IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 8, 2018, accepted January 3, 2019, date of publication January 11, 2019, date of current version February 4, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2892063

Naïve Bayes Classifier-Assisted Least Loaded Routing for Circuit-Switched Networks

LONGFEI LI[1], YA ZHANG[1], WEI CHEN[2], SANJAY K. BOSE[3], (Senior Member, IEEE), MOSHE ZUKERMAN[4], (Fellow, IEEE), AND GANGXIANG SHEN[1], (Senior Member, IEEE)

[1]School of Electronic and Information Engineering, Soochow University, Suzhou 215006, China
[2]Key Laboratory of New Fiber Technology of Suzhou City, Jiangsu Hengtong Fiber Science and Technology Corporation, Suzhou 215200, China
[3]Department of EEE, IIT Guwahati, Guwahati 781039, India
[4]Department of EEE, City University of Hong Kong, Hong Kong

Corresponding author: Gangxiang Shen (shengx@suda.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61671313 and in part by the Science and Technology Achievement Transformation Project of Jiangsu Province, China, under Grant BA2016123.

ABSTRACT Circuit switching is a *de facto* switching technology widely employed in today's networks where the conventional approaches to routing have remained unchanged for many years. This paper develops a new and very different methodology, by incorporating a supervised naïve Bayes (NB) classifier, to assist least loaded (LL) routing and to further improve its performance that has remained the best among all the routing approach for the past several decades. Specifically, by iteratively learning the information of historical network snapshots, the NB classifier predicts potential future circuit blocking probability between each node pair if a service connection is established via a certain route between the node pair. The snapshots are taken for each service request arriving at an operating network that keeps on accepting and releasing dynamic service connections and records the number of busy capacity units on each link at each snapshot instance. The candidate route for serving a new request is selected based on both link loads and the potential future blocking probability in the entire network in case that this route is indeed used. The performance of the proposed approach is studied via simulations and compared with the conventional LL algorithm. The results indicate that the supervised NB classifier-assisted LL routing algorithm effectively reduces the blocking probability of service connections and outperforms the conventional LL routing algorithm. To speed up the learning process (which is based on a large number of network snapshots), we also develop a framework to incorporate the proposed approach in a parallel learning system. A network control system supporting online NB classifier-assisted LL routing algorithm is also described.

INDEX TERMS Machine learning, naïve Bayes classifier, least loaded routing, blocking probability, circuit switched network.

I. INTRODUCTION

Routing algorithms are essential for the efficient operation of circuit-switched networks and have been studied for many years. In general, these can be categorized as fixed shortest path routing [1]–[6], fixed-alternate path routing [7]–[10], and adaptive routing [11]–[16]. Fixed shortest path routing always selects a fixed shortest route between a pair of nodes to establish its service connections. Fixed-alternate path routing works with a set of routes for service establishment, rather than a single one. These are tried in sequence for the service establishment until all the routes are tried. Adaptive routing, also referred to as online routing, does not have a set of predetermined routes. Instead, it selects the most efficient route for each service request based on the current network status, e.g., capacity utilization on each link. Of these, the Least Loaded (LL) routing algorithm (or least congested routing algorithm) [17]–[21] has so far been the most efficient, exhibiting the lowest overall average connection blocking probability, and has remained so for the past several decades. Almost 70 years ago, Shannon [22] predicted the use of "machines for handling routing of telephone calls based on the individual circumstances rather than by fixed patterns." As a step towards a realization of this prediction, this paper achieves further improvement of LL routing performance by incorporating a machine learning approach in the route selection process of the LL algorithm, which can further reduce potential blocking of service connections.

Circuit switching is the *de facto* switching technology widely adopted in today's networks with very little change over the past several decades. We approach this problem here by redefining the routing methodology for the circuit-switched network by incorporating machine-learning

techniques. This allows us to perform significantly better than the benchmark routing performance set by the traditional LL routing algorithm.

A. MACHINE LEARNING TECHNIQUES IN COMMUNICATION NETWORKS

Machine learning (ML) techniques have become popular in many applications because they can provide frameworks for solving difficult problems. They have also been used to solve optimization problems in telecommunication networks. For example, for optical networks, Huang *et al*. [23] presented a set of intelligent pre-adjustment strategies enabled by machine learning to tackle spectrum defragmentation. Ohba *et al*. [24] applied Bayesian inference to a virtual network reconfiguration framework and introduced the Bayesian Attractor Model to infer the current traffic conditions. Barletta *et al*. [25] considered the probability that a candidate lightpath fails to meet Quality of Transmission (QoT) requirement and predicted this probability using random forests. Morales *et al*. [26] proposed a flow controller that enables metro controllers to share traffic modeling information with the core controller. Chen *et al*. [27] proposed a framework for a knowledge-based autonomous service provisioning which is enabled by a deep neural network-based traffic estimator for multi-domain software-defined elastic optical networks. Samadi *et al*. [28] proposed a neural network-based cognitive scalable method for dynamic provisioning of optical physical resources without need of prior network specific knowledge. Meng *et al*. [29] provided a self-learning network that uses real-time monitoring together with Markov Chain Monte Carlo simulations. Similarly, in wireless networks, Forster [30] presented a survey of machine learning applications to data routing in wireless sensor networks (WSNs) and mobile ad-hoc networks (MANETs). Russell *et al.* [31] presented an improved wireless adaptive routing protocol using machine learning techniques. Lee *et al*. [32] applied reinforcement learning for wireless network management to reduce protocol overhead and improve packet delivery ratio.

ML techniques have also been used in the circuit-switched network. Boyan and Littman [33] first applied the reinforcement learning technique for routing in circuit-switched networks. They presented a Q-routing algorithm, which discovers efficient routing policies in a dynamically changing network. Choi and Yeung [34] subsequently extended this to propose a gradient algorithm based on Q-routing. Recently, Li *et al*. [35] also employed a technique similar to reinforcement learning to train a set of fixed routes that are the best to establish service connections in circuit-switched optical networks. Leung *et al*. [36] proposed a neural network-based approach for blocking probability evaluation in optical networks, which can significantly accelerate its blocking probability evaluation. However, these circuit-switched routing algorithms are fixed route oriented, which significantly limits their overall routing performance due to inflexibility in route selection when provisioning online service connections.

B. SUMMARY AND OUR CONTRIBUTION

In the circuit-switched network, for the past several decades, the LL routing algorithm has been used as a benchmark because it generally does best in terms of lower connection blocking probabilities. Even though ML techniques have been tried in various areas, including communication networks, to the best of our knowledge, no ML technique has been applied to enhance the routing performance of a circuit-switched network with adaptive routing. The reason behind this is the large size of the state space of a circuit-switched network with adaptive routing. Accordingly, the key novelty and contribution of this paper is to incorporate an ML approach in the LL routing algorithm. This enables the ML technique to provide extra information, in addition to the link traffic load information, to the LL routing algorithm to decide the best candidate route for provisioning service connections. Specifically, we extend the supervised naïve Bayes classifier based on network snapshots that record historical network state information. This is then used to decide the future connection blocking probability of an entire network if a service connection is set up on a specific candidate route between a pair of nodes. Based on this, we can decide the best candidate route, i.e., with the lowest future connection blocking probability of the network, to serve a service request. The key idea behind this is to ensure that the establishment of a service connection along the selected route would least impact the successful establishment of future service connections. Our results show that this ML-assisted LL algorithm significantly outperforms conventional LL routing. In addition, to speed up the simulation of a learning process based on a large number of network snapshots, we also develop a framework to incorporate the proposed approach in a parallel computing system. A network control system supporting naïve Bayes classifier-assisted LL routing algorithm is also described.

The remainder of this paper is organized as follows. In Section II, we present the machine learning-assisted LL routing algorithm, where we give the motivation for employing ML, introduce the supervised naïve Bayes classifier, and describe how this can be used to enhance the LL routing algorithm. In Section III, we present a network control system for implementing the proposed ML-based LL routing algorithm in a real network. We evaluate the performance of the proposed approach in Section V, where a framework to incorporate the proposed approach in a parallel computing system is designed for speeding up the learning process based on a large number of network snapshots. This is used to simulate and study our proposed approach and compare its performance with other routing schemes. We conclude the paper in Section VI.

II. MACHINE LEARNING-ASSISTED LEAST LOAD ROUTING ALGORITHM

Machine learning techniques can be broadly divided into three main categories, i.e., supervised learning, unsupervised learning, and reinforcement learning. In this study, we employ the supervised naïve Bayes classifier [37], which

belongs to the category of supervised learning. This is used to enhance the performance of the LL routing algorithm.

We first introduce the motivation for developing our ML-assisted LL routing algorithm. We then describe the basic concept of the supervised naïve Bayes classifier and apply this ML technique to assist LL routing for better performance.

A. OCCASIONAL INEFFICIENCY OF LL ROUTING ALGORITHM

Although the LL algorithm is the most efficient to date for routing in circuit-switched networks, it can still be improved because under certain circumstances, some inefficiency is observed that may lead to poor performance. Fig. 1 shows such an example.

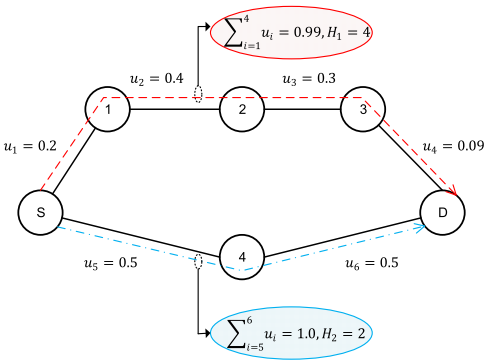


FIGURE 1. An example of the inefficiency of least loaded routing.

Consider the network in Fig. 1, where the capacity utilization of link i is denoted by u_i. For a connection request between nodes s and d, the LL routing algorithm would consider the link capacity utilization as the cost of each link to search for a path between the node pair with the smallest sum cost. Accordingly, in this example, the LL routing selects the route (S-1-2-3-D) with a sum cost of 0.99 though there is a shorter route (S-4-D) with a trivially higher sum cost of 1.0. In this case, the selected LL route is longer than the second route and consequently consumes more link capacity overall to establish the service connection. This presents a dilemma on whether we should choose the LL route with the least congestion or the shorter route with less overall capacity usage. This dilemma has not been addressed in the conventional LL routing studies due to its high challenge. In this study, we address this by using ML-assisted LL routing to improve performance in terms of connection blocking probability over what would otherwise be achievable by the simple LL routing algorithm with the smallest sum cost.

B. SUPERVISED NAÏVE BAYES CLASSIFIER

In the area of machine learning, naïve Bayes classifiers are simple classifiers that are based on Bayes' theorem and on strong (naive) independence assumptions among the various features. It is capable of calculating the probabilities of different output classes for each input instance.

Let a vector $\boldsymbol{X}$ represent a problem instance, and $x_1, \ldots, x_n$ represent n features of an instance. Assume that there are k class labels, $C_1, \ldots, C_k$. Naïve Bayes is a conditional probability model, represented by $P(C_k \mid \boldsymbol{X})$, which denotes the instance probability of C_k given $\boldsymbol{X}$. Using Bayes' theorem, we can express this conditional probability as

$$P(C_k \mid \boldsymbol{X}) = P(C_k \mid x_1, \ldots, x_n) = \frac{P(C_k) \cdot P(x_1, \ldots, x_n \mid C_k)}{P(x_1, \ldots, x_n)} \tag{1}$$

The "naive" conditional independence assumption assumes that each feature is conditionally independent of other features, and the instance probability can therefore be written as

$$P(C_k \mid \boldsymbol{X}) = \frac{P(C_k) \cdot \prod_{j=1}^{n} P(x_j \mid C_k)}{\prod_{j=1}^{n} P(x_j)} \tag{2}$$

C. SUPERVISED NAÏVE BAYES CLASSIFIER-BASED LEAST LOAD ROUTING ALGORITHM

We propose to improve the routing performance of the LL routing algorithm by incorporating the supervised naïve Bayes classifier. This is used to predict the future potential connection blocking probability of an entire network if a service connection is established on a certain candidate route between a pair of nodes, according to the current network capacity utilization status. Based both on this predicted blocking information and the current load on each candidate route, we select the one that has *the best combination of both low load and a small impact on future service connection establishment*. We expect that this will help the LL routing algorithm to reduce further the overall service connection blocking probability in the network.

It should be noted that we use the supervised naïve Bayes classifier for this prediction because it is easy to implement with an online learning process. Of course, there is possibility of employing other ML algorithms for similar purposes, which is however out of the scope of this study.

1) NETWORK SNAPSHOT

In the supervised naïve Bayes classifier, a vector $\boldsymbol{X}$ represents a problem instance. Here we define the network snapshot as a part of the classifier instance. This network snapshot records the information on the capacity status of each network link. Whenever a new service request between a pair of nodes arrives, we record the current network link capacity status as a *snapshot*. With time, we can then form a sequence of network snapshots as shown in Fig. 2. In each snapshot, the information on the total link capacity and capacity used on each link is recorded. For example, in Fig. 2, at t_i there is a service request arriving and the current link state is recorded as $(W_j^{(t_i)}, U_j^{(t_i)})$, where $W_j^{(t_i)}$ denotes the total capacity on link j and $U_j^{(t_i)}$ denotes the capacity used on link j. It should be noted that these snapshots are not independent over time, but change with new service establishment and old service departure as in a Markov chain.

Though it is possible for the link total capacity $W_j^{(t_i)}$ to vary with time, we assume that it is fixed. For a

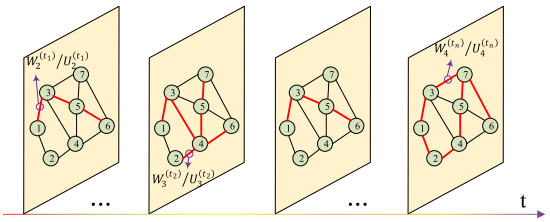


FIGURE 2. Network snapshots with time.

circuit-switched network, e.g., telephony network, TDM-based circuit-switched network, and wavelength division multiplexing (WDM) network, it is practically justifiable to assume a link to have a fixed capacity. Though it is possible that, after some time, a link is upgraded with more capacity, the time scale for that would typically be of the order of a half-year or one year for a backbone network. This would be much longer than the time interval of service connection provisioning. In particular, we represent the network snapshot by a vector

$$S^{(i)} = \left[U_1^{(i)}, \cdots, U_j^{(i)}, \cdots, U_L^{(i)}\right]^T \tag{3}$$

Here superscript i denotes the i^{th} network snapshot, which is the one when the i^{th} service connection request arrives, L is the total number of network links, and $U_j^{(i)}$ is the total number of capacity units (e.g., circuits, time slots, or wavelengths) used on link j. $U_j^{(i)}$ can be considered as a feature x_j in vector $\boldsymbol{X}$.

2) PREDICTING BLOCKING PROBABILITY FOR A SERVICE REQUEST BETWEEN A PAIR OF NODES

Given a network snapshot (which includes the information on the network link capacity usage), when a new service request between a pair of nodes arrives, we use the supervised naïve Bayes classifier to predict its blocking probability. For this, we first define the problem instance (or vector) to be classified as

$$\boldsymbol{X} = [\boldsymbol{S}, sd]^T \tag{4}$$

Here $\boldsymbol{S}$ denotes a network snapshot defined as in (3) and sd is the index of the node pair requesting a service connection; note that the latter is also to be included as a feature of the vector $\boldsymbol{X}$. This new request may be either established or declined depending on the current network capacity utilization. Let the binary variable Y denote the outcome of the classifier when it is given the input $\boldsymbol{X}$. Specifically, $Y = 0$ if the call request between sd is served (i.e., a route is found and the request can be satisfied); otherwise, $Y = 1$ if the call request between sd is blocked.

We define the conditional probability $P(Y = 1 \mid \boldsymbol{X}) = P(Y = 1 \mid \langle \boldsymbol{S}, sd \rangle)$ as the potential blocking probability of a service request between node pair sd based on the current network snapshot $\boldsymbol{S}$. Using Bayes' theorem, this conditional probability can be written as

$$P(Y = 1 \mid \boldsymbol{X}) = \frac{P(Y = 1) \cdot P(\boldsymbol{X} \mid Y = 1)}{P(\boldsymbol{X})} \tag{5}$$

For a circuit-switched network where connection requests arrive dynamically, $P(Y = 1)$ denotes the overall network-wide blocking probability of the service requests that have arrived. Let H be the total number of requests that have arrived and $I\left\{Y^{(i)} = 1\right\}$ is an indicator function, which is 1 if the i^{th} arrived request is blocked, then statistically

$$P(Y = 1) = \frac{\sum_{i=1}^{H} I\left\{Y^{(i)} = 1\right\}}{H} \tag{6}$$

This effectively calculates the ratio of blocked service requests to the total number of requests that have arrived.

We now consider the probability $P(\boldsymbol{X} \mid Y = 1)$ which is the probability of finding the network in state $\boldsymbol{S}$ with a request initiated between node pair sd given that the service request is blocked. Using suitable independence assumptions, this may be expanded as follows.

$$\begin{aligned} P(\boldsymbol{X} \mid Y = 1) &= P(\boldsymbol{S} \mid Y = 1) \cdot P(sd \mid Y = 1) \\ &= \prod_{j=1}^{L} P\left(U_j = U_j^{\boldsymbol{S}} \mid Y = 1\right) \\ &\quad \cdot P(sd \mid Y = 1) \end{aligned} \tag{7}$$

Here we assume that the capacity usage on the links is independent of each other, and that they are also independent of the node pair that initiates the request. It should be noted that for the performance analysis of a circuit-switched network, it is very common, but valid to assume the independence of capacity usage on each link. Representative works with such an assumption are ample in the literature, e.g., [3], [6]. $U_j^{\boldsymbol{S}}$ is the number of capacity units used on link j in the network snapshot $\boldsymbol{S}$, and $P\left(U_j = U_j^{\boldsymbol{S}} \mid Y = 1\right)$ is the probability that the number of capacity units used on link j is equal to $U_j^{\boldsymbol{S}}$ when a service request is blocked.

We can now calculate $P\left(U_j = U_j^{\boldsymbol{S}} \mid Y = 1\right)$ and $P(sd \mid Y = 1)$ based on statistics using the following two equations.

$$P\left(U_j = U_j^{\boldsymbol{S}} \mid Y = 1\right) = \frac{\sum_{i=1}^{H} I\left\{U_j^{(i)} = U_j^{\boldsymbol{S}} \wedge Y^{(i)} = 1\right\}}{\sum_{i=1}^{H} I\left\{Y^{(i)} = 1\right\}} \tag{8}$$

$$P(sd \mid Y = 1) = \frac{\sum_{i=1}^{H} I\left\{sd^{(i)} = sd \wedge Y^{(i)} = 1\right\}}{\sum_{i=1}^{H} I\left\{Y^{(i)} = 1\right\}} \tag{9}$$

In (8), $I\left\{U_j^{(i)} = U_j^{\boldsymbol{S}} \wedge Y^{(i)} = 1\right\}$ is an indicator function which is equal to 1 if the i^{th} service request is blocked and the number of capacity units used in link j is equal to $U_j^{\boldsymbol{S}}$. In (9), $I\left\{sd^{(i)} = sd \wedge Y^{(i)} = 1\right\}$ is an indicator function, which is equal to 1 if the i^{th} service request is initiated by node pair sd and it is blocked. The denominator in both (8) and (9) is the total number of connections out of H which were blocked

We also need to find $P(\boldsymbol{X})$ for use in (5), which is derived as follows.

$$P(\boldsymbol{X}) = P(\boldsymbol{S}, sd) = P(\boldsymbol{S}) \cdot P(sd) = \prod_{j=1}^{L} P\left(U_j = U_j^{\boldsymbol{S}}\right) \cdot P(sd) \tag{10}$$

Here $P\left(U_j = U_j^S\right)$ and $P(sd)$ are calculated as follows.

$$P\left(U_j = U_j^S\right) = \frac{\sum_{i=1}^{H} I\left\{U_j^{(i)} = U_j^S\right\}}{H} \tag{11}$$

$$P(sd) = \frac{\sum_{i=1}^{H} I\left\{sd^{(i)} = sd\right\}}{H} \tag{12}$$

Here $I\left\{U_j^{(i)} = U_j^S\right\}$ is an indicator function which is equal to 1 if the number of capacity units used on link j is equal to U_j^S when the i^{th} service request arrives, and $I\left\{sd^{(i)} = sd\right\}$ is an indicator function which is equal to 1 if the i^{th} request is initiated by node pair sd.

3) NAÏVE BAYES CLASSIFIER-ASSISTED LL ROUTING

Having explained how to use the naïve Bayes classifier to predict the blocking probability of a service request between a pair of nodes, we next describe an enhanced LL routing algorithm assisted by the supervised naïve Bayes classifier.

Assume that there is a new service request arriving between a node pair sd and that, at that moment, we have a network snapshot $\boldsymbol{S}$ that records the capacity usage on each link. We also assume that between node pair sd, multiple candidate routes are available for establishing this service connection. These routes can be pre-calculated offline. The issue then would be *which route should be selected for service connection establishment*. The conventional LL routing algorithm would always choose the one with the least load, even though it may lead to capacity over-consumption under some situations (and consequently, higher blocking for subsequent connection requests). Here the term *capacity over-consumption* is defined as the amount of extra capacity required by a selected LL path compared to the shortest path otherwise chosen. For example, if the shortest path of a service connection has two hops, while its selected LL path has three hops, then the capacity over-consumption in this scenario is one unit if the service connection uses one unit of bandwidth. In the naïve Bayes classifier-assisted LL routing algorithm, *in addition to the traffic load on a route, we also consider how the establishment of a service along this route would affect the successful establishment of future service connections from the perspective of the service blocking probability of the whole network*. Thus, the naïve Bayes classifier-assisted LL routing algorithm aims to achieve the best balance between the two objectives.

To implement the naïve Bayes classifier-assisted LL routing algorithm, some prior route calculations are required. That is, for each node pair sd, we first find offline all possible (or K shortest) routes between all pairs of nodes based on a network topology. The set of candidate routes between node pair sd is denoted as R^{sd}. This route information is static and stored in a network central controller.

Then for an online service request between node pair sd received, we first remove from R^{sd} all the routes that do not have sufficient remaining capacity. Next, considering each route in R^{sd}, we find how the service establishment along this route would affect the establishment of other future service connections in the network. Here for each candidate route $\vec{r_k^{sd}} \in R^{sd}$, let us first assume that we use it to establish a service connection, after which the network snapshot $\boldsymbol{S_c}$ would be updated to

$$\boldsymbol{S_k} = \boldsymbol{S_c} + \vec{r_k^{sd}} \tag{13}$$

Then based on $\boldsymbol{S_k}$, we estimate the potential service blocking probability between any node pair $s'd'$ (after a service connection is established along $\vec{r_k^{sd}}$) as

$$BP_{s'd'}^{sd,k} = P\left(Y = 1 \mid \langle \boldsymbol{S_k}, s'd' \rangle\right) \tag{14}$$

This blocking probability can be calculated by equations (5)-(12).

We can then calculate a network-wide blocking probability (after a service connection is established along $\vec{r_k^{sd}}$) as

$$BP_{net}^{sd,k} = \sum\nolimits_{s'd'} l_{s'd'} \cdot BP_{s'd'}^{sd,k} \tag{15}$$

Here $l_{s'd'}$ is the ratio of traffic load between node pair $s'd'$ to the total traffic in the entire network. The relationship $\sum_{s'd'} l_{s'd'} = 1$ holds and $l_{s'd'}$ is statistically calculated as

$$l_{s'd'} = \frac{\sum_{i=1}^{H} I\left\{s'd'^{(i)} = s'd'\right\}}{H} \tag{16}$$

where $I\left\{s'd'^{(i)} = s'd'\right\}$ is an indicator function to tell if the i^{th} service request is initiated by node pair $s'd'$. Obviously, the route that has the smallest $BP_{net}^{sd,k}$ should have more preference to establish a service connection since its establishment would result in the lowest blocking probability for future service connections.

In addition to $BP_{net}^{sd,k}$, for each route $\vec{r_k^{sd}} \in R^{sd}$, we also estimate their sum load on their traversed links. Specifically, this sum load is calculated as

$$u_k^{sd} = \sum\nolimits_{i \in \vec{r_k^{sd}}} u_{k,i}^{sd} \tag{17}$$

where $u_{k,i}^{sd}$ is the capacity utilization on the i^{th} link of route $\vec{r_k^{sd}}$, defined as

$$u_{k,i}^{sd} = \frac{U_i^c}{W_i} \tag{18}$$

Here W_i is the number of total capacity units on link i and U_i^c is the number of capacity units used on link i in the network snapshot $\boldsymbol{S_c}$. A route with the smallest u_k^{sd} is considered the least congested or having the least load and we would prefer to establishing a service connection along this route to avoid network congestion.

The naïve Bayes classifier-assisted LL routing considers *both the load on each route as well as how the establishment of a service connection along this route would affect the successful establishment of future service connections in the network*. Specifically, we select the route based on the following equation.

$$k_{sd}^* = arg\min_k \left(BP_{net}^{sd,k} \cdot u_k^{sd}\right) \tag{19}$$

which chooses a route with the least load as well as having the lowest impact on the success of future service connection establishment.

It should also be noted that there is no significant change of computational complexity for the LL routing algorithm after it is incorporated with the naïve Bayes classifier since the classifier just provides the weight parameters for the LL algorithm to choose the best route. The parameters are estimated earlier before a new service connection request arrives, which would not become an extra computational burden for the LL algorithm.

4) ONLINE LEARNING PROCESS

The proposed approach can be implemented as the following online learning process. Upon the arrival of a new service request, the process described in the previous sub-section selects a route for establishing this new service based on historical network data (i.e., all the information on snapshots and service provisioning before this service request). Once the current service request is either satisfied or blocked, the network snapshot (when this request arrives) and this request become historical data for the route selection of future service connections. The process continues during the course of network service provisioning, and more historical data is accumulated as more service requests arrive.

To simulate the above online learning process based on the naïve Bayes classifier-assisted LL routing algorithm, we present its pseudocode as follows.

```
Repeat
{
  For a new service request between node pair sd, Z^sd
  {
    Decide its eligible candidate route set R^sd based
    on all the routes found offline for the node pair;
    //Here remove routes without sufficient capacity
    If R^sd = Ø, then the service request is blocked;
    Otherwise
    {
      For each route r_k^sd ∈ R^sd
      {
        Calculate its sum load u_k^sd using (17), (18);
        Calculate the potential network-wide
        blocking probability BP_net^{sd,k} after Z^sd
        is established along r_k^sd, using (13)-(16);
        specifically, in (14), P(Y = 1 | ⟨S_k, s'd'⟩)
        is calculated using (5)-(12);
      }
      Select route k*_sd = arg min_k (BP_net^{sd,k} · u_k^sd) and
      establish the service request along the route;
      Update all the related parameters in (5)-(12)
      accordingly.
    }
  }
}
```

In the above learning process, Laplacian smoothing is used when initially calculating some terms. This is performed as follows.

$$P(Y=1) = \frac{\sum_{i=1}^{H} I\left\{Y^{(i)}=1\right\}+1}{H+2} \tag{20}$$

$$P\left(U_j=U_j^S \mid Y=1\right) = \frac{\sum_{i=1}^{H} I\left\{U_j^{(i)}=U_j^S \wedge Y^{(i)}=1\right\}+1}{\sum_{i=1}^{H} I\left\{Y^{(i)}=1\right\}+W_j+1} \tag{21}$$

$$P(sd \mid Y=1) = \frac{\sum_{i=1}^{H} I\left\{sd^{(i)}=sd \wedge Y^{(i)}=1\right\}+1}{\sum_{i=1}^{H} I\left\{Y^{(i)}=1\right\}+m} \tag{22}$$

$$P\left(U_j=U_j^S\right) = \frac{\sum_{i=1}^{H} I\left\{U_j^{(i)}=U_j^S\right\}+1}{H+W_j+1} \tag{23}$$

$$P(sd) = \frac{\sum_{i=1}^{H} I\left\{sd^{(i)}=sd\right\}+1}{H+m} \tag{24}$$

$$u_{k,i}^{sd} = \frac{U_i^c}{W_i}+\alpha \tag{25}$$

where W_j is the total number of capacity units on link j, m is the total number of node pairs in the network, and α is a small value set as 10^{-6}.

It may also be noted that during the course of service provisioning and learning, all the system parameters in (5)-(12) are updated for each arriving service request, regardless of whether it is served or blocked.

III. IMPLEMENTATION OF ML-ASSISTED NETWORK CONTROL SYSTEM

In Section II, we described the mathematical fundamentals of applying the naïve Bayes classifier to assist the LL routing algorithm. In this section, we introduce how this ML-assisted routing algorithm can be implemented for handling real-time service requests in an operating network in the context of a network control plane as shown in Fig. 3.

We assume a centralized network control system with a single network controller. This assumption is reasonable since today's Software Defined Network (SDN) control system is implemented in just such a way. In addition to providing a user interface for service establishment, the central controller monitors and records the state information of an entire network. When provisioning a service connection, the central controller consults its ML-processor to provide the information on whether the request should be blocked or accepted and, if the service request is accepted, then to decide the best route to be used for establishing a service connection.

The ML-processor first carries out initial learning for an actual network based on a *forecasted traffic load matrix*, which provides an approximate prediction of traffic load between each node pair. After this initial learning, the ML-processor can obtain an initial set of learning parameters for (5)-(18). Subsequently, the control plane starts to accept real network service requests. It consults the ML-processor on whether a service connection can be provisioned, and if so, it will provide a route for service establishment. Thus, *the service provisioning process itself is a*

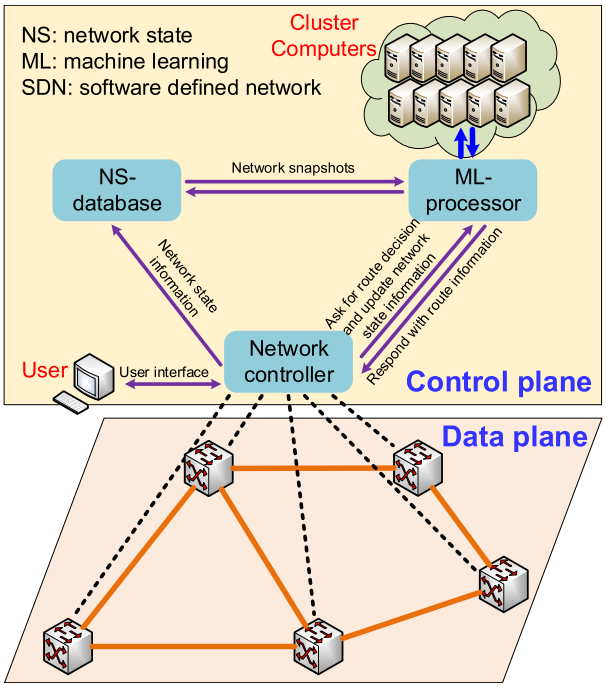


FIGURE 3. A network control system embedded with the ML-assisted routing algorithm.

learning process. In addition to provisioning services, the ML-processor performs online learning based on the information of the current service request and the network snapshot when the request arrives. The parameters in (5)-(18) will also be updated accordingly when online services are provisioned.

The ML-processor feeds back the route information to the network central controller and the latter triggers signaling to establish a service connection. Meanwhile, it also sends the information on the current network snapshot and service connection to a network state database (i.e., NS-database) for storing the network historical state information. Note that this database is only used to store the historical network state information, not for online ML-processor learning. The ML-processor carries out its learning process in an iterative way. Specifically, the arrival of each service request would trigger a new learning iteration to update the parameters in (5)-(18). It should be noted that the learning process only updates the parameters in (5)-(18) based on a single network snapshot when the service arrives, rather than starting from zero to carry out learning based on all the snapshot information stored in the NS-database.

For the ML-processor, in addition to the initial offline learning based on a forecast traffic load matrix, other offline learning may also be implemented for some situations. For example, if we can forecast that there will be a clear change of traffic load between a pair of nodes in the near future, then we can add that to the learning process to tune the related parameters in (5)-(18). This is carried out by jointly considering the simulation data and the data of real network state information stored in the NS-database. Similarly, if there is upgradation of network link capacity, we can implement a similar re-learning process to update the related parameters.

Finally, in addition to service arrival, we also need to consider the scenario of service release after a service completes its mission. When this happens, the central controller instructs the data plane to release the related network resources and informs the ML-processor to update its current network state information. However, in this process, the related parameters in (5)-(18) are not updated and no network snapshot is forwarded to the NS-database for storage.

IV. SIMULATIONS AND PERFORMANCE ANALYSES

A. TEST CONDITIONS

We assume that new service connection requests arrive following a Poisson arrival process with an arrival rate of λ per second. Each service connection request has a mean holding time of $1/\mu$ seconds following an exponential distribution. We normalize our time measurement using $1/\mu = 1$ so that the traffic load between each pair of nodes may be considered to be λ erlang. For all the study schemes, we used the same random seed to generate the random service request arrivals. We consider two test networks: (1) the 14-node, 21-link NSFNET network and (2) the 21-node, 25-link ARPA-2 network as shown in Fig. 4. In the NSFNET network, the number of capacity units in each link is random and is uniformly distributed within the range of [5, 27]. The traffic load between each node pair is generated randomly with a uniform distribution within the range of $[0.45, 0.45+X]$ erlang where $X \in \{0.15, 0.3, \ldots, 1.05\}$. In the ARPA-2 network, the number of capacity units in each link is similarly randomly distributed within a range of [5, 31] and the traffic load between each node pair is generated within a range of $[0.07, 0.07+X]$ erlang where $X \in \{0.07, 0.14, \ldots, 0.49\}$.

For performance comparison, we have also run simulations based on the adaptive shortest path routing algorithm and the

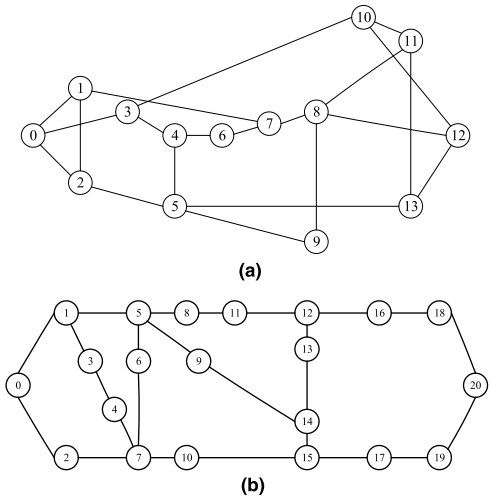


FIGURE 4. Test networks. (a) 14-node, 21-link NSFNET network. (b) 21-node, 25-link ARPA-2 network.

conventional LL routing algorithm. The shortest path routing algorithm always finds an eligible route with the smallest number of hops based on the current network capacity usage. This is adaptive shortest path routing, not a fixed one. The conventional LL routing algorithm uses the link capacity utilization as the ''cost'' of each link and then finds a feasible route with the smallest u_k^{sd} in (17) by using Dijkstra's algorithm. Both the algorithms try to find their own best eligible routes between a pair of nodes. If no such routes can be found, the service request would be blocked and discarded. Each blocking probability point of the shortest path routing and the conventional LL algorithms is evaluated based on 10^6 service request arrival events.

B. INCORPORATE THE PROPOSED LEARNING APPROACH IN A PARALLEL COMPUTING SYSTEM

We have designed a framework to incorporate the proposed approach in a parallel computing system as shown in Fig. 5 so as to fast learn a large number of service arrivals and network snapshots. The system is made up of a cluster controller and cluster computers (or computing clients). The cluster controller is the central brain of the whole parallel learning system, which is responsible for distributing the learning tasks to the computing clients and gathering the results fed back by the latter. The computing clients work in parallel and feedback the results of their learning to the cluster controller. We have constructed a prototype of this parallel learning system in our laboratory as shown at the two bottom corners of Fig. 5. The system contains 10 mini-computers, one Ethernet switch, one power supply module, and three fans. Each mini-computer has a 4.0-GHz quad-core CPU and 8-GB memory [38].

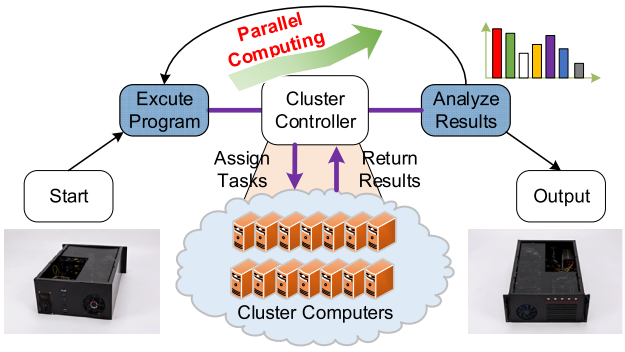


FIGURE 5. Parallel learning system.

Fig. 6 shows how the proposed naïve Bayes classifier-assisted LL routing algorithm is incorporated in a parallel learning system. For each round of learning, the cluster controller splits the learning data into multiple sub-data sets. For example, consider a learning round with 100 million service arrival events. We first generate 10 learning sub-tasks with each learning one million arrival events. The cluster controller distributes each one-million-event sub-data set to a particular cluster computer for parallel learning. We assume that there are ten such cluster computers. Upon receiving

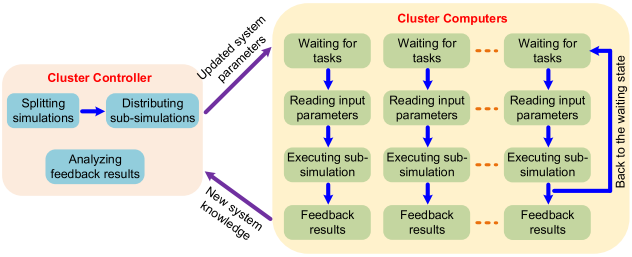


FIGURE 6. Procedure of parallel learning.

the learning sub-tasks, each cluster computer starts to learn independently and in parallel. In addition to counting the number of blocked events, they also learn network snapshots and update the related parameters in (5)-(18). Once a cluster computer completes its one-million-event learning sub-task, it feeds back the number of blocked events as well as the parameters that are learned to the cluster controller.

Once the cluster controller receives this information, it updates the total number of blocked events of the entire system and the related parameters in (5)-(18) by integrating all the feedback information from all the cluster computers. For example, $\sum_{i=1}^{H} I\left\{U_j^{(i)} = U_j^{\boldsymbol{S}} \wedge Y^{(i)} = 1\right\}$ counts the total number of situations when the i^{th} service request is blocked and the number of capacity units used on link j is just equal to $U_j^{\boldsymbol{S}}$ in an H-arrival-event sub-task. For each cluster computer, after learning one million arrival events, we can find its corresponding number $s_n^k = \sum_{i=1}^{H=10^6} I\left\{U_j^{(i)} = U_j^{\boldsymbol{S}} \wedge Y^{(i)} = 1\right\}$, where n is the index of the cluster computer in the parallel system and k is the round index of learning that the cluster computer executes. Thus, for each round of learning, with the cluster computers executing the parallel tasks, we have a corresponding sum parameter at the k^{th} round as $s^k = \sum_{n \in \boldsymbol{Cluster}} s_n^k$, where ***Cluster*** denotes the set of cluster computers. In a similar way, we can update other related parameters for the learning process.

In order to learn the whole routing system as much as possible, we need to have multiple rounds of learning. Continuing with the previous example, as the total number of simulated arrival events is 100 million and each round of learning performed by the parallel system can simulate 10 million arrival events, we need to run 10 rounds of parallel learning. Each time when one round of learning is completed, the cluster controller initiates another round of learning. When doing this, the cluster controller forwards the parameters in (5)-(18) learned based on the previous round to each of the cluster computers. The latter use these updated parameters to carry out learning for another round of arrival events and meanwhile collect the information for updating the related parameters. The entire learning process terminates when the total number of planned learning events are reached. Then the cluster controller finds the final service connection blocking probability and the final updated parameters in (5)-(18).

For the naïve Bayes classifier-assisted LL routing algorithm, we employ the above parallel learning system to

carry out dynamic-event driven learning with 10^8 call arrival requests. Between each node pair, all the eligible routes are considered for the route selection in the algorithm based on the current network capacity utilization status. As mentioned before, these routes are found offline and stored in a central controller. We employed the parallel learning system containing ten min-computers as shown in Fig. 5 to carry out the learning process, in which one computer functions as a cluster controller as well as a computing client and all the other computers function as computing clients. Ten rounds of learning are executed with each round learning 10 million service arrival events and one client learning one million service arrival events. The overall blocking probability of the naïve Bayes classifier-assisted LL routing algorithm is then calculated based on the total 10^8 call arrival requests.

C. SERVICE CONNECTION BLOCKING PERFORMANCE

We first compare the service connection blocking probability for the different routing algorithms. Fig. 7 shows the blocking probabilities of the NSFNET and ARPA-2 networks with an increasing interval of random traffic load per node pair X. In the legend, "LLA" corresponds to the conventional LL routing algorithm, "SP" corresponds to the adaptive shortest path routing algorithm, and "ML-NB-LL" corresponds to the naïve Bayes classifier-assisted LL routing algorithm.

Fig. 7(a) shows the results of the SP and conventional LL algorithm for the NSFNET network with 95% confidence intervals based on Student-t distribution. For the naïve Bayes classifier-assisted LL routing algorithm, we do not show its confidence interval because of the extremely large number of network snapshots learned (i.e., up to 10^8 arriving requests). We can see that the naïve Bayes classifier-assisted LL routing algorithm achieves the lowest blocking probability among all the three routing algorithms, the conventional LL routing algorithm ranks second in terms of blocking probability, and the shortest path routing algorithm performs the worst. It is reasonable to observe that the conventional LL algorithm outperforms the SP algorithm since the former aims for balancing the load in the whole network by always choosing a route with the least congestion. However, our proposed naïve Bayes classifier-assisted LL routing algorithm outperforms the conventional LL algorithm because in addition to the traffic load on each route, it also considers the future potential network-wide blocking probability predicted based on the historic service provisioning data when a service connection is established on a specific route. By considering this data, the LL algorithm chooses a lightly loaded route that also has the least adverse impact on service connection establishments in the future, if a service connection is indeed established on this route. A similar observation can be made for the ARPA-2 network (see Fig. 7(b)) where the performance ranking of the three algorithms in terms of blocking probability is the same.

Currently, there is a general consensus that the LL or least congested routing algorithm is the method of choice for good routing performance and therefore it can be used as a benchmark for routing methods aiming at minimizing blocking probability. However, as shown by our earlier example of Fig. 1, the LL routing algorithm may suffer from resource over-consumption in certain situations, which would adversely affect its efficiency. In this study, with the assistance of a machine learning technique, i.e., the naïve Bayes classifier, we have demonstrated that the blocking performance of the LL algorithm can be improved further by learning from the historical network service provisioning data (i.e., network snapshots).

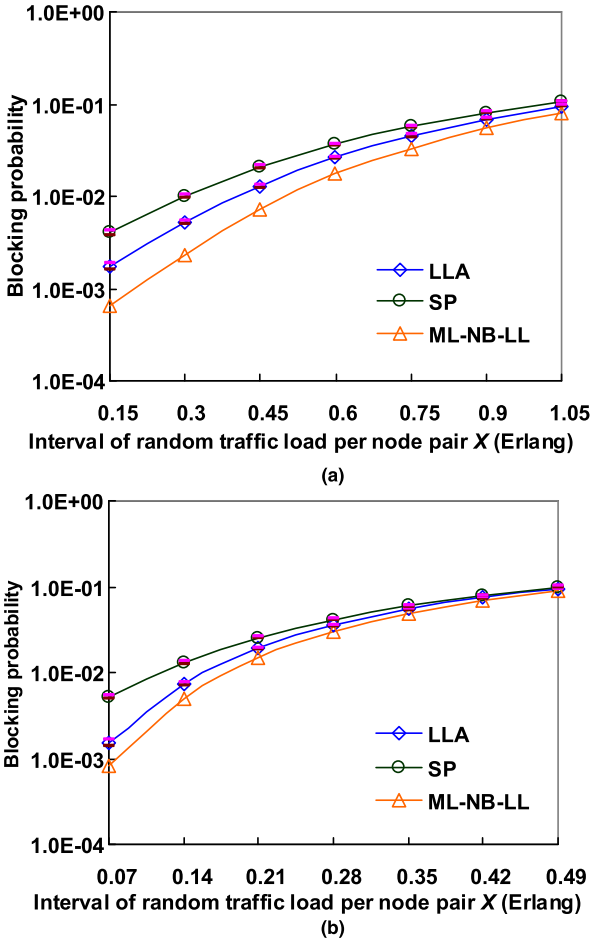


FIGURE 7. Blocking performance comparison between the three routing algorithms. (a) NSFNET. (b) ARPA-2.

D. IMPACT OF NUMBER OF SNAPSHOTS LEARNED

Learning network snapshots is an important step to ensure good performance for the naïve Bayes classifier-assisted LL routing algorithm. Thus, we also evaluated how the number of network snapshots learned (which is equal to the number of arrived service requests) can impact the service connection blocking performance. The results are shown in Fig. 8, where the legend "Single" corresponds to the case of a single computer for the simulation, the legend "Para" corresponds to the case of parallel computing, and the legend "Time"

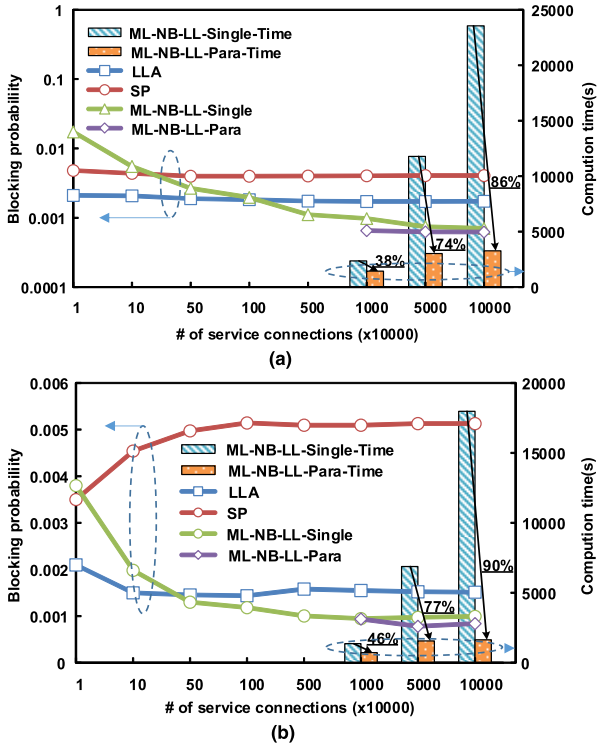


FIGURE 8. Blocking performance and computing time comparison between the different routing algorithms under different numbers of network snapshots learned. (a) NSFNET. (b) ARPA-2.

corresponds to the time required for simulating to learn a certain number of arrival events.

In the NSFNET network, the traffic load between each node pair is randomly generated within the range of [0.45, 0.6] erlang, and the total number of capacity units on each link is randomly distributed within the range of [5, 27]. For the SP and conventional LL algorithms, as there is no learning process, their blocking performances remain almost unchanged after one million service arrival events. Nonetheless, for the naïve Bayes classifier-assisted LL routing algorithm, we see that with an increasing number of service connection requests, i.e., the number of network snapshots learned, its blocking performance at first improves and then stays almost the same when 50 million arrival events are learned. Moreover, the final blocking performance of the naïve Bayes classifier-assisted LL routing algorithm is always better than that of the other two routing algorithms.

In addition, specifically for the naïve Bayes classifier-assisted LL routing algorithm, because there are up to 100 million arrival events to learn, which is very time-consuming for a single computer, the parallel learning system has also been employed for the study. In Fig. 8(a), we also compare the times taken by a single computer and the parallel computing system. We see that the learning time can be significantly shortened by the parallel learning system. For example, for learning 100 million arrival events, the computation time can be reduced by over 85% with the parallel learning system, from almost 7 hours to only about an hour. This indicates the computational efficiency of the parallel learning system for the machine learning process. Note that, in the future, we may further increase the number of computers in the parallel system for an even shorter computation time since this system can be easily expanded.

We have also conducted similar studies for the ARPA-2 network, in which the traffic load between each node pair is randomly generated within the range of [0.07, 0.14] erlang, and the total number of capacity units in each link is randomly distributed within the range of [5, 31]. As shown in Fig. 8(b), we observe that the blocking performance and the computation times for the different schemes are similar to those of the NSFNET network. Here, for the naïve Bayes classifier-assisted LL routing algorithm, the blocking performance also almost does not change when 10 million arrival events are learned and the naïve Bayes classifier-assisted LL routing algorithm can always achieve better performance than the other two routing algorithms. We see that parallel computing technology can help to reduce up to 90% computation time for the machine learning process.

E. HOW DOES NAÏVE BAYES CLASSIFIER HELP TO AVOID NETWORK CAPACITY OVER-CONSUMPTION FOE LL ALGORITHM

The improved performance of the naïve Bayes classifier-assisted LL routing algorithm is attributed to its learning capability from the historical network service provisioning data (i.e., network snapshots). As a result, this effectively avoids the network capacity over-consumption that the conventional LL algorithm may suffer from in some situations. In the following, we demonstrate by our analysis and simulations how this has been achieved.

In Fig. 9, we show the extra hop count for each established service connection compared to their shortest paths. For example, suppose that the shortest path between the node pair of a service connection (based on the physical topology) has K hops. However, considering the network resource usage status, the final selected route by the LL routing algorithm is not the shortest one, but a longer one. The number of hops Δ by which this is larger than K is defined as the extra hop count for the route compared to the shortest path. The hop count of this selected route is $K+\Delta$. Thus, in Fig. 9, $\Delta = 0$ means the case of the shortest path and $\Delta > 0$ means that the selected route is longer than the shortest path. We compare the hop count distributions of established service connections for the shortest path routing algorithm, the conventional LL routing algorithm, and the naïve Bayes classifier-assisted LL routing algorithm. For the former two, only 10^6 arriving requests are simulated as its blocking probability stabilizes after this number of arriving requests occur, as shown in Fig. 9. For the last one, we simulated 10^8 arrival requests and collected the above distribution data only for the last 10^6 arrival requests.

According to the results in Fig. 9, most of the service connections are provisioned on their shortest routes with $\Delta = 0$. However, large percentages of the service connections are also provisioned on the second and third short-

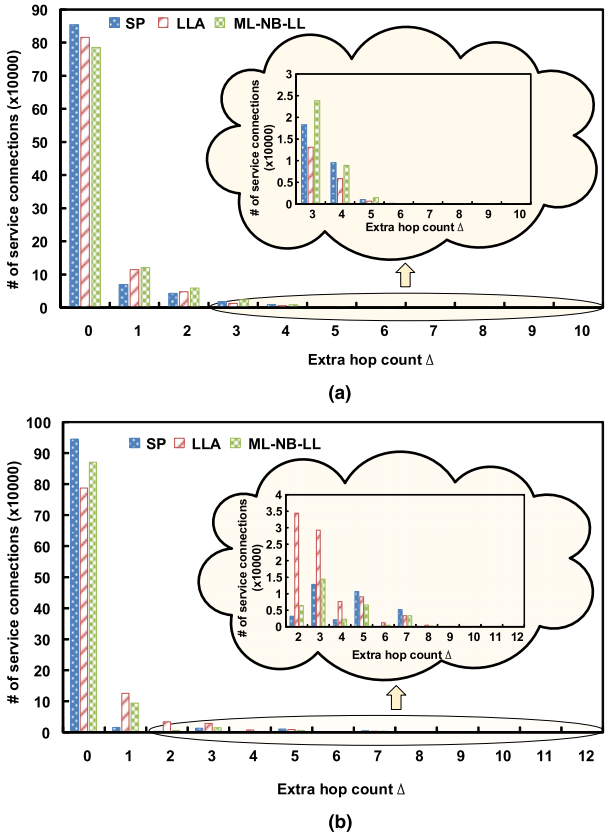


FIGURE 9. Extra hop count distributions of the different routing algorithms. (a) NSFNET (average numbers of extra hops = 0.2576 (SP), 0.2781 (LLA), 0.2194 (ML-NB-LL)). (b) ARPA-2 (average numbers of extra hops = 0.1602 (SP), 0.3942 (LLA), 0.3575 (ML-NB-LL)).

est routes. In some cases, very long routes with $\Delta = 10$ could be used under the LL routing algorithms since they only consider the least congested routes while ignoring the actual hop counts of the selected routes. This occurs even though the LL algorithms also by default partially consider the hop count, which is weighted by the link capacity utilization. We also calculated the average number of extra hops for the provisioned service connections for the other three routing algorithms (see the sub-captions of Fig. 9). It can be found that in the NSFNET network, the naïve Bayes classifier-assisted LL routing algorithm shows the smallest number of extra hops compared to the other two routing algorithms. This implies that the naïve Bayes classifier-assisted LL routing algorithm has the lowest network capacity over-consumption when implementing the LL routing algorithm, thereby achieving a lower overall blocking probability. The same comparison can be made for the ARPA-2 network, where similar phenomena can be observed. Most of the service connections are served by the first, second, and third shortest routes between node pairs. The shortest path routing algorithm has the smallest number of extra hops but for the two LL routing algorithms, the naïve Bayes classifier-assisted LL routing algorithm shows a smaller number of extra hops. This implies that it wastes less network capacity when choosing the LL routes than the conventional LL routing algorithm. Based on these results, we can conclude that naïve Bayes classifier-assisted LL routing algorithm manages to control well the extra hops needed by the routes to avoid over-consumption of network capacity when choosing the LL routes. Consequently, it significantly reduces the service connection blocking probability as shown earlier in Fig. 7.

F. CAN WE IMPROVE BLOCKING PERFORMANCE OF CONVENTIONAL LL ROUTING ALGORITHM BY SIMPLY UNIFORMLY CONTROLLING EXTRA HOP COUNTS?

We see that the ML-assisted LL routing algorithm can improve the blocking performance over the conventional LL routing algorithm by *differently* controlling the extra hop count to the shortest path for each established service connection. Then a further question for us is: *Can we also improve the blocking performance by simply uniformly setting a limit on the extra hop count to the shortest path for the conventional LL routing algorithm when establishing a service connection*? We did this by extending the conventional LL routing algorithm to incorporate this extra hop count limit in the algorithm when searching for a least loaded route in a network. We show the related results of blocking performance with the various extra hop counts as shown in Fig. 10. Here we consider the NSFNET network with the capacity units on each link distributed within the range of [5, 27], and with the traffic load per node pair distributed within the range of [0.45, 0.45 + X] erlang, where X is the interval of traffic load between different node pairs. From the result, we can see that with an increasing extra hop count to the shortest path, the lightpath blocking performance improves at the beginning but, however, gets saturated after a certain threshold for the extra hop count. More specifically, when the extra hop count grows to 4, a further increase of this parameter would not bring the improvement of blocking performance, till it becomes a full version of the conventional LL routing algorithm when the extra hop count becomes infinite. This observation implies that this strategy does not help to improve the blocking performance for the conventional LL routing algorithm by simply controlling the extra hop count *uniformly* for all the node pairs. Instead, this kind of control is more

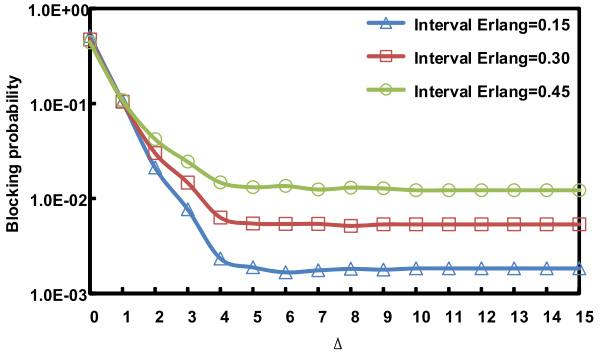


FIGURE 10. Blocking probability versus hop count limit for an LL route (NSFNET). Δ: the difference between hop count of an LL route and the count of the shortest route between each node pair.

effective when it is done *differently* for different node pairs according to the current network status, as in the naïve Bayes classifier-assisted LL routing algorithm. This verifies our claims on the unique capability and benefit of the naïve Bayes classifier-assisted LL routing algorithm.

V. CONCLUSION

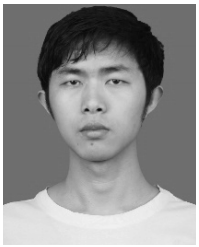
We have considered the blocking probability benchmark established by the conventional LL routing algorithm. We have demonstrated that this benchmark can be further improved because LL routing may over-consume network capacity under certain circumstances when selecting an LL route to provision a service connection. In the past, it has been difficult to find an effective way to avoid this type of inefficiency. In this paper, we employed a machine learning technique to develop a naïve Bayes classifier-assisted LL routing algorithm to effectively overcome the above capacity inefficiency. This allows us to break through the blocking performance benchmark of the conventional LL algorithm. We have considered network snapshots as the historical data for machine learning. This has been applied to a supervised naïve Bayes classifier to predict the potential blocking probabilities of future call requests in the whole network if a service request is established along a certain route. With this information, a route that has a low load as well as a low adverse impact on the success of future service connection establishment is selected. To implement the learning process based on a large number (hundreds of millions of) of network snapshots, we have designed a parallel learning system built in our laboratory to implement parallel learning and performance evaluation. Moreover, the implementation of this ML-assisted network control system was also addressed.

Simulation studies have demonstrated that the proposed naïve Bayes classifier-assisted LL routing algorithm indeed achieves significant improvement over the benchmark performance set by the conventional LL routing algorithm. Moreover, we have also investigated the phenomenon of over-consumption of network capacity by the conventional LL routing algorithm. We demonstrated that the conventional LL routing algorithm has a much larger number of extra hops for service connections provisioning than that of naïve Bayes classifier-assisted LL routing algorithm. Thus, the naïve Bayes classifier-assisted LL routing algorithm is effective in controlling well the extra hops needed for service connections when these are provisioned based on the LL routes. For the naïve Bayes classifier-assisted LL routing algorithm, we also observe that its blocking performance is closely related to the number of network snapshots learned. An increasing number of network snapshots can lead to a lower blocking probability through more effective learning. Finally, we find that the parallel learning system that we employed specifically for learning the required large numbers of network snapshots is computationally efficient. Up to 90% of the time can be saved for the learning process, which significantly accelerated our simulation studies. We hope that this work would revive the research interest in the routing technique for the circuit-switched network by adopting the routing methodology proposed in this paper.

REFERENCES

[1] F. P. Kelly, "Blocking probabilities in large circuit-switched networks," *Adv. Appl. Probab.*, vol. 18, no. 2, pp. 473–505, Jun. 1986.
[2] H. G. Badr and S. Podar, "An optimal shortest-path routing policy for network computers with regular mesh-connected topologies," *IEEE Trans. Comput.*, vol. 38, no. 10, pp. 1362–1371, Oct. 1989.
[3] A. Girard, *Routing and Dimensioning in Circuit-Switched Networks*. Boston, MA, USA: Addison-Wesley, 1990.
[4] Z. Wang and J. Crowcroft, "Analysis of shortest-path routing algorithms in a dynamic network environment," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 22, no. 2, pp. 63–71, Apr. 1992.
[5] R. Ramaswami and K. N. Sivarajan, "Routing and wavelength assignment in all-optical networks," *IEEE/ACM Trans. Netw.*, vol. 3, no. 5, pp. 489–500, Oct. 1995.
[6] A. Birman, "Computing approximate blocking probabilities for a class of all-optical networks," *IEEE J. Sel. Areas Commun.*, vol. 14, no. 5, pp. 852–857, Jun. 1996.
[7] F. P. Kelly, "Routing in circuit-switched networks: Optimization, shadow prices and decentralization," *Adv. Appl. Probab.*, vol. 20, no. 1, pp. 112–144, Mar. 1988.
[8] G. R. Ash, P. Chemouil, A. N. Kashper, S. S. Katz, K. Yamazaki, and Y. Watanabe, "Robust design and planning of a worldwide intelligent network," *IEEE J. Sel. Areas Commun.*, vol. 7, no. 8, pp. 1219–1230, Oct. 1989.
[9] D. Mitra and J. Seery, "Comparative evaluations of randomized and dynamic routing strategies for circuit-switched networks," *IEEE Trans. Commun.*, vol. 39, no. 1, pp. 102–116, Jan. 1991.
[10] R. Ramamurthy and B. Mukherjee, "Fixed-alternate routing and wavelength conversion in wavelength-routed optical networks," *IEEE/ACM Trans. Netw.*, vol. 10, no. 3, pp. 351–367, Jun. 2002.
[11] G. R. Ash, R. H. Cardwell, and R. P. Murray, "Design and optimization of networks with dynamic routing," *Bell Syst. Tech. J.*, vol. 60, no. 8, pp. 1787–1820, Oct. 1981.
[12] G. Bel, P. Chemouil, J. M. Garcia, F. Le Gall, and J. Bernussou, "Adaptive traffic routing in telephone networks," *Large Scale Syst.*, vol. 8, no. 3, pp. 267–282, 1985.
[13] P. Chemouil, J. Filipiak, and P. Gauthier, "Analysis and control of traffic routing in circuit-switched networks," *Comput. Netw. ISDN Syst.*, vol. 11, no. 3, pp. 203–217, Mar. 1986.
[14] B. Hurley, C. Seidl, and W. Sewell, "A survey of dynamic routing methods for circuit-switched traffic," *IEEE Commun. Mag.*, vol. 25, no. 9, pp. 13–21, Sep. 1987.
[15] P. Chemouil, J. Filipiak, and P. Gauthier, "Performance issues in the design of dynamically controlled circuit-switched networks," *IEEE Commun. Mag.*, vol. 28, no. 10, pp. 90–95, Oct. 1990.
[16] A. Mokhtar and M. Azizoglu, "Adaptive wavelength routing in all-optical networks," *IEEE/ACM Trans. Netw.*, vol. 6, no. 2, pp. 197–206, Apr. 1998.
[17] L. Li and A. K. Somani, "Dynamic wavelength routing using congestion and neighborhood information," *IEEE/ACM Trans. Netw.*, vol. 7, no. 5, pp. 779–786, Oct. 1999.
[18] K.-M. Chan and T. P. Yum, "Analysis of least congested path routing in WDM lightwave networks," in *Proc. IEEE INFOCOM*, Toronto, ON, Canada, Jun. 1994, pp. 962–969.
[19] E. W. M. Wong and T.-S. Yum, "Maximum free circuit routing in circuit-switched networks," in *Proc. IEEE INFOCOM*, San Francisco, CA, USA, Jun. 1990, pp. 934–937.
[20] G. Shen, S. K. Bose, T. H. Cheng, C. Lu, and T. Y. Chai, "Efficient heuristic algorithms for light-path routing and wavelength assignment in WDM networks under dynamically varying loads," *Comput. Commun.*, vol. 24, nos. 3–4, pp. 364–373, Feb. 2001.
[21] E. W. M. Wong, A. K. M. Chan, and T. S. P. Yum, "Analysis of rerouting in circuit-switched networks," *IEEE/ACM Trans. Netw.*, vol. 8, no. 3, pp. 419–427, Jun. 2000.
[22] C. E. Shannon, "XXII. Programming a computer for playing chess," *London, Edinburgh, Dublin Philos. Mag. J. Sci.*, vol. 41, no. 314, pp. 256–275, Mar. 1950.
[23] Y. Huang, P. B. Cho, P. Samadi, and K. Bergman, "Dynamic power pre-adjustments with machine learning that mitigate EDFA excursions during defragmentation," in *Proc. Opt. Fiber Commun. Conf. Exhib.*, Los Angeles, CA, USA, Mar. 2017, pp. 1–3.

[24] T. Ohba, S. Arakawa, and M. Murata, "A Bayesian-based approach for virtual network reconfiguration in elastic optical path networks," in *Proc. Opt. Fiber Commun. Conf. Exhib.*, Los Angeles, CA, USA, Mar. 2017, pp. 1–3.
[25] L. Barletta, A. Giusti, C. Rottondi, and M. Tornatore, "QoT estimation for unestablished lighpaths using machine learning," in *Proc. Opt. Fiber Commun. Conf. Exhib.*, Los Angeles, CA, USA, Mar. 2017, pp. 1–3.
[26] F. Morales *et al.*, "Experimental assessment of a flow controller for dynamic metro-core predictive traffic models estimation," in *Proc. 43th Eur. Conf. Opt. Commun.*, Sep. 2017, pp. 1–3.
[27] X. Chen *et al.*, "Leveraging deep learning to achieve knowledge-based autonomous service provisioning in broker-based multi-domain SD-EONs with proactive and intelligent predictions of multi-domain traffic," in *Proc. 43th Eur. Conf. Opt. Commun.*, Sep. 2017, pp. 1–3.
[28] P. Samadi, D. Amar, C. Lepers, M. Lourdiane, and K. Bergman, "Quality of transmission prediction with machine learning for dynamic operation of optical WDM networks," in *Proc. 43th Eur. Conf. Opt. Commun.*, Sep. 2017, pp. 1–3.
[29] F. Meng *et al.*, "Robust self-learning physical layer abstraction utilizing optical performance monitoring and Markov Chain Monte Carlo," in *Proc. 43th Eur. Conf. Opt. Commun.*, Sep. 2017, pp. 1–3.
[30] A. Forster, "Machine learning techniques applied to wireless ad-hoc networks: Guide and survey," in *Proc. 3rd Int. Conf. Intell. Sensors, Sensor Netw. Inf.*, Dec. 2007, pp. 365–370.
[31] B. Russell, M. L. Littman, and W. Trappe, "Integrating machine learning in ad hoc routing: A wireless adaptive routing protocol," *Int. J. Commun. Syst.*, vol. 24, no. 7, pp. 950–966, Jul. 2011.
[32] M. Lee, D. Marconett, X. Ye, and S. J. B. Yoo, "Cognitive network management with reinforcement learning for wireless mesh networks," in *Proc. Int. Workshop IP Oper. Manage.*, 2007, pp. 168–179.
[33] J. A. Boyan and M. L. Littman, "Packet routing in dynamically changing networks: A reinforcement learning approach," in *Proc. Adv. Neural Inf. Process. Syst.*, Denver, CO, USA, vol. 6, Nov. 1994, pp. 671–678.
[34] S. P. Choi and D. Y. Yeung, "Predictive Q-routing: A memory-based reinforcement learning approach to adaptive traffic control," in *Proc. Adv. Neural Inf. Process. Syst.*, Denver, CO, USA, Nov. 1996, vol. 8, pp. 945–951.
[35] Y. Li, L. Peng, and G. Shen, "Load-balanced fixed routing for wavelength routed optical networks," *IEEE Commun. Lett.*, vol. 17, no. 6, pp. 1256–1259, Jun. 2013.
[36] H. C. Leung, C. S. Leung, E. W. M. Wong, and S. Li, "Extreme learning machine for estimating blocking probability of bufferless OBS/OPS networks," *IEEE/OSA J. Opt. Commun. Netw.*, vol. 9, no. 8, pp. 682–692, Aug. 2017.
[37] K. P. Murphy. (Oct. 2006). *Naive Bayes Classifiers*. Accessed: Dec. 19, 2017. [Online]. Available: http://www.cs.ubc.cal-murphyk/TeachinglCS340-Fall06/readingiNB.pdf
[38] *Qyclouds*. Accessed: Dec. 19, 2017. [Online]. Available: http://www.qyclouds.com



LONGFEI LI received the master's degree from Soochow University, China, in 2013, where he is currently pursuing the Ph.D. degree with the School of Electronic and Information Engineering. His research interests include optical network design, cloud computing, and machine learning and optimization.

YA ZHANG is currently pursuing the master's degree with Soochow University, China. His current research interest includes optical networks.

WEI CHEN is currently the General Manager and the CTO with Jiangsu Hengtong Fiber Science and Technology Corporation, China. His research interest includes fiber-optic communications.

SANJAY K. BOSE received the B.Tech. degree from IIT Kanpur, Kanpur, in 1976, and the M.S. and Ph.D. degrees from the State University of New York at Stony Brook, Stony Brook, in 1977 and 1980, respectively. From 1980 to 1982, he was with the Corporate Research and Development Center, General Electric Company, Schenectady, NY, USA, with a focus on projects associated with power line communications, optical fiber communications, and mobile-satellite communications. He subsequently joined the faculty of the Department of Electrical Engineering, IIT Kanpur, from 1982 to 2003. From 2003 to 2008, he was on the faculty of the School of EEE, Nanyang Technological University, Singapore. He returned to India, in 2009, and joined the faculty of the Department of EEE, IIT Guwahati, where he was also the Dean, Alumni Affairs, and External Relations, from 2011 to 2014. He has also held short-term and long-term visiting appointments at the University of Adelaide, the Queensland University of Technology, Nanyang Technological University, and the University of Pretoria.

MOSHE ZUKERMAN (M'87–SM'91–F'07) received the B.Sc. degree in industrial engineering and management and the M.Sc. degree in operation research from the Technion-Israel Institute of Technology, and the Ph.D. degree in engineering from the University of California Los Angeles, in 1985. From 1986 to 1997, he was with Telstra Research Laboratories. From 1997 to 2008, he was with The University of Melbourne. In 2008, he joined the City University of Hong Kong, where he is a Chair Professor of information engineering. He has over 350 publications in scientific journals and conference proceedings. He holds several national and international patents. He has served as a member and the Chair of the IEEE Koji Kobayashi Computers and Communications Award Committee. He has served on the editorial boards of various journals, such as the IEEE JSAC, the IEEE/ACM TRANSACTIONS ON NETWORKING, the *IEEE Communications Magazine*, *Computer Networks*, and *Computer Communications* and in numerous conference committees.

GANGXIANG SHEN received the B.Eng. degree from Zhejiang University, China, the M.Sc. degree from Nanyang Technological University, Singapore, and the Ph.D. degree from the University of Alberta, Canada, in 2006. He is a Distinguished Professor with the School of Electronic and Information Engineering, Soochow University, China. Before joining Soochow University, he was a Lead Engineer with Ciena, Linthicum, MD, USA. He was also an Australian ARC Postdoctoral Fellow with the University of Melbourne. He has authored and co-authored more than 150 peer-reviewed technical papers. His research interests include integrated optical and wireless networks, spectrum efficient optical networks, and green optical networks. He was a recipient of the Young Researcher New Star Scientist Award in the 2010 Scopus Young Researcher Award Scheme in China. He was also a recipient of the Izaak Walton Killam Memorial Award from the University of Alberta and the Canadian NSERC Industrial R&D Fellowship. From 2014 to 2017, he was a Highly Cited Chinese Scholar selected by Elsevier. He is currently an IEEE ComSoc Distinguished Lecturer, from 2018 to 2019, and a member of the IEEE ComSoc Strategic Planning Standing Committee. He is a Lead Guest Editor of the IEEE JSAC Special Issue on Next-Generation Spectrum-Efficient and Elastic Optical Transport Networks and a Guest Editor of the IEEE JSAC Special Issue on Energy-Efficiency in Optical Networks. He is an Associate Editor of the IEEE/OSA JOCN, and an Editorial Board Member of *Optical Switching and Networking* and *Photonic Network Communications*.