Mobile Network Resource Optimization under Imperfect Prediction

Nicola Bui[1][2]
[1]IMDEA Networks Institute, Leganes (Madrid), Spain
[2]UC3M, Leganes (Madrid), Spain

***Abstract*—A highly interesting trend in mobile network optimization is to exploit knowledge of future network capacity to allow mobile terminals to prefetch data when signal quality is high and to refrain from communication when signal quality is low. While this approach offers remarkable benefits, it relies on the availability of a reliable forecast of system conditions. This paper focuses on the reliability of simple prediction techniques and their impact on resource allocation algorithms. In addition, we propose a resource allocation technique that is robust to prediction uncertainties. The algorithm combines autoregressive filtering and statistical models for short, medium, and long term forecasting. We validate our approach by means of an extensive simulation campaign for different network scenarios. We show that our solution performs close to an omniscient optimizer as well as the simple solution that always maintains a full buffer in terms of prefetching data before it is needed, while at the same time using 20% less network resources than the simple full buffer strategy.**

I. Introduction

"Every form of behavior is compatible with determinism. One dynamic system might fall into a basin of attraction and wind up at a fixed point, whereas another exhibits chaotic behavior indefinitely, but both are completely deterministic" [1]. In his provocative essay, Ted Chiang is suggesting that unpredictability is just a consequence of the limitedness of human comprehension. While we do not assume that mobile networks are deterministic, in this paper we take resource optimization in mobile networks one step further by exploiting the predictability of future network capacity.

Mobile networks are increasingly constrained by limited spectral resources, while at the same time user traffic demands are growing steadily [2]. Researchers are addressing this challenge from a variety of perspectives including massive multiple-input multiple-output communications, heterogeneous networks combining femto, micro and macro cells, device-to-device communication, and the concept of exploiting knowledge about user behavior and the network itself for performance optimization.

Recent studies [3] highlight how network dynamics [4] can be understood, predicted and linked to human mobility patterns [5]. This ability to predict user and network behavior allows to optimize resource allocation [6], [7]. As such, it is a highly interesting approach to increase the efficiency of mobile networks and deal with future traffic growth.

In this paper, we propose a resource allocation algorithm for mobile networks that leverages link quality prediction and prediction reliability. We first design an optimal resource allocation algorithm that assumes perfect knowledge of the future, as well as a general user capacity forecasting model for mobile networks. We then combine the two to obtain an iterative algorithm achieving near-optimal performance by taking into account the forecast reliability. While some prior work dealt with prediction-based optimization and uncertainty [8]–[10], our approach is the first to combine multi-scale forecasting with a resource allocation scheme that accounts for imperfect throughput prediction in mobile networks.

The rest of the paper is structured as follows: Section II provides a brief summary of the related work. Section III describes the system model and assumptions. In Section IV we detail the omniscient resource allocation algorithm. Section V analyzes future prediction feasibility and its limits: Section V-A gives details about the filtering technique used to obtain short term predictions and Section V-B provides the statistical tools for medium to long term forecasting. Section VI provides our solution for resource allocation under imperfect predictions and the performance of this algorithm is analyzed in Section VII. Section VIII provides our final considerations and next objectives.

II. Related work

Several recent papers, for example [6] and [7], optimize mobile video delivery by exploiting future knowledge in order to save both energy and cost. The main idea is that it is better to communicate when the signal quality is good and refrain from doing so when the signal quality is bad: better signal quality results in higher spectral efficiency and less resources have to be used to send the same amount of data.

This paper starts from a more general formulation of the problem. We do not limit our approach to video delivery but address data exchange in general. Also, we relax the assumption of perfect knowledge of future system conditions, taking into consideration prediction techniques and their reliability. (Note that in this paper we do not address content prediction and we assume it is known in advance what content the user will be interested in [11].)

Network capacity prediction has been studied, for example, in [12], [13] and [14]. These papers evaluate ARMA and GARCH filtering techniques that account for sequences which random variables have the same (homoscedastic) or different (heteroscedastic) finite variance respectively. Other papers [15], [16] investigate mobility prediction using either

Markovian estimators or trajectory-based forecasting techniques.

A further key aspect of our paper is the statistic description of the per user capacity in mobile networks. To this end, we modified the models proposed in [4] and [17] to account for imprecise information.

Finally, a number of papers deal with system optimization under imperfect state prediction: [8] studies throughput maximization under imperfect channel knowledge, [10] investigates the impact of imperfect load prediction in cloud computing, and [9] studies resource allocation under uncertainties. However, to the best of our knowledge, our paper is the first attempt combining actual prediction techniques, statistical models, and resource allocation into a practical algorithm for mobile network optimization.

III. System model

In this paper we address the downlink from a base station of a mobile network (eNodeB) to a single receiver (UE). To simplify the description of the problem, we consider slotted time with slot duration t and thus the quantities discussed in the paper are discrete time series. We use i, j, and k to refer to slot indices. The quantities of interest are:

• Position $P = \{p_i \in [0, P_{\max}],\ i \in \mathbb{N}\}$, where p_i is the distance between UE and eNodeB and $P_{\max}$ is the coverage range.

• Active users $N = \{n_i, i \in \mathbb{N}\}$, where n_i is the number of active users that are in the same cell as the UE. It reflects the congestion level of the cell.

• Signal to interference plus noise ratio (SINR) $S = \{s_i \in \mathbb{R},\ i \in \mathbb{N}\}$, where s_i is obtained from p_i as follows:

$$s_i = s_0 p_i^{-\alpha} f_F. \tag{1}$$

Here, s_0 is a system constant, α is the path loss exponent and f_F is a random multiplicative term to account for fast fading.

• User cell capacity $C = \{c_i \in [0, C_{\max}],\ i \in \mathbb{N}\}$, where c_i represents the average capacity obtained by the user during slot i. $C_{\max}$ is the maximum capacity allocable to the UE, given the specific mobile technology. We compute c_i as a function of s_i and n_i through

$$c_i = c_0 g_c(s_i, n_i), \tag{2}$$

where c_0 is a system constant and g_c is a technology dependent function which models system level variables such scheduling policy, congestion, spectral efficiency, etc. In the rest of the paper we consider LTE as the mobile network technology and we adopt the model in [17], which provides a closed form expression for f_F and g_c for a user at a given distance from the base station, when another $n-1$ users are uniformly distributed in the cell area and proportionally fair scheduling is used.

• Receive rate $R = \{r_i \in [0, c_i],\ i \in \mathbb{N}\}$: this is the rate at which the base station sends data to the UE in slot i.

• Download requirement $D = \{d_i \in [0, D_{\max}], i \in \mathbb{N}\}$, where $D_{\max}$ is the maximum data consumption rate of the most communication intensive application. In slot i, the user consumes d_i bytes of data if they are available. If at any time the user receives more data than required, the excess can be stored in a buffer for later use.

• Buffer state $B = \{b_i \in [0, B_M],\ i \in \mathbb{N}\}$, where b_i is the buffer level and B_M is the buffer size in bytes.

• Buffer under-run time $U = \{u_i \in [0, 1],\ i \in \mathbb{N}\}$ is the fraction of slot i for which no data was available to satisfy the download requirements.

The aforementioned quantities are linked as follows:

$$b_{i+1} = \min\{\max\{b_i + r_i - d_i, 0\}, B_M\} \tag{3}$$

$$u_i = \begin{cases} \max\{d_i - r_i - b_i, 0\}/d_i & d_i > 0 \\ 0 & d_i = 0 \end{cases}. \tag{4}$$

The buffer fills (up to the full buffer B_M) whenever the download rate is higher than the consumption rate, $r_i > d_i$. In case $r_i < d_i$, the algorithm empties the buffer and accumulates buffer under-run time whenever $b_i + r_i < d_i$.

In what follows, we refer to function $y = g_y(x)$ as g_y. Similarly, we refer to the probability density function and the cumulative density function (CDF) of a random variable X as $f_X(x)$ and $F_X(x) = \int_{-\infty}^{x} f_X(y)\mathrm{d}y$ and with μ_X and σ_X to its mean and standard deviation.

IV. Resource allocation optimization with perfect forecast

The resource allocation problem aims at finding the optimal rate time series R that satisfies the download requirements D by using the available capacity C in the best way. We define the following objective function:

$$O = \{o_i = r_i/c_i \in [0, 1],\ i \in \mathbb{N}\}, \tag{5}$$

where o_i is the fraction of the available capacity used in slot i and represents a cost. Note that the same rate r has a different cost $o_i > o_j$ if the available capacity $c_i < c_j$. We obtain the following optimization problem:

$$\begin{aligned} \underset{R}{\text{minimize}} \quad & \sum_i o_i \\ \text{subject to:} \quad & \sum_i u_i = \sum_i u_i^*, \\ & b_i \le B_M,\ \forall i \in \mathbb{N}, \end{aligned} \tag{6}$$

where $\sum_i u_i^*$ is the minimum feasible buffer under-run time. To minimize this cost function, the base station should send more data when the available capacity is high and use just the minimum rate required to avoid a buffer under-run when the capacity is low.

The solution of Eq. 6 is the optimal resource allocation strategy R^* that achieves the minimum buffer under-run time $\sum_i u_i^*$ at the lowest cost $\sum_i o_i^*$. If the sequence C is known a priori, various offline algorithms can be used to determine the optimal resource allocation. We use a simple water-filling algorithm, which is able to achieve optimality using the following rules: *i*) define the break-point e_l as the last slot for which all previous rates are finalized (i.e., no more rate can be used in slots up to e_l) if either $b_{e_l} = B_M$ or $r_k = c_k, \forall e_{l-1} < k \le e_l$; *ii*) define an optimization window $[e_l + 1, m]$, where e_l is the last break-point slot and the rate

allocated in all slots in $e_{l-1} < k \leq e_l$ is finalized; *iii*) starting from $l = 0$, $e_l = 0$ and $m = 1$ the algorithm accounts for the slots in the set $\{e_l+1, \ldots, e_l+m\}$ to satisfy the requirements up to slot $e_l + m$; the algorithm chooses a slot if it has the highest capacity among the unused ones in the set. *iv*) the algorithm either increments l, updates e_l and resets $m = 0$ if a break-point is found or increments m. The complete water-filling algorithm is given in Algorithm 1. $\mathrm{sAdd}(X, x)$ adds the element x to the sorted list X in the correct position, $\pi(c_i)$ gives the position in C of the element c_i and $\mathrm{shift}(D, u_j, j)$ is a shift function that recomputes the requirements sequence D accounting for a buffer under-run event u_j in slot j. The following conditions are used:

- $\mathrm{I}_1 := \exists\ e_l < j \leq e_l + m \mid b_j = B_M$ to verify whether a full buffer state is reached,
- $\mathrm{I}_2 := \sum_{j=e_l+1}^{e_l+m} c_j - r_j = 0$ to verify whether all of the available capacity is used, and
- $\mathrm{I}_3 := \sum_{j=e_l+1}^{e_l+m} r_j - d_j = 0$ to verify whether all of the download requirements have been satisfied.

In the following we prove the optimality of Algorithm 1 and discuss the behavior of the algorithm when knowledge of the future capacity is not perfect.

Theorem 1 (Water-Filling Optimality): If R is a solution of Algorithm 1 with C and D as inputs and achieves a buffer under-run time $\sum_i u_i$ and cost $\sum_i o_i$, then there exists no other allocation strategy $R' \neq R$ for C and D that obtains performance $\sum_i u'_i$ and $\sum_i o'_i$, for which $(\sum_i u'_i < \sum_i u_i) \vee (\sum_i u' = \sum_i u_i \wedge \sum_i < \sum_i o_i)$, i.e., it has either a lower buffer under-run time or the same buffer under-run time and a lower cost.

Proof: Theorem 1 can be proven by contradiction on the following hypotheses:

1) $\sum_i u'_i < \sum_i u_i$
2) if $\sum_i u'_i = \sum_i u_i \Rightarrow \sum_i o'_i < \sum_i o_i$

For 1) R cannot satisfy the requirements D in all the slots, thus $\sum_i u'_i < \sum_i u_i \Rightarrow \exists\ j$ s.t. $r_j + b_j < r'_j + b'_j < d_j$. Since $R' \neq R$, they must differ before or on slot j in order to cause the larger under-run time, because any variation later than that cannot decrease $\sum_i u'_i$. Since R is obtained using Algorithm 1 and must result in $u_j > 0$, then for all the slots belonging to the analysis window $[e_{l-1} + 1, e_l]$, where $e_l = j$ the whole available capacity must have been used, which means R' cannot use more capacity there to avoid the buffer under-run. R' cannot use more capacity before slot e_{l-1} either, since that would impact in a window already completed (ended because of condition I_1). Thus, $\sum_i u'_i \geq \sum_i u_i$ if the two strategies are different, which contradicts the first hypothesis.

For 2) it is $(R \neq R') \wedge (\sum_i u_i = \sum_i u'_i) \wedge (\sum_i o_i < \sum_i o'_i)$, thus the two strategies must differ in at least two slots j, k, where $c_j > c_k$ and $(r_j < r'_j) \wedge (r_k > r'_k)$. The two slots j, k cannot belong to the same window, because Algorithm 1 uses the slots from a sorted list and finishes either with a full buffer or when the whole capacity has been used. The two slots j, k cannot belong to different windows either, because if $j < k$, it would have been possible to use more capacity earlier in the allocation which is not possible due to the stopping conditions of the algorithms, whereas if $j > k$, a cheaper slot later in the sequence could have been used instead of a more expensive one earlier in the sequence. However, this is not possible due to either the fact that the more expensive slot must have been used in order not increase $\sum_i u_i$ (stopping condition I_2) or because of the ordered selection of the slots (stopping conditions I_1 or I_3). Thus, $\sum_i o'_i \geq \sum_i o_i$ if the two strategies are different, which contradicts the second hypothesis.

Thus, assuming that an allocation strategy R' provides a better solution than that obtained using Algorithm 1 violates the hypotheses of the theorem, which is therefore proved. ■

Algorithm 1 will be later used in Section VI in an iterative procedure to compute resource allocation when the knowledge of future capacity is inaccurate. The following Section V provides a general forecast solution and its reliability.

Algorithm 1 Water-Filling Algorithm (WF)

Input: the knowledge of the future capacity availability C, the future download requirements D and the initial buffer level B_0.
Output: $R = \mathrm{WF}(C, D, B_0)$

```
l = 0, e_l = 0 // set the starting point
b_{e_l} = B_0 // set the starting buffer
r_{e_l} = 0, R = ∅ // set the starting allocation
while |R| < |D| do
  m = 1 // set the initial window size
  S = ∅ // sorted capacity vector initialization
  while ¬I_1 ∧ ¬I_2 ∧ |R| + m < |D| do
    S = sAdd(S, c_{e_l+m}) // add an element to the sorted capacity list
    i = 1
    while i ≤ m ∧ ¬I_3 do
      r_{π(s_i),old} = r_{π(s_i)} // store previous allocation
      r_{π(s_i)} = min{r_{π(s_i)} + d_{e_l+m}, c_{h_i}, B_M − b_{π(s_i)}} // update the temporary allocation
      b_{π(s_i)+j} = b_{π(s_i)+j} + r_{π(s_i)} − r_{π(s_i),old}, ∀ 1 ≤ j ≤ m − π(s_i)
      i = i + 1
    end while
    m = m + 1 // update the window size
  end while
  if I_1 then
    l = l + 1 // increment break-point index
    e_l = j // set last break-point
  else
    l = l + 1 // increment break-point index
    e_l = e_{l−1} + m // set last break-point
    if I_2 then
      u_j = max{d_j − r_j − b_j, 0}/d_j
      D = shift(D, u_j, j) // shift D proportionally to u_j starting from slot j
    end if
  end if
  R = {R, r_{e_{l−1}}, ..., r_{e_l}} // update the allocation
end while
return R
```

V. General forecast model

In this section we propose a general model describing the forecasting reliability of a system. In particular, we split our model in three time periods based on the prediction horizon:

The **short term** period considers the near future and predicts capacity through time-series filtering techniques [12], [13]. It is characterized by the reliability time τ_p, which defines how many slots of the sequence can be predicted and that we will discuss in Section V-A.

The **medium term** period describes the evolution of the system in terms of available capacity statistics. During this period one or more network cells can be accounted according to the mobility predictor: Markovian predictors [15] can usually compute the likelihood of visiting a given cell, while trajectory-based predictors [16] provide a more accurate estimate by computing the actual distribution of the user position along time.

The **long term** period provides an overall statistical evaluation of the available capacity availability based on the steady state distribution of the user position in the network. Both the medium and the long term periods are discussed in Section V-B. Section VI will then provide our resource optimization algorithm.

A. Short term forecast with filters

This section addresses the reliability time τ_p achievable by filtering techniques applied to available capacity time series. In particular, we study autoregressive-moving average (ARMA) filters and their setup according to the system dynamics defined by the slot time t and the user speed v. We opted for ARMA instead of GARCH [12], since capacity elements belonging to the short term period are characterized by the same finitte variance.

For each $(t \in [0.5, 5], v \in [0.5, 5])$ tuple we consider a set of 100 capacity traces computed using Eqns. (1,2) as per [17] starting from the mobility paths of a user moving at constant speed in a random network deployment. We apply the Box-Jenkins [18] method to determine the type and the order of the filter to be used with each sequence. Through the analysis of autocorrelation and partial autocorrelation plots, we find that the best technique for our sequences consists of simple autoregressive (AR) filters of order τ_F, and that τ_F is inversely proportional to the tv product.

Subsequently, for each of the sequences we estimate filter coefficients by means of the linear least squares procedure [19] and we use the obtained filter to forecast the values of the other sequences with the same (t, v) parameters. We refer to a forecast sequence as $\tilde{C} = \{\tilde{c}_i \in [0, C_{\max}],\ i \in \mathbb{N}\}$, obtained from C and to the corresponding error $\Delta = \{\delta_i = \tilde{c}_i - c_i \in [-C_{\max}, C_{\max}],\ i \in \mathbb{N}\}$.

We consider a prediction to be reliable as long as the error Δ is statistically smaller than estimating the capacity from its distribution $f_C(c)$ or, in other words, when the standard deviations of the two processes are equal $\sigma_\Delta = \sigma_C$.

Thus, we compute μ_Δ and σ_Δ as the average and the standard deviation of all the error sequences with the same (t, v) parameters. Fig. 1 shows on the abscissa the prediction time index normalized on t and on the ordinate σ_Δ/σ_C the standard deviation of the prediction normalized on the standard deviation σ_C of the original series C.

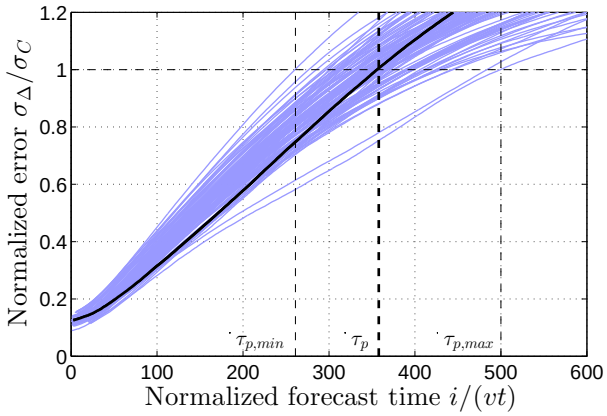

Fig. 1. The shaded area represents how the standard deviation of the short term prediction error increase with increasing prediction distance varying the user speed v and the slot time t. τ_p represents the time after which $\sigma_\Delta \geq \sigma_C$.

While the actual steepness of the curves varies with the parameters, for all of them the normalized error standard deviation σ_Δ/σ_C approaches 1 almost linearly. Hence, we set $\tau_p = \text{argmin}_i \text{s.t.} \sigma_{\Delta_i}/\sigma_C > 1$. In addition, we observe that both τ_p and the filter order can be approximated with simple linear models with the inverse of the tv product and that τ_p is usually 10 times as large as the order of the AR filter.

Finally, it is sufficient to tune a set filters for varying t and v and select the one to use according to the actual user mobility. Also, since filters can be normalized on σ_C it is not needed to have different filters for different numbers of active users in the cell, but it is sufficient to rescale the constant and the variance parameters of the filter.

B. Statistical models and uncertainties

This section describes the second technique of our general forecast model, which adopts statistical models to describe the user capacity availability for medium and long term prediction. In particular, in order to describe the distribution of per user capacity we started again from the model proposed in [17], since to the best of our knowledge it is the only one which takes into account the scheduler impact and thus is able to model user contentions.

In order to account for the impact of uncertainties on the user position and/or the number of active users in the cell we need to modify the expression of the capacity distribution $f_C(x)$ obtained for a specific position p_i and number of users n_i to the actual distribution of the user position $f_P(x)$ and the probability mass function $f_N(n)$ of the number of active users in the cell. This can be achieved through the following equation:

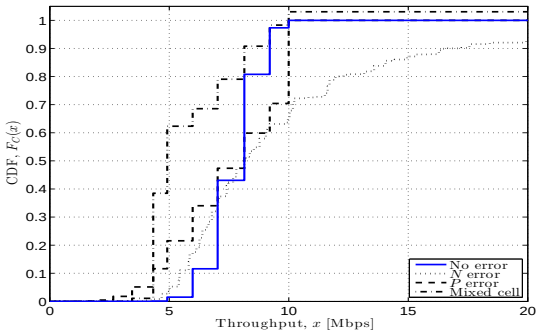

Fig. 2. A few examples of the impact of imperfect knowledge on the capacity distribution: the solid line is obtained with accurate information; the dotted and the dashed with imprecision on N and P respectively, while the dash-dotted considers two cells with different N.

$$f_C(x) = \sum_{i \in \mathcal{N}} f_N(i) \int_0^\infty f_{F,P|i}(g_C^{-1}(x,p),p|i) \left| \frac{\partial g_C^{-1}(x,p)}{\partial x} \right| \mathrm{d}p, \tag{7}$$

where $f_{F,P}$ is the joint distribution of fading and position, g_C is the function linking the per user capacity to p and n and $\mathcal{N}$ is the support of $f_N(n)$. Since fading and user position are statistically independent their joint distribution $f_{F,P}(x,y) = f_F(x)f_P(y)$ is the product of their distributions. Eq. (7) modifies the original capacity distribution weighting it through the active user probability mass function $f_N(i)$ and the user position probability $f_P(y)$; the partial derivative normalizes the integrand.

For what concerns our analysis, it is sufficient to be able to compute the per user capacity distribution by accounting for limited knowledge of the user position and traffic in the cell by means of their respective distributions.

So far, our model describes capacity only for the case when the cell the user is connected to is known perfectly. However, to account for different cells it is sufficient to consider the weighted sum of the capacity distributions of single cells $f_{C,i}(x)$ is the capacity distribution related to cell i and ρ_i is the probability of visiting cell i in the next time period, we obtain:

$$f_C(x) = \sum_{i \in \mathcal{C}} \rho_i f_{C,i}(x), \tag{8}$$

where $\mathcal{C}$ is the set of cells that can be visited in the next time period with some probability.

Fig. 2 provides a few examples of the CDF obtained using the model. The solid line is representative of the capacity CDF $F_C(x)$ when both the active user number $n = 5$ and the user position $p = 500$ meters are exactly known so that the distribution is equal to the fading distribution. The dotted line accounts for an error on the number of active users in the cell so that $f_N(x) = \{0.2, 0.6, 0.2\}$ for $x = \{4, 5, 6\}$ respectively. Conversely the dashed line is obtained by accounting for an error on the user position which has a normal distribution with parameters $\mu_P = 500$ meters and $\sigma_P = 100$ meters. Finally, the dash-dotted line is obtained by mixing together two cells with 5 and 10 users with 20 and 80% of visiting probability respectively. The piecewise-constant shape of the curves is due to discrete relationship between SINR and bitrates.

In a practical implementation of this solution, the capacity statistical models should be known in advance, while the user position statistic $f_P(x)$ and the cell traversal time $\tau^{(i)}$ will be obtained by analyzing user mobility patterns; the number of active users statistic $f_N(y)$ in the cell will be estimated from historic information about that cell at that time of the day.

Finally, to define the statistical model for the long term period, we consider a capacity distribution obtained as a mixture of all the capacity distributions of cells in the network weighted through their steady state visiting probability.

VI. Resource allocation optimization under uncertainties

The objective of this section is leveraging the concepts of the previous ones to design a network resource allocation algorithm which takes into account imperfect forecast and that we called Imperfect Capacity prediction-Aware Resource Optimization (ICARO). ICARO aims at minimizing the communication cost while avoiding buffer under-runs. In particular, we exploit the water-filling algorithm of Section IV in an iterative way. At each iteration, Algorithm 1 makes a single decisions about which rate r to use by exploiting both the AR predictor described in Section V-A and the statistical models designed in Section V-B.

Using an iterative algorithm allows to ensures that the optimization algorithm is only making decision about actual capacity values, but taking into account the future evolution of the sequence.

Before describing the new algorithm, we first have to combine the aforementioned tools into a single general capacity prediction which can be used with Algorithm 1. In order to account for the three time periods described in Section V we proceed as follows (see Fig. 3):

1) The short term prediction $\tilde{c}_i^{(F)}$ with $i \in [0, \tau_p]$ is obtained from the past capacity information collected, for example, by means of lightweight measurements [20] and choosing the filter order τ_F and coefficients based on the user speed v.

2) The medium term model $f_{C,i}(x)$ is computed as the superposition of the cells $j \in \mathcal{C}$ that the user is likely to visit in the i-th time period, each of them accounted for according to their user position $f_{P,j}(y)$ [16] and active user number $f_{N,j}(z)$ [3], [21] statistics by Eq. (8). Similarly, the duration of the i-th time period $\tau_i - \tau_{i-1}$, is obtained as a weighted sum of cells traversal time $\tau^{(j)}$ related to cell $j \in (C)$.

3) During the i-th time period $D_i = \sum_{j=\tau_{i-1}}^{\tau_i} d_j$ bytes has to be downloaded to avoid buffer under-run. The maximum cell efficiency is achieved when only the slots with the highest capacity are used.

4) The highest threshold $c_{T,i}$ is computed so that the average amount of data obtained by selecting only the slots with larger a capacity than c_T is larger than $D_i/(\tau_i - \tau_{i-1})$:

$$c_{T,i} = \max_y \ \text{s.t.} \int_y^\infty x f_{C,i}(x) \mathrm{d}x \geq D_i/(\tau_i - \tau_{i-1}). \tag{9}$$

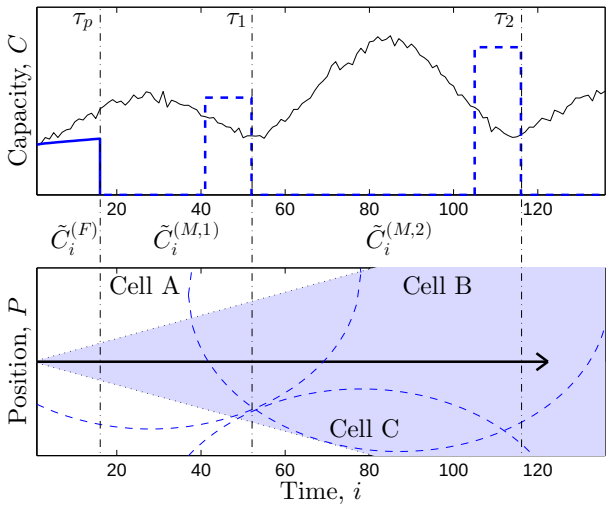

Fig. 3. The upper part of the figure illustrates a sequence obtained through our mixed predictor: autoregressive filtering until τ_p, statistical model until τ_2. The lower part shows a possible example of the user movement, where the horizontal arrow is the user direction and the shaded area contains the set of positions where the user can be predicted to be.

5) The i-th time period is modeled as a sequence of $\tau_i - \tau_{i-1}$ values

$$\tilde{c}_j^{(M,i)} = \begin{cases} c_{T,i} & j > (1 - F_{C,i}(c_{T,i}))(\tau_i - \tau_{i-1}) \\ 0 & \text{otherwise} \end{cases}, \quad (10)$$

where $F_{C,i}(c_{T,i})$ is the probability of the capacity being lower than $c_{T,i}$, thus $(1-F_{C,i}(c_{T,i}))(\tau_i-\tau_{i-1})$ is the average number of slots with larger capacity than the threshold.

6) Steps 2 to 5 are repeated and new time periods are added in the sequence if their reliability is sufficient (two cells, if Markovian predictors are used [15]).

7) Compute τ_o as the offset time when the user first entered in the cell.

8) Obtain the predicted capacity sequence as the concatenation of the previously computed time period sequences:

$$\tilde{c}_i = \begin{cases} c_0 & i = 0 \\ \tilde{c}_i^{(F)} & 0 < i \leq \tau_p \\ \tilde{c}_i^{(M,1)} & \tau_p < i \leq \tau_1 \wedge \tau_1 > \tau_p + \tau_o \\ \tilde{c}_i^{(M,2)} & \max(\tau_o + \tau_p, \tau_1) < i \leq \tau_2 \\ \cdots & \\ \tilde{c}_i^{(M,n)} & \tau_{n-1} < i \leq \tau_n \end{cases}, \quad (11)$$

where τ_n is the duration of the whole sequence, c_0 is the known present capacity, $\tilde{c}_i^{(M,1)}$ is modified by removing slots from the beginning ($\tilde{c}_i^{(M,1)} = 0$) if the past capacity values and those predicted with $\tilde{c}_i^{(F)}$ are lower than $c_{T,1}$ or from the end ($\tilde{c}_i^{(M,1)} = c_{T,1}$) if the opposite is true.

Fig. 3 shows an example of a mixed model sequence: in the upper part, the thin solid line represents the ground truth available capacity C, the thick solid line is the short term prediction $\tilde{C}^{(F)}$ and the dashed line represents two cells through their statistics by means of $\tilde{C}^{(M,1)}$ and $\tilde{C}^{(M,2)}$ respectively. The lower part of the figure represents a map where the user is moving from the left to the right following the central horizontal solid line. The shaded area highlights the area where the user is likely to be The dashed circles represent the coverage areas of different cells. Finally, dash-dotted lines crossing the two parts of the figures mark τ_p, τ_1 and τ_2 instants.

ICARO, which is detailed in the following Algorithm 2, uses the water-filling algorithm to allocate rate iteratively based on the mixed forecast sequence of Eq. (11). The algorithm ends if the total remaining required bytes is smaller or equal than the current buffer level.

Algorithm 2 Imperfect Capacity prediction-Aware Resource Optimization (ICARO)

Input: the future download requirement D, user speed v and position p, τ_F past values of the capacity sampled with t period, the capacity statistics $f_{C,i}(x)$ and time period traversal time τ_i for the next predictable time periods.
Output: R, O, U

```
s = 0 // set the starting point
b_s = B_0 // set the starting buffer
r_s = 0, R = ∅ // set the starting allocation
while Σ_{i=s}^{|D|} d_i ≥ b_s do
    compute C̃ as per Eq. (11)
    run R̂ = WF(C̃, D, b_s)
    r_s = min(r̂_1, c_s, B_M − b_s) // rate to be used
    if c_s then
        o_s = r_s / c_s // cost
    else
        o_s = 0
    end if
    if d_s > 0 then
        u_s = max((d_s − b_s + r_s), 0)/d_s // buffer under-run
    else
        u_s = 0
    end if
    b_{s+1} = min(max(b_s + r_s − d_s, 0), B_M) // next buffer
    if u_s > 0 then
        D = shift(D, u_s, s)
    end if
    s = s + 1
    D = {d_i, s < i ≤ |D|} // remove the first element from the requirements sequence
end while
return R, O, U
```

The rationale for using the water-filling algorithm on the mixed forecast sequence is that its operational principle, that selects which slot to use in descending order, still works under uncertainties and provides a solution which is conservative (as the highest capacity slots are placed last) to avoid under-runs, and aggressive (as the allocation priority is given to the most reliable slots) to optimize allocation costs. In the following, we provide a few examples of the algorithm:

Ordering the short term forecast: the elements of the short term prediction sequence can be assumed to have the same

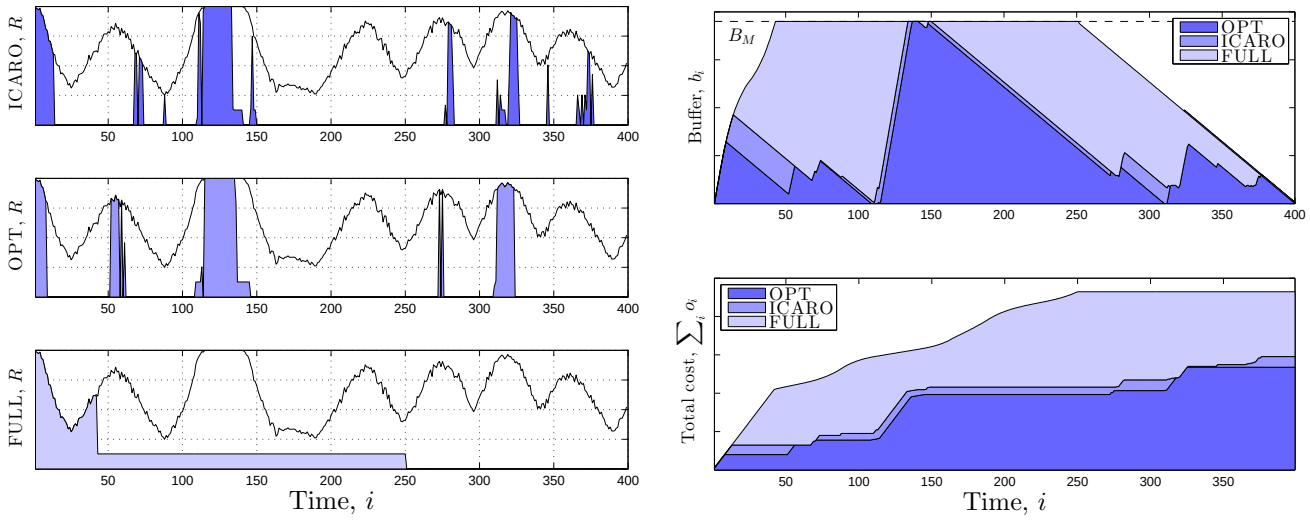

Fig. 4. ICARO algorithm output is compared to the optimal allocation (OPT) and the most conservative approach (FULL) on the left. On the right the differences between the buffer state and the cost evolutions of the three algorithms.

order of those of the actual sequence. In fact, as we showed in Section V-A, $\sigma_{\delta,i}$ is increasing with i, thus if $\tilde{c}_i^{(F)} > \tilde{c}_j^{(F)}$ and $j > i$, then the same ordering probability is larger than that of opposite order ($\mathrm{P}[c_i > c_j] > \mathrm{P}[c_i \leq c_j]$). Thus, the water-filling algorithm can be used on the short term prediction, because its order is likely to match that of the actual sequence.

Ordering between short and medium term forecast: the i-th medium term period is represented as a sequence of $(\tau_i - \tau_{i-1})F_{C,i}(c_{T,i})$ zero capacity slots while the remaining slots are equal to $c_{T,i}$. Hence, if the short term prediction is lower than $c_{T,1}$ only the minimum rate is used, since from the statistical model slots of higher capacity are expected to come later. Conversely, if the short term prediction is larger than $c_{T,1}$, then it is more likely that the remaining slots will be lower than the threshold (compare with step 8 of the sequence creation).

Buffering: the algorithm will always try to use the slots above threshold in each time periods and bridge those by using the buffer. By positioning the slots with highest capacity at the end of each time periods we ensure that the algorithm is conservative. Finally, the maximum buffer size B_M limits the optimization horizon of the algorithm: in fact, the maximum time that the system can last without using any capacity is given by $B_M/(\sum_i d_i/|D|)$. Hence, the buffer size has a significant impact on the algorithm performance which we analyze in the next section.

Fig. 4 shows an example of ICARO's performance compared to the optimal boundary (OPT) obtained with perfect forecast and to the trivial (FULL) solution which maintains the buffer as full as possible. Fig. 4(a) shows three plots of the used rate R of the three algorithm: ICARO in the topmost part, the OPT in the center and FULL at the bottom. In all three plots the shaded area represents the used part of the total available capacity which is drawn as a solid line.

The main difference among the three solutions is that FULL continues to fill the buffer during the low quality period at about $i = 25$, while OPT just use the needed quantity to be able to harness the best part of the second cell ($i = 50$), while ICARO, being more conservative than the optimal solution, accumulates more in the beginning and needs to make some suboptimal decisions (i.e.: $i = 80$). In the rest of the trace, FULL continues downloading just the needed to maintain the buffer full, OPT is able to use the best slots only, while ICARO performs very close to OPT.

Similar considerations can be derived from Fig. 4(a): the upper part shows the buffer variation for the three schemes, while the lower part reports the cumulative cost. We can remark that ICARO performs very close to OPT, but it is always slightly more conservative as the buffer is always a bit fuller earlier on. Also, even though the cost is higher than the optimal, the two algorithms perform quite the same.

VII. Results

In this section we provide an analysis of the overall performance of our algorithm. Since, to the best of our knowledge no other solution is able to compute resource allocation while accounting for the impact of prediction uncertainties, we compare our solution with the optimum offline allocation (OPT) computed with the optimal water-filling algorithm on the exact capacity time series and the most conservative approach (FULL) which just fills up the buffer as soon as possible and maintains it as full as possible until the download requirements are satisfied.

The main performance metrics we are interested in are the objective function O and the buffer under-run time U. In order to be able to mix the results of every tested configuration, we adopt the average cost $\xi = \sum_i o_i/|O|$, the average cost saving $\eta = (\sum_i o_{i,\mathrm{FULL}} - o_{i,\mathrm{ICARO}})/\sum_i o_{i,\mathrm{FULL}}$ obtained by our algorithm, and the average buffer under-run time increase $\zeta = \sum_i u_{i,\mathrm{ICARO}} - \sum_i u_{i,\mathrm{OPT}}$.

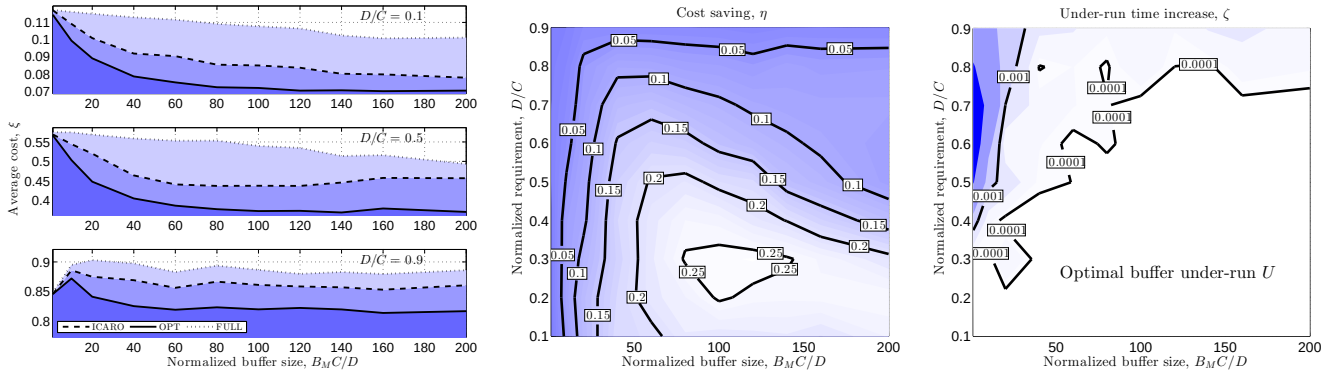

Fig. 5. Overall performance comparison among ICARO, OPT and FULL algorithms in the suburban scenario. ξ, η and ζ for different operational conditions ($B_M \sum_i c_i / \sum_i d_i$ and $\sum_i d_i / \sum_i c_i$) are plotted on the left, center and right respectively.

We analyze two LTE network scenarios: a suburban environment with users moving at moderate vehicular speed and a pedestrian urban environment ($\mu_v = 5$ and 1 meters per second). Both scenarios have been simulated by generating random networks of 100 adjacent cells with average distance between base stations of 500 meters and 1.5 kilometers respectively. Results obtained in the two scenarios are shown in Fig. 5 and Fig. 6.

To generate capacity traces we use the Hata model [22] for the path loss, the Rayleigh distribution for the fast fading and we follow the analysis of proportional fair scheduling in [17] to obtain the capacity distribution of a UE at a given distance from the eNodeB, when there are N active users uniformly distributed in the cell.

Each simulation group is defined by a network deployment and a reference path which the user follows to cross the network. The time it takes to traverse the path is $2000t/v$ seconds. To train the system we study 50 other paths following random trajectories within a cell coverage range from the reference path. We use trajectory based predictors in suburban simulation and Markovian estimators for the urban ones. Subsequently, we validate the system on 25 other paths generated from the same reference while using the information gathered from the training set for the predictions.

For each tuple ($v \in [0.5, 5], t \in [0.5, 5]$) we generate 20 groups of simulations. Finally, during the validation we vary the requirement over capacity ratio $(\sum_i d_i / \sum_i c_i) \in [0.1, 0.9]$, and the normalized buffer size $(B_M \sum_i c_i / \sum_i d_i) \in [1, 200]$.

Fig. 5 (left) shows the average cost ξ of the three algorithms (OPT, ICARO and FULL as solid, dashed and dash dotted lines, respectively) varying the buffer size (x-axis) for $\sum_i d_i / \sum_i c_i = \{0.1, 0.5, 0.9\}$ (upper, center and lower plots). In the upper plot the download requirements are moderate and both OPT and ICARO are able to obtain a normalized cost lower than 0.08 (corresponding to 80% of the $\sum_i d_i / \sum_i c_i$), while FULL needs 95% of the resource ($\xi = 0.095$). The performance is coherent in the other plots and ICARO is always better than FULL and close to OPT. As expected, ICARO performance improves when the buffer is larger and the requirements are lower. Notably, when the buffer is very small the three algorithms perform the same. However, the performance degradation for large buffer size has to be ascribed to the simulation length: in fact, in order to fully exploit a large buffer a proportionally longer time is needed.

The central figure shows contour plots of ICARO's efficiency η using $B_M \sum_i c_i / \sum_i d_i$ as abscissa and $\sum_i d_i / \sum_i c_i$ as ordinate: the curves are labeled according to the cost savings achieved and the area is colored with a darker shade if the saving is lower. Again the best results are obtained for medium buffer and small requirements where ICARO is more than 25% cheaper than FULL. On average, ICARO is 8% worse than OPT.

The figure on the right shows how close is ICARO to the optimal buffer under-run time obtained by both OPT and FULL. We plot ζ using the same coordinates as those of the previous figure. Here the white part of figure highlights where ICARO is able to achieve optimal performance, while other darker areas correspond to slightly worse performance. Notably, for no parameters the buffer under-run time was larger than $0.01t$ and very rarely larger than $0.001t$, chiefly for high requirement and small buffer size. Also, the $0.0001t$ contour is noisy due to the system sensitivity to input parameters in that particular region.

Fig. 6 provides results equivalent to those of the previous set of figures, but obtained for the urban scenario. Here, ICARO performance is slightly worse than those obtained in the suburban scenario. This is due to two main reasons: first, the urban environment shows a lower time correlation due to a more scattered fast fading and, second, to the lower accuracy of Markovian estimation compared to trajectory-based predictors.

Nonetheless, ICARO maintains, in the urban scenario, the positive outcome obtained in the suburban: the buffer under-run time (right) is most often optimal and, when it is not, the increase never reaches 1% of the buffer under-run time obtained by OPT; the cost savings (center) are mostly larger than 10% and as good as 20% in a limited parameter region.

Since ICARO gives priority to avoiding buffer under-runs, it can obtain higher cost savings when the ratio between requirements and available capacity is lower. Thus, since ζ is always lower than 1%, the algorithm is able to trade off cost efficiency for robustness effectively and it is able to achieve up

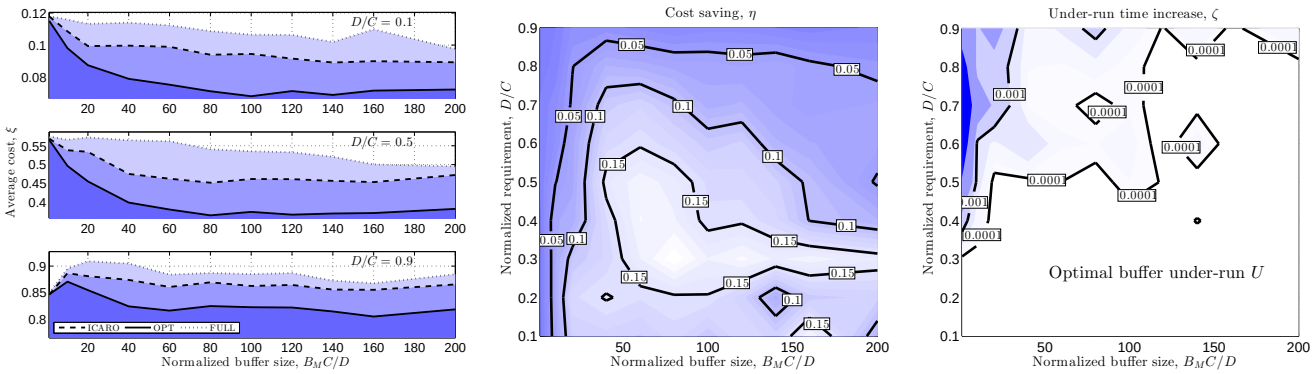

Fig. 6. Overall performance comparison among ICARO, OPT and FULL algorithms in the urban scenario. ξ, η and ζ for different operational conditions ($B_M \sum_i c_i / \sum_i d_i$ and $\sum_i d_i / \sum_i c_i$) are plotted on the left, center and right respectively.

to 25% cost reduction when the conditions are favorable, but it is never too aggressive when the future capacity estimation does not allow to do so.

Finally, when ICARO achieves the best results, the system is able to sustain the same quality of service while saving up 25% of the network resources or, analogously, 25% more users can be served with the same capacity. Conversely, where it obtains the worst performance, it is the system condition itself that offers small improvement margins: in particular for very small buffer sizes and/or high requirements.

VIII. Conclusion

In this paper we addressed the problem of resource optimization for mobile networks under imperfect prediction of future available capacity. To this aim we developed a general prediction model that allowed us to account for different prediction tools to encompass short to medium and long term prediction. This joint estimation technique allowed us to design ICARO, a lightweight resource optimization algorithm which achieves close-to-optimal performance. It achieves a nearly optimal buffer under-run time, which never exceeds the optimal time by more than 1%. In addition, ICARO is effective in the robustness/efficiency trade off. When the system conditions permit, it saves up to 25% of the network resources.

As a final remark, ICARO is the first practical resource optimization algorithm that does not assume perfect knowledge of the future system evolution and thus demonstrate the feasibility of prediction-enhanced optimization techniques for mobile networks.

Acknowledgment

The research leading to these results was partly funded by the European Union under the project eCOUSIN (EU-FP7-318398).

References

[1] T. Chiang, "What's expected of us," *Nature*, vol. 436, no. 7047, pp. 150–150, 2005.

[2] S. Wang, Y. Xin, S. Chen, W. Zhang, and C. Wang, "Enhancing spectral efficiency for LTE-advanced and beyond cellular networks [Guest Editorial]," *IEEE Wireless Communications*, vol. 21, no. 2, pp. 8–9, April 2014.

[3] U. Paul, A. P. Subramanian, M. M. Buddhikot, and S. R. Das, "Understanding traffic dynamics in cellular data networks," in *IEEE INFOCOM*, Shangai, China, April 2011, pp. 882–890.

[4] M. Z. Shafiq, L. Ji, A. X. Liu, and J. Wang, "Characterizing and modeling internet traffic dynamics of cellular devices," in *ACM SIGMETRICS*, New York, NY, USA, 2011, pp. 305–316.

[5] M. C. Gonzalez, C. A. Hidalgo, and A.-L. Barabasi, "Understanding individual human mobility patterns," *Nature*, vol. 453, no. 7196, pp. 779–782, 2008.

[6] Z. Lu and G. de Veciana, "Optimizing stored video delivery for mobile networks: The value of knowing the future," in *IEEE INFOCOM*, Turin, Italy, April 2013, pp. 2706–2714.

[7] H. Abou-zeid, H. Hassanein, and S. Valentin, "Energy-efficient adaptive video transmission: Exploiting rate predictions in wireless networks," *IEEE Transactions on Vehicular Technology*, vol. 63, no. 5, pp. 2013–2026, June 2014.

[8] J. Rasool and G. Oien, "Maximizing the throughput guarantees in wireless networks under imperfect channel knowledge," in *IEEE WCNC*, Paris, France, April 2012, pp. 2225–2229.

[9] K. Akkarajitsakul, E. Hossain, and D. Niyato, "Distributed resource allocation in wireless networks under uncertainty and application of bayesian game," *IEEE Communications Magazine*, vol. 49, no. 8, pp. 120–127, 2011.

[10] F. Jokhio, A. Ashraf, S. Lafond, I. Porres, and J. Lilius, "Prediction-based dynamic resource allocation for video transcoding in cloud computing," in *IEEE PDP*, Belfast, Ireland, February 2013, pp. 254–261.

[11] M. Ahmed, S. Spagna, F. Huici, and S. Niccolini, "A peek into the future: predicting the evolution of popularity in user generated content," in *ACM WSDM*, Rome, Italy, February 2013, pp. 607–616.

[12] N. Sadek and A. Khotanzad, "Multi-scale high-speed network traffic prediction using k-factor gegenbauer arma model," in *IEEE ICC*, vol. 4, Paris, France, June 2004, pp. 2148–2152.

[13] Y. Qiao, J. Skicewicz, and P. Dinda, "An empirical study of the multiscale predictability of network traffic," in *IEEE HDPC*, Honolulu, Hawaii USA, June 2004, pp. 66–76.

[14] N. Bui, F. Michelinakis, and J. Widmer, "A model for throughput prediction for mobile users," in *European Wireless*, Barcelona, Spain, May 2014.

[15] A. J. Nicholson and B. D. Noble, "Breadcrumbs: forecasting mobile connectivity," in *ACM MobiCom*, San Francisco, CA, USA, September 2008, pp. 46–57.

[16] J. Froehlich and J. Krumm, "Route prediction from trip observations," *SAE SP*, vol. 2193, p. 53, 2008.

[17] O. Østerbø, "Scheduling and capacity estimation in lte," in *IEEE ITC*, San Francisco, CA, USA, September 2011, pp. 63–70.

[18] S. Makridakis and M. Hibon, "ARMA Models and the Box-Jenkins Methodology," *Journal of Forecasting*, vol. 16, no. 3, pp. 147–163, 1997.

[19] J. D. Hamilton, *Time series analysis*. Princeton university press Princeton, 1994, vol. 2.

[20] C. Dovrolis, P. Ramanathan, and D. Moore, "Packet-dispersion techniques and a capacity-estimation methodology," *IEEE/ACM Transactions on Networking*, vol. 12, no. 6, pp. 963–977, December 2004.

[21] A. Burulitisz, S. Imre, and S. Szabó, "On the accuracy of mobility modelling in wireless networks," in *IEEE ICC*, vol. 4, Paris, France, June 2004, pp. 2302–2306.

[22] M. Hata, "Empirical formula for propagation loss in land mobile radio services," *IEEE Transactions on Vehicular Technology*, vol. 29, no. 3, pp. 317–325, 1980.