

Alternative method of counting the number of efficient paths in a transportation network

Rong Zhao, Xiangdong Xu & Anthony Chen

To cite this article: Rong Zhao, Xiangdong Xu & Anthony Chen (2021): Alternative method of counting the number of efficient paths in a transportation network, Transportmetrica A: Transport Science, DOI: [10.1080/23249935.2021.1933255](https://doi.org/10.1080/23249935.2021.1933255)

To link to this article: <https://doi.org/10.1080/23249935.2021.1933255>



© 2021 The Author(s). Published by Informa UK Limited, trading as Taylor & Francis Group.



Published online: 15 Jun 2021.



Submit your article to this journal [↗](#)



Article views: 373




View related articles [↗](#)



View Crossmark data [↗](#)

Alternative method of counting the number of efficient paths in a transportation network

Rong Zhao^a, Xiangdong Xu^{a,b} and Anthony Chen ^c

^aThe Key Laboratory of Road and Traffic Engineering, Ministry of Education, Tongji University, Shanghai, People's Republic of China; ^bUrban Mobility Institute, Tongji University, Shanghai, People's Republic of China; ^cDepartment of Civil and Environmental Engineering, The Hong Kong Polytechnic University, Kowloon, Hong Kong, People's Republic of China

ABSTRACT

The number of efficient paths between an origin-destination (O-D) pair provides the route diversity degree of possibly used paths in a transportation network and has many important applications. The existing counting method was based on the Bell loading method for Logit model to determine the number of efficient paths between any two nodes without path enumeration. However, this method has a high time-complexity and requires many unnecessary computations, which significantly hinder its use in large-scale networks. Inspired by the Dial loading method for Logit model, this paper develops a more computationally attractive method to count not only the number of efficient paths, but also the number of efficient paths using each link/node, and the total/average cost of these paths between each O-D pair. Besides circumventing path enumeration, the proposed method has a much lower time-complexity. Numerical examples are then provided to demonstrate the validity and efficiency of the proposed method..

ARTICLE HISTORY

Received 2 December 2020
Accepted 17 May 2021

KEYWORDS

Efficient path; Dial loading;
Bell loading; logit

1. Introduction

1.1. Efficient path and applications of counting efficient path

The concept of *efficient* (or *reasonable*) *path* was originally proposed by Dial (1971) for the Logit stochastic network loading problem. From the computational perspective, it was used to load the origin-destination (O-D) travel demand to a network following the classical Logit route choice model without the need to enumerate and store paths, i.e. the Dial (or STOCH) loading method. From the behavioral perspective, it was used to 'restrain' the path set to only consider a set of reasonable or efficient paths rather than all available paths in a network. For this restriction, Dial (1971) defined efficient path as 'further away from the origin and closer to the destination'. Tong (1990) relaxed it to either 'further away from the origin' or 'closer to the destination'. Leurent (1997) further considered the not-too-long paths with acceptable travel time. Si et al. (2010) suggested that both the link travel cost and the minimum travel costs from the corresponding starting node to the origin and from the ending

CONTACT Xiangdong Xu  xiangdongxu@tongji.edu.cn

© 2021 The Author(s). Published by Informa UK Limited, trading as Taylor & Francis Group.

This is an Open Access article distributed under the terms of the Creative Commons Attribution-NonCommercial-NoDerivatives License (<http://creativecommons.org/licenses/by-nc-nd/4.0/>), which permits non-commercial re-use, distribution, and reproduction in any medium, provided the original work is properly cited, and is not altered, transformed, or built upon in any way.

node to the destination should also be taken into account in the STOCH loading method, and only those paths within a small range beyond the shortest paths are considered as efficient paths.

The number of efficient paths and the number of efficient paths passing through each link/node between an O-D pair in a network have many important applications in practice. For example, the number of efficient paths between an O-D pair has been used as a route diversity/redundancy measure for both road networks (Xu et al. 2018) and metro networks (Yang et al. 2017; Chan et al. 2021; Jing, Xu, and Pu 2019). In freight transportation networks, the resilience measure can be considered as a function of the number of reliable paths between all node pairs (Ip and Wang 2009; Miller-Hooks, Zhang, and Faturechi 2012). The number of efficient paths between an O-D pair passing through each link (or the ratio of the number of efficient paths between an O-D pair passing through a link to the total number of efficient paths between an O-D pair) has been used in the network sensor location problem (Meng, Lee, and Cheu 2005). As mentioned in Meng, Lee, and Cheu (2005), the above ratio can measure the capability of a link (i.e. a located sensor) in intercepting traffic flow information in a certain O-D pair. Hence, it can be used to develop methods for solving the optimal network sensor location problems (i.e. finding 'higher ratio' links for multiple O-D pairs), such as the classical traffic counting location problem for O-D matrix estimation (Bianco, Confessore, and Reverberi 2001; Chootinan, Chen, and Yang 2005; Chen et al. 2007; Larsson, Lundgren, and Peterson 2010) and the traffic detector location problem (Chen, Chootinan, and Pravinongvuth 2004, 2010; Zhou and List 2010; Gentili and Mirchandani 2012; Castillo et al. 2015; Xu et al. 2016). Also, it can be used to identify the critical links/nodes for transportation network vulnerability analysis without assuming the disruption scenarios (Chen et al. 2018; Jing, Xu, and Pu 2020). Similarly, the ratio of the number of paths passing through a node in a network to the total number of paths in the network has been defined as the flow betweenness centrality for measuring the node importance in the complex network theory (Freeman, Borgatti, and White 1991). The total (or average) cost of all (efficient) paths between an O-D pair can be used to measure the spatial accessibility and to address the spatial equity issue in the context of connectivity vulnerability (Kurauchi et al. 2009). Boer et al. (2017) suggested that the travel time reliability indicators at the O-D level must take the number of available alternatives into account. Overall, the above various important applications highlight the necessity and importance of developing an efficient method for counting the number of efficient paths between each O-D pair, the number of efficient paths using each link/node between each O-D pair, and the total/average cost of these efficient paths between each O-D pair in large-scale networks.

Besides transportation applications, counting the number of paths in a network also has many applications in other disciplines, e.g. measuring centrality of a node within a complex network, DNA sequence comparison, measuring parts usage in manufacturing, measuring redundancy in sociometric chains and aircraft door management system, and quantifying the boiling points of organic compounds. In the DNA sequence comparison, the number of optimal alignments of two DNA sequences can be modeled by counting paths in the scoring matrix (Havlin and Cohen 2010; Hochberg 2012). Hence, a similarity measure can be obtained by the counting process. Similar problems also exist in the transportation field, e.g. measuring sequencing similarity for trajectory clustering (Kim and Mahmassani 2015) and for trips of Bluetooth data (Crawford, Watling, and Connors 2018). Path counting problem may provide an alternative way to measure similarity. For measuring the parts usage

in manufacturing, a product family relationship network can be constructed by treating the parts as nodes and the affiliation relationship between the parts as directed links in the network. Then, the total parts usage can be modeled as the number of paths through the parts in the product family relationship network (Liu, Qi, and Che 2006). In the research of cognition, learning, verbal behavior, communication, sociometry, and social interaction, empirical structures are often represented by digraphs, in which each node corresponds to an empirical entity (i.e. a person) and each directed line corresponds to an empirical relationship. In sociometry, the number of redundant paths in these relationship digraphs (usually represented by communication matrix) can be applied to determine the redundancies of sociometric chains (Cartwright and Gleason 1966; Ross and Harary 1952). As for the aircraft door management system (DMS), k -different paths (more strictly, node-disjoint or arc-disjoint paths) can be used as k -redundancy for each function (Schäfer et al. 2018). In the field of molecular topological chemistry, the polarity number p is defined as the number of paths between carbon atoms pairs which are separated by three carbon-carbon (C-C-C-C). This polarity number p can characterize the shape of the molecule and help to quantify the boiling points of organic compounds (Wiener 1947; Zhou, Chu, and Nie 2007).

1.2. Different path counting problems and polynomial algorithms

Counting problem often arises in situations where we have to estimate the probability (i.e. reliability analysis), whose objective is to determine the number of configurations of a particular type (Ball 1986; Sanjeev and Boaz 2009). Most counting problems, including path counting problem, belong to a class of intractable problem called #P-Complete. Valiant (1979a) provided one of the first definitions of these classes and pointed out that #P-Complete problem cannot be solved by a polynomial-time algorithm. Besides transportation networks, some efforts have been done on counting the number of paths in grids: counting paths in the so-called Young's lattice (Gessel 1993) and more general results were obtained by Bartholdi (1999) and Stanley (1996). Another important set of counting path methods lies in the computer path profile (Ball and Larus 1997; Young and Smith 1999), which determines how many times each acyclic path in a routine executes.

From the viewpoint of computational complexity, a polynomial-time algorithm is regarded as an efficient algorithm. Since counting paths in a network is a class of #P-Complete problem, it is usually difficult and less efficient to solve. However, it is possible that computationally efficient algorithms exist for some special classes of networks (e.g. tree networks, directed acyclic networks) (Ball 1986; Dias et al. 2017), or special constrained paths rather than just simple paths. Specifically, polynomial-time algorithms exist in the following special cases of counting path problem, although the problem itself, in general, is a #P-Complete problem.

- (1) **Shortest path counting problem.** Minimum cardinality path set counting problem has been widely studied in the field of computational complexity, and the number of shortest paths can be modified by such algorithms (Ball 1986). For *2-terminal* problem (O-D paths), a simple extension of Moore's Breadth-First Search (BFS) shortest path algorithm can be used to count the shortest paths in polynomial time (Ball and Provan 1983). For *all-terminal* problem (paths from origin to all the other nodes), the historic matrix tree theorem together with efficient algorithm for finding the determinant

of a matrix provides a polynomial algorithm for counting origin-directed pathsets (Kirchhoff 1958).

- (2) **Simple paths with length constraint (SPLC).** SPLC problem is a NP-Complete problem in general networks, unless $NP = P$, otherwise there is no polynomial time algorithm (Wu et al. 2010). The counting of SPLC problem is also #P-Complete. However, polynomial time algorithm can be found for some special networks such as directed acyclic networks (e.g. He et al. 2007). Li et al. (2012) proposed a polynomial-time algorithm of SPLC based on nettree data structure in directed acyclic networks.
- (3) **Simple (efficient) path counting problem.** In general transportation networks, counting the different simple paths between O-D pairs has been proven as a #P-Complete problem (Valiant 1979a; Roberts and Kroese 2007). Valiant (1979b) pointed out polynomial-time algorithm that is able to compute the number of different simple paths between any two nodes does not exist for either directed or undirected graphs. However, counting the different efficient paths (a special case of simple path) between an O-D pair is not a #P-Complete problem (Meng, Lee, and Cheu 2005). That is because the subnetwork after removing inefficient links/nodes is actually an acyclic directed network instead of the original general network. Directed acyclic graphs do have special properties, and simple paths counting problem in an acyclic directed network has a polynomial-time algorithm.

1.3. Observation of the existing method and contribution of this paper

Based on Bell's stochastic loading method (Bell 1995) and Floyd's shortest path method (Floyd 1962), Meng, Lee, and Cheu (2005) proposed a polynomial-time combinatorial method to count the number of efficient paths between an O-D pair as well as the total travel cost of these paths without path enumeration. For convenience, this method is referred to as the *Bell counting method* in this paper. The Bell counting method mainly includes two parts: (1) construct an acyclic sub-network for each origin according to the definition of efficient path, in such a way that network elements that do not satisfy the definition of efficient path are removed; and (2) a triangle operation for any three nodes is then performed in the sub-network of each origin. The second part corresponds to a polynomial-time combinatorial method for counting the different simple paths between any two nodes in the sub-network.

Similar to Floyd's shortest path method (Floyd 1962), the Bell counting method is quite easy to implement. However, it has a high computational time complexity $O(|R| \cdot |N|^3)$, where $|R|$ is the number of origins and $|N|$ is the number of nodes in the network. In other words, it requires a large amount of computations in large-scale transportation networks. Take the Winnipeg network (1,067 nodes, 2,535 links, 135 origin zones, and 4,345 O-D pairs) as an example, $|R| \cdot |N|^3$ is $135 \times 1067^3 = 1.64 \times 10^{11}$. One can envision the non-linear increase of computational burden for even larger networks. A main reason for this burdensome computation is that for each origin, the Bell counting method needs to count the efficient paths for all node pairs instead of the considered O-D pairs. This procedure wastes a lot of computations for unnecessary outputs. However, these calculations cannot be avoided in the *Bell counting method*. Continue taking the Winnipeg network as an example, usually we only need outputs of 4,345 O-D pairs, but the *Bell counting method* needs to generate outputs of 1067×1067 (i.e. 1.14×10^6) node pairs. Moreover, to obtain

the number of efficient paths using a link/node or the total travel cost of all efficient paths between an O-D pair, the *Bell counting method* requires extra computational efforts, to be discussed in detail in Section 2.

Recently, there are renewed interests to the path counting problem in transportation networks. It has some important emerging applications, such as modeling route diversity/redundancy for both road networks (Xu et al. 2018) and metro networks (Yang et al. 2017; Chan et al. 2018; Jing, Xu, and Pu 2019), critical link/node identification in a network (Chen et al. 2018; Jing, Xu, and Pu 2020) and travel time reliability assessment problem (Boer et al. 2017). A computationally efficient algorithm is an important guarantee of these emerging applications. On the other hand, the larger-scale (in terms of the number of nodes, links, zones and O-D pairs) and higher-complexity of modern transportation networks also require new algorithm developments of the path counting problem in order to resolve the high computational burden issue. Besides the above evaluation applications, when the path counting problem is treated as a lower-level subprogram of bi-level programming for optimizing redundancy (Xu et al. 2018) or vulnerability (Xu, Chen, and Yang 2018), the efficiency of solving the lower-level subprogram will become much more critical because it will be repeated for each iteration of the entire bi-level programming. Hence, this paper seeks to answer the following question: *Is it possible to develop an alternative method to count the number of efficient paths between each O-D pair in a network more efficiently?*

The contribution of this paper is the **development of a Dial counting method with a more efficient computational time complexity** of $O(|W| \cdot |A|)$, where $|W|$ is the number of O-D pairs and $|A|$ is the number of links. Inspired by the STOCH loading method by Dial (1971), the proposed method has a much lower time-complexity not only to count the number of efficient paths between each O-D pair, but also the number of efficient paths using each link/node between each O-D pair, and the total/average cost of these efficient paths between each O-D pair in a network. Besides circumventing path enumeration, it is more computationally attractive to better support the applications of the aforementioned measures in large-scale networks. The correctness of the proposed method is also proved along with numerical results on ten transportation networks of various sizes to demonstrate the validity and computational efficiency of the proposed method.

The rest of this paper is organized as follows: Section 2 briefly reviews the existing Bell counting method. Section 3 proposes the Dial counting method, and discusses the relationship of network search algorithms and the Dial counting method in counting paths. Section 4 presents a set of numerical examples to demonstrate the features of the proposed method. Finally, Section 5 concludes the study.

2. The Bell counting method

The Bell counting method proposed by Meng, Lee, and Cheu (2005) consists of two parts: (1) constructing a sub-network for each origin; and (2) calculating the number of efficient paths in the sub-network.

Consider a connected transportation network $G = (N, A)$, where N is the set of nodes, and A is the set of links. Let $|N|$ and $|A|$ denote the number of nodes and the number of links, respectively; R ($R \subset N$) and S ($S \subset N$) are the set of origin nodes and destination nodes, and (r, s) is an O-D pair from origin $r \in R$ to destination $s \in S$.

For the network $G = (N, A)$, two initial matrices can be constructed: the adjacent node matrix $U = \{u(m, n)\}_{|N| \times |N|}$ and the link cost matrix $W = \{w(m, n)\}_{|N| \times |N|}$, where $\forall m, n \in N$.

The initialization of the two matrices is as follows:

$$u(m, n) = \begin{cases} 1, & \text{if there is a link from node } m \text{ to node } n \\ 0, & \text{otherwise} \end{cases}, \quad \forall m, n \in N \quad (1)$$

$$w(m, n) = \begin{cases} t_a, & \text{if there is a link from node } m \text{ to node } n \\ 0, & \text{otherwise} \end{cases}, \quad \forall m, n \in N \quad (2)$$

where t_a denotes the travel cost of link a .

2.1. Constructing a sub-network for each origin node

The main idea of constructing a sub-network for each origin r is to remove the links and nodes that fail to satisfy the definition of efficient path. Following Meng, Lee, and Cheu (2005), we use the following definition of efficient path: ‘always take users further from the origin node’. Mathematically, the sub-network $G_r = (N_r, A_r)$ for origin r is:

$$N_r = \{n | n \in N \text{ and } l_r(n) \neq \infty\} \quad (3)$$

$$A_r = \{a | a \in A \text{ and } l_r(m_a) < l_r(n_a)\} \quad (4)$$

where $l_r(n)$ is the shortest path cost from origin r to node n ; m_a and n_a are the tail node and head node of link a , respectively. The main procedure of constructing a sub-network for each origin is as follows:

```

Step 1 Initializing the sub-network  $G_r(N_r, A_r) = G(N, A)$ 
Step 2 Removing the nodes and links that do not meet the definition of efficient path
    for  $\forall r \in R$ 
        for  $\forall n \in N_r$ 
            if  $l_r(n) = \infty$ 
                 $N_r = N_r \setminus \{n\}$ 
        for  $\forall a \in A_r$ 
            if  $l_r(m_a) \geq l_r(n_a)$ 
                 $A_r = A_r \setminus \{a\}$ 

```

2.2. Calculating the number of efficient paths in the sub-network

The number of efficient paths and their total cost between any two nodes can be calculated by the following triangle operation for any three nodes based on the initialized adjacent node matrix $U_r = \{u_r(m, n)\}_{|N_r| \times |N_r|}$ and link cost matrix $W_r = \{w_r(m, n)\}_{|N_r| \times |N_r|}$ in the sub-network.

```

Step 1: Initialization of matrices  $U_r$  and  $W_r$ 
     $\mathbf{U}_{r|N_r| \times |N_r|} = \mathbf{0}, \mathbf{W}_{r|N_r| \times |N_r|} = \mathbf{0}$ 
    for  $\forall a \in A_r$ 
         $u_r(m_a, n_a) = 1$ 
         $w_r(m_a, n_a) = t_a$ 
Step 2: Triangle operation for any three nodes
    for  $\forall j \in N_r$ 
        for  $\forall m \in N_r$  and  $m \neq j$ 
            for  $\forall n \in N_r$  and  $n \neq j, n \neq m$ 

```

$$u_r(m, n) = u_r(m, n) + u_r(m, j) \cdot u_r(j, n) \quad (5)$$

$$w_r(m, n) = w_r(m, n) + w_r(m, j) \cdot u_r(j, n) + u_r(m, j) \cdot w_r(j, n) \quad (6)$$

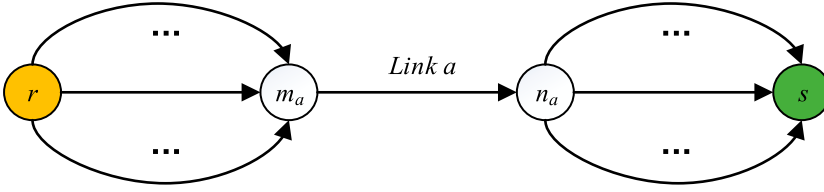


Figure 1. Topological relationship between the set of efficient paths connecting node pair (r, m_a) and node pair (n_a, s) .

where $u_r(m, n)$ is the number of efficient paths between node pair (m, n) in the sub-network of origin r , and $w_r(m, n)$ is the total travel cost of these efficient paths. Then, the number of efficient paths between O-D pair (r, s) n^{rs} and their total cost c^{rs} can be expressed as follows:

$$n^{rs} = u_r(r, s) \quad (7)$$

$$c^{rs} = w_r(r, s) \quad (8)$$

From Meng, Lee, and Cheu (2005), if $G_r = (N_r, A_r)$ is an acyclic network, then $u_r(m, n)$ and $w_r(m, n)$ calculated above are equal to the number of paths and the total cost of these paths from node m to node n in the network $G_r = (N_r, A_r)$. In addition, the sub-network $G_r = (N_r, A_r)$ generated above can be proved to be a connected and acyclic network. Then, the number of paths in $G_r = (N_r, A_r)$ is equal to the number of efficient paths between O-D pair (r, s) in $G = (N, A)$. For detailed proof, interested readers may refer to Meng, Lee, and Cheu (2005). The computational rationale of Equation (5) is that the number of paths between O-D pair (r, s) is equal to the summation of the number of paths without intermediate node and with all possible numbers of intermediate nodes (i.e. from 1 to $|N_r|-2$). Note that the convergence issue of the Bell's Logit loading method when the network includes cycles (Lam and Chan 1998; Wong 1999; Huang and Bell 1998) does not exist in the Bell counting method. Meng, Lee, and Cheu (2005) have proved that the sub-network constructed for each origin (i.e. only include efficient links in the sub-network) is a connected and acyclic network, so it does not have the convergence issue. From this perspective, the Bell counting method actually combines the Dial's concept of efficient path and the Bell's Logit loading method for developing the path counting method.

Figure 1 illustrates the topological relationship between the set of efficient paths connecting the two node pairs separated by link a . With the number of efficient paths between each node pair as calculated above, we can calculate the number of efficient paths using each link a between O-D pair (r, s) as follows:

$$n_a^{rs} = u_r(r, m_a) \cdot u_r(n_a, s), \forall a \in A, \forall r \in R, \forall s \in S \quad (9)$$

Remark 2.1: From the Bell counting procedure, we can find the following disadvantages despite its easy implementation:

- (1) What we usually need is the number of efficient paths between each considered O-D pair. However, for each origin, this method needs to count the efficient paths for all node pairs instead of the considered O-D pairs (see Equation (7)). The number of nodes

in a network is usually much greater than the number of zones. Hence, this procedure requires a lot of computations for unnecessary outputs, and these calculations cannot be avoided in the whole triangle nested operation.

- (2) When we need the number of efficient paths using each link/node between O-D pair (r, s) , this method requires two steps: the first step is to count and store the number of efficient paths between all node pairs for origin r , i.e. $u_r(m, n)$ rather than $u_r(r, s)$; and the second step is to multiply $u_r(r, m_a)$ with $u_r(n_a, s)$ as in Equation (9).
- (3) As shown in Equation (6), the total travel cost of all efficient paths is jointly calculated with the number of efficient paths. This joint calculation makes the total travel cost computation much more expensive than those for calculating the number of efficient paths.

3. The proposed Dial counting method

Inspired by the STOCH loading method by Dial (1971), this section presents a more efficient method for counting the number of efficient paths, the number of efficient paths using each link/node, and the total/average cost of these efficient paths. The proposed method can be divided into four steps: initialization, efficient path check, forward pass, and backward pass. These steps are repeated for each O-D pair in the network.

Step 1: Initialization

Calculate the minimum travel cost $l_r(i)$ from origin r to all the other nodes. There are many algorithms for solving the shortest path problem with different computational efficiency (Hung and Divoky 1988). In order to ensure a fair comparison, this paper uniformly uses the Dijkstra's method to obtain $l_r(i)$.

Step 2: Efficient path check

Efficient path check L_{ij} is performed for each link $(i \rightarrow j)$. Using the same definition of efficient path in Equation (4), the efficient path check is performed as follows:

$$L_{ij} = \begin{cases} 1, & \text{if } l_r(i) < l_r(j) \\ 0, & \text{otherwise} \end{cases} \quad (10)$$

where $l_r(i)$ denotes the minimum travel cost from origin r to the (upstream) node i ; $l_r(j)$ denotes the minimum travel cost from origin r to the (downstream) node j ; and L_{ij} is the binary result of the efficient path check for each link $(i \rightarrow j)$: when the efficient path condition is satisfied, it is recorded as 1; and 0 otherwise.

Step 3: Forward pass

Starting from origin r , consider nodes in an ascending order of $l_r(i)$. For each node i , we calculate the link weight W_{ij} for each link emanating from node i until destination s is reached.

$$W_{ij} = \begin{cases} L_{ij}, & \text{if } i = r \\ L_{ij} \cdot \sum_{m \in I_i} W_{mi}, & \text{otherwise} \end{cases} \quad (11)$$

where I_i represents the set of upstream nodes of all links arriving at node i . The summation term in Equation (11) is the number of efficient paths from origin r to any node i (including

destination s), i.e.

$$n^{ri} = \begin{cases} 0, & \text{if } i = r \\ \sum_{m \in l_i} W_{mi}, & \text{otherwise} \end{cases} \quad (12)$$

If we only need the number of efficient paths, then we just stop here; otherwise, the following ‘backward pass’ step is continued to obtain the number of efficient paths using each link/node and the total/average travel cost of all efficient paths between an O-D pair.

Step 4: Backward pass

Starting from destination node s , consider nodes in an descending order of $l_r(i)$. For each node j , calculate n_{ij}^{rs} for each link entering node j until origin node r is reached.

$$n_{ij}^{rs} = \begin{cases} W_{ij}, & \text{for } j = s \\ \left(\sum_{m \in O_j} n_{jm}^{rs} \right) \cdot W_{ij} / \sum_{m \in l_j} W_{mj}, & \text{for all other nodes } j \end{cases} \quad (13)$$

where O_j represents the set of downstream nodes of all links leaving node j , and n_{ij}^{rs} is the number of efficient paths between O-D pair (r, s) using link $(i \rightarrow j)$. A byproduct of Equation (13) is the number of efficient paths between O-D pair (r, s) using node j as shown below, which is the summation of the number of efficient paths between O-D pair (r, s) using all links leaving node j .

$$n_j^{rs} = \begin{cases} n^{rs}, & \text{for } j = s \\ \sum_{m \in O_j} n_{jm}^{rs}, & \text{for all other nodes } j \end{cases} \quad (14)$$

Note that n_{ij}^{rs} can be viewed as the usage of link $(i \rightarrow j)$ and t_{ij} is the travel cost of link $(i \rightarrow j)$. Then, we can calculate the total/average travel cost of all efficient paths between O-D pair (r, s) as follows:

$$c^{rs} = \sum_{(i \rightarrow j) \in A} n_{ij}^{rs} \cdot t_{ij} \quad (15)$$

$$\bar{c}^{rs} = \frac{1}{n^{rs}} \sum_{(i \rightarrow j) \in A} n_{ij}^{rs} \cdot t_{ij} \quad (16)$$

Alternatively, we can calculate the total travel cost of all efficient paths from origin r to any node i after the calculation of W_{ij} in the *Forward pass* step:

$$c^ri = \begin{cases} 0, & \text{if } i = r \\ \sum_{m \in l_i} L_{mi}(c^{rm} + W_{mi} \cdot t_{mi}), & \text{otherwise} \end{cases} \quad (17)$$

This step also considers nodes in an ascending order of $l_r(i)$. When $i \neq r$, we focus on all upstream nodes $m \in l_i$. Since link weight W_{mi} represents the number of efficient paths via link $(m \rightarrow i)$ between node pair (r, i) , $L_{mi}(c^{rm} + W_{mi} \cdot t_{mi})$ is the total cost of these paths as L_{mi} ensures the efficient path condition is satisfied. Hence, c^ri is the summation of the total cost of efficient paths using all upstream links arriving at node i as shown in Equation (17).

Table 1. Comparison between STOCH loading method and Dial counting method.

Steps	STOCH loading method (Dial 1971)	Dial counting method (this paper)
Initialization	$l_r(i), l_s(i)$	$l_r(i)$
Link likelihood / Efficient path check	$L_{ij} = \begin{cases} e^{\theta[l_r(i)-l_r(j)-t_{ij}]}, & \text{if } l_r(i) < l_r(j) \text{ and } l_s(i) > l_s(j) \\ 0, & \text{otherwise} \end{cases}$	$L_{ij} = \begin{cases} 1, & \text{if } l_r(i) < l_r(j) \\ 0, & \text{otherwise} \end{cases}$
Forward pass	$W_{ij} = \begin{cases} L_{ij}, & \text{if } i = r \\ L_{ij} \cdot \sum_{m \in l_i} W_{mi}, & \text{otherwise} \end{cases}$	$W_{ij} = \begin{cases} L_{ij}, & \text{if } i = r \\ L_{ij} \cdot \sum_{m \in l_i} W_{mi}, & \text{otherwise} \end{cases}$ $n^r = \begin{cases} 0, & \text{if } i = r \\ \sum_{m \in l_i} W_{mi}, & \text{otherwise} \end{cases}$ $c^r = \begin{cases} 0, & \text{if } i = r \\ \sum_{m \in l_i} L_{mi}(c^m + W_{mi} \cdot t_{mi}), & \text{otherwise} \end{cases}$
Backward pass	$x_{ij} = \begin{cases} q_{rs} \cdot W_{ij} / \sum_{m \in l_j} W_{mj}, & \text{for } j = s \\ \left(\sum_{m \in O_j} x_{jm} \right) \cdot W_{ij} / \sum_{m \in l_j} W_{mj}, & \text{for all other nodes } j \end{cases}$	$n_{ij}^{rs} = \begin{cases} W_{ij}, & \text{for } j = s \\ \left(\sum_{m \in O_j} n_{jm}^{rs} \right) \cdot W_{ij} / \sum_{m \in l_j} W_{mj}, & \text{for all other nodes } j \end{cases}$ $n_j^{rs} = \begin{cases} n_{ij}^{rs}, & \text{for } j = s \\ \sum_{m \in O_j} n_{jm}^{rs}, & \text{for all other nodes } j \end{cases}$

In summary, after repeating the above steps for all O-D pairs in the network, we can obtain the following measures

- (1) n^{rs} : the number of efficient paths between O-D pair (r, s) .
- (2) n^r : the number of efficient paths from origin r to any node i .
- (3) n_i^{rs} : the number of efficient paths between O-D pair (r, s) using node i .
- (4) $n_a^{rs}(n_{ij}^{rs})$: the number of efficient paths between O-D pair (r, s) using link a (link $(i \rightarrow j)$).
- (5) c^r : the total travel cost of all efficient paths from origin r to any node i .
- (6) c^{rs} : the total travel cost of all efficient paths between O-D pair (r, s) .
- (7) \bar{c}^{rs} : the average travel cost of all efficient paths between O-D pair (r, s) .

The proposed modifications from the STOCH loading algorithm are shown in Table 1. It is known that the STOCH algorithm is one of the widely used algorithms for solving the Logit-based stochastic assignment problem. This loading algorithm generates the assigned flow on each link that satisfies the logit model without the need of path enumeration, so it is applicable for large-scale transportation networks. The proposed Dial counting method inherits the structure (as shown in Table 1) and advantages of the STOCH algorithm, but it works for a different problem: calculating the number of efficient paths between O-D pairs in a network.

In the STOCH algorithm, link cost t_{ij} is used in the 'initialization' and 'link likelihood' steps. In our Dial counting method, t_{ij} is implicitly used in the 'initialization' step when calculating $l_r(i)$ and $l_r(j)$. However, our method does not make use of t_{ij} under this alternative path definition in the 'efficient path check' step. Herein t_{ij} is not used in the likelihood function as in the STOCH algorithm. The reason is that we just need the number of paths. L_{ij} (i.e. 0 or 1) is used to indicate whether link $(i \rightarrow j)$ is an efficient link or not, and there is no exponential form to calculate the probability. Regarding the information provided by algorithms, different from the only concern on the assignment link flow results of *backward pass* in the STOCH algorithm (i.e. x_{ij}), the Dial counting method deeply explores the physical meaning

Table 2. Comparison between Dial counting method and Bell counting method.

Variable	Dial counting method (this paper)	Bell counting method (Meng, Lee, and Cheu 2005)
# Efficient paths	Computational time-complexity $O(W \cdot A)$ $W_{ij} = \begin{cases} L_{ij}, & \text{if } i = r \\ L_{ij} \cdot \sum_{m \in l_i} W_{mi}, & \text{otherwise} \end{cases}$	Computational time-complexity $O(R \cdot N ^3)$ for $\forall j \in N_r$ for $\forall m \in N_r$ and $m \neq j$ for $\forall n \in N_r$ and $n \neq j, n \neq m$ $u_r(m, n) = u_r(m, n) + u_r(m, j) \cdot u_r(j, n)$
# Efficient paths using a link	Byproduct of the above calculation (one-step) $n_{ij}^{rs} = \begin{cases} W_{ij}, & \text{for } j = s \\ \left(\sum_{m \in O_j} n_{jm}^{rs} \right) \cdot W_{ij} / \sum_{m \in l_j} W_{mj}, & \\ \text{for all other nodes } j \end{cases}$	Multiplication of the number of node pair efficient paths separated by the link (two-step) $n_a^{rs} = u_r(r, m_a) \cdot u_r(n_a, s), \forall r \in R, s \in S$
Total cost of efficient paths	In a straightforward way $c^{rs} = \sum_{ij \in A} n_{ij}^{rs} \cdot t_{ij}$	Additional calculation for $\forall j \in N_r$ for $\forall m \in N_r$ and $m \neq j$ for $\forall n \in N_r$ and $n \neq j, n \neq m$ $w_r(m, n) = w_r(m, n) + w_r(m, j) \cdot u_r(j, n) + u_r(m, j) \cdot w_r(j, n)$

of both *forward pass* (i.e. W_{ij} , n^{ri} and c^{ri}) and *backward pass* (i.e. n_{ij}^{rs} and n_{ij}^{rs}) without additional computational burden.

Below we prove the correctness of computing the number of efficient paths between each O-D pair. After having the number of efficient paths between each O-D pair, the proofs for the number of efficient paths using each link/node and the total costs of efficient paths are more straightforward, and hence are skipped in this paper.

Proposition 3.1: *The forward pass step can obtain the number of efficient paths between each O-D pair.*

Proof: We use the induction approach for the proof. Without loss of generality, consider an O-D pair (r, s) and there are k upstream links arriving at destination node s , denoted as $(p_1 \rightarrow s), (p_2 \rightarrow s), \dots, (p_i \rightarrow s), \dots, (p_k \rightarrow s)$. For any p_i from 1 to k , assume the inductive hypothesis is true. In other words, the forward pass step has generated the number of efficient paths from origin node r to node p_i , i.e. $n^{r,p_i} = \sum_{m \in l_{p_i}} W_{m,p_i}$. Then we consider destination node s . By using the network topological order (i.e. all the efficient paths from origin node r to node p_i will pass through link $(p_i \rightarrow s)$) and the above inductive hypothesis for node p_i , we have

$$\begin{aligned}
 n^{rs} &= \sum_{p_i=p_1}^{p_k} n^{r,p_i} \cdot L_{p_i,s} \\
 &= \sum_{p_i=p_1}^{p_k} L_{p_i,s} \cdot \sum_{m \in l_{p_i}} W_{m,p_i}
 \end{aligned} \tag{18}$$

which is exactly the inductive hypothesis for $p_i = s$, i.e.

$$n^{rs} = \sum_{p_i=p_1}^{p_k} W_{p_i,s} = \sum_{p_i=p_1}^{p_k} L_{p_i,s} \cdot \sum_{m \in l_{p_i}} W_{m,p_i} \tag{19}$$

where the first and second equalities are according to Equations (12) and (11), respectively. This completes the proof. ■

Remark 3.2: The proposed method is not restricted to the definition of efficient path in Equation (10). For example, if we want to count the number of effective paths (i.e. not only efficient, but also not-too-long) as in Xu et al. (2018), Chan et al. (2018) and Jing, Xu, and Pu (2019), we can simply replace Equation (10) by the following equation, and all the other steps keep intact.

$$L_{ij} = \begin{cases} 1, & \text{if } (1 + \tau_r^{ij}) \cdot (l_r(j) - l_r(i)) \geq t_{ij} \\ 0, & \text{otherwise} \end{cases} \quad (20)$$

where τ_r^{ij} represents the acceptable elongation ratio of the link $(i \rightarrow j)$ with respect to the origin r . It can be set to 1.6 for inter-urban studies or between 1.3 and 1.5 for urban studies (Leurent 1997; Tagliacozzo and Pirzio 1973). In summary, the proposed method can be extended to any other non-negative and link-additive conditions in the efficient path check criterion, as long as the generated subnetwork is acyclic. Essentially, the purpose of ‘efficient path check’ step is to construct an acyclic subnetwork through the efficient path definition, and then we calculate the number of simple paths in the subnetwork, which is equal to the number of efficient paths in the original network.

Remark 3.3: Table 2 compares the Bell counting and Dial counting methods. The main advantages of the Dial counting method are summarized as follows:

- (1) We pay attention to the computational complexity, since it significantly affects the computational efficiency of counting the number of efficient paths between O-D pairs. From the ‘forward pass’ step, the Dial counting method has a computational time-complexity of $O(|W| \cdot |A|)$. Using the Winnipeg network as an example, the computational time-complexity of the Dial counting method is $4345 \times 2535 = 1.10 \times 10^7$, which is 10^4 times lower than that of the Bell counting method (i.e. $135 \times 1067^3 = 1.64 \times 10^{11}$).
- (2) The Dial counting method calculates the number of efficient paths between the considered O-D pairs, rather than all node pairs as in the Bell counting method. In fact, the ‘forward pass’ step for each origin can also calculate the number of efficient paths between the origin and all other nodes as shown in Equation (12).
- (3) More importantly, this streamlined calculation does not affect counting the efficient paths using a link/node. Instead of the node-pair-level information used in the Bell counting method (i.e. $u_r(r, m_a)$ and $u_r(n_a, s)$ in Equation (9)), the Dial counting method only uses the link-level information (i.e. n_{ij}^{rs} in Equation (13)). This further guarantees the advantage of the proposed method in terms of both computational memory and efficiency.
- (4) As shown in Equation (15), the Dial counting method calculates the total travel cost of all efficient paths between an O-D pair in a simple manner, which can be considered as a by-product of the forward and backward passes. Hence, it is much more efficient than the calculation procedure of the Bell counting method in Equation (6) (i.e. requiring two full node-pair matrices of $U_r(|N_r|, |N_r|)$ and $W_r(|N_r|, |N_r|)$).

Remark 3.4: Search algorithms are fundamental graph techniques that attempt to find all the nodes in a network satisfying a particular property. Breadth-First Search (BFS) and Depth-First Search (DFS) are two typical strategies of search algorithms (Ahuja 1993). Both BFS and DFS are enumeration procedures in counting paths, which can obtain the path details between O-D pairs and the number of paths as a byproduct. With the path details, we can generate a route choice set for modeling route choice behaviors (Prato and Bekhor 2006; Prato 2009). Rich information (i.e. path details) come with the expense of efficiency and storage, but they are inevitable even we just need the number of paths. In contrast, the Dial counting method does not need to enumerate and store path details. Even though BFS and DFS can be used for search paths in a network, to our best knowledge, the formulations in Equations (11)–(12) have not been provided in the literature. The Dial counting method has a BFS tree structure, but it avoids multiple traversals of a node as in the BFS, increasing the traversal efficiency. In summary, compared with the BFS and DFS algorithms in counting paths, the Dial counting method avoids enumerating and storing path details, and traverses nodes more efficiently as it traverses each node only once to obtain the number of paths between O-D pairs. For more details, please refer to Appendix.

4. Numerical experiments

This section provides two sets of numerical experiments to verify the validity, efficiency and large-scale network applicability of the proposed method. The first experiment uses a nine-node grid network to demonstrate the detailed steps of the method. The second experiment adopts nine realistic transportation networks to examine the computational efficiency of the proposed method.

4.1. Nine-node grid network

We use the nine-node grid network shown in Figure 2 to illustrate the detailed steps of the Dial counting method. The network has 9 nodes and 12 links. The travel cost of each link shown in Figure 2(a) is modified from Sheffi (1985). We consider the O-D pair from node 1 to node 9 unless specified otherwise.

Following the procedure presented in Section 3, the initialization step (i.e. Step 1) calculates the minimum travel cost from origin node 1 to all the other nodes, as shown in Figure 2(a). Then, the efficient path check (i.e. Step 2) finds that link (3→6) does not satisfy Equation (10) because $I_r(3) = I_r(6) = 4$, while all the other links are efficient links with L_{ij} being equal to 1, as shown in Figure 2(b).

After that, the forward pass step (i.e. Step 3) can determine the number of efficient paths from origin node 1 to all the other nodes. Specifically, we calculate the link weight W_{ij} according to Equation (11) and then the number of efficient paths according to Equation (12). Their results are shown in Figure 2(c,d). For example, at node 5, both W_{56} and W_{58} are equal to the link weight summation of the upstream links of node 5, i.e. $W_{25} + W_{45} = 1 + 1 = 2$. The above summation term is also equal to the number of efficient paths from origin node 1 to node 5, i.e. $n^{r5} = W_{25} + W_{45} = 1 + 1 = 2$. Similarly, the number of efficient paths between O-D pair (1, 9) is equal to the link weight summation of the upstream links of node 9, i.e. $n^{r9} = W_{69} + W_{89} = 2 + 3 = 5$. This is consistent with the complete path enumeration. In this network, there are 6 paths between O-D pair (1, 9), while

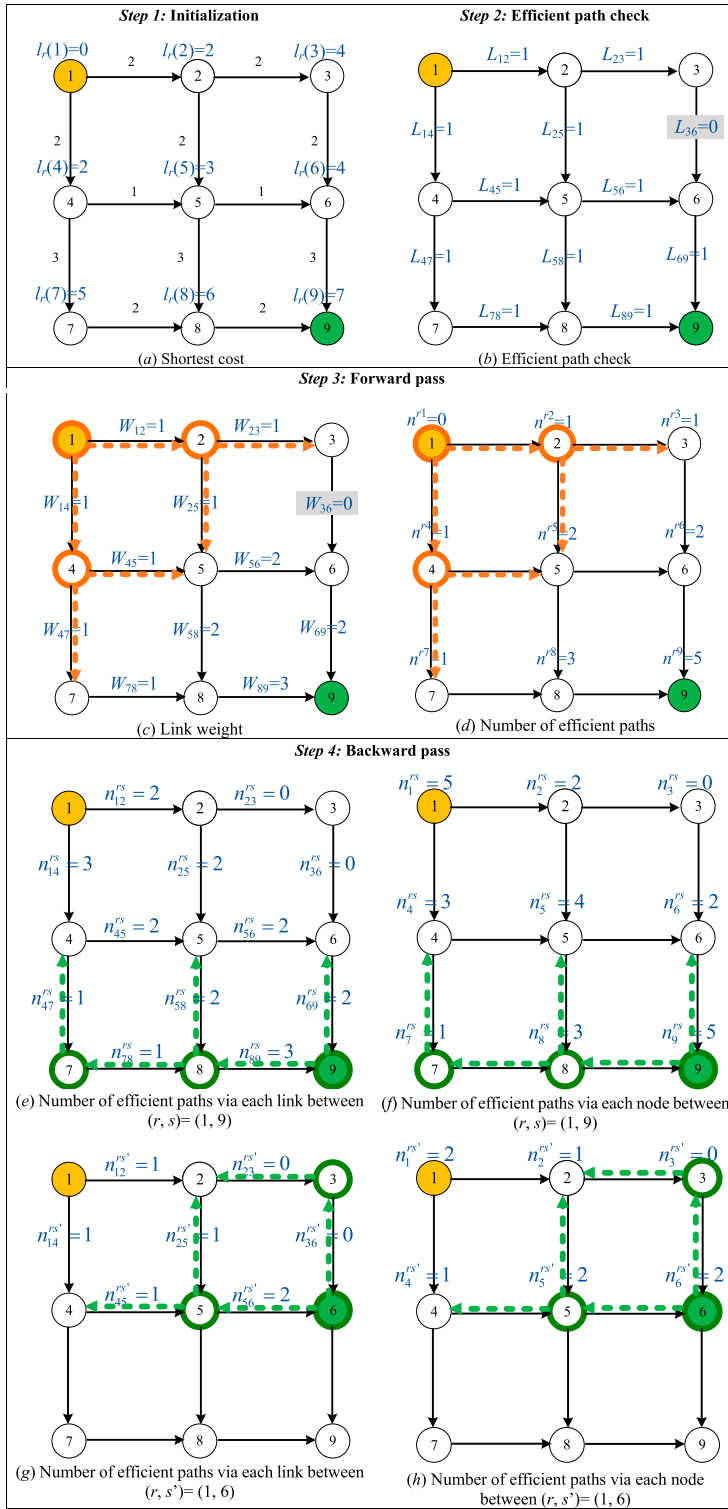








Figure 2. The detailed steps of the Dial counting method in the nine-node grid network.

Note: Origin node ; Destination node ; First three nodes in forward pass ; First three nodes in backward pass ; Forward pass direction ; Backward pass direction .

only 5 paths are efficient paths. Path 1–2–3–6–9 is not efficient as shown in Figure 2(b) (i.e. $L_{36} = 0$). In the forward pass, we adopt the search order of ‘further away from the origin’. Without loss of generality, Figure 2(c,d) only shows the first three traversed nodes (i.e. node 1, node 2, and node 4) and their related search directions in the forward pass. The above traversal process has formed a tree structure, which is consistent with the discussion in Remark 3.4 and Appendix.

The backward pass step (i.e. Step 4) determines the number of efficient paths between a given O-D pair using each link or node according to Equations (13) and (14). The search order of the backward pass (i.e. descending order of $l_r(i)$) is shown in Figure 2(e,f). For demonstration purposes, we only show the first three traversed nodes and their related search directions. For example, the number of efficient paths between O-D pair (1, 9) via node 5 (i.e. n_5^s) is equal to $n_{56}^s + n_{58}^s = 2 + 2 = 4$, these four efficient paths are split by the two upstream links of node 5 with equal link weight $W_{25} = W_{45} = 1$, and hence $n_{25}^s = n_{45}^s = 2$.

Note that the calculation results from Steps 1–3 (i.e. $l_r(i)$, L_{ij} , W_{ij} and n^i) correspond to any O-D pair originated from node 1, while the backward pass in Step 4 is conducted for a given O-D pair. Hence, if we want to obtain the number of efficient paths via each link/node between another O-D pair, we just need to rerun the backward pass for that particular O-D pair while inheriting the calculation results from Steps 1–3. For example, Figure 2(g,h) shows the backward pass results for O-D pair (1, 6). In other words, although the calculation procedure presented in Section 3 is for each O-D pair in the network, Steps 1–3 are performed for each origin node only once, which are shared by the O-D pair specific backward pass in Step 4.

In summary, the total number of efficient paths between the O-D pair and the number of efficient paths using each link/node obtained by the Dial counting method are the same as those obtained by the complete path enumeration. This verifies the correctness of the results.

In addition, the algorithm can accept different definitions of alternative paths for efficient path check criterion. When each link satisfies Equation (10), we will have the alternative path of ‘always further away from the origin’. When each link satisfies Equation (20), we will have the alternative path of ‘always further away from the origin and not too long’. When each link satisfies the condition $l_r(i) < l_r(j) - t_{ij} + \delta$ (δ is a small tolerance relative to t_{ij}), we will have the alternative path of ‘further away from the origin with a certain degree of tolerance’. As for the computational process, we only need to change the ‘Efficient path check’ step and leave the other steps unchanged. Figure 3 illustrates the computational process under different definitions of alternative path using the nine-node grid network. Note that we only show the results of the efficient path check and the resultant number of efficient paths, because the forward pass and backward pass steps are not affected by the different definitions.

- **Condition 1:** $l_r(i) < l_r(j)$. As shown in Figure 3(a), link (3→6) does not satisfy the definition because $l_r(3) = l_r(6) = 4$. Hence, the subnetwork under Condition 1 does not include link (3→6). There are 5 efficient paths between O-D pair (1, 9) (i.e. $n^9 = 5$ in Figure 3(b)).
- **Condition 2:** $(1 + \tau_r^j) \cdot (l_r(j) - l_r(i)) - t_{ij} \geq 0$, where $\tau_r^j = 1.4$. As shown in Figure 3(c), link (3→6) does not satisfy this definition, because $(1 + \tau_r^{36}) \times (l_r(6) - l_r(3)) - t_{36} = (1 + 1.4) \times (4 - 4) - 2 = -2 < 0$. Hence, the subnetwork under Condition 2 does not

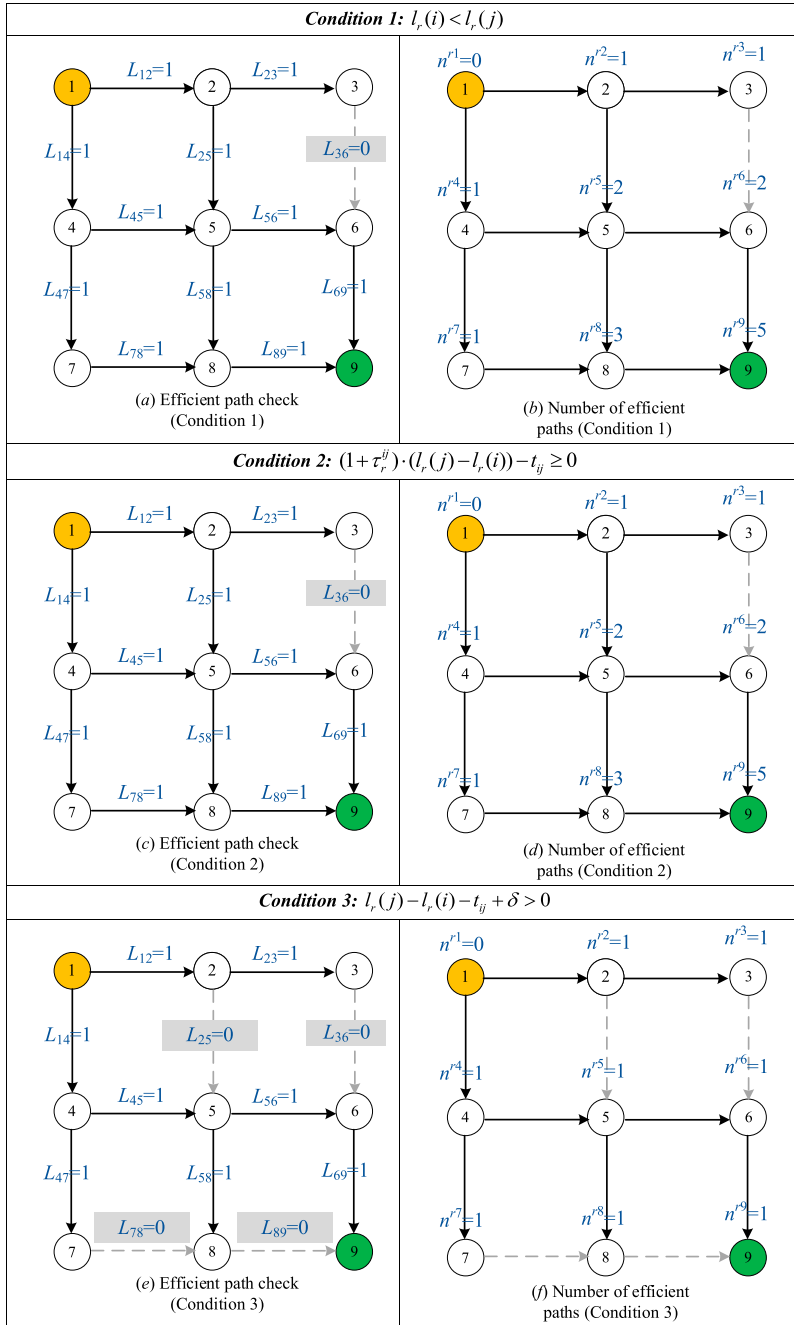


Figure 3. The detailed process of efficient path check and the number of efficient paths under three different conditions in the nine-node grid network.

include link (3→6), and there are also 5 efficient paths between O-D pair (1, 9) (i.e. $n^r9 = 5$ in Figure 3(d)).

- **Condition 3:** $l_r(i) - l_r(j) - t_{ij} + \delta > 0$, where $\delta = 0.5$. As shown in Figure 3(e), four links do not satisfy this condition. They are: link (2→5) because $l_r(5) - l_r(2)$

Table 3. Basic information of the nine large-scale test networks.

Network	Descriptions			
	Zones	Links	Nodes	O-D Pairs
Berlin-Friedrichshain	23	523	224	506
Berlin-Tiergarten	26	766	361	644
Berlin-Prenzlauerberg-Center	38	749	352	1406
Berlin-Mitte-Center	36	871	398	1260
Anaheim	38	914	416	1406
Winnipeg	147	2836	1052	4345
Barcelona	110	2522	1020	7922
Berlin-Mitte-Prenzlauerberg-Friedrichshain-Center	98	2184	975	9505
Chicago-Sketch	387	2950	933	93,135

– $t_{25} + 0.5 = 3 - 2 - 2 + 0.5 = -0.5 < 0$; link $(3 \rightarrow 6)$ because $l_r(6) - l_r(3) - t_{36} + 0.5 = 4 - 4 - 2 + 0.5 = -1.5 < 0$; link $(7 \rightarrow 8)$ because $l_r(8) - l_r(7) - t_{78} + 0.5 = 6 - 5 - 2 + 0.5 = -0.5 < 0$; and link $(8 \rightarrow 9)$ because $l_r(9) - l_r(8) - t_{89} + 0.5 = 7 - 6 - 2 + 0.5 = -0.5 < 0$. Hence, the subnetwork under Condition 3 does not include the above 4 links, and there is only 1 efficient path between O-D pair $(1, 9)$ (i.e. $n^{r9} = 1$ in Figure 3(f)).

Moreover, we can replicate the Nguyen-Dupuis network results (with 13 nodes, 19 links and 4 O-D pairs) as presented in Meng, Lee, and Cheu (2005) using the Bell counting method. The results include the number of efficient paths between an O-D pair, the number of efficient paths between an O-D pair passing through each link, and the total cost of all efficient paths between an O-D pair. To avoid repetition, we do not present the results herein.

4.2. Test in large-scale networks

This section uses nine well-known transportation networks shown in Table 3 to compare the computational efficiency between the Bell counting method and the Dial counting method. The input data of these networks are obtained from the website: <https://github.com/bstabler/TransportationNetworks>. The largest number of links is 2950 in the Chicago-Sketch network, the largest number of nodes is 1052 in the Winnipeg network, and the largest number of O-D pairs is 93,135 in the Chicago-Sketch network.

To enhance the behavioral realism, we consider the definition of effective path (i.e. not only efficient path, but also not-too-long path) as in Equation (20). In order to ensure a fair comparison, we calculate the three measures by both methods, and we use the same computer code for the common modules of the two methods, e.g. the shortest path module (via Dijkstra's method), file reading module and output module. Both methods are coded in Intel Visual Fortran.XE2013 and run on a Desktop PC with i7-7700 CPU 3.60GHZ and 32G RAM. Both methods generate the same results for each of these networks (see Table 4), but with significantly different computational performances (see Table 5).

Comparing the statistics of the number of effective paths between O-D pairs among different networks shown in Table 4, we can see:

- (1) The average number of effective paths varies greatly with the network size (e.g. measured by the number of nodes in a network). The minimum average number of effective

Table 4. Statistics of the number of effective paths in nine test networks.

Network	Statistics of the Number of Effective Paths						
	Total	Min	Max	Mean	Std.	Skewness	Kurtosis
Berlin-Friedrichshain	986	1	22	1.95	1.90	4.43	31.09
Berlin-Tiergarten	906	1	6	1.41	0.77	3.20	14.63
Berlin-Prenzlauerberg-Center	4017	1	60	2.86	4.40	7.72	86.00
Berlin-Mitte-Center	2558	1	12	2.03	1.72	2.74	8.51
Anaheim	4320	1	125	3.07	6.14	9.41	141.07
Winnipeg	50,522	1	634	11.63	31.68	9.06	115.87
Barcelona	1,016,677	1	69,988	128.34	1477.29	34.06	1391.48
Berlin-Mitte-Prenzlauerberg-Friedrichshain-Center	45,246	1	170	4.76	8.71	7.25	82.73
Chicago-Sketch	8,533,130	1	72,618	91.62	870.12	42.75	2570.70

paths is 1.41 in the Berlin-Tiergarten network, and the largest average number reaches 128.34 in the Barcelona network.

- (2) Each O-D pair contains at least one effective path. The maximum number of effective paths varies greatly in the nine test networks. For example, the largest number of effective paths between O-D pairs is 69,988 in the Barcelona network.
- (3) Without loss of generality, Figure 4 shows the frequency distribution of the number of effective paths between O-D pairs in the Winnipeg network. One can see that the distribution is highly right-skewed with a long distribution tail. Table 4 also presents the skewness and kurtosis for each network. Overall, all these networks have large positive skewness and kurtosis. This phenomenon implies that there are some spatial inequities among different O-D pairs in terms of the number of effective paths available to travelers. In the event of a disruption, this route diversity measure may have quite different impacts on different O-D pairs in a large-scale network. An O-D pair with fewer effective routes is more likely to become disconnected or become more congested due to the flow rerouting from a disconnected route to other alternative routes. An O-D pair with more effective routes has a lower chance to be affected by such disruptions.

Table 5 compares the computational time between the Dial counting and Bell counting methods. For each network, we calculate the three measures: the number of effective paths between O-D pairs, the number of effective paths between O-D pairs passing through each link, and the total cost of the effective paths between O-D pairs (the average cost can be calculated similarly). One can see that the computational time of the Dial counting method is much lower than that of the Bell counting method in all the nine test networks.

- (1) n^{fs} : In all test networks, the computational time of the Dial counting method is 2.81 to 32.95 times faster than that of the Bell counting method. The lowest and largest reductions in computational time appear in the Berlin-Friedrichshain network and the Winnipeg network, respectively. In the Winnipeg network, the proposed Dial counting method only needs 50.21s, which is 32.95 times lower than the Bell counting (i.e. 1654.03s). Even for the Chicago sketch network with more than 93,000 O-D pairs, the

Table 5. Computational performance of the two methods in the nine test networks.

Network	Computing time (s) – n^{rs}			Computing time (s) – n_a^{rs}			Computing time (s) – c^{rs}		
	Dial counting	Bell counting	Times	Dial counting	Bell counting	Times	Dial counting	Bell counting	Times
Berlin-Friedrichshain	0.59	1.67	2.81	0.59	1.67	2.81	0.61	4.25	6.97
Berlin-Tiergarten	1.23	8.52	6.90	1.23	8.54	6.96	1.23	20.53	16.63
Berlin-Prenzlauerberg-Center	2.28	11.56	5.07	2.28	11.56	5.07	2.31	30.10	13.02
Berlin-Mitte-Center	2.52	15.60	6.20	2.52	16.03	6.37	2.56	42.57	16.62
Anaheim	2.85	19.20	6.73	2.91	19.23	6.62	2.94	49.54	16.87
Winnipeg	50.21	1654.03	32.95	50.80	1688.10	33.23	53.64	5940.14	110.73
Barcelona	80.37	1178.62	14.66	80.82	1189.93	14.72	81.61	3786.15	46.40
Berlin-Mitte-Prenzlauerberg-Friedrichshain-Center	95.10	1022.05	10.75	96.07	1023.05	10.65	96.51	3363.94	34.86
Chicago-Sketch	757.84	2740.66	3.62	795.94	2756.28	3.46	802.36	9771.19	12.18

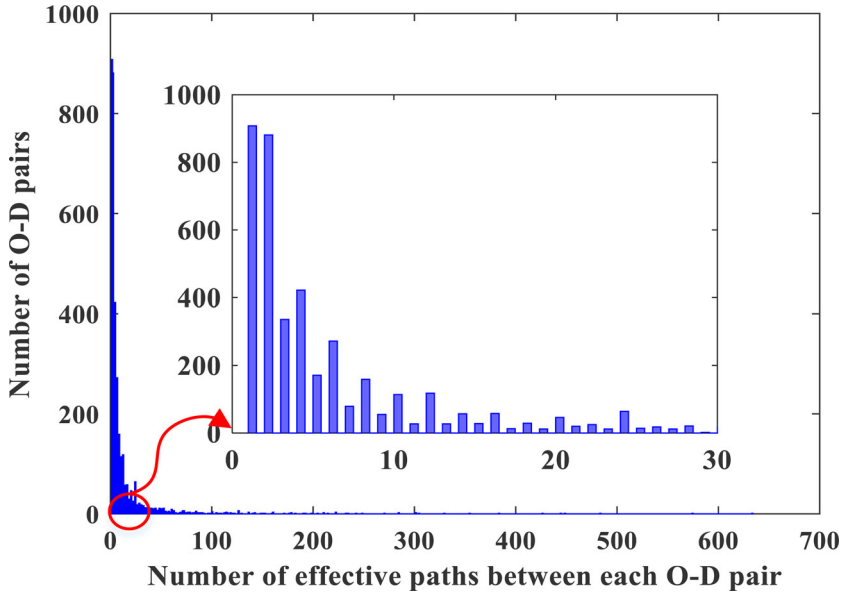


Figure 4. The frequency distribution of the number of effective paths in the Winnipeg network.

Dial counting method only needs 12.63 minutes (i.e. 757.84s), which is 3.62 times lower than the Bell counting method.

- (2) n_a^{rs} : The computational time of n_a^{rs} is generally a little longer than that of n^{rs} in each test network. That is because the calculation of n_a^{rs} in both methods is based on n^{rs} , where these additional steps do not take up a large proportion of the total computational time. The Dial counting method still shows a clear advantage, which has several times (i.e. from 2.81 to 33.23 times) of computational time reduction than the Bell counting method.
- (3) c^{rs} : The computational efficiency improvement of the Dial counting method in calculating c^{rs} is even more significant than n^{rs} . In the Dial counting, c^{rs} can be obtained by two ways as in Equation (15) or Equation (17), where their difference in computational time is trivial. Without loss of generality, Table 5 shows the less efficient results by Equation (15). Using the Winnipeg network as an example, the computational time of the Dial counting method for calculating c^{rs} is 110.73 times lower than that of the Bell counting method, while the computational time reduction for calculating n^{rs} is 32.95 times. This may be because c^{rs} in the Bell counting method requires the nested triangle operation on two full node-pair-level matrices (i.e. $w_r(m, j)$ and $u_r(j, n)$ in Equation (6)), and it is much more expensive than n^{rs} (as in Equation (5)). In contrast, the Dial counting method has less difference in computational time between the above two measures.

Below we explore the relationship between computational time and network size. In order to eliminate the influence of the number of O-D pairs among different networks, we calculate the average computational time of each O-D pair as shown in Figure 5.

- (1) With the increase of network size, the average computational time of each O-D pair of the three measures is all strictly increasing for both methods. Since the computational

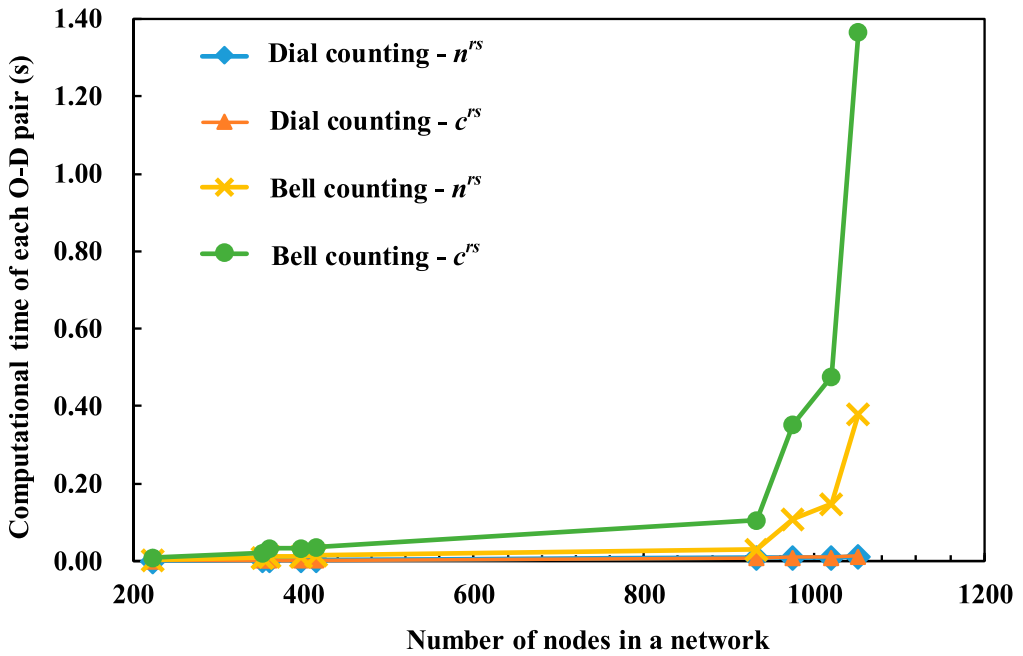


Figure 5. Relationship between average computational time and network size.

time curves of n^{rs} and n_a^{rs} are almost identical, we only show n^{rs} curves from the two methods in Figure 5.

- (2) The computational time required by the Dial counting method increases slowly, while the computational time required by the Bell counting method increases more sharply, especially when the number of nodes exceeds 800. This indicates that the computational advantage of the Dial counting method is more significant in large-scale network applications. This also confirms the conclusion obtained from the above theoretical time-complexity analysis.
- (3) The average computational time of the Bell counting method increases exponentially while the Dial counting method increases linearly with the network size. The R-squares of the linear regression fitting in the Dial counting method are respectively 0.978 (n^{rs}), 0.982 (n_a^{rs}), and 0.975 (c^{rs}) for the three measures. The R-squares of the exponential fitting in the Bell counting method are respectively 0.972 (n^{rs}), 0.972 (n_a^{rs}), and 0.971 (c^{rs}). This growth trend can be explained by the above time-complexity analysis of the Dial counting method ($O(|W| \cdot |A|)$) and Bell counting method ($O(|R| \cdot |N|^3)$).

5. Concluding remarks

In this paper, we developed an alternative method of counting the number of efficient paths in a transportation network. The existing Bell counting method is able to count the number of efficient paths and the total cost of the efficient paths between any two nodes without the need to enumerate path. However, the nested triangle operation for any three nodes incurs a high computational time complexity $O(|N|^3)$ for each origin node. This procedure requires a lot of computations for unnecessary outputs, and these calculations cannot be avoided in the Bell counting method. This high

computational effort can significantly hinder its use and further applications in large-scale networks.

Inspired by the Dial loading method for the Logit model, this paper proposed a Dial counting method as a more computationally efficient alternative to counting the number of efficient paths between each O-D pair, the number of efficient paths using each link/node between each O-D pair, and the total/average cost of these efficient paths between each O-D pair. The computational efficiency of the proposed method was due to the much lower time-complexity of $O(|A|)$ for each O-D pair. We have rigorously proved the correctness of the proposed method for counting the number of efficient paths. Relative to the classical BFS and DFS algorithms, the proposed method does not need to enumerate and store path details in counting paths, and it has a more efficient traversal order as it traverses each node only once to obtain the number of paths between O-D pairs.

Numerical examples using a set of networks with different sizes, ranging from 500 O-D pairs to 100,000 O-D pairs were then provided to demonstrate the validity, efficiency and large-scale network applicability of the proposed method.

- (1) In all test networks, the computational time of the number of efficient paths (n^{rs}) of the Dial counting method is around 3 to 30 times faster than that of the Bell counting method. The computational time of the number of efficient paths using each link/node (n_a^{rs}) is generally similar to that of n^{rs} , but the computational efficiency improvement of the total/average cost of these efficient paths (c^{rs}) is even more significant than n^{rs} .
- (2) With the increase of network size, the average computational time of each O-D pair required by the Dial counting method increases linearly, while the computational time required by the Bell counting method increases exponentially, especially in large-scale transportation networks.
- (3) The frequency distribution of the number of effective paths between O-D pairs are highly right-skewed with a long distribution tail. This phenomenon implies that there are some spatial inequities among different O-D pairs in terms of the route diversity.

For future research, we will apply the proposed method to other travel modes of transportation systems or other fields of non-transportation network analysis (e.g. communication network as in Brumbaugh-Smith and Shier (2002)). Also, it is interesting to explore the possibility of embedding a zone-to-zone shortest path algorithm (e.g. Wang, Johnson, and Sokol 2005) in counting the number of efficient paths for each O-D pair. Note that different efficient routes between an O-D pair may share some common links, and the disruption of one common link may significantly reduce the number of efficient routes. Hence, it is meaningful to count the number of efficient paths with route overlapping consideration. Furthermore, parallel line (or common line) is widespread in a public transport network but it is nontrivial to deal with. In the future study, we plan to develop a method to calculate the number of efficient paths between O-D pairs in public transport networks and multi-modal networks.

Acknowledgements

Conceptualization, Methodology, Writing – Original draft preparation done by Rong Zhao; Conceptualization, Methodology, Writing – Reviewing and Editing done by Xiangdong Xu and Anthony Chen.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Funding

This study is supported by the National Key Research and Development Program of China [No. 2018YFB1600900], the National Natural Science Foundation of China [71971159; 71890973; 72021002; 72071174], and the Fundamental Research Funds for the Central Universities. These supports are gratefully acknowledged.

ORCID

Anthony Chen  <http://orcid.org/0000-0003-4363-5041>

References

- Ahuja, R. 1993. *Network Flows: Theory, Algorithms, and Applications*. Englewood Cliffs, NJ: Prentice Hall.
- Ball, M. O. 1986. "Computational Complexity of Network Reliability Analysis: An Overview." *IEEE Transactions on Reliability* 35 (3): 230–239.
- Ball, T., and J. R. Larus. 1997. "Efficient Path Profiling." *Proceedings of the 29th Annual International Symposium on Microarchitecture*, 46–57. Paris, France.
- Ball, M. O., and J. S. Provan. 1983. "Calculating Bounds on Reachability and Connectedness in Stochastic Networks." *Networks* 13 (2): 253–278.
- Bartholdi, L. 1999. "Counting Paths in Graphs." *Enseignement Mathématique* 45 (1–2): 83–131.
- Bell, M. G. H. 1995. "Alternatives to Dial's Logit Assignment Algorithm." *Transportation Research Part B* 29 (4): 287–295.
- Bianco, L., G. Confessore, and P. Reverberi. 2001. "A Network Based Model for Traffic Sensor Location with Implication in O-D Matrix Estimates." *Transportation Science* 35 (1): 50–60.
- Boer, C., M. Snelder, R. Van Nes, and B. Van Arem. 2017. "The Impact of Route Guidance, Departure Time Advice and Alternative Routes on Door-to-Door Travel Time Reliability: Two Data-Driven Assessment Methods." *Journal of Intelligent Transportation Systems* 21 (6): 465–477.
- Brumbaugh-Smith, J. P., and D. R. Shier. 2002. "Minimax Models for Diverse Routing." *INFORMS Journal on Computing* 14 (1): 81–95.
- Cartwright, D., and T. C. Gleason. 1966. "The Number of Paths and Cycles in a Digraph." *Psychometrika* 31 (2): 179–199.
- Castillo, E., Z. Grande, A. Calviño, W. Y. Szeto, and H. K. Lo. 2015. "A State-of-the-Art Review of the Sensor Location, Flow Observability, Estimation, and Prediction Problems in Traffic Networks." *Journal of Sensors* 2015: 1–26.
- Chan, H. Y., A. Chen, G. Li, X. Xu, and W. H. K. Lam. 2021. "Measuring Route Diversity for Urban Rail Transit Networks." *Journal of Transport Geography* 91, 102945.
- Chen, A., P. Chootinan, and S. Pravinongvuth. 2004. "Multiobjective Model for Locating Automatic Vehicle Identification Readers." *Transportation Research Record* 1886 (1): 49–58.
- Chen, A., S. Pravinongvuth, and P. Chootinan. 2010. "Scenario-Based Multiobjective AVI Reader Location Models Under Different Travel Demand Patterns." *Transportmetrica* 6 (1): 53–78.
- Chen, A., S. Pravinongvuth, P. Chootinan, M. Lee, and W. Recker. 2007. "Strategies for Selecting Additional Traffic Counts for Improving O-D Trip Table Estimation." *Transportmetrica* 3 (3): 191–211.
- Chen, J., Z. Zang, X. Xu, and C. Yang. 2018. "Identifying Critical Links in Transportation Networks Based on Route Diversity Redundancy." *Proceedings of the 17th COTA International Conference of Transportation Professionals*, Shanghai, China, 3965–3977.
- Chootinan, P., A. Chen, and H. Yang. 2005. "A Bi-Objective Traffic Counting Location Problem for Origin-Destination Trip Table Estimation." *Transportmetrica* 1 (1): 65–80.

- Cormen, T. H., C. E. Leiserson, and R. L. Rivest. 2009. *Introduction to Algorithms*. Cambridge, MA: MIT Press.
- Crawford, F., D. P. W. Watling, and R. D. Connors. 2018. "Identifying Road User Classes Based on Repeated Trip Behaviour Using Bluetooth Data." *Transportation Research Part A* 113: 55–74.
- Dial, R. B. 1971. "A Probabilistic Multipath Traffic Assignment Model Which Obviates Path Enumeration." *Transportation Research* 5 (2): 83–111.
- Dias, F. C., M. Campêlo, C. Souza, and R. Andrade. 2017. "Min-Degree Constrained Minimum Spanning Tree Problem with Fixed Centrals and Terminals: Complexity, Properties and Formulations." *Computers & Operations Research* 84: 46–61.
- Floyd, R. W. 1962. "Algorithm 97: Shortest Path." *Communications of the ACM* 5 (6): 345.
- Freeman, L. C., S. P. Borgatti, and D. R. White. 1991. "Centrality in Valued Graphs: A Measure of Betweenness Based on Network Flow." *Social Networks* 13 (2): 141–154.
- Gentili, N., and P. B. Mirchandani. 2012. "Locating Sensors on Traffic Networks: Models, Challenges and Research Opportunities." *Transportation Research Part C* 24: 227–255.
- Gessel, I. 1993. "Counting Paths in Young's Lattice." *Journal of Statistical Planning and Inference* 34 (1): 125–134.
- Havlin, S., and R. Cohen. 2010. *Complex Networks: Structure, Robustness, Function*. Cambridge: Cambridge University Press.
- He, D., A. N. Arslan, Y. He, and X. Wu. 2007. "Iterative Refinement of Repeat Sequence Specification using Constrained Pattern Matching." *Proceedings of the 2007 IEEE 7th International Symposium on BioInformatics and BioEngineering*, Boston, MA, 1199–1203.
- Hochberg, R. 2012. *Matrix Multiplication with CUDA – a Basic Introduction to the CUDA Programming Model*. SHODOR Technical Document.
- Huang, H. J., and M. G. H. Bell. 1998. "A Study on Logit Assignment Which Excludes All Cyclic Flows." *Transportation Research Part B* 32 (6): 401–412.
- Hung, M. S., and J. J. Divoky. 1988. "A Computational Study of Efficient Shortest Path Algorithms." *Computers & Operations Research* 15 (6): 567–576.
- Ip, W. H., and D. Wang. 2009. "Resilience Evaluation Approach of Transportation Networks." *Proceedings of the International Joint Conference on Computational Sciences and Optimization*, Sanya, China, 681–622.
- Jing, W., X. Xu, and Y. Pu. 2019. "Route Redundancy-Based Network Topology Measure of Metro Networks." *Journal of Advanced Transportation* 2019: 1–12.
- Jing, W., X. Xu, and Y. Pu. 2020. "Route Redundancy-Based Approach to Identify the Critical Stations in Metro Networks: A Mean-Excess Probability Measure." *Reliability Engineering & System Safety* 204: 107204.
- Kim, J., and H. S. Mahmassani. 2015. "Spatial and Temporal Characterization of Travel Patterns in a Traffic Network Using Vehicle Trajectories." *Transportation Research Part C* 59: 375–390.
- Kirchhoff, G. 1958. "On the Solution of the Equations Obtained from the Investigation of the Linear Distribution of Galvanic Currents." *IRE Transactions on Circuit Theory* 5 (1): 4–7.
- Kurauchi, F., N. Uno, A. Sumalee, and Y. Seto. 2009. "Network Evaluation Based on Connectivity Vulnerability." *Proceedings of the 18th International Symposium on Transportation and Traffic Theory*, Boston, MA, 637–649.
- Lam, W. H. K., and K. S. Chan. 1998. "A Link-Based Alternative to Bell's Logit Assignment Algorithm." *HKIE Transactions* 5 (2): 11–18.
- Larsson, T., J. T. Lundgren, and A. Peterson. 2010. "Allocation of Link Flow Detectors for Origin-Destination Matrix Estimation – a Comparative Study." *Computer-Aided Civil and Infrastructure Engineering* 25 (2): 116–131.
- Leurent, F. M. 1997. "Curbing the Computational Difficulty of the Logit Equilibrium Assignment Model." *Transportation Research Part B* 31 (4): 315–326.
- Li, Y., L. Sun, H. Z. Zhu, and Y. X. Wu. 2012. "A Nettee for Simple Paths with Length Constraint and the Longest Path in Directed Acyclic Graphs." *Chinese Journal of Computers* 35 (10): 2194.
- Liu, F., G. Qi, and H. Che. 2006. "Research on Algorithm for Detecting Simple Path in Complex Network and Its Application." *System Engineering Theory and Practices* 26 (4): 9–13.

- Meng, Q., D. H. Lee, and R. L. Cheu. 2005. "Counting the Different Efficient Paths for Transportation Networks and Its Applications." *Journal of Advanced Transportation* 39 (2): 193–220.
- Miller-Hooks, E., X. Zhang, and R. Faturechi. 2012. "Measuring and Maximizing Resilience of Freight Transportation Networks." *Computers & Operations Research* 39 (7): 1633–1643.
- Prato, C. G. 2009. "Route Choice Modeling: Past, Present and Future Research Directions." *Journal of Choice Modelling* 2 (1): 65–100.
- Prato, C. G., and S. Bekhor. 2006. "Applying Branch & Bound Technique to Route Choice set Generation." *Transportation Research Record* 1985 (1): 19–28.
- Roberts, B., and D. P. Kroese. 2007. "Estimating the Number of s-t Paths in a Graph." *Journal of Graph Algorithms & Applications* 11 (1): 195–214.
- Ross, I. C., and F. Harary. 1952. "On the Determination of Redundancies in Sociometric Chains." *Psychometrika* 17 (2): 195–208.
- Sanjeev, A., and B. Boaz. 2009. *Computational Complexity: A Modern Approach*. Cambridge: Cambridge University Press.
- Schäfer, L., S. García, A. Mitschke, and V. Srithammavanh. 2018. "Redundancy System Design for an Aircraft Door Management System." *Computers & Operations Research* 94: 11–22.
- Sheffi, Y. 1985. *Urban Transportation Networks: Equilibrium Analysis with Mathematical Programming Methods*. Englewood Cliffs, NJ: Prentice Hall.
- Si, B., M. Zhong, H. Zhang, and W. Jin. 2010. "An Improved Dial's Algorithm for Logit-Based Traffic Assignment Within a Directed Acyclic Network." *Transportation Planning and Technology* 33 (2): 123–137.
- Stanley, R. P. 1996. "A Matrix for Counting Paths in Acyclic Digraphs." *Journal of Combinatorial Theory Series A* 74 (1): 169–172.
- Tagliacozzo, F., and F. Pirzio. 1973. "Assignment Models and Urban Path Selection Criteria: Results of a Survey of the Behaviour of Road Users." *Transportation Research* 7 (3): 313–329.
- Tong, C. 1990. "Modification of Dial's Algorithm by Redefining Path Efficiency." *Traffic Engineering & Control* 31 (9): 483–486.
- Valiant, L. G. 1979a. "The Complexity of Computing the Permanent." *Theoretical Computer Science* 8 (2): 189–201.
- Valiant, Leslie G. 1979b. "The Complexity of Enumeration and Reliability Problems." *SIAM Journal on Computing* 8 (3): 410–421.
- Wang, I., E. L. Johnson, and J. S. Sokol. 2005. "A Multiple Pairs Shortest Path Algorithm." *Transportation Science* 39 (4): 465–476.
- Wiener, H. 1947. "Structural Determination of Paraffin Boiling Points." *Journal of the American Chemical Society* 69 (1): 17–20.
- Wong, S. C. 1999. "On the Convergence of Bell's Logit Assignment Formulation." *Transportation Research Part B* 33 (8): 609–616.
- Wu, Y., X. Wu, F. Min, and Y. Li. 2010. "A Nettee for Pattern Matching with Flexible Wildcard Constraints." *Proceedings of the IEEE International Conference on Information Reuse & Integration*, Las Vegas, ND, 109–114.
- Xu, X., A. Chen, S. Jansuwan, C. Yang, and S. Ryu. 2018. "Transportation Network Redundancy: Complementary Measures and Computational Methods." *Transportation Research Part B* 114: 68–85.
- Xu, X., A. Chen, and C. Yang. 2018. "An Optimization Approach for Deriving Upper and Lower Bounds of Transportation Network Vulnerability Under Simultaneous Disruptions of Multiple Links." *Transportation Research Part C* 94: 338–353.
- Xu, X., H. K. Lo, A. Chen, and E. Castillo. 2016. "Robust Network Sensor Location for Complete Link Flow Observability Under Uncertainty." *Transportation Research Part B* 88: 1–20.
- Yang, X., A. Chen, B. Ning, and T. Tang. 2017. "Measuring Route Diversity for Urban Rail Transit Networks: A Case Study of the Beijing Metro Network." *IEEE Transactions on Intelligent Transportation Systems* 18 (2): 259–268.
- Young, C., and M. D. Smith. 1999. "Static Correlated Branch Prediction." *ACM Transactions on Programming Languages and Systems* 21 (5): 1028–1075.

- Zhou, C., X. Chu, and C. Nie. 2007. "Predicting Thermodynamic Properties with a Novel Semiempirical Topological Descriptor and Path Numbers." *The Journal of Physical Chemistry B* 111 (34): 10174–10179.
- Zhou, X., and G. List. 2010. "An Information-Theoretic Sensor Location Model for Traffic Origin-Destination Demand Estimation Applications." *Transportation Science* 44 (2): 254–273.

Appendix: Discussion on the Relationship with BFS and DFS

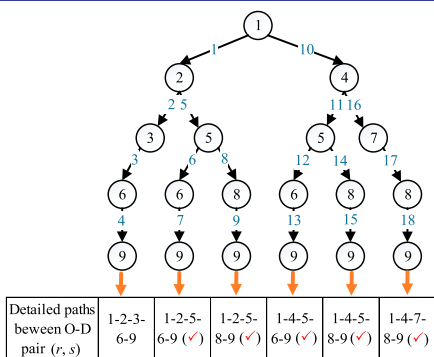
Below we discuss the relationship between the two search algorithms and the Dial counting method.

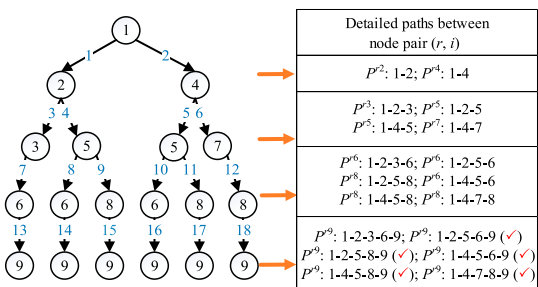
- (1) BFS maintains the nodes LIST as a *queue* and selects the labeled nodes in a first-in, first-out order (Ahuja 1993; Cormen, Leiserson, and Rivest 2009). If we define the number of links from origin node r to node i as the distance of node i , BFS first labels nodes with distance 1 (layer 1), and then labels those with distance 2 (layer 2), and so on.
- (2) DFS maintains the nodes LIST as a *stack* and selects the marked node in a last-in, first-out order (Ahuja 1993; Cormen, Leiserson, and Rivest 2009). This algorithm performs a deep probe to create a path as long as possible, and then backtracks to the last explored node and resumes the search from that node.

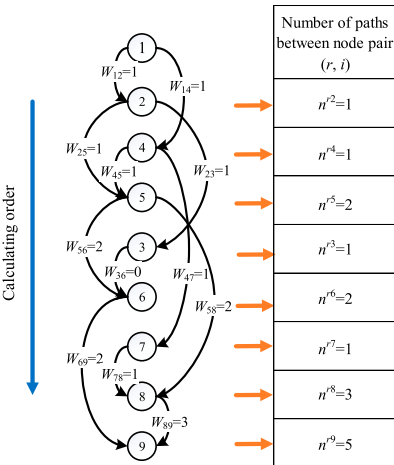
The nine-node grid network is used in the illustration as shown in Table A1.

- (1) As shown in Table A1, starting from the origin node 1, DFS attempts to generate a path as long as possible (i.e. 1–2–3–6–9). The blue numbers on the arrows indicate the traversal order. When the destination node 9 is reached, we get a detailed path and then the search backtracks to another branch (i.e. link (2→5)). The stored path details (i.e. the first path 1–2–3–6–9) are used to determine the new search (i.e. the new branch link (2→5)) after backtracking, and then we can identify whether the search could generate a new path.
- (2) As shown in Table A1, BFS traverses the network in a layer-based structure: starting from the origin node 1 (layer 0), BFS first traverses all the downstream nodes of the origin r (i.e. node 2 and node 4), adds these nodes into the next layer (layer 1) and then generates a simple path from origin r to these nodes (i.e. P^{r2} : 1–2 and P^{r4} : 1–4); next, BFS traverses the downstream nodes of the nodes in layer 1 (i.e. nodes 3 and 5 of previous node 2; and nodes 5 and 7 of previous node 4), adds these nodes into layer 2 and generates new simple paths. Note that P^{ri} denotes the simple path set between node pair (r, i) and may include more than one path. As discussed above, a node may be traversed multiple times in the BFS procedure since it may belong to different layers. Moreover, generating new paths in a layer depends on the path information stored in the previous layers. Hence, storing path details is also inevitable in the BFS counting method.
- (3) The Dial counting method has a BFS tree structure (as in Equation (11), link weight W_{ij} is calculated for all downstream links of node i), but each node is treated as a layer. This operation avoids multiple traversals of a node as in the BFS, increasing the traversal efficiency. Our search order is defined as 'further away from the origin' instead of just 'first-in, first-out'. This search order guarantees that we only need to traverse each node once to obtain the number of paths between O-D pairs correctly. As shown in Table A1, the BFS traverses nodes as 1–2–4–5–3–7–6–8–9, while the Dial counting method traverses nodes as 1–2–4–5–3–6–7–8–9. The Dial counting method searches 1–2–4–5–3 in the same way as in the BFS, but the next traversed node is node 6 rather than node 7 as in the BFS. Despite that the two methods share similar search order in the illustrative network, there are substantial differences when searching in large-scale networks. Also note that the number of paths from an origin node to each traversed node is obtained as a by-product in the Dial counting method.

Table A1. Comparison of BFS, DFS and Dial counting method.

Method	Algorithm procedure	Example							
Depth-First Search	<p>Initialization: node i = origin node r.</p> <p>DFS(i): label node i;for \forall link($i \rightarrow j$) $\in A$ if j is not labelled if $j \neq s$ DFS(j); else output a new path between O-D pair (r, s); else continue; unlabel node i.</p>	 <table> <tr> <th>Detailed paths between O-D pair (r, s)</th><td>1-2-3-6-9</td><td>1-2-5-6-9 (✓)</td><td>1-2-5-8-9 (✓)</td><td>1-4-5-6-9 (✓)</td><td>1-4-5-8-9 (✓)</td><td>1-4-7-8-9 (✓)</td></tr> </table>	Detailed paths between O-D pair (r, s)	1-2-3-6-9	1-2-5-6-9 (✓)	1-2-5-8-9 (✓)	1-4-5-6-9 (✓)	1-4-5-8-9 (✓)	1-4-7-8-9 (✓)
Detailed paths between O-D pair (r, s)	1-2-3-6-9	1-2-5-6-9 (✓)	1-2-5-8-9 (✓)	1-4-5-6-9 (✓)	1-4-5-8-9 (✓)	1-4-7-8-9 (✓)			

Breadth-First Search	<p>Initialization: layer = layer 0, node i = origin node r.</p> <p>BFS(i): for \forall node $j \in$ layer for \forall link($i \rightarrow j$) $\in A$ add node j into path between node pair (r, i), output a new path between node pair (r, j); add node j into layer+1;layer = layer+1.</p>	 <table> <tr> <th>Detailed paths between node pair (r, i)</th></tr> <tr> <td>P^{r2}: 1-2; P^{r4}: 1-4</td></tr> <tr> <td>P^{r3}: 1-2-3; P^{r5}: 1-2-5 P^{r6}: 1-4-5; P^{r7}: 1-4-7</td></tr> <tr> <td>P^{r6}: 1-2-3-6; P^{r6}: 1-2-5-6 P^{r8}: 1-2-5-8; P^{r6}: 1-4-5-6 P^{r8}: 1-4-5-8; P^{r8}: 1-4-7-8</td></tr> <tr> <td>P^{r9}: 1-2-3-6-9; P^{r9}: 1-2-5-6-9 (✓) P^{r9}: 1-2-5-8-9 (✓); P^{r9}: 1-4-5-6-9 (✓) P^{r9}: 1-4-5-8-9 (✓); P^{r9}: 1-4-7-8-9 (✓)</td></tr> </table>	Detailed paths between node pair (r, i)	P^{r2} : 1-2; P^{r4} : 1-4	P^{r3} : 1-2-3; P^{r5} : 1-2-5 P^{r6} : 1-4-5; P^{r7} : 1-4-7	P^{r6} : 1-2-3-6; P^{r6} : 1-2-5-6 P^{r8} : 1-2-5-8; P^{r6} : 1-4-5-6 P^{r8} : 1-4-5-8; P^{r8} : 1-4-7-8	P^{r9} : 1-2-3-6-9; P^{r9} : 1-2-5-6-9 (✓) P^{r9} : 1-2-5-8-9 (✓); P^{r9} : 1-4-5-6-9 (✓) P^{r9} : 1-4-5-8-9 (✓); P^{r9} : 1-4-7-8-9 (✓)
Detailed paths between node pair (r, i)							
P^{r2} : 1-2; P^{r4} : 1-4							
P^{r3} : 1-2-3; P^{r5} : 1-2-5 P^{r6} : 1-4-5; P^{r7} : 1-4-7							
P^{r6} : 1-2-3-6; P^{r6} : 1-2-5-6 P^{r8} : 1-2-5-8; P^{r6} : 1-4-5-6 P^{r8} : 1-4-5-8; P^{r8} : 1-4-7-8							
P^{r9} : 1-2-3-6-9; P^{r9} : 1-2-5-6-9 (✓) P^{r9} : 1-2-5-8-9 (✓); P^{r9} : 1-4-5-6-9 (✓) P^{r9} : 1-4-5-8-9 (✓); P^{r9} : 1-4-7-8-9 (✓)							

DialCounting Method	<p>Initialization: minimum travel cost $l_r(i)$.</p> <p>Efficient path check: $L_{ij} = \begin{cases} 1, & \text{if } l_r(i) < l_r(j) \\ 0, & \text{otherwise} \end{cases}$</p> <p>Forward pass: consider nodes in ascending order of $l_r(i)$ $W_{ij} = \begin{cases} L_{ij}, & \text{if } i = r \\ L_{ij} \cdot \sum_{m \in I_i} W_{mi}, & \text{otherwise} \end{cases}$ $n^{ri} = \begin{cases} 0, & \text{if } i = r \\ \sum_{m \in I_i} W_{mi}, & \text{otherwise} \end{cases}$</p>	 <table> <tr> <th>Number of paths between node pair (r, i)</th></tr> <tr> <td>$n^2=1$</td></tr> <tr> <td>$n^4=1$</td></tr> <tr> <td>$n^5=2$</td></tr> <tr> <td>$n^3=1$</td></tr> <tr> <td>$n^6=2$</td></tr> <tr> <td>$n^7=1$</td></tr> <tr> <td>$n^8=3$</td></tr> <tr> <td>$n^9=5$</td></tr> </table>	Number of paths between node pair (r, i)	$n^2=1$	$n^4=1$	$n^5=2$	$n^3=1$	$n^6=2$	$n^7=1$	$n^8=3$	$n^9=5$
Number of paths between node pair (r, i)											
$n^2=1$											
$n^4=1$											
$n^5=2$											
$n^3=1$											
$n^6=2$											
$n^7=1$											
$n^8=3$											
$n^9=5$											

Notes: (1) Red check marks indicate the efficient paths among all simple paths generated by DFS/BFS; (2) In the examples of BFS/DFS, the blue numbers on the arrows indicate the traversal order; and (3) In the example of the Dial counting method, left arrow in blue indicates the traversal order.