GCN-Assisted SnF Scheduling Method for Inter-Datacenter Bulk Transfers

Xiao Lin,*[1] Lanfang Zheng,[1] Yajie Li,[2] and Keqin Shi,[3]

[1]*College of Physics and Information Engineering, Fuzhou University, Fuzhou, China*
[2]*Beijing University of Posts and Telecommunications, Beijing , China*
[3]*Shanghai Jiao Tong University, Shanghai, China*

[*]*linxiaocer@fzu.edu.cn*

Abstract: A GCN-assisted scheduling method is presented to schedule bulk transfers across the inter-DCN in a store-and-forward manner. Simulations demonstrate that the proposed method obtains better network performance and lower complexity than the conventional methods. © 2022 The Author(s)

OCIS codes: (060.4251) Networks, assignment and routing algorithms.

1. Introduction

Emerging services have driven overwhelming bulk flows moved between datacenters (DCs) around the world [1]. In the presence of traffic fluctuation, DC providers find it difficult to provision high-bandwidth and long-lasting end-to-end (E2E) connections for bulk flows. To meet the growing demand, they have to upgrade their link capacities or lease more bandwidth even if the average utilization remains low [2]. Store-and-Forward (SnF) approach is an effective solution to mitigate this dilemma. SnF exploits abundant storage resources on DCs. A bulk flow can be temporarily stored on intermediate DCs until its next hop is less congested, as long as it can be completed before its deadline. In essence, SnF relaxes the E2E bandwidth constraint by splitting the E2E connection into segments, which provides extra flexibility for the provisioning and improves the utilization [3].

The effectiveness of SnF relies heavily on how to solve a SnF scheduling (SS) problem, where both routing path and intermediate storage DCs should be selected, both transmission and storage should be orchestrated, and both bandwidth and storage resources should be assigned. Prior studies formulated it as static optimization problems and solved them using heuristics [2–4]. However, they may be computationally impractical for dynamic traffic and large networks. Alternatively, a routing framework, namely time-shifted multilayer graph (TS-MLG), was proposed for SnF [5]. TS-MLG models the dynamics of the network as a sequence of layers, with each layer interpreted as a snapshot of network status at a certain time. The SS problem can be formulated as a routing problem on the TS-MLG, where both transiting and storing can be incorporated into a routing path across the TS-MLG. This greatly simplified the SS problem under the dynamic traffic. However, due to the snapshot nature, the size of TS-MLG may dramatically increase when the network status frequently changes or the temporal scheduling window widens. This results in a huge expansion of the TS-MLG and hinders its practicality.

Machine learning (ML) has attracted a strong attention for its excellent performance in solving complex problems. Prior studies directly learnt the optimal next hop, path selection or scheduling schemes under particular network scenarios [6]. They may suffer from unsatisfied accuracy when the network scenario changes. Besides, they were trained on static graphs and hence are difficult to handle the dynamics of the TS-MLG.

In this paper, we present a graph convolutional network (GCN)-assisted SnF scheduling method (GCN-SnF). Instead of routing on the entire TS-MLG, GCN-SnF decomposes it into multiple single-route-based subgraphs and performs the routing search respectively. Instead of predicting the routing results directly, the GCN model is used to predict whether or not a node of the subgraph could be reached via a viable routing path. These unreachable nodes will be excluded from the routing search. Our studies show that GCN-SnF obtains lower blocking probability while shortening the computation time simultaneously.

2. GCN Model

An inter-DCN is modelled by an undirected graph $G = (V, E)$. A TS-MLG G^L is composed of L layers of G at different time instants, as depicted in Fig. 1(a). Each layer is a copy of G and its connectivity represents the network status at a time instant. In G^L, spatial links connect different nodes within the same layer. Traversing a spatial link denotes transmitting data from one node to another. Temporal links connect the same nodes in different layers. Traversing a temporal link denotes storing data at a node for a certain period. By applying a routing algorithm on G^L, a viable path with sufficient resources can be found, which elaborates a scheduling solution in detail. However, the layers are dynamically added to or removed from G^L with the network status changing. A significant increase

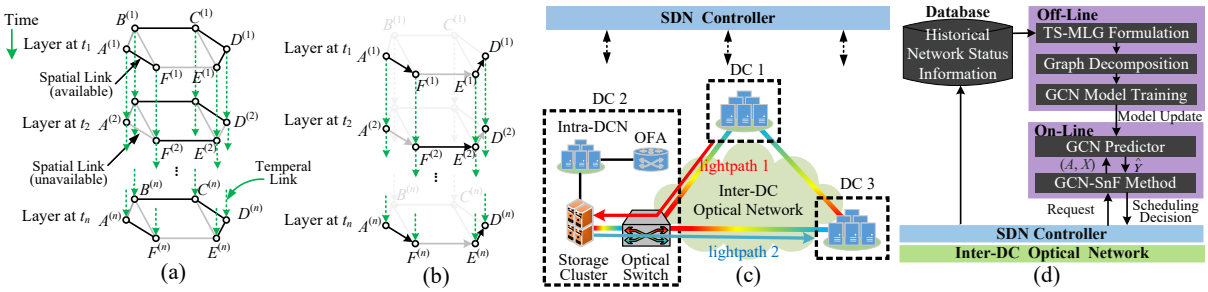


Fig. 1: (a) A TS-MLG G^L and (b) a subgraph G^L_k of G^L. (c) The network scenario. (d) The schematic of GCN-SnF.

in the layers imposes a heavy computational burden on the routing search as well as the training. In addition, the structure of the TS-MLG depends on a particular network topology. The GCN model has to be re-trained when scheduling on a different network topology.

To tackle this, we aim to decompose G^L into multiple single-route-based subgraphs. Let $R_{(s,d)}$ denote a set of pre-computed routes from a source s to a destination d. Let G^L_k denote a subgraph based on a route R_k, which contains the nodes in R_k and the links that connect these nodes, as shown in Fig. 1(b). The graph decomposition greatly reduces the size of G^L as well as the difficulty of the training. Moreover, the characteristics of the network topology are naturally excluded from the training, which improves the generalization.

We use the SnF scheduling method proposed in [5] and run extensive simulations to generate the training samples. In each run, we record the concurrent G^L periodically. Each TS-MLG sample is decomposed into subgraphs, where both the source-destination node pair and the route between them are randomly selected. For a subgraph, its adjacency matrix is denoted as A. For each node v_i in A, $x_i = [In_i, Out_i, SHC_i, THC_i, HC_i, RLU_i]$ is the feature vector of v_i. In_i and Out_i are the in-degree and the out-degree of v_i, respectively. SHC_i, THC_i and HC_i are spatial, temporal and total hop counts from s to v_i, respectively. RLU_i is the link utilization ratio of all the spatial and temporal links from s to v_i. The feature vectors of all the nodes in A form the feature matrix X. Let o_i denote the output label of v_i, which is a binary variable. When v_i can be reached via a viable path, o_i is 1; otherwise, o_i is 0. We use a two-layer GCN model with the 32-neuron hidden layer, the binary cross-entropy loss function, the ReLU activation function for the hidden layer and the softmax activation function for output layer.

3. GCN-SnF: Proposed Method

Fig. 1(c) shows an inter-DC optical network, where WDM is the infrastructure layer. DC can temporarily store bulk data at its storage cluster and enable high-speed lightpath for SnF. Optical switch is capable of wavelength conversion. SDN controller centrally manages the bandwidth and storage resources via OFA. Consider a request that aims to transfer data from DC 1 to DC 3. Link 1-3 is congested currently. The data can be first stored on DC 2 over lightpath 1 and then forwarded to DC 3 over lightpath 2 if link 2-3 becomes available earlier than link 1-3.

A request r is defined by a tuple $r = \{s, d, F, ddl\}$, where s is the source, d is the destination, F is the file size, and ddl is the deadline. D is defined as the transmission time of r, which is equal to F divided by the data rate of a wavelength. Assume request arrivals follow a Poisson process with an arrival rate of λ requests per hour. File size for each request follows the negative exponential distribution with an average of F TB. Each request occupies a wavelength for its transmission and each wavelength carries a given data rate. Request deadline is set to be $\delta \cdot D$, where δ denotes the tightness of deadlines.

We present a GCN-SnF method, whose schematic is depicted in Fig. 1(d). GCN-SnF performs admission control to arriving requests based on the current G^L and the GCN prediction. Once admitted, GCN-SnF guarantees the request completion before the deadline. The main features of GCN-SnF are twofold:

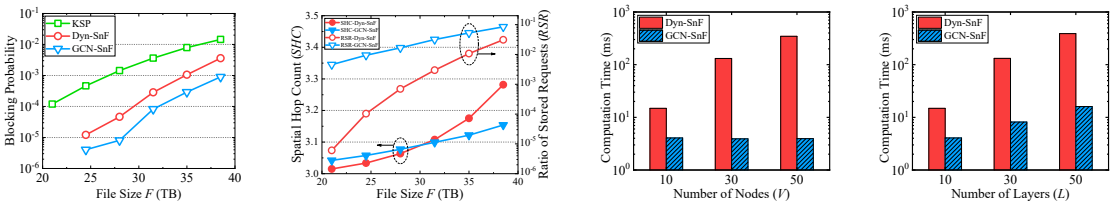
First, **route-based graph decomposition**. Instead of solving the SS problem over the entire network directly, GCN-SnF decomposes the TS-MLG into single-route-based subgraphs. As a result, the complex SS problem is decomposed into simple single-route-based SS problems. This not only alleviates the computational burden of the routing search on the TS-MLG, but also reduces the difficulty of the model learning and improves the generalization to a different network topology.

Second, **GCN-assisted reachability predictor**. Instead of learning the optimal solutions directly, GCN-SnF uses the GCN model to predict the reachability of nodes in the TS-MLG. The unreachable nodes will be excluded from the routing search. This further reduces the computational burden.

Algorithm 1 presents the overall procedure of GCN-SnF. Line 4 decomposes G^L into a smaller subgraph G^L_i based on R_i, where $R_i \in R_{(s,d)}$. The route set $R_{(s,d)}$ is pre-computed using the K-shortest routing algorithm. Lines 5-9 create an auxiliary graph G' and decide its connectivity. Line 7 calculates the maximum throughput T_e of each link in G^L_i using the algorithm in [5]. In line 8, a link e in G^L_i will be added to G' if its $T_e \geq F$. Lines 10-11 create an adjacency matrix A and a feature matrix X of G'. Lines 12-13 use the GCN model to predict the reachability $\hat{Y}$ of all the destination nodes in A and store the reachable nodes in the node set $Dest$. Lines 14-19 repeatedly apply the Breath-First Search (BFS) algorithm on $A(1{:}d_j, 1{:}d_j)$ until $Path$ from s to d_j is found, where $d_j \in Dest$ and A is

Algorithm 1 GCN-SnF Method

1: Input: r, G^L, $R_{(s,d)}$
2: Output: a viable path *Path*
3: **for** each route $R_i \in R_{(s,d)}$ **do**
4: Decompose G^L into G_i^L
5: Create an auxiliary graph G'
6: **for all** link e in G_i^L **do**
7: Calculate T_e of link e
8: Add link e to G' if $T_e \geq F$
9: **end for**
10: Get an adjacency matrix A of G'
11: Extract the feature x_i and form them into a feature matrix X
12: $\hat{Y} \leftarrow GCN\_Model(A,X)$
13: Find reachable destinations in $\hat{Y}$ and store them in the set *Dest*
14: **for** a node $d_j \in Dest$ **do**
15: Apply BFS on $A(1{:}d_j,1{:}d_j)$ to find *Path* from s to d_j
16: **if** *Path* is found **then**
17: **return** *Path*
18: **end if**
19: **end for**
20: **end for**
21: Reject r and **return** $Path \leftarrow \varnothing$



(a) (b) (c) L is fixed to 10 (d) V is fixed to 10

Fig. 2: Simulation results in USNET (a) and (b), computation time in random network topologies (c) and (d).

incrementally input into the routing search with d_j. Line 17 returns *Path* and accepts r if it is succeeded; otherwise line 21 rejects r. Moreover, the historical network status will be collected and used as new samples. The GCN model should be updated periodically and adapted to the change of the network condition and the traffic pattern.

4. Evaluation and Discussion

We compare GCN-SnF with two methods as follows. (i) **Dyn-SnF** [5]: a SnF scheduling method. It applies Dijkstra's algorithm on the entire TS-MLG. To bound its complexity, Dyn-SnF limits the number of layers used for routing. Herein, the number is 100. A request will be rejected if no viable path can be found within either the layer constraint. (ii) **KSP** [7]: a single-path scheduling method without SnF capability. It uses K-shortest-path routing algorithm. A request will be rejected if no viable E2E path can be found immediately. Besides, the USNET network topology is used in simulations. The storage capacity on each DC for SnF is 500 TB. Five wavelengths on each link are dedicated for requests, and each wavelength carries 10 Gbps. K=5, λ=6, δ=2, $F \in$[20,40] TB.

Fig. 2(a) shows with SnF capability, GCN-SnF and Dyn-SnF obtain lower blocking probability than KSP. GCN-SnF has the lowest blocking probability. To bound the complexity, Dyn-SnF only searches on a limited number of layers and hence suffers from degraded performance. Unlike Dyn-SnF, GCN-SnF would not impose any layer limitation on the routing search, which suggests more layers within the deadline constraint can be taken into account by GCN-SnF. Fig. 2(b) shows that more requests are provisioned with SnF in GCN-SnF than that in Dyn-SnF. GCN-SnF searches each G_i^L one-by-one until a viable path is found. As a G_i^L represents the dynamics of a route, GCN-SnF executes the routing search on a route-by-route basis. In other words, GCN-SnF may provision a request with SnF on the first route, irrespective of whether or not it could provision the request without SnF on the second route. As a result, GCN-SnF prefers to use shorter routes at the cost of more SnF. Figs. 2(c) and (d) show the average computation time to perform a routing search on a randomly generated TS-MLG. In Fig. 2(c), the computation time of Dyn-SnF greatly increases with V. The increase in GCN-SnF is negligible, because GCN-SnF decomposes G^L into G_i^L and its complexity mainly depends on the size of G_i^L, regardless of V. In Fig. 2(d), the increase in Dyn-SnF is higher than GCN-SnF, because GCN-SnF can search fewer layers than Dyn-SnF.

5. Conclusion

GCN-SnF is presented for inter-DC transfers, which uses the graph decomposition and the GCN prediction. Studies show that GCN-SnF obtains lower blocking probability and computation time than Dyn-SnF and KSP.

Acknowledgement: This research is supported by the NSFC Grant 61901118 and Grant 61901053.

References

1. L. Luo, et al, in *IEEE/ACM Int. Symp. Quality of Service (IWQoS)*, (2022), pp. 1-10.
2. Y. Liu, et al, IEEE Trans. Netw. Serv. Manag. **18**, 2598-2611 (2021).
3. P. Lu, et al, J. Light. Technol. **35**, 5335-5346 (2017).
4. L. Luo, et al, IEEE J. Sel. Areas Commun. **38**, 1584-1599 (2020).
5. X. Lin, et al, J. Opt. Commun. Netw. **8**, 162-174 (2016).
6. F. Tang, et al, IEEE Commun. Surv. Tutor. **23**, 1578-1598 (2021).
7. W. Lu, et al, in *Int. Conf. Opt. Commun. Netw. (ICOCN)*, (2016), pp. 1-3.