On the Capacitated Loss Network with Heterogeneous Traffic and Contiguous Resource Allocation Constraints

Chunxiao Chigan* Ramesh Nagarajan* Zbigniew Dziong* Thomas G. Robertazzi+
Bell Labs* State University of
Lucent Technologies New York at Stony Brook+

Keywords: Capacitated loss network, Multi-service, Contiguous resource allocation, call admission and packing, Markov Decision Theory

ABSTRACT
Given a capacitated loss network, traffic arrivals with different bandwidth demands and reward rates (heterogeneous) - in the case that each traffic demand must be assigned in contiguous position, the problem is to find call admission/packing algorithms, such that the objective function (long-run average revenue) will be maximized. With contiguous allocation constraint, the First-Fit (FF) and Best-Fit (BF) policy simulation was examined, assuming Poisson arrivals with exponential holding time. The blocking probabilities from these two policies are compared with the Complete-Sharing (CS) policy and Optimal-Complete-Partitioning (OCP) policy, which have no contiguous allocation concern. A loose optimal lower bound is obtained by using the theory of Semi-Markov Decision Processes (SMDP). The value-iteration algorithm is applied due to the huge cardinality of the system state space. Our systemic numerical study (among CS, FF, BF, OCP, SMDP policies) suggests that two novel heuristic admission/packing policies, Best-Fit with Reservation (BFR) and Moving Boundary First-Fit (MBFF), might have higher efficiency.

INTRODUCTION
The call admission and routing problem have been widely studied for variety of modern telecommunication networks. In this paper, the call admission/packing problem relates to the automation needs of the current SONET transport network switching. Due to hardware design limits, each traffic stream with a demand of STS-N must be allocated in the link contiguously once it is admitted, say, from \( n \) to \( n+N \). Here, \( n \) is the start position of this traffic allocation in the link, and \( n+N \) is the end position of its allocation in this link. Therefore, the issue itself is not only a call admission problem, dealing with whether or not to accept a traffic arrival once it comes, but also a packing problem, dealing with where to put the traffic once it is decided to accept that traffic.

This is an extremely complicated problem in the sense of obtaining the optimal solution. Even without admission control consideration, the packing problem (Dynamic Storage Allocation) itself is a NP hard problem [1]. Another research topic related to our problem is the call admission control in Multi-service loss systems (flow control problem in telecommunication), which is widely studied in the ISDN system and ATM network literature up to the present time [2,3,4,5]. Our problem is an outgrowth of the flow control problem with an extra contiguous allocation constraint. Indeed, there are few references studying this problem. To simplify the slot assignment and tracking that are needed at both ends of the transmission system for ISDN, CCITT recommend that the time slots assigned to a broad band call should form one of the four pre-defined “channels” with six time slots each. V. Ramaswami, et al [6] have applied a Markov Process Model in a performance study of this problem in the case of two types of traffic arrivals, with the constraint that the wide-band calls are placed in fixed-starting-point contiguous regions of the channel and the narrow-band calls have to be packed. Although the performance analysis of this model is extremely complicated due to the explosion of the state space, it is clear that the problem in [6] is a special case of our call admission/packing problem.

We give a systematic study of this problem. The Semi-Markov Process Model is applied to formulate the relaxed version of this problem, and the resulting optimal policy serves as the lower bound of our numerical study. First-Fit, Best-Fit, Optimal Complete Partitioning, as well as the well known Complete Sharing policies are studied under a variety of system parameters and traffic patterns.

PROBLEM DEFINITION AND COMPLEXITY ANALYSIS
Formally, the problem can be defined as follow:

Given:
- Capacitated loss network, with capacity \( C \) on the link.
- Traffic demands, with \( k \) possible different bandwidth requirements \( b_1, b_2, \ldots, b_k \), and reward parameters \( r_1, r_2, \ldots, r_k \) that is, there are heterogeneous traffic arrivals.
- Assume type-\( k \) traffic demand follows a Poisson process with mean arrival rate \( \lambda_k \), and the traffic

Here, \( n \) is the start position of this traffic allocation in the link, and \( n+N \) is the end position of its allocation in this link. Therefore, the issue itself is not only a call admission problem, dealing with whether or not to accept a traffic arrival once it comes, but also a packing problem, dealing with where to put the traffic once it is decided to accept that traffic.

This is an extremely complicated problem in the sense of obtaining the optimal solution. Even without admission control consideration, the packing problem (Dynamic Storage Allocation) itself is a NP hard problem [1]. Another research topic related to our problem is the call admission control in Multi-service loss systems (flow control problem in telecommunication), which is widely studied in the ISDN system and ATM network literature up to the present time [2,3,4,5]. Our problem is an outgrowth of the flow control problem with an extra contiguous allocation constraint. Indeed, there are few references studying this problem. To simplify the slot assignment and tracking that are needed at both ends of the transmission system for ISDN, CCITT recommend that the time slots assigned to a broad band call should form one of the four pre-defined “channels” with six time slots each. V. Ramaswami, et al [6] have applied a Markov Process Model in a performance study of this problem in the case of two types of traffic arrivals, with the constraint that the wide-band calls are placed in fixed-starting-point contiguous regions of the channel and the narrow-band calls have to be packed. Although the performance analysis of this model is extremely complicated due to the explosion of the state space, it is clear that the problem in [6] is a special case of our call admission/packing problem.

We give a systematic study of this problem. The Semi-Markov Process Model is applied to formulate the relaxed version of this problem, and the resulting optimal policy serves as the lower bound of our numerical study. First-Fit, Best-Fit, Optimal Complete Partitioning, as well as the well known Complete Sharing policies are studied under a variety of system parameters and traffic patterns.
holding time is **exponentially distributed** with mean \(1/\mu_k\).

**Constraints:**
- Each traffic demand must be assigned in contiguous position on the link.
- The total accepted traffic occupancy is no more than the link capacity \(C\), i.e.,
  \[
  \sum_k \frac{\lambda_k}{\mu_k} \cdot b_k \leq C
  \]
  where, \(k\) is the traffic type allocated in the link.
- Once a call traffic is accepted, the traffic demand will stay at the same link position during the traffic holding time.

**Objective:**
- Find a call admission/packing algorithm, such that the objective function (e.g., operator revenue/resource utilization or normalized blocking probability) will be maximized or minimized. Since our system is allowed to operate infinitely, we introduce the **long-run average revenue** \(R\) as the objective function. For policy \(l\), its associated objective function is given by
  \[
  R(l) = \sum_{S \in \Omega(l)} r(s) \cdot \pi_l(s)
  \]
  Where, \(r(s)\) is the revenue function for state \(S\), \(\pi_l(s)\) is the probability of state \(S\) under policy \(l\), and \(\Omega(l)\) is the set of all possible states under policy \(l\).

**Assumptions:**
- The rejected calls are lost and have no influence on future traffic arrival patterns.
- The admitted traffic is served immediately, this implies that \(\mu_j(n) = n \cdot \mu_j\).
- No departure is withheld in our loss system.

Due to the contiguous allocation constraint, the link state occupancies of our multi-service (corresponding to heterogeneous traffic arrival) loss network become much more complex, depending not only on how many of each traffic type are in the link, but also on the pattern of bandwidth occupancy, that are of interest to us. The link occupancies of our system with this occupancies pattern concern leads to an extremely huge cardinality of the link state space, compared to the system without contiguous constraint, where the total number of each traffic type can be chosen as the link state. That is, for the latter case in a heterogeneous traffic arrival environment, with the assumptions that the traffic arrivals follow Poisson processes and the traffic holding times are exponentially distributed, if we let vector \(X(t) = (X_1(t), X_2(t), \ldots, X_k(t))\) be the link state of the associated multi-service loss network, where \(X_i(t)\) is the number of type-\(i\) traffic in the link at time \(t\), our system then forms a continuous-time Markov process.

**Figure 1.** Markov chain for the multi-service loss link model with no contiguous constraint

Fig 1 is the state transition diagram of the Markov chain for our multi-service loss link model without contiguous constraint. Here the link capacity is 3 units, and there are 2 kinds of traffic demands, one with 1 unit of bandwidth requirement, another with 2 units of bandwidth requirement. Notice that within each state, “\(m \times n\)” indicate that there are \(m\) type-\(n\) traffic arrivals in the link.

**Figure 2.** Link occupancy patterns (with contiguous allocation constraint)
Figure 2 shows the link occupancy patterns in the same case with the contiguous allocation constraint. Notice that within super-state S1, all the three sub-states have 1 type-1 traffic demand, but due to the different position in the link, there is different impact to the incoming call. In state S11 and state S13, the link can accept an incoming type-2 call, but state S12 can not accommodate a type-2 traffic request. Therefore, with this contiguous constraint, the number of link states increases to 12 instead of 6 (the case that there is no contiguous constraint). In general, if the link capacity is $C$, and there are $k$ types of traffic demands, approximately, there are $2^C \sim K \times 2^C$ link states in the case with a contiguous constraint. For the case without contiguous constraint, approximately, there are at most

$$\frac{C^k}{b_1 \cdot b_2 \cdots b_k}$$

link states, where $b_k$ is the bandwidth requirement of type-$k$ traffic.

**SEMI-MARKOV DECISION MODELING**

With the Poisson arrival process and the exponential holding time assumptions, any given call admission/packing policy for our system can be simply modeled as a Markov process. However, what is the optimal admission/packing control policy among all the feasible policies? Markov Decision Theory provides a tool to find the optimal policy [7].

Due to the huge cardinality of the state space of our call admission/packing problem, this original problem is relaxed by taking off the contiguous constraint. Therefore, a loose lower bound (of blocking probabilities) will be given from the result of the corresponding Semi-Markov Decision Process (SMDP) model. To simplify the formulation of the Value-Iteration algorithm, beside the arrival epochs of the traffic call, we include the call departure epochs as fictitious epochs. We characterize the SMDP model for our call admission problem without contiguous constraint as follows:

1). System State Space:

We define the link state descriptor $l$, which gives the complete state information of the link at any time, as a vector

$$l = (n_1, n_2, \ldots, n_k)$$

where, $n_i$ is the number of type-$i$ traffic in the link.

Therefore, at each decision epoch, the system state descriptor $S$ can be defined as a vector,

$$S = (l, e)$$

where, the variable $e \in \{a_i, a_2, \ldots, a_k, d_1, d_2, \ldots, d_k\}$ represents the epoch event type, which could be either an arrival ($a_i$) or a departure ($d_i$) of type-$i$ traffic.

The system state space $I$ is defined as the set of all possible decision states,

$$I = \{(l, e) : l \in \Lambda, e \in \{a_1, a_2, \ldots, a_k\} \cup \{d_i \text{ for all } i \text{ if } n_i > 0\} \}$$

where $\Lambda$ is the set of all possible link states, which is defined as,

$$\Lambda = \{l : \sum n_i \cdot b_i \leq C, i = 1, 2, \ldots, k\}$$

2). State Action Space:

At each decision epoch, the system has to make the decision among all the possible actions. For our admission control problem, each time traffic arrives, the decision must to be made either as “accept” or as “reject”. While whenever traffic departs, the only action the system should take is “no action”. That is, the state action $A(s)$ is defined as,

$$A(s) \in \{ "accept", \ "reject", \ "no action" \} = \{0, 1, 2\}$$

3). Expected Time until a New State:

If at a present decision epoch, the system is at state $s$ and action $A$ is chosen, the expected time until the next decision epoch $\tau_s(A)$ is defined as,

$$\tau_s(A) = [\sum \lambda_i + \sum n_i \cdot \mu_i + \delta_{ej}(A) \cdot \mu_j]^{-1}$$

where, the event-action indicator $\delta_{ej}(A)$ is defined as,

$$\delta_{ej}(A) = \begin{cases} 
0 & \text{if at present decision epoch, a type-j traffic arrives, and "reject" action is taken} \\
1 & \text{if at present decision epoch, a type-j traffic arrives, and "accept" action is taken} \\
-1 & \text{if at present decision epoch, a type-j traffic departs, and "no action" is taken} 
\end{cases}$$

4). Probability of the System State Being at Next Decision Epoch:

If at a present decision epoch, the system is at state $s$ and action $A$ is chosen, the probability that the system will be at state $t$ at the next decision epoch is referred as $p_{st}(A)$, which is defined as follows, assuming the current system state $s = (l, a_i)$ or $s = (l, d_i)$:

- When action $A = 0$ ("reject") is made for the type-$i$ traffic at the present decision epoch:

$$p_{st}(A) = \begin{cases} 
\lambda_j \cdot \tau_s(A) & \text{if next decision epoch is due to a type-j traffic arrival, } 1 \leq j \leq k \\
n_j \cdot \mu_j \cdot \tau_s(A) & \text{if next decision epoch is due to a type-j traffic departure, } 1 \leq j \leq k 
\end{cases}$$
- When action \( A = 1 \) ("accept") is made for the class- \( i \) traffic at the present decision epoch:

\[
p_{st}(A) = \begin{cases} 
\lambda_j \cdot \tau_s(A) & \text{if next decision epoch is due to a type-} j \text{ traffic arrival, } 1 \leq j \leq k \\
n_j \cdot \mu_j \cdot \tau_s(A) & \text{if next decision epoch is due to a type-} j \text{ traffic departure, } 1 \leq j \leq k \\
(n_j + 1) \cdot \mu_j \cdot \tau_s(A) & \text{if next decision epoch is due to a type-} j \text{ traffic departure, } 1 \leq j \leq k 
\end{cases}
\]

- When action \( A = 2 \) ("no action") is made for the type- \( i \) traffic at the present decision epoch:

\[
p_{st}(A) = \begin{cases} 
\lambda_j \cdot \tau_s(A) & \text{if next decision epoch is due to a type-} j \text{ traffic arrival, } 1 \leq j \leq k \\
n_j \cdot \mu_j \cdot \tau_s(A) & \text{if next decision epoch is due to a type-} j \text{ traffic departure, } 1 \leq j \leq k \\
(n_j - 1) \cdot \mu_j \cdot \tau_s(A) & \text{if next decision epoch is due to a type-} j \text{ traffic departure, } 1 \leq j \leq k , \text{ and } j = i 
\end{cases}
\]

5). The Expected Reward Incurred:

In addition to the probabilistic structure of Markov models, a reward structure has been attached into the Markov Decision Process. We can think a reward as a random variable associated with the state occupancies and transitions. If at a present decision epoch, the system is at state \( s \) and action \( A \) is chosen, the expected rewards \( R_s(A) \) incurred until the next decision epoch is defined as:

\[
R_s(A) = \sum_i n_i \cdot r_i + \delta_{st}(A) \cdot r_j
\]

By data transformation, the SMDP model specified above can be converted into a Discrete-Time Markov Decision Process Model, such that for any stationary policy, the long-run average rewards per unit time are the same in both models (The proof can be found in [7]). Therefore, the optimal stationary call admission policy for the original SMDP model is the same as that from the converted Discrete-Time Markov Decision Process model.

Choose a data transformation factor \( \tau \) with

\[
0 < \tau \leq \min_{s,t} \tau_s(A)
\]

For example, in our problem under the interested parameters, \( \tau \) can be chosen as

\[
\tau = \left[ \sum_i \lambda_i + \sum_i \frac{C}{b_i} \cdot \mu_i \right]^{-1}
\]

The system state space, action space for the discrete-time MDP model is the same as the original SMDP model, while the one-step reward \( \tilde{R}_s(A) \) and the one-step transition probabilities \( \tilde{p}_{st}(A) \) of the discrete-time MDP model can be transformed from the SMDP model by,

\[
\tilde{R}_s(A) = \frac{R_s(A)}{\tau_s(A)}
\]

\[
\tilde{p}_{st}(A) = \begin{cases} 
\frac{\tau}{\tau_s(A)} \cdot p_{st}(A), & t \neq s \\
\frac{\tau}{\tau_s(A)} \cdot p_{st}(A) + (1 - \frac{\tau}{\tau_s(A)}), & t = s 
\end{cases}
\]

Also, notice that the transformed discrete-time model is a aperiodic Markov chain, since this transformation results in all the state with self-visit probability \( \tilde{p}_{st}(A) > 0 \).

According to the study results from [7,8,9]: Under unichain property assumption, if the system has finite state space, an optimal stationary policy which maximizes the long-run average rewards exists; The Linear Programming Algorithm, and the Policy-Iteration Algorithm converge under the unichain property; With an aperiodic Markov chain, Value-Iteration Algorithm converges.

It is straightforward that our system meets the unichain property. Therefore all the three algorithms for the MDP will converge for our system. However, both the Linear Programming Algorithm and the Policy-Iteration Algorithm are computationally unattractive for the system with large state space and action space, like our system, since both methods need to simultaneously solve a set of linear equations whose size equals the number of system states at each iteration. Therefore, we implement our SMDP model by the Value-Iteration algorithm [7].

Value-Iteration Algorithm (VIA):
In the Value-Iteration Algorithm, the value function $V_m(s)$ is computed for $m=1,2,\ldots$ recursively by the following:

$$V_m(s) = \max \left\{ \tilde{R}_s(A) + \sum_{t \in \mathcal{A}} \tilde{p}_m(A) \cdot V_{m-1}(t) \right\}$$

for all $s,t \in I$.

For a large $m$, the one-step difference $V_m(s) - V_{m-1}(s)$ will be very close to the maximum long-run average reward per unit time. The algorithm begins from an arbitrarily chosen function $V_0(s)$, and ends at iteration $m$ if

$$0 \leq U_m - u_m \leq \varepsilon \cdot u_m,$$

where

$$u_m = \min_{s \in I} \left\{ V_m(s) - V_{m-1}(s) \right\}$$

and

$$U_m = \max_{s \in I} \left\{ V_m(s) - V_{m-1}(s) \right\}.$$

The recursion of the algorithm terminates when $0 \leq U_m - u_m \leq \varepsilon \cdot u_m$, and the stationary policy $P^*(m)$ obtained is within $\varepsilon$ of optimal, where $\varepsilon$ is a prespecified tolerance (accuracy) number.

By this algorithm, the stationary optimal call admission control policy with heterogeneous traffic arrival is obtained. At each state $s \in I$, the optimal action $A^*(s)$ is the action according to the policy $P^*(m)$. That is, this stationary policy is the optimal rule which prescribes the single action whenever the system is observed in a state at a decision epoch. In other words, the optimal policy is the collection of the optimal actions for all the system states, which together will maximize the long-run average reward.

With $\varepsilon = 0.001$, the optimal policy for our call admission problem (normalized offered load range from 0.1 to 2.0) can be obtained by this VIA within 1000 iterations.

By this VIA algorithm, we get the optimal call admission policy and the state transition probabilities for our SMDP model. To get the steady state probabilities, we have a set of linear equations:

$$\pi_j = \sum_{i \in I} \pi_i \cdot \tilde{p}_{ij} \quad \text{for all } j \in I$$

$$\sum_{j \in I} \pi_j = 1$$

Here $\pi_j$ is the steady state probability for state $j$, and $\tilde{p}_{ij}$ is the transition probability from state $i$ to state $j$, which is the result from the VIA. This set of linear equations can be solved recursively. For the parameters of interest, the solutions can be obtained within 1500 iterations with $10^{-10}$ accuracy.

**POLICIES STUDY:**

The above SMDP model based policy can serve as a loose lower bound for the numerical study, since it has no contiguous constraint. Notice that the ideal call admission/packing policy for SONET switching automation needs should be simple to implement and also less memory and computation intensive. Therefore, a study will be conducted by comparing the numerical result of a Complete Sharing (CS) policy in the case without contiguous constraint, and that of Complete Sharing-First Fit (CS-FF), Complete Sharing-Best Fit (CS-BF) policies, as well as Optimal Complete Partitioning (OCP) policy.

**Complete Sharing Policy:**

The well-known complete sharing policy with no contiguous constraint works as follows: A customer $k$ with bandwidth requirement $b_k$ is blocked if and only if the total residual capacity (flexibly allocated) of the resource is less than $b_k$.

The CS policy is a greedy algorithm, it may be unfair to all the traffic types. However, observations from K. W. Ross and H. K. Tsang [10] indicate that for the case of 2 types of traffic arrivals, once the arrival rate ratio between traffic load for the small size traffic and that of the large size traffic is close or bigger than 1, the performance of the CS policy is very close to the optimal policy (obtained by Markov Decision Model). Arieh Gavious and Zvi Rosberg indicate in [11], that for equal importance indexes (defined as $\mu_i r_i$), the performance of CS policy is quite close to the optimal policy in the measure of the long-run average revenue.

Credited to Joseph Kaufman [12], a simple one-dimensional recursion has been developed for computing performance quantities of interest, which eliminates all difficulties of computation. In this study, our numerical study of the CS policy without contiguous constraint is from the analytical result based on Joseph Kaufman’s recursive algorithm.

**Complete Sharing-First Fit (CS+FF) and Complete Sharing-Best Fit (CS+BF) Policy:**

The CS policy gives us some basic sense of the multidimensional optimal access control of our loss system, but the CS policy above does not consider the contiguous constraint. With the contiguous constraint, the well known First-Fit and Best-Fit packing policy in dynamic storage allocation research field can be added to the CS call admission policy. The new hybrid policies are “Complete
Sharing admission + First-Fit packing policy (CS+FF)” and “Complete Sharing admission + Best-Fit packing policy (CS+BF)” respectively.

The CS+FF policy can be described as follows: Whenever a traffic request arrives, the system always starts to scan the link pipe from one end, say, the left end. Once it found the first available empty block (which is at least the same size as that of the traffic arrival) on its way of scanning and checking, this traffic request is admitted and packed in this first available empty block, otherwise, the traffic is blocked.

The CS+BF policy works as follows: Whenever a traffic request arrives, the system scans over the link pipe from one end to the other. If there is at least 1 empty block in the link pipe big enough to contiguously accommodate the requested traffic arrival totally, it will be filled into the block which has the smallest size among all the available empty blocks, otherwise, the traffic is blocked.

These two admission/packing policies under different system parameters and traffic patterns are examined by simulations. All the performance measures of interests are collected from the steady-state simulation period.

Optimal Complete Partitioning Policy:

Compared with the pure admission policies, the admission/packing policies bring “extra” blocking probability into the system due to the fragmentation effect from the contiguous constraint. In other words, an improper combination of admission and packing scheme may bring in more fragmentation impact to the system.

This suggests a simple strategy: What if one divides the whole link pipe into $k$ completely separate areas, such that there is no overlap between any two areas at all? Therefore, area $i$ is dedicated to type-$i$ traffic. This results in another well-known call admission policy - Complete Partitioning (CP) policy. Notice, unlike other sharing-packing kinds of policies, there is no extra fragmentation brought into the link due to the packing strategy for the CP policy. That is, the performance of CP policy for call admission with flexible allocation requirement is the same as that of call admission with contiguous allocation constraint problem. This brings us some hope that, the CP policy, though not an optimal call admission policy, may be a good call admission/packing policy after all.

In Optimal Complete Partitioning (OCP) policy, we maximize the long-run average revenue by the followings:

Max: \[ R_i(C_i) + R_2(C_2) + \cdots + R_k(C_k) \]

Subject to: \[ C_1 + C_2 + \cdots + C_k = C \]

Where, \( R_i(C_i) \) is the long-run average reward for type-$i$ traffic, which is defined as in (1). \( C_i \) is the assigned capacity for type-$i$ traffic. This is the variable need to be decided. \( C \) is the link capacity.

\[
R_k(C_k) = \frac{\sum_{n=0}^{b_k} r_k(n) \prod_{m=0}^{n-1} \frac{\lambda_k(m)}{\mu_k(m+1)}}{\sum_{n=0}^{b_k} \prod_{m=0}^{n-1} \frac{\lambda_k(m)}{\mu_k(m+1)}}
\]

where, \( r_k(n) \) is the revenue function when there is \( n \) type-$k$ traffic in the link, in our case, \( r_k(n) = n \cdot r_k \); \( \lambda_k(m) = \lambda_k; \mu_k(m) = m \cdot \mu_k; k = 3 \)

This problem can be solved by dynamic programming [13].

NUMERICAL RESULTS:

In our numerical study, the following parameters and notation are defined:

-Link capacity \( C=192; k=3; b_1 = 3, b_2 = 12, b_3 = 48 \)

-Traffic follows arrival rate fraction distribution \( p_1, p_2, p_3 \), for traffic of type-1, type-2, and type-3 respectively. That is, \( \sum_i p_i = 1 \), and, if the total traffic mean arrival rate is \( \lambda \), the mean arrival rate \( \lambda_i \) for traffic type-i is: \( \lambda_i = \lambda \cdot p_i \). Three sets of traffic arrival rate fraction distribution scenarios are of interest in our study.

-Set 1 (uniform traffic arrival rate):
  \[ p_1 = \frac{1}{3}, p_2 = \frac{1}{3}, p_3 = \frac{1}{3} \]

-Set 2 (biased traffic arrival rate 1):
  \[ p_1 = \frac{1}{13}, p_2 = \frac{4}{13}, p_3 = \frac{8}{13} \]

-Set 3 (biased traffic arrival rate 2):
  \[ p_1 = \frac{8}{13}, p_2 = \frac{4}{13}, p_3 = \frac{1}{13} \]

-Load of traffic arrival type-$k$ : \( \frac{\lambda_k}{\mu_k} \)

-Normalized offered load in this study:

\[
L = \frac{1}{C} \sum_i b_i \cdot \frac{\lambda_i}{\mu_i}, \text{ where i = 1, 2, 3}
\]

Therefore, for the same normalized offered load, there could be different kinds of combinations among all \( \lambda_i \)’s and \( \mu_i \)’s. In this study, we assume all the traffic has the same mean holding time \( \mu \). Therefore, the normalized offered load in our study is calculated by
\[
L = \frac{1}{C} \cdot \sum_i b_i \cdot \frac{\lambda_i \cdot p_i}{\mu_i} = \frac{\lambda}{\mu} \cdot \left( \sum_i b_i \cdot p_i \right) \cdot \frac{1}{C}
\]

- Numerical study objectives:
  - Individual traffic blocking probability \( P_{b1}, P_{b2}, P_{b3} \)
  - Normalized blocking probability
    \[
P_b = \sum_i \frac{\lambda_i \cdot P_{bi} \cdot r_i}{\mu_i} / \sum_i \frac{\lambda_i \cdot r_i}{\mu_i}
\]

Notice that the blocking probability \( P_b \) is normalized by the reward parameter \( r_i \) of all the traffic arrival \( i \).
Therefore, the optimal policy for minimizing normalized blocking probability is equivalent to the optimal policy which maximizes the long-run average revenue.

Under uniform traffic arrival rate \((1/3, 3/11, 3/12)\), the performance behavior (figure 3) of CS policy is only slightly worse than that of lower bound policy (notice that these two policies are without contiguous constraint). Impressively, the performance behavior of the OCP is much better than that of CS+FF and CS+BF policies when the traffic load is bigger than 0.5. As traffic load increasing to 1.2, it is converge to that of the optimal lower bound. Therefore, in this case, OCP is not a bad policy to choose.

Under the biased traffic arrival rate 1 distribution \((1/13, 4/13, 8/13)\), the overall normalized performance of the OCP policy converges to the pure CS policy with no contiguous constraint, which is up to 2% higher than that of the optimal lower bound policy with no contiguous constraint (figure 4). Obviously, the real lower bound with contiguous constraint will be tighter than the one we have (with no contiguous constraint), therefore, the OCP policy is different from the real lower bound less than 2% under any traffic load tested here. In this scenario, the overall performance behavior of the OCP is better than that of CS+FF as well as CS+BF policy.

Under the biased traffic arrival rate 2 distribution \((8/13, 4/13, 13/13)\), the pure CS policy with no contiguous constraint is almost the optimal policy, not only in terms of the overall performance behavior (figure 5), but also in terms of individual traffic type (figures omitted). The performance behavior of the OCP is poor under this scenario (much worse than CS+FF and CS+BF), especially when the traffic load is less than 0.9. Once the traffic load gets larger than 0.9, the OCP policy is comparable to both CS+FF and CS+BF policies. Notice in this case, the overall performance of OCP is piecewise like.
Figure 5 comparison of normalized blocking probability under biased traffic arrival rate 2 for different policies

**Observations:**
- Under all traffic scenarios tested: the CS+BF policy has a better overall normalized blocking probability than that of CS+FF policy; CS+BF and CS+FF policies have a smaller blocking probability for small size traffic than that of pure CS policy with no contiguous constraint, while the CS policy performs better for large size traffic; The absolute normalized blocking probability difference between CS+FF/CS+BF and pure CS policy is less than 9%, and the relative performance behavior is much bigger; In general, the performance difference at light-load and heavy-load is smaller than that at medium-load.
- As the fraction of small size traffic arrival rate becomes bigger, the pure CS policy converges to the optimal policy (from SMDP) without contiguous constraint, and this loose lower bound is further away from our three upper bound policies with contiguous constraint.
- When the fraction of larger size traffic arrival rate becomes bigger, the normalized performance of the OCP policy (as one of the upper bound policies with contiguous constraint) is very close to that of the optimal policy with no contiguous constraint (therefore, the loose lower bound). Therefore this policy might be an ideal choice for our admission/packing task.

**CONCLUSION:**
The numerical study for the call admission/packing policies (OCP, CS+FF and CS+BF) shows that, under different traffic scenario, partial OCP and partial CS+FF/CS+BF policies, namely, Best-Fit with Reservation (BFR) and Moving Boundary First-Fit (MBFF), might be the ideal call admission/packing policies. The moving boundary will probably depend on the traffic arrival rate fraction, traffic load, and traffic size, as well as link capacity. More about these policies will be presented in the future papers.

**References**