# Streaming Scalable Video over WiMAX Networks

Somsubhra Sharangi
School of Computing Science
Simon Fraser University
Surrey, BC, Canada

Ramesh Krishnamurti
School of Computing Science
Simon Fraser University
Burnaby, BC, Canada

Mohamed Hefeeda
School of Computing Science
Simon Fraser University
Surrey, BC, Canada

***Abstract*—Broadcasting multiple scalable video streams over wireless broadband access networks in real time is a challenging problem, because of the limited channel capacity and variable bit rate of the videos. The difficulty is further increased in the presence of receiver buffer size limitations which may introduce buffer overflow possibilities. The Multicast/Broadcast Service feature of mobile WiMAX network is a promising technology for providing wireless video broadcast services. In this article, we describe a substream selection problem which arises when multiple scalable video streams are broadcast based on the Multicast/Broadcast Service feature to a number of buffer size constrained receivers. We first show that the problem is NP-Complete and design a polynomial time approximation algorithm based on convex optimization and dynamic programming techniques. We mathematically prove that the solution obtained through our algorithm is always within a constant factor of the optimal solution. Through simulation we show that under real time requirements our algorithm provides solutions which are within 1dB of the optimal solutions.**

## I. Introduction

Video multicasting to mobile devices has emerged as one of the popular services over upcoming next generation wireless networks. Several promising applications like mobile TV, mobile video-conferencing and mobile multiplayer gaming make use of these services. Technologies from traditional cellular wireless telecommunication networks, terrestrial TV broadcast networks, last mile Internet access networks and other such domains are being proposed for realizing these applications. The IEEE standard 802.16 [1] or WiMAX is a technology which was originally designed for providing last mile wireless broadband access and it is now being considered for providing mobile TV services. For example, Yota Telecom [2] has recently started a TV service with 25 channels over its 10Mbps mobile WiMAX network, and UDCast [3] has announced plans for developing broadcast TV service supporting around 50 channels over mobile WiMAX. Observing this trend we expect to see more deployments of WiMAX based mobile TV services in the near future. Maintaining high quality of service in video delivery is one of the main challenges for these applications. Even though a considerable amount of work has been done to make these deployments a reality, several issues still remain unattended.

In the WiMAX physical layer, data is transmitted over multiple carriers in Time Division Duplex (TDD) frames. As illustrated in Figure 1, each frame contains header information and upload/download maps followed by bursts of user data. Since video dissemination is expected to be a prevalent traffic pattern in future networks, the WiMAX standard defines a service called Multicast and Broadcast Service (MBS) in the MAC layer to facilitate broadcast and multicast. Although streaming video can be transmitted as a unicast service, broadcast and multicast are more resource efficient. Using MBS, a certain area in each TDD frame can be set aside for multicast-only or broadcast-only data, as shown in Figure 1. The entire frame can also be designated as a download-only broadcast frame. A major task of the MBS module is to allocate video data to the multicast/broadcast data area in each frame, such that, the real time nature of the video stream is maintained. This results in stringent QoS and efficiency demands on the allocation algorithm.

In this paper, we propose an efficient multimedia broadcast framework over mobile WiMAX networks utilizing the MBS features. In particular, our paper considers broadcasting multiple scalable video streams to mobile receivers. We mathematically formulate the problem of selecting the best set of substreams from the scalable video streams in order to maximize the quality for mobile receivers. We show that this problem is NP-Complete. We propose an approximation algorithm based on dynamic programming techniques, which produces near optimal solutions. Using simulation and mathematical analysis we show that the algorithm is efficient in terms of execution time and achieves high radio resource utilization.

The rest of this paper is organized as follows. In Section II, we summarize the previous works related to scalable video broadcast over WiMAX. We state our problem and present the analytical formulation for it in Section III. In Section IV, we present an approximation algorithm to efficiently solve our problem. Section V describes our simulation setup and results. Finally, we summarize our findings and conclude the paper in Section VI.

## II. Related Work

In this section, we first describe previous work with respect to video transmission over WiMAX network, and then describe earlier approaches related to scalable video broadcast in WiMAX networks. Hosein [4] describes the frame allocation problem for broadcasting variable bit rate video over WiMAX, but does not consider scalable video content. Wang et al. [5] discuss an architecture for video broadcasting in a multi-base-station WiMAX system. Their work focuses on coverage and spectral efficiency issues and considers only temporal video scalability. Cohen et al. [6] combine a group of TDD frames together into a super-frame. They describe a cost based scheme where a cost function is associated with each user-channel pair. Three user interaction models are considered: (i) user can either be statically hooked to a channel, (ii) user can choose to listen to a channel, or (iii) the user channel association can keep changing based on the transmission medium conditions. The work in [6] does not consider the delay requirements which are central to video streaming.

Jiang et al. [7] propose a scheme to transmit scalable video streams in which two layers of each video are transmitted separately. The base layer is transmitted as one stream over a reliable channel while the enhancement layer is transmitted as a different stream over a less reliable channel. Conceptually, this work implements a rate adaptive multiple description coding. However, it describes only one stream and it does not address the resource management problem arising in multi stream transmission scenarios. Reguant et al. [8] consider splitting a video stream into two streams and transmitting them over two different broadcast networks. The first stream is transmitted over a DVB-H network at all times while the second stream is transmitted over WiMAX network most of the time. If the user wants to use some other non-video application in parallel, the stream going through WiMAX is degraded to accommodate that application. This ensures a minimum video quality at all times while maintaining the flexibility of using other applications. While this approach has its benefits, it is not very attractive from a deployment point of view since the service provider has to install and manage the infrastructure for two different kinds of networks. Also the solutions described in both [7] and [8] evaluate the performance of video streaming as an application along with other WiMAX applications and do not utilize MBS. In contrast, our approach considers a multimedia-intensive system with extensive use of MBS.

## III. Problem Statement and Hardness

Our work focuses on optimally utilizing the Multicast/Broadcast Service (MBS) to stream multiple scalable videos to mobile receivers. In this section we state the problem and show that it is NP-Complete. We also present the mathematical formulation of the problem. For quick reference, we list all symbols used in the formulation in Table I.

### A. Problem Statement

We consider a scenario where a number of scalable video streams are available at a WiMAX base station. Each stream is to be broadcast using MBS to a group of mobile subscribers. At the WiMAX base station, the MBS module allocates a fixed-size data area in the download section of each TDD frame. All video streams are to be allocated only within this MBS data area. As per the mobile WiMAX standard, each MBS data area can transmit a different amount of data depending on the modulation scheme chosen, which is in turn selected based on the wireless channel conditions. For a broadcast application, the modulation scheme is to be selected for a group of subscribers. Since subscribers receiving a broadcast service may have different channel conditions, selecting a modulation scheme which is beneficial to all the subscribers in the group is computationally expensive. Therefore, we assume that the modulation scheme for the broadcast service is a pre-calculated parameter and does not change throughout the transmission. Thus, each MBS area transmits a fixed amount of data, in effect, creating a fixed bandwidth broadcast channel. We consider a scheduling window composed of a number of MBS data areas. Data from the video streams are to be allocated to the MBS areas in the scheduling window. Due to the variable bit rate (VBR) nature of the video streams, the aggregate data rates may exceed the broadcast channel capacity. Hence, in each scheduling window, we need to decide which layers to send for each stream. We assume that the base station has enough buffer space to hold the VBR traffic for one scheduling window. This way, the data rates can be assumed to be constant during a scheduling window, but they vary across scheduling windows. Since the bit rates and the receiver buffer states change in each scheduling window, the allocation has to be computed for every scheduling window. We also assume that all subscribers served by the base station have a fixed amount of buffer which is used to temporarily store the incoming video data before playing out. Thus the optimal substream selection problem we need to solve can be stated as follows.

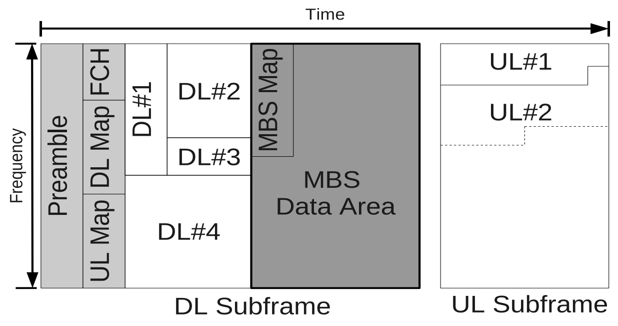


Fig. 1. WiMAX OFDMA TDD Frame structure.

*Problem 1 (Optimal Substream Selection Problem:):* Select the optimal subset of layers from each scalable stream to broadcast over a WiMAX network such that: (1) the total data transmitted in a scheduling window does not exceed the window capacity, (2) the average quality of all selected substreams is maximized, and (3) the subscriber playout buffer does not overflow or underflow.

### B. Problem Hardness

Let us assume that for a given radio modulation scheme, the MBS data area in each frame can accommodate $F$ amount of data and the TDD Frame takes $\tau$ time to be transmitted. Let the scheduling window consist of $P$ such frames. Then, the maximum amount of data that can be transmitted within the scheduling window is given as $C = PF$. We have $S$ scalable video streams. Each scalable stream $s$, $1 \le s \le S$, has at most $L$ layers. The value of $L$ can be different for each stream.

Therefore, for each stream we have $L$ substreams to choose from, where a substream $l$ includes layer $l$ and all layers below it. Let the data rates and quality values for selecting substream $l$ of stream $s$ be $r_{sl}$ and $q_{sl}$ respectively. Here $r_{11}$ denotes the data rate of the base layer of the first stream. Thus, we have the problem of choosing the substreams such that the average quality across the video streams is maximized subject to the following constraints. The first constraint is that the total data to be transmitted must fit into the MBS areas in the current scheduling window. The second constraint is that the buffers at the subscribers must not run out of data anytime during the scheduling window, and the third constraint is that the base layer of each stream must be transmitted to guarantee a basic service level agreement.

*Theorem 1: The Optimal Substream Selection Problem is NP-Complete.*

*Proof:* First, we consider a relaxed version of the problem with no buffer overflow or underflow constraints. Thus, we are left with the problem of selecting the substreams such that the average quality is maximized. We assume that in each scheduling window at least all the base layer streams have to be transmitted due to service level agreement. Thus, we further modify the problem by eliminating the base layer constraints, which can be trivially done, by reducing the scheduling window capacity by the sum of data rates of all base layers. Therefore, the modified data capacity can be given as $C' = C - \sum_{s \in S} r_{s1}$. Now we are left with the problem of deciding which substreams to chose from each stream. We show that this problem is equivalent to the NP-Complete 0-1 Multiple Choice Knapsack Problem (0-1-MCKP) [9], which is defined as follows. Given $M$ classes $N_1, \ldots, N_M$ of items to pack in some knapsack of capacity $W$. Each item $(i, j)$, where $i \in M, j \in N_i$, has a profit $p(i, j)$ and a weight $w(i, j)$. The problem is to choose at most one item from each class such that the profit sum is maximized without having the total weight exceed $W$. We reduce the 0-1-MCKP problem to Optimal Substream Selection problem in polynomial time as follows. We make the data rates of choosing a substream represent the item weight and the corresponding quality values represent the profit of choosing an item. We also make the streams represent the multiple choice classes and the scheduling window capacity represent the knapsack capacity. Thus, we have an MCKP instance with $S$ classes, $L-1$ items per class and a knapsack capacity of $C'$. This means that an efficient solution for the simplified Optimal Substream Selection Problem could be employed to efficiently solve the NP-Complete 0-1-MCKP problem. In other words, the substream selection problem is NP-Hard. In addition, clearly a solution for the simplified Optimal Substream Selection Problem can be verified in polynomial time. Thus the simplified Optimal Substream Selection Problem is NP-Complete. Consequently, the more general Optimal Substream Selection Problem subject to buffer overflow and underflow constraints is also NP-Complete. ■

TABLE I
LIST OF SYMBOLS USED IN FORMULATION

| Symbol | Description |
|---|---|
| $S$ | Number of streams |
| $L$ | Number of layers |
| $q_{sl}$ | PSNR of substream $sl$ |
| $r_{sl}$ | Data rate of substream $sl$ |
| $b_{sl}$ | Number of frame sized blocks of substream $sl$ |
| $n_{sl}$ | Number of frame bursts for substream $sl$ |
| $t_{sl}^{k}$ | Start of burst $k$ of substream $sl$ |
| $w_{sl}^{k}$ | Width of burst $k$ of substream $sl$ |
| $\tau$ | Duration of a TDD frame |
| $F$ | Capacity of MBS data in a TDD frame |
| $P$ | Number of frames in scheduling window |
| $C$ | Data capacity of scheduling window |
| $B$ | Buffer size at the receiver |
| $u_s$ | Initial buffer level for stream $s$ |

## C. Analytical Formulation

We assume all subscribers have $B$ amount of buffer available for the video streaming application and the data rate and quality values for all substreams of each stream are known ahead of the scheduling window. This information can either be obtained as a separate meta data for each stream or if the scalable video is encoded using H.264/SVC [10] and the base station is media-aware, this information can be obtained directly from the encoded video stream itself using the Supplementary Enhancement Information (SEI) messages. Let the data rate values of substreams be $\{r_{s1}, r_{s2}, \ldots, r_{SL}\}$ and the corresponding quality values be $\{q_{s1}, q_{s2}, \ldots, q_{SL}\}$. Each scheduling window is of duration $\tau P$. If substream $l$ of stream $s$ is selected, the amount of data to be transmitted during a scheduling window can be given as $\tau P r_{sl}$. Let binary variables $x_{sl}$ take the value 1 if substream $l$ of stream $s$ is selected for transmission in the current scheduling window and 0 otherwise. For a substream, we define a burst as a consecutive set of MBS data areas allocated to the substream in the scheduling window. For any schedule, let $n_{sl}$ be the number of bursts for substream $l$ of stream $s$. We denote by variables $t_{sl}^{k}$ the starting frame number and by variable $w_{sl}^{k}$ the number of MBS data areas in burst $k$ for substream $l$ of stream $s$. The solution of the optimum substream selection problem should generate a list $< l, n, < t_{sl}^{1}, w_{sl}^{1} >, \ldots, < t_{sl}^{n}, w_{sl}^{n} >>$ for each stream. In the list, $l$ denotes the selected substream, $n$ denotes the number of bursts required for transmitting substream $l$, and $< t_{sl}^{k}, w_{sl}^{k} >$ denote the starting point and width of burst $k$, respectively. For a subscriber receiving channel $s$, let the buffer level at the beginning of scheduling window be $u_s$. We need to ensure that all data received during a scheduling window are also consumed in the same window. In other words $\sum_{k=1}^{n_{sl}} w_{sl}^{k} = \tau P r_{sl}$. At the same time we need to ensure that buffer overflow and underflow does not occur. At the end of each burst, the total data received is given by $\sum_{i=1}^{k} w_{sl}^{i}$. During that period the total data consumed is given by $\tau(t_{sl}^{k} + w_{sl}^{k}) r_{sl}$. Now in order to avoid underflow, the difference of these two terms must be greater than zero for all bursts. Similarly, the overflow conditions can be applied by constraining the difference to be never greater than $B$. Our objective is to maximize the average video quality over all the streams. We use the PSNR values of the streams to denote quality and take an arithmetic average of the PSNRs of the selected streams to denote the average quality. Let us assume that the data to be transmitted for each substream can be divided into $b_{sl}$ number of $F$ sized data blocks. Consequently, we have the following optimization problem.

$$\text{Maximize} \quad \frac{1}{S}\sum_{s=1}^{S}\sum_{l=1}^{L} x_{sl}q_{sl} \tag{P1}$$

$$\text{such that} \quad \sum_{s=1}^{S}\sum_{l=1}^{L} x_{sl}b_{sl} \quad \leq \quad P \tag{1a}$$

$$\sum_{l=1}^{L} x_{sl} \leq 1 \tag{1b}$$

$$u_s + \sum_{i=1}^{k} w_{sl}^{i}F - \tau(t_{sl}^{k} + w_{sl}^{k})r_{sl} \leq B \tag{1c}$$

$$u_s + \sum_{i=1}^{k-1} w_{sl}^{i}F - \tau(t_{sl}^{k-1} + w_{sl}^{k-1})r_{sl} \geq 0 \tag{1d}$$

$$[t_{sl}^{k}\ldots t_{sl}^{k} + w_{sl}^{k}] \cap [t_{sl}^{\bar{k}}\ldots t_{sl}^{\bar{k}} + w_{sl}^{\bar{k}}] = \emptyset \tag{1e}$$

$$\sum_{k=1}^{n_{sl}} w_{s}^{k} = x_{sl}b_{sl} \tag{1f}$$

In the above formulation the constraint (1a) makes sure that the selected substreams can be transmitted within the broadcast bandwidth. Constraint (1b) ensures that at most one substream is selected for each stream. Constraints (1c) and (1d) represent the buffer overflow and underflow constrains, respectively. Constraint (1e) implies that no two bursts of data blocks should be allocated to the MBS area of the same TDD Frame. Here the operator $[\ldots]$ denotes integer interval. This constraint is required because the streams are transmitted over a time-shared, multiple-access wireless channel where only one burst can be transmitted at a time. Constraints (1f) implies that if a layer is selected, then all the data blocks corresponding to the layer must be allocated in the schedule.

## IV. Proposed Approximation Algorithm

### A. Overview of the Proposed Algorithm

The proposed algorithm is called *Substream Selection Algorithm* and is denoted by SSA. The high level idea of the algorithm is as follows. We first find a set of near optimal substreams given the data capacity of a scheduling window. Then, we allocate them to the MBS areas in the frames of the scheduling window. If no feasible allocation is found, we reduce the problem instance by discarding the substream with lowest quality among all substreams. We solve the optimal substream selection problem again for the reduced set of substreams. This cycle is repeated until, either a feasible solution is found, or none of the substreams are selected. The block diagram of this general scheme is shown in Figure 2. First, we design an approximation algorithm for the multiple choice knapsack problem. Then, the frame allocation is done in a modified weighted round robin manner. As we showed

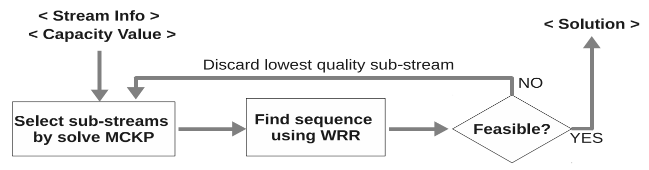


Fig. 2. High-level diagram of the Substream Scheduling Algorithm (SSA).

in Theorem 1, the problem of selecting optimal scalable substreams is equivalent to solving a 0-1 Multiple Choice Knapsack Problem. This problem has been well studied in the mathematical programming community and several near optimal solution schemes exist. The reader is referred to the survey by Lin [11] for a summary of the main results. Dynamic programming is one of the techniques used for designing approximation algorithms for the 0-1 Multiple Choice Knapsack Problem. However, dynamic programming solutions are often memory intensive and may involve large constants. In our algorithm we derive both an upper bound and lower bound on the value of the solution. These bounds significantly reduce the solution search space [12].

### B. Details of the Proposed Algorithm

Figure 3 summarizes the pseudo-code of the SSA algorithm. There are three main steps of the algorithm: (i) Finding an approximate solution to the substream selection problem, (ii) Allocating the selected substreams to the MBS data areas of the scheduling window, and (iii) Validating the schedule to confirm that there are no buffer underflow or overflow conditions. We describe each step in the following.

*1) Approximate Substream Selection:* In a naive dynamic programming solution we construct a table of all possible data rates for the given streams (i.e., $1\ldots\sum r_{SL}$) and their resulting quality values. We note that multiple quality values can result for a single aggregate data rate value depending on the composition of the substreams selected. Then, we search for the highest quality entry in the table such that the data rate is less than the scheduling window capacity. In our proposed algorithm, we first derive bounds on the solution value which will reduce the size of the search space. Then, we construct a dynamic programming table for all quality values within the bounds and find the solution substreams using backtracking.

**Bounding the Optimum Solution Value:** A lower bound on the optimal solution is obtained from a solution to the linear relaxation of the 0-1 MCKP problem as follows. A solution $x_c^*$ to the linear relaxation has the following two properties: (1) $x_c^*$ contains at most two fractional values and (2) when there are two fractional values in $x_c^*$, they belong to substreams of the same stream. For the proof of these properties the reader is referred to [13]. Let $z_c^*$ be the value of the objective function corresponding to $x_c^*$. Let $Q_0$ be the maximum of (a) the objective function value when both the fractional values are dropped from the solution and (b) maximum of the quality values of the fractional variables. If the optimal solution for the integer problem is $Q^*$, it is evident that $Q_0 \leq Q^*$. From the properties of $x_c^*$ it is evident that at most two variables are dropped. Since at most two variables are dropped, $z_c^*$ can be bounded as $z_c^* \leq 2Q_0$. Also, since the solution obtained by the linear relaxation must be greater than or equal to the solution obtained by the integer program, we have an upper bound on the optimum integer solution as $Q_0 \leq Q^* \leq z_c^* \leq 2Q_0$. We note that although the bound is obtained from linear programming theory, we do not require an LP solver to calculate $Q_0$. $Q_0$ can be calculated using the median finding algorithms in linear time [14].

**Recursive Table Generation:** Now that we know the bounds of the optimal solution value we define a dynamic programming formulation as follows. For all streams $s \in \{1,\ldots,S\}$ and all quality values $q \in, \{0\ldots,2Q_0\}$, we define $V(s,q)$ as the set of substreams from streams $1,\ldots,s$ such that no two substreams are selected from the same stream and the total quality of the selected substreams is $q$. If for a quality value $q$ the *at most one substream per stream* constraint is violated we set the corresponding sum of weights to infinity. Let $R(s,q)$ denote the sum of data rates selected in $V(s,q)$. We assume that the sum of data rates to produce zero quality is zero, i.e., $R(s,0) = 0$. Also, for the first stream, the data rate values can be computed easily as just the data rate of the substream, or the minimum of the data rates if more than one substream has the same quality, i.e., $R(1,q) = \min_l \{r_{sl}\}$ where $q_{sl} = q$. In mathematical terms, the first stream data rates can be expressed as in equation (2a). The data rates for the other quality values and other streams can be computed by the recursive definition described in (2b) and the optimum quality can be expressed by equation (2c).

$$R(1,q) = \begin{cases} \min_l\{r_{sl}\}, & \text{where } l \in L \text{ and } q_{sl} = q, \\ \infty, & \text{otherwise.} \end{cases} \tag{2a}$$

$$R(s,q) = \begin{cases} \min\{R(s-1,q), \min_{l \in L}\{r_{sl} \\ \quad +R(s-1,q-q_{sl})\}\}, & \text{when } q_{sl} \leq q, \\ R(s-1,q), & \text{otherwise.} \end{cases} \tag{2b}$$

$$Q^* = \max\{q | R(s,q) \leq PF\}. \tag{2c}$$

However, the size of the table can still be very large as it is bounded only by $Q_0$. Therefore we select a scaling factor $K = \frac{\varepsilon Q_0}{S}$, and scale down the quality values to $q'_{sl} = \frac{q_{sl}}{K}$. This operation considerably reduces the table size while admitting only a small error factor. We bound the quality degradation due to scaling in our mathematical analysis in Section IV-C.

**Backtracking:** Once we have computed the dynamic programming table, the solution quality value is obtained by a simple scanning of the table as in equation (2c). The solution substream vectors are found using a backtracking mechanism as follows. While constructing the recursive table, we store the composition of substreams leading to the data rates $R(s,q)$ as a list for each table cell. The solution substream vector is found using the additional information by backtracking from the cell containing the solution quality value.

**Substream Selection Algorithm (SSA)**

```
1. For each enhancement layer $i$ across all streams do
2.     Compute $\rho_i = r_{sl} - r_{sl-1}$ and $\phi_i = (q_{sl} - q_{sl-1})$
3. Select $k$ largest $\frac{\phi_i}{\rho_i}$ such that $\sum_{i \in k} \rho_i < PF - \sum r_{s1}$
4. Determine lower bound $Q_0 = \sum \phi_i + \sum q_{s1}$
5. Compute scale factor $K = \varepsilon Q_0 / S$
6. Scale the quality values such that $q'_{sl} = \frac{q_{sl}}{K}$
7. For $q = 1$ to $2Q_0$ do
8.   For $s = 1$ to $S$ do
9.       If $s$ is 1, Compute $R(s,q)$ using equation (2a)
10.      Else, Compute $R(s,q)$ using equation (2b)
11. Backtrack table $R(s,q)$ to find the substreams $s^*$
12. Until all streams are allocated do
13.     Arrange substreams in ascending order of $\frac{B_s}{r_s}$
14.     Allocate $\sigma_s = \min \frac{B_s}{\tau r_s}$ frames to stream $s$
15.     Update $B_s = B_s + \sigma_s * F - \sigma_s \tau r_s$
16. If no valid allocation found do
17.     Find substream $(\hat{l},\hat{s})$ such that $q_{\hat{s}\hat{l}} = \min_{s \in S, l \in L} \{q_{sl}\}$
18.     Discard substream $(\hat{l},\hat{s})$
19.     Go to step 3
```

Fig. 3. The proposed Substream Selection Algorithm.

*2) Data allocation:* Once the substreams are selected, it remains to allocate them to the MBS data area such that the subscriber playback buffers do not overflow or underflow. We use a modified version of the *weighted round robin* algorithm to allocate data to frames. The weighted round robin has been used for scheduling constant bit rate traffic before [15]. However, for a variable bit rate stream the stream priorities are not static. We derive the priority of a stream based on its buffer level at the subscriber. At the beginning of the scheduling window, for a stream $s$ let the data rate of the selected substream be $r_s$ and the buffer level be $B_s$. Then stream $s$ is assigned priority $\frac{B_s}{r_s}$. A lower value of $\frac{B_s}{r_s}$ denotes higher priority. We also need to allocate the number of frames to a stream in the current round, that is, the length of the burst. The burst length is chosen such that none of the other streams suffer from starvation, nor does it cause overflow or underflow at the receiver buffer. For a stream $s$ the length of the burst is given by $\min\{\frac{B_s}{\tau r_s}\}$.

*3) Buffer State Validation:* After the schedule is constructed, we check if any buffer constraints are violated. This can be easily determined by verifying the buffer overflow and underflow constraints described in equations (1c) and (1d). If the buffer constraints are violated, the current substreams cannot be allocated within the current scheduling window. Hence, we reduce the problem size and re-compute substreams. The problem size is reduced by discarding the substream with minimum quality value among all substreams. This process is repeated until a feasible solution is found or none of the substreams is selected. We note that even though the scheduler is located at the base station, it is aware of the subscriber buffer size and stream data rates. From this information it can calculate the change in buffer states for a given schedule without any involvement from the subscribers.

## C. Correctness and Performance Analysis

We first prove in Lemma 1 that the data rate and quality values of substreams of a scalable stream constitute a non-dominated set. In Lemma 2, we prove the correctness of the recursive formulation described in equations(2a)-(2b). Using Lemma 1 and Lemma 2 we prove the correctness of SSA.

*Lemma 1:* Data rates and quality values of substreams extracted from scalable streams constitute non-dominated set.

Let $l$ and $l'$ be two substreams of a given stream $s$. Substream $l$ is said to be dominated by substream $l'$ if including $l'$ in the solution always leads to better quality than including substream $l$. For example, let $r_{sl}, r'_{sl}$ be the data rates and $q_{sl}, q'_{sl}$ be the quality values of substreams $l$ and $l'$. If $r_{sl} > r'_{sl}$ and $q_{sl} < q'_{sl}$ then $l$ is dominated by $l'$. Greet et al. [16] have shown that, for H.264 PSNR scalability, when there is sufficient variability in the video, its rate-distortion characterization is close to a quadratic function which is convex. In our problem, since the streams are variable bit rate videos and layer encoded, the data rate and quality value pair of the layers within a stream can be assumed to form a convex set. This means that our problem instances are already in non-dominated form and we can easily solve the linear relaxation of the best quality substream selection problem. Efficient solution to the linear relaxation will help us in efficiently computing the final solution value. We empirically validate this assumption of rate variability and convexity. In Figure 4a we plot the sizes of frames of one of our test streams to show the high data rate variability. In Figure 4b we plot the data rate and PSNR values of three test streams and validate that they indeed form a convex envelop.

*Lemma 2:* The recurrence relations described in equations (2a)-(2c) produce a near optimal substream selection solution.

*Proof:* According to Lemma 1, all instances consist of only non-dominated substreams. Thus we only need to prove the correctness of the recurrence relation. We can prove the correctness of the recursive expression by induction. The basis step where $s = 1$ is true since it will lead to the selection of the maximum quality substream such that the data rate is less than the scheduling window capacity. Now let us assume that it is also true for the case of $s-1$ streams. For stream $s$ the expression $R(s-1, q-q_{sl})$ retrieves the weight of the solution and updates it by adding the current data rate. Then all such data rates are compared which can result in quality $q$ and only the minimum is chosen. Since $R(s-1, q)$ is already minimum, this results in $R(s, q)$ also being minimum for every quality value. Also, since data rates are accumulated only between two different streams only one substream from each stream is selected for all quality values. The solution is not optimal since the quality values are scaled. ■

*Theorem 2 (Correctness):* SSA algorithm described in Figure 3 returns a valid solution for the Substream Selection Problem.

*Proof:* By Lemma 2, the solution to the dynamic programming formulation selects substreams such that the average quality is close to the optimal and the total data requirement is less than the scheduling window capacity. Therefore, the capacity constraint and the *at most one substream per stream* constraint are satisfied. The round robin algorithm assigns MBS data areas to streams one frame at a time. Thus, it guarantees that no two frame bursts are assigned to the same frame. Finally, the buffer state validation step of the algorithm ensures no buffer overflow or underflow instances occur in the schedule. Hence, the SSA algorithm generates a valid solution for the substream selection problem. ■

Next we analyze the approximation factor and time complexity. Let $\Pi$ be a maximization problem with objective function $z$, optimal solution $S^*$ and optimal objective function value $z^*$. Let $I$ be an instance of $\Pi$ and $\varepsilon > 0$ be an error parameter. $A'$ is an approximation algorithm for the problem $\Pi$ if for the input pair $(I, \varepsilon)$, $A'$ outputs a solution $S'$ with an objective function value $z$ such that $z \geq (1-\varepsilon)z^*$.

*Theorem 3 (Approximation Factor):* The SSA algorithm described in Figure 3 is a constant factor approximation algorithm for a constant $\varepsilon$.

*Proof:* Let us describe a scaled instance obtained by using quality values $q'_{sl}$ as $I'$. Now for all substreams we have $q_{sl} - K \leq Kq'_{sl} \leq q_{sl}$. Let $A^*$ be an optimal algorithm and $\Pi(I, A^*)$ the solution obtained by algorithm $A^*$ on all instances $I$. Therefore $\Pi(I : A^*) = Q^*$. Considering the data rates for all substreams, we have $Q^* - \Pi(I', A^*)K \leq SK$. Since $A$ is optimal in $I'$ we know that $\Pi(I', A) \geq \Pi(I', A^*)$ and applying the above inequality we have

$$K\Pi(I', A) \geq K\Pi(I', A^*) \geq Q^* - SK.$$

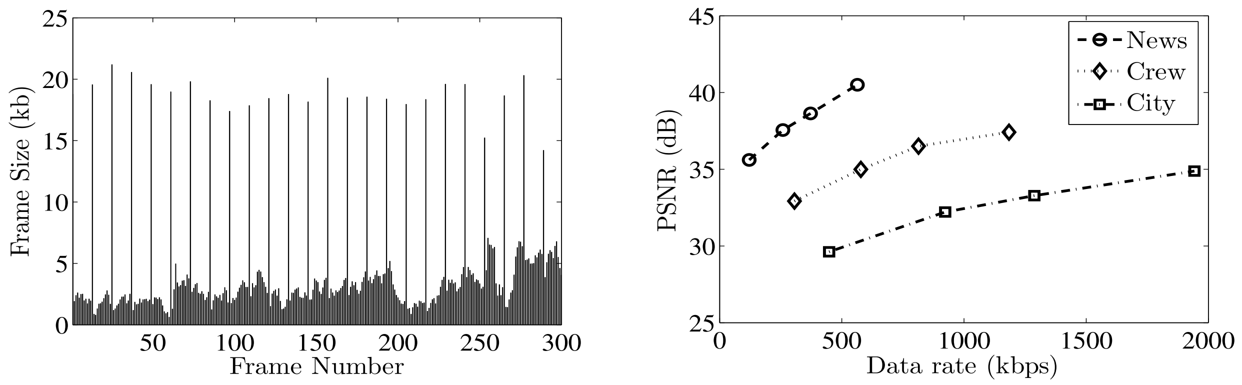
From properties of $q_{sl}$, we also know for any feasible solution $T$ that, $\Pi(I, T) \geq \Pi(I', T)K$. We thus have:

$$\Pi(I, A) \geq \Pi(I', A^*)K \geq Q^* - SK = Q^* - \varepsilon Q_0.$$

Also the algorithm always returns a solution of value at least $Q_0$. Therefore, $\Pi(I, A) \geq Q^* - \varepsilon Q_0 \geq Q^* - \varepsilon\Pi(I, A)$, which gives us $\Pi(I, A) \geq \frac{1}{1+\varepsilon}Q^* \geq (1-\varepsilon)Q^*$. Thus algorithm SSA is a constant factor approximation algorithm for constant $\varepsilon$. ■

*Theorem 4 (Time and Space Complexity):* The SSA in Figure 3 has a time complexity of $O(\frac{nL}{\varepsilon} + Pn\log n)$, where $n = O(\sum L)$ is the total number of substreams, $L$ is the maximum number of substreams within a stream, $P$ is the scheduling window size, and $\varepsilon > 0$ is a small constant. The space complexity is $O(SQ^*)$.

*Proof:* The dynamic programming algorithm computes $S \times 2Q_0$ entries for constructing the $R(s, q)$ table. Computing each entry takes $O(L)$ time. Hence the table can be completely constructed in $O(L \cdot S \cdot 2Q_0)$ time or $O(nQ^*)$ time. Computation of $Q_0$ takes $O(n\log n)$ time, leading to a total time complexity of $O(n\log n + nQ^*)$. This is not polynomial in $n$ since the value of $Q^*$ may not be bounded polynomially in $n$. For a small



(a) Foreman sequence

(b) Non-dominating property of scalable videos

Fig. 4. Rate-Distortion characteristics of scalable video.

constant $\varepsilon$, let us select a scale factor $K$ and scaled quality value for each substream $q'_{sl}$ according to equations (3a)-(3b).

$$K = \frac{\varepsilon Q_0}{S} \tag{3a}$$

$$q'_{sl} = \lfloor \frac{q_{sl}}{K} \rfloor \tag{3b}$$

Since $Q^* \leq 2Q_0$ we have $\frac{Q^*}{K} \leq \frac{2L}{\varepsilon}$. Thus the table computation can now be computed in $O(\frac{nL}{\varepsilon})$ time. The round robin allocation takes $O(n\log n)$ time for sorting the buffer levels in each round. The number of rounds depends on the size of the scheduling window $P$. Hence the total time complexity is $O(\frac{nL}{\varepsilon} + Pn\log n)$. The storage of the table requires $O(SQ^*)$ space which includes the $O(Q^*)$ space for saving the back-tracking information for constructing the solution. ■

## V. Trace-driven Evaluation

### A. Simulation Setup

We have implemented a point-to-multipoint WiMAX multimedia broadcast simulator and evaluated our algorithm in it using actual scalable video traces. For the WiMAX network parameters we use the 16-QAM modulation scheme with 3/4 convolution turbo coding and 10MHz channel frequency width. Since each TDD frame is 5ms, for a one second scheduling window we will have to allocate data to 200 TDD frames. Also we assume that within each TDD frame we have an MBS data area of 50kb. This gives us a broadcast channel bandwidth of 10Mbps [17]. At the receiver side we assume a buffer limit of 512kb. For generating the video traffic we use 10 raw (YUV files in 4:2:0 format) video files from the video trace repository of Arizona State University [18]. For each video we generate a 10 minute workload by starting from a random initial frame and then repeating the frame sequences. Then we encode the videos into H.264/SVC format using the JSVM reference software version 9.18 [19]. We encode each stream into four PSNR scalable layers using the medium grain scalability (MGS) feature and tune the encoding parameters such that the substreams have average bit rate between 100kbps and 2.5Mbps. In Table II, we summarize the information of the data rates and quality values of each layer of the different video files.

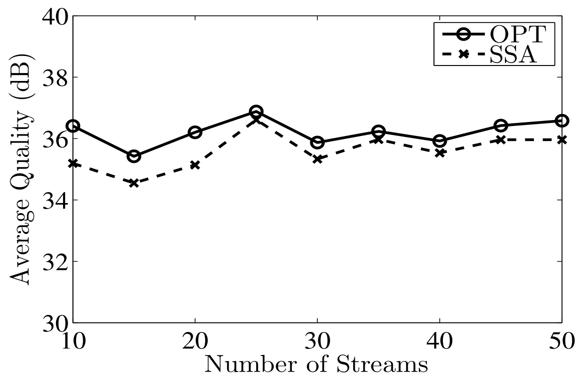
TABLE II
Data Rates (kbps) and PSNR (dB) Values of the Scalable Videos Used in Evaluation

| Name | 1 Layer | | 2 Layers | | 3 Layers | | 4 Layers | |
|---|---|---|---|---|---|---|---|---|
| | $r_1$ | $q_1$ | $r_2$ | $q_2$ | $r_3$ | $q_3$ | $r_4$ | $q_4$ |
| CREW | 306 | 32.92 | 578 | 34.99 | 814 | 36.5 | 1184 | 37.41 |
| FOOTBALL | 442 | 30.50 | 827 | 32.91 | 1114 | 33.98 | 1621 | 35.55 |
| MOBILE | 189 | 35.76 | 322 | 37.87 | 442 | 38.93 | 649 | 40.36 |
| CITY | 448 | 29.62 | 923 | 32.21 | 1288 | 33.28 | 1943 | 34.88 |
| FOREMAN | 170 | 32.9 | 407 | 34.86 | 589 | 36.0 | 890 | 37.43 |
| BUS | 185 | 33.17 | 390 | 35.43 | 567 | 36.41 | 857 | 37.65 |
| HARBOUR | 577 | 31.8 | 1025 | 33.46 | 1379 | 34.67 | 1929 | 36.25 |
| NEWS | 121 | 35.6 | 259 | 37.55 | 372 | 38.63 | 564 | 40.5 |
| SOCCER | 385 | 29.92 | 795 | 32.18 | 1095 | 33.13 | 1651 | 34.68 |
| ICE | 277 | 32.35 | 548 | 34.71 | 767 | 35.82 | 1123 | 37.32 |

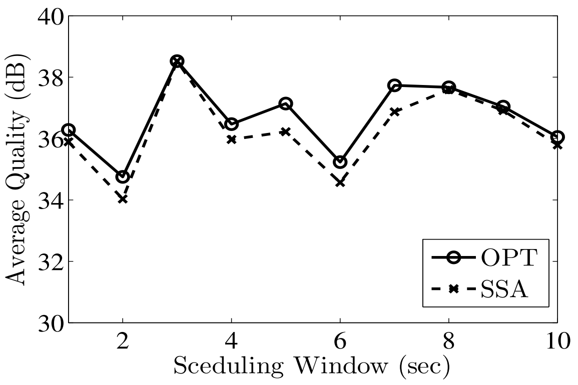
### B. Simulation Results

*1) Video Quality:* In our first experiment, we compare the performance of the SSA algorithm versus the optimum algorithm in terms of video quality. We perform this comparison over a period of 100 consecutive scheduling instances. We keep the receiver buffer size fixed at 512kb, the scheduling window size at 1 sec and vary the number of streams from 10 to 50. The results shown in Figure 5a show that our SSA algorithm produces near optimal solutions, which are less than 1dB from the absolute optimal solutions computed using optimization software GLPK [20]. We also observe that the proposed SSA algorithm scales well when the number of streams increase. In another experiment we keep the number of streams fixed at 20 and vary the scheduling window from 1 to 10 sec. As we can see from the results of this experiment in Figure 5b, the solution quality improves as the scheduling window grows. These two experiments show that our algorithm produces close to optimal results and it scales well, which means that it can support large scale WiMAX streaming services.

*2) Time Efficiency:* We evaluate the running time of our algorithms by changing the problem size in two ways. First we keep the scheduling window capacity fixed and increase

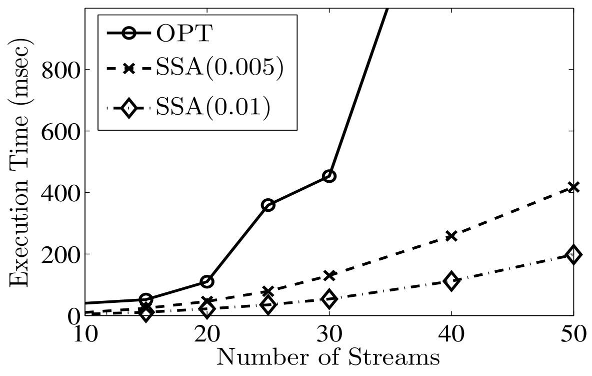


(a) Fixed window size at 1 sec

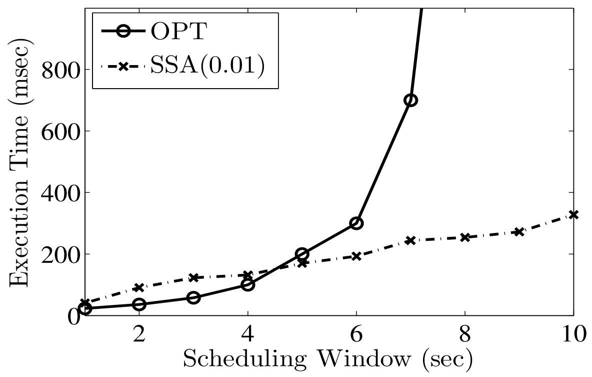


(b) Fixed number of streams at 20

Fig. 5. Near optimality of the solutions obtained by the SSA algorithm.



(a) Fixed window size at 1 sec

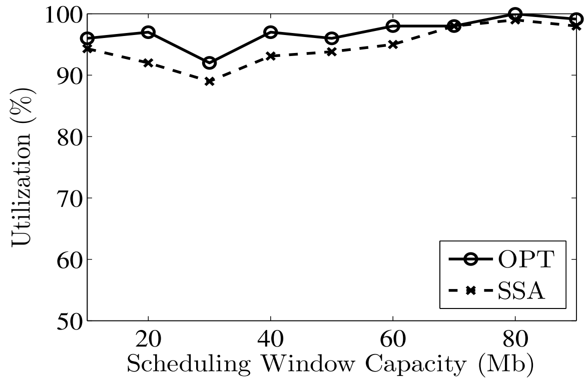


(b) Fixed number of streams at 20

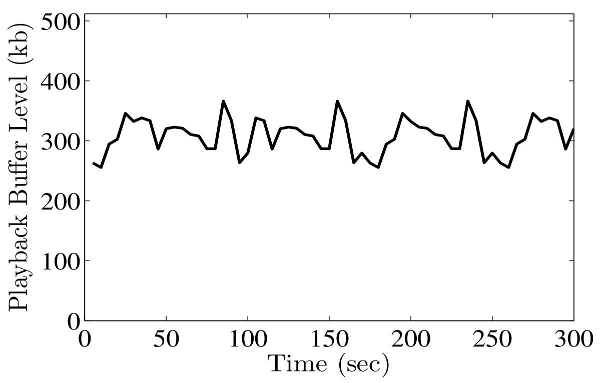
Fig. 6. Running time of the SSA and the optimal algorithms.

the number of streams. In a second experiment we keep the number of streams fixed and increase the scheduling window. We compare the execution times of our algorithms to that of the optimum. For deriving the optimum solution we first use the GLPK LP solver [20] to determine the substreams that can be scheduled and then sequence the frames within a scheduling window in a weighted round robin manner. For the SSA algorithm, we compute the running time with error parameters $\varepsilon = 0.01$ and $\varepsilon = 0.005$. For a one second scheduling window we vary the number of streams from 10 to 50 and observe their asymptotic behavior. The execution times of these algorithms are measured on a computer with 1.6GHz dual-core processor and 1GB of memory. The results of the first experiment are shown in Figure 6a. As expected, computing the optimum solution using GLPK takes much longer as the number of streams grow. The SSA algorithm runs well within the time window even as the problem instance sizes grow. For the second experiment we keep the number of streams fixed at 20 and vary the scheduling window size from 1 sec to 10 sec. From the results of the second experiment, depicted in Figure 6b, we can see that the SSA algorithm scales efficiently with increase in window size. For real time operation, the algorithm needs to compute the solution within the scheduling window duration. From Figure 6b, we see that with every one second increment in the scheduling window size the execution time increases by only a few milliseconds. This shows that the SSA algorithm scales well for large problem instances under real-time constraints.

*3) Resource Utilization and Buffer Validation:* Next, we evaluate the resource utilization of the SSA algorithm in terms of the scheduling window capacity used. Let the total schedulable data capacity of a scheduling window be $PF$. If $\{\bar{r_1}, \ldots, \bar{r_S}\}$ are the data rates of the chosen substreams, the total data sent in the schedule is $\sum P\tau\bar{r_s}$. The capacity utilization is then given by $\frac{\sum P\tau\bar{r_s}}{PF}$. As seen in Figure 7a, the resource utilization of the SSA algorithm remains close to optimal for different scheduling window capacity sizes. Finally, we verify that the buffer conditions are satisfied by the SSA algorithm. That is, we check if the SSA algorithm causes any buffer overflow or underflow. We found neither overflow nor underflow occurrences for any of the streams. In Figure 7b, we display the buffer level dynamics of subscribers receiving the stream *Foreman* for 300 consecutive scheduling windows, which show that the buffer level never exceeds 500kb, that is, there are no overflow instances. It also shows the the buffer level never goes below zero, which means there



(a) Resource utilization



(b) Receiver buffer dynamics

Fig. 7. Performance of the SSA Algorithm in terms of resource utilization and receiver buffer dynamics.

are no underflow instances. Similar results were obtained for subscribers receiving other video streams.

## VI. Conclusion and Future Work

We studied a framework for disseminating scalable video streams over mobile WiMAX networks. We mathematically analyzed the problem of selecting the optimal substreams of scalable video streams under bandwidth constraints. Solving this problem is important because it enables the network operator to transmit higher quality video or more number of video streams at the same capacity. We showed that substream selection problem in presence of bandwidth limitation is NP-Complete. We proposed a novel approximation algorithm for this problem. We proved that our algorithm has a small approximation factor of $(1-\varepsilon)$, and it has a time complexity of $O(\frac{nL}{\varepsilon})$, where $n = O(\sum L)$ is the total number of layers, and $L$ is maximum number of layers in a scalable stream. We implemented and validated our algorithm in a simulation setup and studied the impact of a wide range of parameters using multiple video traces. Our simulation results show that the approximation factor of the proposed algorithm is very close to one for practical scenarios. We also verified that our algorithm can run in real time and that it scales well to larger scheduling problems.

The work in this paper can be extended in different directions. For example, we are currently extending our algorithm to consider the battery constraints of mobile receivers such that, the produced solution not only optimize the video quality but also minimize the energy consumed by mobile receivers.

## References

[1] "IEEE 802.16 : Broadband Wireless Metropolitan Area Network ," 2009, http://standards.ieee.org/getieee802/802.16.html.

[2] "Yota Mobile WiMAX," 2009, http://www.yota.ru/en/info/main/.

[3] "UDCAST WiMAX TV," 2009, http://www.udcast.com/products/udcast_wimax_tv_products.htm.

[4] P. Hosein, "Broadcasting VBR traffic in a WiMAX network," in *Proc. of IEEE Vehicular Technology Conference (VTC'08)*, Calgary, Canada, September 2008, pp. 1–5.

[5] J. Wang, M. Venkatachalam, and Y. Fang, "System architecture and cross-layer optimization of video broadcast over WiMAX," *IEEE Journal on Selected Areas in Communications*, vol. 25, no. 4, pp. 712–721, May 2007.

[6] R. Cohen, L. Katzir, and R. Rizzi, "On the trade-off between energy and multicast efficiency in 802.16e-like mobile networks," *IEEE Transactions on Mobile Computing*, vol. 7, no. 3, pp. 346–357, March 2008.

[7] H. Juan, H. Huang, C. Huang, and T. Chiang, "Scalable video streaming over mobile WiMAX," in *Proc. of International Symposium on Circuits and Systems*, New Orleans, LA, May 2007, pp. 3463–3466.

[8] V. Reguant, F. Prats, R. de Pozuelo, F. Margalef, and G. Ubiergo, "Delivery of H.264 SVC/MDC streams over WiMAX and DVB-T networks," in *IEEE International Symposium on Consumer Electronics (ISCE'08)*, Algarve, Portugal, April 2008, pp. 1–4.

[9] M. Garey and D. Johnson, *Computers and Intractability: A guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.

[10] T. Wiegand, G. Sullivan, G. Bjøntegaard, and A. Luthra, "Overview of the H.264/AVC Video Coding Standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 560–576, July 2003.

[11] E. Lin, "A bibliographical survey on some well-known non-standard knapsack problems," *Information Systems and Operation Research*, vol. 36, pp. 274–317, November 1998.

[12] M. Bansal and V. Venkaiah, "Improved fully polynomial time approximation scheme for the 0-1 multiple-choice knapsack problem," International Institute of Information Technology, Tech. Rep., February 2004, available online at http://www.iiit.net/cgi-bin/techreports/display_detail.cgi?id=IIIT/TR/2004/3.

[13] A. Sinha and A. Zoltners, "The multiple-choice knapsack problem," *Operations Research*, vol. 27, no. 3, pp. 503–515, June 1979.

[14] E. Zemel, "An $O(n)$ Algorithm for the linear multiple choice knapsack problem and related problems," *Information Processing Letters*, vol. 18, no. 3, pp. 123–128, March 1984.

[15] P. Hosein and T. Gopal, "Radio resource management for broadcast services in OFDMA-based networks," in *Proc. of IEEE International Conference on Communications Workshops*, Beijing, China, May 2008, pp. 271–275.

[16] V. Geert, P. David, M. Reisslein, and L. Karam, "Traffic and quality characterization of the H.264/AVC scalable video coding extension," *Advances in MultiMedia*, vol. 2008, no. 2, pp. 1–27, January 2008.

[17] D. Gray, "Mobile WiMAX Part I - Overview and Performance," WiMAX Forum, White Paper, August 2006, available online at www.wimaxforum.org/technology/downloads/Mobile_WiMAX_Part1_Overview_and_Performance.pdf.

[18] "Arizona State University: Video Traces Research Group," 2009, http://trace.eas.asu.edu.

[19] "H.264 AVC/SVC Reference Software," 2009, http://ip.hhi.de/imagecom_G1/savce/downloads/SVC-Reference-Software.htm.

[20] "GNU Linear Programming Kit," 2009, http://www.gnu.org/software/glpk/glpk.html.