J Glob Optim (2016) 65:745–776
DOI 10.1007/s10898-015-0399-8

Bandwidth packing problem with queueing delays: modelling and exact solution approach

Navneet Vidyarthi[1] · **Sachin Jayaswal**[2] ·
Vikranth Babu Tirumala Chetty[3]

Received: 27 July 2014 / Accepted: 28 December 2015 / Published online: 16 January 2016

Abstract We present a more generalized model for the bandwidth packing problem with queuing delays under congestion than available in the extant literature. The problem, under Poison call arrivals and general service times, is set up as a network of spatially distributed independent M/G/1 queues. We further present two exact solution approaches to solve the resulting nonlinear integer programming model. The first method, called finite linearization method, is a conventional Big-M based linearization, resulting in a finite number of constraints, and hence can be solved using an off-the-shelve MIP solver. The second method, called constraint generation method, is based on approximating the non-linear delay terms using supporting hyperplanes, which are generated as needed. Based on our computational study, the constraint generation method outperforms the finite linearization method. Further comparisons of results of our proposed constraint generation method with the Lagrangean relaxation based solution method reported in the literature for the special case of exponential service times clearly demonstrate that our approach outperforms the latter, both in terms of the quality of solution and computation times.

This research was supported by the Research and Publication Grant, Indian Institute of Management Ahmedabad, provided to the second author and Natural Sciences and Engineering Research Council of Canada (NSERC) Discovery Grant of the first author.

✉ Navneet Vidyarthi
navneet.vidyarthi@gmail.com; N.Vidyarthi@concordia.ca

Sachin Jayaswal
sachin@iimah.ernet.in

Vikranth Babu Tirumala Chetty
vikranth@uw.edu

1 Department of Supply Chain and Business Technology Management, John Molson School of Business and Interuniversity Research Centre on Enterprise Networks, Logistics, and Transportation (CIRRELT), Concordia University, Montreal H3G 1M8, Canada

2 Production and Quantitative Methods, Indian Institute of Management, Vastrapur, Ahmedabad, India

3 Industrial and Systems Engineering College of Engineering, University of Washington, Seattle, WA, USA

Keywords Bandwidth packing · Telecommunications · Call routing · Integer programming · Queuing delay · Finite linearization · Constraint generation

1 Introduction

Technological improvements in the telecommunications industry have led to a massive growth of services like video-conferencing, social networking, collaborative computing, etc. At the same time, the arrival of cheaper and smarter devices have resulted in demand for faster and better telecommunication services. This has increased the pressure on telecommunication firms to efficiently manage their limited bandwidth to provide satisfactory end-user services. In this context, one of the fundamental problems that arises is the bandwidth packing problem (BPP). The BPP can be stated as: given a set of calls, and their associated potential revenues and bandwidth requirements (demand), arising at an instant on a telecommunication network with limited bandwidth on its links, (i) decide which of these calls to accept/reject, and (ii) select a single path (sequence of links) to route each selected call, such that the total revenue generated from the accepted calls is maximized without violating the bandwidth capacities on the links [8].

Several variants of BPP have been studied in the literature. For example, Amiri and Barkhi [3] present a multi-hour BPP to account for the variation in traffic between peak and off-peak hours of the day. Another version of BPP that involves scheduling of the selected calls within given time windows is reported by Amiri [2]. Amiri and Barkhi [4] present an extension of BPP that has applications in telecommunication services like video conferencing and collaborative computing. They consider a case wherein each request from users consists of a set of calls between various pairs of nodes, and a request cannot be partially accepted/rejected. Recently, Bose [7] has studied another version of the problem wherein the calls belong to two priority classes: the calls belonging to the higher priority class are shorter in length and generate more revenue but consume more bandwidth compared to the calls belonging to the lower priority class.

Other extensions of BPP account for the delays arising as a result of calls waiting at nodes due to congestion on the links. Excessive delays may arise if the solution to BPP, or its variants, result in certain links getting utilized close to their bandwidth capacities. Explicit consideration of such delays in the modelling and solution of BPP is important to guarantee quality service to customers. Amiri et al. [5], Han et al. [11] and Rolland et al. [15] explicitly account for such network delays due to congestion by incorporating queuing delay terms in their model. All of these papers model the links in the network as independent M/M/1 queues with the implicit underlying assumption that call arrivals are Poisson and their service times on links have exponential distribution. Amiri et al. [5] penalize such delays in the objective function, while Han et al. [11] and Rolland et al. [15] impose a constraint to limit such delays. Bose [7] extends the problem to a setting where calls may be classified into different priority classes. 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. The problem presented by [10] is also related to BPP with delays due to congestion, although the acceptance/rejection of calls is not a decision in their problem.

The single path requirement in BPP, which arises in various telecommunications services like video teleconferencing, etc., makes the problem NP-hard [14]. As such, various solution methods are presented in the literature. Anderson et al. [6] and Laguna and Glover [12], for instance, use Tabu Search metaheuristic, while Cox et al. [8] apply Genetic Algorithms. Lagrangean relaxation has been a popular choice of solution method in the literature [1–

5,10,15]. Branch-and-Price and Column Generation is used by Parker and Ryan [14], while Park et al. [13] and Villa and Hoffman [16] report the use of Branch-and-Price-and-Cut and Column Generation. Han et al. [11] use Branch-and-Price technique with their Dantzig–Wolfe decomposition based reformulation of their model.

1.1 Contribution

From the review of the literature, we observe that most studies on BPP in telecommunications networks that account for delays on links due to congestion are based on the simplifying assumption that call arrivals are Poisson and service times on links have exponential distribution [1,5,7,10,11,15]. This is primarily to make the problem, which is already otherwise NP-hard, tractable. The current study is an attempt to overcome this limitation in the extant literature by presenting a more generalized model for BPP with queuing delays, wherein the links in the network are modeled as independent M/G/1 queues and call delays are captured using their steady state waiting times at various nodes in their path. The proposed model penalizes the excessive use of link capacities.

Capturing call delays in the problem using M/G/1 queueing model for the links in the network gives rise to non-linearity in the integer programming model. Hence, the second contribution of the paper lies in the two exact solution approaches developed to solve the resulting non-linear integer programming model for BPP with queuing delays. Both the solution approaches are natural generalizations of the corresponding methods used to solve a class of non-linear knapsack problems proposed by Elhedhli [9]. By rearranging the queuing delay terms in the objective function and using additional auxiliary variables, we linearize the model as is done by Elhedhli [9]. This, however, comes at the cost of an additional set of constraints linking the auxiliary variables with the delay terms. The two approaches differ in the way the non-linearity in the resulting additional constraints is handled. The first method is a conventional Big-M based linearization, resulting in a finite number of constraints, and hence can be solved using an off-the-shelve MIP solver like CPLEX. The second method is based on approximating the non-linear terms using supporting hyperplanes, which are generated as needed. Hence, we call this method a constraint generation method. We also provide a proof of finiteness of this method along similar lines as Elhedhli [9].

Our computational results show that, in terms of computation time, the constraint generation method outperforms the finite linearization method. We also compare the results of the constraint generation method with the Lagrangean relaxation based solution method reported in the literature for the special case of exponential service times, and demonstrate that our approach outperforms the latter, both in terms of the quality of solution and computational times.

The remainder of the paper is organized as follows. In Sect. 2, we formally describe the problem and present its non-linear integer programming formulation. Section 3 describes the two lineaziation approaches, followed by constraint generation based solution method for the second (supporting hyperplane based) liniearization approach. Illustrative example, computational results, and insights are reported in Sect. 4. Section 5 concludes with some directions for future research.

2 Problem formulation

We introduce the following notations used to describe the problem.

N	Set of nodes in the network

i, j	Indices for nodes in the network; $i, j \in N$
E	Set of undirected links in the network
(i, j)	Undirected links in the network; $i < j$
M	Set of calls
m	Index for a call; $m \in M$
$O(m)$	Origin node of call m; $O(m) \in N$
$D(m)$	Destination node of call m; $D(m) \in N$
d^m	Demand (bits per unit time) of call m
r^m	Potential revenue from call m
Q_{ij}	Bandwidth capacity of link (i, j)
$1/\mu$	Mean of message length
σ	Standard deviation of message length
cv	Coefficient of variation of message length; $cv = \mu\sigma$
c	Unit queuing delay cost per unit time

In line with the literature [5,10,11,15], we assume that the arrivals of calls/messages on the network occur according to a Poisson process. Further, links are assumed to have finite capacities Q_{ij} for transmission of messages, and that nodes have unlimited buffers to store messages waiting for transmission. However, unlike the existing literature, we allow the message lengths (in bits) to follow a general distribution with a mean $1/\mu$, standard deviation σ, and coefficient of variation $cv = \mu\sigma$. The service rate (in bits per second) of the link (i, j) is proportional to the capacity of the link Q_{ij}. Then, the service time per message on link (i, j) also follows a general distribution with a mean $1/(\mu Q_{ij})$, standard deviation σ/Q_{ij}, and coefficient of variation $cv = \mu\sigma$. Each link is thus modeled as a single server M/G/1 queue, and the telecommunication network is modeled as a network of independent M/G/1 queues.

Assume the bits composing message $m \in M$ arrive at a rate d^m per unit time. Further, let X^m_{ij} $(X^m_{ji}) = 1$ if call m is routed through link (i, j) in the direction from i to j (j to i), 0 otherwise. Then, the arrival of bits on link (i, j), due to superposition of Poisson processes, follows a Poisson process with a rate $\sum_{m\in M} d^m(X^m_{ij} + X^m_{ji})$ per unit time, and the arrival rate of messages per unit time on link (i, j) is $\lambda_{ij} = \mu \sum_{m\in M} d^m(X^m_{ij} + X^m_{ji})$. The average utilization of link (i, j) is given by:

$$\rho_{ij} = \frac{\lambda_{ij}}{\mu Q_{ij}} = \frac{\sum_{m\in M} d^m \left(X^m_{ij} + X^m_{ji}\right)}{Q_{ij}} \tag{1}$$

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 message on link (i, j), which is modeled as an M/G/1 queue, is given by the Pollaczek-Khintchine (PK) formula as: $E[w_{ij}] = \left(\frac{1+cv^2}{2}\right)\frac{\lambda_{ij}}{\mu Q_{ij}(\mu Q_{ij}-\lambda_{ij})} + \frac{1}{\mu Q_{ij}}$. The expected network delay can be estimated as the weighted average of the expected delays on links: $\frac{1}{\Lambda}\sum_{(i,j)\in E} \lambda_{ij} E[w_{ij}]$, resulting in the following:

$$E[W] = \frac{1}{\Lambda} \sum_{(i,j)\in E} \left\{ \left(\frac{1+cv^2}{2}\right) \frac{(\lambda_{ij})^2}{\mu Q_{ij}\left(\mu Q_{ij} - \lambda_{ij}\right)} + \frac{\lambda_{ij}}{\mu Q_{ij}} \right\}, \tag{2}$$

where $\Lambda = \mu \sum_{m\in M} d^m$ is the total arrival rate of messages in the network. Substituting $\lambda_{ij} = \mu \sum_{m\in M} d^m(X^m_{ij} + X^m_{ji})$, as defined above, $E[W]$ can be further expressed as:

$$E[W] = \frac{1}{\Lambda} \sum_{(i,j)\in E} \left\{ \left(\frac{1+cv^2}{2}\right) \frac{\left(\sum_{m\in M} d^m \left(X_{ij}^m + X_{ji}^m\right)\right)^2}{Q_{ij}(Q_{ij} - \sum_{m\in M} d^m \left(X_{ij}^m + X_{ji}^m\right)} + \frac{\sum_{m\in M} d^m \left(X_{ij}^m + X_{ji}^m\right)}{Q_{ij}} \right\} \tag{3}$$

Using the above notations, the problem BPP under queuing delay that we study can be stated as follows: given a set of calls M, their associated potential revenues ($r^m, m \in M$) and bandwidth requirements ($d^m, m \in M$), arising at an instant on an undirected telecommunication network consisting of nodes N and links E with fixed arc/link capacities ($Q_{ij}, (i, j) \in E$), determine a subset of calls $M^{'} \subseteq M$ and a subset of $E^{'} \subseteq E$ for each $m \in M^{'}$, such that the total net revenue minus queuing delay costs is maximized. Let $Y^m = 1$ if call m is accepted, 0 otherwise, then the mathematical model for BPP with queuing delays can be stated as:

$[PN]$:

$$\max\ Z(\mathbf{X}, \mathbf{Y}) = \sum_{m\in M} r^m Y^m - C \sum_{(i,j)\in E} \left\{ \left(\frac{1+cv^2}{2}\right) \frac{\left(\sum_{m\in M} d^m \left(X_{ij}^m + X_{ji}^m\right)\right)^2}{Q_{ij}(Q_{ij} - \sum_{m\in M} d^m \left(X_{ij}^m + X_{ji}^m\right)} + \frac{\sum_{m\in M} d^m \left(X_{ij}^m + X_{ji}^m\right)}{Q_{ij}} \right\} \tag{4}$$

$$\text{s.t.} \sum_{j\in N} X_{ij}^m - \sum_{j\in N} X_{ji}^m = \begin{cases} Y^m & \text{if } i = O(m); \\ -Y^m & \text{if } i = D(m); \\ 0 & \text{otherwise} \end{cases} \qquad \forall (i, j) \in E, m \in M \tag{5}$$

$$\sum_{m\in M} d^m \left(X_{ij}^m + X_{ji}^m\right) \leq Q_{ij} \qquad \forall (i, j) \in E \tag{6}$$

$$X_{ij}^m \in \{0, 1\}, \quad X_{ji}^m \in \{0, 1\} \qquad \forall (i, j) \in E, m \in M \tag{7}$$

$$Y^m \in \{0, 1\} \qquad \forall m \in M \tag{8}$$

The first term in the objective function (4) is the total revenue from accepted calls. The second term captures the average queuing delay cost due to all accepted calls, where $C = c/\Lambda$ (a constant). Constraint set (5) represents the flow conservation equation on each link for each call. Constraint set (6) ensures that the total demand on each link is less than its bandwidth capacity, required for the stability of the queue ($\lambda_{ij} \leq \mu Q_{ij}$). Constraint sets (7) and (8) are binary restrictions on the variables. For $cv = 1$, the above formulation reduces to the $M/M/1$ model studied by [5] and others.

The formulation $[PN]$ is a non-linear integer program. In the following section, we present two approaches to transform the above model, using auxiliary variables, into a Mixed Integer Linear Program (MILP). While the Big-M based finite linearization model can be solved directly using a commercial solver, the supporting hyperplane based linearization results in a large number of constraints, which is amenable to a constraint generation based solution method.

3 Solution approaches

3.1 Finite linearization

After rearranging the terms in (2), $E[W]$ can be rewritten as:

$$E[W] = \frac{1}{\Lambda} \sum_{(i,j)\in E} \frac{1}{2} \left\{ \left(1 + cv^2\right) \frac{\lambda_{ij}}{\mu Q_{ij} - \lambda_{ij}} + \left(1 - cv^2\right) \frac{\lambda_{ij}}{\mu Q_{ij}} \right\}$$
$$= \frac{1}{\Lambda} \sum_{(i,j)\in E} \frac{1}{2} \left\{ \left(1 + cv^2\right) \frac{\sum_{m\in M} d^m \left(X_{ij}^m + X_{ji}^m\right)}{Q_{ij} - \sum_{m\in M} d^m \left(X_{ij}^m + X_{ji}^m\right)} \right.$$
$$\left. + \left(1 - cv^2\right) \frac{\sum_{m\in M} d^m \left(X_{ij}^m + X_{ji}^m\right)}{Q_{ij}} \right\}$$

We define non-negative auxiliary variables R_{ij}, such that:

$$R_{ij} = \frac{\lambda_{ij}}{\mu Q_{ij} - \lambda_{ij}} = \frac{\sum_{m\in M} d^m \left(X_{ij}^m + X_{ji}^m\right)}{Q_{ij} - \sum_{m\in M} d^m \left(X_{ij}^m + X_{ji}^m\right)} \tag{9}$$

This is equivalent to:

$$Q_{ij} R_{ij} - R_{ij} \sum_{m\in M} d^m \left(X_{ij}^m + X_{ji}^m\right) = \sum_{m\in M} d^m \left(X_{ij}^m + X_{ji}^m\right) \tag{10}$$

Let $Z_{ij}^m = R_{ij} X_{ij}^m$ and $Z_{ji}^m = R_{ij} X_{ji}^m$. Then, (10) can be linearized using (11)–(17) as given below:

$$Q_{ij} R_{ij} - \sum_{m\in M} d^m \left(Z_{ij}^m + Z_{ji}^m\right) = \sum_{m\in M} d^m \left(X_{ij}^m + X_{ji}^m\right) \tag{11}$$
$$Z_{ij}^m \le R_{ij} \tag{12}$$
$$Z_{ij}^m \le \mathbf{M} X_{ij}^m \tag{13}$$
$$Z_{ij}^m \ge R_{ij} + \mathbf{M}(X_{ij}^m - 1) \tag{14}$$
$$Z_{ji}^m \le R_{ij} \tag{15}$$
$$Z_{ji}^m \le \mathbf{M} X_{ji}^m \tag{16}$$
$$Z_{ji}^m \ge R_{ij} + \mathbf{M}\left(X_{ji}^m - 1\right) \tag{17}$$

where, **M** refers to a large number (commonly referred to as Big-M). The above Big-M based linearization is a fairly standard approach. The resulting linear model is given by:

$$[FL] : \max \sum_{m\in M} r^m Y^m - \frac{C}{2} \sum_{(i,j)\in E} \left\{ \left(1 + cv^2\right) R_{ij} + \frac{\left(1 - cv^2\right)}{Q_{ij}} \sum_{m\in M} d^m \left(X_{ij}^m + X_{ji}^m\right) \right\} \tag{18}$$
$$\text{s.t. (5)–(8), (11)–(17)} \qquad \forall (i,j) \in E, m \in M$$
$$Z_{ij}^m \ge 0, \quad Z_{ji}^m \ge 0 \qquad \forall (i,j) \in E, m \in M \tag{19}$$
$$R_{ij} \ge 0 \qquad \forall (i,j) \in E \tag{20}$$

Model $[FL]$ can be solved directly using an off-the-shelf MIP solver like CPLEX.

3.2 Constraint generation method

Using the non-negative auxiliary variables R_{ij}, the relationship (9) between R_{ij} and routing variables X_{ij} can be alternatively expressed as follows:

$$\sum_{m\in M} d^m \left(X_{ij}^m + X_{ji}^m\right) = \frac{R_{ij}}{1+R_{ij}} Q_{ij} \tag{21}$$

We use the following lemma to linearize the non-linear term $\frac{R_{ij}}{1+R_{ij}}$ in (21) .

Lemma 1 *The function* $f(R_{ij}) = \frac{R_{ij}}{1+R_{ij}}$ *is concave in* $R_{ij} \in [0, \infty)$.

Proof Differentiating the function w.r.t. R_{ij}, we get the first derivative $\frac{\delta f}{\delta R_{ij}} = \frac{1}{(1+R_{ij})^2} > 0$, and the second derivative $\frac{\delta^2 f}{\delta R_{ij}^2} = \frac{-2}{(1+R_{ij})^3} < 0$, which proves that the function is concave in R_{ij} for $R_{ij} > 0$.

Lemma 1 implies that the function $f(R_{ij}) = \frac{R_{ij}}{1+R_{ij}}$ can be approximated by a large set of supporting hyperplanes to $f(R_{ij})$ at points $\{R_{ij}^h\}_{h\in H}$, such that:

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

This is equivalent to the following set of constraints:

$$\frac{R_{ij}}{1+R_{ij}} \le \frac{1}{\left(1+R_{ij}^h\right)^2} R_{ij} + \frac{\left(R_{ij}^h\right)^2}{\left(1+R_{ij}^h\right)^2} \qquad \forall (i,j) \in E, h \in H$$

Using (21), the above set of constraints can be rewritten as:

$$\sum_{m\in M} d^m \left(X_{ij}^m + X_{ji}^m\right) - \frac{Q_{ij}}{\left(1+R_{ij}^h\right)^2} R_{ij} \le \frac{Q_{ij}\left(R_{ij}^h\right)^2}{\left(1+R_{ij}^h\right)^2} \qquad \forall (i,j) \in E, h \in H \tag{22}$$

provided $\exists h \in H$ such that (22) holds with equality.

The above substitutions result in the following linear MIP model:

$[PL(H)]$:

$$\max \sum_{m\in M} r^m Y^m - \frac{C}{2} \sum_{(i,j)\in E} \left\{ \left(1+cv^2\right) R_{ij} + \left(1-cv^2\right) \frac{\sum_{m\in M} d^m \left(X_{ij}^m + X_{ji}^m\right)}{Q_{ij}} \right\} \tag{23}$$

$$\text{s.t. } (5)-(8), (22)$$

$$R_{ij} \ge 0 \qquad \forall (i,j) \in E \tag{24}$$

For equivalence between $[PN]$ and $[PL(H)]$, there should exist at least one $h \in H$ such that (22) holds with equality. Proposition 1 confirms that there indeed exists one such $h \in H$ at optimality. □

Proposition 1 *At least one of the constraints (22) in $[PL(H)]$ will be binding at optimality.*

Proof After rearranging the terms, (22) can be rewritten as:

$$R_{ij} \geq \left(1 + R_{ij}^h\right)^2 \frac{\sum_{m \in M} d^m \left(X_{ij}^m + X_{ji}^m\right)}{Q_{ij}} - \left(R_{ij}^h\right)^2 \qquad \forall (i,j) \in E, h \in H \tag{25}$$

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 $\forall (i,j) \in E, \exists h \in H$ such that (25) holds with equality if $(1 + R_{ij}^h)^2 \frac{\sum_{m \in M} d^m (X_{ij}^m + X_{ji}^m)}{Q_{ij}} - (R_{ij}^h)^2 \geq 0$, else $R_{ij} = 0$.

Further,

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

Thus, to prove that $\exists h \in H$ such that (22) 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 is selected using (9), then we obtain:

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

This proves that $\forall (i,j) \in E$, $\exists h \in H$ such that, at optimality, (22) always holds with equality. □

Proposition 2 *For every subset of points $\{R_{ij}^h\}_{h \in H^q \subseteq H}$, $v(PL(H^q))$ is an upper bound to $[PL(H)]$, and hence to $[PN]$, where $v(\bullet)$ is the optimal objective function value of the problem $(\bullet)$.*

Proof The proof follows along similar lines as in Elhedhli [9]. Suppose, at any iteration, we use supporting hyperplanes at points $\{R_{ij}^h\}_{h \in H^q \subseteq H}$, and solve the corresponding problem $[PL(H^q)]$, which yields the solution $(\mathbf{X}^q, \mathbf{Y}^q, \mathbf{R}^q)$ with the objective function value $v(PL(H^q))$. Since $[PL(H^q)]$ is a relaxation of the full problem $[PL(H)]$, $v(PL(H^q)) \geq v(PL(H))$, and hence $v(PL(H^q))$ provides an upper bound, given by:

$$\begin{aligned}
UB = v(PL(H^q)) = \sum_{m \in M} r^m Y^{mq} - \frac{C}{2} \\
\sum_{(i,j) \in E} \left\{ \left(1 + cv^2\right) R_{ij}^q + \left(1 - cv^2\right) \frac{\sum_{m \in M} d^m (X_{ij}^{mq} + X_{ji}^{mq})}{Q_{ij}} \right\}
\end{aligned} \tag{26}$$

□

Proposition 3 *For every subset of points* $\{R_{ij}^{h}\}_{h \in H^q \subseteq H}$, *the lower bound to* $[PN]$ *is given by:*

$$LB = Z\left(\mathbf{X^q}, \mathbf{Y^q}\right) = \sum_{m \in M} r^m Y^{mq} - C \sum_{(i,j) \in E} \times \left\{ \left(\frac{1+cv^2}{2}\right) \frac{\left(\sum_{m \in M} d^m \left(X_{ij}^{mq} + X_{ji}^{mq}\right)\right)^2}{Q_{ij}(Q_{ij} - \sum_{m \in M} d^m \left(X_{ij}^{mq} + X_{ji}^{mq}\right)} + \frac{\sum_{m \in M} d^m \left(X_{ij}^{mq} + X_{ji}^{mq}\right)}{Q_{ij}} \right\} \tag{27}$$

where $(\mathbf{X}^q, \mathbf{Y}^q, \mathbf{R}^q)$ *is the optimal solution to* $[PL(H^q)]$.

Proof The proof follows from the argument used by [9] for the non-linear knapsack problem. For every subset of points $\{R_{ij}^{h}\}_{h \in H^q \subseteq H}$, the solution $(\mathbf{X}^q, \mathbf{Y}^q, \mathbf{R}^q)$ to $[PL(H^q)]$ is also a feasible solution to $[PN]$, and hence the objective function (4) evaluated at the solution $(\mathbf{X}^q, \mathbf{Y}^q)$, which is given by (27), gives a lower bound to $[PN]$. □

3.2.1 Solution algorithm

The model $[PL(H)]$ consists of a large number of constraints of the form (22). However, not all of them need to be generated a priori. The solution algorithm starts with an initial subset $H^1 \subset H$. H^1 may be empty. However, our preliminary computational experiments show that starting with a non-empty H^1 helps in faster convergence of the algorithm. The resulting $[PL(H^1)]$ is solved, giving a solution $(\mathbf{X}^1, \mathbf{Y}^1, \mathbf{R}^1)$. The upper bound (UB^1) and the lower bound (LB^1) are computed using (26) and (27) respectively. The better of the last and the new lower bounds is retained as the new LB^1. If UB^1 equals LB^1 within some accepted tolerance (ϵ), then $(\mathbf{X}^1, \mathbf{Y}^1)$ is an optimal solution to $[PN]$, and the algorithm terminates. Else, a new set of points $R_{ij}^{h_{new}}$ is generated using the current solution (X^1, Y^1, R^1) as follows: $R_{ij}^{h_{new}} = \frac{\sum_{m \in M} d^m (X_{ij}^{m1} + X_{ji}^{m1})}{Q_{ij} - \sum_{m \in M} d^m (X_{ij}^{m1} + X_{ji}^{m1})}$. New cuts of the form (22) are generated using these points, and added to $[PL(H^1)]$ to arrive at $[PL(H^2)]$. Next, $[PL(H^2)]$ is solved, giving a new solution $(\mathbf{X}^2, \mathbf{Y}^2, \mathbf{R}^2)$ and UB^2. The new lower bound is obtained as the greater of LB^1 and $Z(\mathbf{X}^2, \mathbf{Y}^2)\}$. If UB^2 equals LB^2 within the set tolerance (ϵ), then the algorithm terminates with $(\mathbf{X}^2, \mathbf{Y}^2)$ as an optimal solution. Else, the process is repeated until UB^q equals LB^q within the set tolerance for some iteration q. The complete algorithm is outlined below:

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

$q \leftarrow 1$; $UB^{q-1} \leftarrow +\infty$; $LB^{q-1} \leftarrow -\infty$;
Choose an initial set of points $\{R_{ij}^{h}\}_{h \in H^q}$ to approximate the function $R_{ij}/(1 + R_{ij})$ $\forall (i, j) \in E$.
while $(UB^{q-1} - LB^{q-1})/UB^{q-1} > \epsilon$ **do**
  Solve $[PL(H^q)]$ to obtain $(\mathbf{X}^q, \mathbf{Y}^q, \mathbf{R}^q)$.
  Update the upper bound: $UB^q \leftarrow v(PL(H^q))$.
  Update the lower bound: $LB^q \leftarrow \max\{LB^{q-1}, Z(\mathbf{X}^q, \mathbf{Y}^q)\}$.
  Compute new points: $R_{ij}^{h_{new}} = \frac{\sum_{m \in M} d^m (X_{ij}^{mq} + X_{ji}^{mq})}{Q_{ij} - \sum_{m \in M} d^m (X_{ij}^{mq} + X_{ji}^{mq})}$ $\forall (i, j) \in E$
  $H^{q+1} \leftarrow H^q \cup \{h_{new}\}$
  $q \leftarrow q + 1$
end while

Proposition 4 *The proposed algorithm to solve $[PL(H)]$ terminates in a finite number of iterations.*

Proof The proof follows along similar lines as in Elhedhli [9]. Given that $X_{ij}^m \in \{0,1\}$ and $R_{ij} = \frac{\lambda_{ij}}{\mu Q_{ij}-\lambda_{ij}} = \frac{\sum_{m\in M} d^m (X_{ij}^m + X_{ji}^m)}{Q_{ij}-\sum_{m\in M} d^m (X_{ij}^m + X_{ji}^m)}$, the set of values that R_{ij} can take is finite. Therefore, in order to prove that Algorithm 1 is finite, it is sufficient to prove that the generated values of R_{ij}^h are not repeated. For that, consider an iteration q, where Algorithm 1 has not yet converged, that is, $UB^q > LB^q$. Further, suppose $(\mathbf{X}^q, \mathbf{Y}^q)$ is the solution to $[PL(H^q)]$. Then, the new points $R_{ij}^{h_{new}}$ generated at iteration q are given by:

$$R_{ij}^{h_{new}} = \frac{\sum_{m\in M} d^m \left(X_{ij}^{mq} + X_{ji}^{mq}\right)}{Q_{ij} - \sum_{m\in M} d^m \left(X_{ij}^{mq} + X_{ji}^{mq}\right)} \qquad \forall (i,j) \in E$$

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

$$(22) \Rightarrow \frac{R_{ij}^{h_{new}}}{1+R_{ij}^{h_{new}}} \le \frac{1}{\left(1+R_{ij}^{h_{new}}\right)^2} R_{ij}^q + \left(\frac{R_{ij}^{h_{new}}}{1+R_{ij}^{h_{new}}}\right)^2 \qquad \forall (i,j) \in E$$

$$\Rightarrow R_{ij}^{h_{new}} \le R_{ij}^q \qquad \forall (i,j) \in E$$

We now have:

$$\begin{aligned}
UB^q &= \sum_{m\in M} r^m Y^{mq} - \frac{C}{2} \sum_{(i,j)\in E} \left\{ \left(1+cv^2\right) R_{ij}^q + \left(1-cv^2\right) \frac{\sum_{m\in M} d^m \left(X_{ij}^{mq} + X_{ji}^{mq}\right)}{Q_{ij}} \right\} \\
&\le \sum_{m\in M} r^m Y^{mq} - \frac{C}{2} \sum_{(i,j)\in E} \left\{ \left(1+cv^2\right) R_{ij}^{h_{new}} + \left(1-cv^2\right) \frac{\sum_{m\in M} d^m \left(X_{ij}^{mq} + X_{ji}^{mq}\right)}{Q_{ij}} \right\} \\
&= \sum_{m\in M} r^m Y^{mq} - \frac{C}{2} \sum_{(i,j)\in E} \left\{ \left(1+cv^2\right) \frac{\sum_{m\in M} d^m \left(X_{ij}^{mq} + X_{ji}^{mq}\right)}{Q_{ij} - \sum_{m\in M} d^m \left(X_{ij}^{mq} + X_{ji}^{mq}\right)} \right. \\
&\qquad \left. + \left(1-cv^2\right) \frac{\sum_{m\in M} d^m \left(X_{ij}^{mq} + X_{ji}^{mq}\right)}{Q_{ij}} \right\} \\
&= \sum_{m\in M} r^m Y^{mq} - C \sum_{(i,j)\in E} \left\{ \left(\frac{1+cv^2}{2}\right) \frac{\left(\sum_{m\in M} d^m \left(X_{ij}^{mq} + X_{ji}^{mq}\right)\right)^2}{Q_{ij}(Q_{ij} - \sum_{m\in M} d^m \left(X_{ij}^{mq} + X_{ji}^{mq}\right)} \right. \\
&\qquad \left. + \frac{\sum_{m\in M} d^m \left(X_{ij}^{mq} + X_{ji}^{mq}\right)}{Q_{ij}} \right\} \\
&\le max \left(LB^{q-1}, \sum_{m\in M} r^m Y^{mq} - C \right. \\
&\qquad \left. \sum_{(i,j)\in E} \left\{ \left(\frac{1+cv^2}{2}\right) \frac{\left(\sum_{m\in M} d^m \left(X_{ij}^{mq} + X_{ji}^{mq}\right)\right)^2}{Q_{ij}(Q_{ij} - \sum_{m\in M} d^m \left(X_{ij}^{mq} + X_{ji}^{mq}\right)} + \frac{\sum_{m\in M} d^m \left(X_{ij}^{mq} + X_{ji}^{mq}\right)}{Q_{ij}} \right\} \right) \\
&= LB^q
\end{aligned}$$

This contradicts our initial assumption $UB^q > LB^q$. Therefore, at a given iteration, at least one of the values of R^h_{ij} generated is different from all the previously generated values. Furthermore, the number of values that R^h_{ij} can take is finite, and hence the algorithm terminates in a finite number of iterations. □

4 Computational study

We report our computational experience with the solution methodology described in Sect. 3. The Big-M based finite linearization model is coded in Visual C++, and solved directly using IBM ILOG CPLEX 12.4. The supporting hyperplane based constraint generation algorithm is also coded in Visual C++, while $[PL(H^q)]$ at every iteration q is solved using IBM ILOG CPLEX 12.4. All 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 Sect. 4.1, using an illustrative example 10 nodes and 20 calls, we demonstrate the impact of variability in service times of the links on the optimal selection of calls and their routes in the network. Computational performance of the proposed solution approaches on networks with varying sizes are presented in Sect. 4.2.2.

4.1 Illustrative example

Figure 1 shows the network topology for a problem instance with 10 nodes. The bandwidth capacities (Q_{ij}) of different links on the network are given in Table 1. The call table listing the bandwidth requirements (d^m) and the potential revenues (r^m) for 20 calls is shown in Table 2. The optimal solution obtained using the method described in Sect. 3 is presented in Table 3, which displays the optimal routing (collection of links) for each call that is accepted, as well as the total gross revenue (GR) and the total delay cost (DC) , for different values of coefficient of variation cv and unit delay cost (C).

Table 3 demonstrates that the value of cv plays an important role in the call selection. For example, for $C = 5$, Call-9 is *rejected* at $cv = 0.5$, but gets *accepted* at higher values

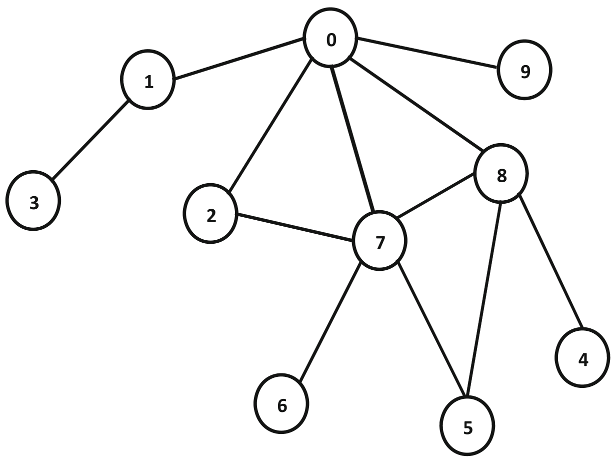

Fig. 1 Network topology for a 10-node illustrative example

Table 1 Bandwidth capacities (Q_{ij}) of links (i, j) for the illustrative example

i	0	0	0	0	0	1	2	4	5	5	6	7
j	1	2	7	8	9	3	7	8	7	8	7	8
Q_{ij}	25	35	40	20	15	10	20	15	10	15	10	10

Table 2 Call table for the illustrative example

Call m	Origin node $O(m)$	Destination node $D(m)$	Call demand d^m	Revenue r^m
1	0	2	10	420
2	0	7	7	380
3	0	5	6	400
4	0	4	6	390
5	1	6	5	500
6	1	5	5	490
7	1	4	7	400
8	2	9	2	150
9	2	3	4	450
10	2	4	8	500
11	3	5	6	850
12	5	2	3	200
13	6	9	5	370
14	7	1	6	500
15	7	9	5	340
16	7	4	2	120
17	8	1	6	460
18	8	2	8	450
19	9	5	5	360
20	9	1	5	170

of $cv = 1, 1.5, 2$. On the other hand, for $C = 5$, Call-11 is *accepted* at $cv = 0.5$, but gets *rejected* at higher values of $cv = 1, 1.5, 2$. Call-16 exhibits an even more interesting pattern: for $C = 15$, it is *accepted* at $cv = 0.5$; *rejected* at $cv = 1, 1.5$; and again *accepted* at $cv = 2$. However, for $C = 20$, Call-16 is *accepted* at $cv = 0.5$; *rejected* at $cv = 1$; *accepted* at $cv = 1.5$; and again *rejected* at $cv = 2$. Table 3 further suggests that cv also plays a vital role in the route selection for the selected calls. For example, for $C = 5$, Call-16 is routed using links $0-8$; $7-0$; $8-4$ at $cv = 0.5, 1, 2$. However, the same call is routed using links $0-8$; $2-0$; $7-2$; $8-4$ at $cv = 2$. Similar observations can be made for $\{Call - 12; C = 15\}$ and $\{Call - 19; C = 20\}$. These results demonstrate the fact that service time variability plays a vital role in the optimal call and route selections in BPP, which, in turn, effect the total net revenue. This example thus illustrates the importance of accurately modelling service time variability for BPP.

Figure 2 shows the effect of varying cv and C, over a wider range of values, on the total gross revenue, the total delay cost and the total net revenue (NR = GR–DC) for the above illustrative example. The figures suggest that as cv or C increases, the total net revenue

Table 3 Solution obtained for the illustrative example

Call m	$C = 5$				$C = 10$			
	cv				cv			
	0.5	1	1.5	2	0.5	1	1.5	2
1	0-2	0-2	0-2	0-2	0-2	0-2	0-2	0-2
2	0-7	0-7	0-7	0-7	0-7	0-7	0-7	0-7
3	0-7; 7-5	0-7; 7-5	0-7; 7-5	0-7; 7-5	0-7; 7-5	0-7; 7-5	0-8; 8-5	0-8; 8-5
4	0-8; 8-4	0-8; 8-4	0-8; 8-4	0-8; 8-4	0-8; 8-4	0-8; 8-4	0-8; 8-4	0-8; 8-4
5	0-7; 1-0; 7-6	0-7; 1-0; 7-6	0-7; 1-0; 7-6	0-7; 1-0; 7-6	0-7; 1-0; 7-6	0-7; 1-0; 7-6	0-7; 1-0; 7-6	0-7; 1-0; 7-6
6	0-8; 1-0; 8-5	0-8; 1-0; 8-5	0-8; 1-0; 8-5	0-8; 1-0; 8-5	0-8; 1-0; 8-5	0-8; 1-0; 8-5	0-7; 1-0; 7-5	0-7; 1-0; 7-5
7	–	–	–	–	–	–	–	–
8	0-9; 2-0	0-9; 2-0	0-9; 2-0	0-9; 2-0	0-9; 2-0	0-9; 2-0	0-9; 2-0	0-9; 2-0
9	–	0-1; 1-3; 2-0	0-1; 1-3; 2-0	0-1; 1-3; 2-0	0-1; 1-3; 2-0	0-1; 1-3; 2-0	0-1; 1-3; 2-0	0-1; 1-3; 2-0
10	–	–	–	–	–	–	–	–
11	0-8; 1-0; 3-1; 8-5	–	–	–	–	–	–	–
12	5-7; 7-2	5-7; 7-2	5-7; 7-2	5-7; 7-2	5-7; 7-2	5-7; 7-2	5-7; 7-2	5-7; 7-2
13	–	–	–	–	–	–	–	–
14	0-1; 7-0	0-1; 7-0	0-1; 7-0	0-1; 7-0	0-1; 7-0	0-1; 7-0	0-1; 7-0	0-1; 7-0
15	0-9; 7-0	0-9; 7-0	0-9; 7-0	0-9; 7-0	0-9; 7-0	0-9; 7-0	0-9; 7-0	0-9; 7-0
16	0-8; 7-0; 8-4	0-8; 7-0; 8-4	0-8; 7-0; 8-4	0-8; 2-0; 7-2; 8-4	0-8; 7-0; 8-4	0-8; 7-0; 8-4	–	–
17	–	–	–	–	–	–	–	–
18	7-2; 8-7	7-2; 8-7	7-2; 8-7	7-2; 8-7	7-2; 8-7	7-2; 8-7	7-2; 8-7	7-2; 8-7
19	–	0-8; 8-5; 9-0	0-8; 8-5; 9-0	0-8; 8-5; 9-0	0-8; 8-5; 9-0	0-8; 8-5; 9-0	0-8; 8-5; 9-0	0-8; 8-5; 9-0
20	–	–	–	–	–	–	–	–
Gross revenue	5013	4948	4848	4707	4868	4747	4573	4368
Delay cost	132	128	183	266	190	256	306	429

Table 3 continued

Call m	$C = 15$				$C = 20$			
	cv				cv			
	0.5	1	1.5	2	0.5	1	1.5	2
1	0–2	0–2	0–2	0–2	0–2	0–2	0–2	0–2
2	0–7	0–7	0–7	0–7	0–7	0–7	0–7	0–7
3	0–7; 7–5	0–8; 8–5	0–8; 8–5	0–7; 7–5	0–7; 7–5	0–8; 8–5	0–7; 7–5	0–7; 7–5
4	0–8; 8–4	0–8; 8–4	0–8; 8–4	0–8; 8–4	0–8; 8–4	0–8; 8–4	0–8; 8–4	0–8; 8–4
5	0–7; 1–0; 7–6	0–7; 1–0; 7–6	0–7; 1–0; 7–6	0–7; 1–0; 7–6	0–7; 1–0; 7–6	0–7; 1–0; 7–6	0–7; 1–0; 7–6	0–7; 1–0; 7–6
6	0–8; 1–0; 8–5	0–7; 1–0; 7–5	0–7; 1–0; 7–5	0–8; 1–0; 8–5	0–8; 1–0; 8–5	0–8; 1–0; 8–5	0–8; 1–0; 8–5	0–8; 1–0; 8–5
7	–	–	–	–	–	–	–	–
8	0–9; 2–0	0–9; 2–0	0–9; 2–0	0–9; 2–0	0–9; 2–0	0–9; 2–0	0–9; 2–0	0–9; 2–0
9	0–1; 1–3; 2–0	0–1; 1–3; 2–0	0–1; 1–3; 2–0	0–1; 1–3; 2–0	0–1; 1–3; 2–0	0–1; 1–3; 2–0	0–1; 1–3; 2–0	0–1; 1–3; 2–0
10	–	–	–	–	–	–	–	–
11	–	–	–	–	–	–	–	–
12	5–7; 7–2	5–7; 7–2	5–7; 7–2	0–2; 5–8; 8–0	5–7; 7–2	5–7; 7–2	0–2; 5–8; 8–0	0–2; 5–8; 8–0
13	–	–	–	–	–	–	–	–
14	0–1; 7–0	0–1; 7–0	0–1; 7–0	0–1; 7–0	0–1; 7–0	0–1; 7–0	0–1; 7–0	0–1; 7–0
15	0–9; 7–0	0–9; 7–0	0–9; 7–0	0–9; 7–0	0–9; 7–0	0–9; 7–0	0–9; 7–0	0–9; 7–0
16	0–8; 7–0; 8–4	–	–	0–8; 2–0; 7–2; 8–4	0–8; 7–0; 8–4	–	0–8; 2–0; 7–2; 8–4	–
17	–	–	–	–	–	–	–	–
18	7–2; 8–7	7–2; 8–7	7–2; 8–7	7–2; 8–7	7–2; 8–7	7–2; 8–7	7–2; 8–7	7–2; 8–7
19	0–8; 8–5; 9–0	0–8; 8–5; 9–0	0–8; 8–5; 9–0	–	0–8; 8–5; 9–0	0–7; 7–5; 9–0	–	–
20	–	–	–	–	–	–	–	–
Gross revenue	4727	4563	4344	4073	4585	4407	4118	3842
Delay cost	285	328	459	451	380	437	442	537

decreases. This is expected since a higher cv or C causes either (i) a higher congestion related cost if the set of accepted calls remains unchanged; or (ii) calls with lower potential revenue getting accepted if they are associated with lower bandwidth demands. In either case, the total net revenue is expected to decrease. Further, when a higher cv or C causes the former (i), then the total gross revenue is expected to remain unchanged. However, when it causes the latter (ii), then the total gross revenue is also expected to decrease with an increase in cv or C. Hence, the total gross revenue in Fig. 2 either remains unchanged or decreases with an increase in cv or C. However, the change in the total delay cost, as cv or C increases, is non-monotonic. This, although appears counter-intuitive, can be explained as follows. When a higher cv or C does not cause any change to the set of accepted calls, then the delay cost is expected to increase. However, when a higher cv or C causes calls with lower bandwidth demands getting accepted, then the total delay cost is expected to decrease due to a decrease in congestion in the network.

4.2 Computations results

4.2.1 Test instances

For our computational study, we adopt the data generation scheme as reported by [5] to generate various network topologies ranging from 10 to 50 nodes. For $|N| = \{10, 20, 30, 40, 50\}$, a network topology is generated as follows. $|N|$ points are located randomly on a 100×100 grid. The degree of each node is also generated randomly such that it has a degree of 2, 3, or 4 with a probability of 0.6, 0.3 and 0.1, respectively. For the network topology generated, each link in the network is randomly assigned a bandwidth capacity (Q_{ij}) of 48, 96, 192 or 500 with equal probability. Call tables consisting of call origin (i), call destination (j), call demand (d^m), and call revenue (r^m)) are generated as follows. The total number of calls ($|M|$) is determined as a percentage (P) of the maximum number of calls possible for the network, which is given by $P|N|(|N| - 1)/200$. For a given call, the origin and destination nodes are randomly chosen. The bandwidth requirement of a call (d^m) is randomly generated from a uniform distribution between 20 and 40, and its revenue (r^m) is randomly generated from a uniform distribution between 10 and 50. The complete scheme for the generation of the network topology is described in Algorithm 2.

Algorithm 2 Network topology generation scheme

Generate $|N|$ points with distinct locations on a 100×100 grid
Randomly generate the degree requirement of each node $i \in N$
for all nodes $i \in N$ **do**
  if degree of node i is unsatisfied **then**
    Find node j closest to node i that has an unsatisfied degree
    if node j is found **then**
      Establish link (i, j)
    else
      Find node j closest to i that is not connected to node i, and establish link (i, j)
    end if
  end if
end for
if Network is not connected **then**
  Make necessary connections to make the network connected.
end if

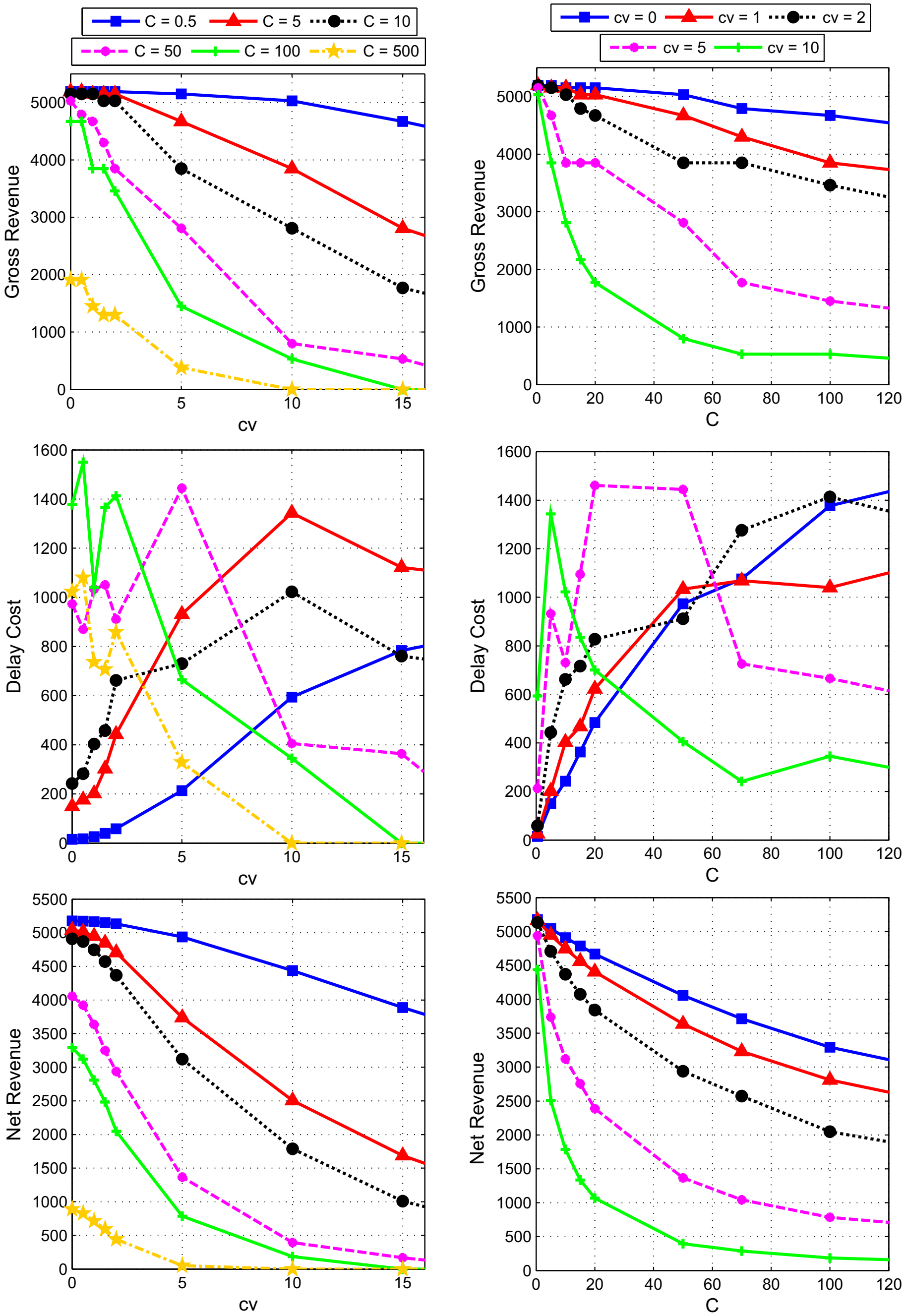

Fig. 2 Gross revenue (GR), delay cost (DC) and net revenue (NR) versus coefficient of variation of service times (cv) and unit delay cost (C) for the illustrative example

Using the data generation scheme described above, we generate 10 sets of networks for each value of $|N| = \{10, 20, 30, 40, 50\}$. For each of these networks, a call table is generated for $P = \{50, 60, 70, 80, 90\}$, where P is the percentage of the maximum possible types of calls (a call type is specified by an origin-destination node pair) for the given network that are included in the call table. Thus, we have $10 \times 5 \times 5 = 250$ different problem sets. Each of these sets is solved for 4 different values of cv ($cv = 0, 0.5, 1, 1.5$) and for 5 different values of C ($C = 0.5, 1, 5, 10, 15, 20$), which together result in $250 \times 4 \times 5 = 5000$ problem instances. For each of the test instances, when using the constraint generation method, we start with a priori set (H^1) of points to approximate the function $f(R_{ij}) = R_{ij}/(1 + R_{ij})$ using supporting hyperplanes $\widehat{f}(R_{ij})$ at these points. These points are generated such that the approximation error measured by $\widehat{f}(R_{ij}) - f(R_{ij})$ is at most 0.001 [9].

4.2.2 Computational results

The first set of experiments compares the performance of the Big-M based finite linearization and supporting hyperplane based constraint generation methods, the results of which are presented in Table 4. The results corresponding to each instance are averaged over 10 different networks. Table 4 shows that the constraint generation method clearly outperforms the finite linearization method in terms of CPU time. While the constraint generation method solves each of the problem instances within a couple of minutes, the finite linearization method leaves a significant optimality gap even after a CPU time limit of 1 hour.

In the remainder of the computational experiments, we use the supporting hyperplane based constraint generation method, and start with a priori set of points (H^1). The value of ϵ used in the convergence criterion is set at 10^{-6} in all the experiments. The results of the computational experiments, which are averages over 10 different networks, are presented for each combination of $|N|$, P, C and cv in Tables 5 and 6. While Table 5 reports the results for percentage of maximum possible types of calls set to $P = 50$, 60, and 70 %, Table 6 reports the results for higher percentage of maximum possible types of calls, i.e. $P = 80$ and 90 %. These tables report the total gross revenue (GR), the delay cost (DC) expressed as percentage of the total gross revenue, the minimum, maximum, and the average link utilizations, iterations (Iter.) of the proposed algorithm, and the CPU time (in seconds). The iterations and the CPU times clearly demonstrate the efficiency of our proposed exact solution method over a wide range of problem instances: it succeeds in finding optimal solutions to several instances with different unit delay costs and service time variability in 2 to 3 iterations within a couple of minutes, with the maximum CPU time being 2153 s (for $|N| = 50$; $P = 90$; $C = 0.5$; $cv = 1$).

The efficiency of our constraint generation method is also highlighted by comparing its results, both the optimal objective function values and CPU times, with those from the Lagrangean relaxation based solution method reported by [5] for the special case of $cv = 1$ ($M/M/1$ queue model for the links). For the completeness of the paper, the mathematical model and the Lagrangean relaxation based solution algorithm reported by [5] are briefly presented in the Appendix. The comparison of the the results are presented in Table 7, which demonstrates that our proposed solution method is much faster than the Lagrangean relaxation approach. Moreover, our proposed method solves the problem to optimality whereas the Lagrangean relaxation leaves an optimality gap of 2–7 % on an average, and 11 % in the worst case.

Table 4 Comparison between Big-M based finite linearization and supporting hyperplane based constraint generation method

$\|N\|$	P	C	$cv = 0.5$					$cv = 1.5$				
			Finite linearization[a]		Piecewise linearization			Finite linearization		Piecewise linearization		
			Gap %	CPU	Gap %	CPU	Iter.	Gap %	CPU	Gap %	CPU	Iter.
35	50	0.5	5.14	3602	$\leq 10^{-0.6}$	267	2.1	9.63	3602	$\leq 10^{-0.6}$	126	2.3
		5	18.89	3602	$\leq 10^{-0.6}$	40	2.1	31.11	3602	$\leq 10^{-0.6}$	49	2.2
		10	25.29	3602	$\leq 10^{-0.6}$	45	2.8	41.37	3602	$\leq 10^{-0.6}$	65	2.8
		15	29.55	3602	$\leq 10^{-0.6}$	34	2.3	44.50	3602	$\leq 10^{-0.6}$	34	2.2
		20	32.60	3602	$\leq 10^{-0.6}$	21	2	45.54	3602	$\leq 10^{-0.6}$	28	2.2
	60	0.5	6.07	3602	$\leq 10^{-0.6}$	279	2.2	10.15	3602	$\leq 10^{-0.6}$	281	2.7
		5	19.39	3604	$\leq 10^{-0.6}$	67	2.1	31.40	3602	$\leq 10^{-0.6}$	106	2.7
		10	26.03	3602	$\leq 10^{-0.6}$	50	2.2	41.11	3607	$\leq 10^{-0.6}$	63	2.2
		15	30.08	3602	$\leq 10^{-0.6}$	31	2	45.44	3603	$\leq 10^{-0.6}$	39	2.2
		20	32.21	3603	$\leq 10^{-0.6}$	38	2	48.14	3602	$\leq 10^{-0.6}$	35	2.4
	70	0.5	5.59	3602	$\leq 10^{-0.6}$	293	2	9.40	3602	$\leq 10^{-0.6}$	161	2
		5	18.68	3603	$\leq 10^{-0.6}$	128	2.5	30.94	3605	$\leq 10^{-0.6}$	130	2.5
		10	25.75	3602	$\leq 10^{-0.6}$	61	2	40.43	3602	$\leq 10^{-0.6}$	77	2.4
		15	28.86	3603	$\leq 10^{-0.6}$	63	2.5	45.36	3602	$\leq 10^{-0.6}$	67	2.2
		20	31.38	3602	$\leq 10^{-0.6}$	37	2.3	45.91	3605	$\leq 10^{-0.6}$	33	2
	80	0.5	5.70	3602	$\leq 10^{-0.6}$	205	2	9.48	3602	$\leq 10^{-0.6}$	297	2.3
		5	19.49	3602	$\leq 10^{-0.6}$	242	2.8	31.48	3602	$\leq 10^{-0.6}$	136	2.6
		10	25.75	3602	$\leq 10^{-0.6}$	81	2	41.83	3602	$\leq 10^{-0.6}$	87	2.1
		15	29.92	3602	$\leq 10^{-0.6}$	59	2.5	46.66	3606	$\leq 10^{-0.6}$	65	2
		20	32.50	3602	$\leq 10^{-0.6}$	70	2.5	47.64	3603	$\leq 10^{-0.6}$	60	2
	90	0.5	5.58	3602	$\leq 10^{-0.6}$	429	2.2	9.52	3602	$\leq 10^{-0.6}$	362	2.3

Table 4 continued

$\vert N\vert$	P	C	$cv = 0.5$					$cv = 1.5$				
			Finite linearization[a]		Piecewise linearization			Finite linearization		Piecewise linearization		
			Gap %	CPU	Gap %	CPU	Iter.	Gap %	CPU	Gap %	CPU	Iter.
		5	19.99	3603	$\leq 10^{-0.6}$	244	2.3	30.96	3612	$\leq 10^{-0.6}$	105	2.3
		10	26.31	3602	$\leq 10^{-0.6}$	160	2.4	41.75	3601	$\leq 10^{-0.6}$	102	2
		15	30.87	3603	$\leq 10^{-0.6}$	81	2.1	48.28	3602	$\leq 10^{-0.6}$	108	2.1
		20	32.90	3603	$\leq 10^{-0.6}$	60	2.3	48.80	3602	$\leq 10^{-0.6}$	86	2.6

[a] Big-M based finite linearization was terminated after 3600 s

Table 5 Computational results for networks different delay costs and coefficient of variation of service times (percentage of calls = 50, 60, 70 %)

N	P	C	$cv = 0$						$cv = 0.5$					
			GR	DC (%)	Avg.	Max.	Iter.	CPU(s)	GR	DC (%)	Avg.	Max.	Iter.	CPU(s)
20	50	0.5	1131	4.6	54	98	2.6	8.3	1122	4.7	51	98	3	11.1
		5	1044	12.9	43	87	2	3.2	1028	13.3	42	84	2.4	4.2
		10	1006	21.4	40	84	2.1	2.5	990	22.5	39	81	2	2.2
		15	943	26.3	35	75	2	1.9	925	27.2	33	74	2	2.1
		20	925	33.2	33	74	2	1.9	909	35.1	33	70	2	2.1
20	60	0.5	1326	3.7	55	98	2	12.7	1315	3.6	55	98	2	9.9
		5	1236	12.1	47	86	2	5.3	1216	12.1	45	84	2	4.3
		10	1182	18.9	42	81	2	3.0	1179	20.8	42	81	2	2.8
		15	1142	25.3	39	73	2	3.0	1116	26.0	37	73	2.5	3.1
		20	1108	31.1	37	72	2	2.4	1077	32.0	35	71	2	3.4
20	70	0.5	1462	3.9	57	98	2.3	15.1	1453	4.0	57	98	2.3	13.4
		5	1349	11.0	48	87	2	7.1	1344	12.1	48	87	2	6.2
		10	1297	17.3	43	79	2	4.4	1296	19.2	43	79	2.5	4.3
		15	1271	24.2	41	78	2	3.0	1249	25.3	40	73	2.5	4.2
		20	1229	29.4	38	73	2.1	3.8	1214	31.3	37	72	2.2	4.0
30	50	0.5	1599	3.2	48	97	2.2	28.7	1597	3.7	47	97	2.3	35.4
		5	1496	13.7	41	88	2	17.0	1478	14.6	39	88	2.2	18.4
		10	1406	20.2	35	84	2.5	17.2	1355	19.2	32	79	2.2	14.5
		15	1321	23.8	30	77	2	13.0	1291	23.9	29	72	2	9.5
		20	1263	27.7	27	70	2.2	9.0	1254	29.5	27	70	2.2	8.8
30	60	0.5	1758	2.5	48	96	2	44.4	1758	2.9	48	96	2	37.1
		5	1673	13.7	42	92	2	22.0	1638	13.7	41	88	2	21.8
		10	1552	18.1	35	86	2	18.7	1520	18.2	33	80	2.1	21.4
		15	1466	21.6	31	74	2.5	17.8	1461	23.3	30	73	2	10.9
		20	1454	28.1	30	73	2.2	10.8	1398	27.7	28	72	2.2	14.3
30	70	0.5	1945	2.9	52	96	2	40.6	1937	3.1	51	95	2.1	42.6
		5	1825	12.4	43	92	2	28.4	1792	12.5	42	87	2	30.2
		10	1732	18.3	38	85	2	21.6	1698	18.8	36	84	2.2	30.5
		15	1624	21.6	33	76	2	18.5	1600	22.4	32	73	2	18.9
		20	1586	26.7	31	73	2	11.9	1538	26.7	29	72	2	13.3
40	50	0.5	2285	3.6	58	98	2	140.9	2282	4.1	57	98	2	110.4
		5	2118	13.2	48	91	2	58.9	2081	13.3	46	89	2	49.2
		10	2016	20.3	42	83	2.3	39.3	1985	21.3	41	81	2.2	32.2
		15	1935	26.5	39	77	2.2	24.4	1899	27.8	38	77	2.3	26
		20	1850	31.7	35	77	2.2	25.5	1775	31.6	32	73	2.2	25.2
40	60	0.5	2455	3.6	59	98	2	181.1	2442	3.7	59	97	2	156.1
		5	2283	13.3	49	91	2	97.2	2249	13.8	47	89	2	72.6
		10	2161	19.9	43	84	2.3	62.8	2120	20.7	42	81	2.5	58.1
		15	2052	25.0	39	78	2.3	33.3	2026	26.5	38	77	2.2	28.7
		20	1974	30.2	35	76	2.2	26.9	1928	31.3	34	72	2.5	33.3

Table 5 continued

N	P	C	cv = 0 GR	DC (%)	Avg.	Max.	Iter.	CPU(s)	cv = 0.5 GR	DC (%)	Avg.	Max.	Iter.	CPU(s)
40	70	0.5	2706	3.0	61	97	2.1	282.7	2700	3.3	60	97	2.1	273.9
		5	2540	12.7	51	89	2	100.3	2502	13.1	50	87	2.1	97.6
		10	2406	19.2	46	84	2	72.7	2371	20.3	45	83	2.5	70.7
		15	2315	25.2	43	78	2.5	54.9	2262	26.0	41	77	2.5	50.3
		20	2204	29.4	38	77	2.3	46.6	2145	30.2	36	72	2.2	40.2
50	50	0.5	2927	3.7	57	97	2.3	1041.9	2916	4.1	57	97	2.2	1044.3
		5	2692	14.4	48	90	2.3	323.6	2649	15.0	47	88	2.4	296.5
		10	2523	21.1	42	84	2	145.1	2464	21.5	41	80	2.2	162.6
		15	2408	27.1	38	78	2.1	84.4	2324	27.1	36	76	2.3	103.9
		20	2276	31.5	34	74	2.2	63.9	2223	32.6	33	72	2.1	72.3
50	60	0.5	3150	3.4	59	97	2.6	1111.6	3144	3.9	59	97	2.1	847.3
		5	2903	13.9	49	89	2.2	345.4	2857	14.5	47	88	2.3	358.5
		10	2704	19.6	43	84	2	209.8	2672	20.8	42	82	2.1	194.2
		15	2596	25.7	39	76	2.3	142.9	2551	26.8	38	73	2.1	128.8
		20	2492	30.8	36	74	2.4	102.4	2441	32.2	35	72	2.2	105.0
50	70	0.5	3484	3.1	60	97	2.3	1196.1	3482	3.6	60	97	2	749.7
		5	3232	13.6	51	90	2.2	420.7	3179	14.0	50	88	2.1	402.9
		10	3036	19.6	45	85	2.5	335.5	2969	20.0	44	81	2.4	278.3
		15	2889	24.6	41	78	2.4	207.8	2833	25.5	40	76	2.1	167.1
		20	2783	29.6	38	75	2	119.3	2741	31.2	37	74	2.4	114.8

N	P	C	cv = 1 GR	DC (%)	Avg.	Max.	Iter.	CPU(s)	cv = 1.5 GR	DC (%)	Avg.	Max.	Iter.	CPU(s)
20	50	0.5	1097	3.8	49	94	2	6.4	1084	4.6	48	92	2	6.0
		5	1007	15.8	40	84	2	3.8	978	19.1	38	81	2.1	3.1
		10	927	22.9	34	74	2	2.7	905	28.7	33	70	2.1	2.6
		15	906	32.5	33	70	2	2.3	826	34.8	28	66	2	2.4
		20	835	36.7	28	67	2	2.2	698	34.0	21	49	2	2.8
20	60	0.5	1297	3.6	53	94	2	9.1	1293	5.2	53	93	2	7.4
		5	1207	15.3	45	82	2.5	6.0	1170	17.9	42	78	2	4.1
		10	1129	22.8	39	73	2.1	4.8	1079	26.5	36	66	2.1	3.7
		15	1079	30.0	36	67	2	4.5	994	32.1	30	66	2.1	3.2
		20	994	33.1	30	66	2.5	4.1	880	33.2	24	52	2.2	3.7
20	70	0.5	1434	4.3	56	97	2.1	19.6	1417	5.1	54	95	2	10.5
		5	1309	13.2	44	80	2	6.0	1283	16.4	42	78	2	4.4
		10	1256	21.9	41	73	2	5.5	1211	26.3	38	70	2.7	6.4
		15	1212	29.6	37	72	2.5	5.9	1094	30.0	31	66	2	4.4
		20	1109	31.7	31	66	2	3.6	989	31.8	26	56	2	3.8
30	50	0.5	1582	4.3	47	96	2.1	43.0	1568	5.5	46	95	2.3	50.6
		5	1412	15.2	36	86	2	23.7	1350	16.3	33	78	2	16.2
		10	1297	20.1	30	72	2	16.7	1268	24.3	28	70	2	11.4
		15	1249	26.8	27	70	2	13.4	1157	28.0	23	57	2.2	21.4
		20	1161	29.7	23	59	2	13.8	1084	31.9	20	56	2	13.3

Table 5 continued

N	P	C	$cv = 1$						$cv = 1.5$					
			GR	DC (%)	Avg.	Max.	Iter.	CPU(s)	GR	DC (%)	Avg.	Max.	Iter.	CPU(s)
30	60	0.5	1753	4.1	48	95	2.1	47.6	1740	5.4	47	94	2.1	57.2
		5	1558	13.5	37	86	2	36.0	1507	15.2	34	79	2.2	26.6
		10	1465	19.7	31	73	2.2	21.7	1407	22.7	29	70	2	18.0
		15	1394	25.3	28	72	2	20.3	1294	26.1	23	57	2.2	19.8
		20	1314	28.4	24	60	2	17.8	1218	29.9	20	55	2.6	17.2
30	70	0.5	1932	4.4	51	95	2.2	87.7	1905	5.0	48	94	2	63.5
		5	1747	14.2	39	85	2.1	42.3	1689	16.5	37	84	2.4	35.3
		10	1611	19.3	33	74	2	31.2	1554	22.4	31	70	2.5	24.3
		15	1541	24.9	30	72	2	26.6	1428	25.5	25	58	2.5	25.1
		20	1447	27.6	25	59	2.5	26.2	1344	28.7	21	56	2.7	25.5
40	50	0.5	2256	4.8	56	97	2	240.0	2222	5.6	54	95	2.1	130.3
		5	2038	15.7	44	83	2.5	87.2	1954	17.7	40	78	2.2	47.8
		10	1920	24.3	39	77	2.5	59.1	1787	26.3	34	73	2.3	37.8
		15	1776	29.4	33	73	2.2	49.4	1569	28.1	25	63	2.4	36.0
		20	1584	30.0	25	64	2.6	60.0	1478	32.7	22	57	2	31.0
40	60	0.5	2414	4.3	56	96	2.1	295.4	2395	5.6	54	95	2	157.1
		5	2194	16.0	45	85	2.7	143.4	2086	17.2	41	79	3.3	106.7
		10	2024	22.3	38	76	2.3	58.7	1929	25.9	35	71	2.3	44.7
		15	1920	28.9	34	71	2.3	66.3	1714	28.4	27	64	2.6	50.2
		20	1719	29.8	27	65	2.4	65.5	1605	32.5	24	56	2.3	48.4
40	70	0.5	2675	3.9	58	96	2	280.4	2657	5.1	58	95	2.1	163.2
		5	2431	15.0	47	84	2.1	155.9	2339	17.2	44	78	2.2	65.8
		10	2280	22.7	42	76	2.2	84.1	2151	25.3	37	72	2.3	57.2
		15	2140	28.1	36	72	2.3	102.9	1947	29.0	30	65	2.4	58.1
		20	1968	30.7	30	67	2.3	95.8	1791	32	25	60	2.3	55.2
50	50	0.5	2885	4.9	56	96	2.1	1629.6	2846	6.0	55	95	2.3	670.6
		5	2550	16.3	43	86	2.1	372.8	2452	18.6	40	80	2.2	207.5
		10	2369	24	38	75	2	186.1	2234	26.7	34	69	2	121.3
		15	2212	29.8	33	70	2.1	163.3	2043	32.5	29	62	2.1	105.6
		20	2079	35	29	64	2.2	194.4	1867	36.7	24	59	2.1	102.4
50	60	0.5	3117	4.9	58	96	2.3	1380.6	3074	5.9	56	95	2.2	757.3
		5	2739	15.3	44	85	2.1	584.3	2646	17.6	42	78	2	199.8
		10	2574	23.2	39	75	2.1	252.4	2428	26.0	35	70	2.1	167.5
		15	2423	29.4	34	71	2.4	313.2	2216	31.2	30	62	2	135.5
		20	2276	34.3	31	63	2.5	309.1	2028	35.1	25	60	2.4	224.1
50	70	0.5	3458	4.6	59	96	2.4	1532.8	3417	5.8	58	95	2.1	688.0
		5	3068	15.4	47	85	2.1	635.8	2928	16.7	43	79	2.4	398.8
		10	2844	21.7	41	76	2	386.9	2738	25.7	38	72	2.1	202.5
		15	2735	29.0	37	72	2.4	332.2	2511	31.0	32	63	2.1	174.7
		20	2550	33.1	33	65	2.3	344.1	2291	34.4	27	60	2.2	203.8

Table 6 Computational results for networks with different delay costs and coefficient of variation of service times (percentage of calls = 80, 90 %)

N	P	C	$cv = 0$						$cv = 0.5$					
			GR	DC (%)	Avg.	Max.	Iter.	CPU(s)	GR	DC (%)	Avg.	Max.	Iter.	CPU(s)
20	80	0.5	1636	4.3	63	98	2.3	29.3	1608	3.4	60	97	2.3	34.3
		5	1494	10.4	50	86	2	7.1	1475	10.6	48	82	2	7.5
		10	1457	17.6	46	81	2	5.4	1439	18.4	45	79	2	4.4
		15	1393	21.9	41	75	2	4.0	1386	23.7	41	73	2	3.9
		20	1375	28.1	40	72	2.1	3.5	1343	29.2	38	72	2	4.9
20	90	0.5	1762	3.6	65	98	2.3	29.5	1748	3.5	64	97	2.1	32.9
		5	1617	10.1	52	85	2	9.6	1603	10.7	51	83	2	7.6
		10	1575	17.1	48	82	2	5.4	1542	17.3	46	76	2.1	5.9
		15	1518	22	43	76	2	4.9	1490	22.8	42	75	2	4.1
		20	1490	27.6	41	75	2.2	4.4	1472	29.5	41	75	2	4.6
30	80	0.5	2120	2.6	52	98	2	69.9	2120	3.2	52	98	2	70.8
		5	2002	12.0	45	92	2	42.3	1967	12.1	44	87	2	34.7
		10	1898	17.4	40	85	2.2	34.4	1878	18.4	39	82	2	27.7
		15	1793	20.8	35	74	2	29.0	1770	21.8	34	74	2	28.3
		20	1741	25.2	32	74	2	17.6	1706	25.8	31	71	2	14.4
30	90	0.5	2362	3.0	57	98	2.8	165.4	2349	3.0	55	98	3.3	194.2
		5	2215	11.3	47	91	2.5	76.7	2156	10.5	45	86	2.1	60.0
		10	2116	16.5	42	85	2.2	37.0	2083	17.0	40	83	2	31.2
		15	2020	20.2	37	79	2.7	27.0	1989	20.9	36	75	2	22.4
		20	1952	23.9	34	73	2.7	23.9	1912	24.4	33	69	2.3	18.7
40	80	0.5	2911	2.9	63	97	2	208.9	2906	3.3	62	97	2	237.0
		5	2723	12	53	89	2	98.8	2706	13.1	52	88	2.1	102.1
		10	2609	19.1	48	85	2.1	96.8	2570	20.2	47	82	2.3	95.0
		15	2505	24.8	44	80	2.5	79.4	2427	25.0	41	77	2	48.2
		20	2387	28.9	40	77	2.4	54.2	2315	29.4	38	74	2.3	47.9
40	90	0.5	3139	3.1	65	98	2.2	324.3	3125	3.3	64	97	2.5	436.2
		5	2935	11.9	55	90	2	148.5	2903	12.6	54	88	2.2	168.8
		10	2799	18.1	49	84	2	97.8	2751	18.8	48	81	2.2	85.3
		15	2705	23.7	46	80	2.1	64.6	2628	24.0	43	76	2.3	61.3
		20	2602	28.3	42	76	2.2	50.0	2526	28.8	40	75	2.2	69.5
50	80	0.5	3766	2.8	62	97	2	764.1	3764	3.4	62	97	2.2	735.1
		5	3524	13.3	54	90	2.1	432.4	3470	13.9	53	89	2.2	449.7
		10	3314	19.3	48	85	2.2	302.5	3253	20	46	82	2.2	282.5
		15	3180	25	44	80	2.5	266.1	3100	25.6	42	76	2.2	232.5
		20	3062	30	41	77	2.4	179	3002	31.4	40	74	2.2	182.6
50	90	0.5	4038	3.2	65	98	2.2	946.3	4033	3.7	65	98	2.5	1094.0
		5	3734	12.5	55	90	2.2	516.3	3702	13.6	54	90	2.2	444.9
		10	3547	19.2	50	86	2.2	325.3	3466	19.5	48	82	2.1	304.7
		15	3395	24.4	46	80	2.6	285.6	3316	25.2	44	78	2.2	268.0
		20	3250	28.8	42	77	2.3	228.8	3175	29.9	40	75	2.4	282.0

Table 6 continued

N	P	C	$cv = 1$						$cv = 1.5$					
			GR	DC (%)	Avg.	Max.	Iter.	CPU(s)	GR	DC (%)	Avg.	Max.	Iter.	CPU(s)
20	80	0.5	1602	4.6	60	97	2.3	40.7	1579	5.3	58	95	2.3	25.1
		5	1461	13.2	46	81	2	8.6	1418	15.5	44	75	2.1	5.7
		10	1386	20.2	41	73	2	5.3	1350	25.1	39	71	2	5.1
		15	1345	27.7	39	72	2.1	4.8	1225	28.4	32	66	2.5	5.9
		20	1249	30.5	33	66	2.1	4.3	1112	30.2	27	60	2.1	4.2
20	90	0.5	1733	4.3	62	97	2	40.3	1711	4.9	61	95	2.5	30.2
		5	1575	12.7	48	82	2.1	9.0	1540	15.5	46	76	2	6.5
		10	1510	20.5	43	75	2	6.9	1472	25.4	41	75	2	6.1
		15	1465	27.8	41	75	2	6.0	1327	28.3	34	66	2	6.0
		20	1349	30.0	35	66	2.6	6.3	1212	30.4	29	59	2.1	5.4
30	80	0.5	2106	3.9	52	95	2	97.8	2092	5.1	51	95	2	65.5
		5	1907	13.2	41	85	2.1	58.7	1871	16.4	40	82	2.4	47.9
		10	1770	18.3	34	74	2.7	53.5	1698	20.5	32	66	2.1	21.0
		15	1692	23.2	31	66	2.2	29.1	1600	25.2	28	58	2.4	23.9
		20	1620	27.2	28	61	2.2	27.9	1508	28.7	24	57	2.2	35.5
30	90	0.5	2335	3.8	55	95	2.3	197.1	2324	5.2	55	94	2.3	163.8
		5	2123	12.5	43	85	2	64.9	2076	15.0	41	83	2.7	70.0
		10	1995	17.8	37	75	2	42.5	1910	19.6	33	69	2.1	24.4
		15	1905	22.2	33	69	2.2	31.1	1800	24.0	29	60	2.1	30.1
		20	1827	26.1	30	64	2.2	32.0	1729	28.7	27	58	2.1	25.9
40	80	0.5	2884	4.0	61	96	2	324.4	2865	5.2	60	95	2	178.1
		5	2628	14.9	49	85	2.1	176.1	2534	17.3	46	80	2.2	96.4
		10	2464	22.4	43	78	2.4	135.4	2301	24.1	38	71	2.2	68.2
		15	2292	26.8	37	72	2.2	107.1	2127	29.1	33	65	2.5	61.0
		20	2149	30.6	33	67	2.2	106.7	1934	30.9	27	59	2.4	68.2
40	90	0.5	3101	3.9	63	96	2	441.4	3073	4.8	61	94	2	214.2
		5	2824	14.2	51	85	2.1	192.9	2729	16.4	47	80	2.3	120.9
		10	2659	21.3	45	78	2.5	148.9	2505	23.5	40	72	2.2	77.8
		15	2510	26.6	40	73	2.3	129.9	2324	28.5	35	65	2.5	73.8
		20	2357	30.2	35	67	2.2	109.2	2121	30.6	28	60	2.1	66.5
50	80	0.5	3743	4.4	62	96	2.1	1482.1	3702	5.6	60	95	2.3	1000.7
		5	3338	15	49	85	2	639.8	3219	17.3	46	81	2.1	361.6
		10	3126	22.3	43	77	2.1	479.5	2975	25.8	40	72	2.2	311.9
		15	2957	28.5	39	72	2.4	448.4	2706	30.2	34	64	2.3	330.9
		20	2763	32.6	34	66	2.5	518.4	2466	33.1	28	61	2.4	261.8
50	90	0.5	4003	4.7	64	97	2.4	2352.8	3939	5.3	62	95	2	914.5
		5	3572	15.0	51	86	2.1	698.1	3430	17.0	47	81	2	399.3
		10	3332	21.8	45	77	2.1	517.5	3150	24.5	40	72	2.2	408.2
		15	3142	27.4	40	72	2.3	565.4	2891	29.2	34	65	2.3	406.0
		20	2955	31.5	35	67	2.1	441.0	2672	32.7	29	62	2.3	309.6

Table 7 Comparison between the proposed exact (constraint generation) solution method and Lagrangean relaxation (for $cv = 1$)

$\|N\|$	C	$P = 50$					$P = 60$					$P = 70$				
		Exact		Lagrangian			Exact		Lagrangian			Exact		Lagrangian		
		NR	CPU	NR	Gap	CPU	NR	CPU	NR	Gap	CPU	NR	CPU	NR	Gap	CPU
10	0.5	482	0.3	482	3.7	6.1	550	0.3	541	6.3	5.4	561	0.5	561	4.7	4.7
	5	373	0.1	373	0.4	10.4	413	0.3	407	4.1	7.2	423	0.3	418	3.5	8.3
	10	293	0.2	291	4.0	7.4	334	0.2	332	3.3	9.1	337	0.2	335	3.5	9.8
	15	247	0.1	247	1.6	9.7	276	0.2	276	3.8	10.4	276	0.2	276	4.0	9.5
	20	205	0.2	205	3.1	8.2	224	0.2	224	7.1	7.5	224	0.3	224	7.1	7.8
	Min.		0.1		0.4	6.1		0.2		3.3	5.4		0.2		3.5	4.7
	Avg.		0.2		2.6	8.3		0.2		4.9	7.9		0.3		4.6	8.0
	Max.		0.3		4.0	10.4		0.3		7.1	10.4		0.5		7.1	9.8
20	0.5	1385	5.8	1360	4.5	4.6	1589	9.9	1559	5.7	3.4	1719	16.3	1683	6.1	2.3
	5	1103	3.6	1101	1.6	7.1	1281	4.2	1276	2.0	7.3	1402	5.2	1402	1.4	6.9
	10	918	2.9	918	2.3	8.6	1097	4.1	1094	1.6	7.4	1207	4.8	1205	1.4	7.8
	15	790	2.7	787	3.3	11.5	965	3.8	965	1.6	7.8	1059	5.1	1059	1.7	7.1
	20	696	3.0	691	3.4	8.8	855	3.3	853	2.4	9.0	939	5.5	932	3.1	6.4
	Min.		2.7		1.6	4.6		3.3		1.6	3.4		4.8		1.4	2.3
	Avg.		3.6		3.0	8.1		5.1		2.7	7.0		7.4		2.7	6.1
	Max.		5.8		4.5	11.5		9.9		5.7	9.0		16.3		6.1	7.8
30	0.5	1360	64.4	1289	11.0	1.9	1571	78.5	1502	9.4	1.8	1689	53.3		8.0	3.3
	5	1035	25.1	1027	6.0	4.6	1220	73.7	1218	5.8	2.0	1364	50.1	1359	3.4	3.5
	10	878	18.7	868	5.0	6.0	1074	22.5	1064	3.8	6.7	1167	26.2	1151	4.5	6.7
	15	766	14.5	763	3.9	7.7	955	16.6	951	2.9	9.3	1009	22.2	988	7.7	8.3
	20	676	14.4	674	4.4	8.1	853	14.5	853	3.1	11.1	881	25.9	871	9.9	7.2
	Min.		14.4		3.9	1.9		14.5		2.9	1.8		22.2		3.4	3.3
	Avg.		27.4		6.1	5.7		41.2		5.0	6.2		35.5		6.7	5.8
	Max.		64.4		11.0	8.1		78.5		9.4	11.1		53.3		9.9	8.3

Table 7 continued

$\lvert N\rvert$	C	$P = 50$					$P = 60$					$P = 70$				
		Exact		Lagrangian			Exact		Lagrangian			Exact		Lagrangian		
		NR	CPU	NR	Gap	CPU	NR	CPU	NR	Gap	CPU	NR	CPU	NR	Gap	CPU
40	0.5	2638	336.8	2557	6.5	1.4	2655	454.8	2525	9.4	1.3	2925	349.4	2869	6.5	2.1
	5	2092	49.6	2079	2.9	9.8	2083	275.4	2074	2.8	2.2	2310	134.5	2298	3.0	5.9
	10	1778	55.9	1773	1.5	8.6	1767	54.1	1752	2.7	11.1	1961	71.1	1947	2.8	10.7
	15	1548	34.3	1548	1.5	14.2	1539	54.5	1538	1.9	11.5	1697	64.2	1695	3.1	11.9
	20	1371	32.2	1368	1.9	15.7	1358	61.6	1355	2.8	10.2	1493	83.6	1485	4.0	9.7
	Min.		32.2		1.5	1.4		54.1		1.9	1.3		64.2		2.8	2.1
	Avg.		101.8		2.9	9.9		180.1		3.9	7.2		140.6		3.9	8.1
	Max.		336.8		6.5	15.7		454.8		9.4	11.5		349.4		6.5	11.9
50	0.5	2728	927.4	2641	7.5	1.2	3007	918.7	2926	6.9	1.6	3581	1634.4	3477	7.4	1.0
	5	2172	386.6	2125	5.9	3.1	2403	512.0	2388	4.9	2.9	2936	529.2	2901	4.3	3.2
	10	1866	276.7	1819	5.0	4.2	2093	282.6	2057	4.7	5.0	2564	267.4	2549	2.9	6.5
	15	1617	195.3	1599	4.2	5.9	1847	225.2	1833	3.9	6.4	2258	282.5	2242	3.4	6.2
	20	1430	200.9	1423	3.8	5.7	1648	225.4	1634	4.1	6.3	2000	348.3	1988	4.7	4.9
	Min.		195.3		3.8	1.2		225.2		3.9	1.6		267.4		2.9	1.0
	Avg.		397.4		5.3	4.0		432.8		4.9	4.5		612.3		4.5	4.4
	Max.		927.4		7.5	5.9		918.7		6.9	6.4		1634.4		7.4	6.5

Table 7 continued

$\|N\|$	C	$P=80$					$P=90$				
		Exact		Lagrangian			Exact		Lagrangian		
		NR	CPU	NR	Gap	CPU	NR	CPU	NR	Gap	CPU
10	0.5	597	0.5	594	5.0	5.0	691	0.5	685	3.9	6.3
	5	450	0.3	444	3.2	7.0	541	0.3	541	1.8	9.3
	10	356	0.3	356	3.5	9.6	451	0.3	450	2.2	8.8
	15	293	0.3	293	4.2	11.0	382	0.3	380	4.7	8.0
	20	242	0.3	242	6.0	9.0	332	0.3	332	5.3	8.6
	Min.		0.3		3.2	5.0		0.3		1.8	6.3
	Avg.		0.3		4.4	8.3		0.3		3.6	8.2
	Max.		0.5		6.0	11.0		0.5		5.3	9.3
20	0.5	1811	25.4	1762	7.0	1.7	2010	47.3	1977	6.2	1.0
	5	1474	11.5	1473	1.7	3.6	1665	6.8	1658	1.7	7.0
	10	1282	4.6	1280	1.0	8.9	1450	5.2	1445	1.2	9.0
	15	1123	4.1	1122	1.3	10.1	1281	5.0	1279	1.4	9.2
	20	994	5.6	988	2.5	7.3	1148	5.8	1141	2.2	7.9
	Min.		4.1		1.0	1.7		5.0		1.2	1.0
	Avg.		10.3		2.7	6.3		14.0		2.5	6.9
	Max.		25.4		7.0	10.1		47.3		6.2	9.2
30	0.5	1808	72.6	1795	6.6	2.8	1897	193.1	1859	9.1	1.3
	5	1471	76.8	1466	3.5	2.7	1537	49.2	1526	4.9	4.7
	10	1262	29.9	1254	3.7	6.8	1338	34.7	1335	2.7	6.8
	15	1102	27.4	1080	6.7	7.4	1184	26.1	1182	2.5	9.0
	20	973	26.2	963	7.9	7.8	1043	33.7	1033	4.9	7.2
	Min.		26.2		3.5	2.7		26.1		2.5	1.3
	Avg.		46.6		5.7	5.5		67.3		4.8	5.8
	Max.		76.8		7.9	7.8		193.1		9.1	9.0

Table 7 continued

$\|N\|$	C	$P = 80$					$P = 90$				
		Exact		Lagrangian			Exact		Lagrangian		
		NR	CPU	NR	Gap	CPU	NR	CPU	NR	Gap	CPU
40	0.5	3241	386.8	3177	6.2	2.1	3476	519.4	3402	6.1	1.8
	5	2614	126.1	2614	2.3	6.4	2815	160.3	2793	2.9	5.8
	10	2216	187.0	2213	3.1	4.2	2426	127.4	2418	2.3	7.4
	15	1935	83.6	1930	4.1	9.4	2149	88.4	2149	1.5	11.0
	20	1725	78.5	1725	4.1	10.0	1930	86.0	1929	1.4	11.5
	Min.		78.5		2.3	2.1		86.0		1.4	1.8
	Avg.		172.4		4.0	6.4		196.3		2.8	7.5
	Max.		386.8		6.2	10.0		519.4		6.1	11.5
50	0.5	3981	1856.75	3864	7.0	1.1	4141	1908.19	3978	7.8	1.2
	5	3258	839.0	3219	4.2	2.3	3345	786.2	3290	5.0	3.0
	10	2820	807.4	2801	3.3	2.7	2906	549.5	2875	3.8	3.9
	15	2481	368.2	2456	4.3	5.4	2585	589.4	2564	3.3	3.7
	20	2215	532.9	2193	5.0	3.6	2323	439.4	2305	3.4	5.1
	Min.		368.2		3.3	1.1		439.4		3.3	1.2
	Avg.		880.8		4.7	3.0		854.5		4.7	3.4
	Max.		1856.8		7.0	5.4		1908.2		7.8	5.1

5 Conclusion

In this paper, we formulated a more generalized model of BPP with queuing delays by modeling the links, which process the calls arriving on the network, as $M/G/1$ queues. We presented a non-linear integer programming model, followed by two different linearization approaches. We further proposed an efficient constraint generation method to solve the resulting supporting hyperplane based linearized model to optimality. Through a computational study, we demonstrated the efficiency of the proposed solution algorithm in solving within minutes problem instances of the size of 50 nodes with varying service time variability and delay costs. The proposed method also outperforms the Lagrangean relaxation approach, reported in the literature for the special case of $cv = 1$.

The work reported in this paper can be extended in several ways. One such extension is to model the links as GI/G/1 queues, although the solution method for it is not immediately obvious. Another possible extension is to consider giving different priorities to calls from different classes of customers.

Appendix

We briefly present the mathematical model and the Lagrangian relaxation based solution approach reported by [5] for the special case when $cv = 1$ such that the links in the network are modeled as $M/M/1$ queues. For this, we introduce an additional set of variables W_{ij}^m as defined below:

$$W_{ij}^m = \begin{cases} 1 & \text{if call } m \text{ uses link}(i, j)\text{in either direction;} \\ 0 & \text{otherwise.} \end{cases}$$

The non-linear integer programming model of this problem is :

$$[P_{M/M/1}] : \max \sum_{m \in M} r^m Y^m - C \sum_{(i,j) \in E} \frac{\sum_{m \in M} d^m W_{ij}^m}{Q_{ij} - \sum_{m \in M} d^m W_{ij}^m} \tag{28}$$

$$\text{s.t. } X_{ij}^m + X_{ji}^m \leq W_{ij}^m \qquad \forall (i, j) \in E, m \in M \tag{29}$$

$$\sum_{m \in M} d^m W_{ij}^m \leq Q_{ij} \qquad \forall (i, j) \in E \tag{30}$$

$$W_{ij}^m, \in \{0, 1\} \qquad \forall (i, j) \in E, m \in M \tag{31}$$

$$(5), (7), (8)$$

On dualizing the constraint set (29) using non-negative lagrangean multipliers α_{ij}^m $\forall (i, j) \in E$ and $m \in M$, the problem $[P_{M/M/1}]$ decomposes into two sets of subproblems: (i) $[L1_{LR}^m]$ $\forall m \in M$; and (ii) $[L2_{LR}^E]$ $\forall (i, j) \in E$, as given below:

$$[L1^m_{LR}] : \max \ r^m Y^m - \sum_{(i,j)\in E} \alpha^m_{ij} \left(X^m_{ij} + X^m_{ij} \right) \tag{32}$$

$$s.t. \ (5), (7), (8)$$

$$[L2^E_{LR}] : \max \ \sum_{m\in M} \alpha^m_{ij} W^m_{ij} - C \frac{\sum_{m\in M} d^m W^m_{ij}}{Q_{ij} - \sum_{m\in M} d^m W^m_{ij}} \tag{33}$$

$$s.t \ (30), (31)$$

The solution algorithms to solve $[L1^m_{LR}]$, LP relaxation of $[L2^E_{LR}]$ and to generate feasible solutions are presented Algorithms 3, 4, and 5.

Algorithm 3 Solution algorithm for $[L1^m_{LR}]$

1: Solve $[L1^m_{LR}]$ a shortest path problem with α^m_{ij} as the link costs
2: **if** $(r^m > \sum_{(i,j)\in E} \alpha^m_{ij}(X^m_{ij} + X^m_{ij})$ **then**
3: $\quad (Y^m = 1)$
4: **else**
5: $\quad Y^m = 0$ and $X^m_{ij} = 0 \ \forall (i, j) \in E$
6: **end if**

Algorithm 4 Solution algorithm for LP relaxation of $[L2^E_{LR}]$ for link (i, j)

1: Sort the calls $(m \in M)$ in non-increasing order of α^m_{ij}/d^m. Use index m' to represent the calls in this order.
2: $m' \leftarrow 0$
3: **while** $m' < |M|$ **do**
4: $\quad m' \leftarrow m' + 1$
5: $\quad S \leftarrow \sum_{k<m'} d^k W^k_{ij}$
6: $\quad W_0 \leftarrow \min \left\{ 1, \frac{1}{d^{m'}} \left[(Q_{ij} - S) - \left(\frac{C d^{m'} Q_{ij}}{\alpha^{m'}_{ij}} \right)^{1/2} \right] \right\}$
7: $\quad$ **if** $\alpha^{m'}_{ij} > 0$ and $W_0 > 0$ **then**
8: $\quad\quad W^{m'}_{ij} \leftarrow W_0$
9: $\quad$ **else**
10: $\quad\quad W^{m'}_{ij} \leftarrow 0$
11: $\quad$ **end if**
12: $\quad$ **if** $W^{m'}_{ij} < 1$ **then**
13: $\quad\quad W^{m'}_{ij} \leftarrow 0 \ \forall \{k : m' < k \leq |M|\}$, and stop.
14: $\quad$ **end if**
15: **end while**

Algorithm 5 Solution algorithm for generating a feasible solution

1: $A_{ij} \leftarrow Q_{ij}\ \forall(i,j) \in E$
2: $DC \leftarrow 0$; $\Delta \leftarrow 0$
3: Sort the calls ($m \in M$) in non-increasing order of $v(L1^m_{LR})$ obtained from Algorithm 3. Use m' to represent the calls in this order.
4: Get the the values of $X^{m'}_{ij}\ \ \forall m' \in M,\ (i,j) \in E$ obtained using Algorithm 3
5: $m' \leftarrow 0$
6: **while** $m' < |M|$ **do**
7: $\quad m' \leftarrow m' + 1$
8: $\quad$**if** $d^{m'}(X^{m'}_{ij} + X^{m'}_{ji}) < A_{ij}\ \forall(i,j) \in E$ **then**
9: $\quad\quad \Delta \leftarrow C \sum_{(i,j)\in E} \frac{\sum_{k' \le m'} d^{k'}(X^{k'}_{ij}+X^{k'}_{ji})}{Q_{ij}-d^{k'}(X^{k'}_{ij}+X^{k'}_{ji})} - DC$
10: $\quad\quad$**if** $r^{m'} > \Delta$ **then**
11: $\quad\quad\quad Y^{m'} \leftarrow 1$
12: $\quad\quad\quad A_{ij} \leftarrow A_{ij} - d^{m'}(X^{m'}_{ij} + X^{m'}_{ji})\ \forall(i,j) \in E$
13: $\quad\quad\quad DC \leftarrow DC + \Delta$
14: $\quad\quad$**else**
15: $\quad\quad\quad Y^{m'} \leftarrow 0$ and $X^{m'}_{ij} \leftarrow 0\ \forall(i,j) \in E$
16: $\quad\quad$**end if**
17: $\quad$**else**
18: $\quad\quad Y^{m'} \leftarrow 0$ and $X^{m'}_{ij} \leftarrow 0\ \forall(i,j) \in E$
19: $\quad$**end if**
20: **end while**

The pseudocode to solve the BPP using Lagrangian Relaxation method is outlined in Algorithm 6.

Algorithm 6 Lagrangean relaxation based solution method

1: $\alpha^m_{ij} \leftarrow 0\ \forall(i,j) \in E$ and $m \in M$; $UB \leftarrow +\infty$; $LB \leftarrow -\infty$; $iter \leftarrow 1$; $max\_iter \leftarrow 500$; $\epsilon \leftarrow 10^{-6}$
2: **while** $(UB - LB)/LB > \epsilon$ AND $iter < max\_iter$ **do**
3: $\quad$Solve $L1^m_{LR}\ \forall m \in M$ using Algorithm 3.
4: $\quad$Solve $L2^E_{LR}\ \forall(i,j) \in E$ using Algorithm 4.
5: $\quad UB \leftarrow \sum_{m\in M} v(L1_{LR}) + \sum_{(i,j)\in E} v(L2_{LR})$
6: $\quad$Generate a feasible solution using Algorithm 5
7: $\quad LB \leftarrow v(P_{M/M/1})$
8: $\quad$Update α^m_{ij} using sub-gradient method.
9: $\quad iter \leftarrow iter + 1$
10: **end while**

References

1. Amiri, A.: The multi-hour bandwidth packing problem with response time guarantees. Inf. 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 (2012)
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.A., 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.L.: 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. **25**, 585–596 (2013)
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. Manage. 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. Villa, C., Hoffman, K.: A column-generation and branch-and-cut approach to the bandwidth-packing problem. J. Res. Nat. Inst. Stand. Technol. **111**, 161–185 (2006)