Optim Lett (2017) 11:225–239
DOI 10.1007/s11590-016-1005-9

SHORT COMMUNICATION

A cutting plane approach to combinatorial bandwidth packing problem with queuing delays

Sachin Jayaswal[1] · Navneet Vidyarthi[2] ·
Sagnik Das[3]

Received: 22 December 2014 / Accepted: 17 January 2016 / Published online: 9 February 2016

Abstract The combinatorial bandwidth packing problem (CBPP), arising in a telecommunication network with limited bandwidth, is defined as follows. Given a set of requests, each with its potential revenue, and each consisting of calls with their bandwidth requirements, decide: (1) a subset of the requests to accept/reject; and (2) a route for each call in accepted requests, so as to maximize the total revenue earned in a telecommunication network with limited bandwidth. However, telecommunication networks are generally characterized by variability in the call (bits) arrival rates and service times, resulting in queuing delays in the network. In this paper, we present a non-linear integer programming model to account for such delays in CBPP. Using simple transformation and piecewise outer-approximation, we reformulate the model as a linear mixed integer program (MIP), but with a large number of constraints. We present an efficient cutting plane approach to solve the resulting linear MIP to ϵ-optimality.

✉ Sachin Jayaswal
sachin@iimahd.ernet.in

Navneet Vidyarthi
navneetv@jmsb.concordia.ca

Sagnik Das
sdas18@illinois.edu

1 Production and Quantitative Methods, Indian Institute of Management Ahmedabad, Ahmedabad, India

2 Department of Supply Chain and Business Technology Management, John Molson School of Business, Concordia University, Montreal, Canada

3 Department of Industrial and Enterprise Systems Engineering, University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA

Keywords OR in telecommunications · Bandwidth packing · Integer programming · Queueing · Cutting plane algorithm

1 Introduction

The recent advent of bandwidth intensive telecommunication services, like video conferencing, social networking, online interactive gaming, interactive television, real-time simulations, etc., makes the efficient management of the available telecommunication network bandwidth critical. In most of such applications, users generally schedule their requests in advance [14]. In this respect, a critical problem that network administrators commonly face is defined as follows. Given a set of submitted call requests, each with its potential revenue and bandwidth requirement, decide: (1) a subset of the calls to accept/reject; and (2) a route for each accepted call, so as to maximize the total revenue earned in a telecommunication network with limited bandwidth. This is the classical version of what is referred to in the literature as the bandwidth packing problem (BPP) [6,8,12–14]. Several variants of BPP have been widely studied in the literature. Amiri [2] presents a version of BPP involving scheduling of selected calls within given time windows. A multi-hour version of BPP, which accounts for the variation in traffic between peak and off-peak hours of the day, is studied by Amiri and Barkhi [3].

The objective of maximizing revenue in a BPP results in accepting as many requested calls as allowed by the limited bandwidth of the network. This may result in a very high bandwidth utilization on some of the links in the network. This, in the presence of usually bursty bandwidth requirements (variable bit rates) of telecommunication calls involving data and video transmission, can cause excessive (queuing) delays in the network. Amiri et al. [5], Han et al. [11], Rolland et al. [15], Vidyarthi et al. [16] explicitly account for such network delays by incorporating queuing delay terms in their models. Most of these papers model the links on the network as independent M/M/1 queues with the implicit underlying assumption of Poisson bit (of calls) arrivals and exponential service times (on the links). While Amiri et al. [5], Vidyarthi et al. [16] discourage such delays in their model by penalizing them in the objective function, Rolland et al. [15], Han et al. [11] impose an explicit constraint on the maximum delay. Bose [7] presents a two-priority version of BPP, wherein the calls of lower priority consume less bandwidth, but are longer and generate lower revenue compared to those of higher priority. For this, he models each link as a preemptive priority M/M/1 queue. Amiri [1] extends the multi-hour BPP, earlier studied by Amiri and Barkhi [3], with delay guarantees. Gavish and Hantler [10] also present BPP with queuing delays, although the acceptance/rejection of calls is not a decision in their problem.

Solution methods for BPP and its variants reported in the literature can be broadly classified into two groups: metaheuristics and decomposition based approaches. Within metaheuristics, Tabu Search has been applied by Anderson et al. [6], Laguna and Glover [12], while Cox et al. [8] have reported the use of genetic algorithm. Within decomposition based approaches, Lagrangian relaxation has been a popular choice, reported by Gavish and Hantler [10], Rolland et al. [15], Amiri et al. [5], Amiri and

Barkhi [3,4], Amiri [1,2], Vidyarthi et al. [16], besides column generation [13,14,17], and Dantzig–Wolfe decomposition [11].

An interesting generalization of the classical BPP presented by Amiri and Barkhi [4] considers the case where each request consists of a set of calls, requiring one-to-many or many-to-many connections. The resulting problem, referred to as combinatorial bandwidth packing problem (CBPP), is defined as follows. Given a set of requests, each with its potential revenue, and each consisting of calls with their bandwidth requirements, decide: (1) a subset of the requests to accept/reject; and (2) a route for each call in accepted requests, so as to maximize the total revenue earned in a telecommunication network with limited bandwidth. CBPP is NP-hard since it is a combinatorial extension of BPP, which is itself NP-hard.

CBPP, like BPP as discussed above, may also result in very high bandwidth utilization on certain links, potentially causing delays in the network. In this paper, we, therefore, extend CBPP to account for queuing delays on the links. For this, links in the network are modelled as independent single server queues with Poission arrivals and exponential service times (M/M/1 queues). This results in a non-linear integer programming (IP) model. We, therefore, reformulate the model, using simple transformation and piecewise linear outer-approximation, as a linear mixed integer program (MIP), but with a large number of additional constraints. To handle this, we present an efficient cutting plane based solution approach, in which we start with a subset of these cuts/constraints, and generate the rest as needed. At every iteration of the algorithm, we solve the linear MIP with a subset of these cuts to obtain a lower and an upper bound to the original problem. These bounds are tightened as cuts are generated and appended to the model. The algorithm terminates when the gap between the upper and lower bounds is obtained within a desired tolerance (ϵ). Computational results show that the proposed approach performs very well in terms of optimality gap and computational time.

The remainder of the paper is organized as follows. In Sect. 2, we formally describe the problem and present its non-linear IP formulation. Section 3 describes an approach to linearize the model, followed by an exact cutting plane based method to solve the resulting linear MIP. Computational results are reported in Sect. 4. Section 5 concludes with possible directions for future research.

2 Problem formulation

Consider a telecommunication network comprising nodes, indexed by $\{i \in N\}$ or $\{j \in N\}$. Each link $\{(i, j) \in E\}$ has a limited bandwidth Q_{ij} (bits per unit time). Assume a set of requests $\{b \in B\}$ submitted by users. Each request has its potential revenue r^b, and consists of calls $\{c \in C_b\}$. Further, each call c in request b is characterized by an origin node $O(b, c)$, a destination node $D(b, c)$, and a bandwidth requirement d^{bc}. In this setting, the objective of CBPP is to select: (1) a subset of these requests; and (2) a single path (sequence of links) to route each call of the selected requests, such that the total revenue generated from the accepted requests is maximized. For this, we define the following sets of decision variables. $V^b = 1$ if request b is accepted, 0 otherwise. $X^{bc}_{ij} = 1$ if call c in request b is routed through a path containing the link (i, j) in the direction from i to j, 0 otherwise. We summarize below the notations used in the model.

Parameters:

N:	Set of nodes in the network, indexed by $i, j \in N$
E:	Set of directed links (i, j) in the network, where $i, j \in N$
B:	Set of requests, indexed by $b \in B$
C_b:	Set of calls of request b, indexed by $c \in C$
$O(b, c)$:	Origin node of call c of request b; $O(b, c) \in N$
$D(b, c)$:	Destination node of call c of request b; $D(b, c) \in N$
d^{bc}:	Demand (bits per unit time) of call c of request b
r^b:	Potential revenue from request b
Q_{ij}:	Bandwidth capacity (bits per unit time) of undirected link (i, j)

Decision variables:

V^b $= 1$ if request b is accepted; 0 otherwise
X_{ij}^{bc} $= 1$ if call c of request b is routed through a path that uses directed link (i, j) in the direction from i to j; 0 otherwise

Using the above notations, CBPP can be mathematically stated as:

$$\max \sum_{b \in B} r^b V^b \tag{1}$$

$$\text{s.t.} \sum_{j:(i,j)\in E} \left(X_{ij}^{bc} - X_{ji}^{bc} \right) = \begin{cases} V^b & \text{if} \quad i = O(b,c); \\ -V^b & \text{if} \quad i = D(b,c); \\ 0 & \text{otherwise} \end{cases} \quad \forall i \in N, b \in B, c \in C_b \tag{2}$$

$$\sum_{b\in B} \sum_{c \in C_b} d^{bc}(X_{ij}^{bc} + X_{ji}^{bc}) \leq Q_{ij} \qquad \forall (i, j) \in E : i < j \tag{3}$$

$$V^b \in \{0, 1\} \qquad \forall b \in B \tag{4}$$

$$X_{ij}^{bc} \in \{0, 1\} \qquad \forall (i, j) \in E, b \in B, c \in C_b \tag{5}$$

The objective function (1) is the total revenue from accepted requests. Constraint set (2) are the flow conservation equations at each node for each call. Constraint set (3) ensures that the total demand on each undirected link is less than its available bandwidth. Constraint sets (4) and (5) are binary restrictions on the variables. Note that a slightly different model for CBPP presented in Amiri and Barkhi [4] uses an additional set of binary variables to represent whether or not call c in request b is routed through a path containing link (i, j), irrespective of the direction (i to j or j to i). These variables are used in Amiri and Barkhi [4] only to aid in their Lagrangean Relaxation based solution method. As such, they are not needed in our model (1)–(5). Further, Amiri and Barkhi [4] use an additional set of binary variables (Y^{bc}) to represent whether call c in request b is accepted or not. These variables are linked to the variables V^b to ensure that all the calls of an accepted request are routed through the network. Since in CBPP, a request b can be either completely accepted or completely rejected, additional set of variables Y^{bc} to represent whether or not a call c in request b is accepted is not necessary to model the problem correctly.

As discussed in Sect. 1, the above model may result in network delays due to excessive bandwidth utilization on certain links in presence of variable call bit rates d^{bc}. To overcome this drawback, instead of maximizing the total revenue, we determine the optimal trade-off between the revenue and the response time delay due to excessive bandwidth usage. To model response time delays in the network, we assume that the arrivals of bits associated with calls in the network occur according to a Poisson process. Further, the nodes are assumed to have unlimited buffers to store calls waiting for transmission (due to finite capacities on links). We also assume that the call lengths (in bits) follow an exponential distribution with a mean $1/\mu$. The service rate (in bits per second) of the undirected link (i, j) is proportional to the capacity of the link Q_{ij}. Then, the service time per call on undirected link (i, j) also follows an exponential distribution with a mean $1/\mu Q_{ij}$. Under these assumptions, each link can be modelled as a single server M/M/1 queue. These assumptions are in line with the literature [5,10,11,15]. Thus, the arrival of bits on undirected link (i, j), due to superposition of several Poisson processes, follows a Poisson process with a rate $\sum_{b\in B}\sum_{c\in C_b} d^{bc}(X_{ij}^{bc} + X_{ji}^{bc})$, and the arrival rate of calls on link (i, j) is $\lambda_{ij} = \mu \sum_{b\in B}\sum_{c\in C_b} d^{bc}(X_{ij}^{bc} + X_{ji}^{bc})$. The average utilization of undirected link (i, j) is given by:

$$\rho_{ij} = \frac{\lambda_{ij}}{\mu Q_{ij}} = \frac{\sum_{b\in B}\sum_{c\in C_b} d^{bc}(X_{ij}^{bc} + X_{ji}^{bc})}{Q_{ij}} \tag{6}$$

Under steady state conditions ($\rho_{ij} < 1$) and first-come first-serve (FCFS) queuing discipline, the *mean sojourn time* (waiting time in queue + service time) of a call on undirected link (i, j), which is modelled as an M/M/1 queue, is given by: $E[w_{ij}] = \frac{\rho_{ij}}{1-\rho_{ij}}$. The total network delay (W) is, therefore, given by:

$$W = \sum_{(i,j)\in E:i<j} \frac{\rho_{ij}}{1-\rho_{ij}} = \sum_{(i,j)\in E:i<j} \frac{\sum_{b\in B}\sum_{c\in C_b} d^{bc}(X_{ij}^{bc} + X_{ji}^{bc})}{Q_{ij} - \sum_{b\in B}\sum_{c\in C_b} d^{bc}(X_{ij}^{bc} + X_{ji}^{bc})} \tag{7}$$

Using the above notations, the mathematical model for CBPP with queuing delays can be stated as:

$$\begin{aligned}
&[P] : \max \ Z(\mathbf{X}, \mathbf{V}) \\
&= \sum_{b\in B} r^b V^b - K \sum_{(i,j)\in E:i<j} \frac{\sum_{b\in B}\sum_{c\in C_b} d^{bc}\left(X_{ij}^{bc} + X_{ji}^{bc}\right)}{Q_{ij} - \sum_{b\in B}\sum_{c\in C_b} d^{bc}\left(X_{ij}^{bc} + X_{ji}^{bc}\right)} \\
&\text{s.t. } (2)-(5)
\end{aligned} \tag{8}$$

The objective function (8) is the net revenue (gross revenue—total queuing delay cost) from the accepted requests. The second term in (8) captures the average queuing delay cost due to all accepted calls, where K is the queuing delay cost per unit time. The above formulation $[P]$ is a non-linear integer program due to the queuing delay term in the objective function. In the following section, we present a linearization scheme and an efficient solution approach.

3 Solution approach

3.1 Linear reformulation

To linearize the queuing delay term in (8), we define non-negative auxiliary variables R_{ij}, such that:

$$R_{ij} = \frac{\sum_{b\in B}\sum_{c\in C_b} d^{bc}(X_{ij}^{bc} + X_{ji}^{bc})}{Q_{ij} - \sum_{b\in B}\sum_{c\in C_b} d^{bc}(X_{ij}^{bc} + X_{ji}^{bc})} \qquad \forall (i,j) \in E : i < j \tag{9}$$

This implies,

$$\sum_{b\in B}\sum_{c\in C_b} d^{bc}(X_{ij}^{bc} + X_{ji}^{bc}) = \frac{R_{ij}}{1+R_{ij}} Q_{ij} \qquad \forall (i,j) \in E : i < j \tag{10}$$

For a given set of points $h \in H$, the function $f(R_{ij})$ can be outer-approximated, using Taylor series approximation, by a set of piecewise linear functions that are tangent to $f(R_{ij})$ at points $(R_{ij}^h)_{h\in H}$ as follows: $f(R_{ij}) \approx f(R_{ij}^h) + f'(R_{ij}^h)(R_{ij} - R_{ij}^h) = \frac{1}{(1+R_{ij}^h)^2} R_{ij} + \frac{(R_{ij}^h)^2}{(1+R_{ij}^h)^2}$. Since $f(R_{ij})$ is concave in $R_{ij} \in [0, \infty)$, it can be expressed as follows:

$$\frac{R_{ij}}{1+R_{ij}} = \min_{h\in H}\left\{\frac{1}{(1+R_{ij}^h)^2} R_{ij} + \frac{(R_{ij}^h)^2}{(1+R_{ij}^h)^2}\right\}$$

This is equivalent to the following set of constraints:

$$\frac{R_{ij}}{1+R_{ij}} \leq \frac{1}{(1+R_{ij}^h)^2} R_{ij} + \frac{(R_{ij}^h)^2}{(1+R_{ij}^h)^2}, \qquad \forall (i,j) \in E : i<j, h\in H$$

Using (10), the above set of constraints can be rewritten as:

$$\sum_{b\in B}\sum_{c\in C_b} d^{bc}(X_{ij}^{bc} + X_{ji}^{bc}) - \frac{Q_{ij}}{(1+R_{ij}^h)^2} R_{ij} \leq \frac{Q_{ij}(R_{ij}^h)^2}{(1+R_{ij}^h)^2} \quad \forall (i,j) \in E : i<j, h\in H \tag{11}$$

provided $\exists h \in H$ such that (11) holds with equality.

The above substitutions result in the following linear MIP model:

$$[PL(H)]: \ \max \sum_{b \in B} r^b V^b - K \sum_{(i,j) \in E: i<j} R_{ij}$$

$$\text{s.t. } (2){-}(5),\ (11) \tag{12}$$

$$R_{ij} \geq 0 \qquad \forall (i,j) \in E : i < j \tag{13}$$

For equivalence between $[P]$ and $[PL(H)]$, there should exist at least one $h \in H$ such that (11) holds with equality. Proposition 1 confirms that there indeed exists one such $h \in H$ for every link $(i,j) \in E : i < j$ at optimality.

Proposition 1 *There exists at least one of the constraints* (11) *in* $[PL(H)]$ *that will be binding at optimality.*

Proof After rearranging the terms, (11) can be rewritten as:

$$R_{ij} \geq (1 + R_{ij}^h)^2 \frac{\sum_{b \in B} \sum_{c \in C_b} d^{bc}(X_{ij}^{bc} + X_{ji}^{bc})}{Q_{ij}} - (R_{ij}^h)^2 \tag{14}$$

Since R_{ij} appears in the objective function with a negative coefficient, $[PL(H)]$ attains its optimum value only when R_{ij} is minimized. This implies that $\forall (i,j) \in E : i < j$, $\exists h \in H$ such that (14) holds with equality if $(1 + R_{ij}^h)^2 \frac{\sum_{m \in M} d^{bc}(X_{ij}^{bc} + X_{ji}^{bc})}{Q_{ij}} - (R_{ij}^h)^2 \geq 0$, else $R_{ij} = 0$.

Further,

$$\begin{aligned}
0 &\leq (1 + R_{ij}^h)^2 \frac{\sum_{b \in B} \sum_{c \in C_b} d^{bc}(X_{ij}^{bc} + X_{ji}^{bc})}{Q_{ij}} - (R_{ij}^h)^2 \\
&= (1 + R_{ij}^h)^2 \rho_{ij} - (R_{ij}^h)^2 \quad \text{(using (6))} \\
&= (\rho_{ij} - 1)(R_{ij}^h)^2 + 2\rho_{ij} R_{ij}^h + \rho_{ij} \\
&\Leftrightarrow R_{ij}^h \in \left[0, \frac{\rho_{ij} + \sqrt{\rho_{ij}}}{1 - \rho_{ij}}\right] \ \forall h \in H \quad (\text{since } \rho_{ij} \leq 1 \text{ and } R_{ij} \geq 0 \text{ using (9)})
\end{aligned}$$

Thus, to prove that $\exists h \in H$ such that (11) holds with equality, we need to show that $R_{ij}^h \in \left[0, \frac{\rho_{ij} + \sqrt{\rho_{ij}}}{1 - \rho_{ij}}\right]$. If R_{ij}^h selected using (9), then we obtain:

$$\begin{aligned}
0 \leq R_{ij}^h \approx R_{ij} &= \frac{\lambda_{ij}}{\mu Q_{ij} - \lambda_{ij}} = \frac{\rho_{ij}}{1 - \rho_{ij}} \\
&\leq \frac{\rho_{ij} + \sqrt{\rho_{ij}}}{1 - \rho_{ij}}
\end{aligned}$$

This proves that $\forall (i,j) \in E : i < j$, $\exists h \in H$ such that, at optimality, (11) always holds with equality. □

3.2 Bounds and cutting plane approach

For any subset of points $\{R_{ij}^{h}\}_{h \in H^q \subseteq H}$, $[PL(H^q)]$ is the relaxation of the full problem $[PL(H)]$. Hence, $v(PL(H^q)) \geq v(PL(H))$, where $v(\bullet)$ represents the optimal objective function value of the maximization problem $(\bullet)$. Thus, $v(PL(H^q))$ provides an upper bound to $[PL(H)]$, given by:

$$UB = v(PL(H^q)) = \sum_{b \in B} r^b V^{bq} - K \sum_{(i,j) \in E : i<j} R_{ij}^{q} \tag{15}$$

For any subset of points $\{R_{ij}^{h}\}_{h \in H^q \subseteq H}$, the part of the solution $(\mathbf{V}^q, \mathbf{X}^q)$ to $[PL(H^q)]$ is also a feasible solution to $[P]$, as constraints (2)–(5) are satisfied. Hence, the objective function (8) evaluated at the solution $(\mathbf{V}^q, \mathbf{X}^q)$, gives a lower bound to $[PN]$, as follows:

$$\begin{aligned} LB = Z(\mathbf{V^q}, \mathbf{X^q}) = & \sum_{b \in B} r^b V^{bq} \\ & - K \sum_{(i,j) \in E : i<j} \frac{\sum_{b \in B} \sum_{c \in C_b} d^{bc} (X_{ij}^{bcq} + X_{ji}^{mq})}{Q_{ij} - \sum_{b \in B} \sum_{c \in C_b} d^{bc} (X_{ij}^{bcq} + X_{ji}^{bcq})} \end{aligned} \tag{16}$$

Note that the model $[PL(H)]$ consists of a large number of constraints (11). However, not all of them need to be generated a priori. The algorithm starts with an initial subset of carefully selected points, $H^q \subset H$. The resulting model $[PL(H^q)]$ is solved and the upper bound (UB^q) and the lower bound (LB^q) are computed using equations (15) and (16) respectively. If the upper bound (UB^q) equals the best known lower bound (LB^q) within accepted tolerance (ϵ) at any given iteration q, then $(\mathbf{V}^q, \mathbf{X}^q)$ is an optimal solution to $[P]$ and the algorithm is terminated. Otherwise, a new set of candidate points $R_{ij}^{h_{new}}$ is generated using the current solution $(\mathbf{X}^q)$ as follows:

$$R_{ij}^{h_{new}} = \frac{\sum_{b \in B} \sum_{c \in C_b} d^{bc} (X_{ij}^{bcq} + X_{ji}^{bcq})}{Q_{ij} - \sum_{b \in B} \sum_{c \in C_b} d^{bc} (X_{ij}^{bcq} + X_{ji}^{bcq})}$$

This new set of candidate points is used to generate "cuts/constraints" of the form:

$$\sum_{b \in B} \sum_{c \in C_b} d^{bc} (X_{ij}^{bc} + X_{ji}^{bc}) - \frac{Q_{ij}}{(1 + R_{ij}^{h_{new}})^2} R_{ij} \leq \frac{Q_{ij} (R_{ij}^{h_{new}})^2}{(1 + R_{ij}^{h_{new}})^2}$$

This set of "cuts/constraints" eliminates the best solution found so far and improves the upper bound on the remaining solutions. This new set of points is appended to $(R_{ij}^{h})_{H^q \subset H}$ and the procedure is repeated until the gap between the current upper bound and the best lower bound is within the tolerance limits. The algorithm is outlined below:

Algorithm 1 Solution Algorithm for $[PL(H)]$

```
1: $q \leftarrow 1$; $UB^{q-1} \leftarrow +\infty$; $LB^{q-1} \leftarrow -\infty$;
2: Choose an initial set of points $\{R_{ij}^h\}_{h \in H^q}$ to approximate $R_{ij}/(1+R_{ij})$ $\forall (i,j) \in E : i < j$
3: while $(UB^{q-1} - LB^{q-1})/UB^{q-1} > \epsilon$ do
4:     Solve $[PL(H^q)]$ to obtain $(\mathbf{V}^q, \mathbf{X}^q)$
5:     Update the upper bound: $UB^q \leftarrow v(PL(H^q))$
6:     Update the lower bound: $LB^q \leftarrow \max\{LB^{q-1}, Z(\mathbf{V}^q, \mathbf{X}^q)\}$.
7:     Compute new points: $R_{ij}^{h_{new}} = \frac{\sum_{b\in B}\sum_{c\in C_b} d^{bc}(X_{ij}^{bcq}+X_{ji}^{bcq})}{Q_{ij}-\sum_{b\in B}\sum_{c\in C_b} d^{bc}(X_{ij}^{bcq}+X_{ji}^{bcq})}$ $\forall (i,j) \in E : i < j$
8:     $H^{q+1} \leftarrow H^q \cup \{h_{new}\}$
9:     $q \leftarrow q+1$
10: end while
```

Proposition 2 *The proposed cutting plane algorithm is finite.*

Proof Note that X_{ij}^{bc} is binary and $R_{ij} = \frac{\sum_{b\in B}\sum_{c\in C_b} d^{bc}(X_{ij}^{bc}+X_{ji}^{bc})}{Q_{ij}-\sum_{b\in B}\sum_{c\in C_b} d^{bc}(X_{ij}^{bc}+X_{ji}^{bc})}$, hence the number of values that R_{ij} can take is finite. Therefore, in order to prove that the algorithm is finite, we prove that the generated values of R_{ij}^h are not repeated as the iterations progress.

Consider an intermediate iteration q, where the algorithm has not yet converged, that is, $UB^q > LB^q$. Further, suppose $(\mathbf{V}^q, \mathbf{X}^q)$ is the solution to $[PL(H^q)]$. Then, the new points $R_{ij}^{h_{new}}$ generated at iteration q are given by

$$R_{ij}^{h_{new}} = \frac{\sum_{b\in B}\sum_{c\in C_b} d^{bc}(X_{ij}^{bcq}+X_{ji}^{bcq})}{Q_{ij}-\sum_{b\in B}\sum_{c\in C_b} d^{bc}(X_{ij}^{bcq}+X_{ji}^{bcq})}$$

Suppose the values of $R_{ij}^{h_{new}}$ were already generated in one of the earlier iterations $\forall (i,j) \in E : i < j$. Then

$$\begin{aligned}(11) \Leftrightarrow \frac{R_{ij}^{h_{new}}}{1+R_{ij}^{h_{new}}} &\le \frac{1}{(1+R_{ij}^{h_{new}})^2} R_{ij}^q + \frac{(R_{ij}^{h_{new}})^2}{(1+R_{ij}^{h_{new}})^2} \\ \Rightarrow R_{ij}^{h_{new}} &\le R_{ij}^q\end{aligned}$$

We have:

$$\begin{aligned}UB^q &= \sum_{b\in B} r^b V^{bq} - K \sum_{(i,j)\in E:i<j} R_{ij}^q \le \sum_{b\in B} r^b V^{bq} - K \sum_{(i,j)\in E:i<j} R_{ij}^{h_{new}} \\ &= \sum_{b\in B} r^b V^{bq} - K \sum_{(i,j)\in E:i<j} \frac{\sum_{b\in B}\sum_{c\in C_b} d^{bc}\left(X_{ij}^{bcq}+X_{ji}^{bcq}\right)}{Q_{ij}-\sum_{b\in B}\sum_{c\in C_b} d^{bc}\left(X_{ij}^{bcq}+X_{ji}^{bcq}\right)}\end{aligned}$$

$$\leq \max \left\{ LB^q, \sum_{b \in B} r^b V^{bq} \right.$$
$$\left. -K \sum_{(i,j) \in E: i<j} \frac{\sum_{b \in B} \sum_{c \in C_b} d^{bcq} \left(X_{ij}^{bcq} + X_{ji}^{bcq} \right)}{Q_{ij} - \sum_{b \in B} \sum_{c \in C_b} d^{bcq} \left(X_{ij}^{bcq} + X_{ji}^{bcq} \right)} \right\} = LB^q$$

This is in contradiction with our initial assumption $UB^q > LB^q$. Therefore, at any given iteration, at least one of the values of R_{ij}^h generated is different from all the previously generated values. Since, the number of values that R_{ij}^h can take is finite, the algorithm will terminates in a finite number of iterations.

4 Computational study

We conduct computational experiments to evaluate the performance of the proposed exact solution procedure. The solution procedure is coded in Visual C++, while $[PL(H^q)]$ at every iteration q is solved using IBM ILOG CPLEX 12.4. The experiments are conducted on a machine with the following specifications: Intel Core i5-3230M, 2.60 GHz CPU; 4.00 GB RAM; Windows 64-bit Operating System.

In the implementation of the algorithm, initial set $H^1 \subset H$ can be empty. However, we start with an initial set of cuts, generated at points $h \in H^1$, based on the piecewise linear approximation $\hat{f}(R_{ij})$ to the function $f(R_{ij})$ such that the approximation error measured by $\hat{f}(R_{ij}) - f(R_{ij})$ is at most 0.001 (see [9]). This is in part motivated by our initial computational results, which suggest that starting the algorithm with a carefully chosen initial set of cuts improves the performance of the solution approach substantially (i.e., requires less cuts to be generated and less CPU times).

4.1 Computational results

The test instances used in the experiments are generated using the scheme proposed by Amiri and Barkhi [4]. Accordingly, the number of nodes in the network is varied between 20 and 100. The links are generated such that the network is connected, and each node has a degree equal to 2, 3, or 4 with a probability of 0.6, 0.3, and 0.1, respectively. This results in the number of links varying between 30 and 160. Further, each link in the network is randomly assigned a capacity (Q_{ij}) equal to 960, 1920, 5000, or 10,800 with equal probabilities. The number of calls per request b is randomly generated using a uniform distribution between 2 and 10. The bandwidth requirement of each call c in request b is randomly generated using a uniform distribution between 20 and 40.

Table 1 shows the computational results for 9 different combinations of the number of nodes $(|N|)$, number of submitted requests $(|B|)$, and the total number of calls $(|C|)$. For each of these 9 combinations, the queuing delay cost per unit time (K) is selected from the set $\{1, 10, 50, 100, 500\}$. This results in a total of 45 instances. The algorithm is terminated once the Gap (%), computed as $\frac{\text{Upper bound} - \text{Lower bound}}{\text{Lower bound}} \times 100$,

Table 1 Computational results

$\|N\|$	$\|B\|$	$\|C\|$	K	$\|B_s\|$	$\|C_s\|$	Gross revenue (GR)	Delay cost (DC) (% of GR)	Avg. util. (%)	Max. util. (%)	Gap (%)	Iter.	CPU (s)
20	500	2967	0	52	267	11,146	0	39.46	99.95	0	–	1205
			1	52	262	10,958	1.6	35.8	98.0	0.006	2	3250
			10	48	250	10,572	6.3	33.7	94.9	0.002	2	198
			50	45	231	9743	15.1	30.0	89.2	0.002	3	406
			100	42	214	9118	22.4	27.4	84.0	0.002	2	158
			500	30	137	5745	50.2	15.5	52.8	0.004	2	126
40	500	2962	0	61	292	11,616	0	37.59	100	0.129	–	3600[a]
			1	60	289	11,477	1.7	32.4	98.5	0.028	1	3590
			10	57	276	11,043	6.7	30.7	95.5	0.033	2	2708
			50	51	254	10,187	19.0	27.1	87.0	0.002	2	2924
			100	48	228	9253	27.6	23.6	77.2	0.007	2	2253
			500	26	105	4517	61.1	9.3	35.2	0.009	3	2926
60	500	3000	0	49	236	9465	0	25.43	100	0.403	–	3600[a]
			1	50	232	9281[a]	2.7	23.3	97.7	0.361	1	3600[a]
			10	Memory								
			50	43	195	7730[a]	22.5	18.8	81.3	0.148	1	3600[a]
			100	38	175	6936[a]	34.2	16.7	69.5	0.203	1	3600[a]
			500	17	61	2205[a]	74.2	5.3	22.1	1.282	1	3600[a]
80	500	3054	0	54	245	8895	0	23.92	100	0.025	–	3600[a]
			1	52	234	8727[a]	1.6	22.6	97.1	0.324	1	3600[a]
			10	50	223	8444[a]	9.2	21.3	95.0	0.100	1	3600[a]
			50	44	194	7542[a]	24.5	17.9	87.6	0.175	1	3600[a]

Table 1 continued

$\|N\|$	$\|B\|$	$\|C\|$	K	$\|B_s\|$	$\|C_s\|$	Gross revenue (GR)	Delay cost (DC) (% of GR)	Avg. util. (%)	Max. util. (%)	Gap (%)	Iter.	CPU (s)
			100	39	171	6634[a]	36.3	14.8	79.4	0.246	1	3600[a]
			500	12	40	15,94[a]	74.7	3.5	19.6	1.714	1	3600[a]
100	500	3023	0	52	264	11023	0	27.5	100	0.307	–	3600[a]
			1	52	263	10,864[a]	2.2	26.0	98.7	0.291	1	3600[a]
			10	49	254	10,695[a]	15.1	24.7	98.2	4.328	1	3600[a]
			50	46	227	9400[a]	27.2	20.8	84.1	0.147	1	3600[a]
			100	Memory								
			500	Memory								
20	1000	5993	0	88	377	15499[a]	–	53.822	100	0	–	3593
			1	84	366	15,222	1.6	47.9	98.1	0.031	1	3512
			10	82	352	14,669	6.2	45.0	94.8	0.008	2	1728
			50	76	320	13,474	14.6	39.9	88.3	0.002	3	2069
			100	72	303	12,702	22.5	37.1	83.6	0.001	3	804
			500	49	188	8006	49.7	19.8	57.2	0.009	3	729
40	1000	6056	0	90	468	19290		45.98	100	0	–	3499
			1	Memory								
			10	Memory								
			50	Memory								
			100	75	370	152,70[a]	25.8	29.6	81.2	0.128	1	3600[a]
			500	Memory								
20	1500	9099	0	77	306	11,498[a]	–	38.81	100	0.001	–	3600[a]
			1	77	304	114,19[a]	1.3	33.0	98.8	0.035	2	3600[a]

Table 1 continued

$\|N\|$	$\|B\|$	$\|C\|$	K	$\|B_s\|$	$\|C_s\|$	Gross revenue (GR)	Delay cost (DC) (% of GR)	Avg. util. (%)	Max. util. (%)	Gap (%)	Iter.	CPU (s)
			10	77	298	11,152	4.6	31.4	95.5	0.037	1	3487
			50	71	276	10,579[a]	11.9	28.0	89.4	0.057	1	3600[a]
			100	65	258	10,047	17.4	25.8	83.7	0.001	2	2407
			500	50	182	7462	41.5	15.8	60.5	0.081	2	2578
40	1500	8944	0	62	262	9898	–	29.32	99.9	0.023	–	3600[a]
			1	60	255	9728[a]	1.2	26.0	97.5	0.401	1	3600[a]
			10	58	243	9230[a]	4.1	24.3	90.3	1.166	1	3600[a]
			50	54	227	8694[a]	14.0	22.6	84.2	0.416	1	3600[a]
			100	50	213	8264[a]	23.2	20.9	79.2	0.161	1	3600[a]
			500	31	120	4997[a]	55.7	10.4	47.9	0.467	1	3600[a]

“Memory”: refers to instances that run out of memory; $|N|$: Number of nodes;
$|B|$: Number of submitted requests; $|C|$: Number of Calls; K: Unit Delay Costs; $|B_s|$: Number of accepted requests; $|C_s|$: Number of accepted calls; Iter.: Number of iterations; CPU (s): Computation time (in seconds)

[a] Instance was terminated after 3600 sec of CPU time;

falls below 0.1, or the CPU time exceeds a threshold of 3600 s. In addition to the 45 instances described above, we also report the results for $K = 0$, which corresponds to the absence of the queuing delay penalty in the objective function. Please note that such instances without the non-linear queuing delay term are solved directly using CPLEX. The upper and lower bounds used in the computation of Gap (%) for each such instance are obtained from the LP relaxation and the best integer feasible solution, respectively. On the other hand, the upper and lower bounds at each iteration of the algorithm for the instances with $K > 0$ are computed using (15) and (16), respectively. It is obvious from the results that not explicitly accounting for queuing delays in the model (corresponding to $K = 0$) always results in the maximum link utilization of almost 100 %. Such a high bandwidth utilization on any of the links in the network can potentially cause enormous delays to the accepted calls. As expected, the maximum link utilization decreases with increasing queuing penalty (K), thus reducing delays in the network.

Results in Table indicate that 18 out of the 45 instances (with $K > 0$) are solved within one hour of CPU time, with an optimality gap ranging from 0.001 to 0.081 %. Of the remaining 27 instances (with $K > 0$) that are terminated after one hour of CPU time, the optimality gap is in the range 0.035–4.328 %. Only the remaining 7 instances (with $K > 0$) failed to solve due to insufficient memory. This highlights the efficiency of our proposed solution approach, whereas the number of iterations (≤ 3) for all instances implies that only a very few of the constraints in (11) are required. As the number of nodes, the number of calls, and the number of requests increase, the problem becomes increasingly difficult to solve, requiring more CPU time. Results in Table 1 also show the effects of changes in the number of submitted requests. As the delay cost per unit time K increases, we observe that the number of accepted requests ($|B_s|$) and the number of routed calls ($|C_s|$) decrease, accompanied by a reduction in the average link utilization. However, the reduction in average link utilization may also be attributed to accepting more requests/calls with lower bandwidth requirements.

5 Conclusion

Bandwidth is a scarce resource in telecommunication networks, calling for its judicious use in presence of competing requests by bandwidth intensive video and data services. Ideally, a network administrator would like to accept as many requests, each possibly consisting of several one-to-many or many-to-many calls, to maximize the revenue earned. However, this revenue maximization objective is likely to degrade the response time due to excessive utilization of the limited bandwidth. So, in this paper, we extended this problem to finding the optimal trade-off between revenue maximization and response time delays. To this end, we formulated a non-linear IP model for the problem. We linearized the model, and presented an efficient cutting plane based approach to solve the resulting linear MIP by generating/adding constraints as needed. Our computational experiments over a wide range of problem instances, obtained by varying the size of the networks, number of requests and calls, and call bandwidth requirements, indicate that the proposed solution method can produce near optimal solutions (within an optimality gap of 0.1 %) in a reasonable computational time (≤ 1 h).

Acknowledgements This research was supported by the Research and Publication Grant, Indian Institute of Management Ahmedabad, provided to the first author, and by the Canadian Natural Science and Engineering Research Council Discovery Grant 386501-2010 provided to the second author. This support is gratefully acknowledged. The authors also acknowledge Vikranth B. T. Chetty for his assistance in computational experiments. The authors would like to thank the editor-in-chief as well as the referees for the constructive comments on the earlier version of the paper.

References

1. Amiri, A.: The multi-hour bandwidth packing problem with response time guarantees. Inform. Technol. Manag. **4**, 113–127 (2003)
2. Amiri, A.: The selection and scheduling of telecommunication calls with time windows. Eur. J. Oper. Res. **167**(1), 243–256 (2005)
3. Amiri, A., Barkhi, R.: The multi-hour bandwidth packing problem. Comput. Oper. Res. **27**, 1–14 (2000)
4. Amiri, A., Barkhi, R.: The combinatorial bandwidth packing problem. Eur. J. Oper. Res. **208**, 37–45 (2011)
5. Amiri, A., Rolland, E., Barkhi, R.: Bandwidth packing with queuing delay costs: bounding and heuristic procedures. Eur. J. Oper. Res. **112**, 635–645 (1999)
6. Anderson, C., Fraughnaugh, K., Parkner, M., Ryan, J.: Path assignment for call routing: an application of tabu search. Ann. Oper. Res. **41**, 301–312 (1993)
7. Bose, I.: Bandwidth packing with priority classes. Eur. J. Oper. Res. **192**, 313–325 (2009)
8. Cox, L., Davis, L., Qui, Y.: Dynamic anticipatory routing in circuit-switched telecommunications networks. In: Davis, L. (ed.). Handbook of Genetic Algorithms, vol. 11, pp 229–340. Van Norstrand/Reinhold, New York (1991)
9. Elhedhli, S.: Exact solution of a class of nonlinear knapsack problems. Oper. Res. Lett. **33**(6), 615–624 (2005)
10. Gavish, B., Hantler, S.: An algorithm for optimal route selection in SNA networks. IEEE Trans. Commun. **31**(10), 1154–1161 (1983)
11. Han, J., Lee, K., Lee, C., Park, S.: Exact algorithms for a bandwidth packing problem with queueing delay guarantees. INFORMS J. Comput. (2012). doi:10.1287/ijoc.1120.0523
12. Laguna, M., Glover, F.: Bandwidth packing: a tabu search approach. Manag. Sci. **39**, 492–500 (1993)
13. Park, K., Kang, S., Park, S.: An integer programming approach to the bandwidth packing problem. Manag. Sci. **42**, 1277–1291 (1996)
14. Parker, M., Ryan, J.: A column generation algorithm for bandwidth packing. Telecommun. Syst. **2**(1), 185–195 (1993)
15. Rolland, E., Amiri, A., Barkhi, R.: Queueing delay guarantees in bandwidth packing. Comput. Oper. Res. **26**, 921–935 (1999)
16. Vidyarthi, N., Jayaswal, S., Tirumala Chetty, V.B.: Exact solution to bandwidth packing problem with queuing delays. J. Glob. Optim. (2016). doi:10.1007/s10898-015-0399-8
17. Villa, C., Hoffman, K.: A column-generation and branch-and-cut approach to the bandwidth-packing problem. J. Res. Natl. Instit. Stand. Technol. **111**, 161–185 (2006)