_x \frac{p(x)}{q_i(x)} s_{i,x} \leq \mu,\ \forall i \tag{8}$$

$$\sum_i s_{i,x} = 1,\ \forall x, \quad s_{i,x} \geq 0,\ \forall i, x \tag{9}$$

Using these pre-computed $s_{i,x}$ (which we note is a p.m.f. over i for each x), for each arriving request x, the scheduler selects an expert i randomly and independently according to the p.m.f. $s_{i,x}$, and then inserts that request into the topic queue $Q_{x,i}$ of expert i. In each time slot, the scheduler also assigns a request randomly to expert i from among the requests queued up at that expert's queues $Q_{x,i}$. Expert i is kept idle if and only if her own queues are all empty. Thus, the expert is work conserving with respect to her own queues.

As opposed to single expert scheduling, in this case, any one expert mismatched to the request p.m.f. $p(x)$ may not be catastrophic. In fact, the following case shows that a *diversity* of experts may be preferable. Suppose there are n experts, with each expert i having expertise $R_i \doteq \sum_x q_i(x) = 1$. Consider a toy case where $|\mathcal{X}| = n$ and $p(x) = \frac{1}{n}, \forall x$. If the experts are identical, i.e., $q_i(x) = q(x), \forall x$, then the capacity in (6) is maximized for $q(x) = \frac{1}{n}, \forall x$ and it is $\lambda^* = 1$. Instead, suppose we have diverse experts $q_i(x) = 1\,(x = x_i)$ (where $1\,(A) \in \{0, 1\}$ is the indicator function of statement A), each of which also has expertise $R_i = 1$ as in the case of identical experts. Then, the capacity in (6) is increased to $\lambda^* = n$, showing the benefit of diversity.

3.3 Collaborating Experts

Consider one last case that shows the utility of the theoretical framework in this paper. Referring to the same situation as in Section 3.2, illustrated in Figure 1b, suppose that experts $1, 2$ can collaborate. This means that they can decide to jointly research a common request x, jointly arriving at the correct answer after an average time $T_{1,2}(x)$. Thus, the success probability of collaboration in a given time slot is $q_{1,2}(x) = \frac{1}{T_{1,2}(x)}$. If these experts have complementary areas of expertise, it is possible that $q_{1,2}(x) \gg q_1(x) + q_2(x)$, so that collaboration on x is significantly more beneficial than non-collaborative work on separate requests. In fact, we have the following.

Lemma 4 *The capacity with the additional collaboration between* $1, 2$ *compared to Lemma 3 is,*

$$\lambda^* = \left(\max_{(\alpha_i)} \sum_x \min\left((\alpha_1 + \alpha_2) \frac{p(x)}{q_{1,2}(x)},\ \min_i \left(\alpha_i \frac{p(x)}{q_i(x)} \right) \right) \right)^{-1} \quad \textit{where} \tag{10}$$

$$\sum_i \alpha_i = 1, \quad \alpha_i \geq 0$$

As a toy example, consider the case of $n = 2$ coordinating and also collaborating experts. Assume $p(x) = 1, x = x_1$, so that only the case $x = x_1$ is of interest. Assume $q_1(x_1) =$

$q_2(x_1) = \frac{1}{100}$ because each finds the request difficult, while $q_{1,2}(x_1) = 1$ because here the two experts pool their complementary expertise to solve x_1. Then, the solution to (10) is $\lambda^* = (\min(1, 50, 50))^{-1} = 1$. On the other hand, the coordinating, but non-collaborating case (6) yields $\lambda^* = (\min(50, 50))^{-1} = 0.02$, which shows the likely benefits of collaboration.

4 Conclusions

This paper set up a theoretical framework to analyze the dynamic process by which requests for information arrive in a social network, so that either a single expert or a collection of experts can search for the needed information. Preliminary results on queuing and scheduling analysis in this framework were presented. Future work will look at online and distributed schedulers for the scenarios analyzed in this paper.

A Proofs

We will use Lyapunov analysis and invoke the well-known Foster-Lyapunov theorem [4], which we state below for completeness.

Theorem 1 (Foster-Lyapunov theorem) *Suppose a Markov chain $Q(t)$ in a countable state space E is irreducible and suppose there exists a function $L : E \to \mathbb{R}$ bounded below as $L \geq 0$. Suppose also that there is a finite set F and some $\delta > 0$ such that,*

$$E[L(Q(t+1))|Q(t)] < \infty, \quad \forall i \in F, \tag{11}$$
$$E[L(Q(t+1))|Q(t)] < L(Q(t)) - \delta, \quad \forall i \notin F. \tag{12}$$

Then the Markov chain is positive recurrent.

With a slight abuse of notation, the Lyapunov function is often written as $L(t)$.
Proof [Lemma 1]: Let $\mathbf{Q}(t) \doteq [Q_x(t)]$ be the vector of topic queue lengths. Since the request arrivals are independent in the time slots, $\mathbf{Q}(t)$ is a Markov chain. It is irreducible since any state can be reached from any other state by adding arrivals or subtracting departures. Assume that $q(x) > 0, \forall x$, since otherwise $\lambda^* = 0$ and the Lemma is trivially proved . To show that any load $\lambda < \lambda^*$ is achievable using any work conserving scheduler, consider the Lyapunov function $L(t) = \sum_x \frac{1}{q(x)} Q_x(t)$ for $\mathbf{Q}(t)$. Then, $\Delta L(t) \doteq L(t+1) - L(t) = \sum_x \frac{1}{q(x)}(a_x(t) - d_x(t))$. So, $E[\Delta L(t)|\mathbf{Q}(t)] = \sum_x \frac{1}{q(x)} E[a_x(t) - d_x(t)|\mathbf{Q}(t)] = \sum_x \frac{1}{q(x)}(\lambda p(x) - q(x)\sigma_x(t)) = \lambda \sum_x \frac{p(x)}{q(x)} - \sum_x \sigma_x(t)$, where $\sigma_x(t) = 1$ if the scheduler assigns a request from topic x to the expert, else 0. This is because, if an expert works on request x, it has a probability $q(x)$ of successfully answering it in that slot. Let $B = \{\mathbf{Q}(t) : \mathbf{Q}(t) = \mathbf{0}\}$. For any work conserving scheduler, $\sum_x \sigma_x(t) = 1$ if $\mathbf{Q}(t) \notin B$. So, for the case $\mathbf{Q}(t) \notin B$, $E[L(t+1)|\mathbf{Q}(t)] = L(t) + \lambda \sum_x \frac{p(x)}{q(x)} - 1 = L(t) - \delta$, where $\delta \doteq 1 - \lambda \sum_x \frac{p(x)}{q(x)} > 0$ since $\lambda < \lambda^*$. Further, for the case $\mathbf{Q}(t) \in B$, $E[L(t+1)|\mathbf{Q}(t)] \leq \sum_x \frac{1}{q(x)} E[a_x(t)|\mathbf{Q}(t)] = \sum_x \frac{\lambda p(x)}{q(x)} = c < \infty$ since we assumed $q(x) > 0, \forall x$.

Considering both cases, by Foster-Lyapunov theorem, the irreducible Markov chain $\mathbf{Q}(t)$ is positive recurrent, which proves stability. Alternatively, stability-in-the-mean can be directly obtained by telescoping the $E[\Delta L(t)|\mathbf{Q}(t)]$ terms.

$E[L(t)] \leq E[L(0)] + \sum_{\tau=1}^{t} (-\delta 1(\mathbf{Q}(\tau) \neq \mathbf{0}) + c1(\mathbf{Q}(\tau) = \mathbf{0})) \leq \max(E[L(0)], c)$. So, $\limsup_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \sum_x E[Q_x(t)] \leq (\max_x q(x)) \limsup_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} E[L(t)]$ $\leq \max(E[L(0)], c)(\max_x q(x)) < \infty$. Thus, the chosen λ also achieves queue stability-in-the-mean.

For the converse, if $\lambda > \lambda^*$, $E[L(t+1)|\mathbf{Q}(t)] = L(t) + E[\sum_x \frac{1}{q(x)}(a_x(t) - d_x(t))|\mathbf{Q}(t)] = L(t) + \sum_x (\lambda \frac{p(x)}{q(x)} - \sigma_x(t)) \geq L(t) + \lambda \sum_x \frac{p(x)}{q(x)} - 1 = L(t) - \delta$, since the expert can only work on one request in each time slot. However, since $\lambda > \lambda^*$, we now have $\delta < 0$. Telescoping this result, we get $E[L(t)] \geq E[L(0)] - t\delta$. Letting $q_{min} = \min_x q(x) > 0$, we have $\frac{1}{T} \sum_{t=1}^{T} \sum_x E[Q_x(t)] \geq q_{min} \frac{1}{T} \sum_{t=1}^{T} \sum_x \frac{1}{q(x)} E[Q_x(t)] = q_{min} \frac{1}{T} \sum_{t=1}^{T} E[L(t)] \geq q_{min} E[L(0)] - \frac{1}{2} q_{min} \delta (T+1) \to \infty$ as $T \to \infty$. Thus, the queues are not stable-in-the-mean. □

Proof [Lemma 2]: Note that the stated optimization problem can be re-written as

$$\max_{\mu(x)} \ \lambda \quad \text{s.t.} \tag{13}$$

$$\lambda \sum_x \mu(x) \frac{p(x)}{q(x)} \leq 1, \tag{14}$$

$$\lambda \sum_x (1 - \mu(x)) p(x) \leq \varepsilon, \tag{15}$$

$$0 \leq \mu(x) \leq 1, \forall x. \tag{16}$$

This is because λ is maximized when both inequalities (14),(15) are equalities. Probability $\mu(x)$ can be shifted from one inequality to the other until both are equalities. Thus, in the optimal solution, $\sum_x \mu(x) \frac{p(x)}{q(x)} = \sum_x (1 - \mu(x)) \frac{p(x)}{\varepsilon}$. This is equality (4) stated in the Lemma.

The offline scheduler, which drops requests randomly, is equivalent to reducing the expected arrival rate at the queue of x to $\lambda p(x) \mu(x)$. So, by Lemma 1, $\lambda = \left(\sum_x \frac{p(x)\mu(x)}{q(x)} \right)^{-1}$ is indeed achievable with stable queues. For the losses, $E[e_x(t)] = E[e_x(t) a_x(t)] = \lambda p(x)(1 - \mu(x))$. So, $\limsup_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \sum_x E[e_x(t)] = \limsup_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \sum_x \lambda p(x)(1 - \mu(x)) \leq \varepsilon$ due to (15). Thus, loss is within the acceptable bound.

□

Proof [Corollary 1]: Here, $\lambda^* = \left(\sum_x \frac{p(x)}{\hat{q}(x)} \right)^{-1}$, where $\hat{q}(x) = \frac{1}{\hat{T}(x)}$, since the erroneous $\hat{T}(x)$ is used to calculate capacity. Since $\hat{q}(x) \leq \frac{1}{\gamma} q(x)$, $\lambda^* \leq \left(\sum_x \gamma \frac{p(x)}{q(x)} \right)^{-1}$. Thus, if the load satisfies $\lambda < \gamma \lambda^*$, we also get $\lambda < \left(\sum_x \frac{p(x)}{q(x)} \right)^{-1}$, where the right hand side is the true capacity of the system. Therefore, by Lemma 1 (scheduling without errors), such λ is achievable with stable queues.

□

Lemma 5 *The problem (7) is the convex dual of problem (6).*

Proof: The problem (6) can be written as

$$\max_{\alpha_i,\ \beta(x)} \sum_x \beta(x), \qquad \text{s.t.} \tag{17}$$

$$\begin{aligned} \beta(x) &\leq \alpha_i \frac{p(x)}{q_i(x)},\ \forall i, x, \\ \sum_i \alpha_i &= 1, \quad \alpha_i \geq 0 \end{aligned} \tag{18}$$

With $s_{i,x} \geq 0$ being the dual variables for inequalities (18), the Lagrangian is $J = \sum_x \beta(x) - \sum_i \sum_x s_{i,x}(\beta(x) - \alpha_i \frac{p(x)}{q_i(x)}) = \sum_x \beta(x)(1 - \sum_i s_{i,x}) + \sum_i \alpha_i \sum_x \frac{p(x)}{q_i(x)} s_{i,x}$. Maximizing the Lagrangian over $\beta(x)$ shows that it is finite only when the condition $\sum_i s_{i,x} = 1, \forall x$ is imposed. Then, maximizing the Lagrangian over the p.m.f. α_i gives the dual function $\max_{\alpha_i, \beta(x)} J = 0 + \max_i \sum_x \frac{p(x)}{q_i(x)} s_{i,x}$. Thus, the convex dual problem is

$$\min_{s_{i,x}} \ \max_i \ \sum_x \frac{p(x)}{q_i(x)} s_{i,x}, \quad \text{s.t.} \tag{19}$$

$$\sum_i s_{i,x} = 1,\ \forall x, \quad s_{i,x} \geq 0,\ \forall i, x \tag{20}$$

The minimization in (19) can be re-written as $\min_{\mu, s_{i,x}}\ \mu$, where $\mu \geq \sum_x \frac{p(x)}{q_i(x)} s_{i,x}, \forall i$. This gives the dual problem specified in (7).

□

Proof [Lemma 3]: Let $s_{i,x}, \mu^*$ be the optimal solution of the dual problem (7). Recollect that the offline scheduler uses this optimal $s_{i,x}$ to assign requests to experts' individual queues. For expert i, the arrival of request $a_{x,i}(t)$ into its queue $Q_{x,i}$ is independent of arrival of other requests to its own queues or to other experts' queues, and has a rate of $\lambda p(x) s_{i,x}$ with load λ. Since the scheduling of expert i only considers its own queues, its schedule is independent of schedules of other experts. So we can analyze the queue stability of each expert i separately. By Lemma 1, expert i's capacity is $\lambda_i^* = \left(\sum_x \frac{p(x) s_{i,x}}{q_i(x)} \right)^{-1} \geq (\mu^*)^{-1}$ by (8). By strong duality, the solutions of (6) and (7) are the same, i.e., $\mu^* = (\lambda^*)^{-1}$. So, $\lambda_i^* \geq \lambda^*$ and also $\lambda^* > \lambda$ by choice of the load. Thus, the load λ seen by expert i is indeed below its capacity λ_i^*, and so, Lemma 1 guarantees its queue stability.

□

References

[1] J. Powell, "The Rise of the Knowledge Market," Forbes, June 27, 2011.

[2] `https://www.quora.com/`

[3] P.R. Kumar and S.P. Meyn, "Stability of Queueing Networks and Scheduling Policies," IEEE Trans. Automatic Control, vol. 40, pp. 251-260, Feb. 1995.

[4] P. Bremaud, Markov Chains: Gibbs Fields, Monte Carlo Simulation, and Queues, Springer.

[5] S. Boyd and L. Vandenberghe, Convex Optimization, Cambridge.