Approximation and Complexity in Numerical Optimization: Continuous and Discrete Problems (P. M. Pardalos, Editor), pp. 1-2

Models and Solution for On-Demand Data Delivery Problems

Michael C. Ferris (ferris@cs.wisc.edu)
Computer Sciences Department, University of Wisconsin, 1210 West Dayton Street, Madison, WI 53706

Robert R. Meyer (rrm@cs.wisc.edu)
Computer Sciences Department, University of Wisconsin, 1210 West Dayton Street, Madison, WI 53706

Abstract

Supporting on-demand access to large, widely shared data such as popular video objects and internet web sites requires effective use of caching at regional (proxy) servers. This caching problem is very complex because of the need to consider bandwidth as well as storage constraints at the regional servers, and because of the bandwidth sharing possibilities provided by recently proposed delivery techniques.

A linear mixed integer model is developed to configure the allocation of data objects in a hierarchical architecture containing one or more shared central servers and multiple regional servers. We will describe motivation and format of the basic model, and several enhancements to the model formulation and solution process that are necessary to solve the problem within reasonable time limits.

Keywords: on-demand access, data delivery techniques, integer programming, redundant constraints, modeling, variable selection priorities.

1 Introduction

We are all becoming familiar with the notion of "pay-per-view" television. As this becomes an increasingly popular form for delivering movies, news clips, or more general information over a network, the need to design more effective delivery techniques becomes increasingly important. This paper considers models allowing for regional servers who can store and deliver data over a local network, therefore reducing global congestion. This kind of model could also be used for internet applications, where data can be cached at proxy servers, again reducing the need of global network bandwidth and remote server bandwidth.

In internet applications, data is typically required as soon as possible. As such, it resembles video-on-demand systems where a customer requests a movie to be played as soon as possible. To generate reasonable response to such demand, a number of channels K can be allocated to each movie, guaranteeing a waiting period of at most L/K, where L is the movie length. In Section 2, we describe a new delivery technique that can reduce this waiting period still further, and a mathematical formulation that allows us to consider this delivery technique in our models. The new technique splits up the movie into increasing length segments, and delivers these segments over different channels.

While reducing the waiting time for a customer is likely to be important from a customer standpoint, the principal concern is network delay and bandwidth saturation. In practice, there are large numbers of movies being sent over the network. One idea to alleviate the delay that has garnered much attention is the notion of storing the most popular movies at a regional site. This "caching" strategy has the potential to save bandwidth at the remote server and also on the remote network, with the potential of generating cost savings and improved performance for the service provider.

In the model we consider here, we assume there are a number P of regional servers and n objects (movies) . The regional servers only have the ability to store and deliver a fraction of the objects at hand. Given that the objects are being delivered in the segmented fashion mentioned above, the issue we address is whether to store a particular object completely at the regional sites, the first k segments of the object at the regional sites, or store none of it there. Physical arguments are used in [3] to outline why it will never be optimal to store only the last segments of an object at the regional site.

The specific optimization problem is a linear mixed integer program, defined as follows:

$$
\begin{aligned}
&\min_{\theta} && C_{remote}(\theta^{\rho}, \theta^{\rho r}) + P\beta C_{regional}(\theta^{\rho r}, \theta^{r}) \\
&\text{subject to} && C_{regional}(\theta^{\rho r}, \theta^{r}) \leq N_{channels} \\
& && D_{regional}(\theta^{\rho r}, \theta^{r}) \leq N_{segments} \\
& && \theta_i^{\rho} + \theta_i^{\rho r} + \theta_i^{r} = 1, \quad i = 1, 2, \ldots, n \\
& && \theta_i^{\rho}, \theta_i^{\rho r}, \theta_i^{r} \in \{0, 1\}, \quad i = 1, 2, \ldots, n.
\end{aligned}
\tag{1}
$$

This model is a simple assignment model with two side constraints. The decision to be taken is whether to allocate all the segments of object i to the remote server $\theta_i^{\rho} = 1$, all the segments to the regional server $\theta_i^{r} = 1$, or store the initial k (a given parameter) segments regionally and deliver the remainder from the remote server $\theta^{\rho r} = 1$. As we shall see in the next section, the expressions for $C_{remote}(\theta^{\rho}, \theta^{\rho r})$, $C_{regional}(\theta^{\rho r}, \theta^{r})$ and $D_{regional}(\theta^{\rho r}, \theta^{r})$ are linear functions of the binary variables θ, while β is a problem dependent weighting between remote (satellite) and regional costs.

The problem is a special case of a mixed integer program (MIP), for which the most frequently used algorithm is a branch and bound technique [9]. At each node of the branch and bound tree, a linear programming (LP) relaxation of (1) is solved.

The two side constraints limit the amount of bandwidth (channels) available and the number of segments storable at the regional server. $N_{channels}$ represents the total number of channels available at the regional server while $N_{segments}$ is a storage capacity for disks at the regional server. The side constraints result in non-integer solutions for the LP relaxation of this problem. Below we discuss how difficulties with large branch-and-bound solution times with this model can be alleviated by the addition of redundant constraints combined with an appropriately chosen branch-and-bound strategy.

The remainder of this paper is organized as follows. The next section provides details of the skyscraper delivery techniques used to partition the data objects among the available channels. This delivery approach was assumed in generating numerical data for the above MIP model. We then discuss how the model above may be augmented by redundant constraints and variables that in combination with an appropriate solution strategy make a branch-and-bound approach feasible by drastically reducing solution time. Finally, conclusions and directions for further research are presented.

2 Skyscraper Delivery Techniques

In order to explain the expressions for $C_{remote}(\theta^{\rho}, \theta^{\rho r})$, $C_{regional}(\theta^{\rho r}, \theta^{r})$ and $D_{regional}(\theta^{\rho r}, \theta^{r})$ that are used in our model, we need to outline the actual method of data delivery in some more detail.

It is typical to assume first that the number of objects n under consideration is large and arrives according to a zipf distribution, whose probability distribution function is given by

$$\frac{\nu}{i},\ \nu = \frac{1}{\sum_{j=1}^{n} \frac{1}{j}},\ i = 1, 2, \ldots, n.$$

In practice, the overall arrival rate λ multiplies this distribution to generate an object arrival at various time instants. While this distribution models observed behavior in real systems where there are a few very popular objects and a huge number of less popular ones, the actual distribution is irrelevant to the work described here. To generate reasonable response to such demand, a number of channels K could be allocated to each object, guaranteeing a waiting period of at most L/K, where L is the object length. In practice, these K channels are only temporarily assigned to the object, and are reallocated to a different object based on a first-come first-served policy.

Two innovations in the receiver reduce this waiting period still further. The first is the ability to receive on more than one channel concurrently, the second being local storage (read/write) capacity. In tandem, these allow a customer to acquire segments of an object that is already in progress. The key idea is to break up the object (movie) into varying length segments corresponding to disjoint time intervals and to transmit these segments simultaneously on different channels. This has been termed "skyscraper delivery" [4, 6], and is depicted in Figure 1 for a case with 6 channels.

Figure 1: Channel assignment

In this example, the movie is split into segments of length 1,2,2,4,4,8 that partition the movie into disjoint time intervals. Specifically, the segment of length 1 corresponds to (movie) time interval [0,1], the first segment of length 2 corresponds to the next time interval [1,3], the second segment of length 2 corresponds to time interval [3,5], etc. The total time for the movie is thus 21=1+2+2+4+4+8 units as measured in these units. A customer who arrives before the first segment is played receives (and plays) the movie by acquiring all segments of the yellow multicasts, which correspond to all of the disjoint time intervals of the movie.

If a user arrives after the first yellow segment is playing, but before the green segment starts, then he starts playing the green segment while concurrently receiving the second (yellow) segment (on channel 2). Once the green segment is completed, the customer starts playing the second yellow segment from memory (offset) while continuing to receive (store and play later) further segments.

Users who arrive during the green segment start to receive the red segment on channel 1 then picks up the red segment on channel 2 concurrently with the yellow on channel 3. The yellow segment is stored and played back during the time where the ensuing yellow segments are stored and played (at an offset).

The latest possible time that a customer can pick up any portion of the yellow multicast is if he arrives before the blue segment on channel 1. He can then pick up all the blue segments and the final yellow segment on channel 6 and receive a jitter free movie presentation.

After the final blue segment is played on channel 1, the movie can be repeated or another movie can be started. Assuming the same movie plays repeatedly, the maximum waiting time is now $L/21$, compared to $L/6$. The amount of storage required at the receiver corresponds to the offset of the blue segment from the yellow segment on channel 1, namely $W-1$, where W is the length of the last segment.

Taking into account the full collection of objects, a queueing theory analysis [3] shows that the required number of regional and remote channels is then given by

$$C_{remote}(\theta^{\rho}, \theta^{\rho r}) = \sum_{i=1}^{n} (A_i^{\rho}\theta_i^{\rho} + A_i^{\rho r}\theta_i^{\rho r}),$$

and

$$C_{regional}(\theta^{\rho r}, \theta^{r}) = \sum_{i=1}^{n} (B_i^{\rho r}\theta_i^{\rho r} + B_i^{r}\theta_i^{r})$$

Here A_i^{ρ}, $A_i^{\rho r}$, $B_i^{\rho r}$ and B_i^{r} are nonnegative numbers.

In a similar manner, the regional storage is given by

$$D_{regional}(\theta^{\rho r}, \theta^{r}) = \sum_{i=1}^{n} (N_k\theta_i^{\rho r} + N_K\theta_i^{r}),$$

where N_k is the disk space needed to store k segments of the object (in the example above, $N_1 = 1$, $N_2 = 1+2$, $N_3 = 1+2+2$, etc, where disk space is measured in terms of a unit segment). Note that the modeler specifies an appropriate value of k.

Preliminary results [3] from the homogeneous (uniform regional server) model above with a variety of parameter settings show that it is often more cost-effective to cache just the initial segments of objects (rather than the complete data for fewer objects) and to rely on multicast delivery for some portions of hot objects while caching locally corresponding portions of cooler objects (rather than caching the hot data locally). To facilitate reasonable solution times for the mixed integer programs in [3], some additional regularity constraints on the form of the solution were added to the model to improve solution time. Since the constraints imposed further restrictions on the model solution beyond those outlined above, this was deemed unsatisfactory.

Based on those results, the model was extended in [2] to allow for heterogeneous regional servers. The heterogeneous model has two types of servers, the distinct server and a collection of $P-1$ other (homogeneous) servers. We assume that f_d is the fraction of the total requests that are from clients belonging to the distinct region. Under such an assumption, the homogeneous model described above becomes more complicated, but retains the flavor of an assignment model. The actual optimization problem we solve is given below:

$$\min_{\theta} \quad C_{remote}(\theta) + \beta(C^d_{regional}(\theta) + (P-1)C^h_{regional}(\theta))$$

$$\begin{array}{lrl} \text{subject to} & C^d_{regional}(\theta) \leq N^d_{channels} & \\ & C^h_{regional}(\theta) \leq N^h_{channels} & \\ & D^d_{regional}(\theta) \leq N^d_{segments} & \\ & D^h_{regional}(\theta) \leq N^h_{segments} & \\ & \sum_{y,z\in\{\rho,\rho r,r\}} \theta_i^{y,z} = 1, \quad i = 1,2,\ldots,n & \\ & \theta_i^{y,z} \in \{0,1\}, \quad y,z \in \{\rho,\rho r,r\}, i = 1,2,\ldots,n & \end{array} \tag{2}$$

The expressions C_{remote}, $C^d_{regional}$, $C^h_{regional}$, $D^d_{regional}$ and $D^h_{regional}$ are still linear functions of various subsets of the variables θ, each depending on whether the server is the distinct one (d), or a homogeneous one (h). The expressions remain similar to those given in [3] for the homogeneous case.

The results reported in [2] show that in addition to the conclusions drawn for the homogeneous model, in the heterogeneous case it is profitable to cache very similar sets of objects at all the regional servers (rather than to closely tailor cache contents according to local workload and server characteristics), Again, the computational resources needed to draw such conclusions were great. Furthermore, in this nonhomogeneous setting, several of the extra regularity restrictions that we had imposed on the homogeneous model altered the form of the optimal solution of the heterogeneous model, and thus could not be used.

While the results in [3, 2] point to interesting conclusions, the computational experiments were necessarily limited and small scale. We note, for example, that in heterogeneous systems that have asymmetric client workloads or diverse server capacities or bandwidths, the "socially optimal" regional server cache contents that minimize total cost of delivery may be quite different from the "individually optimal" cache content that would result if a particular (competitive) regional service provider attempted to minimize its own delivery cost. To resolve such issues further extensive experimentation with the above models and derived extensions will be necessary. In the next section of this paper we will outline some modeling and solution techniques that overcome the poor computational timings on the existing sets of problems.

3 MIP Model and Solution

Implementations of the models (1) and (2) have been developed in GAMS [1] for determining optimal allocations of objects, within the context of the proposed hierarchical delivery scheme.

The problems were taken directly from the models generated for the papers [3, 2]. The solution times for these models varied dramatically over the set of input parameters that were of interest. In the table below, we give the parameters for four cases of the model (1) described in [3] which were particularly difficult to solve, and four cases of the model (2) described in [2] that also consumed large amounts of computing time. The former set of models that we term hvod has 303 variables (300 binary) and 104 equations, while the latter set, labeled hetvod, have 1504 variables (900 binary) and 706 equations. In all cases the channel parameter setting K=12 was used. While these models are just a subset of all the problems of interest, we note here that they are a representative subset of the hardest models we could generate, and the remaining models still solve as quickly with the modifications we describe here. For completeness, the parameters we set in the different models are given in Table 1. The relevance of most of these parameters was discussed in Section 2. The parameter nbase is the number of multiples of a base disk allocation. The parameters $N_{segments}$ and $N_{channels}$ get multiplied by this factor. For the heterogeneous model, the two numbers represent the multiples for the distinct and homogeneous servers respectively.

Prob Name	k	nbase	λ	β	P	f_d	Opt Soln
hvod1	7	2	1000	0.0	10		68.03
hvod2	5	1	10	0.1	10		47.93
hvod3	5	1	100	0.0	10		58.69
hvod4	5	1	20	0.01	10		51.83
hetvod1	5	(0.25,1)	10	0.01	10	0.0125	451.60
hetvod2	7	(0.375,1.5)	100	0.1	10	0.2	818.95
hetvod3	7	(1,1)	20	0.0	11	0.5	408.48
hetvod4	7	(0.25,1)	90	0.01	10	0.0125	520.51

Table 1: Problem Parameters

In the homogeneous model, considering the LP relaxation as the problem in which the binary constraint on the θ's is replaced by non-negativity constraints (setting upper bounds to 1 is not needed), the resulting LP has $n + 2$ general constraints. Therefore, each basic solution has at most $n + 2$ variables with nonzero values. We now establish that this property in conjunction with the n assignment constraints $\theta_i^{\rho} + \theta_i^{\rho r} + \theta_i^{r} = 1$, implies that at most four of the θ variables will take on fractional values when the linear programming relaxation is solved using the simplex method. In each assignment constraint at least one variable must be non-zero. Identifying one such variable from each of the constraints accounts for n of the non-zero values. In the worst case, there are two additional non-zeros in an LP basic solution, and suppose these are associated with variables in assignment constraints j and k (j and k could be the same index). More than one assignment variable in constraints j and k is non-zero, and hence these variables are fractional, but the remaining $n-2$ (or $n-1$) assignment constraints each contain only one non-zero variable, and thus the value of that variable must be 1. (This argument generalizes in an obvious manner to the heterogeneous case.) Although the solution of the initial LP relaxation is thus essentially integer, we nonetheless found that very large trees were generated by applying branch-and-bound to the original formulation above. This occurred because branching on the non-integer values generally resulted in non-integer values in variables that were formerly 0.

To alleviate this, we added redundant constraints and variables that correspond to total assignments of each type, obtaining the following model for problem (1):

$$\begin{array}{ll} \min\limits_{\theta} & C_{remote}(\theta^{\rho}, \theta^{\rho r}) + P\beta C_{regional}(\theta^{\rho r}, \theta^{r}) \\ \\ \text{subject to} & \begin{array}{c} C_{regional}(\theta^{\rho r}, \theta^{r}) \leq N_{channels} \\ D_{regional}(\theta^{\rho r}, \theta^{r}) \leq N_{segments} \\ \theta_i^{\rho} + \theta_i^{\rho r} + \theta_i^{r} = 1, \quad i = 1, 2, \ldots, n \\ \theta_i^{\rho}, \theta_i^{\rho r}, \theta_i^{r} \in \{0, 1\}, \quad i = 1, 2, \ldots, n \\ \sum_{i=1}^{n} \theta_i^{\rho} = \sigma^{\rho} \\ \sum_{i=1}^{n} \theta_i^{\rho r} = \sigma^{\rho r} \\ \sum_{i=1}^{n} \theta_i^{r} = \sigma^{r} \\ \sigma^{\rho}, \sigma^{\rho r}, \ and\ \sigma^{r}\ non-negative\ and\ integer \end{array} \end{array} \tag{3}$$

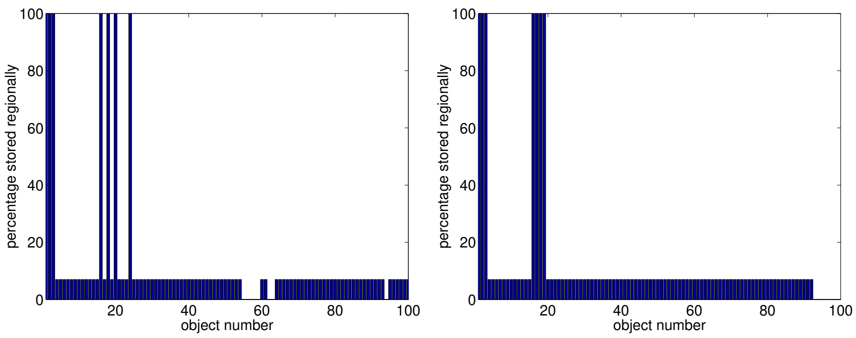
The newly added constraints and variables ($\sigma^{\rho}, \sigma^{\rho r},\ and\ \sigma^{r}$ represent the total number of objects assigned remotely, partially remotely, and regionally) are superfluous from the standpoint of the integer programming model, but serve to tighten the linear programming relaxation and also provide "high-level" integer variables that, when appropriately handled in a branch-and-bound context, tend to force the lower level assignment variables to integer. By giving branching priority to these total allocation variables, many previously feasible (non-integer) relaxations are ruled out, and the branch and bound solution time is considerably reduced. Note that this approach could be extended to include constraints on other significant subsets of the θ variables, but our computational experience has indicated that the subsets chosen were sufficient to drastically reduce tree sizes and computation time in almost all cases. Related modeling tricks can be found in [10].

Similar arguments can be used to motivate the following model to be solved instead of (2):

$$\min_{\theta} \qquad C_{remote}(\theta) + \beta(C^d_{regional}(\theta) + (P-1)C^h_{regional}(\theta))$$

$$\text{subject to} \qquad \begin{array}{c} C^d_{regional}(\theta) \leq N^d_{channels} \\ C^h_{regional}(\theta) \leq N^h_{channels} \\ D^d_{regional}(\theta) \leq N^d_{segments} \\ D^h_{regional}(\theta) \leq N^h_{segments} \\ \sum_{y,z\in\{\rho,\rho r,r\}} \theta_i^{y,z} = 1, \quad i = 1,2,\ldots,n \\ \theta_i^{y,z} \in \{0,1\}, \quad y,z \in \{\rho,\rho r,r\}, i = 1,2,\ldots,n \\ \sum_{i=1}^n \theta_i^{y,z} = \sigma^{y,z}, \quad y,z \in \{\rho,\rho r,r\} \\ \sigma^{y,z}\ non-negative\ and\ integer \end{array} \tag{4}$$

We now detail the effects of this approach on the problems outlined above. We attempted to solve all the problems to optimality using the CPLEX [8] optimizer from within GAMS [1]. The reason for this is that the application is really interested in the optimal cost and the form of the solution. Unfortunately, while the cost function is very flat close to the solution, in many instances the form of the optimal solution is very different. For example, Figure 2 shows two solutions for the problem hetvod1, the first corresponding to a solution with value 453.25, and the second being the optimal solution of value 451.60. In this figure, an object i that is 100% stored regionally corresponds to $\theta_i^r = 1$, an object that is 7% stored regionally corresponds to $\theta_i^{\rho r} = 1$, and an object that is 0% stored regionally corresponds to $\theta_i^\rho = 1$. These figures were



(a) Cost = 453.25 (b) Cost = 451.60 (optimal)

Figure 2: Solution Visualization for hetvod1

produced using the GAMS/MATLAB interface described in [5].

We also show how the lower and upper bounds for a particular problem instance vary over time in Figure 3. These bounds were generated using the default options of CPLEX on (1) for the problem data hvod1.

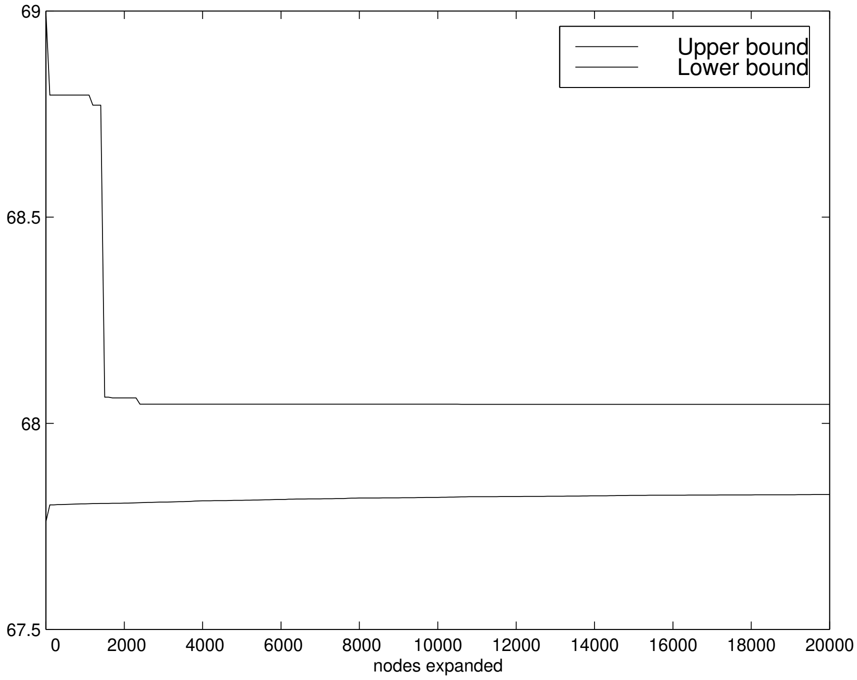


Figure 3: Lower and Upper Bounds for hvod1, default strategy (optimal value=68.03)

We now present two tables showing the options that we used for the two classes of problems. We experimented extensively to determine appropriate parameters, and we note below several key issues that this experimentation uncovered. CPLEX allows

Model	aggregate	priorities	abs tol	rel tol
(1)	∞	no	0	0
(3)	0	no	0	0
(3) + prior	0	yes	0	0

Table 2: CPLEX Options for the hvod problems

priorities to be set on branching variables. The priority ordering we use ensures that variables σ are branched on before variables θ. Our first attempts to use this within CPLEX failed miserably. It transpires that the setting of the aggregator option is critical. By default, CPLEX attempts to aggregate expressions together and remove them from the problem in the preprocessing phase. If we do not prevent this, CPLEX removes the critical redundant constraints we add, and hence cannot use the priority ordering we set.

Independent of the variable we choose to branch on to generate new nodes in the branch and bound tree is the node that we choose to expand next. After some experimentation, the default setting of CPLEX appears to be best for these problems. We attempted to solve all of the given hvod problems to optimality.

We used the options shown in Table 3 for the hetvod problems. As far as we

Model	MIP Preprocessing	aggregate	priorities	abs tol	rel tol
(2)	no	∞	no	1e-6	1e-7
(4)	no	0	no	1e-6	1e-7
(4) + prior	no	0	yes	1e-6	1e-7

Table 3: CPLEX Additional Options for hetvod

could discern, there is no option to turn off MIP preprocessing in CPLEX. What we actually implemented was two solves of the model in hand, the first as a relaxed MIP using the primal simplex method, followed by a solve using the MIP solver with the options given in the table. It appears that when an advanced basis is presented to CPLEX (from the relaxed MIP solve), the MIP solver decides to turn off its MIP preprocessing. While CPLEX may change other options in this context that we are unaware of, the effect on our problems of this procedure was remarkable and allowed the hetvod examples to be solved much more efficiently.

Table 4 summarizes our computational results. The columns of this table correspond to the different models and option settings given in Table 2 and Table 3. The times reported in the table include the GAMS model generation time and the solution time using CPLEX 6.0 running on an Ultrasparc Solaris Workstation. We set a time limit of 1000 seconds on each run; in the case that this time limit was reached, we report the absolute gap determined by CPLEX at this point in parentheses. It should be noted that the times for the hetvod models are considerably larger if MIP preprocessing is used.

Model	(1) or (2)		(3) or (4)		(3) or (4) + prior	
	Time	Nodes	Time	Nodes	Time	Nodes
hvod1	(0.178)	119184	(0.170)	104609	7.35	1711
hvod2	(0.405)	159961	(0.084)	116276	0.07	6
hvod3	(0.289)	52218	(0.005)	135256	0.07	5
hvod4	(0.605)	76995	(0.004)	121925	0.05	7
hetvod1	(0.245)	45391	(0.311)	46199	0.88	85
hetvod2	82.05	16433	27.63	3132	34.37	7854
hetvod3	283.96	82933	309.13	88290	19.19	4387
hetvod4	6.24	706	0.79	157	1.10	202

Table 4: Solution Efficiency Using CPLEX

With the only exceptions being hetvod2 and hetvod4, the expanded models (3) and (4) with priorities set on the added variables significantly outperformed all other techniques. In the latter two cases, just using the expanded models performed slightly better than additionally using priorities. In the first five cases reported in the table, we were able to solve hitherto unsolved instances in less than 8 seconds. The reduction in the size of the search trees for the hvod problems in particular appears astonishing.

Note that adding the constraints to the basic model typically reduces the optimality gap, but does not lead to finding solutions when the original formulation failed.

In all the testing above we used the CPLEX solver. To show that the extra constraints and the use of priorities makes the problems generically easier, we used a different solver, OSL [7] to solve the problems. Apart from setting priorities, OSL was used with default settings. The results given in Table 5 show similar behavior to that given by CPLEX on these models when priorities are used. We note that some variation in the size of the search tree is to be expected due to vastly different implementations.

Model	(3) or (4) + prior	
	Time	Nodes
hvod1	134.26	14326
hvod2	0.26	3
hvod3	0.20	2
hvod4	0.26	3
hetvod1	12.90	726
hetvod2	32.46	1688
hetvod3	55.97	3522
hetvod4	8.87	460

Table 5: Solution Efficiency Using OSL

4 Conclusion

The delivery of large data objects such as movies over a hierarchical network poses interesting and important applications for combinatorial optimization. In this paper we focused on a particular data segmentation approach, skyscraper delivery, that is particularly well-suited to current networks. For two variants of this problem, we formulated the problem of minimizing the critical performance measure, total bandwidth consumed, as generalized assignment problems. From a branch-and-bound solution viewpoint this formulation proved unacceptable, since it often resulted in enormous solution times or failure to achieve optimality (even after thousands of seconds and hundreds of thousands of nodes in the branch-and-bound tree). We then augmented the original formulations by redundant constraints and variables corresponding to assignment subtotals, and gave the corresponding new variables branching priority. These augmented models combined with the appropriate variable selection priority allowed us to solve all of the problems of interest in one minute. We are continuing research into generalizations of this approach to other data delivery architectures.

Acknowledgements

This material is based on research supported in part by National Science Foundation Grant CCR-9619765 and Air Force Office of Scientific Research Grant F49620-98-1-0417. We would like to thank Mary Vernon and Derek Eager for describing the original mathematical models, explaining facets of the problem formulation and interpreting many of the model results.

References

[1] A. Brooke, D. Kendrick, and A. Meeraus. *GAMS: A User's Guide*. The Scientific Press, South San Francisco, CA, 1988.

[2] D. L. Eager, M. C. Ferris, and M. K. Vernon. Optimized regional caching in heterogeneous systems. Mathematical Programming Technical Report 98-15, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin, 1998.

[3] D. L. Eager, M. C. Ferris, and M. K. Vernon. Optimized regional caching for on-demand data delivery. In *Multimedia Computing and Networking, Proceedings of SPIE*, volume 3654, Bellingham, Washington, 1999.

[4] D. L. Eager and M. K. Vernon. Dynamic skyscraper broadcasts for video-on-demand. In *Proceedings of the 4th International Workshop on Multimedia Information Systems (MIS '98)*, pages 18–32, Istanbul, Turkey, 1998.

[5] M. C. Ferris. MATLAB and GAMS: Interfacing optimization and visualization software. Mathematical Programming Technical Report 98-19, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin, 1998.

[6] K. A. Hua and S. Sheu. Skyscraper broadcasting: A new broadcasting system for metropolitan video-on-demand systems. In *Proceedings of the ACM SIGCOMM'97 Conference*, pages 89–100, Cannes, France, 1997.

[7] Ming S. Hung, Walter O. Rom, and Allan D. Warren. *OSL Handbook*. Boyd and Fraser, 1995.

[8] ILOG CPLEX Division, Incline Village, Nevada. *CPLEX Version 6.0.* http://www.cplex.com/.

[9] G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization.* Wiley, New York, NY, 1988.

[10] H. P. Williams. *Model Building in Mathematical Programming.* John Wiley & Sons, third edition, 1990.