A General Optimal Video Smoothing Algorithm

Zhimei Jiang Leonard Kleinrock *
Department of Computer Science
University of California at Los Angeles
Los Angeles, CA 90024

Abstract

Video smoothing is a promising technique for reducing the bandwidth variability of video in order to improve network efficiency. This paper presents a general optimal video smoothing algorithm based on the concept of dynamic programming. The algorithm generates the optimum transmission schedule for different requirements by setting the constraints and the cost function accordingly. It can be used to study the smoothing of both stored video and real time video. In particular, for stored video, we show how the number of rate changes in the smoothed video is affected by the renegotiation cost and buffer size, assuming that the transmission rate is allowed to be lower than the reserved rate. For the real time system, we study the impact of various system parameters, including playout delay, client buffer size, and server buffer size, on the performance of video smoothing.

1 Introduction

Video transmission has posed some most challenging problems to the network design largely because of its high bandwidth demand, the QoS requirements, and the significant rate variability. Furthermore, the burstiness of video can often span multiple time scales and exhibits self-similarity in some cases [6]. This makes it even harder and more expensive to provide the QoS guarantees requested. Video smoothing is a natural approach for reducing the bandwidth variability of video. It has been shown to be effective for improving the network utilization significantly [14]. Since video smoothing changes the characteristics of the video streams transmitted through the network, different resource management mechanisms are required to handle the smoothed video. Zhang et al. investigated the impact of video smoothing on statistical multiplexing and call admission control [14].

In this paper, we are interested in the transmission schedule, which is simply the sequence of transmission rates in the channel for delivering video. The simplest video smoothing algorithms delay the playout time for just a few frames, or in the case of MPEG video, average the frames within a PBB or IBB window [8, 9, 12]. These algorithms introduce small initial delays, but the reduction of the bandwidth variability is very limited. For many multimedia applications, such as video on demand, digital library, etc., the video is stored at the server and a startup delay of up to a few minutes can often be tolerated. This allows us to use a relatively large buffer to substantially reduce the bandwidth variability of video. Several smoothing algorithms have been proposed for such systems to find the optimum transmission schedule according to different criteria, for example minimizing the peak rate, the number of rate changes, or the rate variance, etc [4, 5, 10, 13]. More about these algorithms will be discussed in section 3. Although simple and efficient for the problems above, they are hard to extend to solve systems with more complicated requirements or tradeoffs.

In this paper, we introduce a general optimal video smoothing algorithm, which can generate the optimum transmission schedule for any requirement that can be expressed as an additive cost function over the frames and some constraints to the system parameters, such as rate, buffer size, etc. The algorithm is based on the concept of dynamic programming. We will describe the algorithm in detail and show how to set the cost function and the constraints to solve some specific problems. In particular, for stored video, we are going to discuss how to reduce the number of renegotiations by allowing the transmission rate to be lower than the reserved rate as well as its effect on the smoothing buffer requirement. The same algorithm can also be employed for studying another kind of system in which a relatively long startup delay is still tolerable, while the video frames are generated in real time. One such application is live broadcasting. The efficiency of this kind of system depends on the initial playout delay as well as the buffer size at the client and the server. We show how the performance of smoothing is affected by these parameters.

The paper is organized as follows. In the next section, we describe the transmission systems for stored video and real time video. The general algorithm is developed in section 3. In section 4 and 5, the algorithm is applied to stored video and real time video respectively, to study the performance of smoothing under different system configurations.

Video smoothing adjusts the transmission rate in compliance with the buffer or delay constraints *after* the frames are generated. Similar efforts have also been made at the encoding stage. For example, given the delay, buffer, and network transmission rate constraints, algorithms have been developed to select the encoding quantizer for each frame to

*This work was supported by the Advanced Research Projects Agency, ARPA/CSTO, under Contract DABT-63-94-C-0080 "Transparent Virtual Mobile Environment".

0-7803-4383-2/98/$10.00 © 1998 IEEE.

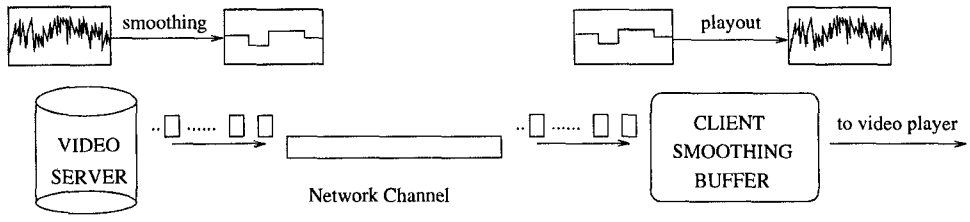


Figure 1: *The basic system structure for the transmission of stored video*

maximize video quality in a CBR or VBR channel environment without violating the constraints [7, 11]. Here the VBR channel environment refers to a system in which the rate is constrained by Leaky Bucket or other policing mechanisms. These algorithms are also based on dynamic programming.

2 Video transmission systems

In this section, we describe two video transmission systems: stored video and real time video. In both cases, we assume that transmission and playout take place in time slots of constant length.

2.1 Stored video transmission system

In this system, the video is stored at the server, and it is transmitted to the client site through the network upon request as shown in figure 1. At connection setup time, a certain amount of bandwidth is reserved. During the session, the bandwidth may or may not be renegotiated depending on the system. A smoothing buffer is set up at the client site. Frames arriving early are stored in the buffer to be played back later. The transmission rate through the network channel must be adjusted such that the smoothing buffer will never overflow or underflow. (The buffer underflows when a frame fails to arrive before its playout time.) More specifically, assume the video has N frames, the size of the smoothing buffer is B_{client}(bits), and initially the buffer is preloaded with B_{init}(bits). In addition, assume the playout starts at time slot 1 and after that exactly one frame is played at the end of each time slot. Denote the transmission rate in slot i by r_i(bits/timeslot). Then in order to avoid overflow and underflow, the sequence of transmission rates $r_1, r_2, ..., r_N$, which we refer to as the *transmission schedule*, must satisfy

$$\sum_{i=1}^{k} s_i - B_{init} \leq \sum_{i=1}^{k} r_i \leq \sum_{i=1}^{k} s_i - B_{init} + B_{client} \quad (1)$$

for all k $(1 \leq k \leq N)$, where s_i(bits) is the size of frame i. We call this the *buffer constraint* for transmitting stored video.

2.2 Real time video transmission system

For some non-interactive applications involving real time video, for example live broadcasting, the video stream can still be smoothed using buffer by delaying the playout for a certain period of time. In this case, the storage media at the server site shown in figure 1 is replaced by a buffer connected to the output of a video encoder (not shown in the figure). New frames from the encoder are stored in the server buffer temporarily before being transmitted to the client. Clearly, without a server buffer, the newly generated frames must be sent out immediately, and no smoothing would be possible at all.

Assuming the playout delay is N_{start} time slots, then during the session each frame will not be generated until N_{start} slots before its playout time. Because of this, the smoothing can only be carried out over a certain number of frames determined by N_{start}. And it is generally impossible to find the optimum transmission schedule in real time while the frames are being generated, because the sizes of the future frames are unknown. However, we do have the luxury of examining the complete trace of the video after it is finished. We may then obtain what would have been the optimum transmission schedule, and we can use it to study the impact of different system configurations on the performance of video smoothing. In addition, this evaluation could be used to compare our optimum (but unachievable) result to that of heuristic algorithms that can be realized.

For the real time video transmission system, in addition to the client buffer constraint given in (1), we must add an additional constraint to prevent server buffer from overflowing or underflowing. (The server buffer underflows when it tries to send more data than what it actually contains.) Assume one frame is generated at the beginning of each time slot and one is played at the end of each time slot once the playout has started. Moreover, assume the playout starts at time slot 1, at which time N_{start} frames have been generated. The constraint for the server buffer is, for any k $(1 \leq k \leq N)$

$$\sum_{N_{start}+1}^{N_{start}+k} s_i + \sum_{1}^{N_{start}} s_i - B_{init} - B_{server} \leq \sum_{1}^{k} r_i \leq \sum_{N_{start}+1}^{N_{start}+k} s_i + \sum_{1}^{N_{start}} s_i - B_{init} \quad (2)$$

where B_{server} is the server buffer size, B_{init} is the buffer occupancy at the client at time slot 1, thus $\sum_1^{N_{start}} s_i - B_{init}$ is the buffer occupancy at the server at time slot 1. Equations (1) and (2), which collectively form the *buffer constraint* for transmitting real time video, must both be satisfied in this system.

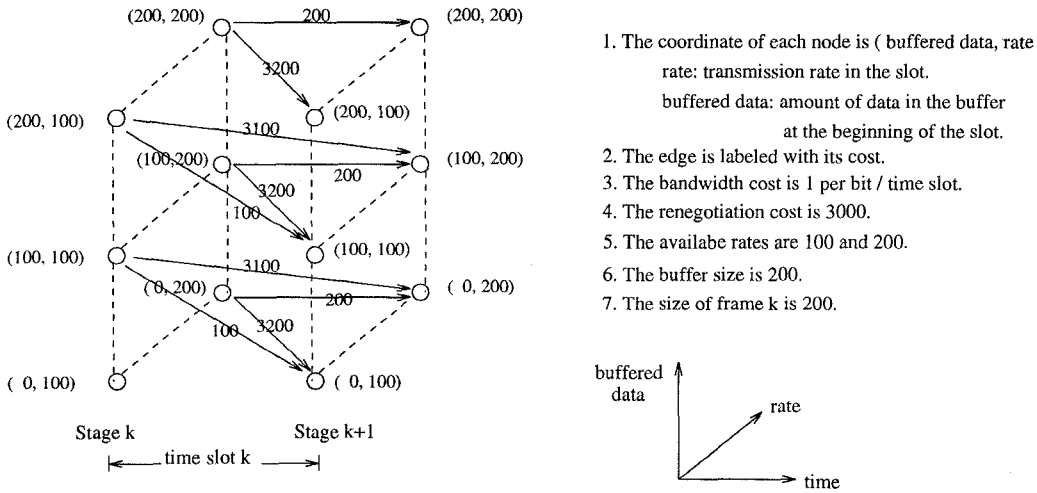


Figure 2: *An example of transitions between states in two consecutive stages. (stored video)*

3 A general optimal smoothing algorithm

For video smoothing, users usually have certain objective functions they wish to optimize. For example, if the same bandwidth is reserved for the whole session, they may want to minimize the peak transmission rate or the rate variability of the smoothed video [5, 13]. In a system where bandwidth can be renegotiated during the transmission, it is usually desirable to minimize the number of rate changes, i.e. bandwidth renegotiations [2, 5]. For a video on demand system, it is important to support the VCR-functionalities such as rewind, fast-forward etc., which can be implemented more efficiently if the requirements on the smoothing buffer and the buffer occupancy are minimized [4]. Our goal is to find the transmission schedule which is optimal according to the user's objective functions, subject to the buffer constraint.

All the existing optimal smoothing algorithms are for stored video only [2, 4, 5, 13]. The common feature of these algorithms is that, at each run, the transmission rate that can last for the longest time without causing buffer overflow or underflow is used. They differ in how the starting point for the next run is chosen after the rate for the previous run has been determined. McManus et al. took a different approach. They assumed the whole session was divided into intervals and determined the transmission rate in each interval as well as the minimum buffer size required to avoid overflow [10]. [3] compares the performance of these algorithms in detail. Each of the algorithms above is designed to optimize the transmission schedule according to a specific requirement. In this section we present a general optimal video smoothing algorithm which can find the optimum transmission schedule for any requirement that can be represented by some constraints and an additive cost function over the frames. The transmission schedule optimization problem is shown to be equivalent to a shortest path problem, which in turn can be solved efficiently with dynamic programming technique.

Assume the playout starts at time slot 1. In time slot i, r_i bits are received and added into the smoothing buffer, and at the end of that slot, frame i is displayed (ie. s_i bits are removed from the buffer). The transmission rate r may only change at the beginning of each time slot. The goal of our algorithm is to determine the transmission rate in each time slot so that all the constraints are satisfied and the total cost of transmitting the video is minimized. The constraints as well as the definition of cost function depend on specific user requirements on the smoothed video. Some examples will be given after we have described the general algorithm. Clearly, all the constraints must include the buffer constraint for the type of the system being studied.

We now transform this transmission schedule optimization problem into a shortest path problem. Suppose the system provides M possible transmission rates $R_1, R_2, ..., R_M$. For simplicity, we say that the system is at stage k when it is at the *beginning* of time slot k. Denote the state of the system at stage k by $D_k(b, r)$, where b is the amount of data in the buffer and r is the transmission rate in time slot k. Denote by $G_k(b, r)$ the set of states which can be reached at stage $k+1$ from state $D_k(b, r)$ without violating any constraint. For example, if the only constraint in the system is the buffer constraint (1), then $G_k(b, r)$ is given by

$$G_k(b,r) = \begin{cases} \{\ D_{k+1}(b', r') \mid & where\ b' = b + r - s_k, \\ & r' \in \{R_1, R_2, ..., R_M\}\ \} \\ & for\ \ 0 \le b + r - s_k < B \\ \emptyset\ \ (\ overflow\ ) & for\ \ b + r - s_k > B \\ \emptyset\ \ (underflow) & for\ \ b + r - s_k < 0 \end{cases} \tag{3}$$

where s_k is the size of frame k, and B is the client buffer size. The transition cost from state $D_k(b, r)$ to state $D_{k+1}(b', r')$, denoted by $C^k_{(b,r)->(b',r')}$, is determined by the optimization requirement. Typically, it is a function of rate, buffer occupancy, etc. For $D_{k+1}(b', r') \notin G_k(b, r)$, the transition cost $C^k_{(b,r)->(b',r')}$ is set to infinity.

An example is given in figure 2 which shows the transitions from each state in stage k to the states in stage $k+1$ in a stored video transmission system with no other con-

straint. The system provides only two rates: 100 and 200. Suppose $s_k = 200$, then $G_k(200,100) = G_k(100,200) = \{D_{k+1}(100,200), D_{k+1}(100,100)\}$, according to expression (3). And $G_k(0,100) = \emptyset$, because the buffer underflows at stage $k+1$ following state $D_k(0,100)$. The transition cost here is defined as the sum of the bandwidth cost (1 cost unit per bit/timeslot) and the renegotiation cost (3000 cost units/renegotiation). Therefore, $C^k_{(200,100)->(100,100)} = 100$, while $C^k_{(200,100)->(100,200)} = 3100$ because the rate changes from 100 to 200 in the latter case.

For any state $D_k(b,r)$ at stage k that can be reached from the initial state through a valid transmission schedule, at least one sequence of rates $r_1, r_2, ..., r_{k-1}$ can be found such that the resulting buffer occupancy at stage k is b, $r_k = r$, and all the constraints are satisfied at each stage up to the kth stage. We can then extend the graph in figure 2 to N stages to construct a three dimensional graph (figure 3) based on the description above. The nodes in the graph correspond to reachable states for the given video, ie. node $D_k(b,r)$ represents state $D_k(b,r)$. All the nodes for the same stage are on the same plane and planes for different stages are parallel to each other. There is an edge connecting two nodes $D_k(b,r)$ and $D_{k+1}(b',r')$, if and only if $D_{k+1}(b',r') \in G_k(b,r)$ and the cost of this edge is equal to the transition cost from state $D_k(b,r)$ to $D_{k+1}(b',r')$. Clearly, only nodes on the adjacent planes may be connected by edges. Based on the construction of the graph, for a cost function which is additive over the frames, the problem of finding an optimum transmission schedule is equivalent to finding the shortest path in this graph from the source - representing the initial state, to the terminal point - representing the state at which all the frames have been transmitted [1]. The source is connected to each node on stage 1 that is given by $D_1(B_{init}, r_i)$ for $r_i \in \{R_1, R_2, ..., R_M\}$, while a node is connected to the terminal point if at that state the transmission has finished. All the edges from the source or to the terminal point are assigned cost zero.

This shortest path problem can be solved efficiently by means of forward search in dynamic programming [1]. The idea is based on that if node j is on the shortest path from node i to k, then the part of this path from i to j must also be a shortest path from i to j itself. Therefore, the algorithm starts with the *first* stage (which corresponds to the first frame in our case), finds the shortest path from the source to each node for the first stage, and then proceeds to find the shortest path from the source to each node for the second stage. In general, assume the shortest path from the source to each node for stage k has been found, then the shortest path to each node for stage $k+1$ is determined based on those to the nodes in stage k and the costs of the edges connecting them. More specifically, if we denote the cost of the shortest path from the source to node $D_k(b,r)$ and $D_{k+1}(b',r')$ by $C_k(b,r)$ and $C_{k+1}(b',r')$ respectively, $C_{k+1}(b',r')$ is computed in the following way

$$C_{k+1}(b',r') = min\{C_k(b,r) + C^k_{(b,r)->(b',r')} \mid \textit{for all nodes } D_k(b,r) \textit{ in stage } k \textit{ such that } D_{k+1}(b',r') \in G_k(b,r)\} \quad (4)$$

The algorithm continues in this way until the last stage is reached, and the shortest path from the source to the terminal point is obtained. The pseudo code of the algorithm is given in appendix.

For fixed buffer size B_{client} and fixed number of possible rates M, the computation time of the algorithm is linear to the number of frames in the video. Even though the basic algorithm can be sped up substantially in various ways [1], it is still not fast enough to be used in the real system for large number of frames. However, the results from this algorithm can give us valuable insight into the problem and may be used as performance measurement for heuristic algorithms. In the next two sections, we apply this algorithm to stored video and real time video respectively.

4 Applications to stored video

In the algorithm described above, the cost function and constraints are not specified. In this section, we show how to define them in order to find the optimum transmission schedules for different user requirements in stored video transmission system.

4.1 Minimize number of rate renegotiations

To minimize the number of rate renegotiations, the constraint is simply the buffer constraint given by (1). Here, we assume that the transmission rate must be the same as the reserved rate. Otherwise, we can always achieve zero renegotiations by reserving the peak rate for the whole session.

Assume the cost of each rate change is C_{nego}, the transition cost $C^k_{(b,r)->(b',r')}$ is defined as

$$\begin{aligned} &\text{If } (\ r' \neq r\ ) \\ &\quad C^k_{(b,r)->(b',r')} = C_{nego} \\ &\text{Else} \\ &\quad C^k_{(b,r)->(b',r')} = 0 \\ &\text{EndIf} \end{aligned} \quad (5)$$

If let H be the number of rate changes in a path, the cost of the path is $H \cdot C_{nego}$. Therefore, the path with the lowest cost is the one with minimum number of rate changes.

4.2 Minimize peak rate

The constraint required in this case is still just the buffer constraint (1). For any k ($1 \leq k \leq N$), if $D_{k+1}(b',r') \in G_k(b,r)$, the transition cost from $D_k(b,r)$ to $D_{k+1}(b',r')$ is defined as

$$\begin{aligned} &\text{If } (\ r' > C_k(b,r)\ ) \\ &\quad C^k_{(b,r)->(b',r')} = r' - C_k(b,r) \\ &\text{Else} \\ &\quad C^k_{(b,r)->(b',r')} = 0 \\ &\text{EndIf} \end{aligned} \quad (6)$$

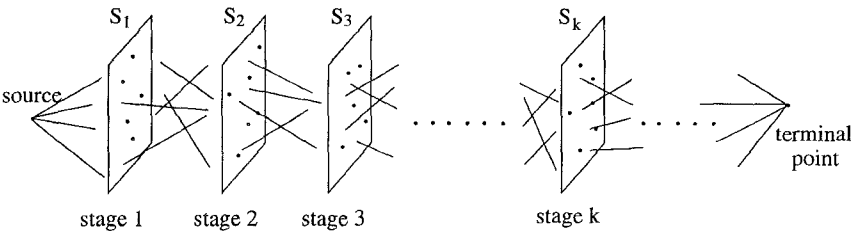


Figure 3: *The transmission schedule optimization problem is equivalent to finding the shortest path in the graph constructed based on the reachable states at each stage and transitions between states in two stages.*

Thus the cost of a path is equal to its peak rate. To find the transmission schedule with minimum peak rate *and* minimum number of rate changes, we can employ the cost function above and at the same time keep track of the number of rate changes in the best path found so far to each node. If two paths to the same node have the same cost, i.e. peak rate, the one with fewer rate changes is kept. Similarly, we can find the transmission schedule with minimum peak rate *and* minimum rate variance by keeping track of the rate variance along with the cost.

As we mentioned before, examples in section 4.1 and 4.2 may be solved more efficiently with the algorithms in [5] and [13]. However, both algorithms make the assumption that the available rates are continuous. For a system which provides a finite number of transmission rates, we expect their performance will depend on the distribution of the available rates.

4.3 Optimize the tradeoff between bandwidth usage and renegotiation

In section 4.1, we assumed that the actual transmission rate is equal to the reserved rate. In particular, the system is not allowed to transmit at a rate lower than the reserved rate. Therefore, if the buffer is about to overflow, a lower rate must be renegotiated. Sometimes this only means that the rate will have to be changed again after a short period of time. Since the total system cost is the sum of the bandwidth cost and the renegotiation cost, and typically rate renegotiation involves large system overhead, it might be more cost-effective to allow the transmission rate to be lower than the reserved rate, which in turn reduces the number of bandwidth renegotiations. In this section, we examine the system in which a transmission rate lower than the reserved rate may be employed if and only if transmitting at the reserved rate will cause overflow in the next time slot.

We now show how to find the transmission schedule with minimum total cost in such systems. To solve this problem, we need to modify the algorithm slightly to represent the state at each stage by the buffer occupancy and the reserved rate, rather than the transmission rate. The real transmission rates in the path, ie. the transmission schedule, are still recorded. And the rest of the algorithm is left unchanged. Assume the cost of reserved bandwidth is 1 cost unit per bit/timeslot, and the cost of each bandwidth renegotiation is C_{nego}. To find the transmission schedule with minimum total cost, the only constraint needed is the buffer constraint (1). However, since the transmission rate is allowed to be lower than the reserved rate, by definition, $G_k(b,r)$ becomes

$$G_k(b,r) = \begin{cases} \{\, D_{k+1}(b',r') \mid & where\ b' = b + r - s_k, \\ & r' \in \{R_1, R_2, ..., R_M\}\ \} \\ & for\ \ 0 \le b + r - s_k < B \\ \{\, D_{k+1}(B,r') \mid & where\ r' \in \{R_1, R_2, ..., R_M\}\ \} \\ & for\ \ b + r - s_k > B \\ \emptyset & for\ \ b + r - s_k < 0 \end{cases} \tag{7}$$

where r and r' are reserved rates instead of transmission rates, as we mentioned above. Formula (7) ensures that, only if transmitting at the reserved rate is going to cause overflow in the next time slot, ie. $b + r - s_k > B$, may a lower transmission rate equals $B + s_k - b$, be used. Otherwise, the transmission rate is equal to the reserved rate. For any $D_{k+1}(b',r') \in G_k(b,r)$, the transition cost $C_{(b,r)->(b',r')}$ is defined as

$$\begin{array}{l} \text{If } (\ r' \neq r\ ) \\ \quad C^k_{(b,r)->(b',r')} = C_{nego} + r \\ \text{Else} \\ \quad C^k_{(b,r)->(b',r')} = r \\ \text{EndIf} \end{array} \tag{8}$$

That is, the cost of a path is equal to its total bandwidth cost plus the total renegotiation cost.

The test results in figure 4 demonstrate how the number of rate renegotiations changes with the renegotiation cost for different smoothing buffer sizes. The test was conducted using a trace from movie Terminator II, which includes 40000 frames and has an average frame size of 10906 bits. Essentially, the cost of a bandwidth renegotiation (C_{nego}) determines the minimum number of time slots that a new rate should last in order to achieve a lower cost by switching to this new rate. For instance, if $C_{nego} = 10^6$, then decreasing the rate by 1000 for a period shorter than 1000 slots will cost more than keeping the original rate or combining the change with the next one. With a large smoothing buffer, each rate in the smoothed video tends to last a long period of time; as a result, the number of renegotiations is hardly affected by the renegotiation cost, as shown in figure 4 for buffer size $2 * 10^6$ and $3 * 10^6$. However, for a smaller buffer, the number of renegotiations decreases significantly as the renegotiation cost C_{nego} increases. More importantly, figure 4 shows that, as the renegotiation cost increases, the number of rate changes for different buffer sizes tends to converge. It

indicates that, for a system with a high renegotiation cost, we may use a relatively small buffer to achieve the performance close to that of a much larger buffer. This result is particularly useful for system which has a small buffer, or requires a small initial delay or a small buffer occupancy.

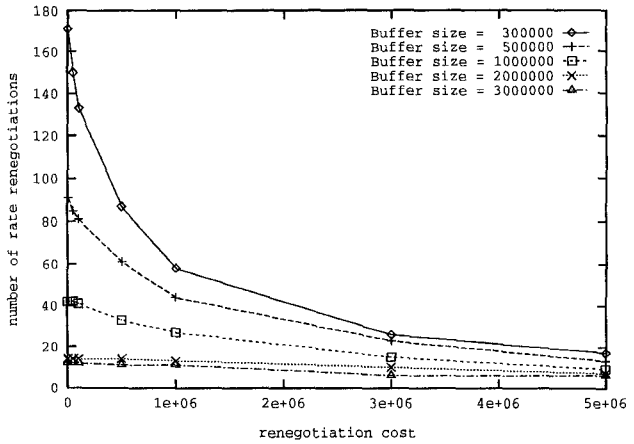


Figure 4: *Number of bandwidth renegotiations vs. renegotiation cost.*

5 Application to real time video

In this section, we use our optimal smoothing algorithm to study real time video transmission system, especially the case of minimum peak rate. In general, for the same optimization requirement, the same cost function can be employed for both stored video and real time video. The only difference for the two transmission systems is the buffer constraint that must be satisfied. Therefore, for the real time system, the transmission schedule with minimum peak rate can be obtained with the cost function (6) and the constraints given by (1) and (2).

We now examine how the minimum peak rate is affected by the playout delay and the size of client and server buffer. Again, the trace file of Terminator II is used in all the tests. Note that, although the complete trace is needed to obtain the optimum transmission schedule, the server buffer constraint (2) ensures that frame k is not available for smoothing until stage $k - N_{start} + 1$ in the algorithm. Denote the total buffer size at the client and server by B_{total}, ie. $B_{total} = B_{server} + B_{client}$. Once the playout has started, there are exactly N_{start} frames in the client and server buffer combined at any time, until all the frames have been generated. Because of this, B_{total} must be no less than the maximum of the summations of the frame sizes over a window of size N_{start}. Table 1 gives the minimum total buffer sizes required for several different playout delays (N_{start}).

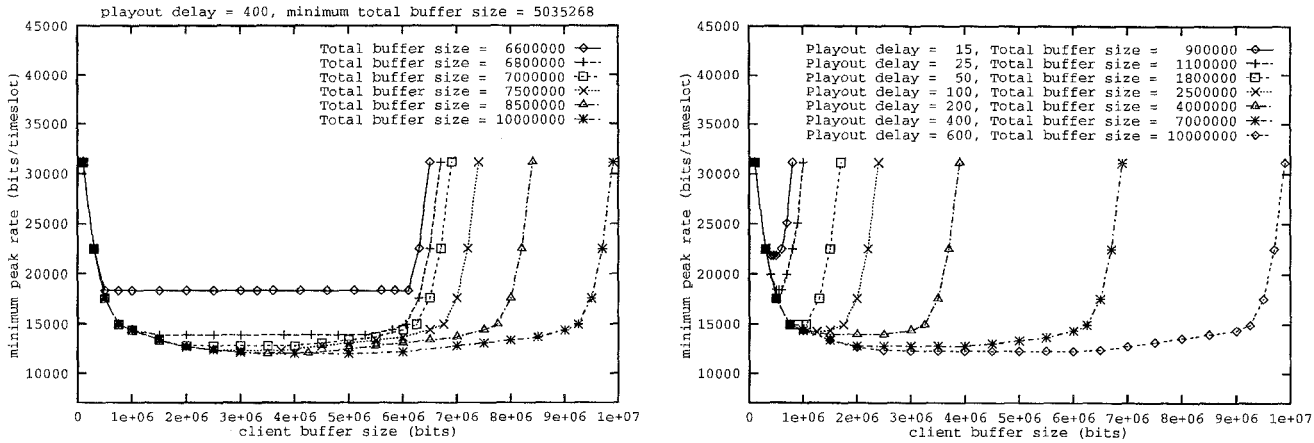
The first thing we investigate is, for a fixed total buffer size B_{total}, how the change in B_{client}, hence B_{server}, affects the minimum peak transmission rate. Figure 5(a) shows the results for several different values of B_{total} while keeping N_{start} at 400. An important observation is that each curve is almost symmetric about the line given by $B_{client} = B_{total}/2$. In other words, if two systems (1 and 2) have the same B_{total} and $B_{client1} = B_{server2}$, hence $B_{client2} = B_{server1}$, then the minimum peak rates for the two systems are about the same. This is because the process of transmitting a video through the *second* system above is equivalent to that of transmitting the original video in the reverse order, ie. the first frame as the last frame, etc., through the *first* system. Since in general the original video and its reverse have very similar characteristics in terms of the distribution of frame sizes and peaks, the minimum peak rate for the two systems should be very close.

Another important point shown in 5(a) is that, for a given N_{start} and two systems with different B_{total}, if the smaller buffer in each system is the same size, for example $B_{client1} < B_{server1}$, $B_{server2} < B_{client2}$, and $B_{client1} = B_{server2}$, then the minimum peak rates in the two systems are about the same, provided that the other two buffers are sufficiently large. An example in figure 5(a) is that, when B_{client} or B_{server} ($= B_{total} - B_{client}$) equals $3 * 10^5$, the corresponding minimum peak rate on all the curves is about 22500. It is also true when B_{client} or B_{server} equals 10^6 except for $B_{total} = 6.6 * 10^6$, in which case the other buffer is not large enough to achieve a lower minimum peak rate. The reason for this phenomenon is, if one buffer is relatively small compared to the minimum total buffer size required for the given N_{start}, and the other buffer is sufficiently larger, the transmission rate is determined by the small buffer rather than the playout delay. In fact, the same result holds even for different playout delays as shown in figure 5(b), where 7 different values of N_{start} ranging from 25 to 600 were tested and for each curve B_{total} was fixed. For example, again at B_{client} or $B_{server} = 3 * 10^5$, the minimum peak rate is about 22500 for all 7 different playout delays. It indicates that, when designing a system, if one buffer is relatively small, simply increasing the playout delay or the size of the other buffer will not further improve the performance, once it reaches the limit determined by the small buffer.

On the other hand, for given N_{start} and B_{total}, as long as neither buffer is very small, the system will operate at a rate close to or equal to the lowest minimum peak rate determined by N_{start} and B_{total}. This is supported by the fact that, for each curve in figure 5, 6, and 7, as either buffer increases from zero, the peak rate quickly reaches a value close to that of the lowest point on the curve. Figure 6 illustrates, for a fixed B_{client}, how the minimum peak rate changes as B_{server} increases when N_{start} equals 400, where B_{server} is chosen such that B_{total} is greater than or equal to $6.6 * 10^6$. Clearly, for the case with relatively large B_{client}, the lowest minimum peak rate is determined by the playout delay of 400. For small B_{client}, this peak rate is limited by the client buffer size as we discussed before. Figure 7 shows how the playout delay affects the relation between the minimum peak rate and the client buffer size, for a total buffer size $B_{total} = 10^7$. Obviously, the playout delay N_{start} should be sufficiently large in order to take full advantage of the available buffer. However, choosing an N_{start} which requires a minimum total buffer size that is close to B_{total}, is not ideal either. This not only increases the initial startup

playout delay	15	25	50	100	200	300	400	500	600
minimum total buffer size	535543	771137	1325002	2114692	3590920	5035268	6538488	8167940	9296832

Table 1: *Minimum total buffer size required for different playout delay N_{start}.*



(a) for playout delay fixed at 400 (b) for different playout delays

Figure 5: *Minimum peak rate vs client buffer size for a given total buffer size*

delay, but also requires larger peak rates, as demonstrated in figure 7 in which the minimum peak rates for N_{start} of 400 and 500 are both lower than that for 600. Comparing the values in table 1 and figure 7, it appears that choosing an N_{start} whose minimum buffer requirement is about $2 * 10^6$ to $3 * 10^6$ less than B_{total} achieves better results, given that neither B_{client} nor B_{server} is very small. These numbers should change with the buffer size and the type of video.

In summary, the minimum peak transmission rate in the real time system is affected by all three system parameters, including the playout delay N_{start}, the server buffer size B_{server}, and the client buffer size B_{client}. If one or two of them are given, we may adjust the others to achieve lower minimum peak rate until it reaches the limit determined by the given buffer size or N_{start}. Similar studies can be conducted to examine the impact of these parameters on rate variance or the number of rate changes, etc.; we expect the results will be very similar to what we have obtained for the minimum peak transmission rate.

6 Conclusion

In this paper, we have presented a general optimal video smoothing algorithm based on the concept of dynamic programming. The algorithm can generate the optimal transmission schedule for any requirement that can be represented by the constraints on the system parameters and an additive cost function over the frames. It can be used to study the smoothing of both stored video and real time video. In fact, for the same optimization requirement, the same cost function can be employed for both systems. The only difference for the two systems is the buffer constraints that must be satisfied.

As an example, we showed how to set the cost function and the constraints in order to find the transmission schedule with minimum peak rate or minimum number of rate changes. For stored video transmission system in which the transmission rate is allowed to be lower than the reserved rate, we studied how the number of rate renegotiations is affected by the renegotiation cost. The main conclusion from this study is that, in a system with a high renegotiation cost, a small smoothing buffer can achieve the performance close to that of a much larger buffer. We also applied the algorithm to the real time video transmission system. The results show how the minimum peak transmission rate in the real time system is affected by three system parameters, including the playout delay, the server buffer size, and the client buffer size. Notably, if one buffer is relatively small compared to the minimum total buffer size for a given N_{start}, and the other buffer is sufficiently larger, then the transmission rate is determined by the small buffer, and it is independent of the playout delay. On the other hand, for fixed playout delay and total buffer size, as long as neither buffer is very small, the system will operate at a rate close to or equal to the lowest minimum peak rate determined by the given playout delay and total buffer size.

The algorithm can also be extented to find the optimum transmission schedule for smoothing in a VBR environment, in which the transmission rate is constrained by Leaky Bucket policy. However, for smoothed video, we expect the performance improvement will not be significant especially for systems with large smoothing buffers.

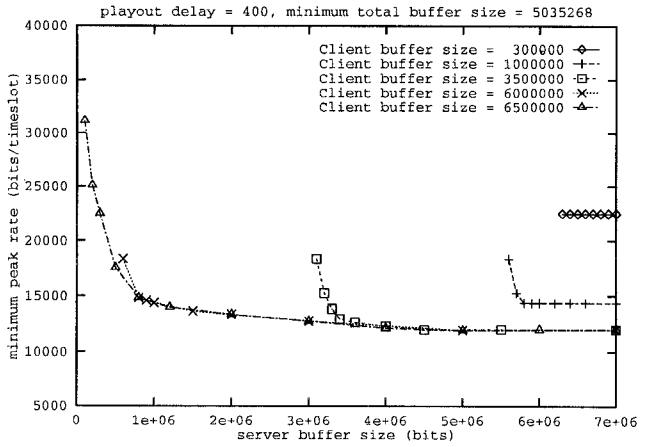


Figure 6: *Minimum peak rate as a function of server buffe r size for different client buffer sizes.* (N_{start} = 400)

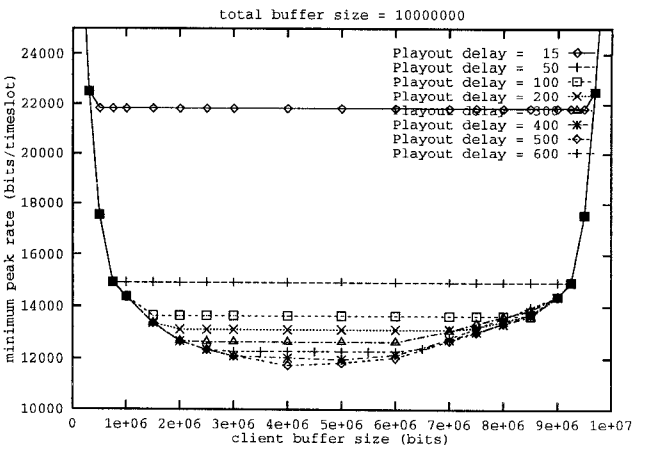


Figure 7: *Minimum peak rate as a function of client buffe r size for different playout delays.* (B_{total} = 10000000)

References

[1] D. P. Bertsekas, *Dynamic Programming: Deterministic and Stochastic Models*, Prentice-Hall Inc., Englewood Cliffs, N.J., 1987.

[2] K. Chang and H. T. Kung, "Efficient time-domain bandwidth allocation for Video-on-Demand system", *Proceedings of ICCCN '96,* Washington D.C, USA, Oct., 1996.

[3] W. Feng, J. Rexford , "A comparison of bandwidth smoothing techniques for the transmission of prerecorded compressed video", *IEEE INFOCOM '97 Proceedings,* Kobe, Japan, 1997, pp. 58-67.

[4] W. Feng, "Rate-Constrained Bandwidth Smoothing for the Delivery of Stored Video", *Proceedings of IS&T/SPIE Multimedia Networking and Computing 1997,* San Jose, CA, 1997, pp. 316-327.

[5] W. Feng, F. Jahanian, and S. Sechrest, "An Optimal Bandwidth Allocation Strategy for the Delivery of Compressed Prerecorded Video", *to appear in ACM/Springer-Verlag Multimedia Systems Journal.*

[6] M. Garrett and W. Willinger, "Analysis, modeling and generation of self-similar VBR video traffic", *Proceedings of ACM SIGCOMM,*, Boston, MA, August 1994, pp. 269-280.

[7] C. Y. Hsu, A. Orgeta, and A. R. Reibman, "Joint selecting of source and channel rate for VBR video transmission under ATM policing constraints", *to appear in IEEE Journal of Selected Areas in Communications.*

[8] K. Joseph and D. Reininger, "Source traffic smoothing and ATM network interfaces for VBR MPEG video encoders", *ICC '95 Proceedings*, Seattle, WA, USA, 1995, pp. 1761-1767.

[9] S. Jung and J. S. Meditch, "Adaptive prediction and smoothing of MPEG video in ATM networks", *ICC '95 Proceedings*, Seattle, WA, USA, 1995, pp. 832-836.

[10] J. M. McManus and K. W. Ross, "Video-on-demand over ATM: constant-rate transmission and transport", *IEEE Journal on Selected Areas in Communications,* vol.14, Aug. 1996, pp. 1087-1098.

[11] A. Ortega, K. Ramchandran, M. Vetterli, "Optimal buffer-constrained source quantization and fast approximations", *1992 IEEE International Symposium on Circuits and Systems Proceedings,* San Diego, CA, USA, 1992, pp. 192-195.

[12] T. Ott, T. V. Lakshman, and A. Tabatabai, "A scheme for smoothing delay-sensitive traffic offered to ATM networks", *IEEE INFOCOM '92 Proceedings,* Florence, Italy, 4-8 May, 1992, pp. 776-785.

[13] J. D. Salehi, Z. L. Zhang, J. Kurose, and D. Towsley, "Supporting stored video: reducing rate variability and end-to-end resource requirements through optimal smoothing", *Proceedings ACM SIGMETRICS,* Philadelphia, PA, May 1996, pp. 222-231.

[14] Z. L. Zhang, J. Kurose, J. D. Salehi and D. Towsley, "Smoothing, statistical multiplexing and call admission control for stored video", *to appear in IEEE Journal of Selected Areas in Communications.*

[15] Trace file of "Terminator II", from ftp://ftp-info3.informatik.uni-wuerzburg.de/pub/MPEG/. The frame sizes are in BITS. Pattern: IBBPBBPBBPBB, 25 frames/second.

A The dynamic programming algorithm for optimum video smoothing

Please refer to section 3 for the definition of the variables.

```
/* INITIALIZATION */
Minimum_Cost = MAX_INTEGER              /* Minimum_Cost is the cost of the shortest path found so far. */
For each possible transmission rate R_i
    If [ D_0(B_init, R_i) satisfies all the constraints ]
        Add D_0(B_init, R_i) to set G_1
        C_1(B_init, R_i) = 0                    /* Initial cost is zero. */
    Endif
EndFor
/* MAIN LOOP */
For k = 1 to N Do                               /* Process each frame. */
1   For each D_k(b, r) in G_k
        If [ at D_k(b, r), all the frames have been transmitted ]
            If ( C_k(b, r) <Minimum_Cost )
              Minimum_Cost = C_k(b, r)
            EndIf
            continue                            /* i.e. go to 1 */
        EndIf
        For each D_{k+1}(b', r') in G_k(b, r)
            If [ ( all the constraints are satisfied) AND (C_{k+1}(b', r') > C_k(b, r) + C^k_{(b,r)->(b',r')}) ]
              Add D_{k+1}(b', r') to set G_{k+1}        /* The set won't change if D_{k+1}(b', r') is already in it.*/
              C_{k+1}(b', r') = C_k(b, r) + C^k_{(b,r)->(b',r')}   /* The new path to state D_{k+1}(b', r') has lower cost. */
            EndIf
        EndFor
    EndFor
EndFor
```