COMENIUS UNIVERSITY IN BRATISLAVA
FACULTY OF MATHEMATICS, PHYSICS AND INFORMATICS



GENERATION OF ORIGIN-DESTINATION MATRICES IN TRANSPORT MODELLING

MASTER'S THESIS

2021 Bc. Ivana BEŇOVÁ

COMENIUS UNIVERSITY IN BRATISLAVA
FACULTY OF MATHEMATICS, PHYSICS AND INFORMATICS

GENERATION OF ORIGIN DESTINATION MATRICES IN TRANSPORT MODELLING

MASTER'S THESIS

Study program:	Mathematics of Economy, Finance and Modeling
Field of study:	1113 Mathematics
Department:	Department of applied mathematics and statistics
Supervisor:	Peter Vanya, PhD.

Bratislava 2021 **Bc. Ivana BEŇOVÁ**

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY FYZIKY A INFORMATIKY

GENEROVANIE ORIGIN DESTINATION MATÍC V DOPRAVNOM MODELOVANÍ

DIPLOMOVÁ PRÁCA

Študijný program:	Ekonomicko-finančná matematika a modelovanie
Študijný odbor:	1113 Matematika
Školiace pracovisko:	Katedra aplikovanej matematiky a štatistiky
Vedúci práce:	Peter Vanya, PhD.

Bratislava 2021 **Bc. Ivana BEŇOVÁ**





Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

THESIS ASSIGNMENT

Name and Surname:	Bc. Ivana Beňová
Study programme:	Mathematical Economics, Finance and Modelling (Single degree study, master II. deg., full time form)
Field of Study:	Mathematics
Type of Thesis:	Diploma Thesis
Language of Thesis:	English
Secondary language:	Slovak

Title: Generation of origin-destination matrices in transport modelling

Annotation: This project explores a new way to generate origin-destination matrices for transport modelling. We exploit the principle of maximum entropy, which has foundations in the field of statistical mechanics, to provide an alternative to established iterative methods. The new algorithm will be thoroughly validated for a range of inputs corresponding to various practical scenarios.

Supervisor:	Peter Vanya, PhD.
Department:	FMFI.KAMŠ - Department of Applied Mathematics and Statistics
Head of department:	prof. RNDr. Marek Fila, DrSc.

Assigned: 17.12.2019

Approved: 21.12.2019 prof. RNDr. Daniel Ševčovič, DrSc.
Guarantor of Study Programme

.....
Student

.....
Supervisor





Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Ivana Beňová
Študijný program: ekonomicko-finančná matematika a modelovanie (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor: matematika
Typ záverečnej práce: diplomová
Jazyk záverečnej práce: anglický
Sekundárny jazyk: slovenský

Názov: Generation of origin-destination matrices in transport modelling
Generovanie origin-destination matíc v dopravnom modelovaní

Anotácia: Táto práca skúma nový spôsob generovania origin-destination matíc v dopravnom modelovaní. Budeme využívať princíp maximálnej entropie, ktorá má základy v štatistickej mechanike, ako alternatívu k zavedeným iteračným metódam. Nový algoritmus bude overený pre celý rad vstupov zodpovedajúcich rôznym praktickým prípadom.

Vedúci: Peter Vanya, PhD.
Katedra: FMFI.KAMŠ - Katedra aplikovanej matematiky a štatistiky
Vedúci katedry: prof. RNDr. Marek Fila, DrSc.

Dátum zadania: 17.12.2019

Dátum schválenia: 21.12.2019 prof. RNDr. Daniel Ševčovič, DrSc.
garant študijného programu

.....
študent

.....
vedúci práce

Acknowledgements My sincere thanks and appreciation go to my supervisor Peter Vanya, PhD. for his patience, guidance, support and time spent explaining the importance of transport modelling.

Abstrakt

BEŇOVÁ, Ivana: Generovanie origin destination matíc v dopravnom modelovaní [Diplomová práca], Univerzita Komenského v Bratislave, Fakulta matematiky, fyziky a informatiky, Katedra aplikovanej matematiky a štatistiky; vedúci práce: Peter Vanya, PhD., Bratislava, 2021, 76 str.

V našej práci navrhujeme alternatívne metódy k iteratívnemu modelu na generovanie origin-destination (OD) matíc v makroskopickom dopravnom modelovaní. Pomocou iteračného gravitačného modelu reprodukujeme OD matice pre mestá s rôznymi disperziami v priestore. Priebeh entropie matice sledujeme počas iterácií, pretože sa dá odvodiť, že gravitačný model je aj riešením maximalizácie entropie matice s obmedzeniami na odchody a príchody do zón. Ako alternatívu používame démonov algoritmus, Monte Carlo simuláciu pre mikrokanonické súbory v štatistickej mechanike. Pozorujeme, že distribúcia prvkov získaná spriemerovaním stavov s nulovou energiou démona je aproximáciou distribúcie s maximálnou entropiou. Tento algoritmus rozšírime pre OD matice, kde zadaná priemerná dĺžka cesty má úlohu energie. Druhým alternatívnym prístupom je konvexná optimalizácia, ktorá sa dá previesť na úlohu kónického programovania nad exponenciálnym kužeľom a potom vyriešiť pomocou solvera ECOS. Následne porovnáme tieto tri algoritmy pre rôzne distribúcie zón a ich populácie populácie. Gravitačný model je lepší v škálovaní výpočtovej zložitosti. Výhodou Monte Carlo simulácie a konvexnej optimalizácie je však ľahké zadanie požadovanej priemernej dĺžky cesty alebo v budúcnosti iných podmienok do algoritmu. Odchýlka od požadovanej priemernej dĺžky cesty a porovnanie entropie výslednej OD matice so vzorom daným gravitačným modelom sú sľubné pre konvexnú optimalizáciu. Simulácia Monte Carlo poskytuje dostatočnú aproximáciu iba pre niektoré systémy. Účel algoritmu, ktorým je schopnosť systematicky pridávať nové väzby predstavujúce experimentálne poznatky o mobilite ľudí, bol testovaný pre konvexnú optimalizáciu na jednoduchom príklade s obmedzeniami pre prvky OD matice.

Keywords: dopravné modelovanie, origin-destination matica, gravitačný model, Démonov algoritmus, entropia, konvexná optimalizácia, primárno-duálna metóda vnútorného bodu

Abstract

BEŇOVÁ, Ivana: Generation of origin-destination matrices in transport modelling [Master Thesis], Comenius University in Bratislava, Faculty of Mathematics, Physics and Informatics, Department of Applied Mathematics and Statistics; Supervisor: Peter Vanya, PhD., Bratislava, 2021, 76 pg.

In this work, we propose alternative methods to the iterative model for the generation of origin-destination (OD) matrices in macroscopic transport modelling. Using the iterative gravity model, we reproduce OD matrices for zones with different spatial population dispersions. We observe the evolution of matrix entropy through iterations because we derive that the formula for the gravity model is also a solution of the entropy-maximizing approach with a set of constraints about zonal trip production and attraction. As an alternative, we apply the demon algorithm, a Monte Carlo simulation for microcanonical ensembles in statistical mechanics. We observe that the distribution of elements averaged over states with zero demon energy approximates the distribution with maximum entropy. We extend this algorithm for OD matrices with a specific mean trip length, which can have the role of energy. The second explored approach is convex optimization, which can be represented as exponential cone programming and then solved using the ECOS solver. We compare these three algorithms for different distributions of zones and their populations. The gravity model is superior in computational complexity scaling with the number of cities in the system. However, the advantage of the Monte Carlo simulation and convex optimisation is the easy input of the desired mean trip length into the algorithm, or in the future other constraints. The deviation from the desired mean trip length and the comparison of the entropy of the resulting OD matrix, with reference given by the gravity model, are promising for convex optimization. Monte Carlo simulation provides sufficient approximations only for some systems. The purpose of an algorithm, the ability to systematically add new constraints representing experimental knowledge about the mobility of people, was tested for the convex optimization on the simple example of a network with constraints on elements of the OD matrix.

Keywords: transport modelling, origin-destination matrix, gravity model, Demon algorithm, entropy, convex optimization, primal-dual interior-point method

Contents

Introduction **11**

1 Transport modelling **13**

1.1 Introduction to transport modelling 13

1.2 Network and zoning system 15

1.3 Classic transport model 16

1.4 Gravity distribution model 23

1.5 Microstate, mesostate and macrostate 27

1.6 Entropy-maximizing approach 27

1.7 Simulations of the iterative method 31

2 Demon algorithm **35**

2.1 Demon algorithm 35

2.2 Simulations of the demon algorithm 37

2.3 Extension of demon algorithm for generation of the origin-destination matrices 40

2.3.1 Initialization 40

2.3.2 Algorithm 42

2.4 Simulations of the extended demon algorithm 44

2.4.1 Constant population and constant spacing between zones 44

2.4.2 Linear growth of population and constant spacing between zones . . 51

3 Application of convex optimisation **56**

3.1 Exponential cone programming 56

3.2 Primal-dual interior-point method 58

4 Comparison of explored methods **60**

4.1 Comparison of alternative methods to the gravity model according to accuracy and time complexity 60

4.2 Summary of comparison 68

4.3 Adding new information and constraints 68

Conclusion **72**

References **74**

Appendix A **77**

Introduction

Transport modelling is an essential part of infrastructure planning all over the world. Before building a new road or railway section, we should first understand the demographic and economic composition of the population and the flow of goods in the region to estimate the current and future demand for the proposed service. With the opportunity of more accessible and faster travelling to the rural areas, there is also the possibility of traffic collapse on sections of the road network. Given the current demographics and the road network, we want to know the intensities of vehicles on the roads. [6]

Such a forecast can be obtained by transport modelling and can be used to construct new or repair existing roads. One of the steps in solving this complex problem is trip distribution, estimation of the flows of people between the origin (for illustration, home) and the destination (for example, work). The result is an origin-destination (OD) matrix. The first one to use the iterative method to generate OD matrices from demography, nowadays called the gravity model, was Casey in [3]. The law of retail gravitation was used to analyse potential annual dollar sales of a proposed shopping centre. The result was translated to the number of customers and automobiles per day. Then the calculation of the parking, access, and highway needed for this demand was executed. From that time, the gravity model developed and was used for realistic transport models like the trip distribution model for regional railway services in Spain [5] just as well for almost any situation that involves flows, for example, the model of migration in [22].

We look at the iterative gravity model and its possible substitutions in this work. The general problem of iterative methods is that sometimes they can be slow or can fail to converge. Also, it would be demanding to add another condition on the OD matrix except for the mean trip length, a crucial observable quantity in transport modelling. The constraints can be in some specific conditions measured and then added, representing partial external information about the traffic in a given area, such as the traffic census of trips between two specific zones or the mean trip length in the sub-area under investigation. It may ensure the higher reliability of the resulting OD matrix. It is in our interest to incorporate as much verified information as possible into the calculation of the OD matrix, so the matrix is as similar as possible to the observed reality. Such direct incorporation is not possible using the existing iterative gravity model. Therefore, we explore alterna-

tive methods through the principle of maximum entropy, where the input of constraints is simple. In case of successful implementation and calculation, the alternative methods become more versatile than the iterative method.

The first chapter will introduce the reader to transport modelling and its most used methods. Using a simple exemplary transport network, we show how to model trip generation, trip distribution, modal split, and assignment. The derivation of the gravity model through the maximisation of entropy of the OD matrix is presented. At the end of the chapter, we analyse on more realistic simulations how the dispersion of zones in space affects the gravity model.

The second chapter explains the motivation for the first alternative method, which is the demon algorithm. The simulations are executed better to understand the algorithm and its results for different systems. This Monte Carlo simulation uses statistical mechanics, on which stands modern physics and chemistry are extended and tested for the OD matrices. We compare the results to the ones obtained by the gravity model.

The second alternative, which is explained in the third chapter, is the convex optimisation problem which can be transformed into exponential cone programming and solved by a solver.

The last chapter of this paper is dedicated to comparing the suitability of the new methods in generating OD matrices for different scenarios. We investigate the computational complexity scaling with the number of zones in the system, the convergence to desired mean trip length, and the entropy of matrices for each method and different scenarios of population and distribution of zones. Then the simple example of the calculation of OD matrix with added constraints for elements of OD matrix is tested for the best alternative.

1 Transport modelling

1.1 Introduction to transport modelling

The world is changing fast, and every day there are quicker or more comfortable ways to transport from one place to another. However, we still register that many transport problems stayed the same for a long time, such as pollution, congestion of roads, accidents, or inadequate access to rural areas. The thriving economic growth develops demand that exceeds the capacity of most of the road network. Underinvestment created a fragile system that breaks down every time something differs from average. [17, page 1]

Simultaneously, we have greater confidence in technical solutions, and Information Technology has advanced enough to help us make new and better conclusions. After years of experimentation, there is now a better recognition of modelling in transport planning. The only way to benefit is to make decisions maximising the advantages of the transport system and minimising the costs and side effects. [17, page 2]

Using transport modelling and implementing it into transport planning, we can find such a decision. The goal is to find the equilibrium of transport demand and supply. People travel to satisfy their needs (with the possible exception of sightseeing), such as working, shopping, or studying at a particular location. Such an example of application of transport modelling is in [16, page 7], the paper which proposes an innovative method of replicating the multi-modal public transport system in Singapore with high precision using smart card database. The following graphs shows the network and demand created in traffic planning software PTV VISUM .

There are different types of transport models. If we classify them according to the model's detail, there are three types: microscopic, macroscopic, and mesoscopic models. Microscopic traffic models focus on traffic flow details and the interaction taking place within them. They simulate single vehicle-driver or pedestrian units like the response to traffic patterns, reactions to variations, interactions with other participants and route choices, i.e. mostly individual behaviours. In [14] Helbing a Molnár suggested that the motion of pedestrians could be described by a measure of the internal motivations of the individual for certain activities. Mesoscopic traffic models look at vehicle groups or represent the probability of choices of an individual driver. Last but not least, macroscopic

transport models represent large networks efficiently and assume a sufficiently large number of vehicles on the road such that each stream of vehicles can be treated as flowing in a tube or a stream [10]. In this work, we will focus on macroscopic transport modelling.



Figure 1: Replicated public transport network in Singapore in PTV VISUM

Source: [16, page 7]



Figure 2: Public transport passenger demand over the entire network in PTV VISUM.

Source: [16, page 8]

1.2 Network and zoning system

It may seem trivial, but the demand is distributed over space. The most common solution to treat space is to divide it into zones with transport networks. The first thing we need to realise about supply is that it is a service, not a good. Therefore it can not be stock and used in times of higher demand. Also, investments in infrastructure take a long time to carry out, and once they are done, there is only a small or no chance of going back in case of wrong decision. [17, page 3-4]

As in every model, it is needed to compromise between two conflicting objectives: accuracy and cost. Greater accuracy would be achieved by using detailed zoning and network. However, it is not possible to consider every household and its behaviour separately because even a minor change (moving of one family) would affect transport demand. [17, page 128]

Firstly the study area, a district that will be modelled, should be distinguished from the rest of the world. One should consider mandatory, optional, long or short distance trips before selecting the roads. The study area should be larger than the area of interest, district, where we particularly want to be aware of the traffic situation, but not too big as most trips should have their origin or destination in the area of interest. For example, when working on the transport model of the Slovak Republic (i.e. Slovak republic is the area of interest), it is reasonable to add into our study area surrounding countries like Czech Republic, Poland, Austria or Hungary, and maybe even Croatia. [17, page 129]

The network model must describe spatial transport supply. Thus, it consists of several network objects containing relevant data. The network is usually modelled as a directed graph, i.e., nodes and links joining them. There are two types of nodes: previously mentioned zones (also called traffic cells) and ordinary nodes. The difference is that zones can produce demand, which means they can be the origin or destination of trips within the transport network, and nodes are the intersection of links. Links connect nodes in a particular direction and can have different attributes that affect the network and model like length, travel speed, number of lanes, capacity in a volume of cars per hour and so on. [17, page 131-134]

Usually, there are other objects as turns, which indicate permitted turning movements at a node, connectors, connecting zones to the link network or stops where passengers may board or alight on the bus. However, for simplification, we do not assume them in our project.

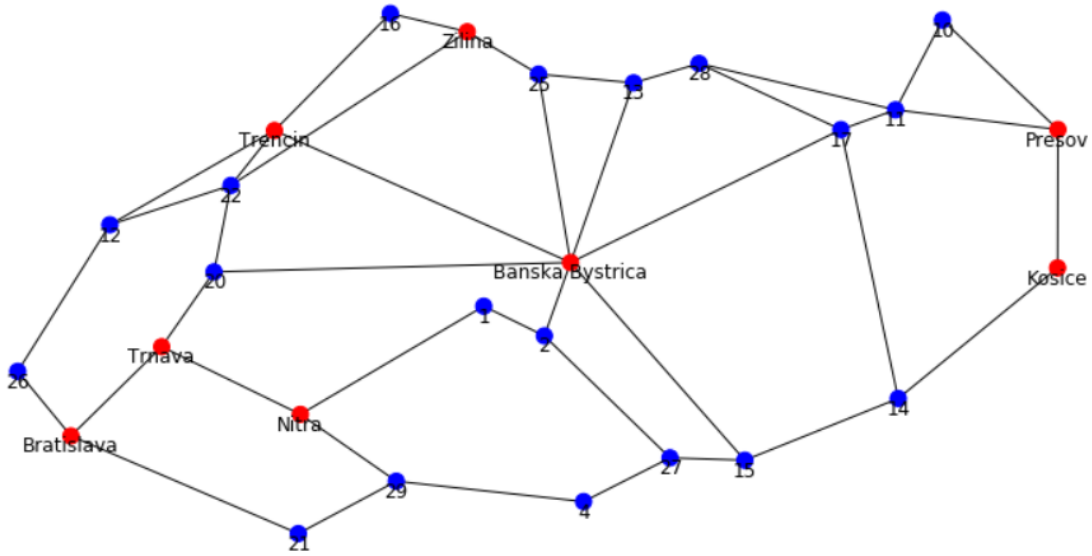


Figure 3: Simple example of transport network of the Slovak Republic in transport modelling, red points represent zones, blue points represent nodes, black lines are links.

1.3 Classic transport model

The general form of the classic transport model starts with previously described zoning, selection of network system and collection of planning, calibration and validation data. These data should include population of different socio-economic and demographic groups, economic parameters such as unemployment, shopping space, educational facilities or the mean trip length in the area. [17, page 21]

The model itself consists of four submodels,

- trip generation,
- trip distribution,
- modal split,
- assignment.

Trip generation

Different aspects can characterise trips; for example, according to the purpose, there are many different trips like trip to work, school or a shopping trip. A fundamental classification is also classification by type of person undertaking the trip. An employed person with a car probably travels more than an unemployed person or pensioner using public transport.

It would be almost impossible to include the behaviour of all of these people and their trips in one equation. Therefore we create multiple demand strata, one for each type of trip, according to the area we want to model, for example, trips to work, school, services or long trips. The collected data are then used to estimate a regression model of the number of these trips generated and attracted by each zone in each demand stratum. For the modelling of school trips, we can use the number of students in each zone or the number and quality of schools in the area. This calculation, in general, does not lead to the equity relation of the sum of trips originating (the origins O_i) at all zones and sum of trips attracted (the destination D_i) to them, i.e.

$$\sum_i O_i = \sum_j D_j. \tag{1}$$

It is said in [17, page 21, 156] that in practice, we consider that sum of all origins O_i is the correct figure for the total number of trips T therefore, all destinations D_j are multiplied by a factor γ, to satisfy the condition (1), given by

$$\gamma = \frac{\sum_i O_i}{\sum_j D_j} = \frac{T}{\sum_j D_j}. \tag{2}$$

As the whole republic is modelled in our example, it is not reasonable to model work trips because not many people regularly travel for work from Košice to Bratislava. Therefore we will model long trips that usually include sightseeing trips. Possible outcome for our example can be vectors of origin and destination for zones Bratislava, Trnava, Trenčín, Nitra, Banská Bystrica, Žilina, Prešov and Košice

$$O = \begin{pmatrix} 225000 \\ 32000 \\ 26000 \\ 36000 \\ 37000 \\ 40000 \\ 44000 \\ 115000 \end{pmatrix}, D = \begin{pmatrix} 351000 \\ 21000 \\ 17000 \\ 32000 \\ 18000 \\ 30000 \\ 20000 \\ 66000 \end{pmatrix}.$$

Vector of origin was obtained as one half of the population of zones, as we assume that around half of the population is economically active, therefore, has the potential to travel. Vector of destination was obtained as a population of zones plus number of nights spent in accommodation facilities, and it was balanced, so the sums of vectors are equal. The data for this calculation can be found in [20]. It can look to the reader as though these numbers are too large. It is important to state that people who choose to stay in their zone, i.e. their origin and destination are the same, are counted in both numbers related to the zone.

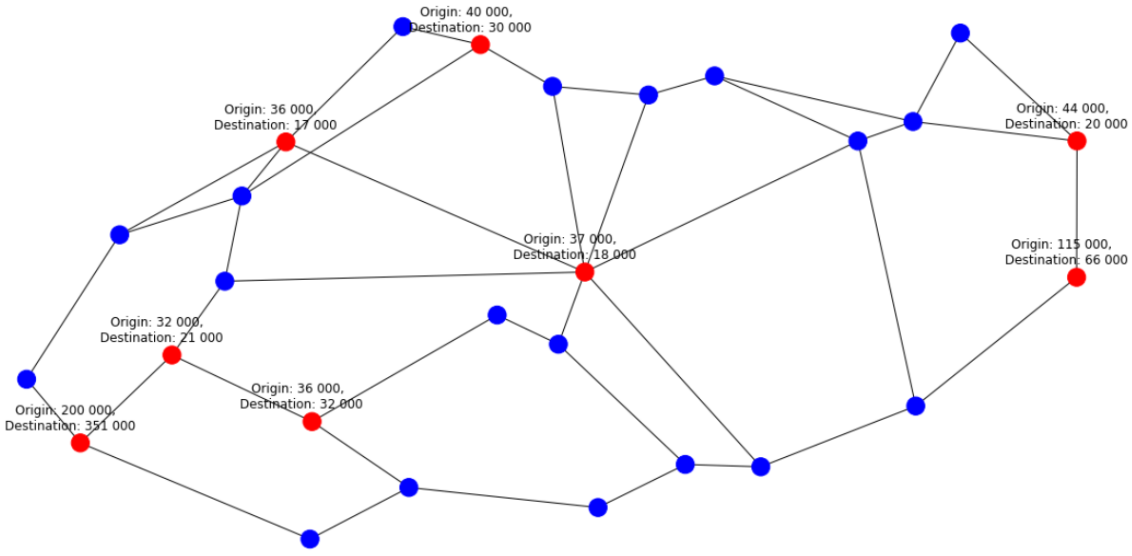


Figure 4: A simple example of the Slovak Republic network in transport modelling with numbers of origin and destination trips assigned to each zone.

Trip distribution

The next step is allocatioon of the obtained sum of trips to particular destinations, in other words, their distribution over space, thus production of an origin-destination matrix. The most frequently used model is the gravity model, which Casey introduced in 1955 in [3] on the example of estimation of parking for a new shopping centre. The gravity model will be explained in more details in section 1.4.

To generate an origin-destination matrix with the gravity model, one would need, for example, the distance of zones, travel time or the time needed to park the car. In the case of public transport, it could also be costs of the ride, the time needed to get to the bus stop or time spent waiting. To simplify our example with the Slovak Republic, we assume that the only parameter people take into account when they are travelling, is the distance. We also assume that the mean trip length of long-distance trips is 90 kilometres. The output of this algorithm is OD matrix

$$T = \begin{pmatrix} 197790 & 8130 & 4037 & 8942 & 1685 & 3445 & 215 & 751 \\ 24809 & 2283 & 1075 & 2258 & 425 & 917 & 54 & 174 \\ 17439 & 1522 & 2242 & 1905 & 607 & 1913 & 106 & 262 \\ 25320 & 2095 & 1249 & 4640 & 874 & 1349 & 111 & 358 \\ 19981 & 1653 & 1665 & 3661 & 4030 & 3850 & 515 & 1640 \\ 21379 & 1866 & 2749 & 2956 & 2015 & 7570 & 422 & 1038 \\ 11750 & 972 & 1348 & 2153 & 2370 & 3711 & 6549 & 15143 \\ 32527 & 2475 & 2630 & 5482 & 5989 & 7241 & 12023 & 46629 \end{pmatrix},$$

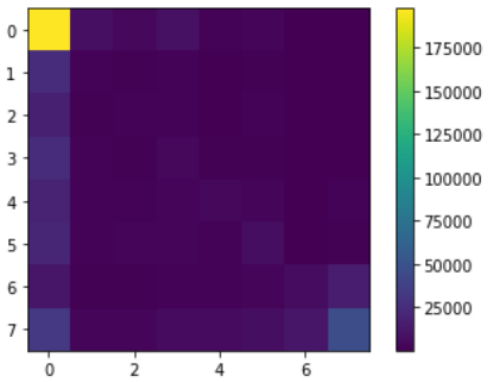


Figure 5: Origin-destination matrix for the example of the network of the Slovak Republic.

The element in the first row and the first column T_{11} represents the number of people who choose to stay in the first zone, which is Bratislava. We can see that it is the most significant number of the matrix. The element in the first row and seventh column T_{17} represents the number of people travelling from the first zone, Bratislava, to the seventh zone, Prešov. As the distance from Bratislava to Prešov is the largest of all, we estimate that only 215 people will decide to travel to Prešov. It is necessary to point out that if the mean trip length was set at higher level, the matrix would look different and more people would travel from Bratislava to Prešov. It is essential to have reliable information about the mean trip length because it dramatically affects the result.

Modal split

The next stage is modelling the choice of mode, and this results in a modal split, i.e. the allocation of trips to different modes, such as car, bus, train, or walk. It depends on car availability, trip purpose or components of monetary costs. [17, page 21]

The common theoretical framework for generating a discrete-choice model is utility theory. It is set that individuals act rationally, i.e. always select the option which maximises their utility. The most important part is that we assume that the net utility function U_{iq} of mode A_i for person q can be represented by two components. One is systematic V_{ij} part, a function of measurable attributes x and random component ϵ_{iq}, which represents a particular taste of individual q. Therefore utility has form

$$U_{iq} = V_{iq} + \epsilon_{iq}.$$

We can assume that the residuals ϵ are random variables with a mean 0 and a certain probability distribution without loss of generality. Usually, the function V_{iq} has form

$$V_{iq} = \sum_k \theta_{ik} x_{ikq},$$

where parameters θ are assumed to be constants for all individuals but may vary across mode alternatives. Measures x_{ikq} can be time spent in a vehicle, cost of travel, time spent looking for a parking plot or walking time to the stop, depending on the mode i. Measures also depend on person q; for instance, if someone has a better car, it uses less gas per kilometre; therefore, the trip is cheaper for this person [17, page 232-233].

The simplest and most used modal split model is the multinomial logit model. Here the probability of choosing i^{th} mode by the q^{th} person is

$$P_{iq} = \frac{e^{\beta V_{iq}}}{\sum_{A_j \in A(q)} e^{\beta V_{jq}}},$$

where $A_j \in A(g)$ represents a set of modes available for person q. [17, page 233]

There are many possible options how to estimate parameters $\beta\theta_{kj}$, but we will focus on estimating the model from a random sample. This estimation stands on the assumption that we can obtain chosen mode of a random sample of travellers Q, their origin and destination. For example, the individual number 1 has chosen mode 1, the individual number 2 has chosen mode 2 and so on. As the observations are independent, the likelihood function of the model probabilities that each individual chooses the option they selected is

$$L(\theta) = P_{11}P_{22}P_{31}\ldots. \tag{3}$$

If we define the variable g_{jq}

$$g_{jq} = \begin{cases} 1, & \text{if } A_j \text{ was chosen by q} \\ 0, & \text{otherwise} \end{cases}$$

we can rewrite the the expression 1.3 as

$$L(\theta) = \Pi_{q=1}^{Q} \Pi_{A_j \in A(q)} (P_{jq})^{g_{jq}}.$$

To maximize the function and estimate parameters, we take natural logarithm, derive and set the expression equal to zero, obtaining the following equation

$$l(\theta) = \sum_{q=1}^{Q} \sum_{A_j \in A(q)} g_{jq} \ln P_{jq}.$$

Because our example is fictional, we do not have a random sample of individuals. We skip the estimation of parameters and assume we have a railway only from Bratislava to Košice through Žilina. Lets suppose that probabilities of travelling by train are equal to 0.3. According to our assumption, there is no possibility to travel from Bratislava to Banská Bystrica or Nitra by train, so the probability of using a train for these trips is

equal to 0. We also assume that people do not use a train for intrazonal trips. The results are two matrices

$$T_{car} = \begin{pmatrix} 197790 & 5691 & 2825 & 8942 & 1685 & 2411 & 150 & 525 \\ 17366 & 2283 & 752 & 2258 & 425 & 641 & 37 & 121 \\ 12207 & 1065 & 2242 & 1905 & 607 & 1339 & 74 & 183 \\ 25320 & 2095 & 1249 & 4640 & 874 & 1349 & 111 & 358 \\ 19981 & 1653 & 1665 & 3661 & 4030 & 3850 & 515 & 1640 \\ 14965 & 1306 & 1924 & 2956 & 2015 & 7570 & 295 & 726 \\ 8225 & 680 & 943 & 2153 & 2370 & 2597 & 6549 & 10600 \\ 22768 & 1732 & 1840 & 5482 & 5989 & 5068 & 8416 & 46629 \end{pmatrix},$$

$$T_{rail} = \begin{pmatrix} 0 & 2439 & 1211 & 0 & 0 & 1033 & 65 & 226 \\ 7443 & 0 & 322 & 0 & 0 & 275 & 16 & 52 \\ 5231 & 456 & 0 & 0 & 0 & 574 & 32 & 79 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 6414 & 559 & 825 & 0 & 0 & 0 & 127 & 312 \\ 3525 & 292 & 404 & 0 & 0 & 1114 & 0 & 4543 \\ 9758 & 743 & 789 & 0 & 0 & 2173 & 3607 & 0 \end{pmatrix}.$$

The rows and columns of zeros in matrix T_{rail} correspond to zones Banská Bystrica and Nitra, where it is not possible in our network to travel by train. Summing those matrices would lead to matrix T calculated in part trip distribution.

Assignment

The last stage is the assignment of the trips by each mode to their corresponding links. The capacities of links are needed for this part, representing the volume of vehicles that do not affect traffic speed. In the process, firstly, the part of trips, usually one fifth, is assigned to the network through the fastest track. The actual time of travel affected by the number of travellers is calculated. The following fifth of trips is assigned to the network with this new information about travel time taking each road. This process is repeated until all trips are assigned to the network. The results are measurable intensities compared to validation data obtained from travel surveys or national traffic census, and the model is validated. [17, page 21]

1.4 Gravity distribution model

After trip generation, one can have an idea about trips originating and attracted to each zone, but this is seldom enough for modelling or decision making. It is needed to have a better idea about the pattern of trips and the trip matrix can represent this pattern. It shows the number of trips from origin to destination during a given period. It is also called the origin-destination matrix and can be disaggregated by type of person n. [17, page 175-176] Therefore:

T_{ij}^n	are trips from origin i to destination j by person n,
T^n	is total number of trips by person n,
O_i^n	is number of trips origining in zone i by person n,
D_j^n	is number of trips attracted to zone j by person n.

We will consider only one person type for simplification, so we will not use the n parameter. The origin-destination matrix has a form like shown bellow in the table 1. [17, page 176]

Table 1: General form of origin-destination matrix.

	Destinations					
Origins	1	2	3	$\dots j$	$\dots z$	$\sum_i T_{ij}$
1	T_{11}	T_{12}	T_{13}	$\dots T_{1j}$	$\dots T_{1z}$	O_1
2	T_{21}	T_{22}	T_{23}	$\dots T_{2j}$	$\dots T_{2z}$	O_2
3	T_{31}	T_{32}	T_{33}	$\dots T_{3j}$	$\dots T_{3z}$	O_3
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
i	T_{i1}	T_{i2}	T_{i3}	$\dots T_{ij}$	$\dots T_{iz}$	O_i
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
z	T_{z1}	T_{z2}	T_{z3}	$\dots T_{zj}$	$\dots T_{zz}$	O_z
$\sum_j T_{ij}$	D_1	D_2	D_3	$\dots D_j$	$\dots D_z$	$\sum_{ij} T_{ij} = T$

Source: [17, page 176]

Suppose reliable information is available to estimate both O_i and D_j. In that case, the sum of the trips in a row of OD matrix (trips having the same origin) is equal to the total number of trips arising from that zone, and the sum of the trips in a column (trips having the same destination) corresponds to the number of trips attracted to that zone. Then the model is called doubly constrained. According to [17, page 177], these conditions can be written as:

$$\begin{aligned} \sum_j T_{ij} &= O_i, \\ \sum_i T_{ij} &= D_j. \end{aligned} \tag{4}$$

Generalized travel costs are typically a linear function of the attributes of the journey weighted by coefficients that represent their importance. Possible representation is

$$c_{ij} = a_1 t_{ij}^v + a_2 t_{ij}^w + a_3 t_{ij}^t + a_4 F_{ij} + \sigma,$$

where

t_{ij}^v	is in-vehicle time spend from zone i to j,
t_{ij}^w	is walking time, to and from bus stop or parking,
t_{ij}^t	is waiting time at the bus stop or time spent looking for parking place,
F_{ij}	is a fare charged for travel from zone i to j,
σ	is parameter representing all the other attributes not included.

As said in [24, page 16], the simplest gravity model was based on Newton's law of gravitation force between two masses m_i and m_j separated by distance d_{ij}

$$F_{ij} = \gamma \frac{m_i m_j}{d_{ij}^2}, \tag{5}$$

where γ is a constant. The transport gravity model therefore is

$$T_{ij} = k \frac{O_i D_j}{c_{ij}^2}. \tag{6}$$

This model from book [24, page 16] has sensible insides. The number of trips is proportional to O_i and D_j and inversely proportional to travel costs between zones. The problem is that if we double a particular O_i and D_j, the number of trips between these

zones would be quadrupled, but one would assume that they should also be doubled. To incorporate equations (4), we have to introduce balancing factors A_i and B_j associated with production and attraction zone, respectively. The modified gravity model is then

$$T_{ij} = A_i B_j O_i D_j f(c_{ij}), \tag{7}$$

where

$$\begin{aligned} A_i &= \left[\sum_j B_j D_j f(c_{ij})\right]^{-1}, \\ B_j &= \left[\sum_i A_i O_i f(c_{ij})\right]^{-1}, \end{aligned} \tag{8}$$

and $f(c_{ij})$ is a generalized function of travel costs. According to [17, page 182], it is often called deterrence function as it demotivates to travel as distance or costs increase. Empirical observations, especially in urban areas, have shown that the number of passengers is small among zones where the cost of travel is low. Many inhabitants travel medium-long journeys, but with the increasing costs, passengers decay again with a few very long trips. Several types of generalised function of the travel costs with one or more parameters for calibration are used to approximate such conditions like

$$\begin{aligned} f(c_{ij}) &= e^{-\beta c_{ij}}, \text{exponential deterrence function}, \\ f(c_{ij}) &= c_{ij}^{-n}, \text{power deterrence function}, \\ f(c_{ij}) &= c_{ij}^{-n} e^{-\beta c_i j}, \text{mixed deterrence function}. \end{aligned} \tag{9}$$

Parameter β, which is contained in these functions, must be calibrated so the mean trip length is as close to the actual parameter obtained from data as possible. We describe the Hyman approach from [15] below.

1. Take $m = 0$, $\beta_0 = \frac{1}{c^*}$, where c^* is the mean cost from the observed trip length distribution.

2. Calculate trip matrix using the gravity model and value β_0.

3. Obtain the mean modelled trip cost c_0 and calculate better value for β as $\beta_m = \beta_0 \frac{c_0}{c^*}$.

4. Make $m = m+1$. If c_{m-1} and c^* a sufficiently close stop the iterations and accepted β_{m-1}. Otherwise go to step 5.

5. Calculate a better estimate of β as

$$\beta_{m+1} = \frac{(c^* - c_{m-1})\beta_m - (c^* - c_m)\beta_{m-1}}{c_m - c_{m-1}}.$$

6. Repeat step 4.

Coefficients A_i and B_j are found iteratively by setting all $A_i = 1$ and solving for B_j. These values are then used to re-estimate A_i, and this process is repeated until the convergence is achieved. This process is sometimes slow if we require a high convergence rate. [18, page 190]

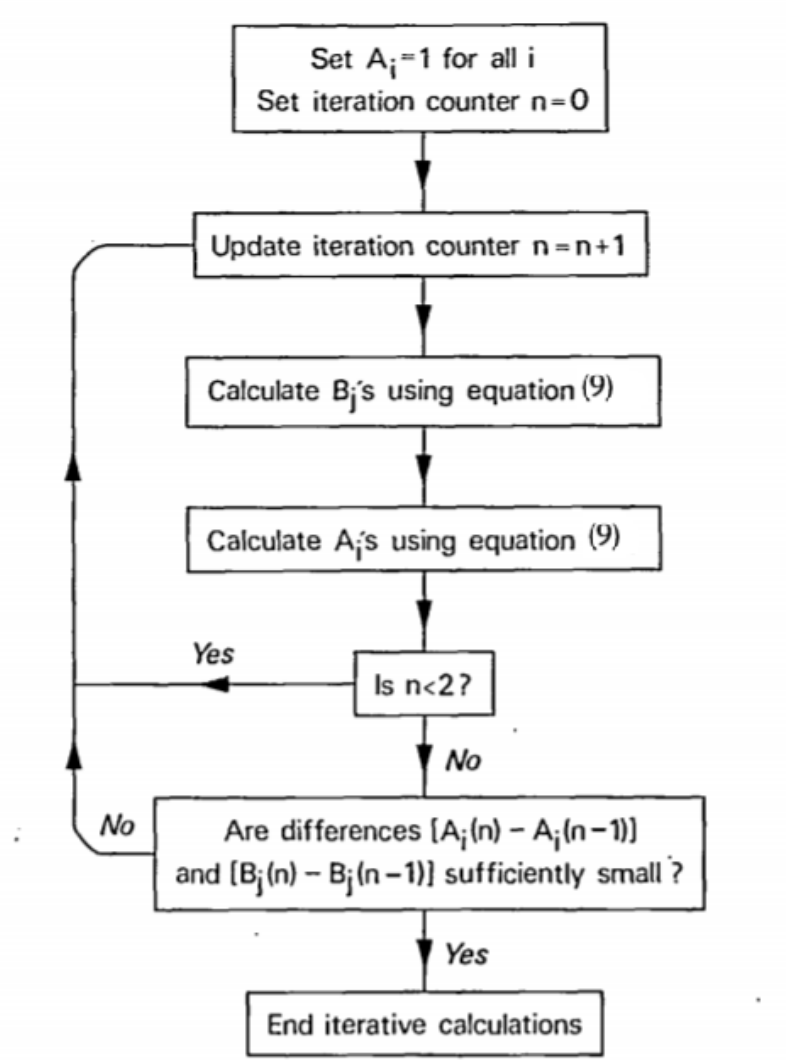


Figure 6: Scheme of the iteration process, of balancing factors A_i and B_j in the gravity model.

Source: [18, page 190]

1.5 Microstate, mesostate and macrostate

We will introduce the maximum-entropy approach used for the generation of many models, including the gravity model, in the next subsection. To do so, it is necessary to explain and define a few concepts, including entropy.

Consider a system of a large number of distinct elements. A complete description of this system requires a specification of its *microstates*, as each element is distinct. In the context of transport modelling, we would need to know about each individual's origin, destination, travel mode, time of the journey and more. This is is more information than we are usually able to get [17, page 185].

A more aggregate or *mesostate* specification requires less information than the microstate. Following our example, it specifies the number of trips between each origin and destination. Two individuals from the same origin could exchange their destination, the microstate in this situation is different, but the meso state stayed the same. That is simply an origin-destination matrix [17, page 185].

There is an even higher level of aggregation in *macrostate* that states, for example, the total number of trips generated and attracted to each zone. This is the level of information we are usually provided after trip generation. [17, page 185]

The assumption, which is the whole basis of statistical physics, is that all accessible microstates are equally probable. Therefore our goal is to find the most probable mesostate trip distribution (with the largest number of microstates) given our constraints about the macrostates. This structure is expected to develop an entropy-maximizing model [17, page 185].

1.6 Entropy-maximizing approach

The approach in [13, page 181-182] explains the following formula. Suppose we have N distinguished elements. In how many ways can the N elements be arranged into $r+1$ piles with n_j objects in pile number j $(j = 0, 1, 2, \ldots r)$, ordering within the piles being unimportant?

Firstly we look at the number of ordered sequences into which these elements can be arranged. This is $N!$. Lets say that $r = 1$, so we need to divide the elements into two piles with n_0 and n_1 elements. We can order all elements and separate them, so the first

n_0 elements are in the first pile, and the remaining n_1 elements are in the second pile. However, the $N!$ arrangements will not all give distinct answers to the presented question since the order within each pile not important. Mark the number of ways with symbol t. Then in equation

$$N! = tn_0!n_1!, \tag{10}$$

the factorials allow for the disordering. Parameter t is then

$$t = \frac{N!}{n_0!n_1!}. \tag{11}$$

For the r being any natural number the result follows as a straightforward extension and it is

$$t = \frac{N!}{\Pi_j n_j!}. \tag{12}$$

Getting back to transport modelling it is easy to see that number of micro states $W\{T_{ij}\}$ given a meso state T_{ij} can be calculated as

$$W\{T_{ij}\} = \frac{T!}{\Pi_{ij} T_{ij}!}. \tag{13}$$

As we are looking for the most probable meso state we want to maximize the number of micro states i. e. the number W. For convenience we are looking for maximum of monotonic function of $\log W$ as they both have the same maximum,

$$\ln W = \ln \frac{T!}{\Pi_{ij} T_{ij}} = \ln T! - \sum_{ij} \ln T_{ij}!. \tag{14}$$

Using a Stirling's approximation for $\log X! = X \log X - X$ we get

$$\ln W = \ln T! - \sum_{ij} \left(T_{ij} \ln T_{ij} - T_{ij}\right). \tag{15}$$

Usually, the coefficient T is a constant; therefore, it can be omitted from the optimization problem. The rest of the equation is

$$\ln W' = -\sum_{ij} \left(T_{ij} \ln T_{ij} - T_{ij}\right), \tag{16}$$

which can be identified with entropy function in statistical mechanics with the linear element. In physics, the concept is used objectively in thermodynamics. The concept of entropy can be related to uncertainty, as the most probable macrostate is that about which we are most uncertain in the knowledge of possible microstates.[23, page 252]

Definition 1.1. *Entropy measures a system's thermal energy per unit temperature unavailable for useful work. Because work is obtained from ordered molecular motion, the amount of entropy is also a measure of the system's molecular disorder or randomness. [12]*

This equation (16) could be written as

$$\ln W' = -\sum_{ij} T_{ij} \ln T_{ij} + \sum_{ij} T_{ij} = -\sum_{ij} T_{ij} \ln T_{ij} - T$$

with the use of $\sum_{ij} T_{ij} = T$ stated in table 1. Therefore without the loss of generality, this addition does not change the maximum of function and could be omitted. We decided to keep this term in the function and from now on we address the whole function as entropy.

Hence our problem can be constrained as maximization problem

$$\begin{aligned} &\max \ln W' \\ O_i - \sum_j T_{ij} &= 0 \\ D_j - \sum_i T_{ij} &= 0 \\ T_{ij} &\geq 0. \end{aligned} \tag{17}$$

We should state that we allow the OD matrix to have elements with zero value, $T_{ij} = 0$. In that case, the entropy term of the element is calculated as zero. Equations (17) describe the problem of finding the global maxima of a function subject to equality constraints, therefore can be handled using Lagrangian function

$$L = \ln W' + \sum_i \alpha'_i \left\{ O_i - \sum_j T_{ij} \right\} + \sum_j \alpha''_j \left\{ D_j - \sum_i T_{ij} \right\}.$$

Taking the first partial derivatives with respect to T_{ij} and equating them to zero we obtain

$$\frac{\partial L}{\partial T_{ij}} = -\ln T_{ij} - \alpha'_i - \alpha''_j = 0,$$

therefore

$$T_{ij} = e^{-\alpha_i} e^{-\alpha''_j}.$$

The Lagrangian multipliers can be rewritten as

$$\begin{aligned} A_i O_i &= e^{-\alpha'_i}, \\ B_j D_j &= e^{-\alpha''_j}, \end{aligned} \tag{18}$$

and therefore we obtained

$$T_{ij} = A_i O_i B_j D_j.$$

This model is close to but not yet the gravity model. That is because we are missing the deterrence function. If we take an additional constraint

$$\sum_{ij} T_{ij} c_{ij} = C, \tag{19}$$

where C is the (unknown) total expenditure in travel in the system. By restraining this constraint as

$$C - \sum_{ij} T_{ij} c_{ij} = 0,$$

and adding it to the problem (17), we get the different Lagrangian function. That is

$$L = \ln W' + \sum_i \alpha'_i \left\{ O_i - \sum_j T_{ij} \right\} + \sum_j \alpha''_j \left\{ D_j - \sum_i T_{ij} \right\} + \beta \left\{ C - \sum_{ij} T_{ij} c_{ij} \right\},$$

and its first partial derivatives with respect to T_{ij} looked like

$$\frac{\partial L}{\partial T_{ij}} = -\ln T_{ij} - \alpha'_i - \alpha''_j - \beta c_{ij} = 0,$$

therefore

$$T_{ij} = e^{-\alpha'_i} e^{-\alpha''_j} e^{-\beta c_{ij}}.$$

Making the same change of variables as before one obtains

$$T_{ij} = A_i O_i B_j D_j e^{-\beta c_{ij}} \tag{20}$$

We can see that the generalized function of travel costs takes the form of an exponential function. The entropy-maximizing approach provides new insight, that if we assume that people perceive their travel costs as increasing linearly, it results in exponential deterrence function. If we think that the relationship is logarithmic, meaning that the perception is $\log(c_{ij})$, which gives less weight to longer trips, then replacing c_{ij} by $\log(c_{ij})$ in equation (19) produces the the power deterrence function. This relationship occurs, for example, in a model that involves many long trips, like in an interurban model [17, page 187], [25].

Lagrangian multipliers are not directly computed. Even for a simple example, as was ours of the Slovak Republic with eight zones, we would get too many equations to calculate multipliers. We would get $8^2 = 64$ constraints for derivation by elements T_{ij}, $2.8 + 1 = 17$ constrains for derivation by Lagrangian multipliers. Calculating all of these parameters could be problematic and time-consuming because of the exponential relationship between multipliers and elements of the OD matrix.

1.7 Simulations of the iterative method

Coordinates of zones and their distance

We created n zones whose coordinates were generated from a normal distribution with mean value $\mu = 0$ and variance $D = \sigma^2$, while the parameter σ^2 could be arbitrarily changed. It can be shown as in source [1] that the limit of the probability density function of these coordinates converges to the probability density function of the standard normal distribution. In the increasing dimension, it is less likely we observe a random point on the sphere's edge.

The distance of the individual zones was the Euclidean norm between the coordinates of the zones, neglecting the curvature of the roads. We saved these values in the matrix d, where d_{ij} represented the distance between the i^{th} and j^{th} zone.

Size of zones

The size or population of zones was generated from a uniform distribution with parameters $R(100, 100000)$. This assumption was based on fact from [4], that every human (or other organisms) living in some area needs for his survival his own space (territory). On average, we assumed that this space is equal for everyone. Therefore people are uniformly distributed in the zones. Generated numbers were rounded to natural numbers, so they represented the number of inhabitants of the zone.

Trip generation

We made a presumption that these n zones were the only zones where inhabitants could travel; consequently, the vector containing the population of each zone was also a vector of trips originating at a given zone. At the beginning of the day, every citizen has to choose whether he stays in the town he lives in or travels to one of the remaining $n-1$ zones. That means that the vector of trips attracted to a given zone is also a vector generated from the uniform distribution. The sum of these vectors have to be the same.

Generalized function of travel costs

An average speed that was allowed between zones was $50\frac{km}{h}$. Financial loss of one hour travelling was calculated as one working hour for average earnings, that was in our case 6€. It was also necessary to include the price of fuel per kilometre and amortization of vehicle. With an approximate fuel consumption of vehicle being $0.07\frac{l}{km}$, an average price of petrol or diesel being $1.05\frac{€}{l}$ and depreciation of the vehicle with each kilometre, we expected an approximate price of $0.12\frac{€}{km}$. As a result, the generalized cost of travel in this case was

$$c_{ij} = 6t_{ij} + 0.12d_{ij}, \tag{21}$$

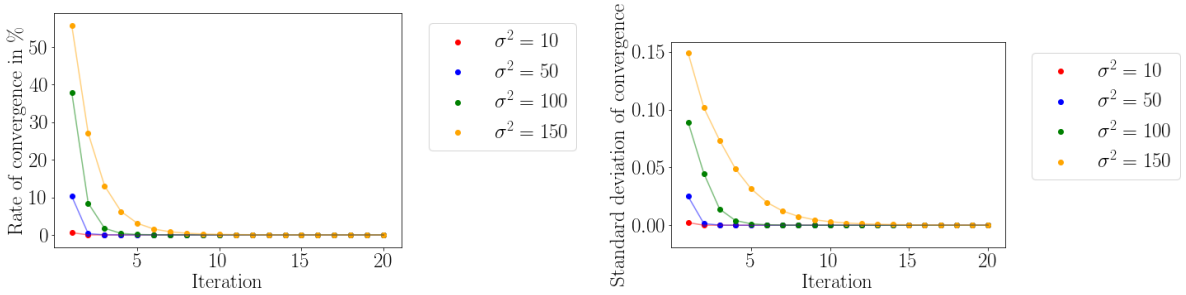
where $t_{ij} = \frac{d_{ij}}{50}$. We used the exponential function $f(cij) = e^{-\beta c_{ij}}$, where the coefficient β is under normal circumstances calibrated in several steps so that the average cost of the observed trips is almost the same as the estimated average cost. Since we had a generated data in our case, we did not have the values of the observed trips, so we chose $\beta = 0.05$, which represents a value usually very close to the actual calculations. After this step, we have had all the values needed to calculate the origin-destination matrix.

Origin-destination matrix

To better understand the process of calculations and behaviour of this city-to-city travelling prediction model, we needed to do a closer analysis. We were interested in how the results would be affected by the increase or decrease of σ^2 dispersion of the normal distribution of zone coordinates. After each mentioned iteration in which we adjusted the coefficients A_i and B_j, we recorded rates of convergence, i.e. the maximum deviation from the destination travel vectors O and D and the entropy of the matrix T using the formula $W = -\sum_{ij}(T_{ij}\log T_{ij} - T_{ij})$. We ended the iteration process after 20 iterations. Our analysis looked at 100 zones whose coordinates were generated from a

normal distribution with a 10, 50, 100, or 150 kilometres dispersion. For reliable results, zone sizes and coordinates were generated 300 times for each option, with the seed set at the beginning of the generation, selected from evenly spaced numbers on a logarithmic scale between 1 and 10^5. The repeated calculations ensured that the conclusions were not random and caused by selection.

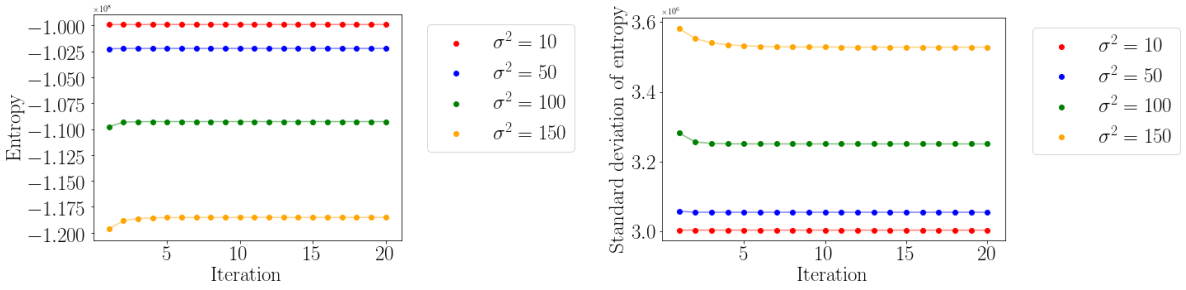
After performing all calculations and saving convergence and entropy for different σ^2, we contributed to the evaluation. We calculated through seeds the average of the values and standard deviation in each iteration.



(a) Rate of convergence through iterations for different σ^2

(b) Standard deviation of convergence through iterations for different σ^2

Figure 7: Convergence rate and standard deviation of convergence rate through iterations of gravity model on generated data with different σ as variance of coordinates



(a) Entropy of OD matrix through iterations for different σ^2

(b) Standard deviation of entropy through iterations for different σ^2

Figure 8: Entropy and standard deviation of entropy od OD matrix through iterations of gravity model on generated data with different σ as variance of coordinates

In the graphs obtained by plotting the calculated values, we saw how the dispersion of the distance of the coordinates affected the rate of convergence and entropy. From Figure 7a it was possible to observe that the convergence rate decreased with the increasing number of iterations. For $\sigma^2 = 10$, the rate of convergence was close to 0%, while the convergence rate for $\sigma^2 = 150$ after the first iteration has reached on average 60%. The greater dispersion between zones means the greater distance and cost of travel. People are therefore motivated to stay in their zone or travel only for short distances. Long-distance passenger numbers were penalised by the $f(c_{ij})$ function, so more iterations were needed to increase short and medium-distance passenger numbers to meet target passenger numbers. Therefore, adopting the matrix to this assumption was a longer process than for the matrix of zones, where the distances between zones were not so large and the value of $f(c_{ij})$ was almost the same for all zones. The graph of standard deviation of convergence in Figure 7b showed a very similar relationship. With increasing number of iterations the standard deviation was decreasing. For a larger dispersion of places, the standard deviation of convergence through simulations was larger.

In Figure 8a and Figure 8b we inspected the relationship between σ^2, entropy and iteration. The entropy increased at the start of the iteration process, and then it stabilised as the origin-destination matrix began to take on a certain ordered form. This behaviour was detected very clearly, especially in very remote zones, where travel prices were high. In that case, most of the population decided to stay in their zone and only a minimal number of people travelled. The matrix, therefore, had large values on the diagonal, while outside it, there were only small numbers. The standard deviation of entropy decreased a little and even though it looked like the numbers were huge, compared to the mean of entropy, the standard deviation was approximately 3%. These graphs presented the fact that the entropy of the OD matrix was maximised during the iterations of the gravity model.

2 Demon algorithm

2.1 Demon algorithm

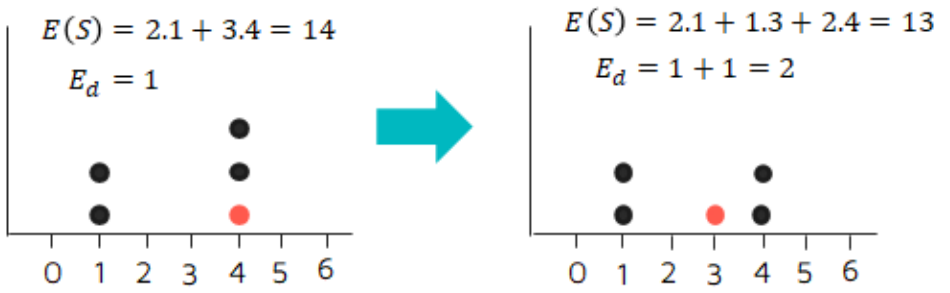
The demon algorithm is the Monte Carlo simulation method for efficient sampling of N elements of a microcanonical ensemble on energy levels with a given energy E. There is an additional degree of freedom called *the demon* that can store or provide energy E_d if needed. The demon itself can not create energy and can not have negative energy. Current energy of a system S is $E(S) = \sum_i \epsilon_i n_i$, where ϵ_i is energy of level i and n_i is number of elements on level i. [26] The demon algorithm:

1. Choose an initial configuration (state) S.

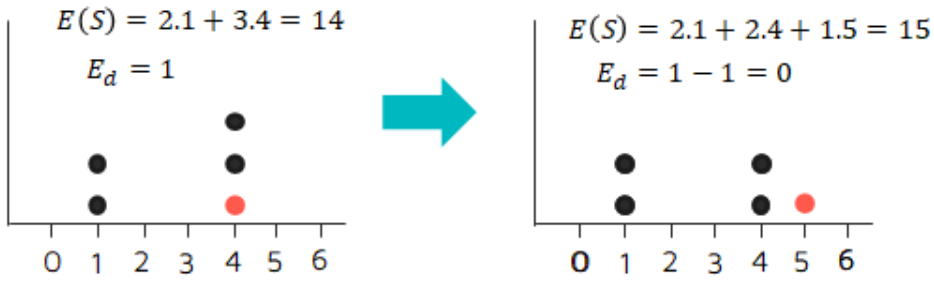
2. repeat

   - Perform a random change in the state of a randomly chosen particle (state S').
   - Let $\Delta E = E(S') - E(S)$.
   - If $\Delta E < 0$, then accept new configuration and excess energy $\mid \Delta E \mid$ is added to the demon.
   - If $\Delta E > 0$, then either the demon provides positive ΔE to keep the total energy constant (if it has sufficient energy $E_d > \Delta E$); or the randomly chosen change is rejected. The algorithm is then restarted from the original microscopic state.

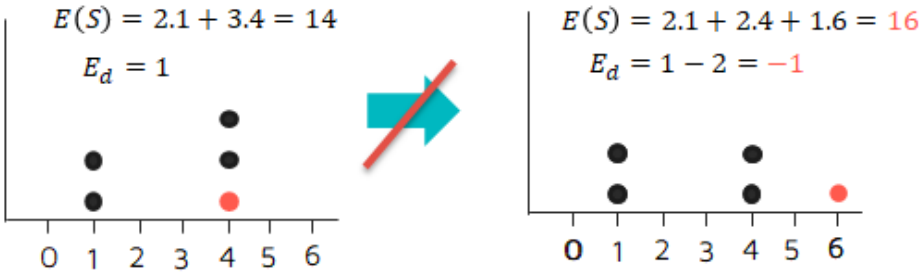
Graphical representation for better understanding of the algorithm is in Figure 9.



(a)



(b)



(c)

Figure 9: Example of steps in demon algorithm for system with $N = 5$ elements, 7 energy levels with $\epsilon_i = i$ and energy of system $E = 15$. In Figure 9a is depicted a possible change of system, when $\Delta E < 0$, the particle is relocated to the level with lower value of energy. In Figure 9b is a change, when the particle is relocated to the level with higher value of energy, therefore $\Delta E > 0$ and the demon has enough energy stored E_d to supplement the change of system. In Figure 9c, the element is relocated to the level with higher value of energy, but the demon does not have enough energy to supplement the change, therefore the element is put back to the previous level.

2.2 Simulations of the demon algorithm

We programmed this algorithm in Python and looked at the different layout of elements when changing the system energy and letting the number of elements and energy levels remain unchanged. The results can be seen in the graphs below. The number of elements was chosen as $N = 50$ and the number of energy levels as $M = 11$, where the energy of levels were integers between 0 and 10. The simulations were done for four scenarios with system energy being $100, 200, 300$ and 400. The results of the algorithm were obtained after 50000 iterations.

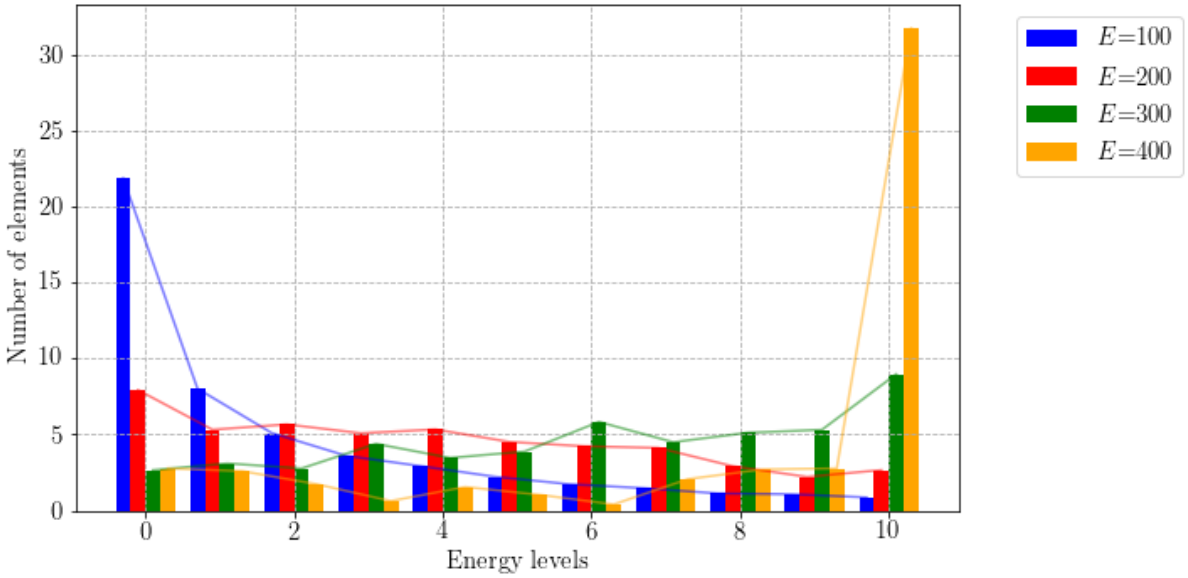


Figure 10: Distribution of elements on energy levels after 50000 iterations of demon algorithm for 4 values of energy system E, averaged over states with zero demon energy, i.e. the whole energy is stored in the system.

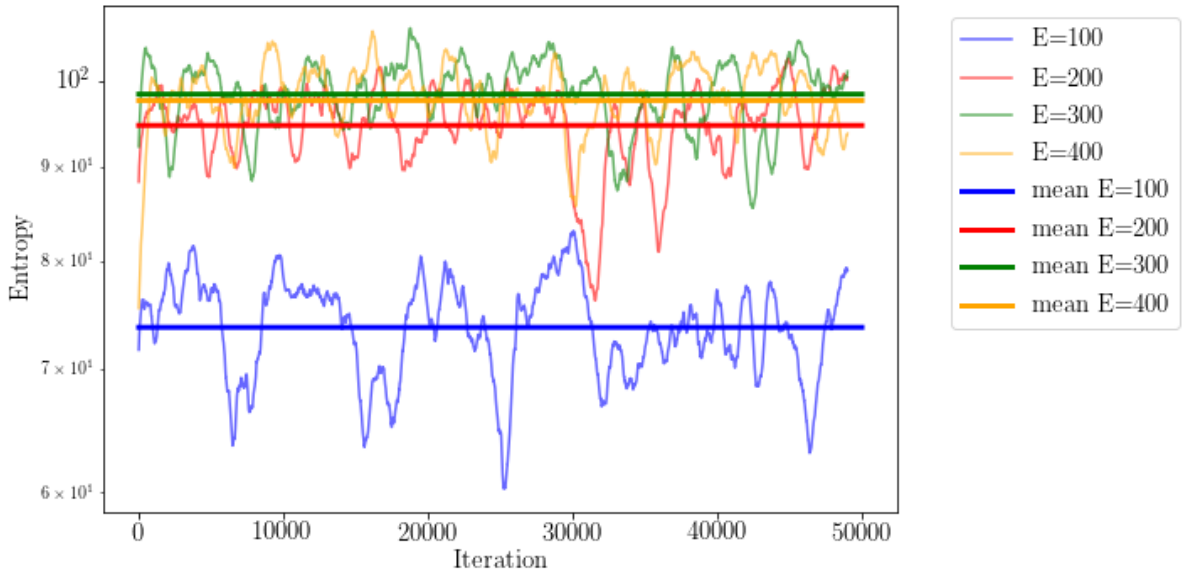


Figure 11: Evolution of average and moving average of entropy with window being 500 iterations, through 50000 iterations of demon algorithm for 4 values of energy system E.

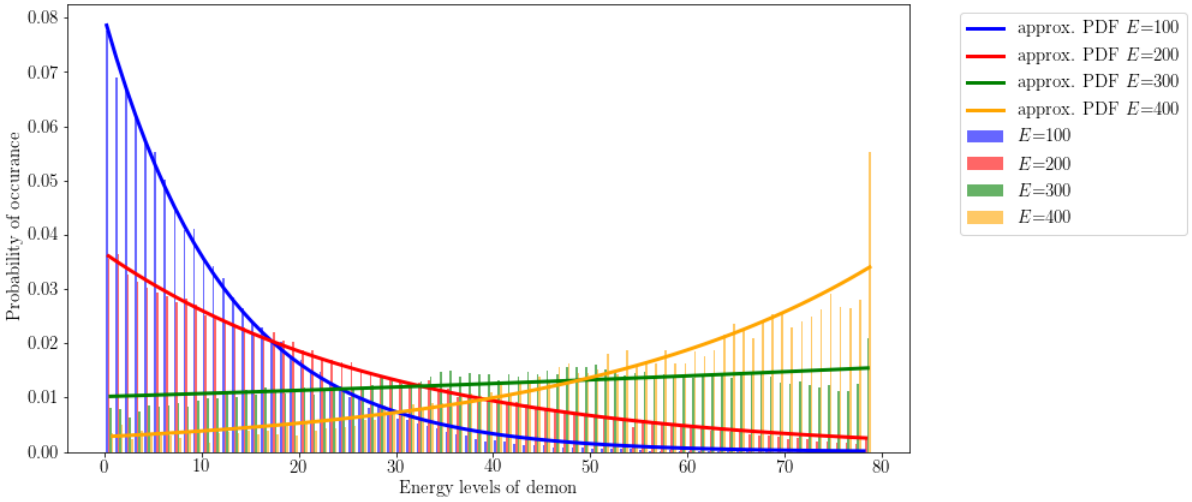


Figure 12: Probability distribution function and approximation of this function of demon from demon algorithm after 50000 iteration for 4 values of energy system E.

Figure 10 shows that the elements were arranged in exponential distribution for the system with low system energy. As the system energy grew ($E = 300$ or $E = 400$), the distribution of elements on levels changed to the exponential distribution of elements with negative temperature (i.e. negative exponent in exponential distribution).

Definition 2.1. *According to [11], the negative thermodynamic temperature is a temperature that can be expressed as a negative quantity on the Kelvin.*

Usually, system temperatures are positive. However, in isolated systems, the temperature defined in Boltzmann's entropy can become negative. This situation usually arises in a system with a bounded energy spectrum. As the temperature increases in such a system, particles move into higher energy levels. The number of particles in the lower and the higher energy levels approaches equality (this is the outcome of the definition of temperature in statistical mechanics for systems with limited states). A system with a negative temperature is not colder than absolute zero, but instead, it is hotter than infinite temperature.[11]

The evolution of each simulation's entropy is in Figure 11. For each scenario, the entropy fluctuated around some level for an extended period, so we could say there was maximum of entropy. The maximum of the system's entropy firstly grew with rising the energy of system. However, we can see that the average value of entropy for the system with energy $E = 400$ was slightly below the average value of entropy for the system with energy $E = 300$. This change was caused by the fact that to achieve the system energy equal to higher values like $E = 400$ most of the $N = 50$ elements had to be on levels with high energy values. Therefore there were fewer options of microstates for this scenario.

The distributions of demon and the approximation with exponential functions are displayed in Figure 12. Again as in the case of distribution of elements on levels, the system with low system energy had demon with exponential distribution. As the system energy grew ($E = 400$), the distribution of demon changed to exponential with a positive exponent, otherwise said, it changed to the exponential distribution with a negative exponent.

2.3 Extension of demon algorithm for generation of the origin-destination matrices

2.3.1 Initialization

First challenge in creating the algorithm for generation of OD matrices, was initialization of matrix, as a starting point of the demon algorithm. As the equations (4) must hold, we created the method with following steps:

1. Start with matrix of ones, denoted T^0 with size nxn, two empty sets denoted C and R, which correspond to the columns and rows where the condition (4) is already satisfied, set $[n] = \{k \in N, k \leq n\}$, select parameter of convergence $\xi \geq 0$ and $k = 0$.

2. Calculate the coefficient $m_1 = \frac{\min(O,D)}{2n}$ and matrix $T^1 = m_1 T^0$.

3. Take $k = k + 1$, calculate $r = (O - T^k\mathbf{1})$, and $c = (D - (T^k)^T\mathbf{1})$, where $\mathbf{1}$ represents column vector of ones with size nx1. If all elements $r \leq \xi$ and $c \leq \xi$, stop the algorithm. Otherwise continue to the next step.

4. Find the minimum value of vectors r and c, i.e $m_{k+1} = \min(r, c)$ and its position p_{k+1} in the vector, that is not in set C or R.

5. If the minimum was found in the vector r and if $p \in C$, than add the number $\frac{m_{k+1}}{n-|C|}$ to elements in the row p and columns $[n] \setminus C$ and create matrix T^{k+1}. If $p \notin C$, than add the value m_{k+1} to element T^k_{pp} and create matrix T^{k+1}. If the minimum was found in the vector c the procedure is the same, just change the set C for set R and rows for columns.

6. Add p to set R if m_{k+1} was found in vector r, otherwise add to set C and repeat step 3.

This matrix satisfied the condition (4) and does not have zero elements, which is useful for the Monte Carlo algorithm. We demonstrated this algorithm for calculating initial matrix, on simple example. Take O, vector of origins and D, vector of destinations, as

shown below and $\xi = 0.01$

$$O = \begin{pmatrix} 10 \\ 10 \\ 20 \end{pmatrix}, D = \begin{pmatrix} 5 \\ 10 \\ 25 \end{pmatrix}.$$

The sums of vectors is equal to 40 and therefore the condition (1) holds. The coefficient m in the step 2 is calculated as $\frac{\min(O,D)}{2n} = \frac{5}{6} \doteq 0.833$ so the matrix T^1 is

$$T^1 = \begin{pmatrix} \frac{5}{6} & \frac{5}{6} & \frac{5}{6} \\ \frac{5}{6} & \frac{5}{6} & \frac{5}{6} \\ \frac{5}{6} & \frac{5}{6} & \frac{5}{6} \end{pmatrix}.$$

The sum in each row was therefore $\frac{15}{6} = 2.5$. For $k = 1$, vector $r = \begin{pmatrix} \frac{45}{6} & \frac{45}{6} & \frac{105}{6} \end{pmatrix}$ and vector $c = \begin{pmatrix} \frac{15}{6} & \frac{45}{6} & \frac{135}{6} \end{pmatrix}$, which are clearly larger than the parameter ξ. The minimum is $m = \frac{5}{6}$ in the vector c, on the position 1. As sets C and R are empty we can add the number m to the element T^1_{11}, so matrix T^2 is

$$T^2 = \begin{pmatrix} \frac{20}{6} & \frac{5}{6} & \frac{5}{6} \\ \frac{5}{6} & \frac{5}{6} & \frac{5}{6} \\ \frac{5}{6} & \frac{5}{6} & \frac{5}{6} \end{pmatrix},$$

and we add the index 1 to set C, because the first column already satisfies the condition (4), so now $C = \{1\}$.

For $k = 2$, the vectors $r = \begin{pmatrix} 5 & 7.5 & 17.5 \end{pmatrix}$ and $c = \begin{pmatrix} 0 & 7.5 & 22.5 \end{pmatrix}$ are still clearly larger than ξ. The minimum is now $m = 5$, therefore $p = 1$, but $p \in C$. Now we add number $\frac{m}{n-|C|} = \frac{5}{(3-1)} = \frac{5}{2}$ to the elements in the first row and columns which indexes are not in set C, that is column 2 and 3. The set $R = \{1\}$ and matrix T^3 is

$$T^3 = \begin{pmatrix} \frac{20}{6} & \frac{20}{6} & \frac{20}{6} \\ \frac{5}{6} & \frac{5}{6} & \frac{5}{6} \\ \frac{5}{6} & \frac{5}{6} & \frac{5}{6} \end{pmatrix}.$$

For $k = 3$, the vectors $r = \begin{pmatrix} 0 & 7.5 & 17.5 \end{pmatrix}$ and $c = \begin{pmatrix} 0 & 5 & 20 \end{pmatrix}$ are still clearly larger than ξ. We would continue in this procedure for overall $2n$ steps (in our example until $k = 7$) and then all conditions (4) would be met.

2.3.2 Algorithm

This alternative algorithm for generation of OD matrices was inspired by the demon algorithm, the elements represent the travellers in our study area, the energy levels represent the distance or cost of travel and the energy of the system is replaced by the desired absolute length of trips chosen as λ. The calculation for the absolute length of trips in the OD matrix is

$$\lambda' = \sum_{ij} T_{ij} d_{ij}. \tag{22}$$

The algorithm then is:

1. Take $m = 0$ and an initial matrix T^m, chose number of iterations it.

2. If $m < it$ generate 4 natural numbers from 1 to n where n is the size of matrix T^m, noted as i, j, k, l.

3. Choose step s and perform a random change in the state. Calculate $T_{ij}^{m+1} = T_{ij}^m - s$ and $T_{il}^{m+1} = T_{il}^m + s$. To satisfy the conditions (4), $T_{kl}^{m+1} = T_{kl}^m - s$ and $T_{kj}^{m+1} = T_{kj}^m + s$.

4. Calculate λ^{m+1} and let $\Delta = \lambda - \lambda^{m+1}$.

5. If $\Delta > 0$, then accept new configuration, T^{m+1} and excess length Δ is added to the demon.

6. If $\Delta < 0$, then either the demon provides positive Δ to keep the absolute length of trips below setted value (if it has sufficient value $E_d > \Delta$). Otherwise the randomly chosen change is rejected and $T^{m+1} = T^m$. Take $m = m + 1$ and repeat step 3.

The algorithm is built to satisfies the condition (4) in every iteration, and through the demon, we can target the mean trip length, which is $\frac{\lambda}{\sum_{ij} T_{ij}}$. By averaging the matrix over states with demon energy lower than the given threshold, we approximate the matrix with the maximal entropy.

T_{11}	T_{12}	T_{13}	T_{14}	T_{15}	O_1
T_{21}	T_{22}	T_{23}	T_{24}	T_{25}	O_2
T_{31}	T_{32}	T_{33}	T_{34}	T_{35}	O_3
T_{41}	T_{42}	T_{43}	T_{44}	T_{45}	O_4
T_{51}	T_{52}	T_{53}	T_{54}	T_5	O_5
D_1	D_2	D_3	D_4	D_5	

(a) Generate two numbers i, j, for example 2 and 1.

T_{11}	T_{12}	T_{13}	T_{14}	T_{15}	O_1
$T_{21} - s$	T_{22}	T_{23}	T_{24}	T_{25}	$O_2 - s$
T_{31}	T_{32}	T_{33}	T_{34}	T_{35}	O_3
T_{41}	T_{42}	T_{43}	T_{44}	T_{45}	O_4
T_{51}	T_{52}	T_{53}	T_{54}	T_5	O_5
$D_1 - s$	D_2	D_3	D_4	D_5	

(b) Reduce the element in i^{th} row and j^{th} column by number s, where $s \leq T_{ij}$. This change disrupts the sums of i^{th} row and j^{th} column.

T_{11}	T_{12}	T_{13}	T_{14}	T_{15}	O_1
$T_{21} - s$	T_{22}	T_{23}	$T_{24} + s$	T_{25}	O_2
T_{31}	T_{32}	T_{33}	T_{34}	T_{35}	O_3
T_{41}	T_{42}	T_{43}	T_{44}	T_{45}	O_4
T_{51}	T_{52}	T_{53}	T_{54}	T_5	O_5
$D_1 - s$	D_2	D_3	$D_4 + s$	D_5	

(c) Generate another number k, for example 4. Increase the element T_{ik}. This change corrects the sum
in the i^{th} row but disrupts the sum in the k^{th} column.

T_{11}	T_{12}	T_{13}	T_{14}	T_{15}	O_1
$T_{21} - s$	T_{22}	T_{23}	$T_{24} + s$	T_{25}	O_2
T_{31}	T_{32}	T_{33}	T_{34}	T_{35}	O_3
T_{41}	T_{42}	T_{43}	T_{44}	T_{45}	O_4
T_{51}	T_{52}	T_{53}	$T_{54} - s$	T_5	$O_5 - s$
$D_1 - s$	D_2	D_3	D_4	D_5	

(d) Generate another number l, for example 5. Reduce the element T_{lk} by number s, where $s \leq T_{lk}$. This change corrects the sum in the k^{th} column but disrupts the sum in the l^{th} row.

T_{11}	T_{12}	T_{13}	T_{14}	T_{15}	O_1
$T_{21} - s$	T_{22}	T_{23}	$T_{24} + s$	T_{25}	O_2
T_{31}	T_{32}	T_{33}	T_{34}	T_{35}	O_3
T_{41}	T_{42}	T_{43}	T_{44}	T_{45}	O_4
$T_{51} + s$	T_{52}	T_{53}	$T_{54} - s$	T_5	O_5
D_1	D_2	D_3	D_4	D_5	

(e) Increase the element T_{lj}. This change corrects the sums in the j^{th} column and k^{th} row. All conditions are now met and the matrix is changed.

Figure 13: Scheme of the extension of demon algorithm for generion of OD matrices.

2.4 Simulations of the extended demon algorithm

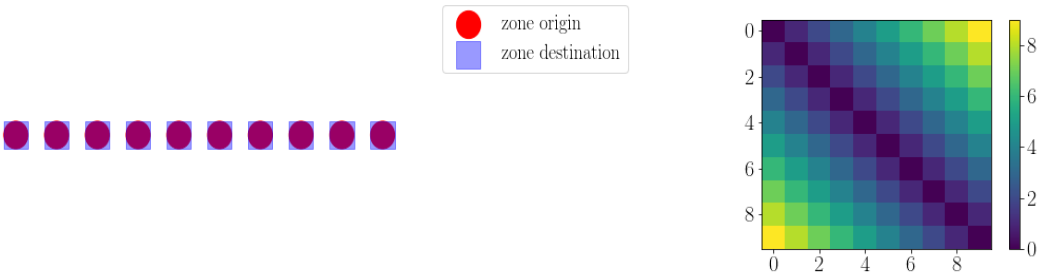
To see whether this algorithm is efficient, we decided to test it and compare it to the gravity model results. We reduced this problem by assuming that people decide to travel only according to distance. Parameter c_{ij} in the gravity model was therefore assumed to be d_{ij}, which is a distance between zone i and zone j. We used the exponential deterrence function. To secure that the condition for the mean trip length (i.e. $\frac{\lambda}{\sum_{ij} T_{ij}}$) in the OD matrix generated by the gravity model was met, we used the iterative method for the calculation of parameter β. This algorithm was repeated until the difference between the mean trip length in OD matrix $\hat{d}$ and desired mean trip length d^* was less than $\frac{d^*}{10000}$.

The results of the Monte Carlo simulation were calculated with 1000000 start iteration, which were not considered for averaging and 2000000 where we observed matrix T. During these iterations, the step was adjustable during the process, so the acceptance ratio would be around 0.85. All 2000000 iterations were considered to be averaged if the condition that demon value $E_d < \frac{\lambda}{25}$ was met, and the averaged matrix was considered as a result of this algorithm. To simplify the process of calculation, we have chosen the number of zones to be 10.

2.4.1 Constant population and constant spacing between zones

In this case, the vectors of origin and destination were equal to the constant vector of 1000. We looked at three different scenarios, requiring the mean trip length being 2, 2.5 and 3.

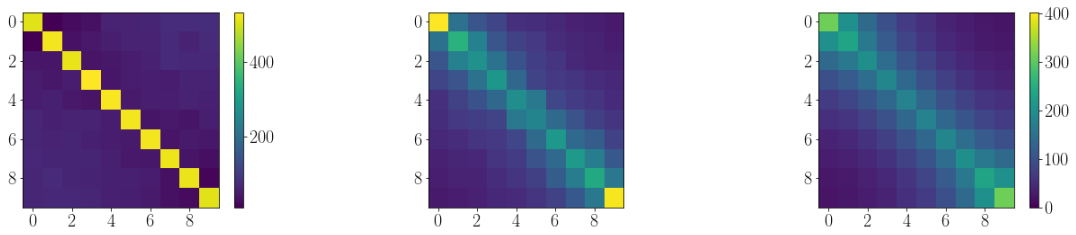
The spacing of the zones was constant on the straight line and with distance of zones equal to 1. Therefore the map of zones would look like the one below, the size of the circle and square representing the zone's population. The distance matrix was calculated as Euclidean distance between all zones, and its shape could be observed as a pseudocolour image.



(a) **(b)**

Figure 14: In Figure 14a is map of zones with constant spacing of one and constant population being 1000. In Figure 14b is matrix of distances for zones with constant spacing of 1.

For the first option of the mean trip length, the initial matrix obtained by algorithm discussed in chapter 2.3.1 is displayed in Figure 15a. In the middle, the Figure 15b shows the matrix obtained by Monte Carlo simulation, averaging matrices where the value of demon was lower than 800. The last of the three figures is a matrix obtained by the gravity model.



(a) **(b)** **(c)**

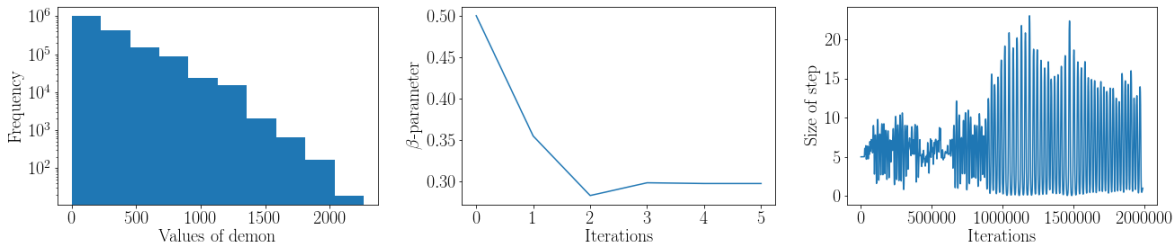
Figure 15: In Figure 15a is initial matrix for the Monte Carlo algorithm in case of constant population and constant spacing of zones. In Figure 15b is OD matrix generated by Monte Carlo simulation after 2000000 iterations. In Figure 15c is OD matrix generated by the gravity model with the mean trip length being 2.

To evaluate the algorithm's efficiency for this scenario, we had to look at numeric evaluations of methods. From results in Table 2, the matrix generated by the Monte Carlo simulation had higher variance, and the maximum element was larger than the one generated by the gravity model. On the other hand, the entropies of the matrices were similar. The ratio of the gravity model entropy to the Monte Carlo simulation entropy

was 1.0042, the convergence rate was, therefore, 0.42%, which could be sufficient if the matrices were similar. Looking at the mean value of absolute differences, which is 12.23, we can not say these matrices are alike. The relative difference is a little bit smaller, being 0.1209. However, it was larger than 10%; therefore, we do not consider this an appropriate example of the convergence of this method.

	Monte Carlo	Gravity model
Mean trip length	1.99	2.00
Entropy	-38062.81	-37902.43
Max element	403.23	314.92
Min element	30.46	21.71
Average element	100.00	100.00
Standard deviation	69.68	62.77

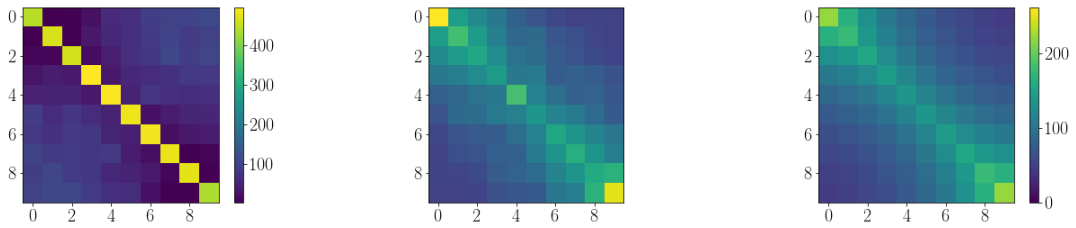
Table 2: Comparison of results of the Monte Carlo simulation and gravity model generating OD matrix in scanario with constant population of zones, constant spacing of zones being 1 and the mean trip length being 2.



(a) **(b)** **(c)**

Figure 16: In Figure 16a is distribution of demon from Monte Carlo simulation of OD matrix for zones with constant spacing and constant population. In Figure 16b is development of parameter β in gravity model. In Figure 16c is development of moving average over 10000 iterations of the step in Monte Carlo simulation for matrix with the mean trip length being 2

For the second example, when the mean trip length was set to be 2.5, the distribution of zones, distance matrix and population was the same. Figure 14a and Figure 14b are therefore relevant to this case. The initial matrix, OD matrix generated by Monte Carlo simulation and averaged over iterations where the value of demon was less than 1000 and matrix generated by gravity model are displayed in the figures below.



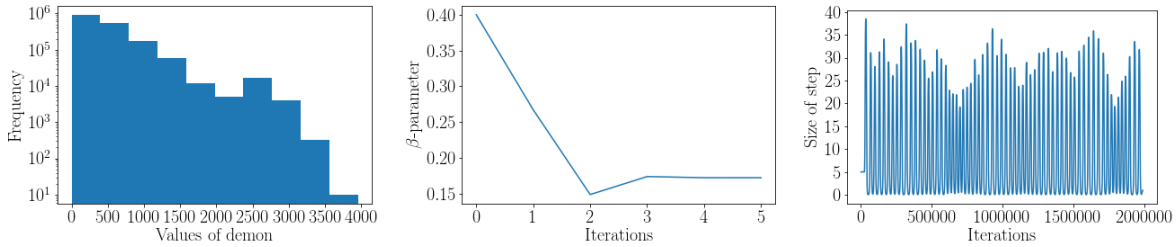
(a) **(b)** **(c)**

Figure 17: In Figure 17a is initial matrix for the Monte Carlo algorithm in case of constant population and constant spacing of zones. In Figure 17b is OD matrix generated by the Monte Carlo simulation after 2000000 iterations. In Figure 17c is OD matrix generated by the gravity model with the mean trip length being 2.5.

Again to evaluate the algorithm's efficiency for this scenario, we had to look at numeric evaluations of methods. From Table 3, the matrix generated by the Monte Carlo simulation is much more comparable to the matrix generated by the gravity model than in the previous example. There were slight differences in entropies, but the ratio is 1.0016, so we could neglect the difference for such large numbers. The mean value of absolute differences, which was 6.42, proves that the Monte Carlo simulation was a sufficient approximation of the gravity model for this scenario.

	Monte Carlo	Gravity model
Mean trip length	2.49	2.50
Entropy	-36798.20	-36738.52
Max element	261.88	220.06
Min element	50.57	46.57
Average element	100.00	100.00
Standard deviation	40.45	37.86

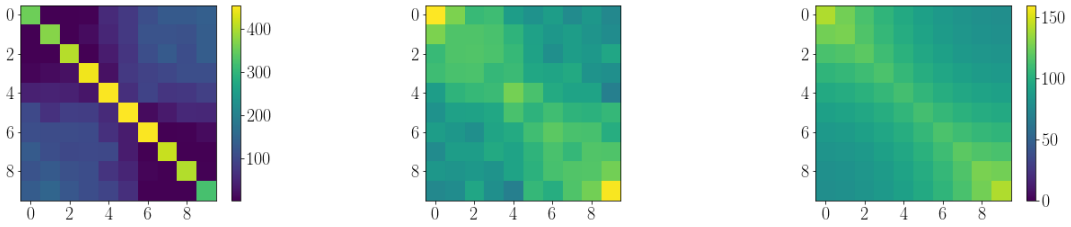
Table 3: Comparison of results of the Monte Carlo simulation and the gravity model generating OD matrix for scanario with constant population of zones, constant spacing of zones being 1 and the meant trip length being 2.5.



(a) **(b)** **(c)**

Figure 18: In Figure 18a is distribution of demon from the Monte Carlo simulation of OD matrix for zones with constant spacing and constant population. In Figure 18b is the development of parameter β in gravity model. In Figure 18c is development of moving average over 10000 iterations of the step in Monte Carlo simulation for matrix with the mean trip length being 2.5

As the last example in this setting of zones, i.e. constant population of 1000 and constant spacing between zones, we have chosen to perform the simulation for the desired mean trip length equal to 3. The matrices in the Monte Carlo simulation were averaged if the value of the demon was less than 1200.



(a) **(b)** **(c)**

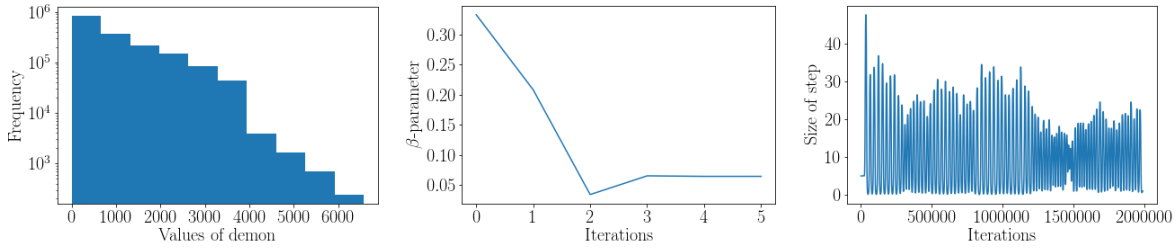
Figure 19: In Figure 19a is initial matrix for the Monte Carlo algorithm in case of constant population and constant spacing of zones. In Figure 19b is the OD matrix generated by the Monte Carlo simulation after 2000000 iterations. In Figure 19c is the OD matrix generated by the gravity model with the mean trip length being 3.

Again, as in the second example with the mean trip length being 2.5, we could see that the Monte Carlo simulation was a sufficient approximation of the gravity model. The ratio of entropies was 1.0011, which means that the convergence was 0.11%, the standard deviation of elements in matrices was close, the same goes for maximum and minimum values. The mean value of absolute differences was 5.40, so we could say these matrices were alike.

	Monte Carlo	Gravity model
Mean trip length	2.98	3.00
Entropy	-36189.16	-36148.85
Max element	159.60	140.44
Min element	78.67	69.12
Average element	100.00	100.00
Standard deviation	16.77	14.08

Table 4: Comparison of results of the Monte Carlo simulation and the gravity model generating OD matrix for scanario with constant population of zones, constant spacing of zones being 1 and the mean trip length being 3.

It can seem from these examples that as the mean trip length decreases, the method's accuracy decreases too. When the mean trip length is small, many zeros occur in the origin-destination matrix because people choose to stay at home or travel only to the



(a) **(b)** **(c)**

Figure 20: In Figure 20a is the distribution of the demon from Monte Carlo simulation of OD matrix for zones with constant spacing and constant population. In Figure 20b is development of parameter β in the gravity model. In Figure 20c is development of moving average over 10000 iterations of the step in Monte Carlo simulation for matrix with the mean trip length being 3.

nearest zone. The Monte Carlo simulation has a problem with creating zero elements in a matrix, therefore straggles to maximize the entropy. For example, let the mean trip length be 1. Monte Carlo simulation generates the diagonal matrix with the mean trip length of 0.98, which is close enough to the desired value. The minimum element of the matrix is almost 13, compared to the minimum element of the matrix generated by the gravity model being only 0.74. This produces smaller entropy of the Monte Carlo simulation matrix. The ratio of entropies is 1.0361, which is high compared to the previous examples. What is even more alarming is that the average absolute difference of matrices is around 46 and the relative difference is 1.66.

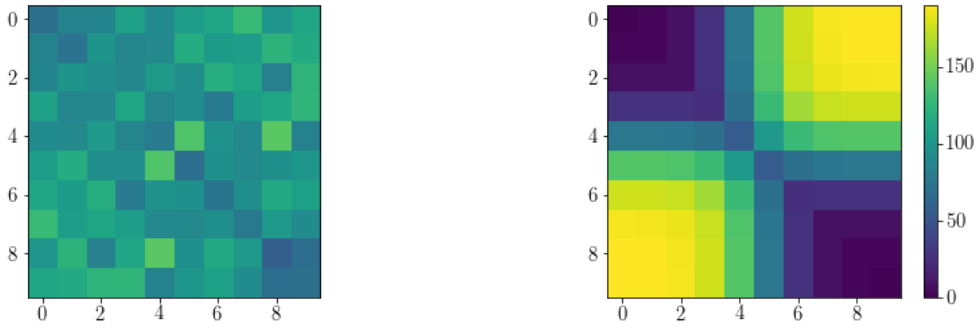
Nevertheless, if we choose to select an even smaller mean trip length, close to zero, meaning almost no one is travelling, we can determine that the gravity model fails. Let us take the mean trip length equal to 0.1. When we took a closer look at the algorithm for parameter β in chapter 1.4 we saw, that the difference of costs of travelling from the following iterations is in the denominator,

$$\beta_{m+1} = \frac{(c^* - c_{m-1})\beta_m - (c^* - c_m)\beta_{m-1}}{c_m - c_{m-1}}.$$

When the difference of costs was small, but the difference from desired mean trip length was large, β took on a large value. In this case, in the fourth iteration, β became negative, around -763, because the argument in the second parentheses was greater than the first one. Generalized function of travel cost therefore should have been $f(c_{ij}) = e^{765c_{ij}}$, which was too large for Python to calculate. The gravity model stopped at this point, generated

the matrix with 999 on the diagonal, and created the mean trip length of 0, far from the desired value. On the other hand, the Monte Carlo simulation generated the matrix with the mean trip length of 0.098, which was better performance than the gravity model.

Setting a high value for the mean trip length leads to degeneration of Monte Carlo simulation. For value 4.7, the gravity model generated a matrix where almost no one wants to stay at their origin, and the majority of the population travelled. On the other hand, Monte Carlo generated an almost constant matrix with elements around 100, and the mean trip length of the matrix was just 3.4.

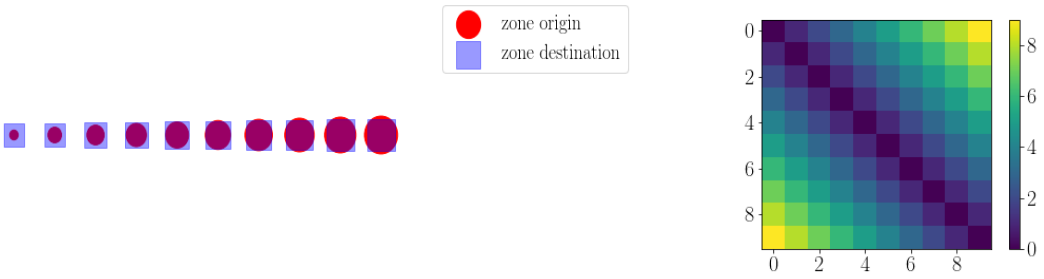


(a) **(b)**

Figure 21: In Figure 21a is OD matrix generated by the Monte Carlo simulation after 2000000 iterations. In Figure 21b is OD matrix generated by the gravity model with the mean trip length being 4.7.

2.4.2 Linear growth of population and constant spacing between zones

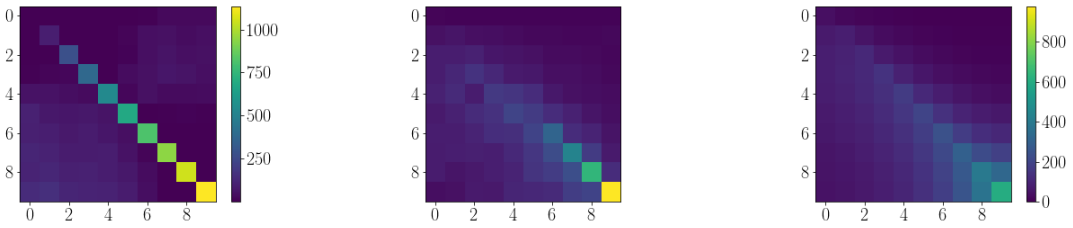
To evaluate the strength and weaknesses of the algorithm, we decided to compute the origin-destination matrices for another scenario, this time with linear growth of a population of zones. This time the vectors of origin and destination are not equal, meaning that the number of people leaving and coming to the zone are not the same, which is a more realistic scenario. We looked at two different mean trip lengths.



(a) (b)

Figure 22: In Figure 22a is map of zones with constant spacing of one on straight line and liner growth of origin and destination. In Figure 22b is a distance matrix for zones with constant spacing of 1.

We started with setting the parameter of the mean trip length to 2. The initial matrix obtained by the algorithm is displayed in Figure 23a, along with the matrix obtained by Monte Carlo simulation, averaging matrices where the value of demon was lower than 800 in Figure 22b. The last of the three figures is the matrix obtained by the gravity model.



(a) (b) (c)

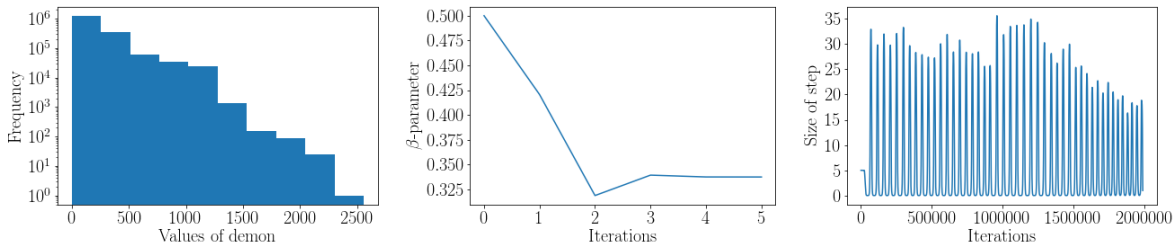
Figure 23: In 23a is initial matrix for Monte Carlo algorithm in case of linear growth of population and constant spacing of zones. In Figure 23b is OD matrix generated by the Monte Carlo simulation after 2000000 iterations. In Figure 23c is OD matrix generated by the gravity model with the mean trip length being 2.

Again, as in the scenario with the constant population, we displayed obtained results in Table 5, showing that the matrix generated by the Monte Carlo simulation had higher variance and the maximum element was larger than the one generated by the gravity model. Even the ratio of entropies was less similar than in previous examples, being 1.0159. The convergence rate was, therefore, 1.59%. Further, the absolute average difference, which was 27.93, was too large to say that these matrices were alike. The relative

difference was even larger 0.52. Again this was not a sufficient approximation of the gravity model.

	Monte Carlo	Gravity model
Mean trip length	1.98	2.00
Entropy	-40566.25	-39930.21
Max element	979.35	608.05
Min element	8.41	1.14
Average element	100.00	100.00
Standard deviation	124.77	95.52

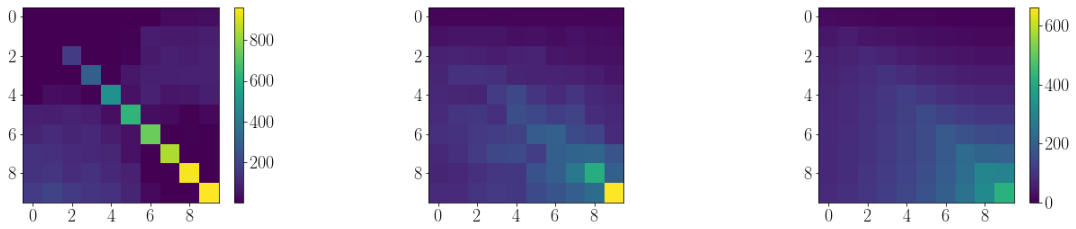
Table 5: Comparison of results of Monte Carlo simulation and Gravity model generating OD matrix in scanario of linear growth of population, constant spacing of zones being 1 and the mean trip length being 2.



(a) **(b)** **(c)**

Figure 24: In Figure 24a is distribution of the demon from Monte Carlo simulation of OD matrix for zones with constant spacing and linear growth of population. In Figure 24b is development of parameter β in the gravity model. In Figure 24c is development of the moving average over 10000 iterations of the step in Monte Carlo simulation for matrix with the mean trip length being 2.

For the following example, the mean trip length was 2.5. The initial matrix obtained by the algorithm is displayed in Figure 25a, along with the matrix obtained by the Monte Carlo simulation in Figure 25b. We averaged matrices, where the value of demon was lower than 1000. The last of three figures is the matrix obtained by the gravity model.



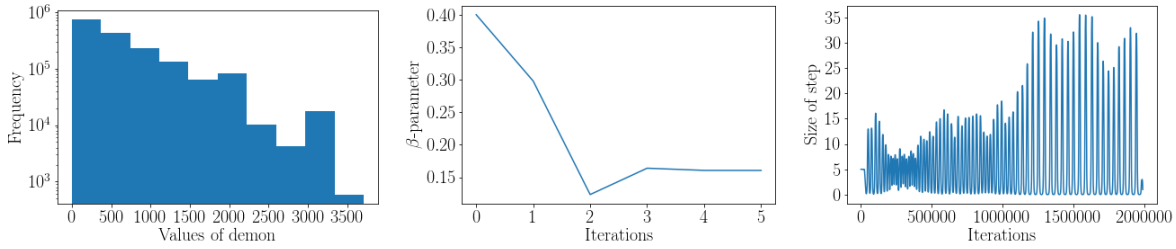
(a) **(b)** **(c)**

Figure 25: In Figure 25a is initial matrix for the Monte Carlo algorithm in case of linear growth of population and constant spacing of zones. In Figure 25b is OD matrix generated by Monte Carlo simulation after 2000000 iterations. In Figure 25c is OD matrix generated by the gravity model with the mean trip length being 2.5.

Table 6 showed results for the matrix with mean trip length being 2.5. This approximation is better, but it was hardly enough. The average absolute difference of matrices was 16.97, and the relative difference was 0.1719, which is still larger than 10%.

	Monte Carlo	Gravity model
Mean trip length	2.43	2.50
Entropy	-38977.96	-38667.24
Max element	663.78	424.75
Min element	8.86	5.54
Average element	100.00	100.00
Standard deviation	87.40	74.72

Table 6: Comparison of results of the Monte Carlo simulation and the gravity model generating OD matrix in scanario with the linear growth of population of zones, constant spacing of zones being 1 and the mean trip length being 2.5.



(a) **(b)** **(c)**

Figure 26: In Figure 26a is distribution of the demon from the Monte Carlo simulation of OD matrix for zones with constant spacing and linear growth of population. In Figure 26b is development of parameter β in the gravity model. In Figure 26c is development of moving average over 10000 iterations of the step in Monte Carlo simulation for matrix with the mean trip length being 2.5

The linear growth population scenarios were examples of the situations when we were not satisfied with the results of the Monte Carlo simulation. We will see that with the growing size of the matrix, i.e. growing number of zones, the accuracy of simulation decreases rapidly.

These examples concluded that the Monte Carlo simulation could be a sufficient approximation of the gravity model, but only for specific mean trip lengths and certain scenarios. For minimal values of the mean trip length, Monte Carlo outperforms the gravity model. However, for large values of the mean trip length, its performance is not satisfactory. The convex optimization introduced in section 3 performs well for both extreme cases.

3 Application of convex optimisation

3.1 Exponential cone programming

The problem (17) stated in section 1.6, with the additional constraint (19) is actually a convex problem. It can be written as

$$\begin{aligned} \min \sum_{ij} T_{ij} \ln T_{ij} \; - T_{ij} & \\ O_i - \sum_j T_{ij} = 0 \qquad & i = 1, \ldots, n \\ D_j - \sum_i T_{ij} = 0 \qquad & j = 1, \ldots, n \\ C - \sum_{ij} T_{ij} c_{ij} = 0. & \end{aligned}$$

We decided to rewrite this problem into vector form, so the vector τ will represent

$$\tau = \begin{pmatrix} T_{11} \\ T_{12} \\ T_{13} \\ \vdots \\ \vdots \\ T_{nn} \end{pmatrix} = \begin{pmatrix} \tau_1 \\ \tau_2 \\ \vdots \\ \tau_k \\ \vdots \\ \tau_{n^2} \end{pmatrix}.$$

The problem is then

$$\begin{aligned} \min \tau^T \ln \tau - \mathbf{1}^T \tau, & \\ b - A\tau = 0_{2n+1}, & \end{aligned} \tag{23}$$

where b is a vector $b = \begin{pmatrix} O & D & C \end{pmatrix}^T$ of size $(2n+1) \times 1$ and matrix A is matrix of size $(2n+1) \times n^2$ constructed by 3 parts

$$A = \begin{pmatrix} A_1 \\ A_2 \\ A_3 \end{pmatrix},$$

where

$$A_1 = \begin{pmatrix} 1 & \dots & 1 & 0 & \dots & \dots & \dots & \dots & 0 \\ 0 & \dots & 0 & 1 & \dots & 1 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & \dots & \dots & \dots & 0 & 1 & \dots & 1 \end{pmatrix},$$

$$A_2 = \begin{pmatrix} 1 & 0 & \dots & 0 & 1 & 0 & \dots & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & 0 & \dots & 0 & 1 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix},$$

$$A_3 = \begin{pmatrix} c_{11} & c_{12} & c_{13} & \dots & c_{nn} \end{pmatrix}.$$

It should be stated that by $\ln \tau$, we mean that we calculate the logarithm of each element of vector τ separately. Therefore the result is the vector of the same size.

This convex optimization can be represented as an exponential cone programming. The idea of transformation is that firstly, we rewrite the problem, so that the objective function is linear,

$$\begin{aligned} \min \sum_{i=1}^{n^2} t_i, \\ \tau_i \ln \tau_i - \tau_i, \leq t_i \\ b - A\tau = 0_{2n+1}, \\ -\tau_i < 0. \end{aligned}$$

The inequalities can be than nested in exponential cone through series of following steps [21]

$$\tau_i \ln \tau_i - \tau_i \leq t_i \Leftrightarrow -\tau_i \ln \tau_i + \tau_i \geq -t_i \Leftrightarrow -\ln \tau_i + 1 \geq -\frac{t_i}{\tau_i}$$
$$-\ln \tau_i + 1 \geq -\frac{t_i}{\tau_i} \Leftrightarrow \ln \frac{1}{\tau_i} + \ln e \geq -\frac{t_i}{\tau_i} \Leftrightarrow \frac{e}{\tau_i} \geq e^{-\frac{t_i}{\tau_i}} \Leftrightarrow e \geq \tau_i e^{-\frac{t_i}{\tau_i}}$$
$$e \geq \tau_i e^{-\frac{t_i}{\tau_i}} \Leftrightarrow (-t_i, \tau_i, e) \in K_{exp}.$$

Therefore the problem (23) can be transformed to exponential cone programming

$$\begin{aligned} &\min \sum_{t=1}^{n^2} t_i, \\ &A\tau = b, \\ &(-t_i, \tau_i, e) \in K_{exp}, \quad i = 1, \ldots, n^2 \end{aligned} \tag{24}$$

and this type of problems can be solved through Python library cvxpy [8] with the use of ECOS solver [9], which uses primal-dual interior-point method.

3.2 Primal-dual interior-point method

The primal-dual interior-point method is used for solving convex optimization problems that include inequality constraints. The goal is to reformulate the inequality constrained problem as a series of smooth convex unconstrained problems that approximate the original problem and can be solved by Newton's method. The first step is to rewrite the problem, making the inequality constraints implicit in the objective function using the barrier function, which is differentiable. Then the following step is the formulation of conditions for points on the central path (i.e. solutions for the series of unconstrained problems), which can also be interpreted as a continuous deformation of the Karush–Kuhn–Tucker (KKT) optimality conditions for the original problem. The last step is the process of finding the primal-dual search direction from Newton's method. The approximation of the problem, calculation of direction, step and the optimal solution are performed until the duality gap is smaller than the desired threshold. A detailed description of the general primal-dual interior-point method can be found in [2, page 609]

The primal-dual interior-point algorithms for conic programming is a complicated process. To fully describe it, we would need to introduce the reader to terms like homogeneous embedding, self-dual embedding, symmetric and nonsymmetric cone, the functional proximity measure or Nesterov-Todd scaling points. Because the explanation of such a complex technique is out of the scope of this work, we will not describe the process of finding the optimal solution, which was derived in [19, page 83] and [7].

We will only state that the (24) problem can be transformed into standard form

$$\begin{aligned} \min c^T x, \\ A'x = b', \\ Gx + s = h, \\ s \in K_{exp}, \end{aligned} \tag{25}$$

where $x^T = \begin{pmatrix} t_1 & \dots & t_{n^2} & \tau_i & \dots & \tau_{n^2} \end{pmatrix}$, matrices A', G and vectors b', h and s take such forms, so the (24) and (25) are equivalent. The dual linear conic problem to (25) is than

$$\begin{aligned} \max -b^T y - h^T z, \\ G^T z + A'^T y + c = 0, \\ z \in K^*, \end{aligned} \tag{26}$$

where K^* is a dual cone to K_{exp}, which means that $K^* = \{z | \langle z, s \rangle \geq 0, s \in K_{exp}\}$.

For this conic problem, the solver ECOS uses the barrier method

$$H^*(u, v, w) = -\ln \left(w - u - u \ln \frac{-v}{u} \right) - \ln -u - lnv \tag{27}$$

The linear constraints of the homogeneous embedding can be derived from the primal and dual problem. The barrier problem arises from replacing the conic constraints with the barrier functions to define a linearly constrained convex optimization problem. Because the ECOS solver uses the Mehrotra predictor-corrector method, it firstly calculates centuring directions, solves for the affine search direction and, with these results, solves the combined second-order search directions. The algorithm and its derivation can be found in [19].

4 Comparison of explored methods

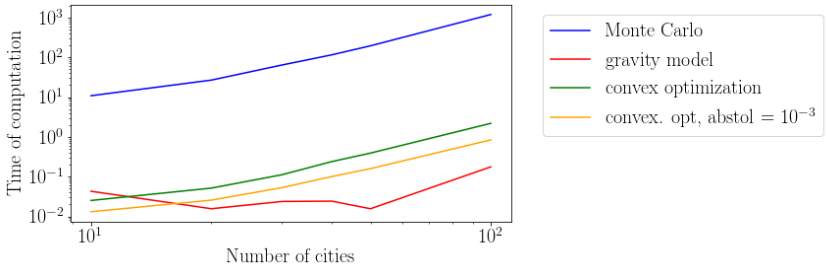
4.1 Comparison of alternative methods to the gravity model according to accuracy and time complexity

In previous sections, we described three methods of generation of origin-destination matrices, and that are iterative gravity model, Monte Carlo simulation and convex optimization using primal-dual interior-point method. In this section, we wanted to compare the time and accuracy of the results for different scenarios.

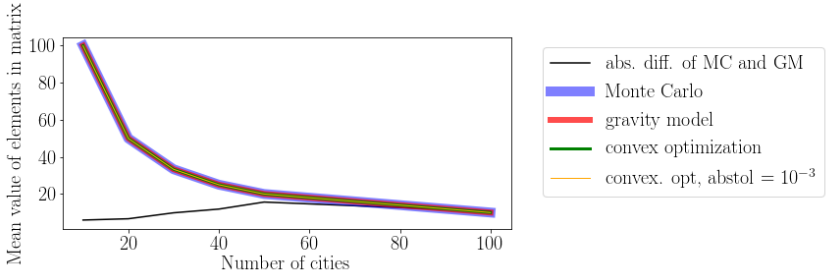
The first one was constant population and constant spacing between zones. The desired mean trip length was chosen as the square root of the number of zones. We computed the time, the mean element of the matrix, the mean trip length in the matrix and entropy of matrix for all methods.

It is precise from Figure 27a that the most time-efficient method is the gravity model. A good substitute could be a convex optimization or convex optimization with more minor but still sufficient accuracy being 10^{-3}. The mean element of matrix in Figure 27b is the same for all methods. However, for comparison, there is also the absolute average difference of matrix generated by Monte Carlo simulation and matrix generated by gravity model, which grows on to the mean value of the matrix. This means that on average the matrices differ in every element. The mean trip length is kept to the desired trip length. Figure 27d shows that the matrices generated by the Monte Carlo simulation have significantly lower entropies than the matrices from the other algorithms.

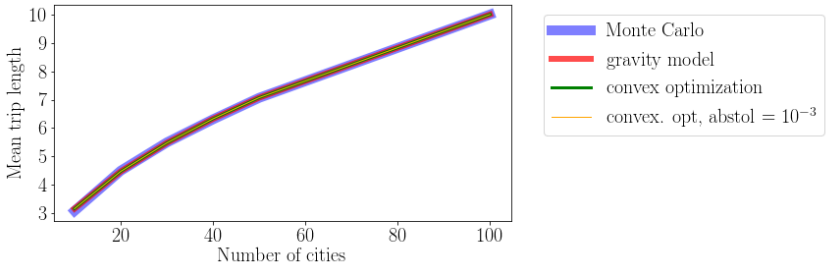
We decided to take a closer look at the systems with the higher number of zones, but only for gravity model and convex optimizations since it would take very long time for Monte Carlo simulation to calculate it.



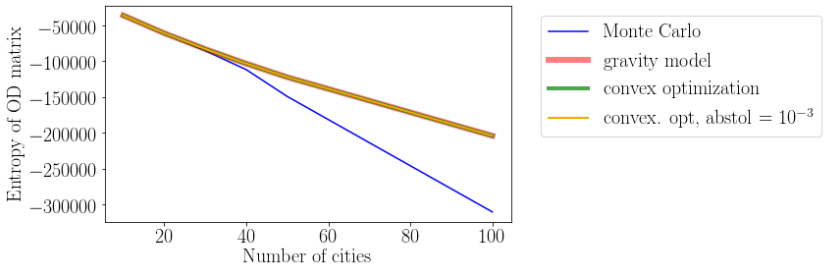
(a)



(b)



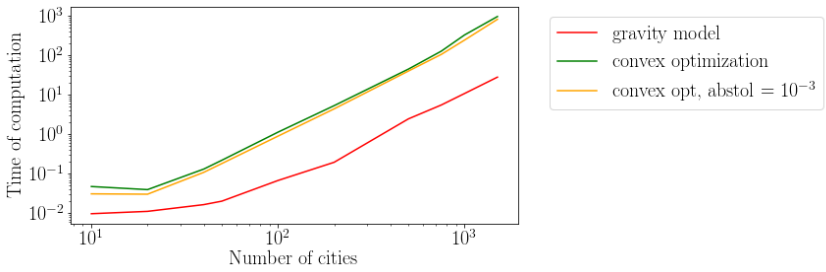
(c)



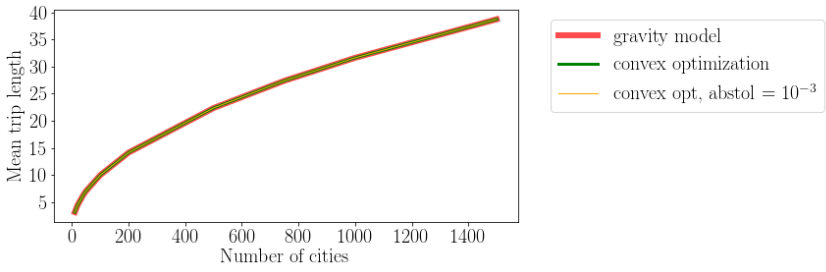
(d)

Figure 27: Comparison of 3 methods ussed on the same scanario of constant population and constant spacing of zones. Figure 27a is the comparison of time of computation for different size of system, Figure 27b is comparison of the mean values of matrices. In Figure 27c we can see the comparison of mean trip length and in 27d is comparison of entropies.

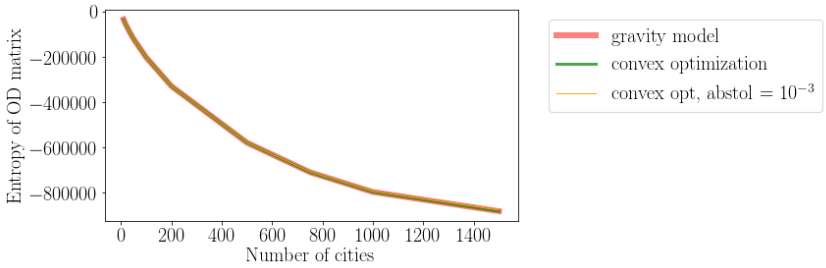
.



(a)



(b)

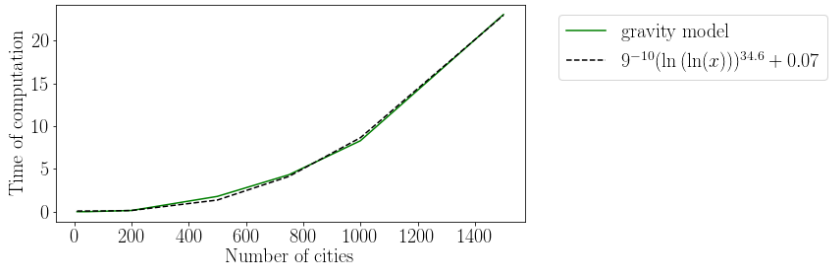


(c)

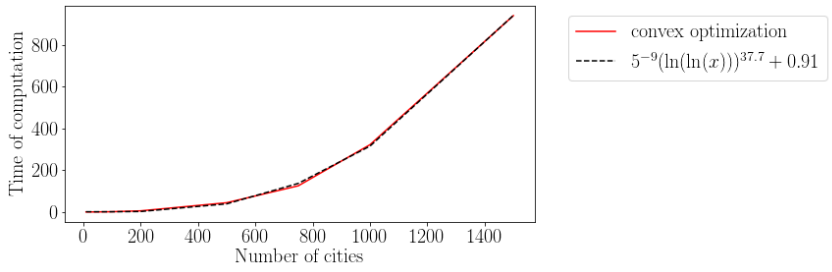
Figure 28: Comparison of 2 methods on the same scanario of constant population and constant spacing of zones for systems with large number of zones in the system. Figure 28a is the comparison of time of computation for different sizes of system. In Figure 28b we can see the comparison of mean trip length and in Figure 28c is comparison of entropies.

Here we see that the function of time given number of zones is not linear in log-log scale; therefore, it is not a power function. For this case, we see that the convex optimization still nicely copies the gravity model. The computation time can not be lowered anymore because after setting the feasible tolerance lower than 10^{-3}, the ECOS solver fails to find the optimal matrix because the problem becomes infeasible. Therefore we have to put up with slower and more accurate algorithm, even though we do not need such high accuracy.

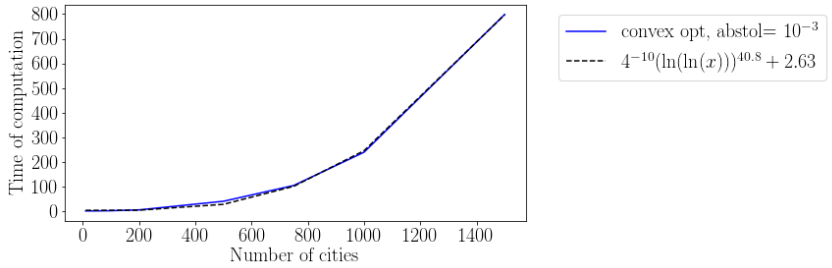
The time of computation is estimated to be a power function of logarithm of logarithm of number of zones. The approximation of the functions for each method can be seen in figures below.



(a)



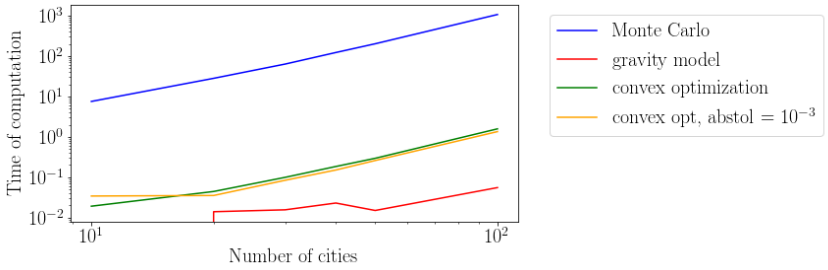
(b)



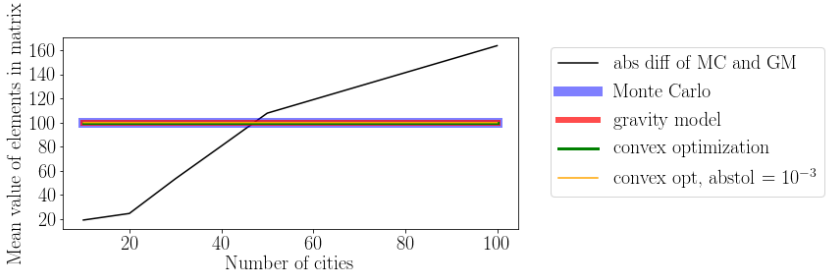
(c)

Figure 29: Aproximation functions for time computation for 3 methods, gravity model in Figure 29a, convex optimization in Figure 29b and convex optimization with absolute tolerance 10^{-3} in Figure 29c

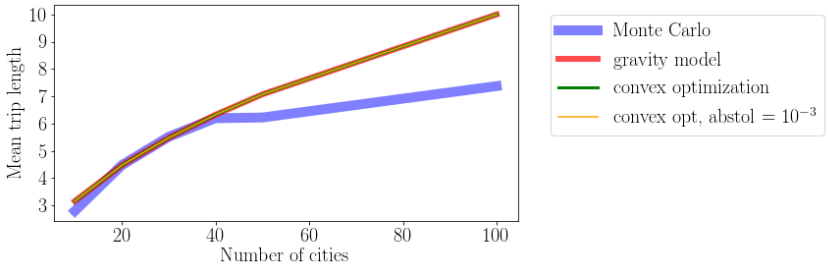
The second scenario is linearly growing population where $O_i \neq D_i$ and constant spacing of zones. The desired mean trip length is again chosen as the square root of the number of zones.



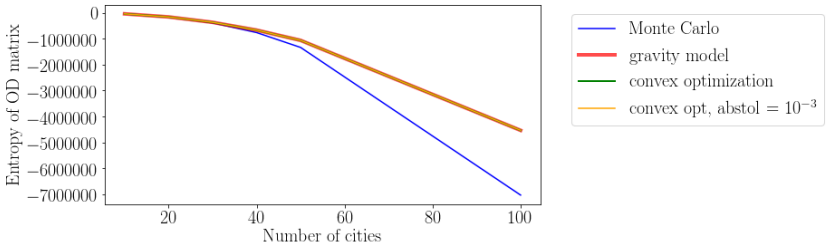
(a)



(b)



(c)

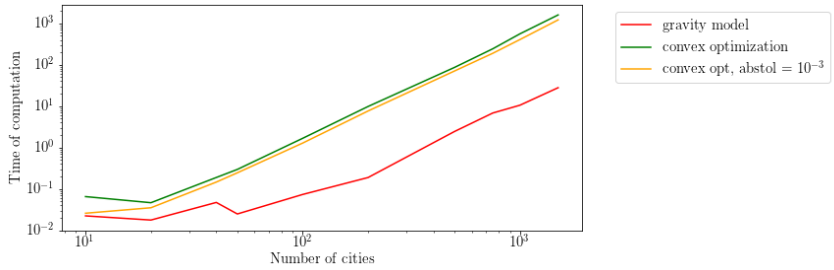


(d)

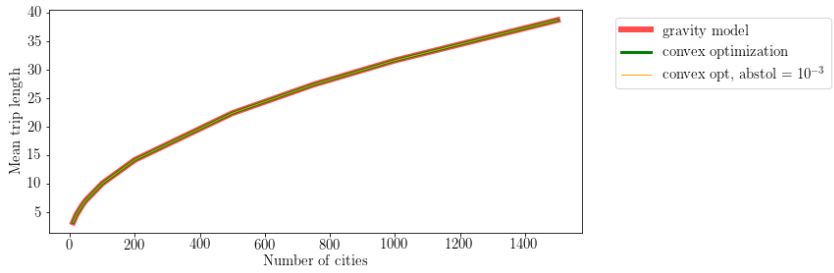
Figure 30: Comparison of 3 methods on the same scanario of linear growth of population and constant spacing of zones. Figure 30a is the comparison of time of computation for different sizes of system, Figure 30b is comparison of mean values of matrices. In Figure 30c we can see the comparison of mean trip length and in Figure 30d is comparison of entropies.

As in the previous example, it is clear from 30a that the most time-efficient method is the gravity model followed by convex optimization. Because this scenario is more com-

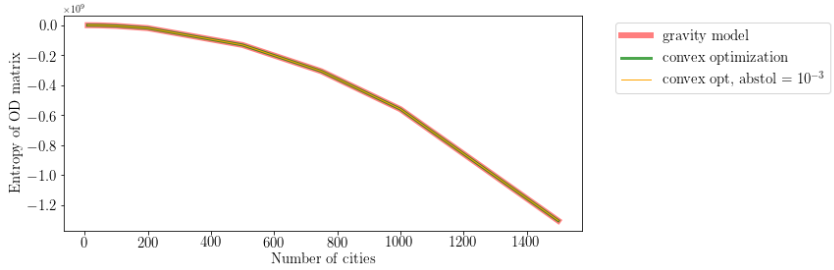
plicated than the one with a constant population, the Monte Carlo simulation starts to struggle. It is interesting that for systems with more than 50 zones, Monte Carlo simulation can not approximate desired mean trip length. This causes that the absolute average difference of matrix generated by the Monte Carlo simulation and matrix generated by gravity model is larger than the average value of the matrices. The problem could be in the number of iterations done by simulation, but because the algorithm is already enough time consuming we did not try to improve its performance by adding more iterations. Again the functions of time are not linear in log-log scale.



(a)



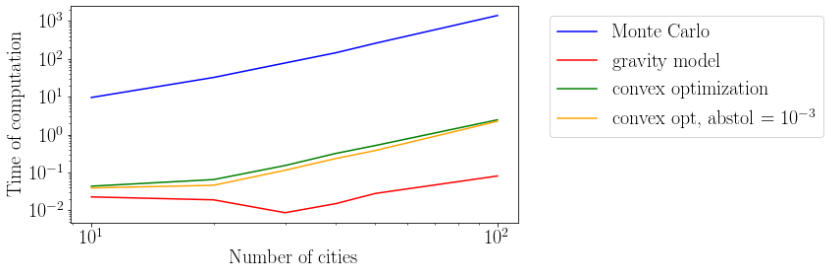
(b)



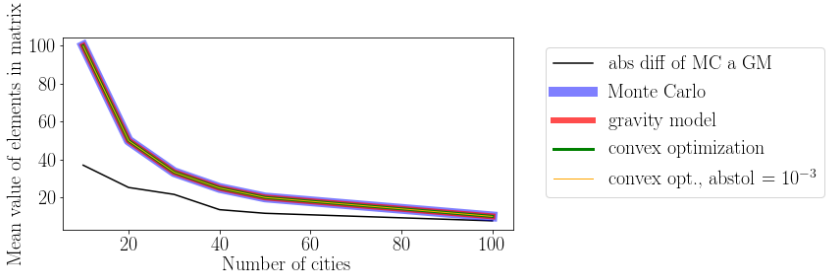
(c)

Figure 31: Comparison of 2 methods on the same scanario of linear growth of population and constant spacing of zones with large number of zones in the system. Figure 31a is the comparison of time of computation for different size of system. In Figure 31b we can see the comparison of mean trip length and in Figure 31c is comparison of entropies.

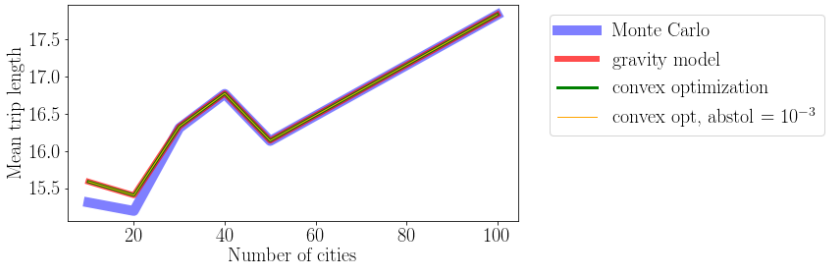
The last scenario is a constant population and normal distribution of zones. The desired mean trip length is the square root of the maximum distance of zones.



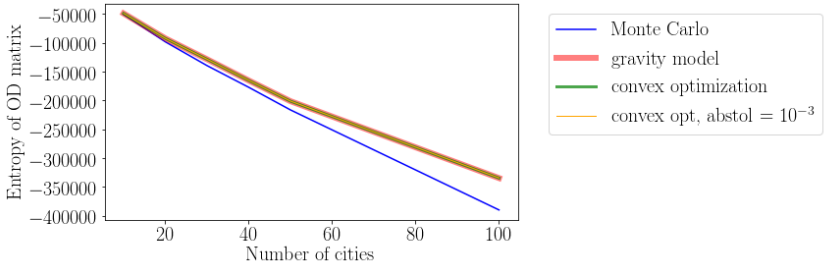
(a)



(b)



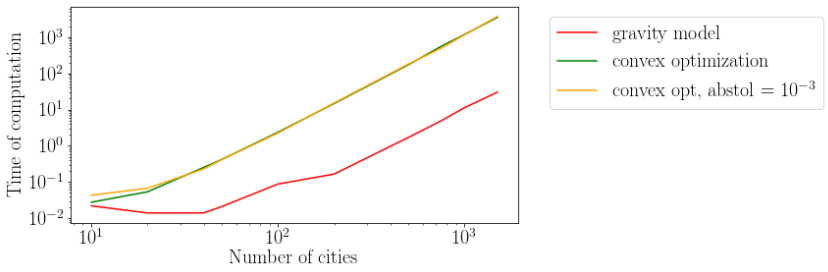
(c)



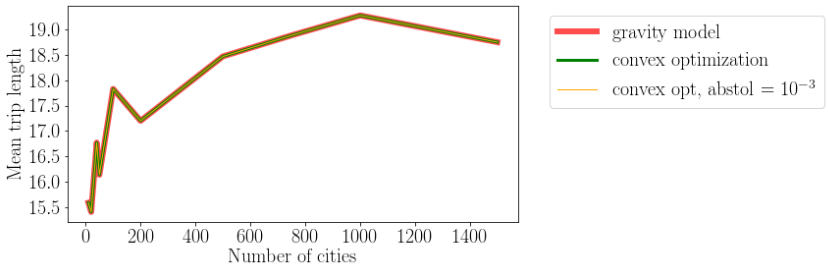
(d)

Figure 32: Comparison of 3 methods on the same scanario with constant population and normal distribution of zones. Figure 32a is the comparison of time of computation for different sizes of system, Figure 32b is comparison of mean values of matrices. In Figure 32c we can see the comparison of mean trip length and in Figure 32d is comparison of entropies.

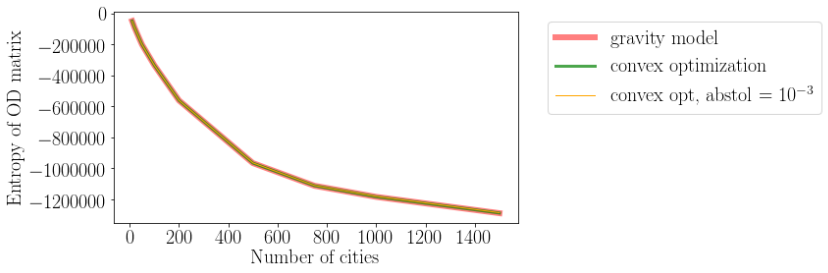
Even though Monte Carlo indicates some problems during the process, like the too-small mean trip length for the system with ten zones or smaller entropy of the system with 100 zones, this approximation could be considered sufficient. Although it is still a time-consuming calculation, the results are sufficient this time.



(a)



(b)



(c)

Figure 33: Comparison of 2 methods on the same scanario with constant population and normal distribution of zones with large number of zones in the system. Figure 33a is the comparison of time of computation for different sizes of system. In Figure 33b we can see the comparison of mean trip length and in Figure 33c is comparison of entropies.

For this scenario, the maximum iterations for the primal-dual interior-point method had to be set for 300 instead of default 100. The computation time of convex optimization is higher than in the previous scenarios.

4.2 Summary of comparison

We conclude that the Monte Carlo simulation was not a reliable instrument for generating origin-destination matrices in scenarios where the mean element of the matrix was small. This alternative does not produce zero elements or small numbers in the matrix; therefore, the difference from the matrix generated by the gravity model was significant, and the OD matrix had smaller entropy. In the following research, we would like to focus on the efficiency of iteration steps and algorithm improvement in extreme scenarios of mean trip lengths.

Convex optimization using primal-dual interior-point method showed promising approximation of gravity model. Its disadvantage could be the absence of self-concordance, so we do not guarantee that the convergence to the optimal solution is polynomial. It is time-efficient only for systems with 500 or fewer zones because of the feasibility tolerance needed for the primal-dual interior method.

4.3 Adding new information and constraints

The previous subsection results suggest that the compromise between the time efficiency and the easy addition of the new constraints is the convex optimization. We decided to test this method on the scenario with ten zones with a constant population of 1000 and constant spacing between them. The desired mean trip length for all following examples was 2.5. New constraints represent the external information obtained from traffic census or other transport models about the particular number of trips between zones.

The first approach was adding equations as a constraints to the problem. We generated the random position of the matrix and random value from uniform distribution $R(0, 500)$ and set it as a necessary condition. We repeated this procedure for 0, 1, 2, 4, 8 and 10 equality constraints. The resulting matrices can be seen in the figures below.

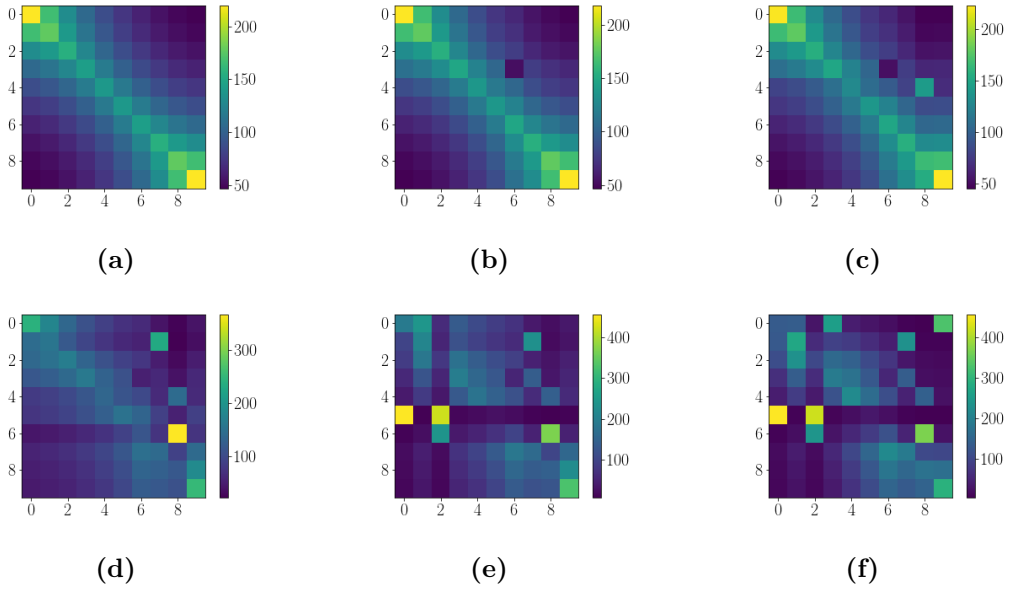


Figure 34: The evolution of OD matrix generated by convex optimization, with 0, 1, 2, 4, 8 and 10 added constrains stabilizing the elements of the matrix. The mean trip length is 2.5

The second approach was subtracting the square of the difference of the actual and desired values of the OD matrix elements from the objective function. Usually, we are not 100% about the exact value of the imputed element, but we assume that the value should be in the surroundings of that value. Therefore it could be better to choose the OD matrix with higher entropy even though the condition is not precisely met. The resulting matrices can be seen in the figure below.

The last approach was subtracting the absolute value of the difference of the actual and desired values of the OD matrix elements from the objective function. The absolute value does not punish higher differences as much as the quadratic function; therefore, the method could choose the OD matrix with very high entropy even though that the constraint is not met. The resulting matrices can be seen in the figure below.

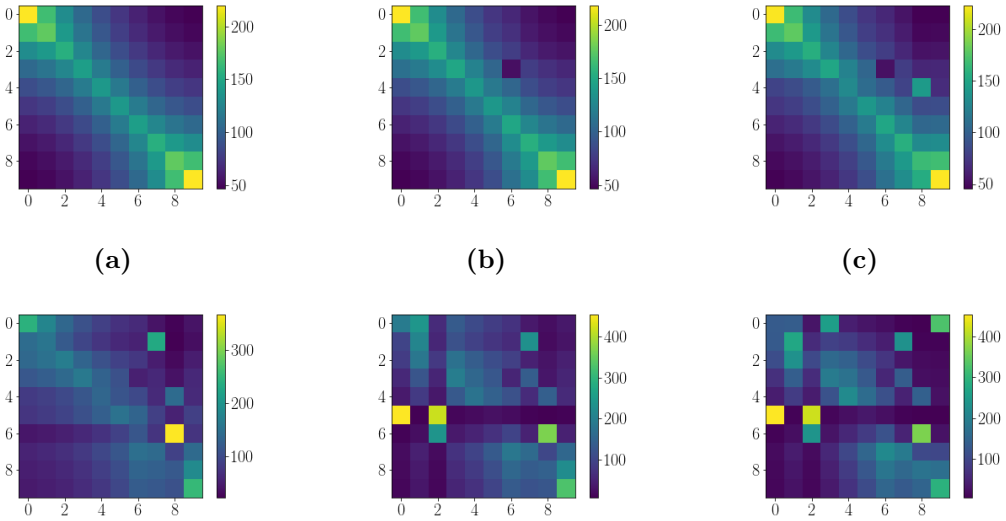


Figure 35: The evolution of OD matrix generated by convex optimization, with 0, 1, 2, 4, 8 and 10 added constrains implemented into object function as square value of difference stabilizing the elements of the matrix. The mean trip length is 2.5

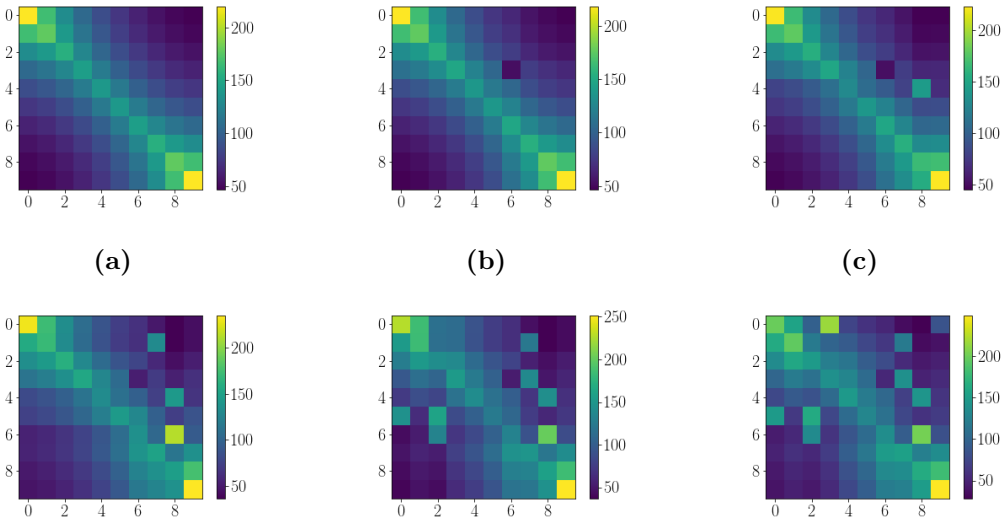
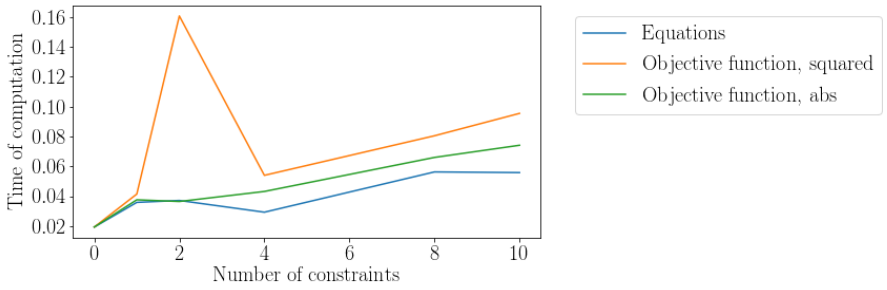
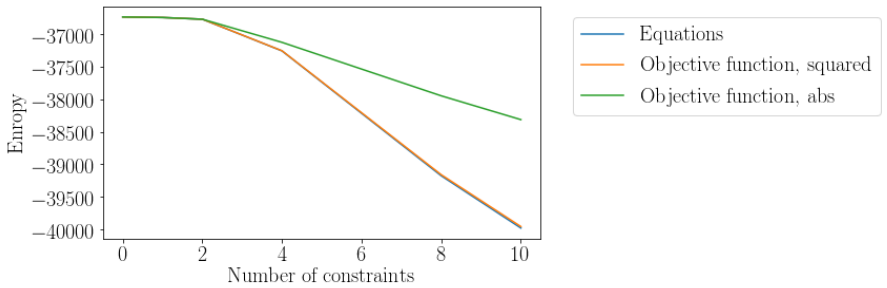


Figure 36: The evolution of OD matrix generated by convex optimization, with 0, 1, 2, 4, 8 and 10 added constrains implemented into object function as absolute value of difference stabilizing the elements of the matrix. The mean trip length is 2.5

We can see that the first and second approach generated almost the same OD matrices. On the other hand, the approach with the absolute value in objective function produced slightly different matrices from figure d, i.e. four constraints on OD matrix. To evaluate these approaches, we had to look at computation time and the actual entropy of matrices.



(a)



(b)

Figure 37: The evolution of compution time and entropy of OD matrix generated by convex optimization, with 0, 1, 2, 4, 8 and 10 added constrains implemented as equations, squared differences or abolute differences in object function.

From Figure 37, we see that the best option for implementing new constraints is the maximization of entropy and subtracting the absolute value of differences of constraints. It is more computationally expensive than the approach with equity constraints, but it stores the information we obtained with the highest entropy of all approaches. The squared difference in the object function follows the equations too strictly. Therefore the OD matrix had lower entropy.

Conclusion

The work aims to modify and apply the demon algorithm to generate OD matrices and compare its advantages, limitations and complexity with the current iterative gravity model. The purpose of such an algorithm is to systematically add new constraints representing experimental knowledge about the mobility of people. The modified demon algorithm was successfully implemented, but the analysis showed that it was not of sufficient quality in accuracy, speed of convergence and time complexity. However, in the creation process, an alternative approach to maximizing entropy by convex optimization emerged, subsequently subjected to the same analysis. The outputs of the second algorithm have the required accuracy and significantly better time scaling.

In the first chapter, we introduced the theory of transport modelling for the reader, who is not familiar with this discipline. In details, we explained the principle of the iterative gravity model, the most used model for trip distribution. We derived the model from the entropy-maximizing approach.

The second chapter presented the demon algorithm, Monte Carlo simulation that maximizes the system's entropy, and it was examined on particular examples. Our contribution was presented when we extended the Monte Carlo simulation for OD matrices and created the algorithm for initial input. The resulting matrix from the algorithm was obtained by averaging the matrices through states when the demon had a value lower than the desired threshold. By this procedure, we approximated the matrix with maximum entropy. We exercised this algorithm for the number of simple scenarios of the different distribution of population and coordinates of zones. The algorithm performed reliably only for a part of these simulations; therefore, we decided to move to the next alternative.

The fact that the maximization from the first chapter is convex and can be transformed into exponential cone programming are stated in the third chapter. This type of cone problem is solved, for example, by the ECOS solver, using the primal-dual interior-point method.

The comparison of the gravity model, Monte Carlo simulation and convex optimization in terms of accuracy, speed of convergence and time complexity for different scenarios of population and distribution of zones is presented in the last chapter. The extended Demon algorithm was not as effective as expected. The time complexity was too large, and the

accuracy was insufficient for the majority of simulations. The convex optimization showed promising results even though it is a more time-consuming method than the gravity model. Therefore, we focused on adding new constraints representing knowledge about the mobility of people to the problem, using convex optimization and ECOS solver. The best result was obtained to implement the constraints about the elements of the matrix as the absolute value of the difference from desired value into the objective function. The time complexity was only a little higher than for the direct input of equations as constraints, and the achieved entropy was the highest from all tested options.

The main contribution of this work are the results presented in the last chapter. We found a suitable alternative for the gravity model, which can take additional constraints in the form of information about the mobility of people, in contrast with the gravity model that is not capable of such implementation. When using the absolute value in the objective function for convex optimization, the method even distinguishes whether the information provided is in agreement or contrast with the most likely state of the OD matrix. Even though the Monte Carlo simulation results do not indicate that this could be an appropriate substitution for the iterative approach, there are possible extensions of this research that could improve its performance.

In the future extension of this work, one could focus on the efficiency of the iteration of the Monte Carlo simulation. It is possible that the sampling of the whole blocks of the matrix could speed up the algorithm and improve convergence. Also, the possibility of the more efficient program used for simulation could be taken into account. Lastly, the convex optimization with added constraints should be tested on the real data and network and then evaluated.

References

[1] Probability density function of the cartesian x-coordinate of the random point inside the hypersphere, 2013. scholarly article in the fields of mathematics.

[2] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, USA, 2004.

[3] Harry J. Casey Jr. The law of retail gravitation applied to traffic engineering. *Traffic Quarterly*, 9(3):313–321, 1955.

[4] Vladimir Jurukovski Connie Rye, Robert Wise. *Biology*. OpenStax, 2016.

[5] Rubén Cordera, Roberto Sañudo, Luigi dell'Olio, and Ángel Ibeas. Trip distribution model for regional railway services considering spatial effects between stations. *Transport Policy*, 67:77–84, 2018. Efficient, Safe and Intelligent Transport. Selected papers from the XII Conference on Transport Engineering.

[6] Michael D. Meyer and Eric Miller. *Urban Transportation Planning: A Decision-Oriented Approach*. 01 2001.

[7] Joachim Dahl and Erling Andersen. A primal-dual interior-point algorithm for non-symmetric exponential-cone optimization. *Mathematical Programming*, 03 2021.

[8] Steven Diamond and Stephen Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016.

[9] A. Domahidi, E. Chu, and S. Boyd. Ecos: An socp solver for embedded systems. In *2013 European Control Conference (ECC)*, pages 3071–3076, 2013.

[10] Antonella Ferrara, Simona Sacone, and Silvia Siri. *Microscopic and Mesoscopic Traffic Models*, pages 113–143. Springer International Publishing, 2018.

[11] Daan Frenkel and Patrick B. Warren. Gibbs, Boltzmann, and negative temperatures. *American Journal of Physics*, 83(2):163–170, February 2015.

[12] Gordon W.F. Grake. Entropy. *Encyclopedia Britannica*, 2018.

[13] Tony Guénault. *Statistical Physics*. Springer, Dordrecht, The Netherlands, 7 edition, 2007.

[14] Dirk Helbing and Péter Molnár. Social force model for pedestrian dynamics. *Phys. Rev. E*, 51:4282–4286, May 1995.

[15] G. M. Hyman. The calibration of trip distribution models. *Environment and Planning A: Economy and Space*, 1(1):105–112, 1969.

[16] Xiaodong Liu, Yuan Zhou, and Andreas Rau. Smart card data-centric replication of the multi-modal public transport system in singapore. *Journal of Transport Geography*, 76, 03 2018.

[17] J. de D. Ortuzar and LG Willumsen. *Modelling transport*. Wiley, 4 edition, 2011.

[18] Martyn L. Senior. From gravity modelling to entropy maximizing: a pedagogic guide. *Progress in Human Geography*, 3(2):175–210, 1979.

[19] Santiago Akle Serrano. *Algorithms for unsymmetric cone optimization and an implementation for problems with the exponential cone*. PhD thesis, Stanford University, 2015.

[20] Štatistický úrad. Krajské mestá z pohľadu kvality života. *Ústredie ŠÚ SR*, April 2020.

[21] Mária Trnovská. Lecture notes in convex optimization, summer semester, March 2021.

[22] Frans Willekens. Monitoring international migration flows in europe. towards a statistical data base combining data from different sources. *European Journal of Population*, 10(1):1–42, 1994.

[23] A. G. Wilson. The use of the concept of entropy in system modelling. *Operational Research Quarterly (1970-1977)*, 21(2):247–265, 1970.

[24] A. G. Wilson. *Entropy in urban and regional modelling*. Routledge, 2011.

[25] Alan Wilson. Entropy in urban and regional modelling: Retrospect and prospect. *Geographical Analysis*, 42:364 – 394, 10 2010.

[26] I. Wood and T. Downs. Demon algorithms and their application to optimization problems. In *1998 IEEE International Joint Conference on Neural Networks Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98CH36227)*, volume 2, pages 1661–1666 vol.2, 1998.

Appendix A

Python codes for the simulations

Python codes for all simulations presented in this work are available on `https://github.com/ivana-13/Master_thesis`. The immutability of codes in the repository is secured by the MD5 Hash code presented in the table below.

Name of the file	Section	MD5 Hash
Gravity_model_simulation	1.7	6015ce24ca5cb0022fecc08e604df5db
demon_algorithm_simulation	2.2	f2aa8020b4436692d65f9d819448e42b
Generation_OD_with_demon_algorithm	2.4 & 4.1	d263f04089e3e99ea7183b41aa74fdcc
convex_optimization	4.1 & 4.3	343789e3a1c46bcc15a714bfd0a0b6af

Table 7: Python codes with their Hash MD5 to secure immutability of scripts for all simulations presented in the work.