Resource Allocation for Wireless Streaming *

DDP final report
Atul - EE15B011
Guide: Prof.Avhishek Chatterjee

1 Introduction

Frequent buffering pauses or playout stalls during multimedia streaming is a source of great dissatisfaction among cellular users. As multimedia is a significant part of today's total data consumption, operators must strive to provide a smooth streaming experience. During video or multimedia streaming, data transmitted by the base station (BS) are first cached in the media player buffer at the application layer. From this, the media player consumes (plays) one multimedia frame at a time at a rate dictated by the quality, encoding and dynamics of the content. Whenever the buffer does not have enough data to play the current frame, there is a playout stall or buffering pause. Real time scheduling [2, 3, 4, 5, 6, 7] is a generic framework for studying delay sensitive applications like multimedia streaming, interactive video, real time sensing, and multi-player gaming. Broadly, it concerns with scenarios where data packets become available at the BS at different times and have to be transmitted by a deadline. This framework closely captures interactive video, real time sensing, VoIP, where the data to be transmitted are generated sequentially.Unfortunately, this framework fails to capture some important aspects of video streaming [8, 9, 10, 11, 12].
First, in most cases the contents to be streamed are stored in a server and the connection from the server to the BS has high capacity. Hence, it is reasonable to assume that the BS always has enough data to transmit to the user. Even for live streams, a constant lag of a few hundred milliseconds to a few seconds, which we often experience in live telecasts, is generally tolerable. In this time, sufficient content can accumulate at the BS. Second, media player buffers at the application layer of most smartphones have enough capacity to store media frames to be played in the future. As the BS has the data waiting there, it can transmit that in advance if the channel conditions are favorable. Third, the process of consumption of data by a media player from its buffer is stochastic and time-varying, whose statistics depend on the nature of the content, multimedia encoding, and scene dynamics. In this work, we minimize total user dissatisfaction due to buffering pause in a multichannel cellular network. Our formulation captures buffering pause using queuing models for the media player buffers and user dissatisfaction as a function of the frequency of pause.
Unlike the traditional stochastic network optimization setting [13], this problem leads to cost minimization problems with interesting non-convex structures. Exploiting combinatorial structures inside the apparent continuous non-convex problem, we develop near

*This report is copied from a manuscript under submission. Please see [1] for an extended version.

optimal joint admission control and channel allocation algorithms. We consider both the scenarios where the BS knows and where the BS does not know the statistics of the streams. The latter case is related to multi-armed bandits with non-i.i.d and delayed cost. The algorithms require little to no feedback from the user equipment regarding the buffer states and are easily implementable.

2 System Model and Objective

We consider the time-slotted downlink of a cellular base station (BS) with m channels. The BS is streaming multimedia content to n users over these m wireless fading channels. At time-slot $s \in \{1, 2, \ldots\}$, user $i \in [n]$ can send $h_{i,j}(s)$ bits on channel $j \in [m]$, where for any positive integer v, the set $\{1, 2, \ldots, v\}$ is denoted by $[v]$.

The BS decides the allocation of channels and time-slots to users every $\mathcal{E}$ time-slots, which we refer to as *epochs* and denote by $t \in \{1, 2, \ldots\}$. We define $\mathbf{H}(t)$ to be an $\mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^{\mathcal{E}}$-valued process with elements $\{h_{i,j}(s) : i \in [n], j \in [m], (t-1)\mathcal{E}+1 \leq s \leq t\mathcal{E}\}$. The process $\mathbf{H}(t)$ is stationary and ergodic. The BS is infinitely backlogged, i.e., all of the content to be served to the users is waiting at the BS. At the beginning of epoch t, the BS knows $\mathbf{H}(t)$ and decides the allocation of m channels and the slots $\{(t-1)\mathcal{E}+1, \ldots t\mathcal{E}\}$ to the n users.

Once the content has been served by the BS to a user, it is stored in the user's media player buffer, from which every epoch the media player either reads one *frame* or none. For each user i, the time of consumption of a frame is denoted by the stochastic process $F_i(t) \in \{0, 1\}$. Here $F_i(t) = 1$ means that the media player at user i consumes one frame during epoch t. This process is stationary and ergodic with $\mathbf{E}[F_i(t)] = p_i \in [0, 1]$. Let D_i^f denote the amount of data (in bits) in frame $f \in \{1, 2, \ldots\}$ of the content streamed to user i. For each i, $\{D_i^f : f \geq 1\}$ is a stationary and ergodic process. So, the amount of data required by the media player of user i at epoch t is $F_i(t) D_i^{\sum_{\tau=1}^t F_i(\tau)}$, where $D_i^{\sum_{\tau=1}^t F_i(\tau)} := D_i^f$ for $f = \sum_{\tau=1}^t F_i(\tau)$.

Let $Q_i(t)$ be the occupancy (in bits) of the media player buffer of user i at the end of epoch $t-1$ and the amount of content (in bits) delivered to user i by the BS in epoch t be $S_i(t)$. As the media player consumes either one frame or none at each epoch, the evolution of the buffer at user i is given by

$$Q_i(t+1) = Q_i(t) + S_i(t) - F_i(t) D_i^{\sum_{\tau=1}^t F_i(\tau)} \cdot \mathbf{1}(F_i(t) D_i^{\sum_{\tau=1}^t F_i(\tau)} \leq Q_i(t) + S_i(t)).$$

We say that the media player at user i has *paused* at time t if

$$\mathbf{1}(F_i(t) D_i^{\sum_{\tau=1}^t F_i(\tau)} > Q_i(t) + S_i(t)),$$

i.e., the media player attempted to play the $\sum_{\tau=1}^t F_i(\tau)$th frame, but there was not enough data in the buffer.

We define a resource allocation policy a to be a sequence of maps $\{a^{(t)}\}$ such that at each t, $\{S_i(t) : i \in [n]\} = a^{(t)}(\{Q_i(\tau) : i \in [n]\}, \ \mathbf{H}(\tau) : 1 \leq \tau \leq t)$. Let $\mathcal{A}$ be the class of all ergodic policies under which the time average of the system vector $\{Q_i(t), S_i(t) : i \in [n]\}$ has an almost sure limit in $\mathbb{R}_+ \cup \{\infty\}$. For any $a \in \mathcal{A}$ we define the asymptotic

frequency of pause for user i as

$$\kappa_i^a = \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \mathbf{1}(F_i(t) D_i^{\sum_{\tau=1}^{t} F_i(\tau)} > Q_i^a(t) + S_i^a(t)) \text{ a.s.},$$

where $S_i^a(t)$ and $Q_i^a(t)$ are the service and the buffer processes under policy $a \in \mathcal{A}$.

For each user i there is a cost function $V_i : [0,1] \to \mathbb{R}_+$ which captures the user's dissatisfaction as a function of its frequency of pause. The asymptotic cost for user i under policy $a \in \mathcal{A}$ is given by $V_i(\kappa_i^a)$. Thus, the total asymptotic cost of the n-user and m-channel system under policy a is $V^{n,m}(a) = \sum_i V_i(\kappa_i^a)$, where κ_i^a may possibly depend on the channel statistics.

As our primary objective is to minimize the total user dissatisfaction due to pause, we find an allocation $a \in \mathcal{A}$ which minimizes the total asymptotic average cost:

$$\arg\min_{a\in\mathcal{A}} V^{n,m}(a).$$

2.1 Practically relevant cost function

Standard resource allocation problems in wireless networks involve either a minimization of a convex function or a maximization of a concave function. A traditional choice of cost function along this line would turn the above problem into a convex problem and thus, would offer more tractability. Unfortunately, in this case, such a choice would be impractical. For choosing the right cost functions, let us relate to our own experience during multimedia streaming.

By definition, $0 \le \kappa_i \le p_i$, because frequency of pause cannot be more than the frame rate. To understand the nature of the functions, it is better to first look at the two extremes: $\kappa_i = 0$ and $\kappa_i = p_i$. Naturally, we must have $V_i(0) = 0$ and $V_i(p_i) > 0$ for all i. It is also obvious that the cost functions $\{V_i\}$ must be non-decreasing to capture increased dissatisfaction at an increased frequency of pause. Near $\kappa_i = p_i$, where almost every frame is paused, a slight decrease in κ_i would have almost no impact on user's dissastisfaction, which is at saturation. On the other hand, near $\kappa_i = 0$, where the streaming experience is smooth, a slight increase in the frequency of pause would annoy the user significantly. This implies that a natural choice for $\{V_i\}$ are monotone increasing functions whose derivatives are non-increasing. Thus, the class of monotone increasing *concave* functions is the right choice for cost.

2.2 Assumptions

So far, in describing the system model and the objective, we have made some generic assumptions on the dynamics of the media player buffer and the fading process. For analytical tractability and simplicity of exposition, we introduce some structural assumptions.

A1: For each i, V_i is a non-decreasing differentiable concave function with $V_i(0) = 0$, the derivative at 0 bounded by G and $V_i(p_i) = V \cdot p_i$ for some positive constant V.

A2: For $i \in [n]$ and $j \in [m]$, $h_{i,j}(s)$ are the same for all $s \in [(t-1)\mathcal{E} + 1, t\mathcal{E}]$ and is denoted by $h_{i,j}(t)$. For each i and j, $\{h_{i,j}(t) : t \in \mathbb{Z}_+\}$ are i.i.d. $\{0,1\}$ with $\bar{h}_{i,j} := \mathbf{P}(h_{i,j}(t) = 1) \ge \bar{h}$ for some $\bar{h} > 0$. Also, for each t and i, $\{h_{i,j}(t) : 1 \le j \le m\}$ are i.i.d.

A3: For each i, $F_i(t) \in \{0,1\}$ is stationary and ergodic with $\mathbf{P}(F_i(t)=1) = p_i$, where p_i is of the form $\frac{z_i}{Z}$ for all i. Here Z is an integer independent of the system size and $z_i \in [Z]$ for all i. For some $k \in \mathbb{Z}_+$, $D_i^f = k\mathcal{E}$ for all i and f.

Assumption **A1** is a consequence of the observations made in Sec. 2.1. Assumption **A2** is the standard ON/OFF i.i.d. block fading assumption which is widely used in studying resource allocation in multi-channel wireless networks. Our algorithms and performance guarantees can be extended to fading processes which are Markov across time and channels.

Video is generally encoded as group of pictures (GoP), each composed of I, P, B, and D frames placed in a certain pattern depending on the encoding scheme [14]. When the system is overloaded, the videos are transmitted at the lowest resolution level, and hence, it is reasonable to assume that I frames, also referred to as the key frames, carry most of the data. The assumptions on $\{F_i(t)\}$ and $\{D_i^f\}$ in **A3** are motivated by this fact and analytical tractability.

The GoP structure and the frame rates are encoded in the header of the stream at the application layer. The MAC scheduler at the BS does not have access to these end-to-end application layer parameters. These parameters are generally used by the media player for decoding and playing the stream. But based on certain metadata shared by the higher network layers or the user equipment, the BS may be able to estimate the frame rate and the GoP structure. In terms of the mathematical model in Sec. 2 and the above assumptions, these parameters (statistics) are equivalent to $\{p_i\}$. We study resource allocation in both scenarios: the BS knows and does not know $\{p_i\}$ a priori.

3 Known $\{p_i\}$: non-convexity and joint admission-allocation

It is apparent that the cost minimization problem posed in Sec. 2 is quite different from traditional utility optimization problems in communication networks, which are generally solved via novel adaptations of convex algorithms, e.g., dual gradient descent (a.k.a. drift plus penalty method) [13], heavy ball method [15], alternating direction method of multipliers [15]. The cost minimization problem in Sec. 2 involves minimization of a differentiable concave cost, and hence is a non-convex problem. Moreover, the input variables of the cost functions are not data rates, rather frequencies of pause. It is not clear how to write the resource constraints directly in terms of frequencies of pause so that we can obtain a suitable static problem [13]. As a result, the widely used network optimization techniques cannot be applied here.

We start with the case when $\{p_i\}$ are known at the BS a priori, since it is the simpler case which helps to separate the complexity in cost minimization from the additional challenges due to the lack of knowledge of $\{p_i\}$.

3.1 A benchmark

To analytically compare the performance of our proposed resource allocation policies, a benchmark is needed. The following theorem provides a universal benchmark for all ergodic allocation schemes.

Theorem 1. *Under assumptions* **A1-A3**, *the cost of any ergodic policy is lower bounded by*

$$\bar{V}^{(n,m)} = \min_{\{0\leq\alpha_i\leq 1\}} \sum_{i=1}^{n} V_i(\max(p_i - \alpha_i, 0)) \ s.t. \sum_i \alpha_i \leq \frac{m}{k}. \tag{1}$$

This bound is applicable for any $\bar{h} > 0$ in assumption **A2**, and thus is independent of the fading statistics. Later, we show comparison of the cost under our proposed policy with this lower bound. The above theorem follows from the following lemma.

Lemma 1. *Under assumptions* **A1-A3**, *for any ergodic policy* $a \in \mathcal{A}$, *if the ergodic service rate to user* i *is* $\bar{s}_i^a := \lim_{\tau\to\infty} \frac{1}{\tau}\sum_{t=1}^{\tau} S_i^a(t)$, *then* $\kappa_i^a = \max(p_i - \frac{\bar{s}_i^a}{k\mathcal{E}}, 0)$.

This expression for κ_i^a is obtained by establishing a simple relation between the probability of buffering pause and the expected change in the buffer state at a time t. Please refer to Appendix A for the details. We can see that setting $\frac{\bar{s}_i^a}{k\mathcal{E}} = \alpha_i^*$ achieves the lower bound in Thm. 1, where $\{\alpha_i^*\}$ are the optimal solutions of (1). This bound might be achievable in the absence of fading or when the system is underloaded. However, for an overloaded system, i.e., when $\sum_i p_i > \frac{m}{k}$, especially in the presence of fading, it is not possible to achieve $\frac{\bar{s}_i^a}{k\mathcal{E}} = \alpha_i^*$ for all i simultaneously, since this would otherwise require that $\sum_i \frac{\bar{s}_i^a}{k\mathcal{E}} = \frac{m}{k}$, i.e., the total ergodic service rate should not be impacted by fading at all. Hence, for fading channels, a gap with the benchmark is expected.

3.2 No fading case: an important building block

For designing resource allocation schemes, we first consider channels without fading, i.e., $h_{i,j}(t) = 1$ for all i, j, t followed by channels with fading. This incremental approach helps to separate the issue of non-convex allocation from the uncertainty due to fading, and offers insights which are useful later. As discussed before, the lack of a convex structure does not allow us to use the traditional network optimization techniques [13].

We take an indirect approach which harnesses a combinatorial structure inside the continuous non-convex problem and gives an optimal joint admission control and channel allocation scheme. Our approach is motivated by the following simple observation based on Thm. 1 and Lem. 1. If we can find $\{\alpha_i^*\}$ that solve the optimization problem in Thm. 1 and can also obtain an allocation scheme $\bar{a}$ such that $\alpha_i^* = \frac{\bar{s}_i^{\bar{a}}}{k\mathcal{E}}$, then $\bar{a}$ is an optimum resource allocation scheme.

ConcMin (Alg 1) is proposed to solve the optimization problem in Thm. 1. In the case of an under-loaded (resource rich) network, i.e., $\sum_{i\in[n]} p_i \leq \frac{m}{k}$, $\alpha_i = p_i$ for all i is the obvious optimal solution (Step 1). The main challenge lies in the overloaded or resource constrained network, i.e., $\sum_{i\in[n]} p_i > \frac{m}{k}$. In this case, ConcMin searches over a collection of extreme points of the constraint set and picks one with the minimum cost. Here the extreme points are the set of tuples $\{\alpha_i : i \in [n]\}$ such that for some $\mathcal{S} \subset [n]$ and $|\mathcal{S}| = n - 1$, $\alpha_i \in \{0, p_i\}$ for all $i \in \mathcal{S}$. This search is carried out in Steps 4-24.

To find the best extreme point, for each $k \in [n]$, ConcMin searches for the subset $\mathcal{S}_k^* \subset [n] \setminus k$ and the best $\alpha_k \in (0, p_k)$ so that if $\alpha_i = p_i$ for $i \in \mathcal{S}_k^*$ and $\alpha_i = 0$ for $i \notin \mathcal{S}_k^* \cup \{k\}$, then the cost is minimized (**for** loop in Step 4). Finally, it picks the best k and the corresponding $\mathcal{S}_k^*$ by comparing cost of $\{\mathcal{S}_k^* : k \in [n]\}$ (Steps 23-24).

The search for $\mathcal{S}_k^*$ is a combinatorial subset selection problem. ConcMin finds L_k which maximizes $\sum_{i\in S\setminus k} p_i$ and R_k which minimizes $\sum_{i\in S\setminus k} p_i$ subject to $\sum_{i\in S\setminus k} p_i >$

Algorithm 1 ConcMin

Input: $\{V_i\}, \{p_i\}, c = \frac{m}{k}$
Output: $\{\bar{\alpha}_i\}$

if $\sum_{i\in[n]} p_i \le c$ **then**
  $\bar{\alpha}_i \leftarrow p_i$ for all $i \in [n]$
else
  for all $k \in [n]$ **do**
    $L_k \leftarrow \textsc{SubsetSum}([n] \setminus k, c)$
    $L \leftarrow V \cdot \left(\sum_{i\in[n]\setminus k} p_i - \sum_{i\in L_k} p_i \right) + V_k(p_k + \sum_{i\in L_k} p_i - c)$
    $\{L$ is cost if $\alpha_i = p_i$ for $i \in L_k\}$
    $R_k \leftarrow \textsc{SubsetSum}([n] \setminus k, \sum_{i\in[n]} p_i - c)$
    $R \leftarrow V \cdot \left(\sum_{i\in R_k} p_i \right) + V_k(\sum_{i\in[n]} p_i - \sum_{i\in R_k} p_i - c)$
    $\{R$ is cost if $\alpha_i = 0$ for $i \in R_k\}$
    if $L \le R$ **then**
      $\alpha_i^k \leftarrow p_i$ for all $i \in L_k$
      $\alpha_k^k \leftarrow c - \sum_{i\in L_k} p_i$
      $\alpha_i^k \leftarrow 0$ for all $i \notin L_k \cup \{k\}$
      $J_k \leftarrow L$
    else
      $\alpha_i^k \leftarrow 0$ for all $i \in R_k$
      $\alpha_k^k \leftarrow c - \sum_{i\in[n]\setminus k} p_i + \sum_{i\in R_k} p_i$
      $\alpha_i^k \leftarrow p_i$ for all $i \notin R_k \cup \{k\}$
      $J_k \leftarrow R$
    end if
  end for
  $k^* \leftarrow \arg\min_k J_k$
  $\bar{\alpha}_i \leftarrow \alpha_i^{k^*}$ for all $i \in [n]$
end if

$1-p_k$. The one with lower cost among them is picked as S_k^*. Finding L_k is related to the well known subset sum problem (Step 5) [16]. It turns out that the problem of finding R_k can be written in an alternate form, which is also a subset sum problem with different parameters (Step 8).

We use the SubsetSum routine to solve the subset sum problem. SubsetSum(W, c), for some $W \subseteq [n]$, returns the set $S \subseteq W$ so that $\sum_{i\in S} p_i$ is maximized subject to $\sum_{i\in S} p_i \leq c$. For SubsetSum the standard dynamic programming based algorithm [16] can be used. Though that algorithm does not solve any general subset sum problem in polynomial time, in our case it does. This is because, for our problem, across all instants the sack sizes are at most $Z \cdot \max(m, n)$. Further, as subset sum is a special case of the knapsack problem and the weights $\{p_i\} \subset \{\frac{z}{Z} : z \in [Z]\}$ for $Z = O(1)$, there exists an accurate algorithm with $O(n)$ complexity [16].

We have the following guarantee on the computational complexity and the correctness of ConcMin.

Theorem 2. *In $O(n^2)$ steps* ConcMin *obtains an optimal solution for the optimization problem in Theorem 1, i.e., $\bar{\alpha}_i = \alpha_i^*$ for all $i \in [n]$.*

Interestingly, the optimization problem in Thm. 1 involves continuous variables with no integer or combinatorial constraint. However, the particular non-convex structure of the problem leads to an optimal combinatorial algorithm. (In Sec. **??** we briefly mention some related non-convex problems with other interesting structures.) Details of proof of Thm. 2 which uses this combinatorial structure can be found in Appendix B. The following is a simple but useful observation.

Lemma 2. *The optima of the optimization problem in Thm. 1 lie in the finite set*

$$\left\{\{\alpha_i : i \in [n]\} : |\{i : \alpha_i \in (0, p_i)\}| \leq 1\right\} \subsetneq [0, 1]^n.$$

This result is related to the fact that the minimizer of a concave function over a convex set lies at an extreme point. Lemma 2 can be proved by starting with a feasible solution and constructing another solution with a lower cost which also lies in the above set, using Jensen's inequality iteratively. Lemma 2 formally proves that searching for $\mathcal{S}_k^* \subset [n] \setminus k$, as described above, is a correct approach. Hence, to prove $\bar{\alpha}_i = \alpha_i^*$ for all $i \in [n]$ it is sufficient to prove the following lemma.

Lemma 3. *Steps 5-21 of* ConcMin *finds $\mathcal{S}_k^*$ for a given $k \in [n]$ in $O(n)$ steps.*

Proof builds on the following insight. Observe that at any $\mathcal{S}_k$, that could potentially be $\mathcal{S}_k^*$, cost for users $\{i \in \mathcal{S}_k\}$ is 0 and the cost for users $\{i \notin \mathcal{S}_k \cup \{k\}\}$ is $V \cdot p_i$. As at the optimum, $\sum_i \alpha_i = c$, α_k can be written as a function of $\sum_{i\in\mathcal{S}_k} p_i$, or equivalently, as a function of $\sum_{i\notin\mathcal{S}_k} p_i$, since $\sum_{i\in[n]} p_i$ is a constant. Thus, the total cost can be written as a function of $\sum_{i\in\mathcal{S}_k} p_i$. Further, this function is concave due to the concavity of V_k. So $\mathcal{S}_k = \mathcal{S}_k^*$ either maximizes or minimizes (subject to $\sum_{i\in\mathcal{S}_k} p_i > 1 - p_k$) $\sum_{i\in\mathcal{S}_k} p_i$ over all $\mathcal{S}_k$. This justifies the approach taken in ConcMin for finding L_k and R_k.

Based on ConcMin, we design a simple randomized channel allocation algorithm RandAlloc in Alg. 2. In fact, effectively, RandAlloc simply allocates channels using independent rolls of an n-sided die with biases $\{\alpha_i^*\}$ output by ConcMin. For a fading channel, this procedure does not succeed as one also has to take states of the channels into account in that case. However, as we discuss later, ConcMin is useful in the case of fading channels and also in the scenario where $\{p_i\}$ are not known a priori.

Algorithm 2 RandAlloc

Pre-computation at time 0**:** obtain $\{\alpha_i^*\}$ from ConcMin

1: At any epoch t: initialize $\mathcal{C} = [m]$
2: **while** $\mathcal{C} \neq \emptyset$ **do**
3: Pick $X_i \sim \exp(\alpha_i)$ for all $i \in [n]$ independently
4: $i^* = \arg\min_{i\in[n]} X_i$
5: Allocate $\min(k, |\mathcal{C}|)$ channels for the entire epoch to user i^*
6: $\mathcal{C} \leftarrow \mathcal{C} \setminus \{\text{those } \min(k, |\mathcal{C}|) \text{ channels}\}$
7: **end while**

Note that RandAlloc works as a joint admission control and channel allocation scheme for an overloaded or resource constrained network. Since the solution obtained from ConcMin gives an automatic admission control by blocking the users with $\alpha_i^* = 0$. Moreover, the following result shows that this scheme also minimizes the asymptotic average cost.

Proposition 1. *Under assumptions* **A1-A3** *and* $\bar{h} = 1$ *in assumption* **A2**, RandAlloc *has an asymptotic average cost* $\bar{V}^{n,m}$ *and per epoch computational complexity* $O(mn)$.

This implies that the resource allocation scheme RandAlloc is optimal in the absence of fading. Along with the analytical performance guarantees, RandAlloc and particularly the subroutine ConcMin have interesting implications. First, RandAlloc offers a quality of experience aware admission control in an overloaded network as well as a simple channel allocation procedure. Second, communicating media player buffer states from the application layer of the user equipment back to the MAC layer of the BS is resource consuming. As RandAlloc does not need this feedback and is computationally inexpensive, it can be easily implemented.

3.2.1 Comparison with standard routing schemes

It is not hard to see that the problem considered here is similar to routing jobs in a multi-server system. For $k = 1$ and $\mathcal{E} = 1$, the problem is equivalent to a discrete time routing problem, where in each epoch m jobs arrive and have to be routed to n queues with binary stationary service processes. As we observe above, for the current problem the optimal scheme does not use $\{Q_i(t)\}$. This is in contrast to the well known high performance routing algorithms like join the shortest queue [17], power-of-d [18, 19], batch-filling [20], where $\{Q_i(t)\}$ are useful even when $\{p_i\}$ are the same and known a priori. The reason behind this is the totally different objectives of the traditional routing problems and the problem considered here. Qualitatively, in the traditional setting the goal of the router is to keep $\{Q_i(t)\}$ small, whereas here the goal is to ensure that the buffers are non-empty. The concave cost is another important difference between these two scenarios. Interestingly, as would be apparent in Sec. 4, when $\{p_i\}$ are not known to the BS a priori, even infrequent one bit feedback regarding the buffer states have a strong impact on the performance.

4 Unknown $\{p_i\}$

As discussed in Sec. 2, $\{p_i\}$ are application layer parameters and hence, not always known to the MAC scheduler of the BS a priori. Moreover, two videos with the same quality

(i.e., HD, 4k) can have different $\{p_i\}$ depending on their dynamism, e.g., sports versus news. Hence, even the application layer may not know accurate values of $\{p_i\}$ a priori.

Unlike the case when $\{p_i\}$ are known a priori, the buffer state information, which indirectly captures $\{p_i\}$, can be useful in this context. Consider a simple setting with $n = 2$, $m = 1$, $k = 1$, $\mathcal{E} = 1$ and $\bar{h} = 1$. Let us also assume that we know the ordering between $\{p_i\}$ (let $p_1 > p_2$), but not their values. As ConcMin needs $\{p_i\}$, AllocateChannels cannot be used here. But, it turns out that the following simple scheme achieves optimal asymptotic average cost: allocate the channel to user 1 whenever $Q_1(t) \leq B$ for some $B > 0$, else allocate to user 2. Though this seems promising, questions remain. Does this scheme extend to general n and m? In a multimedia streaming application, sending $\{Q_i(t)\}$ back to the BS every epoch is resource intensive. Can we use simple and infrequent feedback? It is easy to see that the feedback in the above scheme can be simplified: user 1 sends one bit only when $Q_1(t) \leq B$. But, unfortunately, this scheme does not extend to general n and m.

In practice, all multimedia sessions are of finite duration. Hence, it is also important that the allocation scheme performs well not only in terms of the asymptotic average cost, but also in terms of average cost over all reasonable time windows. Under a policy $a \in \mathcal{A}$, let $\kappa_i^a(T)$ be the empirical frequency of pauses over T epochs. Ideally, we should have a policy a with low $\sum_{i\in[n]} V_i(\kappa_i^a)$ and low $\sum_{i\in[n]} V_i(\kappa_i^a(T))$ for all T. More precisely, if $\bar{\mathcal{A}}$ is the class of ergodic policies which minimize asymptotic average cost, ideally, we would like to have the policy $a^* \in \bar{\mathcal{A}}$, if it exists, such that for all sufficiently large T and any $\bar{a} \in \bar{\mathcal{A}}$,

$$\sum_{i\in[n]} \mathbf{E}\left[V_i(\kappa_i^{a^*}(T))\right] \leq \sum_{i\in[n]} \mathbf{E}\left[V_i(\kappa_i^{\bar{a}}(T))\right]. \tag{2}$$

Note that for all sufficiently large T, $\bar{V}^{n,m}$ is still a benchmark for $\sum_{i\in[n]} \mathbf{E}\left[V_i(\kappa_i^a(T))\right]$. Hence, (2) is equivalent to finding $\bar{a} \in \bar{\mathcal{A}}$ for which

$$v(\bar{a}, T) := \sum_{i\in[n]} \mathbf{E}\left[V_i(\kappa_i^{\bar{a}}(T))\right] - \bar{V}^{n,m}$$

is minimum for all sufficiently large T. Clearly, for any $\bar{a} \in \bar{\mathcal{A}}$ as $T \to \infty$, $v(\bar{a}, T) \to 0$. As the above multi-objective problem is intractable, we find a policy for which the rate (with respect to T) at which $v(\bar{a}, T)$ goes to 0 is the maximum.

4.1 Infrequent buffer feedback and bandits

Using Jensen's inequality to move the expectation inside V_i and then using concavity of V_i and assumption **A1**, it follows that

$$v(\bar{a}, T) \leq G \sum_{i\in[n]} \max(\mathbf{E}[\kappa_i^{\bar{a}}(T)] - \kappa_i^{\bar{a}}, 0).$$

Thus, for upper bounding the rate of decay of $v(\bar{a}, T)$ it is sufficient to upper-bound the rate of decay of $\mathbf{E}[\kappa_i^{\bar{a}}(T)] - \kappa_i^{\bar{a}}$ for each i. Let $\psi_i^{\bar{a}}(T)$ denote the number of pauses for user i over T epochs under policy $\bar{a}$. Then, upper-bounding the rate of decay of $\mathbf{E}[\kappa_i^{\bar{a}}(T)] - \kappa_i^{\bar{a}}$ is equivalent to upper-bounding the rate of growth of $\mathbf{E}[\psi_i^{\bar{a}}(T)]$.

It may be tempting to use the following simple approach. At the beginning, the user estimates p_i by observing the evolution of the media player buffer for some time and

reports it to the BS, then the BS uses AllocateChannels. Though this is a possible approach, it is sub-optimal. This is because for the above estimation steps, the buffers of the users need to have enough frames, for which the BS needs to transmit sufficient contents to all the users. However, during this estimation period, transmissions to the users who are not part of the optimal schedule in ConcMin are in a sense wasted, which could have been used to improve experience of the other users. This implies that we need to strike a balance between exploration and exploitation.

This naturally brings us to the setting of multi-armed bandits [21] with non-i.i.d. cost (instead of reward), where a cost of 1 is incurred for a user every time its stream is paused. The cost is non-i.i.d. because the cost depends on the past states of the buffer, even when $\{F_i(t)\}$ are i.i.d. Moreover, for an action taken at time t, the cost may be incurred at a later time. Though there is a similarity in terms of the non-i.i.d. nature of the system, the dynamics and the costs in this problem are different from the queuing bandits studied in [22, 23, 24].

Drawing intuition from the bandit literature [21, 22, 23, 24] and the analysis of ConcMin and AllocateChannels, we develop an algorithm called *i*nfrequent Feedback, estimate, solve, and allocate (*i*Festival), which takes infrequent one bit feedback about the buffer states, estimates $\{p_i\}$ based on that, and allocates using ConcMin and AllocateChannels.

Algorithm 3 *i*Festival

Input: $\{V_i\}$ and $r, w \in \{2, 3, \ldots\}$

Output: Allocation at each epoch

Initial computation: Define phases $\tau = 1, 2, \ldots$ where τth phase consists of epochs $(\tau-1)(w+1)\lceil\frac{nk}{m}\rceil + 1$ to $\tau(w+1)\lceil\frac{nk}{m}\rceil$

1: **while** System is ON **do**
2: **if** for some $q \in \mathbb{Z}_+ \cup \{0\}$, current phase $\tau = r^q$ **then**
3: Between epochs $(\tau-1)(w+1)\lceil\frac{nk}{m}\rceil + 1$ to $(\tau-1)(w+1)\lceil\frac{nk}{m}\rceil + w\lceil\frac{nk}{m}\rceil$: allocate users k channels each in a work conserving round-robin manner (each user is chosen for w epochs and allocated k-channels in each one of them)
4: Between epochs $(\tau-1)(w+1)\lceil\frac{nk}{m}\rceil + w\lceil\frac{nk}{m}\rceil + 1$ to $\tau(w+1)\lceil\frac{nk}{m}\rceil$: each user sends $\{1,0\}^w$ feedback about increment of $\{Q_i(t)\}$ or not, respectively, in the w epochs they are allocated in Step 3
5: For each $i \in [n]$, based on feedback in Step 4 update $\hat{p}_i$ by the total number of 0s received from user i (since $t = 1$) divided by $w \cdot q$
6: Run ConcMin with $\{\hat{p}_i\}$ to obtain $\{\hat{\alpha}_i\}$
7: **else**
8: Run AllocateChannels with the latest $\{\hat{\alpha}_i\}$
9: **end if**
10: **end while**

*i*Festival, described in Alg. 3, divides time into phases of length $(w+1)\lceil\frac{nk}{m}\rceil$ epochs, where $w \in \mathbb{Z}_+$. For $r \in \mathbb{Z}_+$ and $r \geq 2$, at phases $r, r^2, r^3, \ldots$, *i*Festival serves each user in turn over k channels of an entire epoch and the users record the change (increase or same) of their buffer states at the end of that epoch. In each phase this is done w times in a round-robin fashion over the first $w\lceil\frac{nk}{m}\rceil$ epochs of this phase. From the $(w\lceil\frac{nk}{m}\rceil + 1)$th epoch to $(w+1)\lceil\frac{nk}{m}\rceil$th epoch of this phase, the BS collects all the w one bit feedback regarding change of buffer states. Based on this feedback, it estimates

$\{p_i\}$ and runs ConcMin with these estimates. For any $q \in \mathbb{Z}_+$ between phases r^q and r^{q+1}, iFestival runs AllocateChannels with $\{\bar{\alpha}_i\}$ returned by ConcMin run during phase r^q.

As iFestival collects only infrequent feedback ($\frac{w \log T}{T \log r}$ bits per epoch) from the user equipment, it can be implemented in practice for multimedia streaming in cellular networks. Also, feedback from each user is scheduled a priori (at particular epochs in phases $1, r, r^2, \ldots$) and hence, the uplink traffic due to the feedback is well regulated.

For iFestival, we have the following guarantee on the growth of the expected number of pauses with the horizon T.

Theorem 3. *Under assumptions* **A1-A3** *and i.i.d.* $\{F_i(t)\}$*, if* $T \geq r^2(w+1)\lceil \frac{nk}{m} \rceil$*, and* $w > \frac{2 \ln r}{\min_{i,j} |p_i - p_j|}$*, then in the absence of fading, i.e.,* $\mathbf{H}(t) = \mathbf{1}$*,*

$$\mathbf{E}[\psi_i^{i\text{Festival}}(T)] \leq \max(p_i - \alpha_i^*, 0)T + \bar{C} \log T,$$

for all $i \in [n]$*, where* $\bar{C}$ *is independent of* T*. In addition, if* $T = e^{o(m)}$ *and* $n = \Theta(m)$*, then the above bound is also true for any* $\mathbf{H}(t)$ *satisfying* **A2***.*

Proof of this theorem has two main steps. First, we show that after t epochs, the estimations of $\{p_i\}$, which take values in $\{\frac{z}{Z} : z \in [Z]\}$, are exact with probability at least $1 - \frac{1}{t^{1+\beta}}$ for some $\beta > 0$. Second, we show that between r^qth and r^{q+1}th phases, the expected number of pauses is upper-bounded by $\max(p_i - \alpha_i^*, 0) r^q (r-1) + O(1)$ if the estimate at the end of the r^qth phase is accurate. Combining these two along with some standard probability computations the result follows. Proving the first part is a standard application of Azuma-Hoeffding inequality. The second part requires bounding the expected number of returns to state 0 by the Markov chain $Q_i(t)$ over a finite time window. This Markov chain is positive or null recurrent depending on the value of α_i^*. In the null recurrent case this is obtained by bounding the evolution of the probability of state 0 with time, starting from state 0 itself.

The following result is a consequence of Thm. 3 and the discussions on $v(\bar{a}, T)$ in the beginning of Sec. 4.1.

Proposition 2. *Under assumptions* **A1-A3***, i.i.d.* $\{F_i(t)\}$ *and* $\mathbf{H}(t) = \mathbf{1}$*, if* $T \geq r^2(w+1)\lceil \frac{nk}{m} \rceil$*, and* $w > \frac{2 \ln r}{\min_{i,j} |p_i - p_j|}$

$$v(i\text{Festival}, T) = O\left(\frac{\log T}{T}\right).$$

In addition, if $T = e^{o(m)}$ *and* $n = \Theta(m)$*, then the above bound is also true for any* $\mathbf{H}(t)$ *satisfying* **A2***.*

It can be argued that there are systems for which it is impossible to have a faster decay in an order sense, and thus, implying the (order) optimality of the decay rate under iFestival. The intuition is that the current problem is more involved than regret minimization in i.i.d. multi-armed bandits and hence, a $\log T$ lower bound on the regret is unavoidable.

References

[1] "Resource allocation for smooth streaming: Non-convexity and bandits (extended version)," https://www.dropbox.com/s/isor7b8d7zx2z00/SmoothMultiLong.pdf?dl=0.

[2] V. S. Borkar, *Stochastic Approximation: A Dynamical Systems Viewpoint*. Cambridge University Press, 2008.

[3] I.-H. Hou, P. Kumar *et al.*, "Utility-optimal scheduling in time-varying wireless networks with delay constraints," in *Proceedings of the eleventh ACM international symposium on Mobile ad hoc networking and computing*. ACM, 2010, pp. 31–40.

[4] J. J. Jaramillo and R. Srikant, "Optimal scheduling for fair resource allocation in ad hoc networks with elastic and inelastic traffic," in *2010 Proceedings IEEE INFOCOM*. IEEE, 2010, pp. 1–9.

[5] R. Li, A. Eryilmaz, and B. Li, "Throughput-optimal wireless scheduling with regulated inter-service times," in *2013 Proceedings IEEE INFOCOM*. IEEE, 2013, pp. 2616–2624.

[6] K. S. Kim, C.-p. Li, and E. Modiano, "Scheduling multicast traffic with deadlines in wireless networks," in *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*. IEEE, 2014, pp. 2193–2201.

[7] I. Hou and R. Singh, "Scheduling of access points for multiple live video streams," in *Proceedings of the fourteenth ACM international symposium on Mobile ad hoc networking and computing*. ACM, 2013, pp. 267–270.

[8] P. Dutta, A. Seetharam, V. Arya, M. Chetlur, S. Kalyanaraman, and J. Kurose, "On managing quality of experience of multiple video streams in wireless networks," in *2012 Proceedings IEEE INFOCOM*. IEEE, 2012, pp. 1242–1250.

[9] I.-H. Hou and P.-C. Hsieh, "Qoe-optimal scheduling for on-demand video streams over unreliable wireless networks," in *Proceedings of the 16th ACM International Symposium on Mobile Ad Hoc Networking and Computing*. ACM, 2015, pp. 207–216.

[10] R. Bhatia, T. Lakshman, A. Netravali, and K. Sabnani, "Improving mobile video streaming with link aware scheduling and client caches," in *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*. IEEE, 2014, pp. 100–108.

[11] R. Singh and P. R. Kumar, "Optimizing quality of experience of dynamic video streaming over fading wireless networks," in *2015 54th IEEE Conference on Decision and Control (CDC)*, Dec 2015, pp. 7195–7200.

[12] R. Singh and P. R. Kumar, "Optimal decentralized dynamic policies for video streaming over wireless channels," 2019.

[13] M. J. Neely, *Stochastic Network Optimization with Application to Communication and Queueing Systems*. Morgan & Claypool, 2010.

[14] M.-T. Sun, *Compressed video over networks*. CRC Press, 2000.

[15] J. Liu, A. Eryilmaz, N. B. Shroff, and E. S. Bentley, "Heavy-ball: A new approach to tame delay and convergence in wireless network optimization," in *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*. IEEE, 2016, pp. 1–9.

[16] H. Kellerer, U. Pferschy, and D. Pisinger, *Knapsack Problems*. Springer, 2004.

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

[18] N. D. Vvedenskaya, R. L. Dobrushin, and F. I. Karpelevich, "Queueing system with selection of the shortest of two queues: An asymptotic approach," *Problemy Peredachi Informatsii*, vol. 32, no. 1, pp. 20–34, 1996.

[19] M. Mitzenmacher, "The power of two choices in randomized load balancing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 12, no. 10, pp. 1094–1104, 2001.

[20] L. Ying, R. Srikant, and X. Kang, "The power of slightly more than one sample in randomized load balancing," in *34th IEEE Annual Conference on Computer Communications and Networks, IEEE INFOCOM 2015*. Institute of Electrical and Electronics Engineers Inc., 2015, pp. 1131–1139.

[21] S. Bubeck and N. Cesa-Bianchi, "Regret analysis of stochastic and nonstochastic multi-armed bandit problems," *Found. Trends Mach. Learn.*, vol. 5, no. 1, pp. 1–122, Dec. 2012.

[22] S. Krishnasamy, R. Sen, R. Johari, and S. Shakkottai, "Regret of queueing bandits," in *Advances in Neural Information Processing Systems*, 2016, pp. 1669–1677.

[23] S. Cayci and A. Eryilmaz, "Learning for serving deadline-constrained traffic in multi-channel wireless networks," in *2017 15th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*. IEEE, 2017, pp. 1–8.

[24] S. Krishnasamy, A. Arapostathis, R. Johari, and S. Shakkottai, "On learning the cμ rule: Single and multi-server settings," *Available at SSRN 3123545*, 2018.

A Proofs of Lemma 1 and Theorem 1

Let $X_i(t) = \frac{Q_i(t)}{k\mathcal{E}}$, where $Q_i(t)$ is the buffer evolution process defined in Sec. 2. Using assumption **A3**, we can write the buffer evolution compactly as

$$X_i(t+1) = \left(X_i(t) + \frac{S_i(t)}{k\mathcal{E}} - F_i(t) \right)^+, \tag{3}$$

where $(\cdot)^+$ denotes $\max(\cdot, 0)$. Since we schedule in units of $k\mathcal{E}$, we have $\frac{S_i(t)}{k\mathcal{E}} \in \{0, 1, \ldots\}$.

Following the discussion in Sec. 2, using assumptions **A1-A3**, we can express the frequency of pause as

$$\kappa_i = \mathbf{E}\left[\mathbf{1}\left(X_i(t) + \frac{S_i(t)}{k\mathcal{E}} - F_i(t) < 0 \right) \right].$$

Using the buffer evolution in Eq. (3), this can equivalently be written as

$$\kappa_i = \mathbf{E}\left[\mathbf{1}\left(X_i(t+1) - \left(X_i(t) + \frac{S_i(t)}{k\mathcal{E}} - F_i(t) \right) > 0 \right) \right]. \tag{4}$$

This is because whenever $X_i(t) + \frac{S_i(t)}{k\mathcal{E}} - F_i(t) \geq 0$, $X_i(t+1)$ would be equal to this expression and the argument of the indicator in the above equation for κ_i would be 0. The only way for it to be positive is when $X_i(t) + \frac{S_i(t)}{k\mathcal{E}} - F_i(t) < 0$.

Further, observe that since $X_i(t) \in \{0, 1, 2, \ldots\}$, $\frac{S_i(t)}{k\mathcal{E}} \in \{0, 1, 2, \ldots\}$, and $F_i(t) \in \{0, 1\}$, we have

$$X_i(t+1) - \left(X_i(t) + \frac{S_i(t)}{k\mathcal{E}} - F_i(t) \right) \in \{0, 1\}.$$

This implies that the indicator in Eq. (4) is redundant as its argument is always 0 or 1. So we get

$$\kappa_i = \mathbf{E}\left[X_i(t+1) - \left(X_i(t) + \frac{S_i(t)}{k\mathcal{E}} - F_i(t) \right) \right].$$

When the buffer evolution is positive recurrent, i.e., $\mathbf{E}[F_i(t)] > \mathbf{E}\left[\frac{S_i(t)}{k\mathcal{E}}\right]$, the processes are all stationary, and we have $\mathbf{E}[X_i(t+1)] = \mathbf{E}[X_i(t)]$, and this gives us

$$\kappa_i = \mathbf{E}[F_i(t)] - \mathbf{E}\left[\frac{S_i(t)}{k\mathcal{E}} \right].$$

Using the definition of $\bar{s}_i$ in Lem. 1, and the definition of p_i in assumption **A3**, we get

$$\kappa_i = p_i - \frac{\bar{s}_i}{k\mathcal{E}}$$

which concludes our proof for the positive recurrent case.

When the buffer evolution is null recurrent or transient, i.e., when $p_i < \frac{\bar{s}_i}{k\mathcal{E}}$, we can show that the frequency of pause is 0 using drift arguments.

A.1 Proof of Theorem 1

Let $S_i^*(t)$ be the service under an optimal policy a^*, and let the buffer evolution under such a policy be $Q_i^*(t)$ for each user i. At any epoch, we have a total of $m\mathcal{E}$ slots that can be scheduled, and this means

$$\sum_{i \in [n]} S_i^*(t) \leq m\mathcal{E}$$

for every epoch t.

Since this hold for every epoch, the time average must satisfy this inequality as well, giving us

$$\sum_{i \in [n]} \bar{s}_i^* \leq m\mathcal{E},$$

where $\bar{s}_i^*$ are the ergodic service rates under an optimal policy. This implies that

$$\sum_{i \in [n]} \frac{\bar{s}_i^*}{k\mathcal{E}} \leq \frac{m}{k}. \tag{5}$$

Using Lem. 1, we get

$$V^{n,m}(a^*) = \sum_{i\in[n]} V_i\left(\max\left(p_i - \frac{\bar{s}_i^*}{m\mathcal{E}}, 0\right)\right).$$

Since the optimal policy must satisfy Eq. (5), the solution of the program in Thm. 1 (Eq. (1)), can only have a lower value. This gives us

$$V^{n,m}(a^*) \geq \bar{V}^{n,m}.$$

B Proof of Theorem 2

First we shall prove that ConcMin indeed finds the optimal service rates $\{\alpha_i^*\}$. The optimization problem we are trying to solve can be written as:

$$\begin{aligned} \underset{\{\alpha_i\}}{\text{minimize}} \quad & \sum_i V_i(p_i - \alpha_i) \\ \text{subject to} \quad & \sum_i \alpha_i \leq c \qquad (6) \\ \text{and} \quad & 0 \leq \alpha_i \leq p_i \quad \forall\ i \in [n]. \qquad (7) \end{aligned}$$

Recall that $c = \frac{m}{k}$. Since $\{V_i\}$ are all concave functions, the optimal solution happens at a corner point of the region defined by constraints (6) and (7). We have a total of $2n+1$ linear inequations defining the feasible region (1 in constraint (6) and $2n$ in constraint (7)). Since there are n optimization variables $\{\alpha_i\}$, at every corner point, n of the inequations will hold with equality. However, α_i can't be equal to both 0 and p_i, and so at most n of the inequalities in constraint (7) can hold with equality. As we just have one other constraint in (6), we need at least $n-1$ of the constraints to hold with equality in constraint (7). Therefore, in the optimal solution to the optimization problem, there is at most one user who gets a non-zero rate but is not fully satisfied. This is essentially a proof of Lem. 2.

Let $\mathcal{P} = \sum_i p_i$. When $\mathcal{P} \leq c$, the optimal solution is trivial and we get $\alpha_i^* = p_i$ for all i. This case is handled in line 1 of ConcMin. Now consider the case $\mathcal{P} > c$. Let k^* be such that for all $i \neq k^*$, either $\alpha_i^* = 0$ or $\alpha_i^* = p_i$ in the optimal solution $\{\alpha_i^*\}$. The preceding arguments guarantee that there is at least one such k^*. We find this k^* by looping over all of [n] in line 4 of ConcMin. For each $k \in [n]$, we find the optimal solution $\{\alpha_i^k\}$ that satisfies, for all $i \neq k$, $\alpha_i^k = 0$ or $\alpha_i^k = p_i$. Then we take the best among these over all values of k.

When $\mathcal{P} > c$, given a fixed k, define $S_k, Q_k \subseteq [n] \setminus k$ so that the "optimal" solution $\{\alpha_i^k\}$ satisfies the following properties:

$$\begin{gathered} \alpha_i^k = 0 \quad \forall\ i \in S_k \\ \alpha_i^k = p_i \quad \forall\ i \in Q_k \\ S_k \cap Q_k = \phi \quad \text{and} \quad S_k \cup Q_k \cup \{k\} = [n] \end{gathered}$$

Since $V_i(p_i) = V \cdot p_i$, S_k (or equivalently Q_k) can be found by solving

$$\begin{aligned} \underset{S_k}{\text{minimize}} \quad & V \cdot \sum_{i \in S_k} p_i + V_k \left(\mathcal{P} - c - \sum_{i \in S_k} p_i \right) \\ \text{subject to} \quad & \mathcal{P} - c - p_k \leq \sum_{i \in S_k} p_i \leq \mathcal{P} - c. \end{aligned}$$

The objective is a concave function of $\sum_{i \in S_k} p_i$ and so the minimum objective occurs at the maximum or minimum feasible value of $\sum_{i \in S_k} p_i$. We find $\max_{S_k} \sum_{i \in S_k} p_i$ by solving SubsetSum$([n] \setminus k, \mathcal{P} - c)(= R_k)$ on line 8 of ConcMin. $\min_{S_k} \sum_{i \in S_k} p_i$ subject to $\sum_{i \in S_k} p_i \geq \mathcal{P} - c - p_k$ is the same as solving $\max_{Q_k} \sum_{i \in Q_k} p_i$ subject to $\sum_{i \in Q_k} p_i \leq c$. This we do by SubsetSum$([n] \setminus k, c)(= L_k)$ on line 5 of ConcMin. We then compare the costs of L_k and R_k to get the solution $\{\alpha_i^k\}$ and the correspoinding cost J_k.

Observe that the optimal solution $\{\alpha_i^*\}$ satisfies $\sum_i \alpha_i^* = c$ when $\mathcal{P} > c$. Also, we have $\alpha_i^* = \alpha_i^{k^*}$ for some $k^* \in [n]$. Since we are comparing amongst feasible solutions $\{\alpha_i^k\}$ in line 23 of ConcMin, we get the optimal k^* and hence the optimal solution $\{\alpha_i^*\}$. This shows that ConcMin outputs the optimal solution.

See Sec. 3.2 for a discussion on the computational complexity of ConcMin.