The Origin-Destination matrix estimation for large transportation models in an uncongested network

1st Frantisek Kolovsky[1]
Department of Geomatics
University of West Bohemia
Plzen, Czech Republic
kolovsky@kgm.zcu.cz

2nd Jan Jezek
Department of Geomatics
University of West Bohemia
Plzen, Czech Republic
jezekjan@kgm.zcu.cz

3rd Ivana Kolingerova[2]
NTIS - New Technologies for the Information Society
Department of Computer Science, University of West Bohemia
Plzen, Czech Republic
kolinger@kiv.zcu.cz

***Abstract*—To model the ever-increasing traffic volume is an important problem. The traditional transport model utilizes Origin-Destination matrix to describe how many vehicles travel between the given zones. Estimation of this matrix is usually done using a desktop-based software and so limited by memory size and CPU performance. As computation of real models containing millions of edges and tens of thousand zones is computationally very demanding, there is a big challenge to optimize it and thus enable faster calculation. Our proposed approach contributes to the solution of this problem by distributing the computation on multiple computers with the so-called Map-Reduce model and improving the convergence rate of one existing method for the matrix calibration using Conjugate gradient method. The presented approach was developed as the tool for Apache Spark and successfully tested on the transport data of the whole European Union.**

***Index Terms*—Origin-Destination matrix estimation, traffic assignment, distributed environment, Map-Reduce, traffic volume**

I. Introduction

In recent years, the traffic volume in road network has been increasing. This puts higher demands on the traffic management using intelligent systems. One of the tools for the traffic management is traffic modeling. A state-of-the-art of traffic modeling approach consists of four consecutive independent steps (trip generation, trip distribution, mode choice, route assignment), where each step models one aspect of the transport.

Nowadays these transport models are mostly created using proprietary software tools that work in single-computer desktop environment. This way of creating the transport models is limited by the computational speed and memory size. As the size of the model and computational complexity increases, there is a rise of the demand for a scalable solution that will utilize the benefits of cloud computing. Such a solution might open many new possibilities to model large-scale networks.

[1]This work has been supported by the Project SGS-2016-004 ("Vyuzit matematiky a informatiky v geomatice III"). [2]This work was supported by Ministry of Education, Youth and Sports of the Czech Republic, the project PUNTIS (LO1506) under the program NPU I. Access to computing and storage facilities owned by parties and projects contributing to the National Grid Infrastructure MetaCentrum provided under the programme "Projects of Large Research, Development, and Innovations Infrastructures" (CESNET LM2015042), is greatly appreciated.

This work is focused on the trip distribution - the second step of the traditional transport model. The aim is to create the Origin-Destination matrix (T) that says how many vehicles travel from the place (zone) i to the place j. The travel demand in each zone and measured calibration data on the selected edges (traffic counts) are used for the T estimation.

This paper presents an improved estimation for the Origin-Destination matrix [1] so that it is suitable for large models. Our method was developed for an uncongested network, because a large network (e.g., the important roads in Europe) is usually not congested.

Our solution is based on the improvement of the convergence properties of the method by Spiess [2] by the well-known Conjugate Gradient method (CGM) and an estimate of some parameters of a deterrence function using the traffic counts. The developed algorithms were implemented on the base of Map-Reduce computing model [3]. This model is one of the most widely used for the distributed computing. The whole solution was tested on real data with good results.

Contents of the paper are as follows. First the problem is formulated and description of the state-of-the-art methods that solve this problem is given. A detailed description of the developed algorithms is presented next. After that, experiments and results are shown.

II. Background

The problem to be solved can be more precisely defined in the following way. The goal is to estimate the number of trips between all places (zones) in the area of interest. The set of all zones is called Z. The available input information for determination of the number of the trips between the zones i and j for a time unit (e.g., a day, an hour) T_{ij} is:

- a road network (graph) $G = (V, A)$; V is the set of vertices, A is the set of edges,
- the number of trips starting at zone i (O_i),
- the number of trips ending at zone j (D_j),
- the measured traffic volume (traffic count) $\widehat{v}_a$ on edges $a \in \widehat{A}$, where $\widehat{A}$ is the set of edges such that the value of $\widehat{v}_a$ is available (edges with the traffic count) and $\widehat{A} \subset A$.

The number of trips (T_{ij}) have to be determined for all $i,j \in Z$ and $i \neq j$. There are two basic methods of the Origin-Destination matrix (T) estimation:

- accurate transport data, e.g., using license plate,
- a trip distribution model, e.g., Gravity, Gravity-Opportunity [1].

The created matrices can be calibrated using the traffic count. This calibration is labeled as the third method for T estimation in some literature [4].

The first method is the most accurate, but the equipment for it is too expensive and impractical for a large area of interest.

A. Trip distribution model

The detailed socio-economic data and information about local habits (e.g., how far people travel to work) for the area of interest are needed. The most important part of the model is a deterrence function (f), which describes the trip distribution depending on the travel cost (c_{ij}). Further, the matrix T has to satisfy the following constraints:

$$\sum_j T_{ij} = O_i \qquad \sum_i T_{ij} = D_j \tag{1}$$

The fundamental equation to determine the matrix T is

$$T_{ij} = O_i D_j A_i B_j f(c_{ij}) \tag{2}$$

where A_i and B_j are balancing factors that ensure fulfillment of the conditions 1. These factors might not be quantified, but are important for the upcoming theory [1]. As can be seen, the matrix T is dependent on itself, it is necessary to use an iterative algorithm. The first approximation of the matrix is computed using $A_i = B_j = 1$ according to 2. Every k-th iteration the elements of T are computed as

$$T_{ij}^{row} = T_{ij}^{k-1} \frac{O_i}{\sum_i T_{ij}^{k-1}} \tag{3}$$

$$T_{ij}^{k} = T_{ij}^{row} \frac{D_j}{\sum_j T_{ij}^{row}} \tag{4}$$

The algorithm is terminated when 1 is valid with a sufficient accuracy. This method for balancing the matrix is called the iterative proportional fitting (IPF) [5].

The parameters of the function f are usually determined empirically by a domain expert or using an accurate research in the area of interest. Another approach published by [6] uses the traffic counts to estimate the parameters of the deterrence function. The original model was extended for a trip purpose (p). For example the set of trip purposes contains shopping, sports, hobbies (Let us note that p is an index, not a power). Thus for all p there is one O_i^p, D_j^p, A_i^p, B_j^p, f^p. The relationship between the traffic volume (v_a) on the edge a and the travel demand (O_i, D_j) is [6]:

$$v_a = \sum_p \sum_i \sum_j O_i^p D_j^p A_i^p B_j^p f^p(c_{ij}) \delta_{ij}^a \tag{5}$$

[1]The value of the factors can be calculated as $A_i = \prod_k \frac{O_i}{\sum_j T_{ij}^k}$ and $B_j = \prod_k \frac{D_j}{\sum_i T_{ij}^k}$, where k is the number of the iteration.

The function δ_{ij}^a represents the shortest paths between the zones and is defined as:

$$\delta_{ij}^a = \begin{cases} 1 \text{ if the path from } i \text{ to } j \text{ contains the edge } a \\ 0 \text{ otherwise} \end{cases}$$

The least square method was used by [6] for the estimation of the parameters of the function f^p according to 5.

The creation of the matrix T in our proposed algorithm is based on the method by [6], because this method estimates the parameters of the function f (using the traffic count) unlike the original method (equation 2, 3, 4).

B. Origin-Destination Matrix calibration

This method only calibrates the existing matrix. Let us call the matrix created in the previous step $\widehat{T}$. For such a purpose, less accurate statistical data are usually sufficient as impact on the final model is insignificant. This problem can be formulated as an optimization of the objective function $F(v, T)$ to find the optimized matrix T [7]:

$$F(v, T) = \gamma_1 F_1(T, \widehat{T}) + \gamma_2 F_2(v, \widehat{v}) \tag{6}$$

where F_1 and F_2 are some matrix distance measures, $\widehat{v}$ is a vector of the traffic counts that contains $\widehat{v}_a$ for all $a \in \widehat{A}$, v is a vector of the traffic volume that is determined using the assignment function (the vector contains v_a for all $a \in A$). All-or-nothing (AON) or User Equilibrium (TAP) [8] can be used for the assignment. The parameters γ_1 and γ_2 express the importance of each function (such that $\gamma_1, \gamma_2 \in \mathbb{R}^+$ and $\gamma_1 + \gamma_2 = 1$). Further, it is necessary to set constraints for the minimization as

$$T_{ij}, v_a \geq 0$$

There are a lot of methods available in the literature solving this problem [9], split into five categories. Information Minimization (IM) and Entropy Maximization (EM) approaches are based on maximizing information entropy [10], [11]. Maximum likelihood approach maximizes the likelihood of observing T and the traffic counts [12]. The Generalized least squares method was used by [13]. This method is not suitable for large models because the matrix inversion of an approximate size $|Z|^2$ takes a lot of time. The Bayesian Inference approach provides a method for combining two sources of information [14], [15], [16], [9].

The Gradient based solution (Bi-level programing approach) is the most general optimization approach. The matrix is adjusted in each iteration using a suitable numerical method. The solution (T) converges to a local minimum. The minimization problem is formulated as [7]:

$$\min_{T_{ij} \geq 0} F(T) = \gamma_1 F_1(T, \widehat{T}) + \gamma_2 F_2(v(T), \widehat{v}) \tag{7}$$

Spiess [2] sets the parameters γ_1 and γ_2 as follows:

$$\gamma_1 = 0, \quad \gamma_2 = 1$$

then F_1 is irrelevant and the function F_2 is defined as:

$$F_2 = \frac{1}{2} \sum_{a \in \widehat{A}} (v_a - \widehat{v}_a)^2 \tag{8}$$

The steepest descent with a long step (gradient descent) was used for optimization of the function F. This approach is the most appropriate for large models [7], because the function F is very simple, therefore the method takes less time then other methods. One of the main problems of this method is a bad convergence rate.

Our proposed solution, which will be described in the next section, tries to solve the problem with a bad convergence using a better method for the numerical optimization. Further, all algorithms were modified for the use in the distributed computing environment.

III. The proposed algorithm

Our approach consists of two steps. The first step computes the matrix using the deterrence function and the second step calibrates it using the method based on [2]. Fig. 1 shows the computational workflow including all input and output datasets.

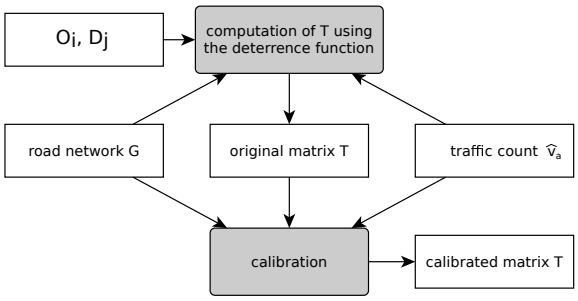


Fig. 1: Computational workflow

A. Computing the matrix T

The proposed method for determining the matrix T is derived from [6] with some adjustments. The used simplified approach has only one trip purpose. Therefore, the fundamental equation 5 can be rewritten as

$$v_a = \sum_i \sum_j T_{ij} \delta_{ij}^a \tag{9}$$

where

$$T_{ij} = \kappa O_i D_j A_i B_j f(c_{ij}) \tag{10}$$

and the constraints for the matrix T are

$$O_i = \sum_j O_i D_j A_i B_j f(c_{ij}) \tag{11}$$

$$D_j = \sum_i O_i D_j A_i B_j f(c_{ij}) \tag{12}$$

and the deterrence function was chosen as

$$f(c_{ij}) = c_{ij}^{\alpha} e^{-\beta c_{ij}} \tag{13}$$

More details about the deterrence function and a proper set of parameters can be seen in [1]. The same iterative method was used for balancing the matrix as in the previous section, c_{ij} represents the travel cost between the zone i and j. The travel costs c_{ij} were computed using Dijkstra's algorithm [17].

In the described model (equations 9 to 13) there are three unknown parameters (α, β, κ). These parameters should be estimated first. The least squares method (LSM) was used for this purpose. The problem can be written as:

$$\min_{\alpha,\beta,k} \sum_{a \in \widehat{A}} (v_a - \widehat{v}_a)^2 \tag{14}$$

Unfortunately, the problem 14 has an analytical solution only for the parameter κ, while the balancing factors A_i and B_j have to be computed iteratively. The solution is:

$$\kappa = \frac{\sum_{a \in \widehat{A}} \widehat{v}_a v'_a}{\sum_{a \in \widehat{A}} v'^2_a} \tag{15}$$

where v'_a is the derivative of v_a with respect to κ:

$$v'_a = \frac{dv_a}{d\kappa} = \sum_i \sum_j O_i D_j A_i B_j f_{ij} \delta_{ij}^a \tag{16}$$

The other two parameters were estimated by domain experts in our team, because the numerical computation of these values (e.g., using the simplex algorithm) is infeasible for large models (as verified by the authors of this paper).

The computation of the T matrix can be efficiently done by the Map-Reduce algorithm [3]. It consists of two basic parts. The first part computes the shortest paths between all zones (using Dijkstra's algorithm) and stores the identifiers of these edges $a \in \widehat{A}$, which lie on the shortest path between the zones i and j (the function δ_{ij}^a). The travel cost (c_{ij}) computed in the previous step is used to determine the matrix T according to 10, where the parameter $\kappa = 1$. The second part estimates κ according to 15.

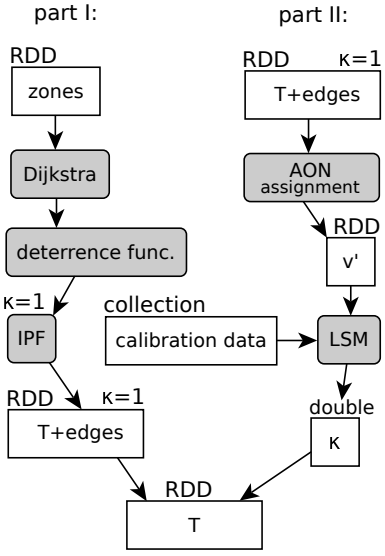


Fig. 2: Data flow of the target ODM computation

The calibration data collection contains the identifiers of the edges and measured values of the traffic on these edges. In Fig. 2 you can see the entire data flow of the T computation. The

white box indicates a dataset and the light gray box indicates a process. The label RDD (Resilient Distributed Dataset) means that the dataset is distributed among cluster nodes.

B. Calibration

The method based on the approach by [2] was chosen for T calibration. Unlike Spiess' approach the CGM was used for the minimization of the objective function F. Spiess supposes that the assignment is the Wardrop's (User) equilibrium assignment (TAP). In this case the drivers use r paths with an equal travel cost between a pair of zones. It is assumed that the network is uncongested (the travel cost is independent of the traffic volume), therefore the AON assignment method is used. Therefore, the mathematical model has to be modified. The objective function F is the same as F_2 in 8. The modeled values of the traffic volume v_a are calculated according to 9 (AON). One iteration of the optimization algorithm is composed of three steps, which are:

1) determining the search direction,
2) searching a minimum of the objective function in the search direction (line search),
3) updating the T.

There are a lot of methods for determining the search direction available in the literature. The main measure of performance is the convergence rate (a speed).

Search direction as a negative value of the gradient: The simplest approach computes the search direction as the negative value of the gradient ($-\nabla F(T)$). This method is called the steepest descent. So the gradient of the objective function 8 is:

$$g_{ij} = \frac{\partial F(T)}{\partial T_{ij}} = \sum_{a \in \widehat{A}} \delta_{ij}^{a} (v_a - \widehat{v}_a) \tag{17}$$

In this case, the convergence rate is poor, because the zig-zag effect occurs. This effect is significant in a narrow valley. Fig. 3 shows the zig-zag effect (the dashed line), x_0 is the initial solution, which converges to the local minimum x.

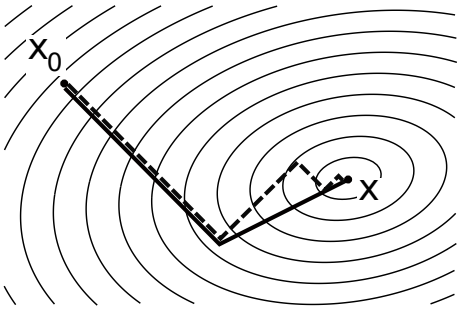


Fig. 3: Steepest descent with long step (dashed line) and CGM (solid line)

Search direction using Conjugate gradient method: The Conjugate gradient method provides a better approach for determining the search direction. In Fig. 3 there is a comparison of the Conjugate gradient with the method of the steepest descent in 2D. The method of the steepest descent (dashed line) slowly converges to the minimum while the Conjugate gradient method (solid line) finds the minimum of the quadratic problem after two steps.

The search direction of the k-th iteration can be written using the Conjugate gradient method as:

$$\mathbf{d}_k = \mathbf{g}_k + \beta_k \mathbf{d}_{k-1} \tag{18}$$

where $\mathbf{d}_k$ is the search direction and $\mathbf{g}_k = \nabla F(\mathbf{T}_k)$, where $\mathbf{T} = (T_{12}, T_{13}, \cdots, T_{|Z||Z|-1})$. The linear coefficient β_k can be determined using several techniques. The well-know PolakRibire method [18] was used. β_k can be expressed as:

$$\beta_k = \frac{(\mathbf{g}_k - \mathbf{g}_{k-1})^T \mathbf{g}_k}{\mathbf{g}_{k-1}^T \mathbf{g}_{k-1}} \tag{19}$$

The Conjugate gradient method depends on the vectors $\mathbf{g}_{k-1}$ and $\mathbf{d}_{k-1}$, therefore it requires more memory than the method of the steepest descent.

Now the matrix T can be updated using the search direction $\mathbf{d}_k$ as

$$\mathbf{T}_{k+1} = \mathbf{T}_k \circ (\mathbf{1} - \lambda_k \mathbf{d}_k) \tag{20}$$

where $\circ$ is the element-wise multiplication and $\mathbf{1}$ is all-ones vector. The step $\lambda_k \in \mathbb{R}$ is determined using a minimization of the subproblem, which is defined as:

$$\min_{\lambda} F(\mathbf{T}_k \circ (\mathbf{1} - \lambda_k \mathbf{d}_k)) \tag{21}$$

subject to

$$\lambda_k d_{ij} \leq 1 \tag{22}$$

where d_{ij} is one element of $\mathbf{d}$ (the same construction as the vector $\mathbf{T}$). This subproblem has an analytical solution equal to

$$\lambda^* = \frac{\sum_{a \in \widehat{A}} v'_a (\widehat{v}_a - v_a)}{\sum_{a \in \widehat{A}} {v'_a}^2} \tag{23}$$

where v'_a is the derivative of v_a with respect to λ_k

$$v'_a = \frac{dv_a}{d\lambda_k} = -\sum_{ij} T_{ij} d_{ij} \delta_{ij}^{a} \tag{24}$$

The coefficient λ^* must obey the constraint 22.

The Map-Reduce algorithm, which calibrates T, can be split into three parts, initialization, main loop and line search (Fig. 4). The initialization part computes the shortest paths between all zones and associates the edges $a \in \widehat{A}$ with the pair of zones ij (the function δ_{ij}^{a}). This is the same as in the previous algorithm.

At the beginning of each iteration the algorithm computes the gradient $\mathbf{g}_k$ using 17. The matrix T, function δ_{ij}^{a} and the calibration data were used for this purpose. The gradient $\mathbf{g}_k$ and the old gradient $\mathbf{g}_{k-1}$ are used for determining the value of β_k, which together with $\mathbf{g}_k$ and the old direction $\mathbf{d}_{k-1}$ determine the new search direction $\mathbf{d}_k$.

The last part of the algorithm called line search searches for the optimal step λ_k of the CGM. As follows the step must be bounded according to constraints 22. First the interval for bounding must be computed.

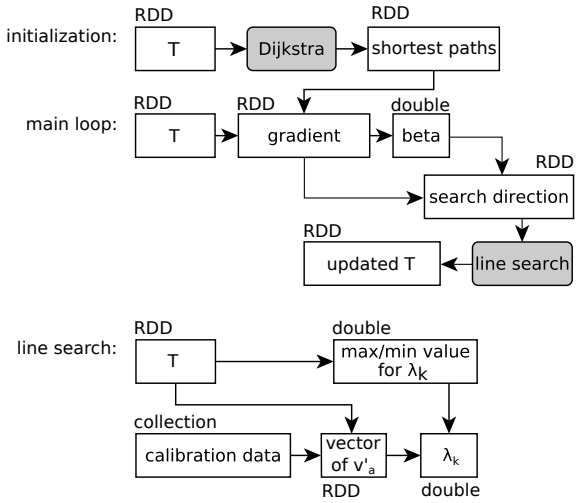


Fig. 4: Data flow of the ODM calibration

At the end of the iteration, the T is updated according to 20. The algorithm terminates when reaching a sufficiently small difference of the objective function values between two iterations or when reaching the maximum number of iterations (used for a large network).

IV. Experiments and Results

All algorithms that were described in the previous text were implemented in the Apache Spark (https://spark.apache.org). This is a framework for large-scale cluster data processing, whose priority is generality and speed. Spark's abstraction is a distributed collection of data called the RDD, which is distributed among cluster nodes. The correctness of the proposed method was tested manually on small datasets. Next we will present tests oriented to speed measurements.

Three datasets were created for testing the developed algorithms (computing and calibration of the T). In Tab. I there are parameters of these datasets. The model of city of Pilsen is small-sized and was used primarily for the development because all algorithms take less time to compute this model. The second model represents the Czech Republic. The set of all zones contains every village and all city districts. The road network consists of all roads, including all streets. The last model covers all the Europe. It includes all roads from motorways to the 3rd class roads. The set Z of the zones is represented by LAU 2 (Local Administrative Units) according to the European Union. The parameters of the deterrence function α, β were set to values 0 and 0.1.

TABLE I: Datasets for testing

dataset name	# edges $\|A\|$	# zones $\|Z\|$	$\|\widehat{A}\|$
City of Pilsen	12 207	115	60
Czech Republic (CR)	2 596 030	22 492	6 407
Europe	4 946 493	156 812	6 381

Testing was realized on two types of hardware. The dataset City of Pilsen was tested on a laptop (Intel(R) Core(TM) i5-4300M CPU @ 2.60GHz, 8GB RAM). The other two models were calculated on the cluster, which runs in YARN mode and has 24 nodes. Every node contains 4 cores (Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz) and has 64 GB RAM. So the program can use 1.5TB of memory and 96 cores.

TABLE II: Performance results for all datasets

dataset name	size of T	comp. the T time [h:m:s]	calibration time [h:m:s]	# iter
City of Pils.	10 302	00:00:05	00:00:15	30
CR	421 686 118	00:41:00	06:00:00	20
Europe	829 051 938	01:36:00	31:06:00	30

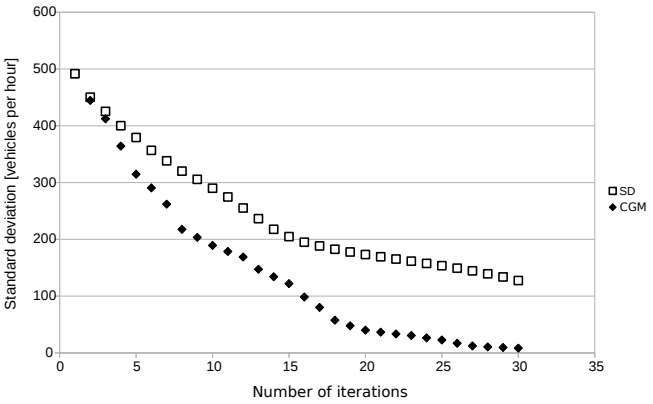


Fig. 5: Convergence rate for the implemented minimization algorithms for the Pilsen model (SD - steepest descent)

In Tab. II there are performance results. The size of T represents the number of the zone pairs ij (the number of cells in the T). For the Europe model the distance constraint was set for the Dijkstra search, because the influence of relationship between far zones is insignificant and the matrix would be too big to be computed. As can be seen in Tab. II, the T computation and calibration by our approach is reachable even for big data.

In Fig. 5 there is a comparison between two methods for determining the search direction for the Pilsen model. As can be seen, the CGM has a better convergence rate for the Pilsen model than the classic steepest descent.

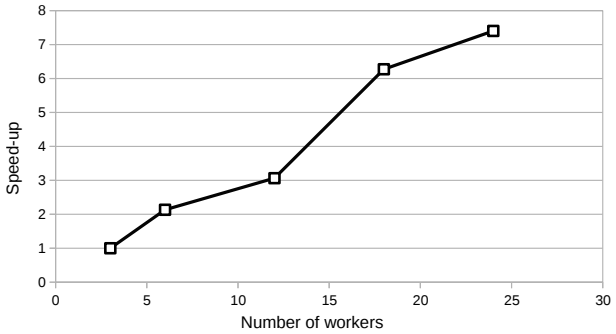
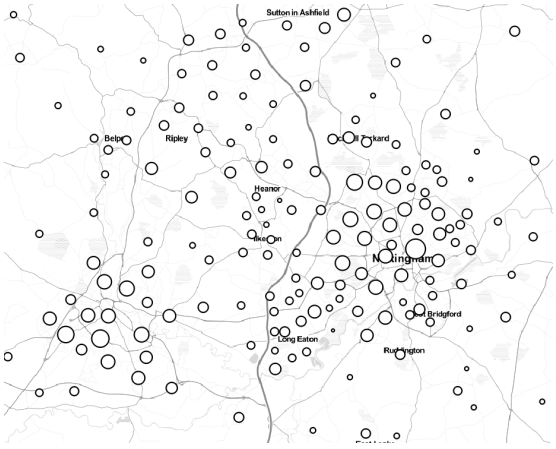


Fig. 6: Application runtime dependent on the number of computing nodes

The last performed test examines the dependence between the speed-up and the number of workers (n). The speed-up



(a) Background map with zones

(b) Traffic volumes

Fig. 7: Part of Europe transport model around Nottingham

(S) is defined as

$$S = \frac{t_3}{t_n} \tag{25}$$

where t_3 is the runtime for three workers and t_n is the runtime for n workers. This definition was chosen because the computation was unusably slow on less than three workers. In Fig. 6 there are results of this test. As can be seen, the dependence between the speed-up and the number of workers is nearly linear.

The results show that the Map-Reduce computing model is suitable for such a problem and that the proposed approach can estimate a huge matrix T. In Fig. 7 you can see part of the Europe model around Nottingham in England. The area of the circles represents O_i and the width of lines represents the traffic volume on the roads.

V. Conclusion

The tests and benchmarks show that the proposed method, where CGM was used to determine the search direction, is suitable to estimate a huge matrix using the traffic count. The solution can create larger models than the standard software for the traffic modeling based on a desktop environment and is scalable. This property was used to create the model of the whole Europe. The model contains approximately 150 000 zones and the road network has 5 000 000 links. This model was used to calculate the traffic volume for an open dataset OpenTransportMap (OTM), which is part of the project OpenTransportNet (OTN).

In the future, the more sophisticated transport model, the Wardrop's equilibrium traffic assignment should be implemented and tested in the distributed environment. It would also be appropriate to examine convergence of the Conjugate gradient method.

References

[1] J. de Dios Ortuzar and L. G. Willumsen, *Modelling Transport*, L. John Wiley & Sons, Ed. John Wiley & Sons, Ltd, 2011, iSBN 978-0-470-76039-0.

[2] H. Spiess, "A gradient approach for the od matrix adjustment problem," vol. 1, p. 2, 1990.

[3] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[4] J. Doblas and F. G. Benitez, "An approach to estimating and updating origin–destination matrices based upon traffic counts preserving the prior structure of a survey matrix," *Transportation Research Part B: Methodological*, vol. 39, no. 7, pp. 565–591, 2005.

[5] W. E. Deming and F. F. Stephan, "On a least squares adjustment of a sampled frequency table when the expected marginal totals are known," *The Annals of Mathematical Statistics*, vol. 11, no. 4, pp. 427–444, 1940.

[6] O. Z. Tamin and L. G. Willumsen, "Transport demand model estimation from traffic counts," *Transportation*, vol. 16, no. 1, pp. 3–26, 1989. [Online]. Available: http://dx.doi.org/10.1007/BF00223044

[7] J. T. Lundgren and A. Peterson, "A heuristic for the bilevel origin–destination-matrix estimation problem," *Transportation Research Part B: Methodological*, vol. 42, no. 4, pp. 339–354, 2008.

[8] J. G. Wardrop, "Road paper. some theoretical aspects of road traffic research." *Proceedings of the institution of civil engineers*, vol. 1, no. 3, pp. 325–362, 1952.

[9] S. Bera and K. Rao, "Estimation of origin-destination matrix from traffic counts: the state of the art," 2011.

[10] H. J. Van Zuylen and L. G. Willumsen, "The most likely trip matrix estimated from traffic counts," *Transportation Research Part B: Methodological*, vol. 14, no. 3, pp. 281–293, 1980.

[11] H. J. van Zuylen and D. M. Branston, "Consistent link flow estimation from counts," *Transportation Research Part B: Methodological*, vol. 16, no. 6, pp. 473–476, 1982.

[12] H. Spiess, "A maximum likelihood model for estimating origin-destination matrices," *Transportation Research Part B: Methodological*, vol. 21, no. 5, pp. 395–412, 1987.

[13] E. Cascetta, "Estimation of trip matrices from traffic counts and survey data: a generalized least squares estimator," *Transportation Research Part B: Methodological*, vol. 18, no. 4-5, pp. 289–299, 1984.

[14] M. Maher, "Inferences on trip matrices from observations on link volumes: a bayesian statistical approach," *Transportation Research Part B: Methodological*, vol. 17, no. 6, pp. 435–447, 1983.

[15] S. S. Dey and J. D. Fricker, "Bayesian updating of trip generation data: combining national trip generation rates with local data," *Transportation*, vol. 21, no. 4, pp. 393–403, 1994.

[16] T. Abrahamsson, "Estimation of origin-destination matrices using traffic counts-a literature survey," 1998.

[17] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[18] E. Polak and G. Ribiere, "Note sur la convergence de méthodes de directions conjuguées," *Revue française d'informatique et de recherche opérationnelle, série rouge*, vol. 3, no. 1, pp. 35–43, 1969.