An Optimal Algorithm for Scheduling Reservations in a Bandwidth Market

David Olshefski[*], Li Zhang[t], Danilo Florissi[*], Yechiam Yemini[*]

[*]Columbia University
1214 Amsterdam Avenue
New York, NY, 10027

[t]IBM T. J. Watson Research
30 Saw Mill River Road
Hawthorne, NY 10532

Abstract—**An algorithm is presented which provides an optimal solution to the problem of scheduling non-relocatable time intervals of bandwidth in differentiated services networks. Simulations found an asymmetry between valuing an interval's length and bandwidth requirement. Longer intervals requiring less resource are favored over shorter intervals requiring more resource. The optimal algorithm is shown to respond appropriately to price as a mechanism of control whereas the offline greedy and the online FCFS algorithms do not. The solution uses integer programming, and it is shown that, except in the general case, the problem can be solved in polynomial time.**

***Index terms*— Differentiated services, bandwidth management, market-based control, scheduling algorithms.**

A. Introduction

In a differentiated services[8][9] or integrated services[16] environment, a service provider will allocate a percentage of its overall bandwidth for use in assured forwarding[18] or guaranteed service[17]. The provider provisions access to this limited resource among multiple price-competing customers. Market-based mechanisms have been successfully applied to the problem of allocation and control of resources in large distributed systems[5][7]. Bandwidth markets as described in [12][3][5] are such examples.

In this model, each customer requires bandwidth during a unique time interval that is fixed in time (non-relocatable) and the bandwidth requirement varies from customer to customer. As such, each interval is uniquely priced. The service provider may price each customer's requirement based on amount of bandwidth required, the length of the interval, historical and current demand for the time frame, and a variety of other variables such as price competition.

Orthogonal to the question of pricing, service providers must decide which customers will receive the guaranteed service. Given a limited amount of bandwidth, in the presence of previous allocations and overlapping intervals of various prices, which subset of customers should a service provider select to service in order to maximize profits? This paper gives an optimal algorithm for solving this problem.

B. Bandwidth Market

Consider an environment in which customers purchase bandwidth in advance for a specific time interval and transmission rate[1]. Customers specify their requirements as a 4-tuple, bid = (s,e,r,p), where s is the start time of the interval, e is the end time of the interval, r is the required transmission rate and p is the price.

In such a competitive environment a service provider can be queried by many comparative shoppers for the price associated for their specific intervals. Upon making their decision, some shoppers may return to the service provider to purchase the service.

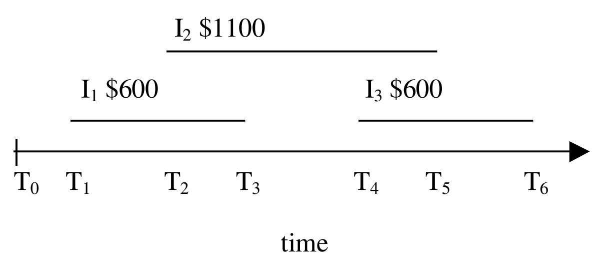


Figure 1 Canonical example of overlapping requests.

The problem facing the service provider is deciding which buyers should be serviced in order to maximize overall profit. Each provider must make this decision in the presence of previous allocations and overlapping bids. For example, take the scenario in Figure 1 where the buyer list contains three customers $I_1=(t_1,t_3,1\text{Mb/s},\$600)$, $I_2=(t_2,t_5,1\text{Mb/s},\$1100)$, $I_3=(t_4,t_6,1\text{Mb/s},\$600)$, where the first and second intervals overlap as well as the second and third. A service provider with a 1 Mb/s line would choose to service intervals I_1 and I_3 since the profit for

[1] With futures contracts, payment is made at the time of product delivery. Here, by the nature of the product, delivery is at a future time, but for our purposes, whether payment is made now or at delivery time is irrelevent.

servicing both intervals out weights the profit for servicing only I_2. This canonical example is simple but in general the decision is non-trivial. If the provider used a greedy approach by choosing the bid with the highest price, I_2, he would be losing $100.

As reservations are accepted, intervals of bandwidth become reserved and the problem recurses to the remaining *free* intervals that the provider has yet to allocate. For example, if the provider services interval I_3 and the customer for I_1 buys from another provider, then the provider is left with allocating the free intervals T_0- T_4 and T_6- $T_{infinity}$.

In the general case, it can be assumed that the provider has a *k* wide channel to allocate and each interval requires up to *k* units of bandwidth. In addition, previous allocations determine how much bandwidth is left for allocation at any point in time. Figure 2 depicts the more general problem.

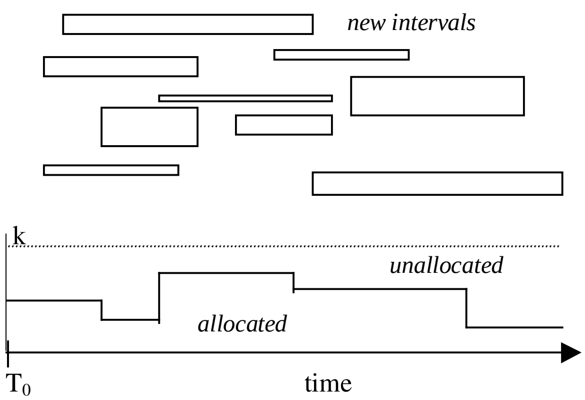


Figure 2 General problem.

Not shown in the figure, each interval has a unique price – deciding which subset of the new intervals to schedule in order to maximize profits is NP-HARD.

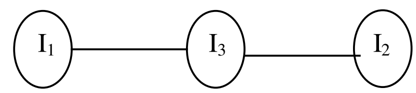
C. Related Work

Cormen et al.[1] describe a similar offline problem termed the *activity-selection problem*. In the activity selection problem, a greedy algorithm is used to select a maximum-size set of non-overlapping intervals for a single resource. Where the greedy algorithm provides an optimal solution w.r.t. scheduling the most amount of resource, our problem seeks an optimal solution w.r.t. maximizing profit.

Work on job scheduling assumes that the intervals (jobs) to be scheduled can be *relocated in time* and focuses on computational measures such as maximizing utilization, minimizing average completion time, minimizing the number of late jobs, reducing the total lateness. For more information on job scheduling algorithms, see Brucker[2].

Solving the initial problem as defined in Figure 1 is equivalent to finding the maximum weight independent set for an interval graph[11]. An interval graph is an undirected graph whose nodes have a 1-to-1 correspondence with a set of intervals of a linearly ordered set (such as the real line). Two nodes in an interval graph are connected by an edge if and only if their corresponding intervals intersect (ie. overlap). An independent set of a graph is a subset of nodes, no two of which are adjacent. The maximum weight independent set on a weighted graph is the independent set whose member's weight sum to the greatest value. In this case, the weight assigned each node is the profit for the associated interval.

The interval graph for the example in Figure 1 is:



The independent sets are $\{I_1\}$, $\{I_2\}$, $\{I_3\}$, and $\{I_1, I_2\}$, the last of which is the maximum weight independent set.

D. The Solution

The following briefly describes the formulation of the solution and leaves further details and an example to the appendix. The solution is optimal and is based on integer programming. The basic approach to integer programming is to state the problem as a maximization of

$$\mathbf{c}^{\mathrm{T}}\mathbf{x} \quad \text{or} \quad \sum_{i=1}^{n} \mathbf{c}_i \mathbf{x}_i$$

Subject to the constraints:

$$\mathbf{Ax} \le \mathbf{b}$$

Where:

$\boldsymbol{x}$	is the *n*-vector of integer unknowns
$\mathbf{c}^{\mathrm{T}}$	is the *n*-vector objective function
$\boldsymbol{A}$	is an $n \times m$ matrix
$\mathbf{b}$	is an *m*-vector
n	number of elements
m	number of constraints

The problem is formulated as follows. Let $\mathbf{c}_j$ be the profit received by the provider for scheduling interval I_j

($p - cost$) and let **x** represent the decision to schedule an interval or not: $\mathbf{x}_j = 1$ if interval I_j is scheduled and $\mathbf{x}_j = 0$ if it is not. The solution to the problem is **x** and maximizing $\mathbf{c}^T\mathbf{x}$ chooses from all possible schedules the one that generates the most profit.

The constraint we wish to satisfy is that the amount of scheduled bandwidth does not exceed that which is available for use. Therefore $\mathbf{b}_i$ is the amount of bandwidth that is available for scheduling at T_i.

Since intervals are continuous in time between their own boundary points, we need only use the set of all boundary points as the test for the constraint. The j^{th} column in matrix **A** represents interval I_j and the i^{th} row in **A** represents T_i, where T is the list of all boundary points sorted by time. $\mathbf{A}_{ij}$ is the amount of bandwidth required by interval I_j at T_i. The general integer problem is formulated as:

$$\overset{\mathbf{A}}{\begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1n} \\ A_{21} & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ A_{m1} & \vdots & \vdots & A_{mn} \\ 1 & 0 & \cdots & \cdots \\ 0 & 1 & 0 & \cdots \\ \cdots & 0 & 1 & 0 \\ \cdots & \cdots & 0 & 1 \end{bmatrix}} \overset{\mathbf{x}}{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}} \le \overset{\mathbf{b}}{\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \\ 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}} , \quad MAX \overset{\mathbf{c}^T}{\begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix}} \overset{\mathbf{x}}{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}}$$

Where:

$$\mathbf{A}_{ij} = \begin{cases} \dfrac{r_j}{br} & T_i \in I_j \quad \text{and} \quad i \le m \\ 1 & i = m + j \\ 0 & \text{otherwise} \end{cases}$$

$$\mathbf{x}_j = \begin{cases} 1 & \text{if } I_j \text{ is scheduled} \\ 0 & \text{otherwise} \end{cases}$$

$$\mathbf{b}_i = \begin{cases} \text{amount of resource to be scheduled at } T_i & if\ i \le m \\ 1 & if\ i > m \end{cases}$$

$\mathbf{c}_j$ is the profit received for scheduling interval I_j

r_j is the amount of resource required to service interval I_j

br is the base unit measure of the resource

n number of intervals

m number of boundary points at which to verify constraint

The identity matrix that is appended as the lower half of $\mathbf{A}_{ij}$ together with the identity vector appended to **b** represents the constraint that an interval be scheduled only once.

E. Simulation Results

Simulations showed an asymmetry between the value of an interval's length and its bandwidth requirement. In all cases, longer intervals were favored over shorter intervals but, surprisingly, customers requiring less bandwidth where favored over customers requiring more bandwidth. This "favoritism" is equivalent to pricing: longer intervals cost less (per time unit) to purchase than shorter intervals. Counter to intuition, the higher the bandwidth requirement, the greater the price per unit.

This inherent aspect of the problem mimics real-life markets. Producers often prefer to sell in bulk due to reduced costs. Consumers often prefer to buy in bulk due to reduced prices. Purchasing a small amount of an item often requires greater expense relative to the amount of product being purchased (i.e., you end up paying more per ounce for a small jar of mouthwash than for the economy size).

One consequence of this effect would be to entice customers in a computation market to purchase larger chunks of resource, which in turn, reduces fragmentation of the provider's resource. This also motivates the existence of brokers or middlemen, speculators who purchase large time intervals of services which they then break-up into smaller pieces and resell for profit. Without the inherent higher prices for smaller intervals this kind of speculation would have much greater risk.

The counter-intuitive effect seen for bandwidth is a result of the difficulty in getting a good "fit" when scheduling a large bandwidth chunk. This is mostly due to the difference between bandwidth and time. The amount of time required is proportionally greater than the amount of bandwidth units required.

The simulations also showed the algorithm responding appropriately to price and that price can be used as a mechanism for control. Customers willing to pay more increase their probability of meeting their requirements. The offline greedy algorithm and online FCFS, except for certain cases, are price insensitive, and therefore ineffective when using price as a mechanism for priority. Unless otherwise specified, in each simulation the price charged by the provider for scheduling an interval was equal to the product of the length of the interval and the bandwidth requirement ($ per bandwidth unit per second). Thus, profit equaled utilization.

1st. Single Resource/Single Requirement

This case is the situation where the provider has a single channel to allocate and the customers require the entire channel during an interval. Figure 3 shows, that with the interval length being uniformly distributed between [1..100], the probability of being scheduled increases with the length of the interval. Scheduling longer intervals reduces fragmentation, maximizing utilization (profit). At the same time, there is a slightly higher probability of scheduling very small intervals, presumably to fill in the small fragmentation gaps left between the longer intervals.

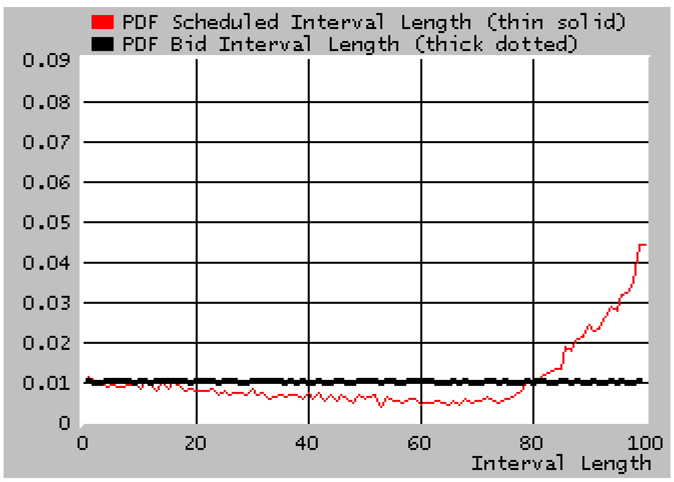


Figure 3 Longer intervals favored over shorter intervals.

Figure 4 shows a similar but less pronounced effect when the interval length is distributed exponentially with a mean of 25 time units. This is due to the increased amount of smaller intervals available for scheduling, which reduces the need to schedule long intervals to avoid fragmentation.

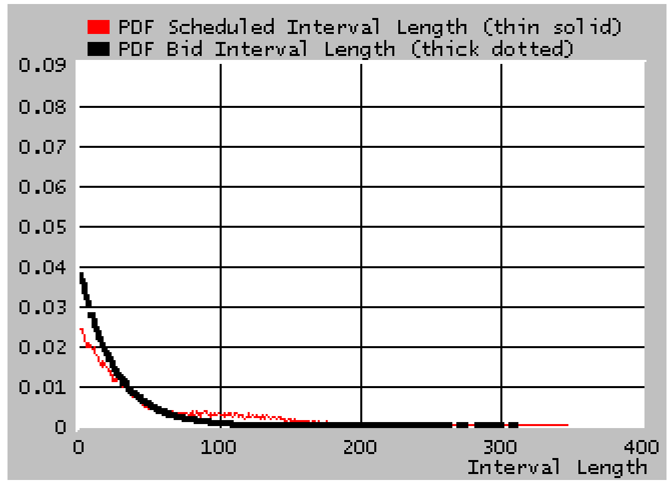


Figure 4 Larger number of shorter intervals competing against longer intervals.

In the next experiment, we doubled the price for scheduling intervals of length 20-30 time units. This was an attempt to increase the probability of these intervals being scheduled, essentially using price as a control mechanism.

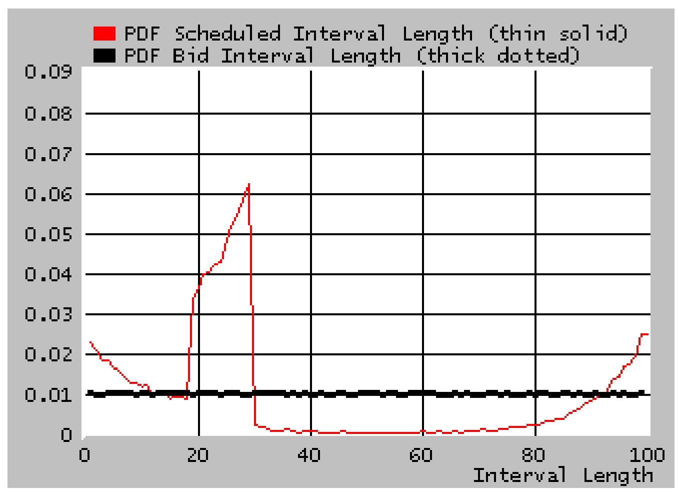


Figure 5 Higher prices improve prob. of being scheduled.

As expected, Figure 5 and Figure 6 shows the higher price for these intervals increased their probability of being scheduled, at the expense of other intervals. Even within the range of 20-30 we see that the longer intervals are favored over shorter ones (and we still see very short intervals being favored to fill in fragmentation gaps). In a highly competitive long running simulation, one would expect to see an impulse spike at 30 and a smaller peak near 1. Figure 7 further supports this – here we also doubled the price for intervals of length 60-70 as well, which do better than the 20-30 length intervals.

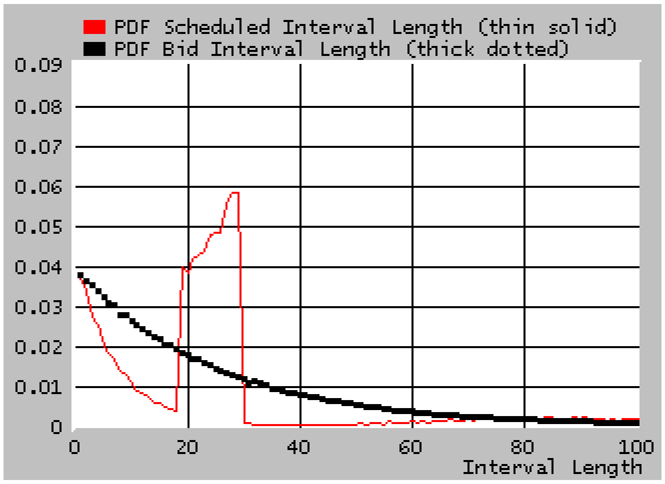


Figure 6 Exponential distribution shows similar shape.

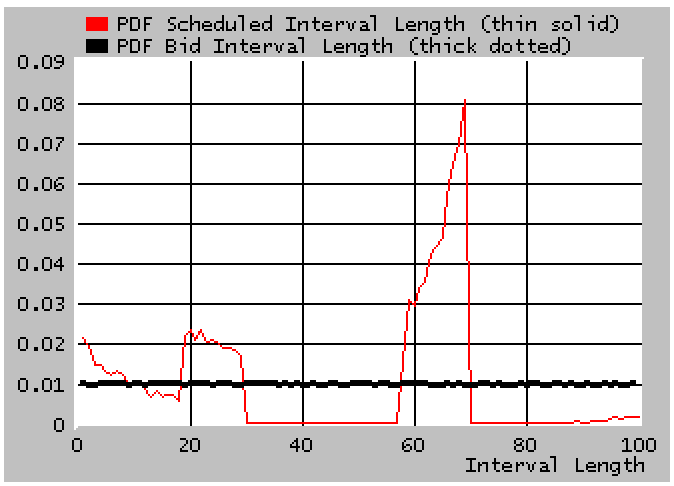


Figure 7 20-30 and 60-70 are doubled in price – the longer set is favored over the shorter set.

2nd. Multiple Resource/Single Requirement

In this experiment the service provider allocated a 10x wide channel among customers each requiring only a 1x channel. In Figure 8 we again see the sensitivity to price for intervals of length 20-30, but it is reduced due to the 10 fold increase in the amount of resources and resulting reduction in competition. Both are reflected in the difference in utilization as well (see Table 1).

This is an example of the general problem, where the provider is allocating a 10x channel and each customer requires an interval between 1 and 100 time units and between 1 and 10x channels (uniformly distributed). Figure 9 shows that the effect of favoring longer intervals is still present but not as strong.

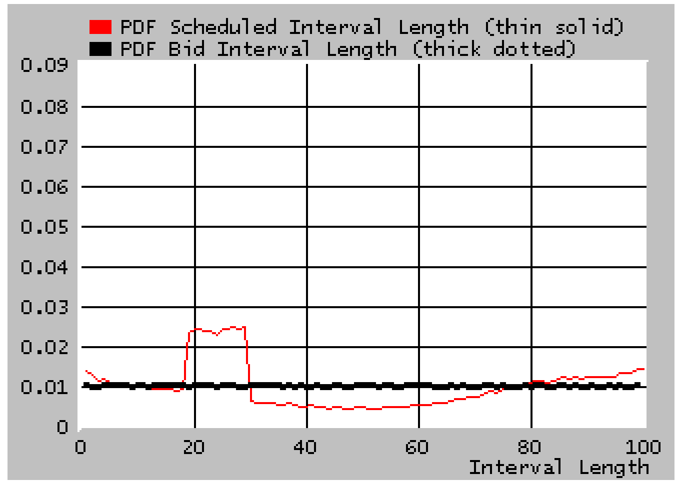


Figure 8 More supply, less competition, less price sensitive

3rd. Multiple Resource/Multiple Requirement

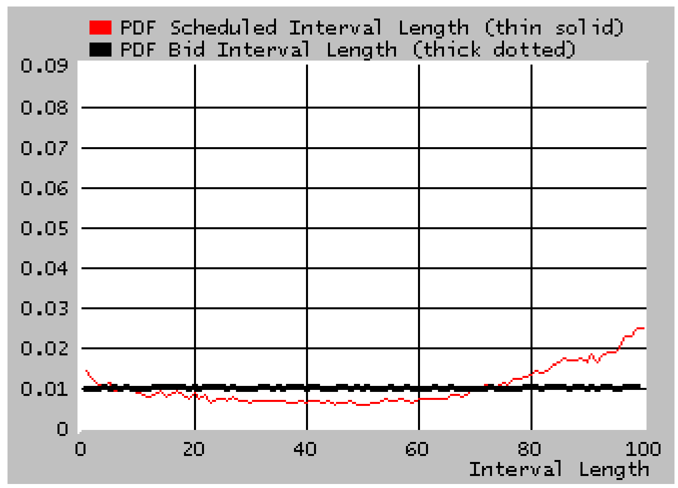


Figure 9 Longer intervals favored, but not as strongly.

Figure 10 shows the probability of being scheduled given the bandwidth requirement. Intervals requiring less bandwidth are favored over intervals requiring larger amounts of bandwidth. This is most likely due to the problem of "fitting" a large bandwidth chunk into the schedule along with other intervals.

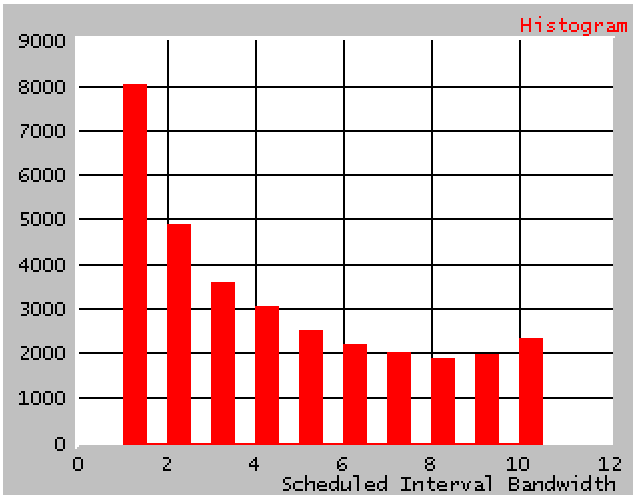


Figure 10 Lighter bandwidth requirements favored over higher.

Figure 11 and Figure 12 show the results for doubling the price for all intervals whose length was between 20 and 30 time units and required 6 units of bandwidth.

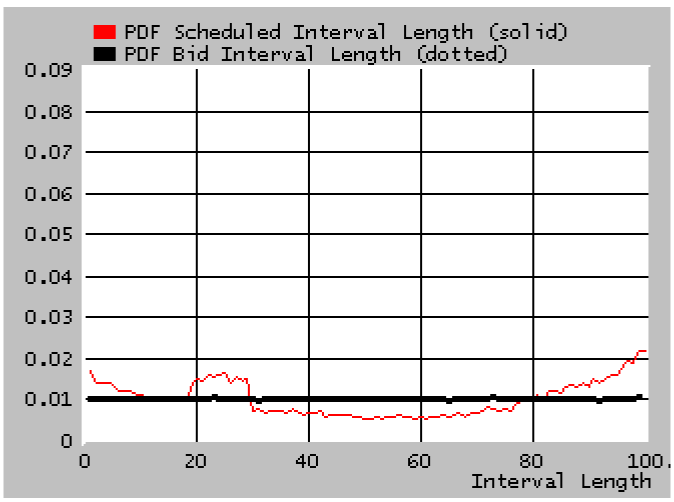


Figure 11 Intervals of length 20-30 priced higher.

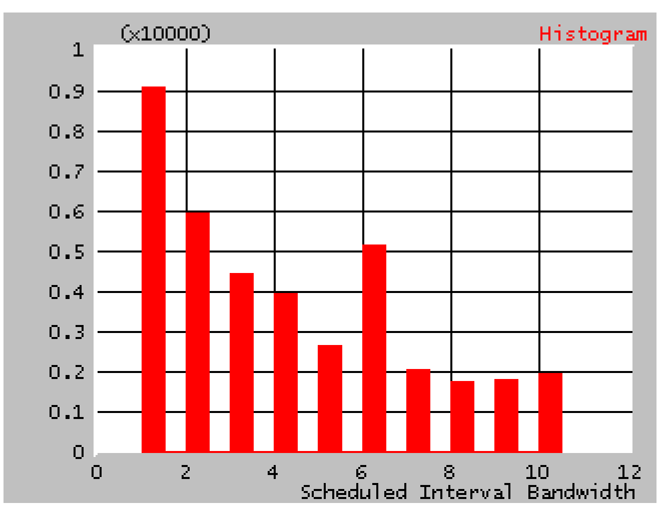


Figure 12 Intervals of 6 bandwidth units priced higher

Both graphs reflect the algorithm's reaction to the increase in price for this narrow range of customer bids.

F. Comparison to Offline Greedy

Table 1 compares the optimal solution with results from a greedy algorithm that selects intervals based on price. These simulations were similar to those in Section E *except* that the price per bandwidth unit per time unit was randomly chosen for each interval in the range of [.80, 1.20]. The greedy algorithm by design would tend to choose longer intervals since it sorted the intervals by profit then checked for "fit" from the list in a FIFO manner. Table 1 shows the percentage of utilization and average profit in dollars.

The greedy algorithm is only able to obtain 70-91% of the profit that the optimal algorithm obtains. Looking at the results for the simulation similar to that of Figure 7, the greedy algorithm is able to obtain almost 89% of the profit of the optimal solution (Figure 13 shows a close match between the two).

Simulation	**Optimal**	**Greedy**	**FCFS**
Figure 3	92% $150	73% $122	59% $83
Figure 4	93% $150	86% $137	48% $70
Figure 5	86% $197	72% $137	59% $95
Figure 6	85% $205	84% $160	48% $84
Figure 7	76% $258	68% $229	59% $107
Figure 8	79% $1400	69% $1180	59% $938
Figure 9,10	86% $1370	70% $1100	55% $775
Figure 11,12	84% $1415	69% $1110	55% $777

Table 1 Optimal, greedy and FCFS

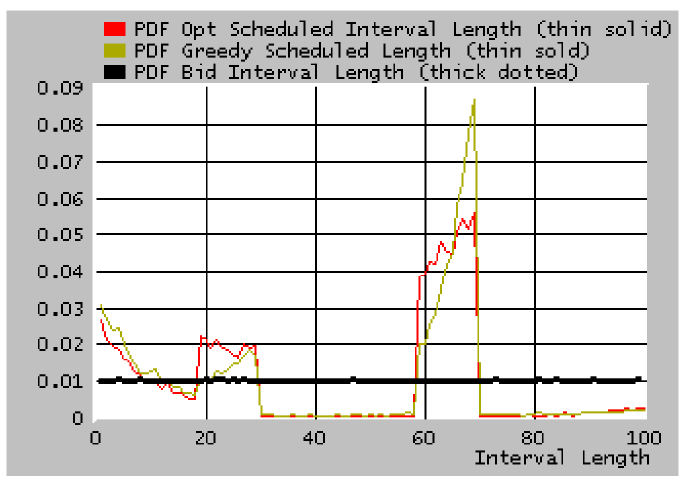


Figure 13 Optimal vs. greedy in simulation similar to Figure 7.

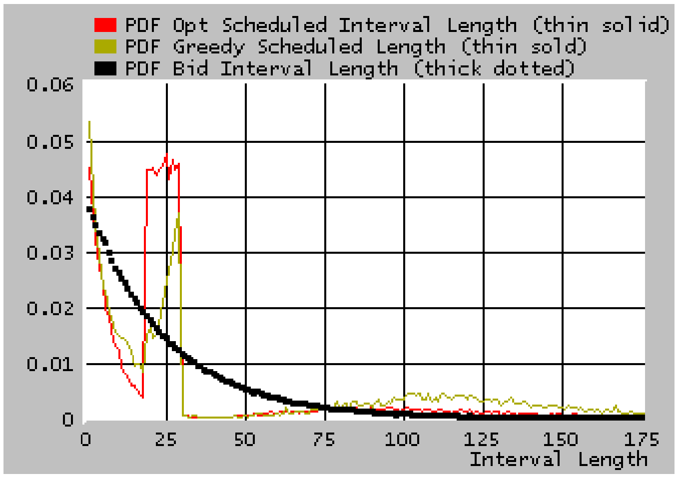


Figure 14 Same utilization but different profit.

The greedy algorithm was able to obtain 79-99% of the utilization that the optimal algorithm obtained. For the simulation in Figure 6, the greedy algorithm obtains about the same utilization as the optimal algorithm but with 22% less profit. Figure 14 shows that the greedy algorithm schedules a higher percentage of longer intervals and a lower percentage of the higher priced intervals in the range 20-30 time units (note the scale for Figure 14 is different compared to previous figures).

The greedy algorithm is particularly ineffective in the general case. Figure 15 shows the result corresponding to the last entry in Table 1. The greedy algorithm produces the opposite effect compared to the optimal algorithm with regard to scheduling bandwidth and is insensitive to price. The same result is obtained with respect to interval length (not shown). Although the greedy algorithm obtained 78% of the profit and 82% of the utilization that the optimal algorithm obtained, its insensitivity to price results in an environment where price could not be used as a mechanism for control.

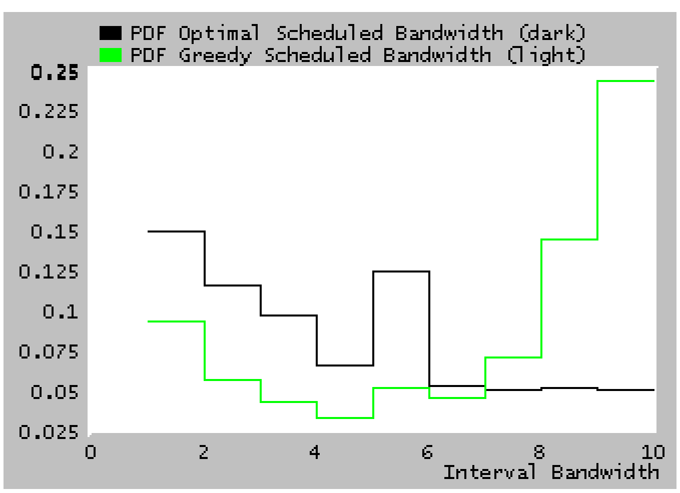


Figure 15 PDF for scheduled bandwidth.

G. Comparison to On-Line FCFS

There is a subtle distinction between the problem presented here which is an *offline* optimization problem and what is referred to as an *online* optimization problem. In our model, the service provider collects a certain amount of bids before deciding which bids to service. The provider may accept all bids during a time period or until a specific deadline. There may be a limit on the number of bids collected, after which point a decision is made. If the limit on the number of bids is set at one, the problem becomes an *online* decision.

In the *online* problem, the service provider decides whether or not to service each incoming bid at the time it is received. Only past knowledge can be used to aid in the decision - all bids to arrive in the near future are strictly unknown. The goal is the same. The service provider attempts to maximize profits by deciding whether or not to service a bid when it arrives.

One simple approach to solve the online problem is to accept a bid if there is enough resources to match its requirements (FCFS). Table 1 includes the utilization and average profit for online FCFS.

The simulation in Figure 6 shows how FCFS ignores price posting a 43.5% reduction in utilization but a 59% reduction in profit. Figure 16 compares FCFS with the optimal algorithm for the simulation similar to Figure 3. The FCFS result is the opposite of that for the optimal algorithm. Since FCFS ignores price, FCFS produces a similar shape across all simulations. Using results from the simulations of the optimal algorithm to construct better online heuristics than FCFS is left for future work.

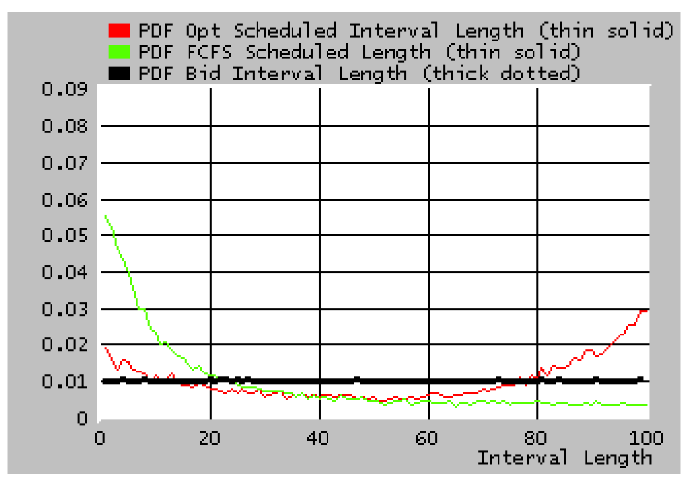


Figure 16 FCFS compared to optimal.

H. Performance

All simulations were executed with lp_solve[19] on a 400 Mhz Pentium machine running Windows NT 4.0 using the high resolution timer. The algorithm as simulated in *1st Single Resource/Single Requirement* (Figure 3 - Figure 7) took approximately 8 msecs. The simulations in *2nd Multiple Resource/Single Requirement* (Figure 8) took approximately 18.5 msecs. Figure 17 shows the probability density functions of the CPU time for the simulations in Figure 3 and Figure 8.

The performance time of the algorithm in the general case (Figure 9 - Figure 12), is about 300 times slower than the simpler cases previously mentioned. Although the average time was 6.5 seconds, the worst case was 6 minutes. Figure 18 shows that the performance is less predictable than the other two simpler cases.

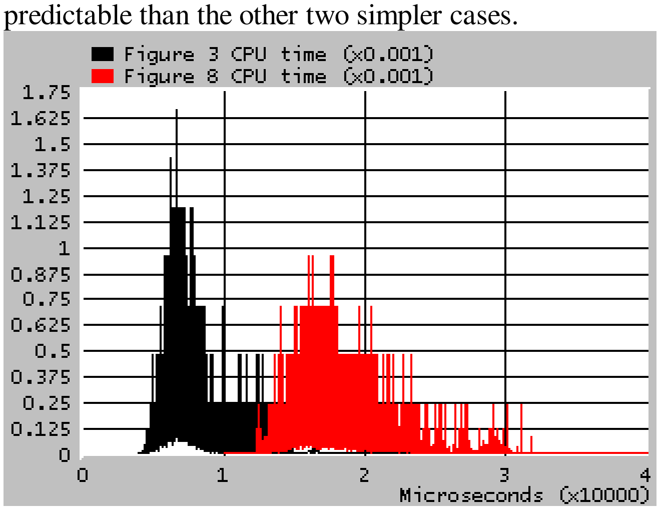


Figure 17 PDF of CPU time for simpler cases.

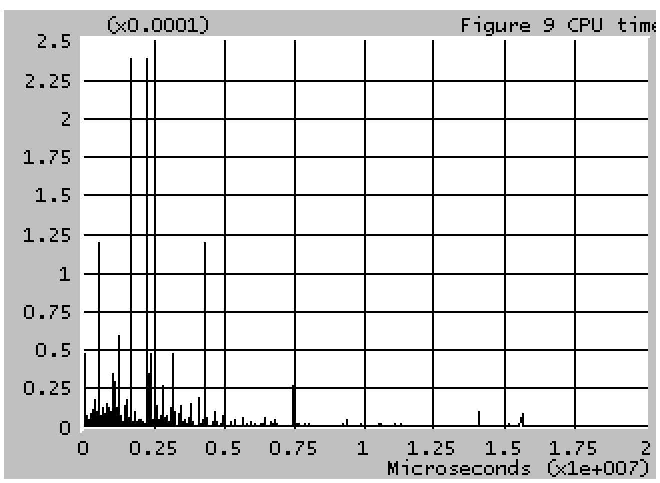


Figure 18 PDF of CPU time for general case.

This has implications on the use of the algorithm, given the time scale of the resource being allocated and how far into the future reservations are being made. Depending on these constraints, it may be necessary to place a limit on the computation time, after which a non-optimal but fast algorithm can be executed to obtain a reasonable result.

I. Conclusion

This paper presented an optimal algorithm for scheduling reservations in a differentiated services environment. The algorithm presented is based on integer programming and can be executed in polynomial time for the single resource/single requirement and multiple resource/single requirement cases. Through simulations it was shown to exhibit the same advantage to buying and selling in bulk that arise in human markets and was shown to respond appropriately to price control. One result was an asymmetry between interval length and bandwidth requirement: that longer intervals requiring less bandwidth are favored over shorter intervals requiring more bandwidth.

Simulations showed that a greedy offline algorithm could perform within 70%-91% of optimal (w.r.t. profit), and is particularly ineffective in the general case with respect to treating price as a mechanism of control. An online FCFS algorithm could obtain anywhere between 41%-67% of optimal and was ineffective w.r.t. price control under all conditions.

Several other problems could be investigated as follow on work. Bids could be modified to be a 5-tuple (s,e,r,p,l) where s is the *earliest start time*, e is the *deadline*, r is the required transmission rate, p is the price and l is the length of a *continuous* interval. In other words, the start time of the interval can be adjusted within a limited bounds.

Similar to the previous would be (s,e,t,p) where s is the *earliest start time*, e is the *deadline*, t is the required transmission rate times the length, and p is the price. Here, the customer is only asking for a transmission block – the provider is free to choose the length and bandwidth of the (continuous) interval which best fits his schedule and is bounded by s and e.

Finding a solution to the online optimization problem, determining how long a provider should collect incoming bids in order to maximize profits, applying the algorithm to more than one competitive supplier, and customer strategies is the subject of future research.

J. References

[1] T. Cormen, C. Leiserson and R. Livest, *Introduction to Algorithms*, McGraw Hill, ISBN 0-262-03141-8

[2] P. Brucker, *Scheduling Algorithms*, Springer Verlag, ISBN 3-540-64105-X.
[3] Band-X, http://www.band-x.com
[4] RateXchange, http://www.ratexchange.com
[5] S. Clearwater, Market-Based Control: A Paradigm for Distributed Resource Allocation, ISBN 9810222548, World Scientific Publishing.
[6] *Multiagent Systems on the Net and Agents in E-commerce*, Communications of the ACM, Marc 1999, Vol. 42, No. 3.
[7] Proceedings of the First International Conference on Information and Computation Economies, ICE-98, ISBN 1-58113-076-7, ACM.
[8] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss, *An Architecture for Differentiated Services*, RFC 2475, Dec. 1998, ftp://ftp.isi.edu/in-notes/rfc2475.txt
[9] Y. Bernet, J. Binder, S. Blake, M. Carlson, B. Carpenter, S. Keshav, E. Davies, B. Ohlman, D. Verma, Z. Wang, W. Weiss, *A Framework for Differentiated Services,* Feb. 1999, http://www.ietf.org/internet-drafts/draft-ietf-diffserv-framework-02.txt
[10] M. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, 1980, ISBN 0-12-289260-7.
[11] A. Frank, *Some Polynomial algorithms for certain graphs and hypergraphs*, Proc. 5th British Combin. Conf., Congressus Numerantium No. XV, Utilitas Math., Winnipeg, MR53 #13500.
[12] F. Reichmeyer, L. Ong, A. Terzis, L. Zhang, R. Yavatkar, *A Two-Tier Resource Management Model for Differentiated Services Networks,* Nov 1998, http://www.ietf.org/internet-drafts/draft-rotzy-2-tier-management-00.txt.
[13] A. Mehra and D. Verna, *Architectural Considerations for DiffServ Servers*, Feb. 1999, http://www.ietf.org/internet-drafts/draft-mehra-diffserv-servers-00.txt
[14] Y. Bernet, D. Durham, F. Reichmeyer, *Requirements of Diff-serv Boundary Routers*, Nov. 1998, http://www.ietf.org/internet-drafts/draft-bernet-diffedge-01.txt.
[15] Nemhauser and Wolsey , *Integer and Combinatorial Optimization*, Wiley, 1988.
[16] R. Braden, D. Clark, S. Shenker, *Integrated Services in the Internet Architecture: an Overview*, rfc1633, June. 1994, ftp://ftp.isi.edu/in-notes/rfc1633.txt
[17] S. Shenker, C. Partridge, R. Guerin, *Specification of Guaranteed Quality of Service*, rfc2212, Sep. 1997, ftp://ftp.isi.edu/in-notes/rfc2212.txt
[18] J. Heinanen, T. Finland, F. Baker, W. Weiss and J. Wroclawski , *Assured Forwarding PHB Group*, Feb 1999, ftp://ftp.isi.edu/in-notes/rfc2597.txt
[19] Eindhoven University of Technology, Department of Electrical Engineering, Information and Communications Systems Group, ftp://ftp.es.ele.tue.nl/pub/lp_solve

K. Appendix I

Further details on the solution presented in Section D are provided here to give the reader a better understanding of the formulation.

Consider the simple *base case*: the transmission rate required by all customers is the same, *r*, and the service provider has a single line at rate *r* in which to allocate. The integer program for the example in Figure 1 is shown below. The solution of $x\begin{bmatrix}1 & 0 & 1\end{bmatrix}$ indicates that the service provider should accept I_1 and I_3, but not I_2.

$$\overset{\mathbf{A}}{\begin{bmatrix}1&0&0\\1&1&0\\1&1&0\\0&1&1\\0&1&1\\0&0&1\end{bmatrix}} \overset{\mathbf{x}}{\begin{bmatrix}x_1\\x_2\\x_3\end{bmatrix}} \le \overset{\mathbf{b}}{\begin{bmatrix}1\\1\\1\\1\\1\\1\end{bmatrix}}, \quad MAXIMIZE \overset{\mathbf{c}^T}{\begin{bmatrix}\$600\\\$1100\\\$600\end{bmatrix}} \overset{\mathbf{x}}{\begin{bmatrix}x_1\\x_2\\x_3\end{bmatrix}}$$

Notice that each column in **A** can be described by the regular expression *0+1+1*0, since each column contains a sequence of (at least two) 1's representing the set of boundary points contained in the interval. Matrices having this special structure are referred to as interval matrices. It is known that if **A** is an interval matrix, then it is totally unimodular. This means that if **b** is integral, then every vertex of the polytope $\mathbf{Ax} \le \mathbf{b}$ is integral. Therefore, it is guaranteed that the linear programming relaxation of the integer programming problem always has an integer optimal solution. We can therefore find this optimal solution in polynomial time using interior-point algorithms.

There is a minor issue concerning whether or not two adjacent intervals are considered overlapping if they only share a single boundary point. The formulation above considers this an overlap and will not schedule two such intervals together. This can be avoided by simply ensuring that if two intervals overlap, they do so at more than one point – start points for intervals can be boundary aligned on a different boundary than what interval end points are aligned on.

From the base case a slightly more general problem can be tackled: the service provider now has *k* resources in which to allocate. Equivalently, the service provider can provide service for up to *k* overlapping customer requests at any point in time. The integer program is modified by changing $\mathbf{b}_i$ to be the number of overlapping intervals that can be serviced at T_i. Initially $\mathbf{b}_i$ is set to *k*, but as intervals are allocated which include T_i, $\mathbf{b}_i$ is reduced from *k*.

Allowing the scheduling of overlapping intervals, requires the additional constraint that the same interval is not scheduled more than once. This is solved by simply appending the identity matrix to the bottom of **A** and set $\mathbf{b}_i = 1$ for $i > m$, where *m* is the number of boundary point constraints.

It should be noted that the new matrix A is an interval matrix plus an identity matrix. This matrix is not an interval matrix. However, Proposition 2.3 in Nemhauer and Wolsey[15] states that if A is totally unimodular, and b, c, d, e are integers, then the polyhedron (b <= Ax <= c, d <= x <= e) is integral. In our case, the new matrix is still totally unimodular. Therefore, using linear programming algorithms, the problem can still be solved in polynomial time.

As intervals are allocated, their boundary points become relevant as checkpoints in the problem, therefore the boundary points of previously allocated intervals are included in T. This requirement can be shown by an example.

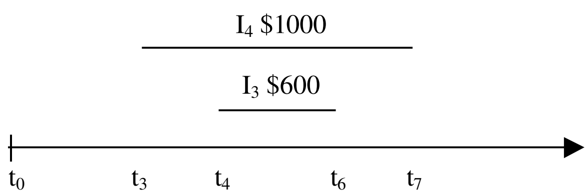


Figure 19 I_3 must be included as a constraint.

Suppose a provider with $k = 1$ resource has allocated an interval I_3=(t_4,t_6,1Mb/s,\$600) and the bid list contains only one bid, I_4=(t_3,t_7,1Mb/s,\$1000). Graphically, this is depicted in Figure 19. Obviously points t_4 and t_6 are needed as constraints to obtain the correct solution.

Referring back to Figure 1, assume the provider starts with a 4Mb/s line, four times the capacity of the 1Mb/s base rate ($k = 4$). The integer program now becomes:

$$\overset{\mathbf{A}}{\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}} \overset{\mathbf{x}}{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}} \le \overset{\mathbf{b}}{\begin{bmatrix} 4 \\ 4 \\ 4 \\ 4 \\ 4 \\ 4 \\ 1 \\ 1 \\ 1 \end{bmatrix}} , \quad \textit{MAXIMIZE} \quad \overset{\mathbf{c}^T}{\begin{bmatrix} \$600 \\ \$1100 \\ \$600 \end{bmatrix}} \overset{\mathbf{x}}{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}}$$

Which solves to $x\begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$ indicating that the service provider has enough bandwidth to service all three customer bids. Now assume that only the counter-offer for interval I_3 is accepted and allocated. This leaves the free intervals T_0-T_4 and T_6-T_{infinity} at an available rate of 4Mb/s and the partially allocated interval T_4-T_6 at the rate of 3Mb/s. Assume that the bid list does not change and the bids for I_1 and I_2 are still on the bid list when the provider next decides to check the market. The formulation of the integer program taking the allocation for I_3 into account is:

$$\overset{\mathbf{A}}{\begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}} \overset{\mathbf{x}}{\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}} \le \overset{\mathbf{b}}{\begin{bmatrix} 4 \\ 4 \\ 4 \\ 3 \\ 3 \\ 3 \\ 1 \\ 1 \end{bmatrix}} , \quad \textit{MAXIMIZE} \quad \overset{\mathbf{c}^T}{\begin{bmatrix} \$600 \\ \$1100 \end{bmatrix}} \overset{\mathbf{x}}{\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}}$$

$\mathbf{b}_4$, $\mathbf{b}_5$, and $\mathbf{b}_6$ are decremented to 3 to reflect that 1Mb/s of the 4Mb/s has already been allocated at T_4, T_5, and T_6. This solves to $x\begin{bmatrix} 1 & 1 \end{bmatrix}$ indicating that enough resource is still available to service both I_1 and I_2.

The general solution obtained in Section D is arrived at by assuming that customers may have an arbitrary transmission rate requirement. The integer program is only slightly modified by changing $\mathbf{A}_{ij}$ to be the amount of resource required by the customer at interval I_j at T_i rather than the boolean value of 0 or 1 to indicate if the interval contains T_i (i.e., that interval I_j requires 1Mb/s of bandwidth at T_i).

Referring to the example suppose that when the provider revisited the market for the second time, I_2 was instead (t_2,t_5,4 Mb/s,\$3100). The problem would be formulated as:

$$\overset{\mathbf{A}}{\begin{bmatrix} 1 & 0 \\ 1 & 4 \\ 1 & 4 \\ 0 & 4 \\ 0 & 4 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}} \overset{\mathbf{x}}{\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}} \le \overset{\mathbf{b}}{\begin{bmatrix} 4 \\ 4 \\ 4 \\ 3 \\ 3 \\ 3 \\ 1 \\ 1 \end{bmatrix}} , \quad \textit{MAXIMIZE} \quad \overset{\mathbf{c}^T}{\begin{bmatrix} \$600 \\ \$1100 \end{bmatrix}} \overset{\mathbf{x}}{\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}}$$

Which solves to $x\begin{bmatrix} 1 & 0 \end{bmatrix}$. The previous allocation of I_3 prevents the service provider from accepting I_2. This general case formulation does not preserve the interval matrix property of **A**, and is therefore, more computationally intensive to solve.

Notice that in the preceding sections we referred to *profit* rather than *revenue*. This is an important distinction that allows the algorithm to distinguish between intervals whose price is the same but whose resource consumption is not. Graphically this is depicted in Figure 20. In this case, the service provider would prefer to schedule I_3 since it requires less resource. If $\mathbf{c}_j$ is set to

$revenue - cost$, then the algorithm will break *ties* such as these by selecting the interval that requires less resource.

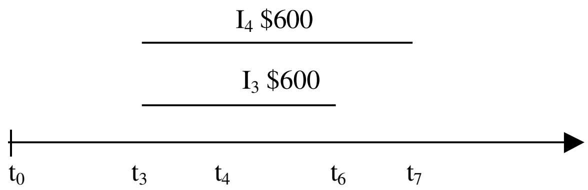


Figure 20 Including resource cost as part of profit.

This shows a characteristic of the problem – multiple optimal solutions may exist which in a sense represent a *tie*. Another situation where multiple optimal solutions may exist for this problem is the following:

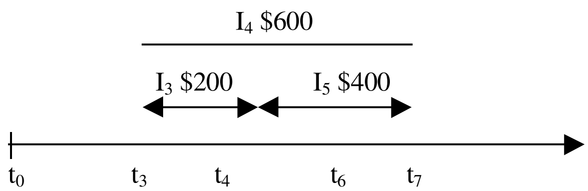


Figure 21 Equally profitable choice.

Scheduling I_4 is equally profitable as scheduling both I_3 and I_5. If the service provider always chose the longer interval over the two shorter ones a bias would exist penalizing shorter intervals – shorter intervals would have to pay more in order to receive equal consideration. To remove this bias, the service provider needs only to alternate between choices over time. This can be done by simply adding or subtracting a small fixed *tax* to each interval. Decreasing each interval's profit by a small amount favors the choice of scheduling fewer but longer intervals, increasing the profit favors scheduling more intervals but of shorter length. In actuality, this kind of tie would probably be so infrequent as to have only a miniscule effect on outcome, but is presented here for completeness.

A simple mechanism can be used with the algorithm described above to allocate a specific resource (channel) to a specific scheduled interval – graph coloring[10]. Sort the list of scheduled intervals by increasing start time. Allocate the first interval to the set of resources that are available at the earliest time. Applying this greedy approach for the rest of the intervals on the list results in a mapping between resources and intervals that satisfies the constraints imposed by the algorithm.