A Dynamic Neural Network for Solving Time-varying Shortest Path with Hop-constraint

Zhilei Xu\textsuperscript{1,2}, Wei Huang\textsuperscript{1,2,*}, Jinsong Wang\textsuperscript{1,2}

\textsuperscript{1}School of Computer Science and Engineering, Tianjin University of Technology, Tianjin 300384, China

\textsuperscript{2}Tianjin Key Laboratory of Intelligent and Novel Software Technology, Tianjin University of Technology, Tianjin 300384, China

*huangwabc@163.com

Abstract. This paper proposes a dynamic neural network (DNN) to solve the time-varying shortest path problem with hop-constraint (HC-TSPP). The purpose of HC-TSPP is to find a path with the shortest transmission time and the restricted number of arcs. The proposed DNN is a novel neural network based on dynamic neurons. All neurons on DNN are computing in parallel, and each dynamic neuron is composed of seven parts: input, wave receiver, filter, status memorizer, wave generator, wave sender, and output. Wave is the carrier of neuron communication, and each wave is composed of three parts. The shortest path report is based on the first wave that reaches the destination node and satisfies the hop constraint. The example and experimental results based on the Internet dataset show that the proposed algorithm can arrive at the global optimal solution and outperform the existing algorithm (viz. Dijkstra algorithm).

1. Introduction

The shortest path problem is a classic combinatorial optimization problem, which is widely used in communication routing [1], traffic planning [2], project scheduling [3], and other fields. Dijkstra [4] first proposed an algorithm for solving the shortest path problem on static graphs, namely the well-known Dijkstra algorithm. Then, a large number of improved algorithms [5, 6] have been proposed to solve this problem.

The shortest path problem with hop-constraint is to find the path with the shortest distance between two nodes and the number of hops does not exceed the constraint. To our best knowledge, this problem was first initialized by Dahl and Gouveia [7], who studied the shortest path problem with hop constraints on a static directed network. After that, Huygens and Ridha [8] considered two 4-hop constrained path problems, and proposed the integer programming formulas of the node version and the edge version of the problem. Guo et al. [9] proposed a minimum cost path algorithm based on the Dijkstra algorithm. Riedl [10] proposed a jump inequality to solve the shortest path problem with hop constraints. Although these methods have many advantages in solving the shortest path problem with hop constraints on static networks, they are difficult to solve the shortest path problem with hop constraints on non-static networks, such as time-varying networks.

The time-varying network is a kind of dynamic network, which exists widely in the real world. Compared with the traditional static network, the length of the arc on the time-varying network is not fixed but time-varying. Recently, the shortest path problem on time-varying networks has received...
widespread attention [11, 12]. However, to our best knowledge, there is no study on the shortest path problem with hop-constraint on time-varying networks.

In this paper, we propose a dynamic neural network (DNN) to solve the time-varying shortest path problem with hop-constraint (HC-TSPP). The purpose of HC-TSPP is to find a path from the start node to the destination node with the shortest transmission time and a limited number of arcs on the time-varying network. The proposed neural network is based on dynamic neurons and without any training requirements. The rest of this paper is organized as follows: in section 2, we describe the definition of the HC-TSPP; section 3 will give the DNN architecture and algorithm; section 4 reports the experimental results and section 5 gives the conclusions.

2. Preliminaries

Definition 1 (Time-varying arc). An arc $e = (x_d, x_a)$ is defined as a time-varying arc if and only if the arc length is regarded the transmission time (delay) and it is time-varying.

It means that the time to pass the same arc in different periods is different, which depends on the departure time $d_{x_d}$ from the tail node $x_d$, it can be represented as $l_e(d_{x_d})$. It is worth noting that when the arc length of the time-varying arc is a constant in the finite time interval, then the time-varying arc can be regarded as a static arc in the time-varying interval. That is, the traditional static arc is a special case of the time-varying arc.

Definition 2 (Time-varying network). A directed connected network $G = (V, A)$ is defined as a time-varying network if and only if the network contains at least one time-varying arc.

According to the characteristics of the time-varying arc and time-varying network, it can be seen that the traditional static network is a special case of a time-varying network.

Definition 3 (Time-varying path). A sequence $P = \{x_1, x_2, ..., x_n\}$ is defined as the time-varying path if and only if $d_{x_{i+1}} \geq d_i + l_{x_i,x_{i+1}}(d_i)$, $i = 1, 2, ..., n - 2$.

Definition 4 (Time-varying shortest path problem with hop-constraint, HC-TSPP). An HC-TSPP $Q = (s, z, t_s, H^{MAX})$ is to find a time-varying path $P^{min}$ with the shortest transmission time from the start node $s$ to the destination node $z$ and the number of hops meets the constraints $H^{MAX}$. Where $t_s$ is the earliest time allowed to depart from node $s$.

The transmission time $T_p$ of the time-varying shortest path $P^{min}$ can be calculated by the following expression:

$$T_p = a_z - t_s$$

Where, $a_z$ is the arrival time of the destination node.

Table 1. The explanation of symbols.

| Symbols | Explanation |
|---------|-------------|
| $H^{MAX}$ | The threshold of hops of a path. |
| $F_{f,i}(d_f)$ | A wave from neuron $f$ to $i$ and generated by $f$ at time $d_f$, which is equal to $\{H_{f,i}(d_f), P_{f,i}(d_f), D_{f,i}(d_f)\}$. |
| $H_{f,i}(d_f)$ | The number of hops of the path $P_{f,i}(d_f)$. |
| $P_{f,i}(d_f)$ | A path from start neuron passed through $f$ to $i$ and left $f$ at $d_f$. |
| $D_{f,i}(d_f)$ | The departure time from each neuron on path $P_{f,i}(d_f)$ except for neuron $i$. |
| $H_i$ | The number of hops that recorded by neuron $i$. |
| $P_i$ | The path that was recorded by neuron $i$. |
| $D_i$ | The departure time sequence recoded by neuron $i$. |
| $\Delta t$ | The step of neuron update. |
3. A Dynamic Neural Network for solving HC-TSPP

3.1. Design the dynamic neural network

The dynamic neural network (DNN) is based on a novel dynamic neuron without any requirements of training. The structure of DNN depends on the topology of the problem’s network, that is, each node corresponds to a neuron, and each arc corresponds to a corresponding link on the neural network. In particular, since the network is directed, the links on the neural network are also directed.

The communication between the neurons on the neural network is carried out through waves. Each wave contains three parts of information. Once received by a certain neuron, the life cycle of the wave is over.

Each dynamic neuron consists of seven parts (see Fig.1): input, wave receiver, filter, status memorizer, wave generator, wave sender, and output. The function and description of each part are portrayed as following, and all symbols are summarized in Table 1.

**Input**: The input is the port where the wave enters the current neuron.

**Wave Receiver**: The wave receiver is used to receive waves from precursor neurons.

**Filter**: The filter consists of two components. The \( \min \) component is used to filter out the wave \( Y_{f,i}(a_i) \) with the least number of hops, and the judging component is used to determine whether the number of hops of the filtered wave exceeds the maximum hop limit, where \( a_i = d_f + l_{f,i}(d_f) \). The expression of \( \min \) component as following:

\[
Y_{f,i}(a_i) = \min \{ Y_{g,i}(a_i), ..., Y_{h,i}(a_i) \}
\]  

**Status memorizer**: The status memorizer is used to store the state of the neuron, which consists of three parts: \( H, P, \) and \( D \).

**Wave generator**: The function of this part is to generate new waves. Which consists of \( H_{i,j}(d_i), P_{i,j}(d_i), D_{i,j}(d_i), \) and \( Y_{i,j}(d_i) \), where, \( d_i = a_{i-1} + l_{i,j}(d_i) \), \( j = m, ..., n \). Their expressions as following:

\[
\begin{align*}
H_{i,j}(d_i) &= H_i + 1 \\
P_{i,j}(d_i) &= P_i \cup \{j\} \\
D_{i,j}(d_i) &= D_i \cup \{t\} \\
Y_{i,j}(d_i) &= \{H_{i,j}(d_i), P_{i,j}(d_i), D_{i,j}(d_i)\}
\end{align*}
\]  

Where, \( d_i = t \) is the current time.

**Wave sender**: The wave sender is used to send the newly generated wave to the successor neuron. Each wave passes through a delay channel before leaving the current neuron. The delay channel simulates the transmission delay of the wave on the real network. The length of the delay channel (that is, the time of the wave passes through the delay channel) is determined by the moment when the wave is generated.

**Output**: The output is the port where the wave leaves the current neuron and it is used to send the wave to the corresponding neuron.

![Fig.1 The structure of a general neuron on DNN.](image-url)
3.2. DNN Algorithm
The basic idea of using DNN to solve HC-TSPP is that the shortest path is carried by the wave that reaches the destination neuron early and satisfies the hop constraint. The DNN also needs to follow the following mechanism when solving HC-TSPP. 1) all neurons receive waves at the same time before the neural network stops running; 2) in addition to receiving waves, activated neurons also generate new waves; 3) the neuron will activate if and only when it receives a wave that meets the hop constraint; 4) the condition for the neural network to stop running is that the destination neuron receives a wave that meets the hop constraint. Algorithms 1-3 describe the process of using DNN to solve HC-TSPP.

Algorithm 1: Dynamic neural network algorithm (DNN)
Input: \( V, E, s, z, t_0, \Delta t \).
Output: \( P_{min} \).
Step 1. Set \( t = t_0 \).
Step 2. Initialize each neuron by using DNI.
Step 3. While \( P = \emptyset \) do
Step 3.1 Update each neuron by using DNU.
Step 3.2 \( t = t + \Delta t \).
Step 4. Output shortest path \( P_{min} = \{P, D\} \).

Algorithm 2: Dynamic neuron initialization algorithm (DNI)
Input: \( s, i, H_{MAX} \).
Output: \( H, P, D \).
Step 1. If \( i = s \), then set \( H = 0, P = \{s\}, D = \emptyset \).
Step 2. If \( i \in V \setminus \{s\} \), then set \( H = H_{MAX}, P = \emptyset, D = \emptyset \).

Algorithm 3: Dynamic neuron update algorithm (DNU)
Input: \( i, t, H_{MAX}, \psi_{d}, \psi_{j} \).
Output: \( H, P, D, \psi_{d}, \psi_{j} \).
Step 1. If neuron \( i \in V \setminus \{s\} \), then
Step 1.1 Calculate \( \psi_{d}, \psi_{j} \) by using (2).
Step 1.2 If \( H_{d} \neq H_{i} \), then
Set \( H_{i} = H_{d}, P_{i} = P_{d}, D_{i} = D_{d} \).
Else return to Step 1 and wait for a time unit.
Step 2. Calculate \( H_{i,j}(d_i), P_{i,j}(d_i), D_{i,j}(d_i) \) by using (3), (4), (5), respectively. /* \( d_i = t \). */
Step 3. Calculate \( \psi_{i,j}(d_i) \) by using (6).
Step 4. Send \( \psi_{i,j}(a_{j}) \) to neuron \( j \). /* where, \( a_{j} = d_i + l_{i,j}(d_i) \). */

4. Test Results and Discussions

4.1. Example
We give an example to help readers understand our proposed method, and to verify our proposed method can solve the HC-TSPP effectively. The network structure of the given example is shown in Fig.2, and the transmission time of all arcs are shown below:

\[
\begin{align*}
l_{A,t}(t) &= \begin{cases} 1, & (0,2] \\ 10, & (2, \infty) \end{cases}, \\
l_{A,B}(t) &= \begin{cases} 1, & (0,2] \\ 10, & (2, \infty) \end{cases}, \\
l_{A,Z}(t) &= \begin{cases} 3, & (0,2] \\ 10, & (2, \infty) \end{cases}, \\
l_{B,Z}(t) &= \begin{cases} 2, & (0,2] \\ 1, & (2, \infty) \end{cases}
\end{align*}
\]

Fig.2 An example of DNN.
Table 2 describes the state values at each moment when using DNN to solve \( Q = (S, Z, 0, 2) \). The value at \( t = 0 \) is after running step 2 of DNN. The values at other times are after each update of the neuron. In the calculation process, at \( t = 3 \), we can notice that the neuron \( Z \) receives a wave from neuron \( B \), but because the number of hops is equal to 3 that exceeds the constraint hops, it cannot be the shortest path. Then, when \( t = 3 \), as shown in Table 2, the destination neuron receives the first wave that meeting the hop-constraint, so the shortest path is \( P_{min} = \langle S, A, Z, <0,1> \rangle \).

| Neuron | State | \( t = 0 \) | \( t = 1 \) | \( t = 2 \) | \( t = 3 \) | \( t = 4 \) |
|--------|-------|-------------|-------------|-------------|-------------|-------------|
| A      | \( 2 \) | \( 0 \)     | \( 0 \)     | \( 1 \)     | \{\( S, A \)\} | \{\( S, A \)\} |
| B      | \( 2 \) | \( 0 \)     | \( 2 \)     | \{\( S, A, B \)\} | \{\( S, A, B \)\} | \{\( S, A, B \)\} |
| Z      | \( 2 \) | \( 0 \)     | \( 2 \)     | \( 0 \)     | \{\( S, A, Z \)\} | \{\( S, A, Z \)\} |

4.2. Experimental results

To evaluate the proposed method, we compare the performance of the well-known Dijkstra algorithm and DNN on the Internet [13] dataset. All programs and instances running a machine with Intel Core i7-8700K CPU and 8G RAM, and all programs are implemented by C#.

For convenience, relative error as an index is used to compare the performance of DNN and Dijkstra. The calculation expression of the relative error is as follows:

\[
\text{relative error} = \frac{|\text{calculate value} - \text{real value}|}{\text{real value}}
\]  

Table 3 illustrates the experimental results of DNN and Dijkstra on data sets with a different number of nodes. As shown in Table 3, the relative error of Dijkstra exceeds 4 on the data with 1000 nodes to 5000 nodes, and the relative error of DNN is always equal to 0. This means that the proposed algorithm can arrive at the global optimal solution regardless of the number of nodes. Also, DNN takes about a quarter of the time of Dijkstra’s algorithm to solve the data of 1000 nodes, it is about seven times faster than Dijkstra when it comes to solving 5000 nodes of data. The above results show that the proposed algorithm outperforms the Dijkstra in terms of calculation accuracy and speed.

| Algorithm | Performance index | Number of nodes |
|-----------|------------------|-----------------|
|           | CPU time (s)     | 1000 2000 3000 4000 5000 |
| Dijkstra  | 0.018 0.081 0.186 0.338 0.509 |                   |
| Relative error | 4.39 4.43 4.51 4.35 4.25   |                   |
| DNN       | 0.005 0.029 0.037 0.057 0.074 |                   |
| Relative error | 0.00 0.00 0.00 0.00 0.00 |                   |

Without loss of generality, all results are taken from the average of 50 random operations.
Fig.3 further describes the impact of network complexity on performance. As shown in Fig.3(a), the relative error of Dijkstra shows an upward trend with the increase in network complexity ($|A|/|V|$), however, the accuracy of DNN is not affected. Fig.3(b) describes the effect of the number of nodes on the calculation time. As shown in Fig.3(b), as the number of nodes increases, the calculation time spent by DNN and Dijkstra both show an upward trend. The difference is that the rising speed of DNN is much slower than Dijkstra. Therefore, DNN can still respond faster than Dijkstra when applied to large-scale networks.

5. Conclusion
This paper proposed a dynamic neural network based on a novel dynamic neuron. Unlike traditional neural networks, DNN is a parallel computing neural network without any training requirements. Under ideal circumstances (each neuron is assigned to a processor), the response time of DNN depends on the calculation time of a single neuron. This can greatly reduce computing time, making it is possible to apply in large-scale networks.

In practical applications, DNN is difficult to solve the shortest path problem with hop-constraint on the uncertain time-varying environment. In the future study, the structure of neurons should be improved to adapt to the uncertain time-varying network, such as a random time-varying network.

Acknowledgments
This work was supported by the National Natural Science Foundation of China (61673295), by the Natural Science Foundation of Tianjin for Distinguished Young Scholars (19JCJQC61500), by the Natural Science Foundation of Tianjin (General Program) (18JCJQJC85200), by the National Key R&D Program of China (2018YFC0831405), and by the Natural Science Foundation of Tianjin (Key Program) (18JCZDJC30700).

References
[1] Guo Y, Li S, Jiang W, et al. Learning automata-based algorithms for solving the stochastic shortest path routing problems in 5G wireless communication[J]. Physical Communication. 2017, 25: 376-385.
[2] He Y, Liu Z, Shi J, et al. K-Shortest-Path-Based Evacuation Routing with Police Resource Allocation in City Transportation Networks[J]. PLoS One. 2015, 10(7): e131962.
[3] Gaujal B, Navet N, Walsh C. Shortest-path algorithms for real-time scheduling of FIFO tasks with minimal energy use[J]. ACM Transactions on Embedded Computing Systems. 2005, 4(4): 907-933.
[4] Dijkstra E W. A note on two problems in connexion with graphs[J]. Numerische Mathematik. 1959, 1(1): 269-271.
[5] Xu M H, Liu Y Q, Huang Q L, et al. An improved Dijkstra's shortest path algorithm for sparse network[J]. Applied Mathematics and Computation. 2007, 185(1): 247-254.
[6] Chen B Y, Chen X, Chen H, et al. Efficient algorithm for finding k shortest paths based on re-optimization technique[J]. Transportation Research Part E: Logistics and Transportation Review. 2020, 133: 101819.
[7] Dahl G, Gouveia L. On the directed hop-constrained shortest path problem[J]. Operations Research Letters. 2004, 32(1): 15-22.
[8] Huygens D, Ridha Mahjoub A. Integer Programming Formulations for the Two 4-Hop-Constrained Paths Problem[J]. Networks. 2007, 49(2): 135-144.
[9] Guo Y, Xu L, Wang J. A dynamic routing algorithm based on jump-bounded[J]. Information Technology. 2009(7): 214-218(in Chinese).
[10] Riedl W F. A complete characterization of jump inequalities for the hop-constrained shortest path problem[J]. Discrete Applied Mathematics. 2017, 225: 95-113.
[11] Hashemi S M, Mokarami S, Nasrabadie E. Dynamic shortest path problems with time-varying costs[J]. Optimization Letters. 2010, 4(1): 147-156.
[12] Huang W, Yan C, Wang J, et al. A time-delay neural network for solving time-dependent shortest path problem[J]. Neural Networks. 2017, 90: 21-28.

[13] Internet datasets, http://www-personal.umich.edu/~mejn/netdata/.