Origin-Destination Matrix Estimation Using Bush-Based User Equilibrium Algorithms

František Kolovský[1(✉)] and Ivana Kolingerová[2]

[1] Department of Geomatics, University of West Bohemia, Univerzitní 2732/8, 301 00 Pilsen, Czech Republic
kolovsky@kgm.zcu.cz

[2] Department of Computer Science and Engineering, University of West Bohemia, Univerzitní 2732/8, 301 00 Pilsen, Czech Republic
kolinger@kiv.zcu.cz

Abstract. This paper deals with origin-destination matrix estimation using traffic counts where the traffic assignment subproblem is solved using modern bush-based algorithms that are very powerful. The proposed algorithm for the origin-destination matrix estimation uses the well-known advantages of bushes (the origin-rooted acyclic subgraph) so that the algorithm does not need to enumerate the paths between origins and destinations. The proposed approach saves 5%–17% of the computation time compared to the methods that enumerate the paths and also is less memory demanding. The algorithm especially speeds up the origin-destination matrix estimation on high congested road networks.

Keywords: Origin-destination matrix estimation · Traffic assignment problem · Maximum entropy user equilibrium · Bush-based solution · Assignment matrix

1 Introduction

The origin-destination matrix (ODM) estimation (also called calibration) is a very important step in transportation modeling. Unfortunately, it is also a very time-consuming step. There is a lot of methods how to estimate the matrix, e.g., using speed data, traffic counts, and partial path data (license plates). In this paper, we deal with ODM estimation using traffic counts. The input is observed traffic flow on selected links and an initial ODM. The output is the calibrated matrix.

The work was supported by TRAFFO: Innovative Approaches to Mathematical Traffic Modelling for Sustainable Development of Cities and Regions. The Technology Agency of the Czech Republic. Programme: DOPRAVA 2020+. Grant agreement no: CK01000096 and by project SGS-2019-015 ("Application of Mathematics and Informatics in Geomatics IV").

© Springer Nature Switzerland AG 2021
O. Gervasi et al. (Eds.): ICCSA 2021, LNCS 12952, pp. 285–294, 2021.
https://doi.org/10.1007/978-3-030-86973-1_20

After the year 2000, the bush-based algorithms for solving user equilibrium (UE) came. These algorithms provide the equilibrated flow on acyclic subgraphs called bushes so these algorithms compute the UE implicitly without path enumeration and they seem the most powerful class of algorithms for solving UE [13].

In the classic formulations of ODM estimation, the flows on paths between origin-destination pairs are needed to compute the search direction during the matrix estimation. It is well-known that the solution of UE in the path space is not unique and the maximum entropy UE (MEUE) should be computed [19]. The best-known method for solving MEUE is also bush-based and computes the solution without the path enumeration.

Our proposed method builds on bush-based methods and estimates the ODM also without path-enumeration and thus saves the memory and computational time.

In Sect. 2 there are definitions of the problem and all subproblems with a literature survey. Section 3 introduces the proposed method for the determination of the assignment matrix. The numerical tests are described in Sect. 4.

2 Problem Definition and State-of-the-Art

The task is to estimate (calibrate) the origin-destination matrix using the traffic counts. The inputs are non-calibrated (target or initial) ODM and traffic counts on selected edges in the road network.

Let $G = (N, A)$ be a directed graph that represents the road network, where N is a set of nodes and A is a set of edges. The set of zones $Z \subset N$ contains all nodes where the vehicles enter/leave the road network. The $g = \{g_{ij} : ij \in W\}$ is the origin-destination matrix (ODM) containing the number of trips between all zones, where W is a set of all origin-destination (OD) pairs. The g_{ij} is the number of trips between the origin zone $i \in Z$ and the destination zone $j \in Z$. The $Q = \{g : g_{ij} \geq 0\}$ is a set of all feasible ODMs.

2.1 Origin-Destination Matrix Estimation

Let $\widehat{v} = (\widehat{v_a} : a \in \widehat{A})$ be a vector of observed traffic flows where $\widehat{A} \subset A$ is a set of edges where the traffic flow was measured. According to [10] the origin-destination matrix estimation problem is defined as bi-level optimization problem as

$$\min_g F(g) = \gamma_1 F_1(g, \widehat{g}) + \gamma_2 F_2(v(g), \widehat{v}) \tag{1}$$

where $\widehat{g}$ is the initial (target) ODM and $v(g)$ is a function that assigns the ODM to the road network. This function provides the static user equilibrium (see next section). The functions F_1 and F_2 return the distance between vectors and can be defined in various way.

The formulation of the objective function F based on entropy maximization was published by Henk [18,22]. The maximum likelihood approach was presented by [16]. Cascetta used the objective functions based on generalized least

squares [4]. The Bayesian inference approach provides a method for combining of two sources of information [6,11]. Solution based on classic least squares was published in [10,14,17].

For the purpose of this paper, we simply choose $\gamma_1 = 0$ and

$$F = F_2 = \frac{1}{2} \sum_{a \in \widehat{A}} (v_a - \widehat{v_a})^2 \tag{2}$$

as in [17]. There is a lot of methods how to solve this bi-level problem. The general approach is to use the steepest decedent with a long step [10,17]. A general solution strategy consists from three steps:

- Compute search direction. The simplest method is to take the negative value of the gradient.
- Determine the size of a step so that the objective function is minimized.
- Update the ODM using the search direction and the step size.

The most difficult task is to compute the gradient of F_2, namely, the value of $\frac{\partial v_a}{\partial g_{ij}}$ [10]. For computation of these values there is several heuristic methods (see [10]). A common feature of these methods is that they need the *assignment matrix*. The assignment matrix expresses the relationship between the edge flow v_a and OD flow g_{ij}.

In this paper we present an effective way how to compute the assignment matrix implicitly without path enumeration. For this purpose, the origin-rooted flow provided by bush-based algorithms, are used.

2.2 User Equilibrium

Let the cost c_a of the edge $a \in A$ is dependent on the traffic flow v_a

$$c_a = c_a(v_a) \tag{3}$$

It is assumed that the cost function $c_a(v_a)$ is monotonically increasing and convex. The user's route choice is dependent on travel costs. It follows that equilibrium must be found. The user equilibrium is the state where every used path from the source zone i to the destination zone j has an equal (minimal) cost. The problem of searching user equilibrium is called Traffic Assignment Problem (TAP) and can be defined as Variational Inequality problem (VI) [5,15]. The optimal solution v_p^* must satisfy

$$\sum_{p \in P} c_p(v_p^*)(v_p - v_p^*) \geq 0, \quad \forall u \in \Lambda \tag{4}$$

where v_p is the flow on the path p, P is the set of all used paths in the road network, c_p is the cost of path p, $u = (v_p : p \in P)$ is a vector of all path flows, and Λ is the set of all feasible solutions in the space of paths

$$\Lambda = \left\{ u > 0 : \sum_{p \in P_{ij}} v_p = g_{ij} \quad \forall ij \in W \right\} \tag{5}$$

where $P_{ij} \subset P$ is a set of used paths for OD pair $ij \in W$. The relationship between the solution in the path space and the edge space is

$$v_a = \sum_{p \in P} \delta_{ap} v_p \tag{6}$$

where $\delta_{ap} \in \{0, 1\}$ is equal to one if the edge a lies on the path p otherwise the δ_{ap} is zero. It should be noted that there is only one solution in edge space. In contrast with this the solution in path space is not unique [19].

There are three basic groups of algorithms for solving TAP [13]:

- *Link-based* algorithms compute the solution in the edge space. They have low memory requirements but very poor convergence rate. A classic representative of this group is Frank-Wolfe algorithm (FW). e.g., [9]
- *Path-based* algorithms search the solution in the path space. They are much faster than the link-based algorithms but generally need a lot of memory for the path enumeration. e.g., [1,20]
- *Bush-based* algorithms decompose the problem to acyclic sub-graphs called *bushes*. These algorithms are fast and do not need to enumerate the paths. e.g., [2,3,7,8].

As was shown by Perederieieva in [13], the bushed-based algorithms generally provide good performance.

2.3 Maximum Entropy User Equilibrium

As mentioned above, the assignment matrix and the flow on paths are necessary for ODM estimation, but the solution in path space is not unique. We must choose the solution with the most likely realization that is generally understood in terms of maximum entropy. The maximum entropy user equilibrium (MEUE) is defined as [19]

$$\max_{v_p} = - \sum_{ij \in W} \sum_{p \in P_{ij}} v_p \log \left(\frac{v_p}{g_{ij}} \right) \tag{7}$$

subject to

$$\sum_{p \in P_{ij}} v_p = g_{ij}, \quad \forall ij \in W \tag{8}$$

$$\sum_{p \in P} \delta_{ap} v_p = v_a, \quad \forall a \in A \tag{9}$$

$$v_p \geq 0, \quad \forall p \in P \tag{10}$$

In literature, two methods can solve the MEUE problem for real size networks. The first method by Xie [21] uses the *paired alternative segments* that are provided by TAPAS algorithm [3]. The second method, also by Xie [19] uses the flow on bushes and maximizes entropy on them so the method does not enumerate the paths. As was shown in [19], the second method provides better results.

3 Proposed Solution

In this section, the implicit method for computing the assignment matrix is introduced. For this purpose, the flow on bushes is used. This bush-based flow has to fulfill the user equilibrium and maximizes the entropy. In this paper, the B-algorithm by Dial [7] is used for providing the equilibrated bash-based flow. For entropy maximization, the algorithm by Xie [19] was implemented.

For purpose of this paper, we build on the estimation approaches according to Spiess [17] because it is simple and effective for large networks [10]. However, the idea of assignment matrix computation can be applied to various estimation methods.

According to [17] the gradient is computed as

$$\frac{\partial F(g)}{\partial g_{ij}} = \sum_{k \in P_{ij}} p_k \sum_{a \in \widehat{A}} \delta_{ak}(v_a - \widehat{v_a}) \tag{11}$$

$$= \sum_{a \in \widehat{A}} (v_a - \widehat{v_a}) \sum_{k \in P_{ij}} p_k \delta_{ak} \tag{12}$$

where $p_k = \frac{v_k}{g_{ij}}$ is probability that user chooses the path k on the trip between origin i and destination j. The $\delta_{ap} \in \{0, 1\}$ is one if the edge a lies on path p else is zero. Let us set

$$p_{ij}^a = \sum_{k \in P_{ij}} p_k \delta_{ak} \tag{13}$$

where p_{ij}^a is the probability that the user crosses the edge a on the trip between origin i and destination j. The $\mathbb{P} = \{p_{ij}^a : ij \in W, a \in A\}$ is the assignment matrix. By substitution (13) to (11) we obtain the negative value of search direction

$$d_{ij} = \sum_{a \in \widehat{A}} (v_a - \widehat{v_a}) p_{ij}^a \tag{14}$$

According to [17] the ODM update is performed as

$$g_{ij} = g_{ij}(1 - \lambda d_{ij}) \tag{15}$$

where λ is the length of the step. Using the assignment matrix $\mathbb{P}$, the optimal step length can be rewritten as

$$\lambda = \frac{\sum_{a \in \widehat{A}} v'_a(\widehat{v_a} - v_a)}{\sum_{a \in \widehat{A}} (v'_a)^2} \tag{16}$$

where

$$v'_a = \frac{\partial v_a}{\partial \lambda} = -\sum_{ij \in W} g_{ij} d_{ij} p_{ij}^a \tag{17}$$

3.1 The Implicit Computation of Assignment Matrix

Let v_a^i be the flow on the edge $a \in A$ that starts in the origin $i \in Z$. The edges where $v_a^i > 0$ creates the acyclic sub-graph called a bush. For more precise definition see [12]. The source node of the edge a is noted as $s(a)$ and the target node $t(a)$.

According to [19], the path flow can be computed as

$$v_k = g_{ij} \prod_{a \in k} \phi_a^i \tag{18}$$

where

$$\phi_a^i = \frac{v_a^i}{\sum_{e \in I(t(a))} v_e^i} \tag{19}$$

where $I(n)$ is a set of incoming edges to the node $n \in N$. The ϕ_a^i represents the proportion of all bush flow inflowing to the target node $t(a)$ of the edge a. The $s(a)$ is the source node of the edge a. Combining (13) and (19) we have

$$p_{ij}^a = \sum_{k \in P_{ij}} \delta_{ak} \prod_{e \in k} \phi_e^i \tag{20}$$

The implicit computation of the assignment matrix p_{ij}^a is based on Breadth First Search (BFS). Let κ_n be a node label representing the probability that a driver crosses the node n on the trip between zones i and j. It follows, that

$$p_{ij}^a = \kappa_{t(a)} \phi_a^i \tag{21}$$

and

$$\kappa_n = \sum_{a \in O(n)} p_{ij}^a \tag{22}$$

where $O(n)$ is a set of outgoing edges from node n. Using the Eqs. (21) and (22) p_{ij}^a can be determined sequentially for all used edges a corresponding with the OD pair ij.

In Algorithm 1 there is a pseudo-code that computes the assignment matrix for one OD pair. The input of the algorithm is the origin node $i \in Z$, destination node $j \in Z$ and the feasible flow v_a^i on the bush originated at i. For whole assignment matrix, the Algorithm 1 must be run for every OD pair.

First, it is necessary to determine how many outgoing edges from the node n lead to the destination j. The edges with $p_{ij}^a = 0$ do not lead to the destination j. For this purpose, the BFS on the reverse graph is used. The searching starts at the destination node j. In Algorithm 1, the searching is represented by lines 4–10.

The second part of the algorithm sequentially determines p_{ij}^a values. The BFS started at the destination node j is also used for this purpose. The origin and destination node must cross all vehicles so $\kappa_i = \kappa_j = 1.0$. In every edge relaxation p_{ij}^a is determined and the node label $\kappa_{s(a)}$ is updated. The node $s(a)$

Algorithm 1: Implicit computation of the assignment matrix. The input is the OD pair $ij \in W$ and the feasible flow v_a^i on bush originated at i.

```
1  o_n = 0   ∀n ∈ N
2  U is the FIFO queue of nodes
3  U.add(j)
4  while |U| > 0 do              // compute the number of outgoing edges o_n
5      n = U.remove()
6      foreach e ∈ I(n) do
7          if v_e^i > 0 then
8              if o_s(e) = 0 then
9                  U.add(s(e))
10             o_s(e) = o_s(e) + 1
11 κ_n = 0   ∀n ∈ N
12 κ_j = 1.0                     // probability at destination node
13 U.add(j)
14 while |U| > 0 do               // determine the p_ij^e for every link in the bush
15     n = U.remove()
16     compute φ_e^i   ∀e ∈ I(n)
17     foreach e ∈ I(n) do
18         if φ_e^i > 0 then
19             p_ij^e = κ_n φ_e^i
20             κ_s(e) = κ_s(e) + p_ij^e
21             o_s(e) = o_s(e) − 1
22             if o_s(e) = 0 then
23                 U.add(s(e))
```

is added to the queue only if $\kappa_{s(a)}$ was updated by all outgoing edges leading to the destination. In Algorithm 1, this procedure is represented by lines 14–23.

In Fig. 1 you can see an example bush with ϕ_a^i that are computed from the bush flow and the result probability values p_{ij}^a. The dashed lines represent the edges leading to other destinations. These edges are eliminated by the first part of the algorithm.

4 Numerical Tests

The proposed method and the original method by [17] for ODM estimation were implemented in the Java programming language. The B-algorithm [7] for solving UE and the algorithm by [19] for determining the MEUE were also implemented in Java. All tests were performed on a laptop with four core processor Intel(R) Core(TM) i5-8250U CPU @ 1.60 GHz and with 16 GB RAM. For testing, the real model of Pilsen (the city in the Czech Republic) was used. The model has 9036 edges, 3727 nodes, 316 zones, and 65411 OD pairs. The relative gap

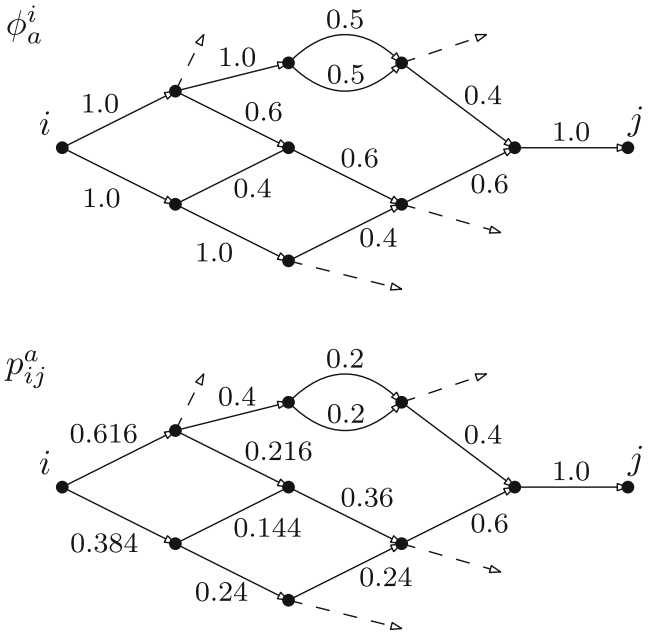

Fig. 1. Example bush with ϕ_a^i and p_{ij}^a values

for B-algorithm was set to 10^{-10}. For a simulation of the high-congested (HC) situation, the original ODM was multiplied by 2.0.

Table 1. Testing results

Model	Number of paths		Update runtime [ms]		Savings [%]
	Initial	Target	Explicit	Implicit	
Pilsen	66 109	69 179	19 467	9 691	50
Pilsen HC	71 083	124 346	26 241	10 041	62

In Table 1 there are results of the tests. The *update runtime* column represents the time that the algorithm spends in the ODM update procedure. The initial number of paths is counted before entropy maximization and the target is measured after entropy maximization. In the case of the original model, there are only 1.06 paths for one OD pair in average. Despite that, the time-savings in the update procedure compared to the original method by [17] are 50%. In high-congested case, there are 1.9 paths for one OD pair in average and the time-savings increase to 62%. It follows that the proposed method is suitable in cases, where there is a lot of route choices.

The most time-consuming part of the ODM estimation is the computation of MEUE so the total time savings are only 6% in the HC case. In most state-of-the-art approaches, the entropy maximization is not taken into account. In this case, the time-savings are 17% for the HC model.

5 Conclusion

The method that computes the assignment matrix implicitly from bush-based flows was introduced. This method does not need to enumerate the paths so the computation of the search direction and the optimal step length is more effective. The tests show that the proposed method saves more than 50% of the time that is needed for ODM updates. In the total computation time, the savings are in the order of percent.

The most time-consuming part is the computation of MEUE. It would be interesting to determine how the precision of the MEUE solution influences the quality of ODM estimation and possibly decreases the accuracy of MEUE computation.

References

1. Babazadeh, A., Javani, B., Gentile, G., Florian, M.: Reduced gradient algorithm for user equilibrium traffic assignment problem. Transportmetrica A Transp. Sci. **16**(3), 1111–1135 (2020). https://doi.org/10.1080/23249935.2020.1722279
2. Bar-Gera, H.: Origin-based algorithm for the traffic assignment problem. Transp. Sci. **36**(4), 398–417 (2002). https://doi.org/10.1287/trsc.36.4.398.549
3. Bar-Gera, H.: Traffic assignment by paired alternative segments. Transp. Res. Part B Methodol. **44**(8-9), 1022–1046 (2010). https://doi.org/10.1016/j.trb.2009.11.004. http://linkinghub.elsevier.com/retrieve/pii/S0191261509001350
4. Cascetta, E.: Estimation of trip matrices from traffic counts and survey data: a generalized least squares estimator. Transp. Res. Part B Methodol. **18**(4–5), 289–299 (1984)
5. Dafermos, S.: Traffic equilibrium and variational inequalities. Transp. Sci. **14**(1), 42–54 (1980)
6. Dey, S.S., Fricker, J.D.: Bayesian updating of trip generation data: combining national trip generation rates with local data. Transportation **21**(4), 393–403 (1994)
7. Dial, R.B.: A path-based user-equilibrium traffic assignment algorithm that obviates path storage and enumeration. Transp. Res. Part B Methodol. **40**(10), 917–936 (2006)
8. Gentile, G.: Local user cost equilibrium: a bush-based algorithm for traffic assignment. Transportmetrica A Transp. Sci. **10**(1), 15–54 (2014). https://doi.org/10.1080/18128602.2012.691911
9. LeBlanc, L.J., Helgason, R.V., Boyce, D.E.: Improved efficiency of the Frank-Wolfe algorithm for convex network programs. Transp. Sci. **19**(4), 445–462 (1985). https://doi.org/10.1287/trsc.19.4.445

10. Lundgren, J.T., Peterson, A.: A heuristic for the bilevel origin–destination-matrix estimation problem. Transp. Res. Part B Methodol. **42**(4), 339–354 (2008). https://doi.org/10.1016/j.trb.2007.09.005. http://www.sciencedirect.com/science/article/pii/S019126150700080X
11. Maher, M.: Inferences on trip matrices from observations on link volumes: a Bayesian statistical approach. Transp. Res. Part B Methodol. **17**(6), 435–447 (1983)
12. Nie, Y.M.: A class of bush-based algorithms for the traffic assignment problem. Transp. Res. Part B Methodol. **44**(1), 73–89 (2010). https://doi.org/10.1016/j.trb.2009.06.005. http://linkinghub.elsevier.com/retrieve/pii/S0191261509000769
13. Perederieieva, O., Ehrgott, M., Raith, A., Wang, J.Y.: A framework for and empirical study of algorithms for traffic assignment. Comput. Oper. Res. **54**, 90–107 (2015). https://doi.org/10.1016/j.cor.2014.08.024. http://linkinghub.elsevier.com/retrieve/pii/S0305054814002354
14. Rostami Nasab, M., Shafahi, Y.: Estimation of origin-destination matrices using link counts and partial path data. Transportation **47**(6), 2923–2950 (2020). https://doi.org/10.1007/s11116-019-09999-1
15. Smith, M.J.: The existence, uniqueness and stability of traffic equilibria. Transp. Res. Part B Methodol. **13**(4), 295–304 (1979)
16. Spiess, H.: A maximum likelihood model for estimating origin-destination matrices. Transp. Res. Part B Methodol. **21**(5), 395–412 (1987)
17. Spiess, H.: A gradient approach for the O-D matrix adjustment problem. EMME/2 Support Center, CH-2558 Aegerten, Switzerland (1990)
18. Van Zuylen, H.J., Willumsen, L.G.: The most likely trip matrix estimated from traffic counts. Transp. Res. Part B Methodol. **14**(3), 281–293 (1980)
19. Xie, J., Nie, Y.M.: A new algorithm for achieving proportionality in user equilibrium traffic assignment. Transp. Sci. **53**(2), 566–584 (2019). https://doi.org/10.1287/trsc.2018.0845
20. Xie, J., Nie, Y.M., Liu, X.: A greedy path-based algorithm for traffic assignment. Transp. Res. Rec. **2672**(48), 36–44 (2018). https://doi.org/10.1177/0361198118774236
21. Xie, J., Xie, C.: New insights and improvements of using paired alternative segments for traffic assignment. Transp. Res. Part B Methodol. **93**, 406–424 (2016). https://doi.org/10.1016/j.trb.2016.08.009. https://www.sciencedirect.com/science/article/pii/S0191261516305902
22. van Zuylen, H.J., Branston, D.M.: Consistent link flow estimation from counts. Transp. Res. Part B Methodol. **16**(6), 473–476 (1982)