A Novel Optimization Algorithm for Video Placement and Routing

Tolga Bektas, 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 the set of nodes and $A$ is the set of edges including the 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 $\mu_k$. 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 $\{i, j\} \in E$ 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:

\[
\begin{align*}
\text{minimize} & \quad \sum_{k \in P} \sum_{j \in V} s_k(j)y_j^k + \sum_{k \in P} \sum_{j \in V} \sum_{i \in V, i \neq j} t_k(i, j)x_{ij}^k \\
\text{s.t.} & \quad \sum_{k \in P} m_ky_j^k \leq C_s(j), \quad \forall j \in V \quad (1) \\
& \quad \sum_{k \in P} \mu_kx_{ij}^k \leq C_t(i, j), \quad \forall i \neq j \in V \quad (2) \\
& \quad x_{ij}^k + y_j^k = 1, \quad \forall j \in V, k \in P \quad (3) \\
& \quad x_{ij}^k \leq y_j^k, \quad \forall i \neq j \in V, k \in P \quad (4) \\
& \quad x_{ij}^k, y_j^k \in \{0, 1\}, \quad \forall i \neq j \in V, k \in P \quad (5)
\end{align*}
\]

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. Bektas 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$02.00 © 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 $\beta_j$ and $\alpha_{ij}$ to each constraint. As a result, we obtain the following relaxed problem (denoted by $F(\beta, \alpha)$):

\[
\text{minimize} \sum_{k \in P} \sum_{j \in V} A_j^k y_j^k + \sum_{k \in P} \sum_{j \in V} B_j^k x_{ij}^k - C_0
\]
\[\text{s.t.} \quad (4), (5), (6)\]

where $A_j^k = s_k(j) + \beta_j m_k$, $B_j^k = t_k(i, j) + \alpha_{ij} h_k$ and $C_0 = \sum_{j \in V} \beta_j s_k(j) + \sum_{i \in V} \sum_{j \in V} \alpha_{ij} C_i(i, j)$. 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 $v(F)$ denote the optimal objective function value of problem $F$. 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 $F$:

The Algorithm:

- Start with an initial vector of multipliers $\beta^0, \alpha^0$. 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 $\hat{F}(\beta^t, \alpha^t)$ using the two-stage procedure that will be described below. If $v(\hat{F}(\beta^t, \alpha^t)) < ub$, set $ub = v(\hat{F}(\beta^t, \alpha^t))$.
  - Update the multipliers as $\beta^{t+1} = \max\{0, \beta^t + s_1 \cdot g_1^t\}$ and $\alpha^{t+1} = \max\{0, \alpha^t + s_2 \cdot g_2^t\}$, 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 $(g_2^t)_{ij} = \sum_{k \in P} h_k x_{ij}^k - C_l(i, j)$. In updating the multipliers, the step-sizes $s_1^t$ and $s_2^t$ are calculated as follows [2]:
    \[
s_1^t = \frac{1.05 \cdot ub - v(F(\beta^t, \alpha^t))}{\|g_1^t\|^2}, \quad i = 1, 2
\]
  - $t = t + 1$.
- Output $ub$ as the best feasible solution.

In equation (7), $\lambda$ is a convergence parameter. We note that if the optimal solution to problem $F$ 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 $\hat{y}_j^k$ and $\hat{x}_{ij}^k$ 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 $\hat{y}_j^k$’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 $O(j, k) = \{j \in V, k \in P | \hat{y}_j^k = 1\}$. Then, the feasibility is accomplished through the use of the following feasibility integer programming model (denoted by FeasY):

\[
\text{minimize} \sum_{k \in P} \sum_{j \in V} s_k(j) y_j^k + \sum_{k \in P} \sum_{j \in V} Rm_j^k
\]
\[\text{s.t.} \quad \sum_{k \in P} m_k y_j^k \leq C_s(j), \quad \forall j \in V \]
\[y_j^k \geq 1 - m_j^k, \quad \forall j, k \in O(j, k) \]
\[\sum_{j \in V} y_j^k \geq 1, \quad \forall k \in P \]
\[y_j^k \in \{0, 1\}, \quad \forall j \in V, k \in P \]
\[m_j^k \in \{0, 1\}, \quad \forall j, k \in O(j, k) \]

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 $y_j^k$ 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):
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 $\mu_k$, $t_k(i,j)$, and $s_k(j)$ are randomly generated from a uniform distribution between 0 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 $k$. $C_l(i,j)$ 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 ($C_u(j)$) is set to be 40% of the total size of all the programs and the penalty parameter $R$ is set to $10 \max_{k \in P, i \in V} \{ s_k(j) \}$. The convergence parameter $\lambda$ is initially set to 2.00 and multiplied by 0.87 if there is no 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 $\frac{v_C - v_{opt}}{v_{opt}} \cdot 100$.

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.

TABLE I

<table>
<thead>
<tr>
<th>$n$</th>
<th>$m$</th>
<th>$n_L$</th>
<th>$t_S$</th>
<th>$t_Y$</th>
<th>$t_X$</th>
<th>$g_i$</th>
<th>$g_f$</th>
<th>$d$</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>12</td>
<td>400</td>
<td>4.00</td>
<td>0.20</td>
<td>2.50</td>
<td>1.20</td>
<td>4.00</td>
<td>2.00</td>
</tr>
<tr>
<td>50</td>
<td>20</td>
<td>100</td>
<td>1.00</td>
<td>0.05</td>
<td>1.50</td>
<td>0.50</td>
<td>1.00</td>
<td>0.50</td>
</tr>
<tr>
<td>80</td>
<td>20</td>
<td>200</td>
<td>2.00</td>
<td>0.20</td>
<td>3.00</td>
<td>1.00</td>
<td>2.00</td>
<td>1.00</td>
</tr>
<tr>
<td>30</td>
<td>10</td>
<td>500</td>
<td>3.00</td>
<td>0.30</td>
<td>5.00</td>
<td>1.50</td>
<td>3.00</td>
<td>1.50</td>
</tr>
<tr>
<td>50</td>
<td>10</td>
<td>1000</td>
<td>5.00</td>
<td>0.50</td>
<td>8.00</td>
<td>2.00</td>
<td>4.00</td>
<td>2.00</td>
</tr>
</tbody>
</table>

REFERENCES