Scheduling Algorithms for the Broadcast Delivery of Multimedia Products

V. Vinay Krithi Ramamritham *

Department of Computer Science and Engineering
Indian Institute of Technology, Bombay
Powai, Mumbai - 400076, India
E-mail: {krithi,vvinay}@cse.iitb.ernet.in

Abstract

Free bandwidth in television channels which is available in the form of the Vertical Blanking Interval (VBI) is currently being utilized to broadcast programme information, HTML pages and closed captioning. This bandwidth can be used to broadcast data and also products like video clips, software, multimedia packages, etc. thereby satisfying several customers with a single transmission at very high speeds. We present several heuristic algorithms that attempt to maximize the revenue earned by an E-commerce merchant providing such a service. These algorithms are tailored to deal with different customer service policies.

1. Introduction

As the reach and use of E-commerce is increasing by the day, new and faster means of product delivery are required to ensure higher levels of customer satisfaction. Products like music albums, movies, multimedia and software packages, etc. can be delivered through the internet by a multicast by batching requests of several customers. But the advantages of using television broadcast infrastructure for the purpose is immense considering the vast reach of television channels. In particular, multimedia products can be broadcast in the spare bandwidth available in a television channel.

Customers can be provided with a set of delivery options corresponding to the time of delivery : for example, the delivery cost for Product A is $4 if it is delivered within 4 hours from the time of request, $3 if it is delivered beyond 4 hours but within 5 hours from the requested time. The E-commerce merchant can also provide the option of free delivery beyond these times. The customer chooses desired products and a subset of delivery options. The cost to the customer is the cost based on the delivery times of the individual products. Clearly, the E-commerce merchant can batch several requests and serve them through a single broadcast of the product. The revenue to the merchant is the sum of the costs to the individual customers. Three algorithms for scheduling the broadcast of multimedia products such that the profit to the E-commerce merchant is maximized are described in [2]. In this paper we present new scheduling algorithms for the same optimization problem. Two of our algorithms yield significantly higher levels of profit, the third has the advantage that customers know precisely when the delivery will commence and hence can turn their set-top box on just in time.

In Section 2 we outline the broadcast model that could be used in such a service. Section 3 provides a detailed description of our algorithms and also the algorithms proposed by [2]. Performance of these algorithms is analyzed and illustrated by the simulation results in Section 4. In Section 5 we comment on some related work. We end with concluding remarks in Section 6.

2. The Broadcast Model

In the current television setup, there are several channels in which programmes are broadcast, with breaks in transmission at certain times. When normal programming is being broadcast, there is a small percentage of free bandwidth available in the form of the VBI. *Vertical Blanking Interval (VBI)* is the part of a television transmission signal that is blanked, or left clear of viewable content, to allow time for the television's electron gun to move from the bottom to the top of the screen as it scans images. VBI is currently being used to transmit programme information, HTML pages, closed captioning, teletext, etc. The following are some services which use VBI to broadcast internet content to home PCs - WaveTop, WebTV, DirecPC. Breaks in transmission leave the full channel to be used for other purposes. We assume that the bandwidth requirements of the television broadcast for conventional TV programmes is known for all channels. In other words the model is deterministic in

* Also affiliated to the Department of Computer Science, University of Massachusetts, Amherst, MA 01003. Supported in part by NSF Grant IRI-9619588.

that the free bandwidth at any point in time is known.

The process of product delivery is as follows. A customer connects to the E-commerce site and chooses the desired products and also specifies the delivery options for each product. The delivery options form a monotonically non-increasing function given that the delivery cost should only decrease with increased delay. The cost function for a product is the sum of the cost functions for all the customers requesting the product. Given a set of requests, the server at the E-commerce site prepares a schedule for broadcast and responds back to the customer with a token and possibly a time of delivery. The token is used for authorizing the customer to receive and decode the broadcast. The token is valid till the product delivery is over or until the largest of the deadline options is over, whichever is earliest. The schedule is not static, we allow for the re-ordering of previously scheduled products. The product being broadcast at any instant is retrieved from the storage server and is broadcast through one of the several television channels along with the television programme scheduled at that time. The product is received and stored in the set-top box attached to the customer's television set. We do not provide a detailed description of the actual mechanism for broadcast, but we concentrate on the heuristic algorithms that could be used in optimizing the profit in a broadcast product delivery service. The revenue earned by the merchant is the sum of the costs incurred by individual customers.

2.1. Customer Service Policies

The E-commerce merchant follows certain policies in serving customers. Possible policies which could be applied on a per-product basis include (but are not restricted to) the following ones :

1. Specify the exact time of delivery of the product. The customer can therefore know when to expect the product and hence knows the cost of delivery. This policy is very desirable from the customer's viewpoint.

2. Specify the deadline that will be guaranteed to be met. The actual broadcast may occur any time within this guaranteed deadline. The delivery cost depends on which of the deadlines before the guaranteed deadline included is met. For example, if a customer chooses product A and the options are ($4, 10hours) and ($3, 20hours), and he is given a guarantee of 20 hours; then the delivery cost can be either $4 or $3 depending on whether it is transmitted within 10 hours or 20 hours.

3. No guarantees are provided. The customer is not informed about probable delivery times. The cost to the customer is based on the actual time of delivery.

In this paper, we present algorithms corresponding to each of these policies. The algorithms proposed by [2] correspond to policy 1, but we evaluate the performance of all our algorithms against these algorithms.

2.2. Atomic and Segmented transmission

Products can be transmitted (we use the terms "transmission" and "broadcast" synonymously) atomically or in segments. Atomic transmission means that the product is transmitted from the start to finish without any interruptions. Since in atomic transmission the product is transmitted as a single segment without interruption, a (transmission) task corresponds to a product which is requested. In segmented (cyclic) transmission, each task is constituted by several subtasks, each representing a segment of the product to be transmitted. This is caused due to either preemption, or batching of a new request with an ongoing transmission. We consider only atomic transmission for our algorithms though we compare with the algorithms for segmented transmission of [2]. In segmented transmission, when a new request arrives, it can start receiving the ongoing broadcast of that product, but a subtask will be created which corresponds to the part missed out.

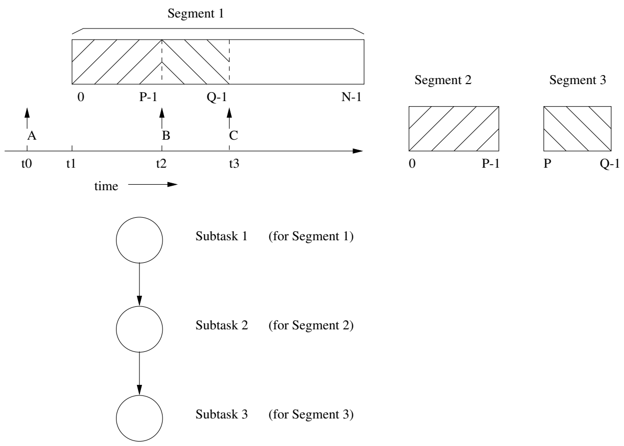

Figure 1. Subtasks corresponding to cyclic transmission

Fig. 1 illustrates the creation of subtasks. Customer A arrives at time t_0 during a period of non-transmission when there are no pending subtasks, hence a subtask (subtask 1) spanning the whole product (bytes 0 to N - 1) is spawned. Customer B arrives at time t_2 when the first P bytes have already been broadcast. B receives the remaining broadcast of subtask 1, but a new subtask is spawned covering the part missed out by B, i.e., bytes 0 to $P-1$. Similarly the subtask for Customer C is created corresponding to the segment between P and $Q-1$. The subtasks form a linear precedence structure because the product is considered as a cyclically glued segment.

The advantages of atomic broadcast are higher levels of batching and lower bandwidth requirements. But in the worst case scenario the product would have to be broadcast to each individual customer because a new customer cannot benefit from an ongoing broadcast. Segmented (cyclic) transmission does not suffer from this disadvantage. The customer does not have to wait if his requested product is currently being broadcast. But the scheduling overhead can be high if a large number of subtasks have to be managed.

3. Algorithms

Let the number of channels be N and the number of products be M. Let K, $1 \leq K \leq M$ be the number of tasks to be scheduled at the current scheduler invocation. Each of these tasks $T_1, T_2, \ldots, T_K$ has a set of associated (cost, deadline) pairs - $\{(c_{i1}, d_{i1}), (c_{i2}, d_{i2}), \ldots, (c_{in_i}, d_{in_i})\}$ representative of the heuristic step cost function for the product, n_i is the number of deadlines associated with the i^{th} task at the current scheduler invocation. Each channel C_j, $1 \leq j \leq N$ has a transmission queue of tasks. The scheduler can insert and re-order tasks in this queue.

In atomic transmission, the cost function associated with the task is the sum of the cost functions of individual requests for that product along time. In segmented transmission the cost function associated with a subtask is the sum of the cost functions for all customers whose reception of the product will be complete by broadcasting the segment represented by this subtask.

We now describe our algorithms in section 3.1, 3.2 and 3.3 followed by the algorithms proposed by [2] in section 3.5 and 3.6. Each of our algorithms uses a heuristic function H to order tasks. This heuristic function is based on some characteristic of the tasks and is described in section 3.4.

3.1. Algorithm 1 (Exact Time - ET)

This algorithm corresponds to customer service policy 1. ET schedules the task with highest H function value so that we gain the maximum profit. But since we are informing the customer about the exact timings of delivery, we cannot alter previously scheduled tasks. A new task is always inserted at the end of the transmission queue of the selected channel.

The pending tasks $T_1, \ldots, T_K$, are ordered based on their highest H function value. We continually choose the task with the highest H function value among all tasks and try to schedule it, till all tasks are exhausted. If a previous instance of a task exists in the schedule, we batch T_h with that task. By batching T_h with T_{prev}, we are saving on a second broadcast for the same product and also possibly increasing profits because we would be completing much earlier. d_{hj} is the j^{th} deadline ($1 \leq j \leq n_h$) of task T_h, and is the deadline corresponding to the current H function value on which T_h is ordered in RL, the *pending request list*. For each task that has been successfully scheduled, the finish time of the task can be sent to the customers requesting the corresponding product and having a deadline option that is greater than or equal to the finish time of the task. All other customers can be informed of the infeasibility of meeting their requests.

Algorithm ET(τ)
begin
  $RL \leftarrow$ tasks in τ ordered on their highest H function value
  while ($RL \neq \phi$)
    $T_h \leftarrow$ Task with highest H function value in RL
    if (T_{prev} exists) then
      Batch T_h with T_{prev}
      $RL \leftarrow RL - T_h$
    else
      Find the $earliest\_completion\_time$ for T_h using the finish times of the last tasks in each transmission queue
      $C_b \leftarrow$ Channel on which the transmission of T_h finishes earliest
      if ($earliest\_completion\_time \leq d_{hj}$) then
        Enqueue T_h at the end of the queue for C_b
        $RL \leftarrow RL - T_h$
      else
        if ($earliest\_completion\_time > d_{hn_h}$) then
          T_h cannot be guaranteed
          $RL \leftarrow RL - T_h$
        else
          Reinsert T_h in RL based on its new H function value
  endwhile
end

3.2. Algorithm 2 (Guaranteed Deadline - GD)

This algorithm corresponds to customer service policy 2. In the exact timings algorithm, we specify the start and finish time of transmission of every requested product. The delivery cost will be the same during any time in the range between the request arrival time and the first deadline, or between any two deadlines. Thus, if we could accommodate new requests while still meeting the deadlines of this and other previously scheduled tasks, it would be more profitable. GD tries to insert new tasks in an existing schedule by using the laxities in meeting task deadlines.

While scheduling the task T_h with the highest H function value, in each channel we find the first available slot large enough for transmitting $Sizeof(T_h)$ bytes. This slot can be between the finish time of a task and the latest start time[1] of the next task in the transmission queue. Among

[1] The *latest_start_time* of a task is the latest time at which the task should begin, failing which its deadline will be missed

these slots the one in which T_h completes earliest is chosen. The corresponding channel and completion time give the "best-channel" C_b and $earliest\_completion\_time$ for T_h. If a previous instance, T_{prev}, of a task exists, we batch T_h with T_{prev}. If the task fails to meet its deadline if allocated the chosen slot, we move on to its next deadline and re-insert it in the pending request list based on its next highest H function value. If we are unable to insert a new task in between the existing tasks in any of the transmission queues, this algorithm reduces to algorithm 1.

Algorithm GD(τ)

```
begin
  RL ← tasks in τ ordered on their highest H function value
  while (RL ≠ φ)
    T_h ← Task with highest H function value in RL
    Find the earliest_completion_time for T_h
    C_b ← Channel on which the transmission of T_h
          completes earliest
    slot ← time slot that results in the
          earliest_completion_time in C_b
    if (earliest_completion_time ≤ d_hj) then
      Allocate T_h in the transmission queue for C_b at
      time slot slot
      Assign the guaranteed deadline of T_h to be d_hj
      if (T_prev exists) then
        Batch T_prev with T_h
      RL ← RL − T_h
    else
      if (T_prev exists) and
        (d_hj is the guaranteed deadline for T_prev) then
        Batch T_h with T_prev
        RL ← RL − T_h
      else if (earliest_completion_time > d_hn_h) then
        T_h cannot be guaranteed
        RL ← RL − T_h
      else
        Reorder T_h in RL based on
        its next highest H function value
  endwhile
end
```

The latest start time of a task is calculated by taking $min(latest\_start\_time$ of next task in the transmission queue, $d_{hj})$. Note that no deadline of the tasks in the previous schedule is violated as a result of insertion. For all tasks that have been successfully scheduled, the customers who have deadline options greater than or equal to the guaranteed deadline for the task are informed of the guaranteed deadline, the remaining customer requests are rejected.

When a task is inserted at a particular position in the transmission queue, the start and finish times of all tasks after it, and the latest start time of all tasks before it will have to be updated.

3.3. Algorithm 3 (No Guarantees - NG)

This algorithm corresponds to customer service policy 3. The motivation for this algorithm is that when a new request comes, we should try and schedule it as early as possible provided that there is an increase in total profits. This algorithm is a "greedy" algorithm. The tasks are ordered based on their highest H function value. While scheduling T_h, we choose the slot which gives the maximum gain. *Gain* is given by $profitability\_inc$ – $profitability\_dec$, where $profitability\_inc$ is the increase in profitability (see equation 1) if a task were to be inserted at a slot and $profitability\_dec$ is the decrease in profitability due to the shifting of previously allocated tasks to later deadlines. The maximum gain is the largest of the gains taking into consideration all channels. We then allocate T_h in the interval corresponding to max_gain and recalculate the start and finish times of affected tasks. Some tasks may miss their deadlines as a result of this insertion, such tasks will be dropped from the queue. This is a result of the policy being used. For tasks with previously scheduled instances, we try to move them to an earlier time and not later.

Algorithm NG(τ)

```
begin
  RL ← Tasks in τ ordered on their highest H function value
  while (RL ≠ φ)
    T_h ← Task with highest profitability
    max ← 0
    for j ← 1 to N do
      cmax ← maximum of profitability_inc – profitability_dec
              for channel j such that profitability_inc
              – profitability_dec ≥ 0
      av_time ← start time of time slot corresponding to cmax
      if (cmax > max) then
        max ← cmax
        C_b ← j
        stime ← av_time
    end for
    Determine the completion_time for T_h
    with start time as stime
    if (completion_time < d_hn_h)
      Allocate T_h in the transmission queue for C_b corresponding
      to the start time stime, discard tasks missing deadlines
      if (T_prev exists) then
        Batch T_prev with T_h
    RL ← RL − T_h
  endwhile
end
```

3.4. Heuristic Functions

The following is a list of H functions that can be used to order the tasks

- Highest Profitability First (MAX_Pa) We define the *Profitability* p_i (of task T_i) as

$$p_i = \frac{c_{ij}}{S_i} \qquad (1)$$

  where $1 \leq j \leq n_i$ is the j^{th} deadline associated with task T_i and S_i is the size of the i^{th} product.

- Highest Profits First (MAX_Pr)

- Most Requests First (MAX_Req)

- Earliest Deadline First (MIN_D)

- Smallest Size First (MIN_S)

The first four heuristic functions change with the deadline option. For e.g., the profitability of a product A is higher for deadline of 10 hours than for 20 hours. The last heuristic function is invariant with the deadline, therefore certain modifications are to be made when using MIN_S with ET and GD. For each task chosen in the order of their sizes, we find the earliest_completion_time. We allocate T_h to the corresponding time slot if the task in consideration has a deadline d_{hl}, $1 \leq j \leq l \leq n_h$ that is greater than or equal to earliest_completion_time. The least such deadline is the guaranteed deadline for the task. If no such deadline exists, we can discard the task. In effect, we are avoiding having to reinsert T_h in RL because the position of T_h in RL would not change. Certain modifications also need to be made when using the last four heuristic functions with Algorithm 3. Instead of profitability_inc and profitability_dec, we need to use profit_inc and profit_dec where profit_inc is the increase in profit if a task were to be inserted at a given time slot, and profit_dec is the decrease in profit as a result of moving some tasks to later deadlines, i.e., we are using only the cost, not dividing it by the size as in equation 1.

The next set of algorithms are from [2], they are being briefly described here as we will be comparing the performance of our algorithms against these algorithms.

3.5. The Non-Cyclic Scheduler

This algorithm is for atomic transmission of products. The channels are divided into equal-sized slots, the number of slots is equal to the number of tasks (K) being scheduled at the current scheduler invocation. An initial allocation of tasks to these slots is then made using one of the algorithms for the transportation problem [3]. After the initial allocation is obtained from the transportation algorithm, the tasks are packed back-to-back so that the gaps caused by task sizes being lesser than the assumed slot sizes are filled. A "greedy heuristic" is then used to optimize this allocation. We iteratively consider two unequal sized tasks i_1 and i_2 and swap them either way or pack them in the same channel provided there is a positive gain in profit in the new allocation. This step is continually employed till there are no more improvements in profit.

3.6. The Cyclic Scheduler

The algorithms in this subsection are for segmented transmission. The subtasks are created as described in section 2.2 (see Fig. 1 for an example). At the current scheduler invocation time, we can construct a composite cost function $h_{ij}(t)$ for the j^{th} subtask of the i^{th} product where t is the time measured from the start of the scheduler invocation. The subtasks for any product form a linear chain of precedence because the product is assumed to be cyclically broadcast. The following are the two heuristic algorithms for cyclic scheduling (see [2] for details):

3.6.1. Avoid Lost Revenue (ALR) For any channel l, we choose the triple (liJ) which maximizes

$$\frac{S_{liJ}}{t_{iJm}}\left[H_i(0) - \left[\sum_{j=1}^{J-1} h_{ij}(S_{lij})\right] - h_{iJ}(t_{iJm})\right] \qquad (2)$$

where m refers to the time (deadline) for subtask J of the i^{th} task at which the step function h_{iJ} increases. S_{lij} is the completion time of the j^{th} subtask of the i^{th} product on channel l and $H_i(0)$ is the value of the cost function of the product at the current scheduler invocation.

3.6.2. Bang-For-the-Buck (BFB) The triple (liJ) which maximizes the following equation is chosen

$$\frac{\sum_{j=1}^{J} h_{ij}(S_{lij})}{S_{liJ}} \qquad (3)$$

Two of the deficiencies in the cyclic schedulers is the possible waste in bandwidth and also the fact that we would have to schedule a subtask even if no deadline can be met because we would be breaking the cyclic transmission of the product otherwise, customers with subtasks later than this one would be missing this part of the product.

4. Experimental Results

In this section, we analyze the performance of the algorithms for the metric 'Percent of maximum possible revenue' PR.

$$PR = \frac{\text{Profit earned by the scheduling algorithm}}{\text{Maximum possible profit}} \times 100 \qquad (4)$$

The maximum possible profit is the sum of delivery costs of the first options of all customers for all products.

A full channel contains 525 lines of resolution while the VBI contains 42 lines. We assume that 20 lines of VBI are available for the broadcast of multimedia products when normal programming is being broadcast. When the channel is free, the full bandwidth can be used for the purpose of broadcast. Given a simulation length S and a VBI percentage v, the total bits that can be transmitted in the free bandwidth is

$Max\_Load =$

$$\left(\frac{v}{100} \times VBI\_BW + \left(1 - \frac{v}{100}\right) \times DC\_BW\right) \times N \times S \quad (5)$$

where VBI_BW is the bandwidth during VBI and DC_BW is the dark-channel (maximum) bandwidth of the channel. For a given load percentage l, we generate requests such that the sum of the product sizes of all the requests made is equal to

$$Actual\_Load = \frac{l}{100} \times Max\_Load \quad (6)$$

The requests are generated according to a uniform distribution such that the transmission of these requests should complete within the simulation length. The results shown in the graphs have been obtained by 10 simulation runs per load percentage per VBI percentage, such that the minimum and maximum values are at most 5% from the mean.

Table 1. Parameters for experiment 1

Parameter	Value(s)
No. of Channels N	10
VBI bandwidth	20 Kbits/s
Dark Channel bandwidth	525 Kbits/s
No. of Products M	25
Product Size range	60 - 659 MB
Zipf parameter θ	1.0
No. of options	4
Cost options chosen with probability 0.5	(10h, \$4) (20h, \$3) (30h, \$2) (40h, \$1)
Simulation Length S	100000000
Confidence interval	5%
No. of simulation runs per point	10
Load percentages	25%,50%,75%,100%
VBI percentages	0 - 100%

In the first experiment, the product sizes were chosen randomly between 60 - 659 MB, this product size range was chosen considering the size of the different categories of products that fit in a single CD. The load generated in this experiment was according to equation 6 with Max_load given by equation 5. Substituting the values for VBI and dark channel bandwidths, we get

$$Actual\_Load = \frac{l}{100} \times (525 - 5.05v) \times 1000 \times N \times S \quad (7)$$

So the load or the requests generated were "proportional" to the free bandwidth as determined by the VBI percentage. The parameters used in the first experiment are summarized in Table 1.

In the second experiment, the product sizes were chosen with the following real-life product categories - CD music, software and movies. Unlike the previous experiment

Table 2. Parameters for experiment 2

Parameter	Value(s)
VBI bandwidth	150 Kbits/s
Dark Channel bandwidth	1000 Kbits/s
No. of Products M	120
Product Size range	60 - 1500 MB
No. of options	4
Cost options chosen with probability 0.5	
Product Size 60 - 200 MB	(10h, \$4) (20h, \$3) (30h, \$2) (40h, \$1)
Product Size 350 - 650 MB	(10h, \$6) (20h, \$5) (30h, \$4) (40h, \$3)
Product Size 1000 - 1500 MB	(10h, \$10) (20h, \$9) (30h, \$8) (40h, \$7)

where we kept the cost of delivery the same independent of the size, we made the cost proportional to the size in this experiment, this would mean that a smaller sized product would be equally profitable as a larger sized product for the same number of requests unlike in the previous experiment. The simulation conducted was for the "full load" for a given load percentage, unlike the proportional load in the previous experiment.

$$Actual\_Load = \frac{l}{100} \times 10^6 \times N \times S \quad (8)$$

The load was kept constant and did not vary with the VBI percentage, i.e., requests were generated to account for the maximum possible free bandwidth and used for different VBI percentages. The parameters for the second experiment that have different values compared to the first experiment are summarized in Table 2.

We show only representative graphs to illustrate the performance of the algorithms, as the results are quite exhaustive to be shown here, see [6] for more results.

4.1. Analysis of the algorithms for percent of maximum possible revenue

4.1.1. Experiment 1 We chose the best 2 heuristics for each algorithm and compared it with TP, ALR and BFB. The best 2 heuristics for ET are MAX_Pr and MIN_S, for GD are MAX_Pa and MAX_Pr, for NG are MAX_Pa and MIN_S. The resulting graph is shown in Fig. 2. All our algorithms give significantly higher revenue levels compared to that of TP, ALR and BFB. MIN_S is the highest revenue yielding heuristic for ET indicating that scheduling products in the increasing order of their sizes should result in more number of products being transmitted and hence higher profits. The graph for MAX_Pr is quite close to that of MIN_S for ET indicating that scheduling smaller sized products will result in higher total profits or the higher profit yielding products have necessarily been small in size. $GD : MAX\_Pa$ gives revenue higher than ET upto 85% VBI after which it falls. This is because the laxity in meeting task deadlines is very low for high VBI because of which the performance of GD declines to that of ET. Here the performance of MAX_Pa is indicative of its performance in ET where it was the lowest yielding heuristic.

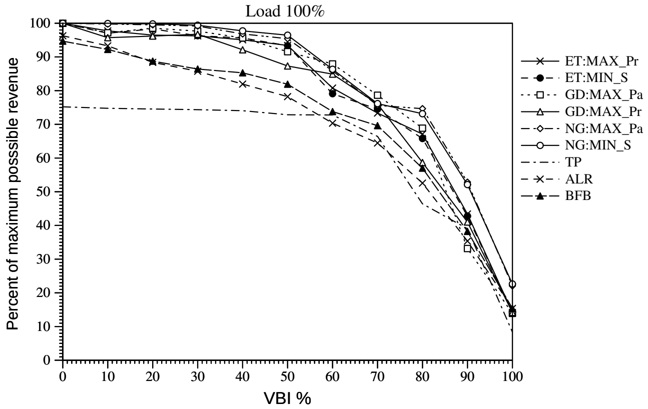

Figure 2. Comparison of the best heuristics for ET, GD and NG against TP, ALR and BFB for PR for simulation parameters 1

TP is the lowest revenue yielding algorithm. This indicates that TP is meeting later deadlines of tasks or tasks are being discarded, though the greedy heuristic should be ensuring a solution that cannot be improved in terms of revenue earned. Also since TP is an exact time algorithm, the schedule is constructed at the end of the transmission queues. BFB is the best heuristic algorithm for segmented transmission which is as expected because BFB tries to schedule the maximum revenue yielding subtask chain for a given duration of time as compared to avoiding the maximum possible loss in revenue like ALR. NG is the highest profit yielding algorithm followed by GD and then ET. GD gives comparable revenue upto medium-high VBI after which there is a significant difference between GD and NG. ET gives lower revenue than these two algorithms because the tasks are always inserted at the end of the transmission queues.

4.1.2. Experiment 2

The best 2 heuristics in experiment 2 for ET are MIN_D and MIN_S, for GD are MIN_D and MIN_S and for NG are MAX_Pa and MIN_D. No algorithm gives 100%

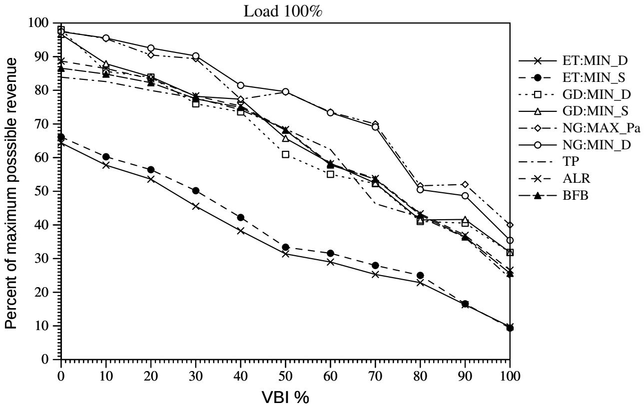

Figure 3. Comparison of the best heuristics for ET, GD and NG against TP, ALR and BFB for PR for simulation parameters 2

maximum revenue as compared to the previous experiment. Again NG is the best algorithm in terms of total revenue earned. For NG, MIN_D is the best heuristic for 0 - 50% VBI whereas MAX_Pa is the best heuristic for 50 - 100% VBI. So deadline ordering seems to be more suitable when there is sufficient free bandwidth resulting in tighter deadline tasks yielding good revenue. For high VBI the revenue per bandwidth unit counts in gaining more revenue. ET gives very low revenue compared to all other algorithms which indicates that ET is not scalable for large number of products. GD yields only slightly better revenue compared to ALR and BFB. TP gives comparable revenue and the initial revenue levels are much higher for low - medium VBI as compared to the previous experiment.

4.2. Discussion of the Results

When there are a small number of products, all our algorithms give high revenue compared to TP, ALR and BFB. NG gives the highest revenue followed by GD and ET. The best heuristics for each of these algorithms have been enumerated. If revenue is the only concern, NG is the best policy to follow. Guaranteeing a task for a deadline (GD) offers the flexibility of rescheduling with certain constraints. Both ALR and BFB are better than TP which yields moderate revenue compared to all other algorithms. Most of the requests are satisfied by our algorithms.

For a large number of products, two of our algorithms (GD and NG) are shown to yield good profit. ET does not scale up to a large number of products and both the revenue earned and the number of requests satisfied are very low. ALR and BFB are quite efficient and give only slightly lower PR than that of the smaller number of products case. So segmented transmission can be used for large number of products with an assured level of revenue. The performance of GD is comparable to that of ALR and BFB. Though TP gives good revenue for a large number of products, its running time is very high because of the greedy heuristic. Guaranteeing a deadline seems to be a better proposition as a policy that can be implemented as compared to explicit times of transmission as suggested by [2].

We also analyzed the performance of the algorithms for the metric 'Success Ratio' SR.

$$SR = \frac{\text{Number of requests that are satisfied}}{\text{Total number of requests}} \quad (9)$$

The success ratio of the different heuristic algorithms were very close to each other for us to distinguish them based on this metric [6], showing that this traditionally used metric is inappropriate when profits are to be maximized.

5. Related Work

The current scheduling problem is different from a multiprocessor scheduling problem unless certain constraints are imposed on the type of free bandwidth. This is because in a multiprocessor scheduling problem the processors speed is known and is always constant and the worst case computation time for any task can be estimated. But since the free bandwidth in a television channel varies with time, the transmission time for a task depends on the time slot assigned for the broadcast.

Our work has been mainly motivated by Aggarwal *et. al.* [2], we have studied their algorithms earlier in this paper. Close in spirit to our work is that of Almeroth and Ammar [4] who have implemented an Interactive Multimedia Jukebox (IMJ) in which users can send requests for audio or video over the web and receive it through a multicast backbone called the Mbone. Their architecture is similar to our proposal, but our model considers the free bandwidth in the television broadcast medium for insertion of digital content (products) in addition to the TV programmes scheduled at a particular time (see [6] for an illustration of the broadcast model) whereas IMJ is intended to send video content only over the backbone. In the IMJ, the current schedule for broadcast can be observed using the browser and users can receive ongoing multicasts over the Mbone by tuning into the desired channels.

[5] contains a comprehensive survey of various data dissemination systems and the corresponding issues. [1] describe several algorithms including preemptive ones for broadcast on demand. But requests for data items do not have associated deadlines and there is no value function (profit) associated with broadcasting data to clients. Their algorithms are intended for an environment where clients are provided with data like weather information, traffic information, stock quotes, etc. [7] have proposed different policies for broadcasting data items to clients which have deadlines associated with the requests. These policies include periodic broadcast, on-demand broadcast with and without batching and a hybrid scheme that includes periodic and on-demand schedule for hot and cold items. There are no profits associated with data items.

6. Conclusion

We have proposed a suite of algorithms for delivering digital products using a broadcast medium for maximizing the profit accrued to the E-commerce merchant. We also have given heuristics for choosing the next best task (product) to schedule for each algorithm. An important contribution of this paper is the recognition that in practice different customer satisfaction policies may exist and so it is necessary to have algorithms tailored to each policy.

References

[1] S. Acharya and S. Muthukrishnan. Scheduling on-demand broadcasts: New metrics and algorithms. In *Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pages 43–54, Dallas, Texas, October 1998.

[2] C. C. Aggarwal, M. S. Squillante, J. L. Wolf, P. S. Yu, and J. Sethuraman. Optimizing profits in the broadcast delivery of multimedia products. In *Proceedings of the Fifth International Workshop on Multimedia Information Systems (MIS)*, Indian Wells, California, October 1999.

[3] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows - Theory, Algorithms and Applications*. Prentice Hall, Englewood Cliffs, 1993.

[4] K. C. Almeroth and M. H. Ammar. The interactive multimedia jukebox (IMJ): A new paradigm for the on-demand delivery of audio/video. In *Proceedings of the Seventh International World Wide Web Conference (WWW7)*, Brisbane, Australia, April 1998.

[5] M. Franklin and S. Zdonik. Dissemination-based information systems. *IEEE Data Engineering Bulletin*, 19(3), September 1996.

[6] V. Vinay and K. Ramamritham. Scheduling algorithms for the broadcast delivery of multimedia products. Technical report, Department of Computer Science and Engineering, Indian Institute of Technology, Bombay, March 2000.

[7] P. Xuan, S. Sen, O. Gonzalez, J. Feranadez, and K. Ramamritham. Broadcast on demand: Efficient and timely dissemination of data in mobile environments. In *Proceedings of the Real-Time Technology and Applications Symposium (RTAS)*, June 1997.