A Novel Optimization Algorithm for Video Placement and Routing

Tolga Bektaş, Osman Oğuz, and Iradj Ouveysi

***Abstract*— In this paper, we propose a novel optimization algorithm for the solution of the video placement and routing problem based on Lagrangean relaxation and decomposition. The main contribution can be stated as the use of integer programming models to obtain feasible solutions to the problem within the algorithm. Computational experimentation reveals that the use of such integer models help greatly in obtaining good quality solutions in a small amount of solution time.**

***Index Terms*— Video-on-demand, placement, routing, integer programming, Lagrangean relaxation, decomposition.**

I. Introduction

VIDEO on Demand (VoD) is a service that provides tens to hundreds of videos (programs) to hundreds to thousands of clients through a network. Commercial VoD services are now available in many areas that the multimedia technologies are developing very fast. An in-depth treatment of the subject is given by Little and Venkatesh [5]. A central problem in structuring a VoD system is load balancing, which can be further separated into two subproblems as indicated by Little and Venkatesh [5]. The first consists of deciding on program allocation and the second is resource location and connection establishment. For relevant studies on VoD, we refer the reader to Kim *et al.* [3], Lee [4], Chan [1], Ouveysi *et al.* [6] and Xue [7].

The main motivation in this paper is to develop a novel optimization algorithm for the solution of the integer linear programming model introduced by Ouveysi *et al.* [6] for the so called Video Placement and Routing Problem (VPRP). In the next section, we formally define the problem and present the integer linear programming model. Section III provides the details of the optimization algorithm, with the implementation results on randomly generated instances given in Section IV. Conclusions and further remarks are given in Section V.

II. Problem Definition and Model

In defining the problem, we adhere to the notation of Ouveysi *et al.* [6], given as follows. There exists a fully meshed network modelled by an undirected graph $G = (V, A)$, where $V = \{1, 2, ..., n\}$ is the set of nodes and A is the set of edges including the $n(n-1)/2$ links of the network. The set of programs (videos) to be placed and routed is denoted by $P = \{1, 2, ..., m\}$, where each program $k \in P$ has a capacity requirement denoted by m_k and a bandwidth requirement for transmission denoted by μ_k. Each node $j \in V$ corresponds to a potential location for storing the programs with capacity denoted by $C_s(j)$. In addition, each node has a demand for each program. The cost of storing a program $k \in P$ at node $j \in V$ is shown by $s_k(j)$ and the transmission cost of the program on the link $\{i, j\} \in A$ is shown by $t_k(i, j)$. Finally, each link $\{i, j\} \in A$ has a transmission capacity that is denoted by $C_t(i, j)$. Given the demand forecast of the programs, the VPRP consists of finding a placement scheme for the programs such that the total cost of storage and transmission of the programs in the network is minimized and the demand of each node for each program is satisfied.

Ouveysi *et al.* [6] previously proposed a model for the VPRP, which uses a binary variable x_{ij}^k that is equal to 1 if program $k \in P$ is transmitted to node $j \in V$ from node $i \in V$ and 0 otherwise and an additional binary variable y_j^k that is equal to 1 if program $k \in P$ is stored at node $j \in V$ and 0 otherwise. The model (denoted by **F**) is as follows :

$$\text{minimize} \sum_{k\in P}\sum_{j\in V} s_k(j) y_j^k + \sum_{k\in P}\sum_{j\in V}\sum_{i\in V, i\neq j} t_k(i,j) x_{ij}^k \quad (1)$$

s.t.

$$\sum_{k\in P} m_k y_j^k \leq C_s(j), \quad \forall j \in V \quad (2)$$

$$\sum_{k\in P} \mu_k x_{ij}^k \leq C_t(i,j), \quad \forall i \neq j \in V \quad (3)$$

$$\sum_{i\in V, i\neq j} x_{ij}^k + y_j^k = 1, \quad \forall j \in V, k \in P \quad (4)$$

$$x_{ij}^k \leq y_i^k, \quad \forall i \neq j \in V, k \in P \quad (5)$$

$$x_{ij}^k, y_j^k \in \{0,1\}, \quad \forall i \neq j \in V, k \in P \quad (6)$$

In this model, constraints (2) and (3) correspond to the capacity constraints related with storage nodes and transmission links, respectively. Constraints (4) state that each node either stores a program or receives it from another node that stores it. Finally, constraints (5) imply that a program can be transmitted from a node only if the program is stored at that node. Constraints (6) impose integrality restrictions on the decision variables. **F** has in general $(n^2m+nm)/2$ binary variables and $(n^2m + n^2 + n)/2$ constraints. As n and m increase, the size of **F** will render the solution of the problem using standard off-the-shelf software impractical. Ouveysi *et al.* [6] have proposed a heuristic to solve problem **F**. In this letter, we propose a novel optimization algorithm for problem **F** based on Lagrangean relaxation and decomposition. Details of our algorithm are presented in the next section.

Manuscript received July 18, 2005. The associate editor coordinating the review of this letter and approving it for publication was Prof. Hamid Jafarkhani.

T. Bektaş and O. Oğuz are with the Dept. of Industrial Engineering, Bilkent University, Ankara, Turkey (e-mail: {tolgab, ooguz}@bilkent.edu.tr).

I. Ouveysi is an honorary fellow in the EEE Dept., University of Melbourne, Victoria, Australia (e-mail: iradjouveysi@yahoo.co.uk).

Digital Object Identifier 10.1109/LCOMM.2006.02007.

1089-7798/06$20.00 © 2006 IEEE

III. The Optimization Algorithm

Our algorithm is based on relaxing the capacity constraints (2) and (3) in a Lagrangean fashion, by respectively associating the Lagrange multipliers β_j and α_{ij} to each constraint. As a result, we obtain the following relaxed problem (denoted by $F(\beta,\alpha)$):

$$\begin{aligned} \text{minimize } & \sum_{k\in P}\sum_{j\in V} A_j^k y_j^k + \sum_{k\in P}\sum_{j\in V}\sum_{i\in V, i\neq j} B_j^k x_{ij}^k - C_0 \\ \text{s.t. } & \qquad (4),(5),(6) \end{aligned}$$

where $A_j^k = s_k(j) + \beta_j m_k$, $B_j^k = t_k(i,j) + \alpha_{ij}\mu_k$ and $C_0 = \sum_{j\in V}\beta_j C_s(j) + \sum_{i\in V}\sum_{j\in V}\alpha_{ij}C_t(i,j)$. Next, we observe that $F(\beta,\alpha)$ decomposes into $|P|$ subproblems, one for each program $k\in P$ and each denoted by $F_k(\beta,\alpha)$. Each subproblem has $(n^2+n)/2$ binary variables and $(n^2+n)/2$ constraints. Let $v(\mathbf{F})$ denote the optimal objective function value of problem $\mathbf{F}$. Then, as a result of the decomposition, the optimal objective function of the formulation $F(\beta,\alpha)$ can be calculated as $v(F(\beta,\alpha)) = \sum_{k\in P} v(F_k(\beta,\alpha)) - C_0$. We now provide below a general outline of the algorithm that can be used to solve problem $\mathbf{F}$:

The Algorithm:

- Start with an initial vector of multipliers β^1, α^1. Let the incumbent lower bound be $lb = -\infty$, incumbent upper bound be $ub = \infty$ and $t = 1$.
- Perform the following until $gap = \frac{ub-lb}{ub} < 1.00$ or the maximum number of iterations have been reached.
  - Solve $F(\beta^t,\alpha^t)$. Set $lb = v(F(\beta^t,\alpha^t))$ if $v(F(\beta^t,\alpha^t)) > lb$.
  - Modify the solution of $F(\beta^t,\alpha^t)$ into a feasible solution $\widehat{F}(\beta^t,\alpha^t)$ using the two-stage procedure that will be described below. If $v(\widehat{F}(\beta^t,\alpha^t)) < ub$, set $ub = v(\widehat{F}(\beta^t,\alpha^t))$.
  - Update the multipliers as $\beta^{t+1} = max\{0, \beta^t + s_1^t \cdot g_1^t\}$ and $\alpha^{t+1} = max\{0, \alpha^t + s_2^t \cdot g_2^t\}$, where g_1^t and g_2^t are the subgradient vectors. The j^{th} component of g_1^t is defined as $(g_1^t)_j = \sum_{k\in P} m_k y_j^k - C_s(j)$ and the $(i,j)^{th}$ component of g_2^t is defined as $(g_2^t)_{ij} = \sum_{k\in P}\mu_k x_{ij}^k - C_t(i,j)$. In updating the multipliers, the step-sizes s_1^t and s_2^t are calculated as follows [2]:

$$s_i^t = \lambda\frac{1.05\cdot ub - v(F(\beta^t,\alpha^t))}{\|g_i^t\|^2}, \quad i = 1,2 \qquad (7)$$

  - $t \longleftarrow t+1$.
- Output ub as the best feasible solution.

In equation (7), λ is a convergence parameter. We note that if the optimal solution to problem $\mathbf{F}$ is equal to that of its LP relaxation (i.e. all the variables are relaxed to be continuous in the interval [0,1]), then the algorithm asymptotically converges to the optimal solution. If this is not the case, this indicates that there exists a duality gap and the algorithm may not be able to find the optimal solution to the problem. However, it is often the case that the algorithm is stopped when a good quality solution is found. The reader is referred to Held *et al.* [2] for details on the convergence of the algorithm. The *gap* calculated at each iteration of the algorithm shows how far at most the current feasible solution may be away from the optimal solution. Therefore, in the case that the algorithm is unable to find the optimal solution, this value is an indicator of the quality of the final solution.

At any step of the algorithm, the solution of $F(\beta^t,\alpha^t)$ provides an integral solution that is feasible with respect to constraints (4) and (5), but may not necessarily satisfy the capacity constraints (2) and (3). This (infeasible) solution needs to be converted into a feasible solution with respect to problem F in order to be able to provide the algorithm with an upper bound. Given an optimal solution $\widehat{y}_j^k$ and $\widehat{x}_{ij}^k$ to $F(\beta,\alpha)$, we attempt to achieve feasibility using a two stage procedure, described below:

1) Stage 1: The first stage consists of modifying the $\widehat{y}_j^k$'s through repositioning any program that violates the capacity constraint, such that the resulting solution satisfies constraint (2). To achieve this, we define the set $O(j,k) = \{j\in V, k\in P | \widehat{y}_j^k = 1\}$. Then, the feasibility is accomplished through the use of the following feasibility integer programming model (denoted by **FeasY**):

$$\begin{aligned} (\mathbf{FeasY}) \quad \text{minimize } & \sum_{k\in P}\sum_{j\in V} s_k(j) y_j^k + \sum_{k\in P}\sum_{j\in V} R m_j^k && (8) \\ \text{s.t. } & \\ & \sum_{k\in P} m_k y_j^k \le C_s(j), \quad \forall j\in V && \\ & y_j^k \ge 1 - m_j^k, \quad \forall j,k \in O(j,k) && (9) \\ & \sum_{j\in V} y_j^k \ge 1, \quad \forall k\in P && (10) \\ & y_j^k \in \{0,1\}, \quad \forall j\in V, k\in P && \\ & m_j^k \in \{0,1\}, \quad \forall j,k\in O(j,k) && (11) \end{aligned}$$

In **FeasY**, the additional binary variable m_j^k is equal to one if program k on node j is repositioned to another node. However, to benefit as much as possible from the current solution, we would like the amount of modification performed to this solution to be minimal. Therefore, each modification is penalized in the integer program, which is reflected in the second summation of the objective function (8) with a penalty coefficient R. Here, R is a sufficiently large constant (e.g. $R >> max_{k\in P, j\in V}\{s_k(j)\}$). Constraints (9) stipulate that if a program k already located at node j is repositioned to another node, then $y_j^k = 0$. Constraints (10) are used to ensure that after the modification, each program is located on at least one node. We also note that in **FeasY**, it is possible to relax the binary variables m_j^k in the interval $[0,1]$, since it is easy to see that in any optimal solution to **FeasY**, no m_j^k will attain a fractional value. In short, the solution to **FeasY** yields a placement scheme for the programs such that no node constraint is violated.

2) Stage 2: In the second stage, we attempt to find a feasible configuration of x_{ij}^k variables, based on the optimal values of the variables $\overline{y}_j^k$ of **FeasY**. In other words, we would like to obtain a vector of x variables satisfying the following integer model (henceforth referred to as **FeasX**):

$$(\textbf{FeasX}) \qquad \text{minimize} \sum_{k\in P}\sum_{j\in V}\sum_{i\in V, i\neq j} t_k(i,j)x_{ij}^k \quad (12)$$

s.t.

$$\sum_{k\in P} m_k x_{ij}^k \leq C_t(i,j), \qquad \forall i\neq j\in V : \overline{y}_i^k = 1, \overline{y}_j^k = 0$$

$$\sum_{i\in V, i\neq j, \overline{y}_i^k=1} x_{ij}^k = 1, \qquad \forall j \in V, k\in P : \overline{y}_j^k = 0 \quad (13)$$

$$x_{ij}^k \in \{0,1\}, \qquad \forall i\neq j\in V : \overline{y}_i^k = 1, \overline{y}_j^k = 0, k\in P \quad (14)$$

Note that in model **FeasX**, the binary variables x_{ij}^k are only defined if $\overline{y}_i^k = 1$ and $\overline{y}_j^k = 0$. Therefore, the size of the model is greatly reduced as compared to problem **F**. The optimal solution to **FeasX** yields a feasible configuration of x_{ij}^k variables. As a result of stages 1 and 2, we obtain the optimal objective value of the corresponding (feasible) solution for problem **F** as $v(\textbf{FeasY}) + v(\textbf{FeasX}) - \sum_{k\in P}\sum_{j\in V} Rm_j^k$.

IV. Numerical Results

The optimization algorithm proposed in this paper has been implemented in C and all the tests are performed on a Sun UltraSPARC 12x400 MHz with 3 GB RAM on randomly generated instances. All the subproblems are solved to optimality using CPLEX 9.0. Parameters $\mu_k, t_k(i,j), s_k(j)$ are randomly generated from a uniform distribution between 1 and 100. m_k is modelled as $m_k = \mu_k T_k$, where T_k is the total transmission time for program k. In the experiments, $T_k = 10$ min. for all k. $C_t(i,j)$ values have been chosen from the uniform distribution between $max_{i,j}\{t_k(i,j)\}$ and $\sum_{i,j}\{t_k(i,j)\}$). The capacity of each node $(C_s(j))$ is set to be 40% of the total size of all the programs and the penalty parameter R is set to $10max_{k\in P, j\in V}\{s_k(j)\}$. The convergence parameter λ is initially set to 2.00 and multiplied by 0.87 if there is not any improvement in the best known upper bound for 5 consecutive iterations. The algorithm is stopped when the given time limit has been reached.

We present the results in Table I, where each row contains the average values of five randomly generated problems. Columns n and m respectively show the number of nodes and the number of programs, column n_L denotes number of iterations required by the algorithm, t_S indicates average time required to solve all the subproblems to optimality. Columns t_Y and t_X show the average time required to solve **FeasY** and **FeasX** to optimality, respectively. Columns g_i and g_f report the average initial and final gaps obtained by the algorithm, respectively. In order to show the effectiveness of the algorithm, we compare it with CPLEX. To be fair in comparisons, we impose a common time limit of 300 seconds on both algorithms, considering the dynamic nature of the problem requiring repeated resolving to adopt to the changes in the demand pattern and available programs. The last column of Table I, column d, shows the average percent difference between the best solution found by the proposed algorithm (denoted by v_{opt}) and that of CPLEX (denoted by v_C) within the given time limit, and calculated as $\frac{v_{opt}-v_C}{v_{opt}} \cdot 100$.

TABLE I

Numerical results for the optimization algorithm

n	m	n_L	t_S	t_Y	t_X	g_i	g_f	d
50	20	13.2	13.78	0.12	0.53	3.23	2.03	0.82
60	20	12.4	23.91	0.15	0.78	4.04	2.20	0.91
70	20	7	36.45	0.17	1.08	3.37	1.82	0.12
80	20	6.4	58.20	0.20	1.39	2.92	2.45	-0.30
50	30	16.4	15.93	0.18	0.97	4.01	2.70	0.98
60	30	10	30.02	0.23	1.37	3.01	2.27	0.72
70	30	6.4	52.20	0.25	2.02	2.95	2.58	-2.74
80	30	4	76.55	0.31	2.47	2.65	1.99	-5.08
50	40	10.6	26.72	0.24	1.49	4.43	2.81	0.53
60	40	7.6	40.64	0.27	2.02	2.95	2.38	-0.56
70	40	5.4	61.94	0.34	2.77	2.56	2.26	-3.63
80	40	3.4	98.26	0.41	3.43	2.15	1.72	-4.80
50	50	9.4	33.75	0.30	2.01	3.28	2.67	-0.10
60	50	5.6	58.94	0.35	2.93	3.38	2.60	-2.61
70	50	4	90.00	0.40	3.83	2.92	2.56	-3.89
80	50	3	139.18	0.47	4.89	2.24	2.02	-4.07

As can be seen from the results given in Table I, the proposed algorithm is able to output solutions of good quality (typically around 2% of the optimal) in a short time. In fact, the algorithm is capable of obtaining near-optimal solutions even in the first iteration. The values shown under columns t_Y and t_X indicate that it is not computationally expensive to obtain feasible solutions at each iteration of the algorithm. In addition, the proposed algorithm is observed to be capable of providing better solutions than those found by CPLEX in the same amount of time, especially as the instances grow in size.

V. Concluding Remarks

In this letter, we have presented a novel optimization algorithm for the resolution of the VPRP. The proposed algorithm is different from similar existing algorithms because we achieve the feasible solutions through the use of integer programming techniques and this is the reason that our solution methodology would result in good quality solutions even at the earlier iterations of the algorithm. This can be stated as the main contribution of this paper. Our algorithm also provides a benchmark to measure the quality of the output results. The algorithm may be useful as a tool to VoD service providers in efficiently planning and managing their services.

References

[1] S.-H. Gary Chan, "Operation and cost optimization of a distributed servers architecture for on-demand video services," *IEEE Commun. Lett.*, vol. 5, pp. 384-386, Sept. 2001.

[2] M. Held, P. Wolfe, and H. P. Crowder, "Validation of subgradient optimization," *Mathematical Programming*, vol. 6, pp. 62-88, Dec. 1974.

[3] Y. K. Kim, J. Y. Kim, and S. S. Kang, "A tabu search approach for designing a non-hierarchical video-on-demand network architecture," *Computers and Industrial Engineering*, vol. 33, pp. 837-840, Dec. 1997.

[4] J. Y. B. Lee, "UVoD: an unified architecture for Video-on-Demand services," *IEEE Commun. Lett.*, vol. 3, pp. 277-279, Sept. 1999.

[5] T. D. C. Little and D. Venkatesh, "Prospects for interactive video-on-demand," *IEEE Multimedia*, vol. 1, pp. 14-24, Autumn/Fall 1994.

[6] I. Ouveysi, L. Sesana, and A. Wirth, "The video placement and routing problem," in E. Kozan, A. Ohuchi, (eds)., *Operations Research / Management Science at Work: Applying Theory in the Asia Pacific Region*. Kluwer Academic Publishers International Series in Operations Research and Management Science, pp. 53–71, 2002.

[7] G. Xue, "Server cost minimization in a distributed servers architecture for on-demand video services," *IEEE Commun. Lett.*, vol. 7, pp. 52-54, Feb. 2003.