How Should Intelligence Be Abstracted in AI Research ...
AAAI Technical Report FS-13-02

Evolutionary Scheduler for Content Pre-Fetching in Mobile Networks

Omar K. Shoukry and Magda M. Fayek
Cairo University
Giza, Egypt

Abstract

Recently, an increasing number of mobile users are eagerly using the cellular network in data applications. In particular, multimedia downloads generated by Internet-capable smart phones and other portable devices has been widely recognized as the major source for strains in cellular networks, to a degree where service quality for all users is significantly impacted. In this paper we explore the novel concept of proactive content caching using evolutionary algorithms inspired by the inherent predictability of the mobile user behavior. Users can then use the cached version of the content in order to achieve a better user experience and reduce the peak-to-average ratio in mobile networks, especially during peak hours of the day. Finally, we confirm the merits of the proposed scheduler using real data traces of different user's requests and Wi-Fi availability. The results after applying the proposed scheduling algorithm show that up to 70% of the user content requests can be fulfilled i.e. the content were successfully cached before request. We also observe that proposed scheduler outperforms a baseline scheduler based on simulated annealing.

Introduction

Recently, an ever increasing demand was witnessed for the wireless spectrum resulting from the exponential growth of mobile data traffic due to the increasing penetration of smart phones and the adoption of bandwidth intensive multimedia applications. This trend has been widely recognized as the major cause of cellular networks congestion, forcing leading wireless operators around the world to consider significant additional investments in the cellular infrastructure. For instance, it has been reported in March 2009 that the traffic load of data services became nine-fold that of of voice services (Report 2012). This, in turn, gives rise to severe network congestion degrading the quality of service (QoS) perceived by mobile users.

Handa (Handa 2009) suggested three alternative approaches to solve the cellular congestion problem: (i) Scaling, (ii) Optimization and (iii) Network offloading. Although, scaling to 4G/LTE networks may help overcome the congestion problem in the short-term, however this solution remains a temporary measure, at best. Second, network Optimization may help relieve the congestion problem via efficient resource utilization, yet, only to a certain extent. Moreover, this approach is faced with serious issues, e.g., isolation of heavy data users, privacy preservation and policing users' traffic (Lee and Rhee 2010). Finally, offloading to secondary infrastructure provides an alternate path to wireless delivery. According to (Research 2011), the majority of traffic (63%) generated by smart phones, tablets and feature phones will transfer onto the fixed network via Wi-Fi by 2015.

This paper embraces a fundamentally different approach that is based on the novel framework of proactive resource allocation pioneered by El Gamal et al. (H. El Gamal 2010). The basic idea is to exploit the inherent predictability of user behavior to pre-fetch content (i.e. before demand) to ease congestion and enhance the user experience. There has been ample evidence in the literature pointing to the predictability of mobile user behavior (C. Song 2010). In (D. Larrabeiti and Serrano 2004), the authors argue that a large portion of the Internet users have repetitive content access patterns over time (i.e. days). In addition, mobile users exhibit Wi-Fi access and phone battery patterns that are repetitive over time. Our prime concern in this paper is to characterize and leverage those patterns to retrieve content before demand. This paradigm shift not only enhances the capacity of the cellular network but also results in an enhanced user experience by eliminating the delays and low connection rates that often impair the data connections over the cellular infrastructure.

Scheduling is a classic problem that has been studied in multiple disciplines(Pinedo 2012). The content scheduling problem at hand is inherently multi-objective (delivery rate, content freshness and resource utilization, among other objectives) and the tasks to be scheduled are not necessarily available at the beginning of scheduling. Techniques as multi-objective Evolutionary Algorithms (Johnston 2008) and multi-objective Meta-heuristics(Kumar 2011) are largely used for solving such problems. According to (Johnston 2008), the results obtained have shown that evolutionary algorithms can be effectively applied to the intrinsically multi-objective scheduling problem of large scale space network communications scheduling. Multi-objective flow shop scheduling using Meta-heuristics was discussed by Kumar(Kumar 2011). The objective therein was to minimize the tardiness, make span, earliness and the number of tardy jobs, concurrently. Genetic algorithms (GA) and simu-

lated annealing (SA) were used to solve this problem, and a hybrid was proposed to provide satisfactory results.Memetic Algorithms (MA) are like GA as both are population based metaheuristics. Operations as selection, reproduction and replacement are found similar to GA. The main difference is the update step in each generation. This step is concerned with local search between current generation individuals to find better solutions (Krasnogor and Smith 2004).

In this paper we introduce a novel architecture for the proactive content caching and its major building blocks, namely the loggers, residing on the mobile phone, the behavioral modelers (profilers) and scheduler, residing on the cloud. Second, the evolutionary proactive scheduler leveraging the developed stochastic models for the user behavior processes of interest, namely content consumption, Wi-Fi access and phone battery state. Motivated by the sheer complexity of the problem, attributed to its combinatorial nature, we introduce an evolutionary approach without explicitly generating the huge search space of all available solutions to the muli-objective problem.

The rest of this paper is organized as follows. The First Section overviews the overall system architecture. Afterwards, the next Section formulates and solves the proactive scheduling problem via an evolutionary algorithm. The following Section conducts a performance evaluation study. Finally, the last Section concludes the work and point out potential directions for future research.

System Overview

The ultimate objective of the system is to find a schedule for pre-fetching content items, over the course of a day, before actual user demand. The entire system hinges on developing trace-based stochastic models (profiles) for the mobile user behavioral processes of interest, namely content consumption, Wi-Fi and battery state. Given the profiles, an estimate of the average download data rate is computed to estimate the success probability SP for content caching at a given time slot. Using the calculated schedule, user content can be pre-fetched and made available ahead of demand.

As shown in Fig. 1, the system consists of three major building blocks: (A) User behavior loggers residing on the mobile side, (B) Trace-based stochastic profilers modeling the user behavior and (C) Proactive scheduler which retrieves "predicted" content of interest before demand. The loggers are responsible for capturing the behavior of the mobile user device and sending them to the cloud. The logged data, pertaining to the demand side (content consumption) and resource side (average data rate and battery state), are then used to create representative users' profiles. The scheduler leverages the user profiles to build schedules for content pre-fetching.

Logging User Behavior

This module reside on the user's mobile phone with a responsibility of logging: The user Wi-Fi connectivity, the phone battery state and the content usage behavior. The Connectivity/Data Rate Logger Module is responsible for monitoring the user's Wi-Fi connectivity state and dynamics. Upon each user entry/exit from a Wi-Fi network, the mobile operating system will notify the connectivity logger to append a new log entry including the current connectivity information. The Wi-Fi connectivity model aims at capturing the visited networks, the residence time in each network and the available resources (i.e. bandwidth) throughout each visit. Active probing is adopted, for Internet downlink bandwidth estimation (H. Lee and Kim 2007) (K. Lakshminarayanan and Padhye 2004) (S. Shah and Nahrstedt 2005). The Battery State Logger Module is responsible for monitoring the battery and listening to the following battery events: Battery charging, Battery low and Battery OK. The Content Usage Logger Module is responsible for logging the content consumption of the users' applications traffic containing the requested content details as application name, application category and request time.

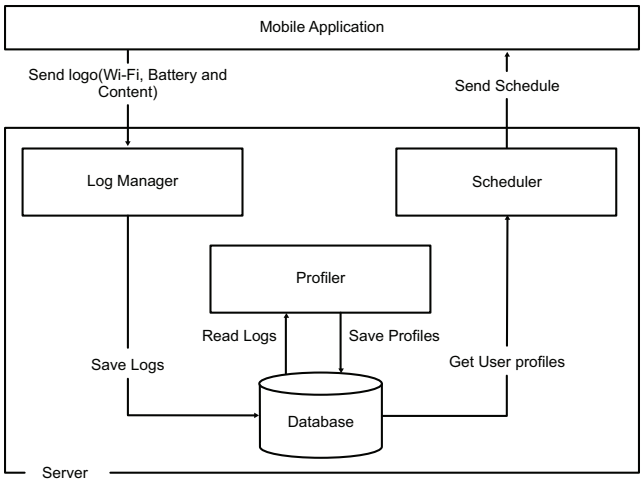

Figure 1: System Architecture

Probabilistic User Behavioral Models (Profiles)

We divide the day into a number of scheduling periods (windows) where each window is divided into a number of time slots with equal intervals (system parameter). The profiler uses the data from the logs (battery usage, average data rate and the accessed content) to generate representative histograms, which constitute the user profile. Generally, the sufficient statistics for a process involves computing the joint probability of events at different time instants in order to capture the temporal correlations. This statement is general and true for all three processes at hand. However, to reduce the model complexity, we assume that the marginal probability in a specific time slot is a sufficient statistic for the data rate and battery state processes. This implies no temporal correlations are captured for these two variables. On the other hand, the content usage process is assumed to capture short-term temporal correlations, manifested through the successive download states from the same category over a sequence of time slots. This short-term correlation is intuitive since a typical user goes through a sequence of download events upon visiting a content category of interest. The three above mentioned processes generally exhibit correla-

tion among them. Based on intuition, we assume that the battery state and the available BW processes are correlated. In order to simplify the model, the content usage process is assumed to be uncorrelated to the bandwidth and Battery state processes.

The Joint Battery-Download BW Model models the user's resource availability. As illustrated earlier, the battery logger periodically logs the state of the battery and lists it in terms of the following three states: CHARGING, OK and LOW . On the other hand, the Wi-Fi model provides a breakdown of the probability of a certain Wi-Fi BW rate available during each Wi-Fi logging period. We assume that the available download rate is such that it can be divided into the following five data rate bins in kbps e.g., 0-20, 20-40, 40-60, 60-80, 80-100. Thus, the profiler creates histograms for the battery and bandwidth with 15 different states (3 battery states * 5 data rate states). For each slot, the probability that the combined state is ("CHARGING and 0-20", "OK and 0-20", "LOW and 0-20","CHARGING and 20-40"...etc) is computed from the logs. This state probability is simply the % time spent in that state over the entire segment duration. Examples of the battery and Wi-Fi histogram profiles are shown in Fig. 2.

The Content Usage Profile models the user's content demand. The content usage logger periodically logs the content (application) requests and lists them in terms of the Application Categories. Afterwards, the content usage modeler generates a histogram profile for a given user. We assume that the system can retrieve one content item per slot and that we have M applications categories. The content usage logs are used to determine the probability of consuming any of the M categories accessed by a given user. Generally, different users may have different values of M, however, M is fixed for a given user in a single time slot. The content usage profiler computes the probability of requesting each application category (Content State) within a specific time slot by dividing the frequency of requesting each content by the total number retrievals over a pre-specified period of time.

Proactive Content Scheduler

Using the generated profiles, the scheduler employs the predictable user behavior, and the capabilities of smart phones to meet the ever-growing demand for the limited, non-renewable wireless spectrum. This proactive content scheduler constitutes the center, and most critical, piece of the proposed content delivery system for mobile users. The resulting schedule delivers content before demand (intelligent prefetching) that allows off-peak data offloading and a better user experience regarding content consumption. Intelligent offloading also improves network resource utilization by offloading across time and/or secondary networks (Wi-Fi).

System Model

The scheduling scope is specified as one day as we assume one segment per day throughout the paper. For any day of the week, a schedule is generated based on the constructed probabilistic model for that day. r_i and b_i denote the average download rate and battery state in slot i, respectively. The system accommodates M different types of content items (application categories). A day is divided into a number of slots with fixed duration, denoted, D. Slot duration is the same, and fixed, throughout constructing the behavioral models and scheduling. A single day is split into N slots where $N = \frac{24*60}{D(mins)}$. We assume that a single content item, at maximum, can be retrieved in a single slot. Notice that no content items are retrieved, in a particular slot, due to the lack of connectivity and/or battery resources or violation of content freshness constraints. A scheduling policy, i.e. the slot-content item assignment, for the N slots is modeled by the vector $\vec{S}$ as follows

$$\vec{S} = [C_i^1, C_j^2 ....C_k^N]$$

where C_i^n represents retrieving content item of type i in slot n and $1 \leq i, j, k \leq M$.

Problem Formulation and Complexity

Our prime objective in this paper is to schedule content item downloads, over the course of a day, in order to maximize the probability of the user having the content cached before demand, subject to data rate, battery and content freshness constraints. Evidently, the optimization problem in hand is combinatorial and, hence, the optimal scheduler has to examine all $|\vec{S}| = (M+1)^N$ combinations where M is the number of application categories of interest and N is the number of slots per day. This, in turn, gives rise to exponential complexity of the optimal policy with the number of slots, which makes the optimal prohibitively complex and practically infeasible.

Evolutionary Scheduler

The scheduling scope is defined as a 24 hours day. For each day a schedule is generated based on the profiled data (prediction) of the same day. For each slot period the probabilities are calculated during the profiling and used by the scheduler to select an item for caching. This gives a maximum of N items (number of slots) to be fetched during each day. The evolutionary representation contains of different solutions (schedules). Each schedule consists of N slots, with an assigned Application Content AC item to cache in each slot. Each AC can be one of the M values (application names). Uniform Crossover is selected for generating subsequent solutions. Randomly exchanging the AC values of two slots is used as mutation. A random set of solutions L (individuals) are initially generated. Crossover and mutation are applied probabilistically to create new generation. This is repeated for a given number of of iterations G. A final set of solutions are generated and the highest fitness solution is selected. The update step in MA is selected to use simulating annealing approach for local search.

The proposed evolutionary proactive scheduler hinges on two important notions, namely Reward R and Utility U where the Reward captures the benefit of assigning a content item (Application Category) to a given slot and the Utility captures the ability to retrieve this item in the given slot based on the availability of battery and bandwidth resources. The proactive scheduler tries to form a list of tasks with a

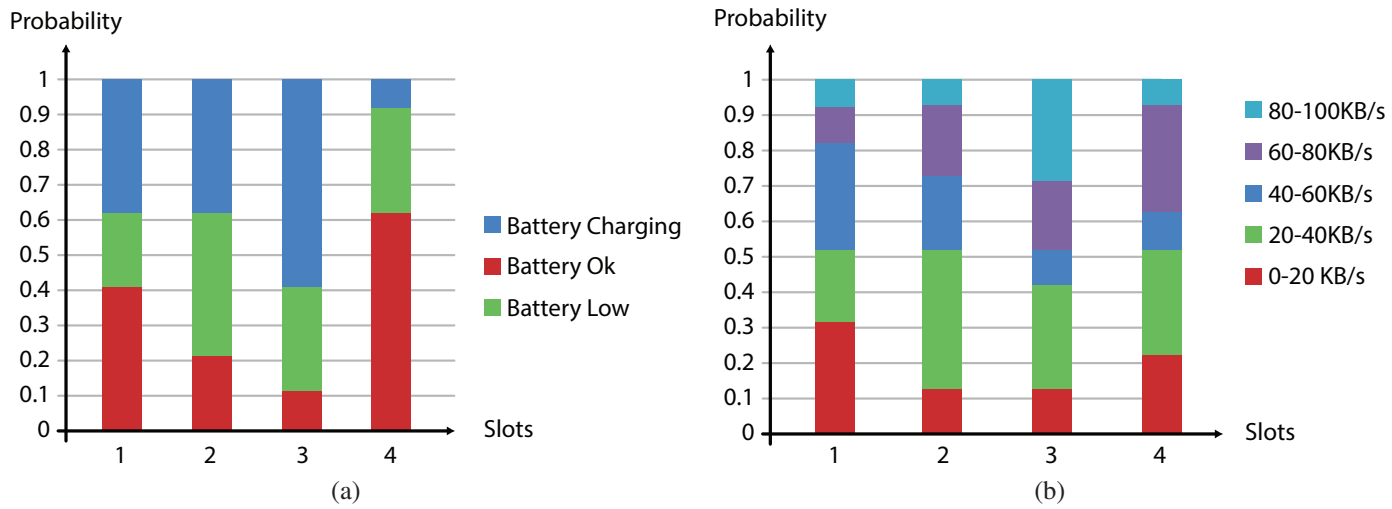

Figure 2: The (a) Battery profiling histogram, and (b) Wi-Fi profiling histogram.

maximum total utility. The utility of the schedule is the summation of all slots utilities, which is the slot Reward R multiplied by the success probability SP. The Reward for each content item in a specific slot depends on the probability of requesting this item within successive slots in a window W. This leads to assigning the content item to a slot based on the highest probability of being requested in the next time slots. The success probability of the slot depends on the bandwidth and battery state profiled in this slot(available resource). An evolutionary based algorithm is used to generate different solutions and select the results with the maximum total utility. Next, we give a detailed description of the evolutionary proactive scheduler.

The algorithm schedules tasks, using one slot for each task, for caching relevant content before its demand. After defining the number of slots N with the beginning and end times of each slot, a random Application Content (AC) is assigned to each slot. The reward R of each slot is calculated and the Utility U is estimated by multiplying the reward with the success probability. The total schedule fitness is the summation of utilities of all slots. The Reward R of an AC at a specific slot (t_{Slot}) is defined as

$$R(AC, t_{slot}) = \sum_{t=t_{slot}+1}^{W} ACProfile[t, AC] \qquad (1)$$

Thus, the $R(AC, slot)$ is determined by summing the profiled values (i.e. probabilities) of the selected AC over all slots following the slot of interest and within a window size (W) which constitutes a parameter of the system. The window size W represents the "freshness" duration selected before this cached content becomes obsolete. The whole process is repeated for all slots after modifying the AC profiles values of each scheduled slot. The AC profiles are modified based on the scheduled slot selected AC. The AC profile values are decreased within a window just after the current slot time. This modification reflects the fact that an AC has

Algorithm 1 Evolutionary Scheduler

READ Content Profiles
READ Battery and Wi-Fi Profiles
$i = 0$
$j = 0$
Initialization: Generate Initial random population
while $i < N$ **do**
  CALCULATE Application Content Reward of each slot
  UPDATE Content Profile
  $i = ++$
end while
CALCULATE Utility of each slot
Evaluate the fitness of each individual (utility summation) in this population
while $j < G$ **do**
  Selection : Select best fit individuals
  Evolution: Generate new individuals by applying crossover and mutation
  Evaluate the fitness of newly generated individuals
  Update individuals (local search using simulated annealing)
  Replace some of the (least fit) individuals from the old population with new (higher fit) individuals
  $j = ++$
end while

been scheduled and would be fresh to use for a given window of time.

The Utility U of a specific slot (t_{Slot}) having a Reward R) is given by

$$U(t_{slot}) = SP(t_{Slot}) * R \quad (2)$$

The slot utility represents the benefit from caching this AC at this time slot. The success probability SP is the probability of successfully caching this AC at this time slot and is defined as the probability of the bandwidth (BW) and Battery at slot (t_{Slot}) allocated by content AC matching the BW and battery profiles.

After generating the first set of solutions (first generation), crossover and mutation are performed to evolve the current set of solutions. This process is repeated for number of iterations till a final set of solutions is generated. The schedule with the highest fitness is selected as the final solution. A final modification is done by combining similar requests (same AC) over consecutive slots. This modification adds the number of items of a specific AC to the first occurrence of consecutive slots and removes duplicates. The final schedule may have more than one item to retrieve in a specific time slot with empty slots to follow.

Performance Evaluation

The Baseline Scheduler

A baseline scheduler is implemented to compare the results with the proposed scheduler algorithm. SA was chosen as a stochastic algorithm for creating random schedules. In the scheduling context, a random list of tasks is generated (assignment of content to time slots). The Utility of the schedule is calculated as shown before using the same Reward notion. The total schedule utility is calculated by summating the utility of all assigned slots. At each iteration a new randomized schedule is generated and selected if it has a higher total utility value than the previous schedule. This is repeated for G times (stopping condition). The value G is a fixed value, chosen to accommodate reasonable results in acceptable computation complexity.

Performance Results

Table 1 includes the numerical values of the system parameters used in our performance evaluation study. In this section, smart phone traces collected by Rice university LiveLab project are utilized (C. Shepard and Kortum 2010) (Z. Wang and Chishtie 2012). LiveLab is a methodology to measure real-world smart phone usage and wireless networks with a re-programmable, in-device, logger designed for long-term user studies. LiveLab was deployed for a number of iPhone 3GS users. This includes 24 Rice University students from February 2010 to February 2011, and 10 Houston Community College students from September 2010 to February 2011.

It is worth mentioning that some preprocessing on the data to fit the proposed system has been performed. For the application data, user application usage frequency and duration were measured and a set of applications was selected to work with (CNN, ESPN, YouTube, Facebook, Twitter and LastFM). Focusing on this set of application categories is due to their popularity and widespread. For the associated Wi-Fi data, the entries were summarized into a set of Wi-Fi enter/exit entries. For the battery usage data, both the connectivity data and battery level data were used. Both data entries were combined and processed to get the entries representing the required battery activities (battery OK, low, and charging).

Table 1: System Parameters

$Parameter$	$Definition$	$Value$
N	Number os Slots	48
M	Number of Applications	10
P	Slot Period (min)	30
W	Window Size (slots)	6
I	Number of Items per Slot	1
G	Number of generations	1000
L	Number of individuals	100
C	Crossover Probability	0.7
T	Mutation Probability	0.3

The MA scheduler, GA scheduler and baseline SA scheduler, were implemented and tested for 25 users. This set was selected out of the 34 users available from the LiveLab data due to the availability of sufficient Wi-Fi, battery and content logs over a sufficient period of time. Data was divided to a training set of 5 weeks and a testing set. A day was randomly picked from the available traces as the testing day and the previous 5 weeks were used for building the user behavioral models. Table 2 outlines the performance results for all 25 users. These results show that an average value of 56% hit ratio can be achieved using the implemented MA schedule and a value of 53% hit ratio using the GA. A maximum of 83% was achievable, which is higher than the baseline SA. The average cache utilization was found to be 61% for the MA and 64% for the GA. The time in cache value was measured to be the average time difference (in minutes) between the request time and the time of caching. This indicates how long the item resided in the memory before actually being consumed by the user. An average of an hour and half is a satisfying result given that most of the cached items are videos. This value indicates that caching happens relatively just before consumption and not so soon to keep content as fresh as possible. It is worth mentioning that while the utilization of the cache reaches 100% in some cases, the memory resources were generally under utilized. The results also show that the proposed GA and MA schedulers outperform the SA w.r.t. the average hit ratio by 43% and 50% respectively, and yields a higher cache utilization as well as lower time in cache per item. The results also shows that the MA is more accurate than the GA in terms of achieving higher hit ratio.

The proposed scheduling algorithm would intuitively perform better in a resource abundant environment due to the Wi-Fi availability and good battery conditions. To get further insights, users were classified into two main groups: Resource abundant (RA) users, and resource challenged (RC) users. Based on this classification, Table 3 outlines the re-

Table 2: Overall performance of the scheduling algorithm compared to the baseline scheduler for all users

Performance Metric	MA	GA	SA
Hit ratio	56 % $\pm$ 20%	53 % $\pm$ 19%	28 % $\pm$ 17%
Cache Utilization	61 % $\pm$ 15%	64 % $\pm$ 12%	31% $\pm$ 15%
Time in Cache	82.24 mins $\pm$ 29	90.33 mins $\pm$ 25	121.58 mins $\pm$ 35

Table 3: Results on different user classifications, resource abundant and resource challenged

Performance Metric	Resource Abundant	Resource Challenged
Hit ratio	74 % $\pm$ 9 %	42 % $\pm$ 7 %
Cache Utilization	90 % $\pm$ 4 %	38% $\pm$ 11%
Time in Cache	94.13 mins $\pm$ 19	72.33 mins $\pm$ 24

sults using the proposed evolutionary MA scheduler and previously mentioned system parameters. The results confirm that the proposed scheduler yields a higher hit ratio (approximately doubled) in an abundant resource environment as opposed to a resource challenged one. Resource abundant users were selected to be users with Wi-Fi availability period of 30% or more during the day and a battery "LOW" state duration less than 50% during the day. The cache utilization is intuitively better since more items are cached in a resource abundant environment. The time in cache, on the other hand, has a relatively lower value in the resource challenged environment due to the fact that fewer number of items are cached (i.e. less competition), so the probability of caching it just before consumption is high.

Conclusion

This paper addresses the congestion problem affecting the current 3G networks caused mostly by an increasingly excessive usage of multimedia applications on Smartphones. In this work, we suggest a method for pre-fetching content to reduce the congestion in 3G networks by intelligent offloading. This technique depends on the estimation of different user patterns as requested content, network availability and battery conditions. The results show that this approach can successfully fulfill the users' requests with a high percent (up to 70%) using the GA and MA. The results also show that both the GA and MA perform better than the baseline SA algorithm.

References

C. Shepard, A. Rahmati, C. T., and Kortum, P. 2010. Livelab: measuring wireless networks and smartphone users in the field. *ACM SIGMETRICS Perform*.

C. Song, e. a. 2010. Limits of predictability in human mobility. *Science 327, 1018*.

D. Larrabeiti, R. Romeral, A. A., and Serrano, P. 2004. Charging for web content pre-fetching in 3g networks. *4th International Workshop on Internet Charging and QoS Technology*.

H. El Gamal, J. Tadrous, A. E. 2010. Proactive resource allocation: Turning predictable behavior into spectral gain. *8th Annual Allerton Conference on Communication, Control, and Computing*.

H. Lee, V. Hail, K. Y., and Kim, E. 2007. Bandwidth estimation in wireless lans for multimedia streaming services. *Journal on Advances in Multimedis*.

Handa, A. 2009. Mobile data offload for 3g networks. *Intellinet-Technology, Whitepaper*.

Johnston, M. 2008. An evolutionary algorithm approach to multi-objective scheduling of space network communications. *Intelligent Automation and Soft Computing*.

K. Lakshminarayanan, V. P., and Padhye, J. 2004. Bandwidth estimation in broadband access networks. *ACM Internet Measurement Conference*.

Krasnogor, N., and Smith, J. 2004. A memetic algorithm with self-adaptive local search. *Proceedings of the Genetic and Evolutionary Computation Conference*.

Kumar, A. 2011. Multi-objective flow shop scheduling using metaheuristics. *PHD thesis in Mechanical Engineering*.

Lee, K., and Rhee, I. 2010. Mobile data offloading: How much can wifi deliver. *10th Annual SIGCOMM Conference on Computer and Data Communication Networks*.

Pinedo, M. 2012. Scheduling: Theory, algorithms and systems. *Springer, New York*.

Report, C. T. 2012. available online at http://www.cisco.com. Report, Cisco Visual Networking Index.

Research, J. 2011. Relief ahead for mobile data networks as 63% of traffic to move onto fixed networks via wifi and femtocells by 2015. http://juniperresearch.com/viewpressrelease.php?pr=240.

S. Shah, K. C., and Nahrstedt, K. 2005. Dynamic bandwidth management in single-hop ad hoc wireless networks. *Kluwer Academic Publishers MONET*.

Z. Wang, F. Xiaozhu Lin, L. Z., and Chishtie, M. 2012. How far can client-only solutions go for mobile browser speed? *Int. World Wide Web Conf*.