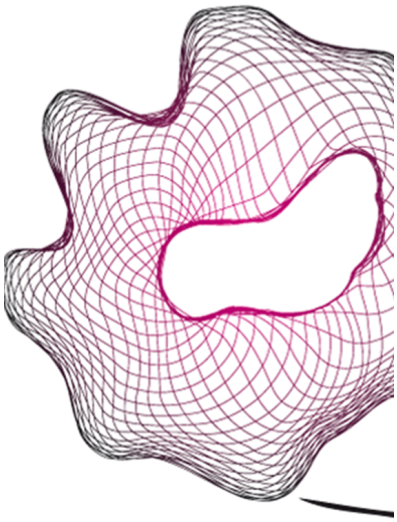
UNIVERSITY OF TWENTE.

**Faculty of Electrical Engineering,
Mathematics & Computer Science**



Origin-destination matrix estimation with SPSA and STAQ

Tanja Gellenbeck

M.Sc. Thesis
August 12, 2016

Supervisors:
dr. G.J. Still
Luuk Brederode
Bastiaan Possel

Applied Mathematics
Faculty of Electrical Engineering,
Mathematics and Computer Science
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

dat. mobility
Don't guess. Know.

Abstract

Origin-destination matrices describe the traffic demand between different zones of a network. Their entries, the number of trips for every origin-destination pair, can be assigned to a network to give corresponding link flows and speeds. The OD matrix calibration problem is the problem of finding an OD-matrix which after the assignment step corresponds as good as possible with some given criteria. During this thesis, the only criteria considered are measured values for flow and speed on specific links of the network. This problem can be formulated as a bi-level optimization problem. In the upper level, a minimization of the difference between the estimated flow and the measured flow is done while in the lower level the relation between the OD matrices and the link flows is derived by means of a traffic assignment model.

Although there are good methods to solve the problem when using static traffic assignment models, when it comes to (semi-) dynamic assignment models (such as STAQ), the problem becomes more complex and computationally intense. (Semi-) dynamic assignment models accurately model the primary effects of congestion (flow metering and spillback), which, in the context of matrix calibration, can cause a misinterpretation of the traffic flow measurements. In the situation of (semi-) dynamic models, the sensitivity of the assignment model plays a crucial role in the minimization part in the upper level. This dependence makes the matrix approximation more difficult. To hold the amount of required calculations low, the stochastic approximation method SPSA is used in this thesis. This method requires less measures of the objective function per iteration than other methods (like e.g. FDSA), hence less computations of the assignment model.

Since the calculation times are still large for most networks and there are a lot of parameters that need to be tuned, the further exploration of SPSA might be sensible. There also exist modifications for SPSA that might help to reduce the computational time until convergence.

The aim of this master thesis is to implement and test the basic version of SPSA and its modifications to minimize the differences in the upper level. This is done in combination with the semi-dynamic assignment model STAQ for the traffic assignment part in the lower level. Also some tests of the different parameters of the SPSA algorithm are performed to improve the convergence of the method further.

Contents

1 Introduction

1.1 Motivation of research

Since traffic plays an important role in modern life, a lot of research is done in traffic engineering to optimize traffic flow and other criteria such as pollution or accessibility. Traffic systems become more and more complex due to the large demand for transport in cities. Since people are choosing their trips based on the current situation and the current situation is influenced by the choices of all people, the situation becomes even more complex. To get more insight into the situation, simulations are conducted on study areas. These areas can include cities or even whole regions with several connected cities. The study area is divided into smaller parts, called zones, which include e.g. a neighbourhood, an area of several companies or a city centre.

The area of interest is modelled by a network consisting of a set of nodes (N) and a set of links (L). The directed links $l = (i, j) \in L$ represent the streets from node i to node j, hence $i, j \in N$. The nodes in N connect several streets with each other. They can also represent origins or destinations, hence places where travelers come from or are attracted to. So the links connect the nodes with each other and thus make up a connected network denoted by $G = (N, L)$. The links of the network have different characteristics such as capacity, speed at capacity and free flow speed. Capacity reflects the amount of vehicles that are able to pass the link in a specific time interval. The speed at capacity gives the average speed when the capacity of the link is reached and the free flow speed defines the possible speed on a link if no congestion occurs. Using this network and its characteristics, transportation engineers can simulate different situations and see the effect of possible measures on the traffic situation. They aim at optimizing the traffic flow, increasing accessibility and decreasing congestion and pollution.

To optimize the traffic, a lot of data is required. This data can be obtained by sensors, traffic counts, license plate investigation or GPS data and gives details about the traffic. This thesis is focused on the information obtained by traffic counts which include values of flow and speed for the observed links in the network. This data gives valuable insight into the situation. Using this data, traffic engineers are able to model and analyse the traffic and to identify congested situations and find solutions.

However, the information obtained by the sensors is only descriptive and not explanatory for the human behaviour behind it. To get explanatory data, other means to get the desired information are required. This can be done by doing surveys among the users and by following the steps of the so-called four step model to model the traffic situation (see section 3.1 for more details). From the first three steps we get a matrix containing the number of all trips from points of origin to points of destination. This matrix is called the origin-destination matrix. However, the initial data is not complete, since it will be too difficult to conduct a survey amoung all people, but it can give an explanatory approximation of the situation.

On the other hand, the data from the sensors only gives information about the flow and speed on one link, but it does not tell what origins and destinations the trips have. In order to make simulations and forecasts reliable, a good origin-destination matrix is required, which can then be assigned to the network in the fourth step of the four step model. This gives an overview of the situation and helps to find and locate the problems that cause the congestion. A good origin-destination matrix is a matrix that reproduces the reality as close as possible. For this reason an OD matrix calibration is performed. Here both descriptive and explanatory data is used to find a model that reflects the reality as close as possible.

The OD matrix calibration problem is hence the problem of calibrating a given OD matrix to match specific measured values. In this study, these values are measurements of the flow and speed at the links. The aim of the calibration is to carefully change the number of trips of some initial OD matrix such that after the assignment is performed with the assignment model, the corresponding values for flow and speed are as close as possible to the measured values at the observed links. The exact procedure is explained in section 3.2. Since there is a complex dependency between the origin-destination matrix and the values for flow and speed on the links, the calibration is not straight forward, but needs to be performed iteratively.

There are some quite reliable methods to solve the OD matrix calibration problem when using classic static traffic assignment models. However, when it comes to static models that are capacity and storage constrained, called quasi-dynamic assignment models, the problem becomes more complex and computationally intense. Due to the complexity of the situation caused by the quasi-dynamic assignment model, most common approximation algorithms do not converge or are likely to converge to local minima. The use of stochastic approximation algorithms might help to solve this problem.

1.2 Problem description

In transportation engineering, origin-destination matrices (OD matrices) describe the traffic demand between different zones of a studied area. Two connected zones of this network form an origin-destination pair (OD pair). An OD matrix D consists of entries d_{ij} with $i, j \in N$. More precise, there is a subset of nodes that is either origin, destination or both. These special nodes are called centroids. So the set of centroids, denoted by C, is a subset of the nodes, $C \subseteq N$. This definition includes that $i, j \in C$. The entries of the OD matrices, d_{ij}, are the number of trips made from origin i to destination j, hence the demand of this pair. With the assignment model the demand from the OD matrix is distributed along the possible routes in the network according to their characteristics such as travel time or travel costs.

As output of the assignment, corresponding values for flow and speed are found for all links. This is the fourth step of the so-called four step model which is explained in more detail in section 3.1. The output of the assignment step differs by the used assignment model. There are static and dynamic models, which can simulate specific traffic phenomena more or less exact.

$$D \xrightarrow[assignment]{} (v(D), s(D)) \tag{1}$$

The problem of OD matrix calibration can be formulated as a bi-level optimization problem. In the upper level, a minimization of a difference function between the estimated values and the measured values is done while in the lower level the relation between the OD matrices and the link flow and link speed is derived by means of a traffic assignment model. This concept is further explained in section 3.2.

The used assignment model in this thesis is STAQ (Static traffic assignment with queueing), a quasi dynamic assignment model. Quasi-dynamic assignment models are able to consider the capacity of the links and can take the queueing in congested systems into account. Classic static assignment models might calculate a link flow that is higher than the capacity of this link. In these situations just the speed decreases, but the vehicles can still pass the link. In reality, queueing occurs in front of the too small link, which is called bottleneck. This behaviour can be modelled

with (quasi-)dynamic models, since they obey the capacity and add the queueing effects to the network.

The main reason that the application of quasi-dynamic models makes the OD matrix calibration problem more complex is that the underlying relation between the flow and the speed on a segment is not anymore bijective. The underlying fundamental diagram gives the relation between flow and speed on a link. This diagram is specific for every link in the network and it is designed based on the value for capacity, free speed and speed at capacity. The relation between flow and speed is bijective for static problems, which means that it gives for every value of flow just one corresponding value for speed. This is not anymore given when dealing with quasi-dynamic models. Here two different speed values for the same flow are possible. One is in uncongested and one in the congested state. These states are also called traffic regimes. The corresponding relation between the flow and the speed is given in Figure 1 for both static and quasi-dynamic assignment models.

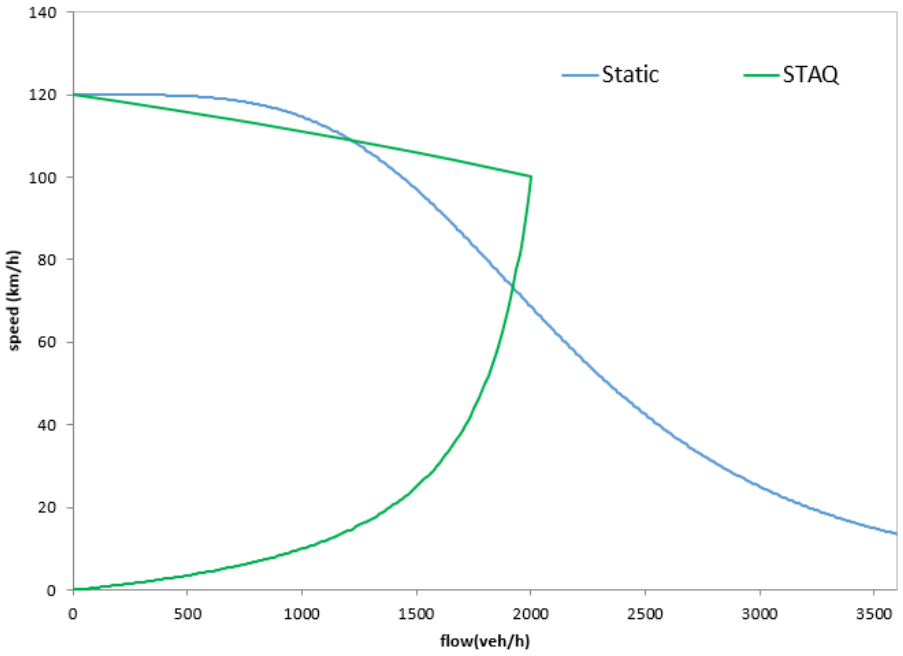


Figure 1: Relation between the flow and the speed for both static assignment (speed flow curve) and STAQ (fundamental diagram) - the capacity of this link is 2000

The minimization in the upper level needs to be done with an optimization algorithm to find the corresponding optimal solution. Due to the fact that the fundamental diagram is not bijective for STAQ, both flow and speed need to be considered in the minimization process. Unfortunately, by doing this, the situation gets more complex and the optimization algorithm encounters some problems.

Since the assignment model used in the lower level does not have one explicit gradient, a gradient approximation algorithm is required to solve the minimization problem. For this thesis it

was chosen to use the SPSA (Simultaneous Perturbation Stochastic Approximation) algorithm to find this gradient approximation. For this method less measurements of the objective function per iteration are necessary compared to other stochastic optimization methods (e.g. Finite Difference Stochastic Approximation (FDSA)). In the case of OD matrix calibration problems, this implies that a lower number of assignments per iteration is required, which might decrease the computation time to solve the problem. The algorithm of SPSA is explained in more detail in section 3.5.

The calibration of an origin-destination matrix can be done for very complex networks including different modes of transport. However, to keep the problem in a more simple scope and to be able to investigate it in more detail, this thesis is restricted to only observing one mode of traffic, namely the individual vehicle mode, where the car and freight traffic is aggregated to pcus (passenger car equivalents). In further research it is possible to expand the method to multiple modes of traffic.

The aim of this master thesis is to implement and test the basic version of SPSA in the programming language Ruby such that it works with the transport planning software OmniTRANS. This is done in combination with the quasi-dynamic assignment model STAQ (Static Traffic Assignment with Queueing), which is used for the traffic assignment part in the lower level. Also the investigation of the different parameters of the SPSA algorithm is important to investigate and improve the convergence properties of the method when combined with STAQ. Furthermore, some techniques are tested that reduce the problems with the algorithm. We are aiming at finding a method based on the SPSA algorithm that solves the OD matrix calibration problem for STAQ as fast and reliable as possible.

1.3 Thesis Outline

During this thesis the focus will be on different aspects of the OD matrix calibration problem with the optimization algorithm SPSA. First some review of the already available literature is done to find a promising approach to the problems in calibrating OD matrices with quasi-dynamic assignment models. This is written in section 2.

In the next section, section 3, more detailed information on the different parts of the OD matrix calibration problem is given, including the four step model, the general OD matrix calibration problem, the assignment model STAQ, the approximation algorithm SPSA and the different convergence criteria.

In section 4 the problems caused by the presence of local minima and other problems for the convergence of the algorithm are investigated. This section shows why local minima occur in the objective function, how the problems they cause can be avoided and gives a new, more advanced method to use for the optimization process. This section also deals with the issue of bounds to restrict the search area and the evaluation of the performance of the algorithm, including the number of assignment steps required and a new convergence criterion to measure the quality of the result.

The section 5 deals with a small, but realistic network. For this a network describing the highway corridor between Tilburg and Eindhoven in the Netherlands is used and different situations are investigated. Here the performance of the algorithm is measured and improved by application of the so-called asymmetric design. Also some of the parameters of SPSA are investigated and tests are performed in order to find the optimal set of parameters for the optimization. At the end of this section, these optimal parameters are tested for several different situations.

Finally a conclusion of this thesis is drawn and some possible modifications of the SPSA algorithm and recommendations for further research are given in section 6.

2 OD matrix calibration and SPSA: literature review

In this section some literature about SPSA as well as about its application in OD matrix calibration problems is reviewed. First some basic outcomes of research about SPSA are given and then its application and problems are discussed. Also the basic idea of STAQ is explained shortly. Then the different results for dynamic OD matrix calibration are summarized.

Spall (1998) [2] gives in his work some general information about the optimization algorithm SPSA. This technique computes a gradient approximation using only two measurements of the objective function per iteration, regardless of the dimension of the problem.. For OD matrix calibration this implies that only two performances of the traffic assignment are required to find a gradien approximation. This approximation of the gradient is especially useful when solving problems where the relationship between the variables to be optimized and the objective function are unknown, which is the case for OD matrix calibration due to the traffic assignment. In another paper of Spall (1992)[5] he proves that the algorithm converges and gives some advice and rules of thumb about how to choose the numerous parameters of the technique. These parameters need to fulfill some conditions such that the algorithm converges, but within this framework the user is free to choose the parameters. However, he proposes that especially the choice of the gain sequences a_k and c_k, as given in section 3.5, is critical for the performance of the algorithm. He also shows how to implement the technique in an efficient way. The rules of thumb proposed by Spall provide helpful guidance to find some set of parameters that works for our situation. More details about the SPSA algorithm will be given in section 3.5.

To use SPSA for the OD matrix calibration problem, an assignment model needs to be chosen. Bliemer et al. (2015) [1] categorized the existing different assignment models by two characteristics: the spatial interaction assumptions and the temporal interaction assumptions. They concluded that quasi-dynamic models, such as STAQ, belong to the static models, but they pay attention to the capacity and are mostly storage constrained. STAQ is able to take both the effect of limited capacity as well as the effect of spillback into consideration and thus belongs to the capacity and storage constrained models. The general ideas of STAQ and how it works are given by the authors. The assignment model STAQ is still a static model, which means that it is not necessary to deal with several time periods.

Within the field of the OD matrix calibration problem, some research has been done regarding the influences of different assignment models as well as different queueing models. Frederix (2012) [6] investigated in chapter 3 of his dissertation the effect of congestion on the dynamic OD matrix calibration problem. He investigated three different queueing models, the point queue, the spatial queue and the kinematic wave model, and found that a too simplified queueing model such as the point queue model might cause large errors in the dynamic matrix calibration. In his analysis he used the stochastic approximation algorithm SPSA with some chosen values for the parameters. In this thesis, SPSA is used as well, but with a different queueing model, the quasi-dynamic assignment model STAQ as introduced in section 1.2 and further explained in section 3.4. Although Frederix investigated the problem of dynamic OD matrix calibration, some problems he found may also occur in quasi-dynamic OD matrix calibration. Here the results of Frederix' work might be helpful.

In chapter 4 of Frederix (2010) [6] the author analyses problems in dynamic OD matrix calibration due to congestion from the perspective of the relationship between the link flows and the OD flows. Especially the presence of local minima causes errors in the calibration process. These might result from incorrect interpretation of the information obtained by the detectors. He proposed that it might be helpful to include the response of the lower level, the traffic assignment,

in the upper level, hence in the minimization of the differences. The main problems here are that this response is non-linear when dealing with spillback and flow metering and that the objective function in the upper level can be non-convex due to the congestion dynamics. Additionally, some ideas to prevent the problems caused by the local minima are discussed, including identifying the correct traffic regime by minimization of the speed and preserving this correct regime in the further optimization. These techniques might be helpful to obtain a good solution in the optimization and are further explained in sections 4.3 and 4.4.

Cipriani (2011) [4] also investigated the OD matrix calibration problem and proposed to include next to the flow differences also the differences between the measured speed and the observed speed in the objective function. He states that if only the flow on a link is measured, it does not indicate whether there is congestion on the link or not. This knowledge is important to estimate the OD matrix in a good way, since a low flow can be caused by both congestion or a low demand. Including the speed helps to interpret the information given in a more exact way. Cipriani also used lower and upper bounds for the matrix entries, which ensures to avoid infeasbile solutions and to restrict the search space. This technique is further investigated in section 4.5. Further, he investigated the computation time of SPSA, where he proposed three modifications to improve the performance of the algorithm: asymmetric design, the use of a third degree polynomial interpolation and the use of an average approximated gradient. The first modification is explained in section 5.3, whereas the last modification is already included in the general formulation of SPSA in section 3.5.1. The use of a third degreee polynomial interpolation is not further investigated, since for simplicity, in this thesis it is chosen to use the line search technique of golden section instead.

In his later work, Cipriani (2013) [3] performs a sensitivity analysis of the parameters of SPSA AD-PI (SPSA with asymmetic design and polynomial interpolation) when applied to a dynamic OD matrix calibration problem. Further a second order SPSA AD-PI approach is explored and another sensitivity analysis is done. Different values for the parameters are tested and some appear to have a lot of influence on the performance. These results are helpful for choosing the parameters in an effective way and to perform our own analysis of some of the parameters in section 5.4.

The literature research gives numerous ways to investigate the problem at hand. Spall (1992, 1998) [5],[2] gives a lot of information on the SPSA algorithm, including the choice of the parameters and how to implement it. It is found that the quasi-dynamic assignment model STAQ is required, since Frederix (2010) [6] found that an assignment model that does not consider the effect of congestion can cause problems for the optimization. On the other hand, Frederix found that there might be problems with a dyamic assignment model, so these problems can also occur for the application of STAQ. Frederix further found that most problems are caused by local minima and can be avoided by finding the correct traffic regime for all measurements and then preventing the algorithm to leave it again. Finally, Cipriani [4],[3] proposes improvements of the SPSA algorithm and performed a sensitivity analysis, which can be used to make one for the problem investigated in this thesis.

3 Background information

3.1 Four step model

In this section the classic four step model for static assignments is given. At the beginning the traffic network of interest is divided into smaller parts, the traffic zones. First specific areas such as the city centre, living areas or areas with several companies are recognized and assigned as zones with their own characteristics such as population, employment, etc. Within these zones the zone centroid is placed, which is a single point with all the attributes and properties of the zone. This point lies somewhere in the zone and is connected to the network via a link that is called centroid connector. This link gives the average time and distance to get to the network from within the zone. The zone centroid is also called gravity point. For all zones, the centroids are connected to the network.

Now the four subsequent steps of the four step model are executed:

- The first step that is made is the trip generation. Here the characteristics of the zones are transformed to trip production and trip attraction values that state how many trips are starting or finishing in this zone. This is done based on the zones characteristics and the socio-economic data such as income, employment, etc and is defined by a chosen model. These values are located at the zone centroid as a point of origin or destination of trips.

- The next step is the trip distribution step. Here the trip productions and trip attractions at the centroids are connected to produce origin-destination pairs. This results in the OD matrix D, a matrix which gives the so-called OD demand d_{ij}, the number of trips for every origin-destination pair (i, j), for the studied network. Hence this matrix gives the number of trips from every centroid i in the studied network to every other centroid j in the network.

- The third step is the modal split. Here the trips are divided according to the used mode of traffic. These modes can be car traffic, public traffic or bicycle, but also walking can be seen as a mode of transport. Often this is already done in step two to get OD matrices that only contain information for one mode.

- The last step of the four step model is the assignment step. Here the demand of the OD matrix is distributed along the network, resulting in values for link flow and link speed. An important factor to assign the OD demand is the route choice for each trip. This choice is based on the travel time and is thus influenced by the level of congestion on every possible route and the assigned traffic influences the amount of congestion. This means that during the assignment step an equilibrium is searched, the so-called user equilibrium, where all the traffic is assigned such that no individual can find a better route to its destination. This is done according to some chosen assignment model.

The assignment step of the model plays a crucial role in the OD matrix calibration problem, since here the OD demand, i.e. the output of steps 2 and 3, is approximated based on the travel times calculated in step 4, the measured flow and speed on the links. As stated, different assignment models can be chosen, which make the problem static, dynamic or quasi-dynamic. In this thesis the quasi-dynamic assignment model STAQ will be used, which makes the OD matrix calibration problem quasi-dynamic.

3.2 OD matrix calibration problem

In this thesis the OD matrix calibration problem is defined as the problem of finding an OD matrix whose values for flow and speed at the links match as good as possible with measured values on these links. This definition includes that whenever the modelled values for flow and speed match with the measured values, the OD matrix calibration problem is solved, independent of the found OD matrix. This implies that there might be more than one solution of the OD matrix calibration problem, since there can be more than one matrix that produce certain values for the flow and the speed. This is due to the underspecified characteristic of the problem, since there are mostly more variables that need to be optimized, the number of non-zero OD pairs, than there are constraints, the measured values for flow and speed on the links.

As described in the last section, the OD matrix calibration problem uses the output of the fourth step, the assignment step, to find the value of the OD matrix, which is the output of steps two and three. By restricting our research to just one mode of traffic, we can ignore the influence of step three.

The problem of OD matrix calibration can be formulated as a bi-level optimization problem. The classic mathematical formulation of the OD matrix calibration problem can be found in Cipriani(2011) [4], but the notation is slightly changed to make it more clear:

In the upper level a minimization problem is solved:

$$\text{upper level: } \min_D z(D) \quad \text{with } z(D) = f_1(v(D), \tilde{v}) + f_2(s(D), \tilde{s}) \tag{2}$$

where D gives the OD matrix, $v(D)$ and $s(D)$ give the flow and speed on the links obtained by the assignment with matrix D and $\tilde{v}$ and $\tilde{s}$ give the measured flow and speed on the links.

In the lower level, the assignment of the matrix D to the network is done to get the values of $v(D)$ and $s(D)$:

$$\text{lower level: } D \xrightarrow[assignment]{} (v(D), s(D)) \tag{3}$$

where assignment denotes the assignment of the OD matrix D according to the chosen assignment model.

This bi-level optimization problem can also be seen as a problem from the field of game theory, called the Stackelberg game. Here the leader, the upper level (where the minimization of the objective function is performed), changes the values of the matrix D and the follower, the lower level (where the assignment is done), reacts on this change with new values for the modelled flow and speed.

To achieve a good estimation of the OD matrix, a search algorithm is applied to iteratively find the optimal solution. In this algorithm we proceed similar to the steepest descent method for solving the minimization problem

$$\min_D z(D) \tag{4}$$

The classical steepest descent method is as following: Starting with an initial guess D_0 we iterate

$$D_{k+1} = D_k - a_k \nabla z(D_k) \tag{5}$$

where the steps a_k are computed e.g. by exact (or inexact) line search. Note that the standard assumption in the steepest descent method is that z is differentiable. However, the function $z(D)$

above is probably not differentiable everywhere due to the assignment step with STAQ. Because of this, instead of the gradient $\nabla z(D_k) = g_k$ we use in our algorithm an approximation $\hat{g}_k$. More detail on the method of SPSA and how it relates to the steepest descent method can be found in section A in the appendix.

The procedure of this search algorithm is shown in Figure 2.

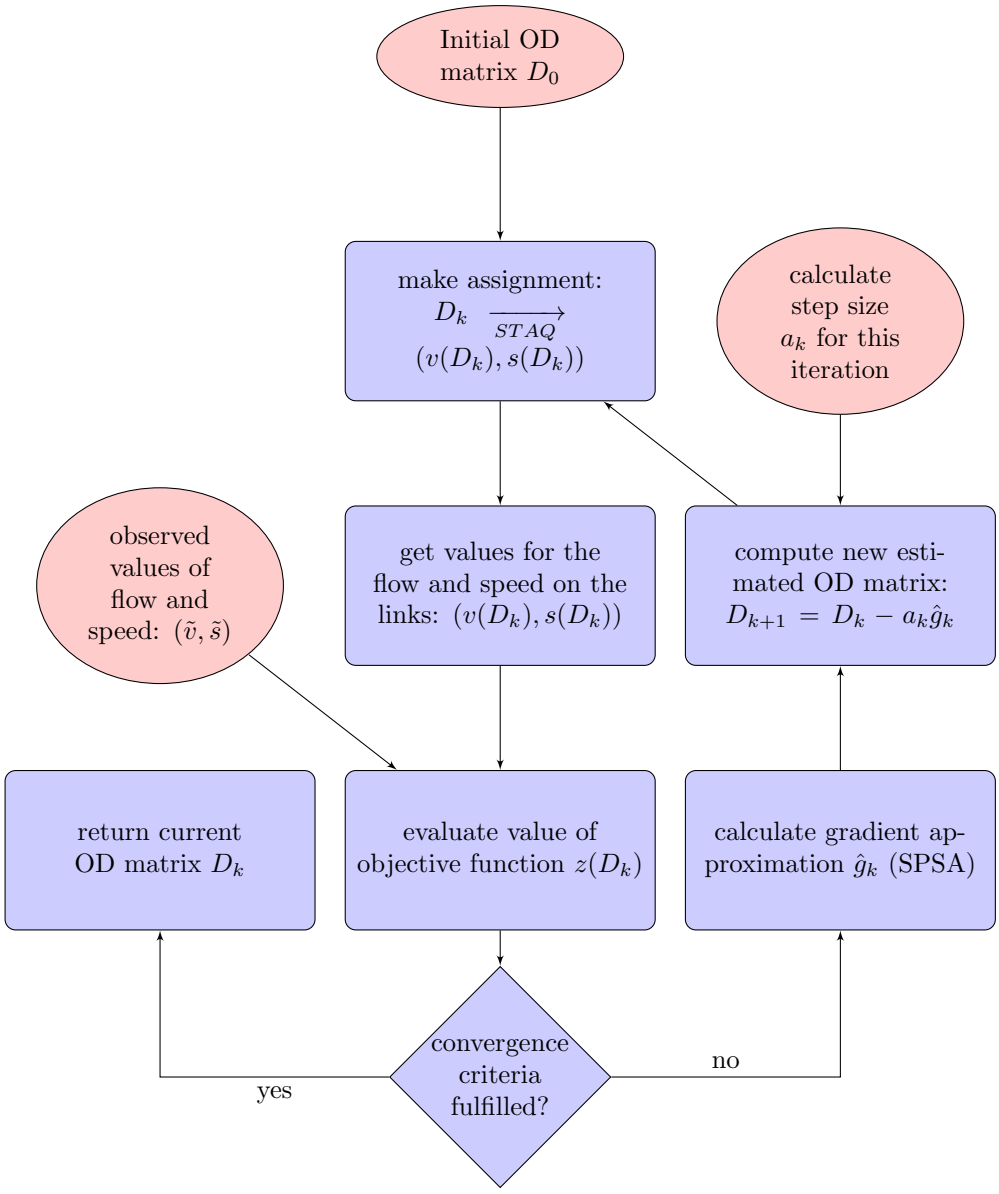


Figure 2: Scheme of the procedure of the OD matrix calibration

This scheme gives the general idea of the algorithm. Starting with an initial OD matrix D_0, which is mostly obtained by the first three steps of the four step model, the assignment is performed, according to the chosen assignment model (in this thesis this model is STAQ). This corresponds to the equation of the lower level (Equation 3) and gives values for the flow and speed $v(D_k)$ and $s(D_k)$. Next the values of the flow and the speed on the links are retrieved from the model and then compared to the observed values $\tilde{v}$ and $\tilde{s}$, which are given beforehand. This comparison is done by evaluating the equation of the upper level (Equation 2) to find the value for the objective function $z(D_k) = f_1(v(D_k), \tilde{v}) + f_2(s(D_k), \tilde{s})$. Next some chosen criteria for convergence are investigated and if the criteria are fulfilled, the process will return the current OD matrix D_k as its best solution. If the convergence criteria are not fulfilled, the process will calculate the gradient approximation $\hat{g}_k$. The way the gradient approximation is calculated is explained in more detail in section 3.5. From the gradient approximation together with some specific step size a_k a new estimated OD matrix is computed by $D_{k+1} = D_k - a_k \hat{g}_k$. Next this new matrix is again assigned to the network, giving the values for the flows and speeds on the links $(v(D_{k+1}), s(D_{k+1}))$, and the following steps are repeated until the convergence criteria are met.

This is the general procedure used here to solve the OD matrix calibration problem. In the next sections, the different parts of the process are further explained.

3.3 The objective function $z(D)$

The choice of the objective function plays an important role in the OD matrix calibration problem. This function determines what values are required to be minimized in the process. In general, these are the differences to the 'real' origin-destination matrix of the network. However, in practice this matrix is unknown. In this study we assume that the only information we have about it are the observed values for flow and speed on the links. Since we want the difference between the observed and modelled values for flow and speed to be as small as possible, it is logical to include them in the objective function $z(D)$. This can be done in different ways. For now, the objective function is chosen to include the euclidean differences of flow and speed with corresponding weights.

In most cases, the euclidian norm is calculated and all squared differences are added up. When doing this, it would be advisable to apply weights to give different values corresponding importance. The following equations give this general objective function for flow and speed:

$$z(D) = f_1(v(D), \tilde{v}) + f_2(s(D), \tilde{s}) \tag{6}$$

$$f_1(v(D), \tilde{v}) = \lambda \sum_{l \in L_m} (\tilde{v}_l - v_l(D))^2 \tag{7}$$

$$f_2(s(D), \tilde{s}) = (1 - \lambda) \sum_{l \in L_m} (\tilde{s}_l - s_l(D))^2 \tag{8}$$

where $0 \leq \lambda \leq 1$ and $l \in L_m$ are the links in the network where measurements are available. Here again $\tilde{v}$ and $\tilde{s}$ represent the measured values for the flow and speed on the links and D is the current matrix approximation. The values $v_l(D)$ and $s_l(D)$ are obtained from the assignment. So,

$$D \xrightarrow[assignment]{} [v_l(D), s_l(D)] \quad \forall l \in L_m \tag{9}$$

These equations result in the objective function $z(D)$ which states the corresponding values for different OD matrices D:

$$z(D) = \lambda \sum_{l \in L_m} (\tilde{v}_l - v_l(D))^2 + (1 - \lambda) \sum_{l \in L_m} (\tilde{s}_l - s_l(D))^2 \tag{10}$$

This results in the upper level equation:

$$\text{upper level: } \min_D z(D) = \min_D [\lambda \sum_{l \in L_m} (\tilde{v}_l - v_l(D))^2 + (1 - \lambda) \sum_{l \in L_m} (\tilde{s}_l - s_l(D))^2] \tag{11}$$

The value of the weight λ is for now chosen such that the influence of the flow and speed are approximately equal to each other. For example, if there is a value of flow of 1000 and the maximum speed is only 100 km/h, then a speed difference of 5 kilometers per hour is more important for the optimization than a flow difference of 5 vehicles per hour. So it is required that the squared values for the flow get a lower weight than the squared differences of the speed. The value of λ is in this thesis chosen below 0.1 and thus the weight of the speed gets a value above 0.9. For now, λ is chosen to be 0.05. This means that the weight for the speed is 0.95. In section 4.3 the role of these weights is further explored.

3.4 STAQ

The assignment model STAQ (Static Traffic Assignment with Queueing) is a quasi-dynamic model. This means that it only investigates one time period (so an average value over some time, e.g. one hour), which makes it static, but on the other hand the capacity constraints are observed and the effect of queueing is taken into account, which needs to be done dynamically. A quasi-dynamic model such as STAQ is required to accurately model congestion spillback and flow metering around congested links.

If the number of vehicles that want to use a link is larger than the capacity of that link, queueing occurs. In normal static models this capacity problem is ignored and the demand is assigned to the links, even if the flow exceeds the capacity of that link. To model these situations more accurately, STAQ consists of two phases, the squeezing phase and the queueing phase.

In the squeezing phase the links are investigated. If the demand at the link exceeds its capacity, the amount of demand that is larger than the capacity is stored as a so-called vertical queue at the beginning of the link. Here the difference between the capacity and the demand is summed up. This part of the model is a static assignment.

During the queueing phase, the length of the vertical queue, hence the total amount of traffic that could not pass the link, is translated into a horizontal queue that propagates back along the links. This is done according to traffic flow theory. This part of the model is dynamic to give good values for the queue length and the travel times.

The output of the assignment are values for flow, speed and density on all links in the network. The values for flow and speed can now be used in the objective function to evaluate the quality of the approximation of the OD matrix. The details about the assignment model of STAQ are outside the scope of this thesis, such that STAQ is used in the optimization algorithm as a black box where given a matrix D values for $v(D)$ and $s(D)$ can be calculated.

3.5 SPSA

3.5.1 General SPSA

SPSA (Simultaneous Perturbation Stochastic Approximation) is a stochastic gradient approximation algorithm that is used in a lot of different fields including aircraft modeling, noise cancellation and robot control.

SPSA is an optimization algorithm. If given an objective function $z(\theta) \in C^1$, SPSA aims at finding the minimum of this function. SPSA uses gradient approximation. This is a good method if the relationship between the objective function and the variable to be optimized, $\theta \in \mathbb{R}^p$, are unknown (which is the case in OD matrix calibration problems due to the assignment step). For using SPSA, the outcome of the objective function needs to be a scalar. The gradient approximation is then found by calculations including the values of the objective function.

SPSA is also a stochastic method, which means that the iterate θ_k converges to the optimal value θ^* in a stochastic sense (almost surely). The term stochastic also includes that there are values Δ_k for the perturbations in every iteration, which are taken at random according to a given distribution. This distribution needs to fulfill some conditions to make sure that the algorithm is able to converge to the optimal solution. These include an independent, symmetric distribution around zero and the fact that the absolute inverse expectation needs to be finite: $E|\Delta_{ki}^{-1}| < \infty$ for every component i of the vector Δ_k. This condition makes most common distributions inappropriate for this situation. All distributions with a positive probability around zero are not allowed for the SPSA algorithm. For this reason other distributions are required. A simple distribution recommended by Spall (1998) [2] and proven to be optimal by Hutchinson (2002) [7] is a distribution with as only possible outcomes 1 or -1, each with probability $1/2$.

Then the algorithm proceeds as follows: SPSA starts with an initial guess of θ, θ_0, and in every iteration k a gradient approximation $\hat{g}_k$ is calculated. This is used to make a step towards a better solution. This step has the following form:

$$\theta_{k+1} = \theta_k - a_k \hat{g}_k(\theta_k) \tag{12}$$

where $\hat{g}_k(\theta_k)$ is the approximation of the gradient $g(\theta) = \partial z/\partial\theta$ at the current iterate θ_k. a_k is some specific step size for iteration k.

For the application of SPSA, the value θ is perturbed by values of the vector Δ times a value c from the sequence c_k. Both sequences a_k and c_k are defined later. The vector Δ consists of p entries Δ_i which are randomly chosen such that they can be either -1 or 1 with probabilty $1/2$. With this Δ the perturbations are given in equation 13.

$$\theta^+ = \theta + c\Delta\,, \qquad \theta^- = \theta - c\Delta \tag{13}$$

In most situations more than one gradient approximation is performed, where the number of gradient approximations is denoted by m. In general $m \ll p$, the number of dimensions, hence there are a lot less gradient approximations than the number of elements in θ. So the vector θ is perturbed with m different vectors Δ^l, resulting in $2m$ different perturbed vectors denoted by θ^{+l} and θ^{-l} with $l \in \{1, 2, ..., m\}$. Then for the perturbed vectors the objective function values are found, denoted by $z(\theta^{+l})$ and $z(\theta^{-l})$.

From this the gradient approximation $\tilde{g}^l$ is found by

$$\tilde{g}^l(\theta^l) = \frac{z(\theta^{+l}) - z(\theta^{-l})}{2c\Delta^l \|\tilde{g}^l\|} = \frac{z(\theta + c\Delta^l) - z(\theta - c\Delta^l)}{2c\Delta^l \|\tilde{g}^l\|} \tag{14}$$

where z is the objective function at the corresponding points and $\|\tilde{g}^l\|$ denotes the norm of the gradient approximation, such that the gradient is normalized. This norm is defined by $\|\tilde{g}^l\| = \sqrt{\sum_i (\tilde{g}_i^l)^2}$. Note that the calculation of the gradient approximation $\tilde{g}^l$ only requires 2 values of the objective function for every gradient approximation, regardless of the dimension of θ. This means that SPSA requires less values of the objective function per iteration than most other optimization algorithms, since these use at least one value of the objective for every dimension of θ.

Finally, to find $\hat{g}_k$, all values of $\tilde{g}^l$ are averaged:

$$\hat{g}_k = \frac{1}{m} \sum_{l=1}^{m} \tilde{g}^l \tag{15}$$

An implementation of the algorithm with $m = 1$ is given in Figure 31 in the appendix. More detail about the SPSA algorithm is given in section A in the appendix.

The sequences a_k and c_k play an important role in the optimization process. c_k defines the "area" in which the objective function is evaluated to calculate the gradient. If it is chosen to be too large, the algorithm might not converge. If c_k is too small, the algorithm might stay in the current position regardless of whether it is optimal or not. The value a_k defines the step size to make. If it is chosen too large, the algorithm might take a step far away from the optimal solution. If it is too small, the algorithm gets stuck in the currect position or needs a lot more iterations to reach the solution.

In the general SPSA algorithm, the sequences are defined such that:

$$c_k = \frac{C}{k^\gamma} \qquad \text{and} \qquad a_k = \frac{a}{(A+k)^\alpha} \tag{16}$$

where k defines the current iteration number and C,γ, a, A and α need to be chosen according to the problem and some other conditions.

These conditions include:

- $\alpha - 2\gamma > 3\gamma - \frac{\alpha}{2} \geq 0$
- $\alpha > 0,\ \gamma > 0$
- $C > 0,\ a, A > 0$

From this it follows that

- $a_k, c_k > 0$ and $a_k \to 0,\ c_k \to 0$ as $k \to \infty$
- $\sum_{k=0}^{\infty} a_k = \infty$
- $\sum_{k=0}^{\infty} \left(\frac{a_k}{c_k}\right)^2 < \infty$

These conditions are shown in more detail in Spall (1992) [5]. Mostly the values for α and γ are chosen to be 1 and 1/6 respectively or 0.602 and 0.101, see Spall (1998) [2].

3.5.2 SPSA for OD matrix calibration

This section is about how to use SPSA in the OD matrix calibration problem. To make it easier to understand, the working scheme from section 3.2 is used in figure 32 in the appendix, where several elements are added to make the situation more precise.

As described in section 3.2 the OD matrix calibration problem starts with an initial guess of the OD matrix, D_0. So D_0 is a matrix of dimension $c \times c$ with $c = |C|$ the number of centroids in the network.

After performing the assignment with STAQ and evaluating the function value $z(D_k)$, the convergence criteria are checked. If they are not fulfilled, the SPSA algorithm is started.

Typically a big part of the entries of the OD matrices have value zero, which means that there is no demand between this origin and destination. For this reason, these values will remain zero during the whole process. So it is for sure that no demand between this origin-destination pair is present.

Due to this high number of zero entries, it is advisable to translate any OD matrix D to a vector of nonzero entries θ which can then be used in the SPSA algorithm. This translation is done by finding all positions of non-zero entries and storing their positions in an extra vector, while the entry values are copied one after another to the vector θ. This process can be reversed with the vector of positions and θ to find the matrix D. So first the matrix D_k is transformed to give the vector of non-zero elements θ_k.

For all nonzero entries, at iteration k the perturbation values Δ_{ki} are computed randomly according to the chosen underlying distribution. During this thesis, this distribution is chosen to be such that Δ_{ki} can be either 1 or -1 as reported in section 3.5.1, such that Δ_k is a vector with these values.

Next the vector θ_k is perturbed in two directions:

$$\theta_k^+ = \theta_k + c_k\Delta_k \quad \text{and} \quad \theta_k^- = \theta_k - c_k\Delta_k. \tag{17}$$

Both θ_k^+ and θ_k^- are now assigned to the network by using the specified assignment model, STAQ in this thesis. To do this, first the vectors are translated back to matrices D_k^+ and D_k^-. Then these are assigned to the network, resulting in values for flow and speed on all links, $(v(D_k^+), s(D_k^+))$ and $(v(D_k^-), s(D_k^-))$.

Next for both situations the flow and speed at the links are compared to the available measured values $\tilde{v}$ and $\tilde{s}$. The differences are calculated according to the objective function as defined in section 3.3. The value of the objective funtion then gives a final measure of the quality of both perturbed OD matrices D_k^+ and D_k^-. These values are $z(D_k^+)$ and $z(D_k^-)$.

Using the values of the objective function $z(D_k^+)$ and $z(D_k^-)$, the gradient approximation is found by function 14 and 15. This gives the vector $\hat{g}_k$ with values for every nonzero entry of the current OD matrix calibration vector θ_k.

Now the values of the current vector are updated according to

$$\theta_{k+1} = \theta_k - a_k\hat{g}_k(\theta_k) \tag{18}$$

respectively.

This new vector is now transformed to the matrix D_{k+1} and the function value $z(D_{k+1})$ is calculated. If the convergence criteria are still not fulfilled, the vector θ_{k+1} is perturbed and the whole algorithm repeats.

3.6 Convergence criteria

There are different ways to evaluate the quality of the approximation and to decide whether the algorithm needs to iterate further or if the approximation is good enough to end the computation.

The first and most straightforward way is to stop whenever the normalized objective function values is low, so below a value chosen beforehand. This includes that most of the values for flow and speed are close to the measured values.

This objective function value is given by $z(D_k)$. This value is normalized by the number of measurement points, denoted by $|L_m|$. So this convergence criterion gives that whenever

$$\frac{z(D_k)}{|L_m|} < \kappa \tag{19}$$

the algorithm stops with D_k as the optimal solution, where κ is a fixed value chosen beforehand.

If $|L_m|$ is large or if the measurements are not good enough, this criterion might never be reached. Then the algorithm needs to stop due to some other criterion.

One possible other criterion is to stop when the number of iterations exceeds a beforehand chosen number k_{max}:

$$k > k_{max} \tag{20}$$

Here the last matrix D_k with $k = k_{max}$ gives the best solution. The same can also be done for the number of assignments of STAQ.

Another criterion would be to stop when the function value does not change significantly anymore. So with some chosen ϵ

$$|z(\theta_k) - z(\theta_{k-q})| < \epsilon \tag{21}$$

holds with $q \in \mathbb{N}$. If q is for instance 5, then the algorithm stops whenever after 5 iterations the function values only changed by at most the amount of ϵ.

Another possible convergence criterion is the often applied T-test. The T-test is used to measure the result of the calibration concerning the flow, hence the found values $v(D^*)$ relative to the measured values $\tilde{v}$. These are compared according to equation 22.

$$T_l = \ln\left(\frac{(v_l(D^*) - \tilde{v}_l)^2}{\tilde{v}_l}\right) \tag{22}$$

with ln denoting the natural logarithm. The value of T_l is calculated at every measurement position $l \in L_m$. All values of T_l need to fulfill the following condition: $T_l < 3.5$.

Another criterion would be to measure the absolute differences in flow or speed. This gives

$$T_l^v = |v_l(D^*) - \tilde{v}_l| < \omega_v \quad \text{and} \quad T_l^s = |s_l(D^*) - \tilde{s}_l| < \omega_s \tag{23}$$

These values need to be smaller than some beforehand chosen values ω_v and ω_s.

All these criteria can be used individually or can be combined such that the moment one or several of them hold, the algorithm will terminate. For the basic instance of SPSA, the criteria concerning the function value are used as well as the maximum number of iterations. The criterion concerning the difference of the objective function value is not used in this thesis to prevent the algorithm from stopping too early, although it might be a good criterion when applied with care. The T-test is also not used in this thesis, for reasons that are explained later in section 4.7. The absolute difference in flow is not needed, but the absolute difference in speed is used later in section 5.

4 Test networks

In this section three different test networks are used for the investigation of using SPSA in combination with STAQ. There are several different situations that might influence the performance of the algorithm. This can cause the algorithm to converge very slow or to end up at a local optimal solution. These solutions are called local minima. For the optimization it is helpful to decrease the number of local minima in the objective function, since otherwise the actual optimum might not be found.

For this, first some test networks are introduced to test the different influences. These networks are given in section 4.1. Then the reasons for the local minima are described in section 4.2. In sections 4.3 and 4.4 options to decrease the number of local minima are found. In section 4.5 a possible solution for situations where the algorithm iterates to values outside of the searched solution space is proposed. The performance of this enhanced method is tested in section 4.6 and finally in section 4.7 the performance of this enhanced method is further improved. Section 4.8 gives a short conclusion of the results of these sections.

4.1 The three test networks

To test the different situations for local minima, some test networks are used before the algorithm can be applied to real networks.

The first test network is from Frederix (2013, chapter 3) [6]. Here he uses a simple corridor network with only one OD pair and 8 links. It has one link whose capacity is smaller than the expected demand, such that congestion occurs and a queue is formed. This link is called the bottleneck of the network. The network looks as shown in figure 3.

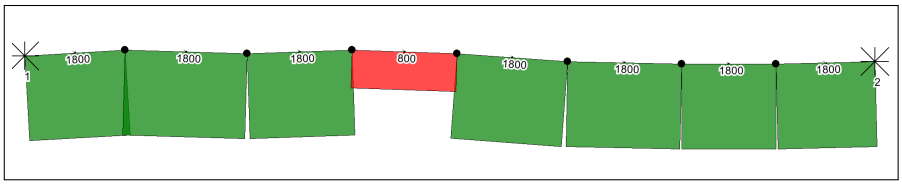


Figure 3: Plot of network 1 with capacity in red if smaller than the expected demand and green if capacity is larger than expected demand

Here the OD demand goes from the left part, denoted by centroid 1, to the right part, denoted by centroid 2. The fourth link, indicated by red, is the bottleneck of this network.

The next network to investigate is called matchstick network, since it is only a straight line as network 1. It consists of 5 links and has one OD pair. Different from network 1, it has two different bottlenecks at links 2 and 4. It can be seen in figure 4.

Here again the OD demand goes from left to right, from centroid 1 to centroid 2. The bottlenecks, indicated in orange and red, have a capacity of 2000 vehicles per hour and 1000 vehicles per hour. With this setting the influences of a second bottleneck on as well the first bottleneck as the whole network can be investigated.

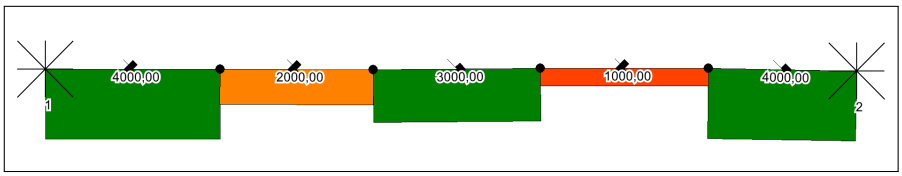


Figure 4: Plot of matchstick network with capacity in red if smaller than expected demand, orange if capacity is smaller than some demands and green if capacity is larger than most demands

Since a network with only one OD pair is not really realistic, the next test network consists of two OD pairs. This is a diverge network, where demand comes from only one origin, the bottom centroid, but has two different destinations, the left centroid and the right centroid. This network is given in Figure 5.

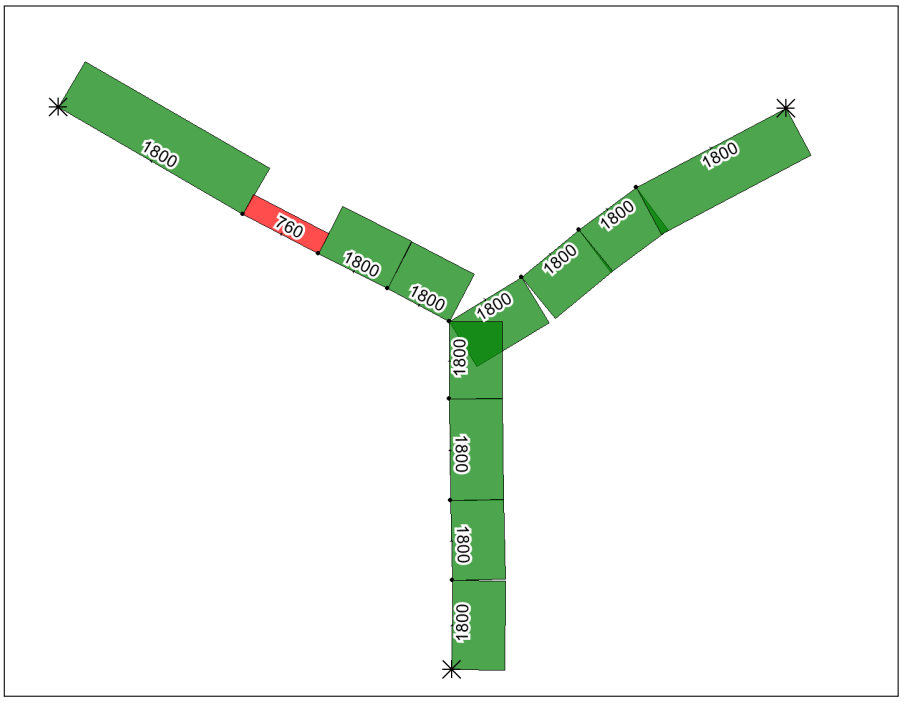


Figure 5: Plot of network 2 with capacity in red if smaller than most demands and green if capacity is larger than most demands

Here the capacity of the second to last link of the left part is smaller than the expected demand, so the queue might propagate back until the point where the routes split and even further back until the origin of the network. This queue might also influence the right part of the network.

These three test networks are used for the further investigations in this section.

4.2 How local minima occur

This section deals with the topic of local minima and investigates why they occur. Local minima define values of OD demand where the gradient of the objective function is zero, but the function value is not optimal, hence there is at least one other set of OD demands where the value of the objective function is smaller, the global minimum. Local minima are present whenever the objective function is not convex with respect to the OD matrix D. Due to the complex relation between the OD matrix D and the values $v(D)$ and $s(D)$ caused by the assignment, this might always be the case. To understand what exactly causes some of the local minima, plots of OD demand versus link flow and OD demand versus link speed are made for the test networks: network 1, the matchstick network and network 2.

To explain the situations, we differentiate between two traffic regimes as shown in figure 6: the congested regime and the free flow regime. In the free flow regime, the flow at the link is lower than the capacity and the average speed lies between the value for speed at capacity and the free flow speed. When the demand at the link approaches the capacity, the average speed decreases from the value of the free flow speed to the value of speed at capacity. When the demand exceeds the capacity, the link becomes congested. If a link is in the congested regime, the average speed drops below the value for speed at capacity and the flow at the link decreases as well. A link at capacity is considered to be congested.

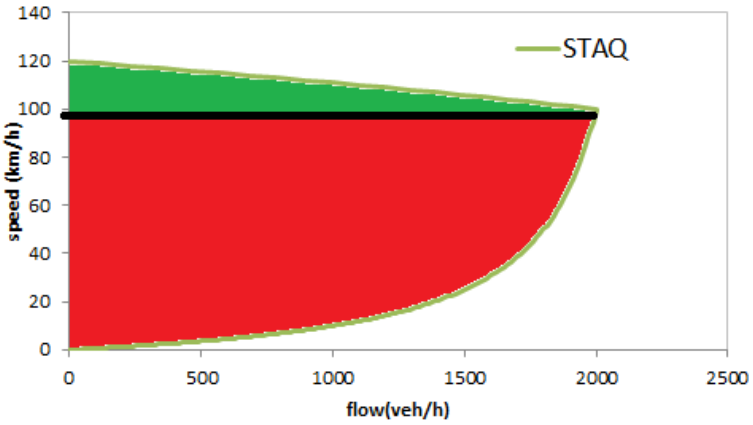


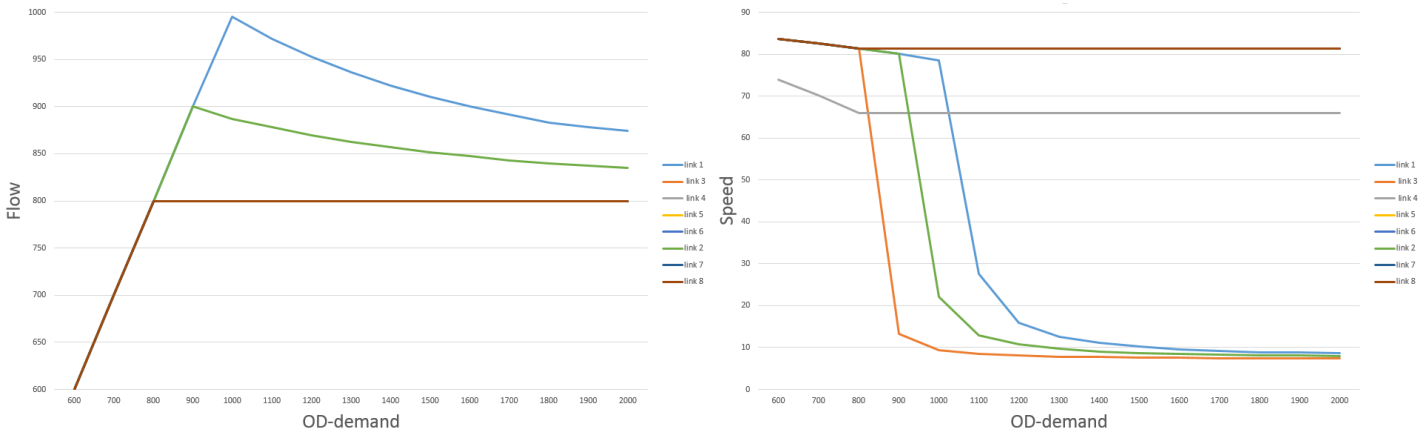
Figure 6: Plot of fundamental diagram with in green the free flow regime and in red the congested regime - the free flow speed is 120, the speed at capacity 100 and the capacity is 2000

To find the reasons for the presence of local minima, the three different test networks from section 4.1 are investigated. For these tests, a true demand is assumed, such that we can generate the vector of observed flows and speeds. In the tests we can choose a different start demand matrix and observe to what extent the method is able to revert to the true OD matrix. The true demand is given by the OD matrix D_m. With this matrix the assignment is performed using STAQ and the corresponding values for $\tilde{v}$ and $\tilde{s}$ are found:

$$D_m \xrightarrow[STAQ]{} (\tilde{v}, \tilde{s}) \tag{24}$$

4.2.1 Local minima in network 1

The first network to investigate is network 1 as described in section 4.1. It can be seen in figure 8 with different colors for each link. The plots of the flow and speed at the links for different OD demands are shown in figures 7a and 7b with the corresponding colors for the different links.



(a) Flow network 1

(b) Speed network 1

Figure 7: Flow and speed of network 1 for different demands, demand is shown on the x-axis whereas the link flow and link speed are shown on the y-axis respectively

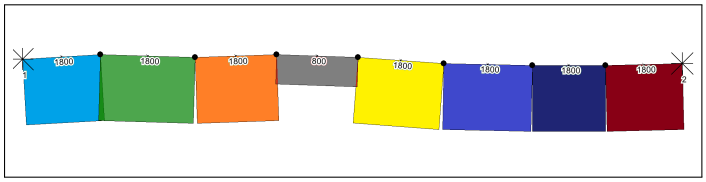


Figure 8: Network 1 with different colors for the links

In figure 7a the flow is shown for different OD demands and the lines indicate the different links 1 to 8, where only links 1, 2 and 8 are visible, the other graphs of links 3-7 are equal to the graph of link 8. It can be seen that for the links 1 and 2 (blue and green curve) the flow is not bijective and hence there are two different OD demands possible for the same link flow. Each of the two OD demands gives a different state of the link, congested or free flow. So for some values of flow, there are two possible situations, one in the free flow regime (left of the peak) and one in the congested regime (right of the peak). For further insight into these situations, more knowledge about STAQ is required, which is outside of the scope of this thesis.

Figure 7b shows the values of the average speed on the different links according to different OD demands. Note that the functions for speed are all bijective, hence there is only one situation for every possible average speed. Also it can be seen that the speed decreases when the links becomes congested. Again further research requires more knowledge of STAQ.

To understand how the local minima occur, consider the following example: Assume the measured OD matrix D_m has the non-zero entry 1050. According to the figures for flow and speed, this produces a link flow of 982 and a speed of 49 km/h at the first link. So $\tilde{v} = 982$ and $\tilde{s} = 49$. However, a flow of 982 can also be produced by an OD demand of 982, see figure 7a. For a demand of 982 in the OD matrix, after the assignment is done, the corresponding values for flow and speed would be 982 and 79 respectively.

Assume now that there is a measurement at link 1 which gives the measured values for flow and speed by $\tilde{v} = 982$ vehicles/hour for the flow and $\tilde{s} = 49$km/h for the speed. This means that for the OD demand of 982, the difference in flow is zero, but the speed has a difference of 30. This results in a local minimum, since a slightly higher or lower demand than 982 produces a larger difference for the flow whereas the speed difference remains nearly the same. To see the local minimum around the value of 982, figure 9 gives the function value with the described measured values for flow and speed for different OD demands.

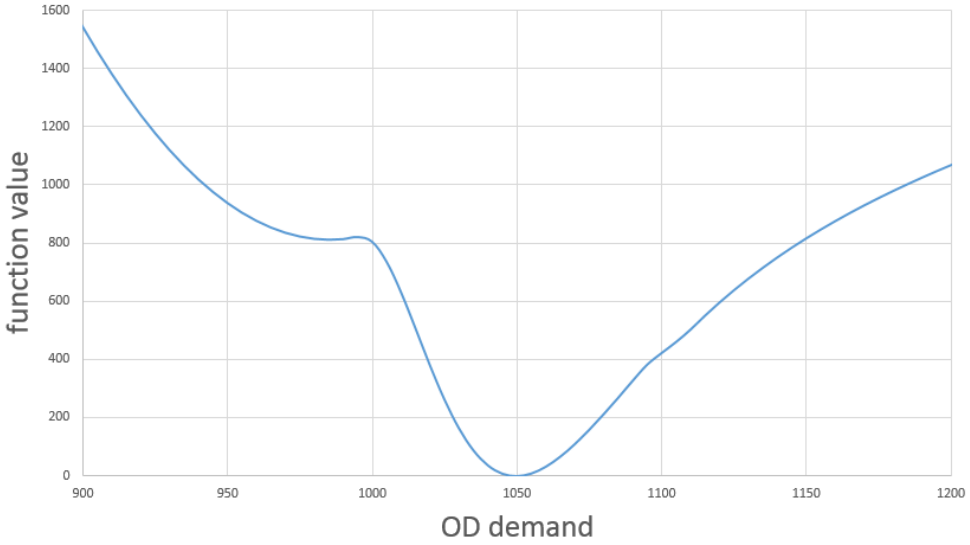


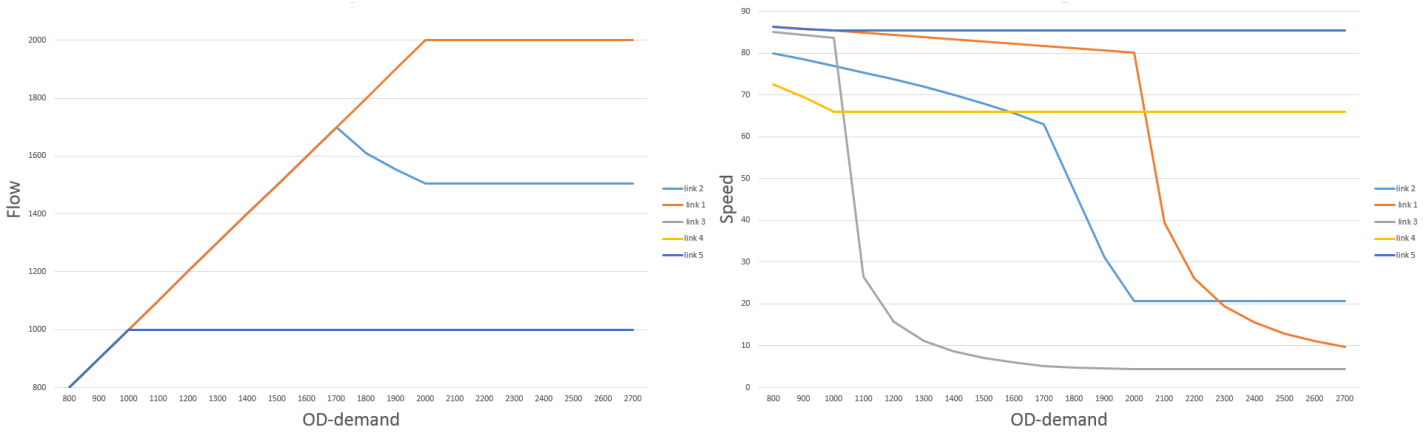
Figure 9: Objective function values for different OD demand of network 1 for measured values of of $\tilde{v} = 982$ and $\tilde{s} = 49$

Around an OD demand of 982 vehicles/hour a small local minimum can be seen, whereas the global minimum is at 1050 vehicles/hour, as expected. This is a typical situation where a local minimum occurs. This local minimum occured since there is more than one element in the objective function and because the flow is not bijective due to the OD demand. There are more similar situations possible for network 1, but in this example the problem is most obvious. When the algorithm reaches the local minimum, it can get stuck there since the gradient points in the direction of the local minimum instead of pointing to the global minimum.

This problem can also occur in other networks.

4.2.2 Local minima in the matchstick network

The matchstick network has characteristics similar to network 1, but there are some different aspects as well. Most important, this network has two potential bottlenecks to deal with, such that it can be investigated how they might influence each other. The plots of the flow and speed for different OD demands are shown in figures 10a and 10b and the corresponding colors for the links are given in figure 11.



(a) Flow matchstick network

(b) Speed matchstick network

Figure 10: Flow and speed of the matchstick network for different demands

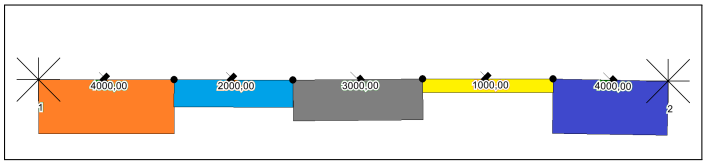


Figure 11: Matchstick network with different colors for the links

In figure 10a the link flow on the different links is shown. The graph for link 1,2 and 5 is visible, whereas the graphs of links 3 and 4 are equal to the graph for link 5. Again no more details are explained, since the characteristics depend on the assignment model STAQ. However, it can be seen that the flow is bijective everywhere except for link 2, the light blue curve, between the OD demands of 1520 and 2000. Another thing to notice is that from the demand of 2000 on, the flow does not change anymore. This is due to the second bottleneck at link 2. It becomes active at a demand of 2000 and thus no more flow can pass any of the links. This interaction between the bottlenecks makes the investigation of larger networks more complicated.

The speed corresponding to the different values of demand is given in figure 10b. Here again the speed decreases whenever the link gets congested.

In this network almost all flow and speed functions are bijective, except the flow function of link 2 between OD demand 1520 and 2000. After this the flow does not change anymore and hence

is bijective from here on. Measurements on this link are used to show the presence of local minima. In this example we assume that the measured values on link 2 are $\tilde{v} = 1613.4$ vehicles/hour and $\tilde{s} = 53.5$ km/h for the speed. Figure 12 gives the function values for different OD demands.

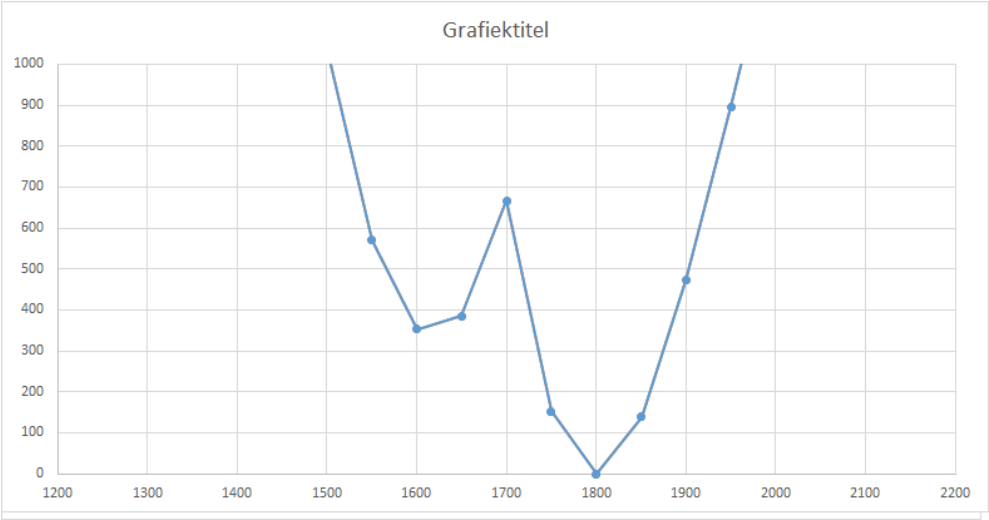


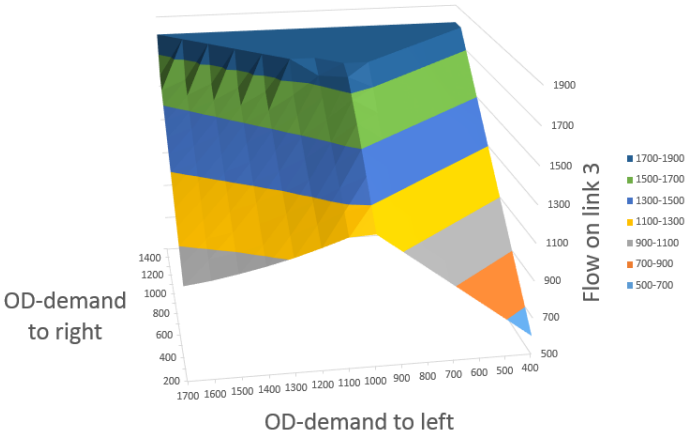
Figure 12: Objective function values for different OD demand of the matchstick network with measured values $\tilde{v} = 1613.4$ and $\tilde{s} = 53.5$

There is a local minimum around an OD demand of 1613.4. Here the difference in flow becomes zero, whereas the difference in speed is still high. When starting from a demand below or near this local minimum, the algorithm is likely to converge to this value instead of converging to the global minimum with an OD demand of 1800.

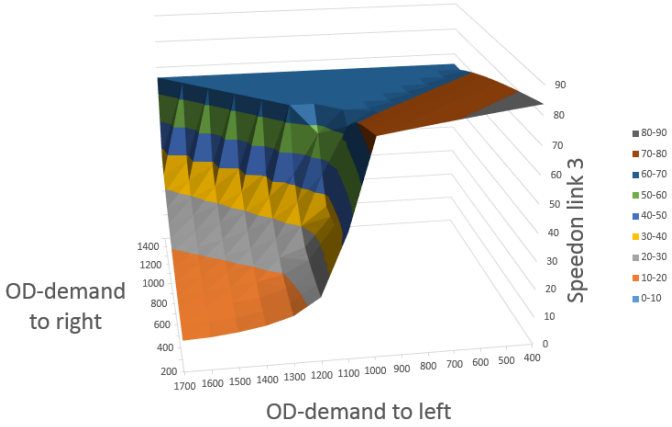
4.2.3 Local minima in network 2

In network 2 the situation becomes even more complicated. Here two OD pairs are present, so there are two values of OD demand. The whole demand originates from one point, at the bottom of the figure, but it splits after the fourth link. The corresponding plots for the flow and speed are shown in 2-dimensional plots for every link, where the different OD demand is given on the bottom axes and the vertical axis gives the values for flow or for speed. The OD demand is differentiated in the demand to the right and to the left of the network as given in figure 14. This gives figures 13a and 13b for link 3, whose location can be seen in figure 14 in orange.

Again it can be seen that the flow function is not bijective. In figure 13a there is a straight area, coloured in dark blue, from which the flow drops to two sides, denoting the free flow regimes (right side) and the congested regime (left side). The link flow is hence not bijective with respect to the OD demand for the left side. Note that in figure 13b the speed is bijective everywhere.



(a) Flow network 2



(b) Speed network 2

Figure 13: Flow and speed of network 2 for different demands

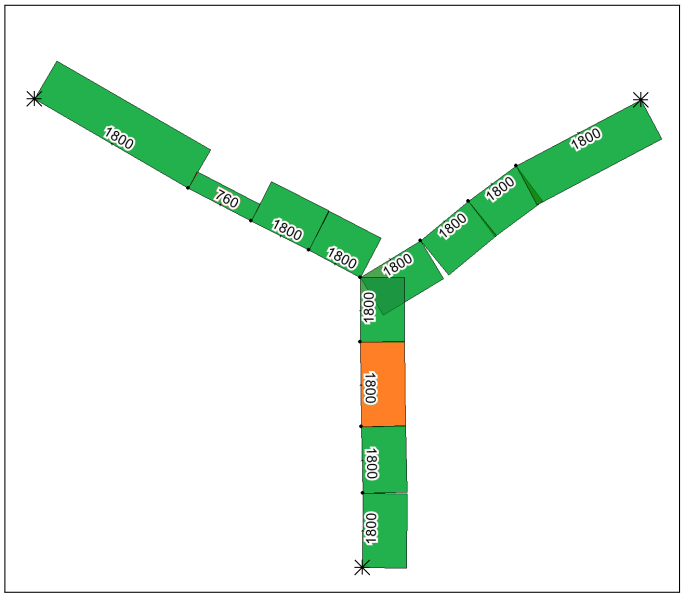


Figure 14: Network 2 with link 3 in orange

Assume that the measured matrix D_m has OD demands of 1200 for the OD pair from the bottom to the left upper side and 400 for the OD pair from the bottom to the right upper side. These give some measured values for flow and speed for link 3. This results in figure 15.

It can be seen that there is some sort of local optimal corridor (orange) at the right side, where the algorithm might end up in. So there is not only one local minimum like in the networks before,

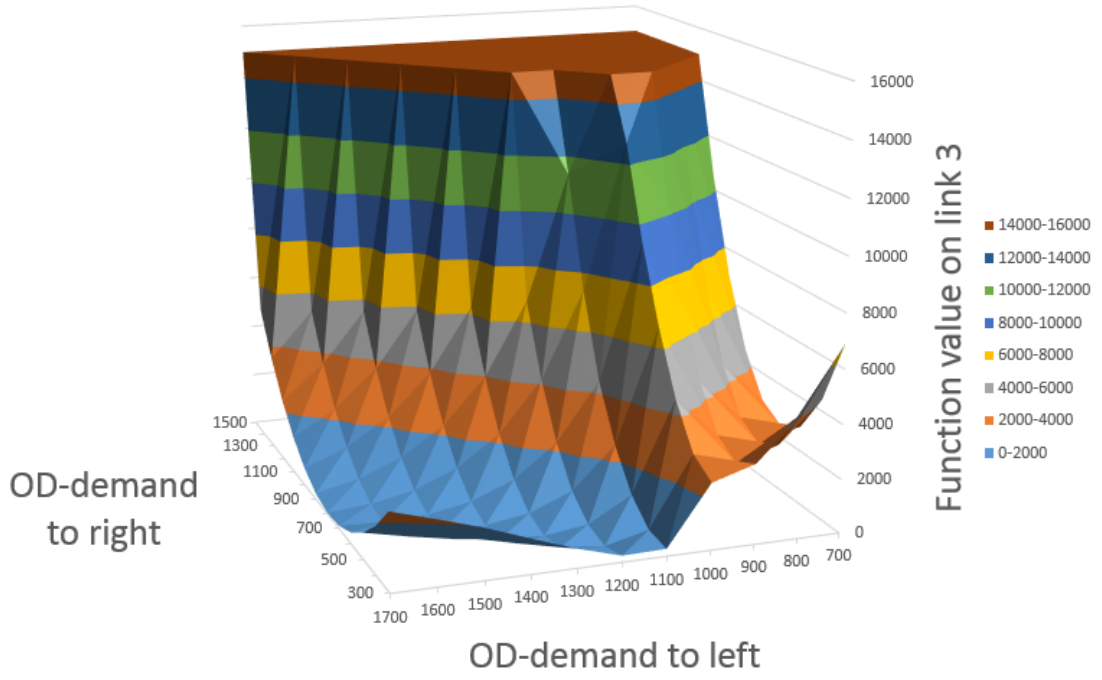


Figure 15: Function values for different demand of network 2 with measured demand 1200 and 400

but a whole set of combinations which end up in local minima. This means that with increasing dimension there are more and more local minima possible and we need to find a way to handle them efficiently.

4.2.4 Conclusion local minima

Local minima can always occur in the objective function. Especially if the flow is not bijective with respect to the OD demand and there is another element in the objective function, like the speed, a lot of local minima can be present. For more than one dimension (OD pair) there is also the possibility that a corridor of local minima is present. With an increasing number of dimensions, also the number of possible local minima increases, which makes the problem even more relevant for real world situations, where a lot more dimensions are present.

4.3 Starting in the correct traffic regime

Due to the fact that local minima might cause the algorithm to converge to a non-optimal solution, we would like to make sure that the algorithm is converging to the optimal solution.

In his chapter 4, Frederix (2013) [6] investigates the causes of local minima and gets to the same results as shown in section 4.2: The fact that the relation between the OD demand and the link flow is not bijective causes most of the local minima in the objective function. These local minima are situations where at least one of the values for speed is in a different regime than the measured value for speed here. So $\tilde{s}$ and $s(D_k)$ are in two different regimes.

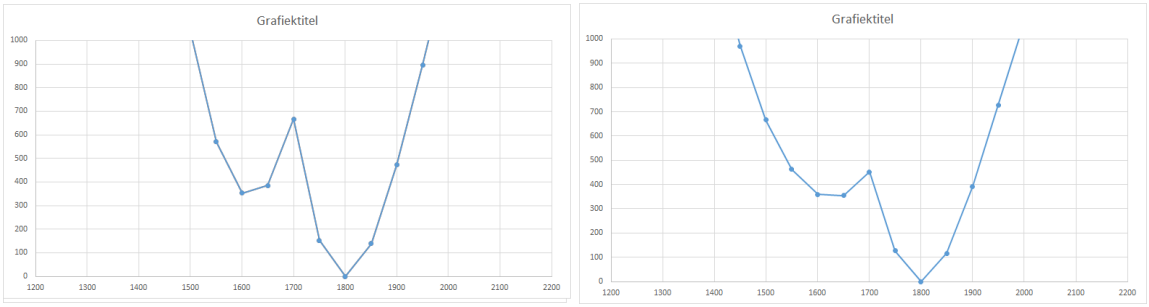
Frederix [6] states an idea to avoid the problems that the presence of local minima causes: He proposes to make sure that the algorithm starts at a good place. This means that the algorithm should start at a point where at all measured points the correct traffic regime is given. To find this good starting point, he proposes to perform the optimization with an objective function that is bijective: by only taking the speed differences into account. To understand why this is an advantage, we consider the plot of the function values for measurements on link 2 of the matchstick network for a measured demand of 1800. Here we found a local minimum at an OD demand of 1613. Now we decrease the weight on the flow λ, see equation 10, and thus increase the weight on the speed until the weight on the speed is 1 and the weight on the flow is 0. The resulting functions can be seen in figures 33a until 33h in the appendix. Some selected plots are shown in figures 16a until 16d.

The local minimum at an OD demand of 1613 vanishes with decreasing value for the weight on the flow λ. As can be seen in the last figure, there is no local minimum anymore, only the global minimum is left. This is due to the fact that the function of link speed against OD demand is bijective and hence there is only one situation where the correct link speed is reached, which is near the global optimum.

The same can be seen for the two-dimensional situation in network 2. Here the measured matrix D_m has an OD demand of 1200 and 400 for the two OD pairs. For the standard weight of $\lambda = 0.05$ a corridor of local minima occurs. This corridor slightly vanishes the smaller the value of λ gets. Plots of this behaviour are found in Figures 34a until 34f in the appendix. Here it can be seen that when we only consider the influence of the speed, most local minima that might cause problems vanish.

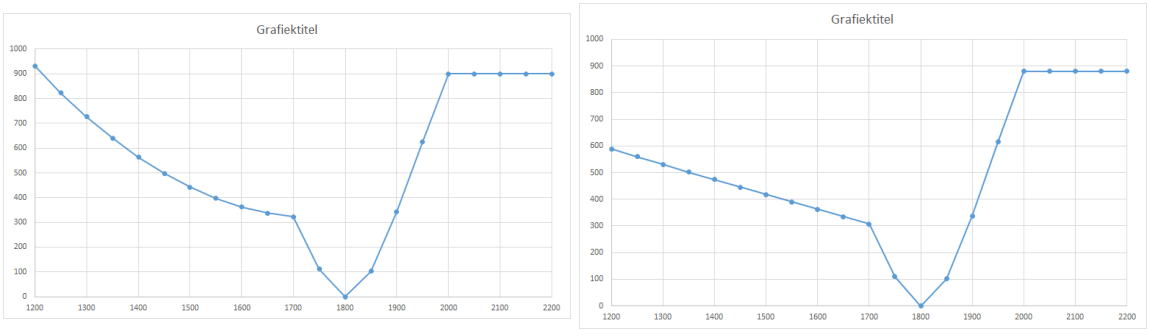
This gives the following method to find a good starting point: We choose some initial matrix D_0. The objective function of the SPSA algorithm is changed such that only the speed is considered, hence the weight on the speed is 1 and the weight on the flow is 0. Now the algorithm is performed to find a solution where the modelled speed $s(D_k)$ is reasonably close to the measured speed $\tilde{s}$, $s(D_k) \approx \tilde{s}$.

This procedure is equivalent to changing the OD demand in the matrix D such that the measured and the approximated speeds all lie in the same traffic regime and thus close to each other. This means if the speed $\tilde{s}$ at link 1 is measured to lie in the congested regime, then also the modelled speed, $s(D_k)$ needs to lie in the congested regime. The algorithm can stop whenever all measured links are in the correct regime, hence when all the differences in speed are small. We can choose the value of κ such that the algorithm only converges if the result is close to this situation or the absolute difference of the speed is calculated and the algorithm ends when all differences are small.



(a) Weight flow 0.05

(b) Weight flow 0.02



(c) Weight flow 0.002

(d) Only speed

Figure 16: Function values for a measured demand of 1800 on link 2 of the matchstick network with different weights on flow

4.4 Staying in the correct traffic regime

4.4.1 Line Search

To avoid the algorithm to leave the correct traffic regime, Frederix (2013) [6] proposed to use a line search procedure instead of a sequence for a_k to avoid too large step sizes during the optimization process. This appears to be a good solution to the issue of staying in the correct traffic regime. With the line search the step size is determined by the function values. In the normal algorithm, the step size depends on the size of the gradient and the choice of the sequence. The main problem with the sequence a_k is that the step size taken might be too large and the algorithm iterates to the wrong traffic regime. With the line search, the step is only taken as far as the function value decreases. This decreases the chances of ending up in the wrong traffic regime.

In the field of line search there are many different procedures. They can be divided in two fields, the exact line search and the inexact line search. Due to the high complexity of the assignment step, the function $z(D_k)$ is not found explicitly. For that reason an exact line search is not possible in the OD matrix calibration process.

The other option is to find a 'quite' good step size such that the objective value decreases as much as possible. This option is called inexact line search and has several popular methods. By virtue of the efficiency of the golden section procedure, this line search strategy is used in this

thesis. The golden section search is a technique which iteratively narrows the search area.

At the beginning two points are chosen between which an optimum is searched. In the OD matrix calibration the first point is the current vector θ_k and the other point is the current vector θ_k plus a chosen maximum stepsize times the normed gradient approximation. So,

$$\begin{aligned} a &= \theta_k \\ b &= \theta_k + \alpha \hat{g}_k(\theta_k) \end{aligned} \tag{25}$$

The choice of the value α depends on the situation to be optimized. A high value means that a large area needs to be searched, which results in a high number of assignments. A small number might be too small to actually find the area where the minimum lies. In general, α is chosen such that there is a minimum between a and b. However, since there is no knowledge about the objective function z, it is not always sure whether there is a minimum between a and b.

To find a good value of α, a value of a maximum step size (maxstep) can be chosen. This value defines the maximum change of values for the matrix in one iteration. The maximum step size is chosen to decrease with the same factor as c_k does in order to prevent too big steps in later iterations. The stepsize for every entry of θ is now calculated in a way such that the biggest step taken is as big as the value of maxstep and all others are smaller, depending on the value of the normed gradient. So,

$$\alpha = \frac{\text{maxstep} * \hat{g}_k}{\max_i \hat{g}_{ki}} \tag{26}$$

Once the values of a and b are found, two inner points, x_1 and x_2 are calculated according to the golden ratio

$$\varphi = \frac{\sqrt{5}-1}{2} \tag{27}$$

such that

$$\begin{aligned} x_1 &= \theta_k + \alpha \hat{g}_k(\theta_k)(1-\varphi) = b - \varphi(b-a) \\ x_2 &= \theta_k + \alpha \hat{g}_k(\theta_k)\varphi = a + \varphi(b-a). \end{aligned} \tag{28}$$

Now for the inner points x_1 and x_2 the objective values $z(x_1)$ and $z(x_2)$ are found. In the OD matrix calibration problem this includes to make the assignment for the given vectors x_1 and x_2 and to evaluate the differences between the measured values $\tilde{v}$ and $\tilde{s}$ and the modelled values $v(x_1), s(x_1)$ and $v(x_2), s(x_2)$. Now the objective values of x_1 and x_2 are compared. Two situations are distinguished: $z(x_1) \geq z(x_2)$ and $z(x_1) < z(x_2)$.

Whenever $z(x_1) \geq z(x_2)$ the minimum is expected to lie between the points x_1 and b. Otherwise, as shown in figure 17, the search area is chosen between points a and x_2. The names of the points are changed accordingly like shown in figure 17

For the next iteration the same procedure is repeated for the new points a' and b'. Figure 17 shows the procedure for θ with one dimension.

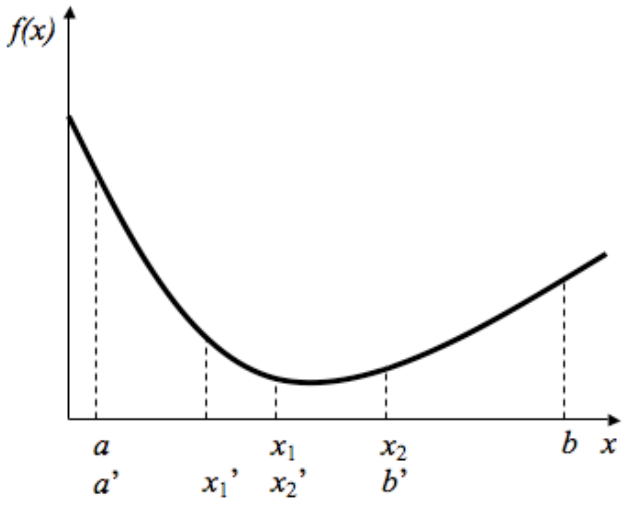


Figure 17: Implementation of the basic line search procedure

This procedure is repeated until the distance between the points x_1' and x_2' is smaller than a chosen value ϵ. To make the search more and more accurate in every iteration, in this thesis ϵ is chosen to decrease with the number of iterations. So $\epsilon_k = \epsilon/k$. So there are two important parameters to perform the line search, which need to be chosen in a good way, the value of maxstep and ϵ. In this thesis it is chosen to set $\epsilon = 1$.

```
gr=(Math.sqrt(5)-1)/2
c=b-gr*(b-a)
d=a+gr*(b-a)
while (c-d).abs>tol
   if f(c) < f(d)
        b=d
        d=c
        c=b-gr*(b-a)
   else
        a=c
        c=d
        d=a+gr*(b-a)
   end
return (b+a)/2
```

Figure 18: Example of golden section search for two iterations

The golden section search is effective if there is actually an optimum between the two starting points a and b. However, this is not always the case during the OD matrix calibration. Since the solution of the line search might be worse than the current, the step $\theta_{k+1} = \theta_k + a_k \hat{g}_k(\theta_k)$ is only done if $z(\theta_{k+1}) < z(\theta_k)$, otherwise it is chosen that $\theta_{k+1} = \theta_k$. This guarantees that after the line search for the new solution θ_{k+1} it holds that $\theta_{k+1} \leq \theta_k$. Due to the random character of the

vector Δ the algorithm does not get stuck here, but the function value decreases further after one or more iterations.

Note that for this reason when using the convergence criterion concerning the change in function values, q needs to be chosen such that it is large enough to avoid the algorithm to stop too early at a non-optimal solution.

4.4.2 Objective function with binary variables

After obtaining an OD matrix D^* such that $s(D^*) \approx \tilde{s}$, it is required to assure that the algorithm stays in the correct traffic regime. To find the optimal solution a different objective function is used during this part of the optimization process. After the correct initial traffic regime is found by optimizing only the speed, in the next step also the flow needs to be considered. At the same time we want to make sure that the algorithm remains in the correct regime without adding local minima in the objective function. This is done by converting the speed into binary variables which state whether the link is in the congested or free flow regime. So,

$$r_l = \begin{cases} 1 \text{ if congested} \\ 0 \text{ if free flow} \end{cases} \tag{29}$$

This is done based on the value of the speed at capacity, denoted by s_c. Then it follows that

$$r_l = \begin{cases} 1 \text{ if } s_l \leq s_c \\ 0 \text{ if } s_l > s_c \end{cases} \tag{30}$$

The corresponding objective function then looks like this:

$$z(D) = \frac{1}{|L_m|} \left[\sum_{l \in L_m} (\tilde{v}_l - v_l(D)^2 + B \sum_{l \in L_m} |r_l(D) - \tilde{r}_l| \right] \tag{31}$$

where $r_l(D)$ is the regime in the modelled situation and $\tilde{r}_l$ gives the regime in the measured situation. B is a parameter that gets a value defining the size of the penalty for reaching the wrong traffic regime. This value should be chosen a lot higher than the squared flow values. For now it is chosen to be 10,000,000.

However, if the measured speed is near the speed at capacity, the penalty for the wrong regime is often present whereas the current approximation is close to the solution. To avoid long calculation times because of a slightly wrong regime, the penalty for the regime is not applied in these situations. So whenever the distance between the measured and the estimated value for the speed is small for a link l, they are assumed to be in the same regime and the difference $r_l(D) - \tilde{r}_l$ is set to zero.

An example of the corresponding function values for the matchstick network is shown in figure 19.

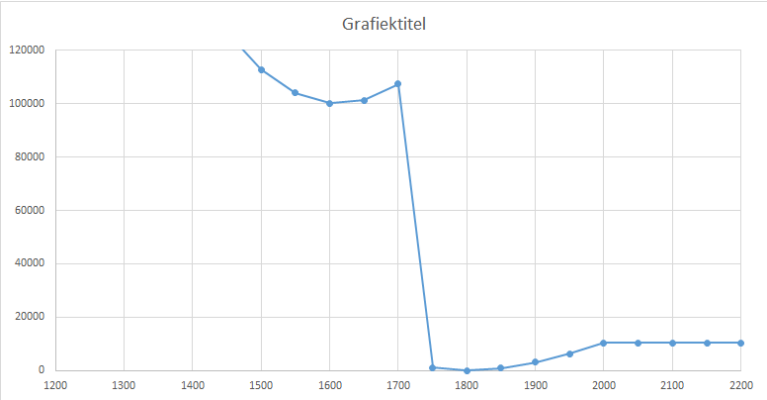


Figure 19: Example of function values of network 2 with new objective function

The local minimum at 1613 can still be seen in this figure, so there is no advantage in using this new objective function without performing the speed optimization first. When doing this, the algorithm finds a starting point right of the optimal value. By using the new objective function it is made sure that the line search is unlikely to take a step that leads back to the wrong traffic regime at the left of the figure.

So if the algorithm starts at the right side of the peak after the speed optimization, there is a high chance of finding the global optimum and not getting stuck in the local optimum. The algorithm stops according to the chosen convergence criteria in section 3.6.

4.5 Bounds

Upper and lower bounds on the entries of the OD matrix are used to avoid infeasible solutions and to restrict the search space. They help to avoid situations where most of the network is congested. Especially when a link that is connected to an origin is congested, this demand can not enter the network. These situations are mostly unwanted and difficult to calibrate.

Most literature proposes to use the difference to the initial matrix in the objective function to prevent these situations. However, since in section 4.2 it is found that local minima can occur due to an objective function with several different elements from which one or more are not bijective, this choice might cause local minima as well. Since there is more than one element considered in the objective function, there might be situations where one of the elements is optimal whereas the other elements are not. For example, the initial matrix we start with is equal to the initial matrix, hence the difference here is zero, but the flow and speed differences might be quite high. On the other hand there might be a solution where the flow and speed differences are small, but the difference to the initial matrix is high. This can cause additional local minima in the objective function and thus should be avoided.

So another way to prevent the algorithm from taking on values that are too high or too low, is considered. First the initial matrix is regarded. Here the sums of the rows, hence the total amount of vehicles leaving this centroid, are calculated. Next this amount is multiplied with chosen factors for the upper and lower bound. This gives the values for the upper and lower bound for every row total during the algorithm.

During the SPSA algorithm, the sums of the matrix rows are compared to their upper and lower bounds. Whenever the sum is too large or too small, a large value, for now chosen to be 100000 is added to the objective function. This prevents the algorithm to iterate towards a solution where the matrix is too far away from the initial matrix.

In this thesis different factors for the upper and lower bound are used, depending on the situation at hand. They are also influenced by the difference to the row totals of the searched OD matrix. If the row totals of the searched OD matrix do not lie within the bounds, the algorithm is not able to find this optimal solution.

With this method the objective function is kept as simple as possible in order to avoid more local minima.

4.6 Application of the enhanced method

The enhanced method now consists of all the parts that help to avoid the problems of local minima and other situations such as a too congested network. Additionally, line search is used according to the gradient and the chosen step size as explained above. First the optimization is performed using only the speed measurements. This results in a matrix that is a quite good approximation of the situation. Then this matrix is used for another SPSA application, where the binary variables for the speed are used and the flow is optimized. Here the line search helps to avoid the wrong traffic regime. In both SPSA applications, the bounds, see section 4.5, are used to restrict the search area.

To test the performance of this enhanced method, tests are run for different situations for network 2. Here some situations are found where local minima occur such that the original algorithm is not able to find the global solution. Then these situations are tested again with the enhanced method and the results are compared.

For the tests the measured demand is chosen to be 1200 for the demand from the bottom to the left and 400 for the demand from the bottom to the right. These demands are considered the true matrix D_m. When applying STAQ we get the values $\tilde{v}$ and $\tilde{s}$ for all links. The initial matrix is always chosen to have demands 700 and 400 respectively. This means that in the begin situation, none of the links is congested, whereas in the measured situation some of the links are congested.

To investigate different situations, the measurement locations for the values $\tilde{v}$ and $\tilde{s}$ are selected. For now we only consider one or two measurement locations. The standard objective function uses both flow and speed with weights 0.05 and 0.95 respectively. The results of the tests are shown in table 1, where position 1 and 2 give the positions of the two measurements. To make the table easier to interpret, the links are named the following: Links 1-4 belong to the common part at the bottom, links 5-8 to the right part and links 9-12 to the left part after the diverge point.

The quality of the convergence is interpreted in different ways: The term small value means that the function value is small at the end of the performance, but the solution might be far away from the demand 1200 and 400. However, since the aim of the OD matrix calibration is to find a matrix D^*such that the values $v(D^*)$ and $s(D^*)$ are close to the measured values $\tilde{v}$ and $\tilde{s}$, a small function value means that this goal is reached. Local minimum means that the algorithm converges to a local minimum where the algorithm is not able to leave again. These situations need to be considered further.

For these situations the enhanced method is tested. The results of this testing are shown in table 2.

position1	position 2	convergence
common part	none	local minimum
left/right part	none	small value
common part	common part	local minimum
common part links 1+2	left part	local minimum
common part links 1+2	right part	correct
common part link 3	left part	local minimum
common part link 3	right part	correct
common part link 4	right part	local minimum
common part link 4	left part	correct
right/left part	right/left part	small value/correct

Table 1: Qualitative table network 2 with normal SPSA

position1	position 2	convergence
common part	none	correct/small value
common part	common part	correct/small value
common part links 1+2	left part	correct/small value
common part link 3	left part	correct/small value
common part link 4	right part	correct/small value

Table 2: Qualitative table network 2 with enhanced method

In all of the cases that go wrong with the normal method, the enhanced method finds a solution with a satisfying small value. This mostly depends on the choice of the parameters, especially the choice of convergence criterion for the speed optimization. If this is chosen in a good way, the second part can quite easily find the correct solution and the chances of ending up in a local minimum are low. To achieve this, it is mostly the best to perform the speed optimization until a high precision or until the maximum number of iterations is reached. Then the chances to find the correct solution are very good.

In total this enhanced method solves the problem of local minima for all the cases with one or two measurements. However, the algorithm might not always be able to solve the situation with more measurements. The more measurement positions, the more difficult it gets to find a solution with a small difference. The enhanced method has a much higher chance to find the correct solution than the former method. Due to this the enhanced method is used for all subsequent tests.

4.7 Improving the performance

The number of assignments required to find the solution are important to measure the performance of the algorithm. The less assignments are required, the less time the algorithm needs, especially for large sized networks. The number of assignments for network 2 are given in table 3 for several runs numbered 1 to 5 and two different matrices. The number of assignments are given for two different optimization techniques. The first technique is the enhanced method where first an optimization

for the speed is done and then both flow and speed are used in the objective function as given in section 4.4.2. The second technique shows the number of assignments when only the optimization for the speed is done and the second part is not performed.

runs	Matrix 1		Matrix 2	
run number	speed and flow	only speed	speed and flow	only speed
1	238	6	169	106
2	185	29	155	37
3	80	6	21	16
4	152	79	218	69
5	70	6	75	48

Table 3: Comparison of the two-step procedure and only the optimization for the speed for five different runs and two different matrices

It can be seen that most of the assignment steps are needed for the second part of the algorithm. However, this part is only needed for some last small changes to the OD matrix, so it should not require too much time. So the idea comes up to only use the optimization for speed and then end with the found OD matrix.

This decision is done based on the following consideration: When starting with the measured values $\tilde{v}$ and $\tilde{s}$, it is advisable to check for all links if these values lie on the fundamental diagram of STAQ with these characteristics. If this is not the case, these values are not feasible to find a solution for the OD matrix calibration and hence need to be neglected or need to be changed such that they fit to the fundamental diagram. So whenever $\tilde{v}$ and $\tilde{s}$ lie on the fundamental diagram, the following holds: if $s(D^*) \approx \tilde{s}$ then it follows that also $v(D^*) \approx \tilde{v}$. So whenever the measured values lie on the fundamental diagram, getting close to the observed speed inplies getting close to the observed flow. Note that this does not hold the other way around due to the flow being not bijective.

To use only the speed optimization, a good convergence criterion is required. During the rest of this thesis, this criterion is chosen that the absolute difference between the modelled and observed speed is smaller than 5 km/h for all measurement points as given in section 3.6.

4.8 Conclusion

Local minima can cause problems in the convergence of the algorithm. They can make the algorithm converge slow or even stop at non-optimal solutions. Applying the algorithm first only with speed and then also with flow helps to solve this problem. One remaining problem is that the optimization now needs a high number of assignments to make the last changes for the optimal solution, due to the two-step procedure. However, the optimization for speed in most cases already delivers satisfying close results to the optimal solution. The enhanced procedure can only optimize the speed until the values of the absolute difference lies below 5km/h for most considered measurements. Then we can accept the solution as the optimum. This saves a lot of assignments and makes the algorithm hence a lot faster. This algorithm is next applied to a real world network, to see how it performs.

5 A small real network

In this section a small real life network, called the CHARM network, is investigated. First some information about the network is given and a way to measure the performance of the algorithm is discussed. Then another modification of the algorithm is given, followed by some tests of different parameters of the algorithm. At the end the algorithm is tested with the optimal found set of parameters for different situations on the CHARM network and a conclusion is drawn.

5.1 The CHARM network

In this section the new method is applied to a small real life network to see how it performs under real conditions. The chosen small network is the CHARM network and it reproduces the A58 between Tilburg and Eindhoven in the Netherlands. It consists of 19 centroids and has 330 links. The network is shown in figure 20.

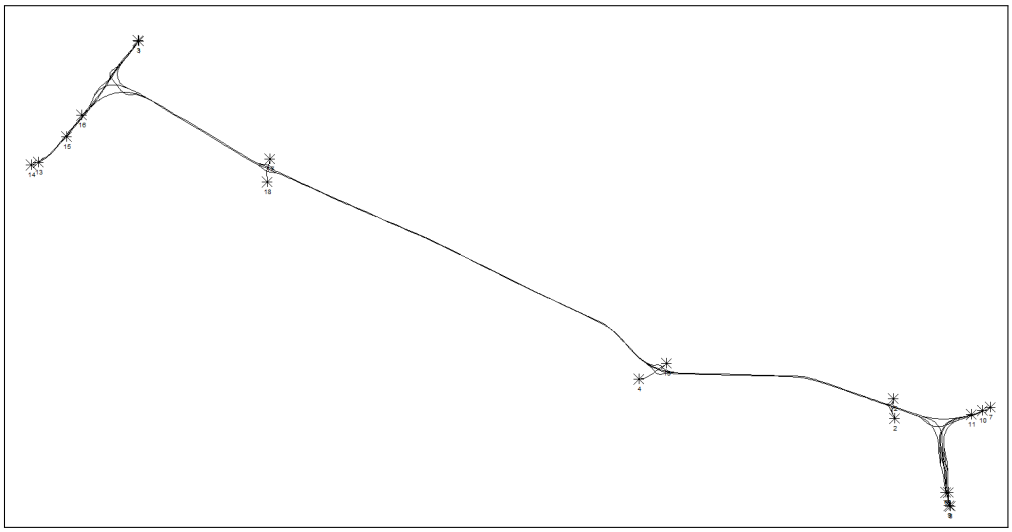


Figure 20: Figure of the CHARM network

The middle part of the network represents the highway, where a free flow speed of 120km/h is allowed. These links have two or more lanes on one side. The other links are connecting roads to the highway with different capacity, number of lanes and free flow speed.

First an original matrix is given for the network. In this matrix there are 89 non-zero OD pairs that can be calibrated. For this original matrix the assignment is performed to give measurement values at chosen links $\tilde{v}$ and $\tilde{s}$. Then the original matrix is changed slightly to give the start-matrix. This change can be done in different ways. For the tests in sections 5.3, 5.4 and 5.5 the relative error of every entry is chosen to be in the interval $[-0.2, 0.2]$. The algorithm is now applied to this new startmatrix to calibrate it such that the final result is reproducing the measured values as good as possible.

5.2 Measures of Performance

To compare the quality of different parameter values and the performance of the modifications of SPSA, some measures to evaluate the resulting approximation are required. We distinguish 3 possible ways of measuring the performance:

- 1) **Measuring the overall performance of the calibration in replicating the observed data**. Here the observed values are compared with the simulated link values and the differences are evaluated in some specific way.
- 2) **Number of assignments**. This measures the speed of the performance and gives information about the number of assignments required to find a satisfying solution. Since in large-scale networks the time for one assignment is quite high, it is important that the number of required assignments until convergence is as small as possible.
- 3) **Computation time**. This measures the actual time the algorithm needs until it converges to a satisfying solution. This can also be approximated by the number of assignments as given in 2) to avoid the influence of any hardware in the measurement of the performance.

For the measuring of the overall performance in replicating the observed data, a lot different measures are possible. Some of them are listed here:

- a) **Total squared difference of the corresponding values** (TSD). Here the differences are calculated and these are squared such that large differences are weighted more. Then the resulting values are added for all compared values, which gives the total squared difference between the observed and the calculated values.

$$TSD = \sum_{l \in L_m} (s_l(D) - \tilde{s}_l)^2 \tag{32}$$

- b) **Absolute difference for every measurement** (AbsD). For every measurement location the absolute difference between the observed values and the modelled values are calculated.

$$AbsD = |s_l(D) - \tilde{s}_l| \tag{33}$$

- c) **Mean Absolute Error** (MAE). Here the absolute differences are calculated as in b), but then the mean of them is taken to give a single value.

$$MAE = \frac{1}{|L_m|} \sum_{l \in L_m} |s_l(D) - \tilde{s}_l| \tag{34}$$

- d) **Root Mean Square Error** (RMSE). Here the differences for every measurement are calculated and squared. Then the mean is taken and from this mean value the root is taken.

$$RMSE = \sqrt{\frac{\sum_{l \in L_m} (s_l(D) - \tilde{s}_l)^2}{|L_m|}} \tag{35}$$

The first measure of the performance is important to see whether the current OD matrix gives values for flow and speed that are close to the measured values. To assure that this is satisfying enough, the new convergence criterion is used as stated in section 3.6. So option b) is used for the further research, but also the other options can be chosen in further research.

Further the computation time is not a good option to use for the comparison of two performances, since it depends a lot on the used computer and the time required for the assignment with STAQ. For these uncertainties, it is better to compare the performances based on the number of assignments required. If these are smaller, then the computation time is also smaller in general. Thus, the performances are evaluated based on the number of assignments, with as convergence criterion that the absolute differences are everywhere below 5km/h.

5.3 Asymmetric Design

To further improve the performance of the algorithm, a different way to calculate the gradient approximation with SPSA is investigated. The method of asymmetric design (SPSA-AD) uses only half the number of objective function evaluations per iteration to find the gradient approximation compared to the normal algorithm. The calculation of the gradient approximation is only based on a perturbation in one direction. This calculation is given in equation 36.

$$\tilde{g}_k^l = \frac{z(\theta_k + c_k\Delta_k) - z(\theta_k)}{c_k\Delta_k^l\|\tilde{g}_k^l\|} \tag{36}$$

This equation gives the gradient approximation $\tilde{g}_k^l$ for iteration k and gradient approximation l.

In Figure 21 a comparison between the normal algorithm and the application of the asymmetric design is shown for the average of three different runs for the same matrix.

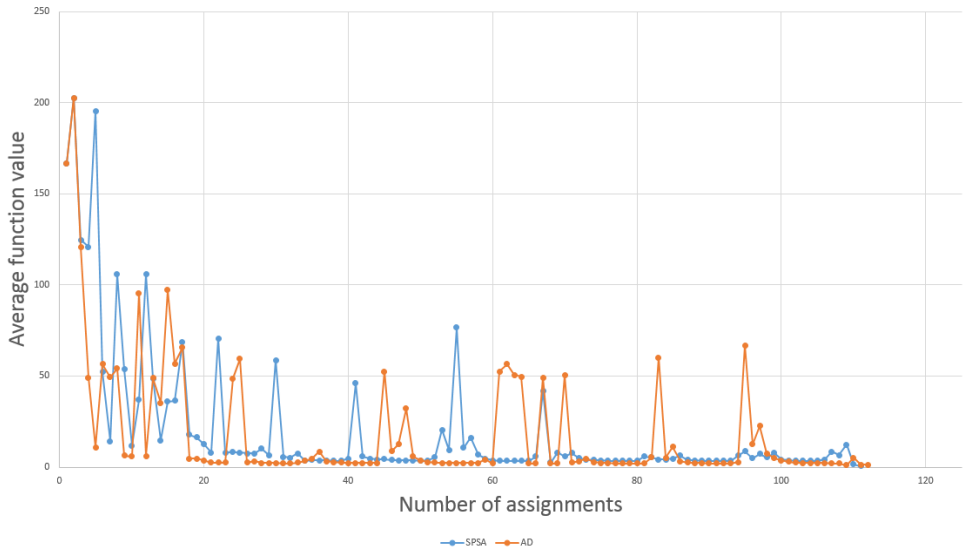


Figure 21: Number of assignments vs. average function value of three runs for both SPSA (blue) and SPSA AD (orange)

The start matrix resulted from a perturbation with a factor from the interval $[0.8, 1.2]$ and the measurements are the same as in section 5.4. It can be seen that on average the asymmetric design requires less assignments than the normal algorithm to reach the same precision. However, there are also situations where the normal algorithm performs better than the asymmetric design, which can be seen in figure 22. Here a different start matrix is used which resulted in a different quality of performance.

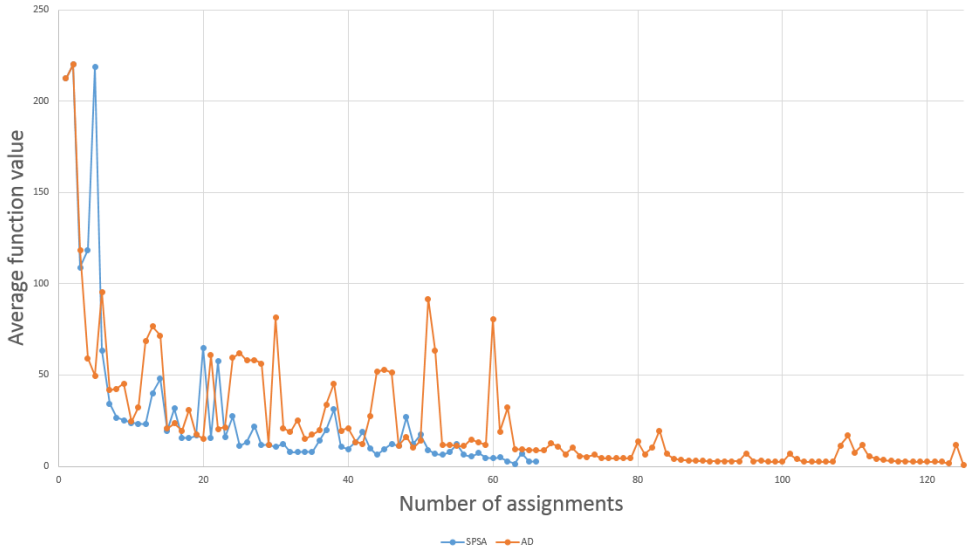


Figure 22: Number of assignments vs. average function value for both SPSA (blue) and SPSA AD (orange)

Although sometimes the application of the asymmetric design requires more assignments than in the symmetric case, in most situations the asymmetric design is able to save some computation time, which is the reason it is applied from now on. Nevertheless, it can be a better choice in some situations to use the normal SPSA algorithm instead of SPSA-AD.

5.4 Sensitivity analysis of the parameters of SPSA

For large scaled networks the assignment with STAQ needs a lot of computation time, so a low number of assignments and hence a fast convergence of the algorithm is required. A good choice of the parameters helps SPSA to converge faster, which has been found by Spall(1998,1992) [5],[2], Frederix(2010) [6] and Cipriani(2011,2013) [4],[3]. So some further exploration of the parameters might be useful to improve the performance of the method.

In this section some different parameters of the SPSA algorithm are discussed and investigated in more detail. We restrict the research in this thesis to the number of gradient approximations m, the choice of C for c_k, the values of γ to calculate c_k and the size of the maximum step for the line search, maxstep.

To test the different parameters, a test environment is made. This is based on some measurements in the network, which are evenly spreaded along the highway. These measurements can be seen in figure 23.

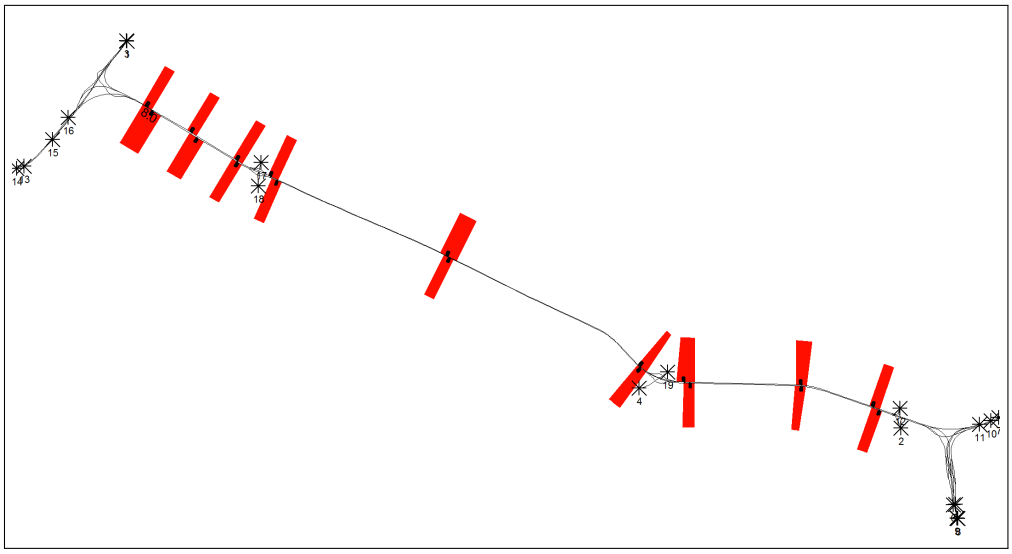


Figure 23: Figure of the measurement points for the analysis of the parameters

Then the original matrix is multiplied by random factors from the interval $[0.8, 1.2]$. During the tests, two different start matrices are used. These produce initial objective function values of 212.446 and 166.7823 respectively.

These matrices form together with the measured values $\tilde{v}$ and $\tilde{s}$ on the shown measurement locations the test environment for the parameters. For both matrices two runs with different values for the vector Δ are performed to obtain more certainty about the outcomes. The resulting four possible combinations are named situation 1 - 4. Here situations 1 and 2 are the two runs with start matrix 1 and situations 3 and 4 are the runs with start matrix 2. The four situations are shown in the tables by sit 1 - sit 4.

As a reference, the following parameters are chosen:

$$C = 10, \gamma = 0.17, \text{maxstep} = 30 \text{ and } m = 2$$

The reference case gives for the chosen four situations the graphs given in figure 24. After this number of assignments the algorithm finds a satisfying solution, which means that at all measurement points the differences between the modelled and observed speed are smaller than 5km/h.

To limit the amount of required test runs, the parameters are investigated one by one. This means that when investigating the first parameter, the others are kept as in the reference case. Once one or several optimal values for this parameter are found, this parameter value is chosen this way and the next parameter is changed until in the end an optimal set of parameter values for this situation is found. In order to do this, we first need to test the sensitivity of the parameters

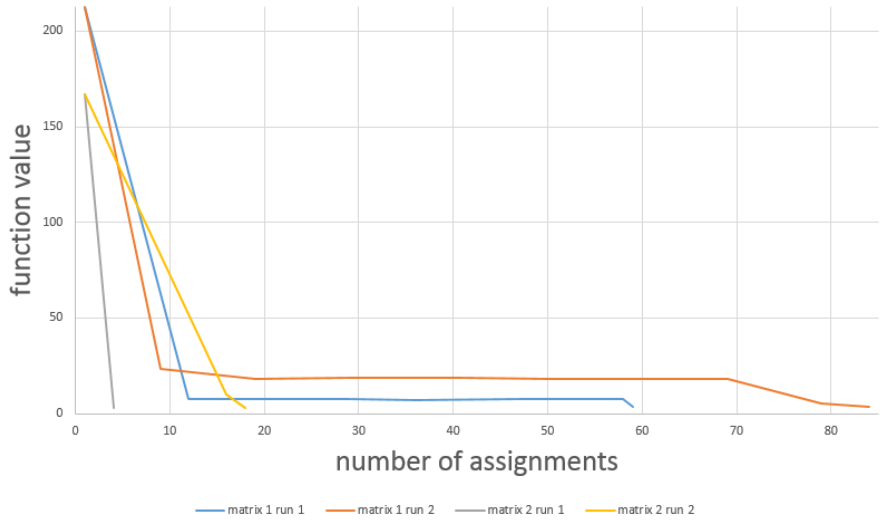


Figure 24: Graph of number of assignments vs. objective function values for all four situations with the reference parameters

with respect to the reference case. For every parameter specific values are investigated which are explained in the corresponding section. Table 4 gives the results of the tests where only one parameter is changed and the others are kept the same. In the column named difference the absolute difference of the average number of assignments is compared to the average number of assignments of the reference case.

m	C	γ	maxstep	sit 1	sit 2	sit 3	sit 4	avg	difference	avg diff
2	10	0.17	30	59	84	4	18	41.25	reference	
1	10	0.17	30	25	4	7	84	30	11.25	19.75
4	10	0.17	30	36	82	6	159	70.75	29.5	
9	10	0.17	30	23	46	11	101	45.25	4	
18	10	0.17	30	36	177	20	69	75.5	34.25	
2	1	0.17	30	141	15	15	68	59.75	18.5	42.25
2	5	0.17	30	66	15	8	12	25.25	16	
2	20	0.17	30	73	100	300	61	133.5	92.25	
2	10	0.101	30	62	94	4	18	44.5	3.25	6.75
2	10	1	30	54	66	4	20	36	5.25	
2	10	0.5	30	31	65	4	18	29.5	11.75	
2	10	0.17	50	69	110	6	300	121.25	80	34.25
2	10	0.17	100	95	99	8	39	60.25	19	
2	10	0.17	10	37	55	24	34	37.5	3.75	

Table 4: Tests of the sensitivity of the parameters with respect to the reference case

From this table we see that the highest sensitivity is for the values of the parameter C, so this is the first investigated parameter. It is followed by maxstep and m and in the end γ is investigated, which has the lowest sensitivity.

5.4.1 The choice of C

In this section the influence of the value of C on the performance of the algorithm is investigated. This value plays a crucial role in the algorithm since it defines in which area around the current matrix the function is evaluated to find a new gradient. C is used for the calculation of the search size c_k in the perturbation. This is done by equation 37.

$$c_k = \frac{C}{k^\gamma} \tag{37}$$

c_k is used in the calculation of the gradient approximation $\hat{g}_k$. This calculation is given in equation 38.

$$\tilde{g}_k^l = \frac{z(\theta + c_k\Delta_k) - z(\theta)}{c_k\Delta_k^l \|\tilde{g}_k^l\|} \tag{38}$$

Looking at the definition of the gradient, given in equation 39, equation 38 is quite similar to this definition.

$$\nabla z = \lim_{h\to 0} \frac{z(\theta + h) - z(\theta)}{h} \tag{39}$$

Choosing $h = c_k\Delta_k$ and neglecting the normalization in equation 38, the two equations are almost the same. Whenever $c_k\Delta_k$ is chosen small, they are close to each other.

So from theory it would be a good choice to make C as small as possible such that c_k is small as well. However, C can not be very small since then the speed does not change at all and the algorithm gets stuck.

The different values of C that are considered are $C = 10$ as reference, $C = 5, C = 1$ and $C = 20$. The other parameters are the same as in the reference case, hence maxstep $= 30, m = 2$ and $\gamma = 0.17$. This gives table 5. The results can also be seen in figures 25 and 26.

C	sit 1	sit 2	sit 3	sit 4	avg
10	59	84	4	18	41.25
20	73	100	300	61	133.5
1	141	15	15	68	59.75
5	66	15	8	12	25.25

Table 5: Tests for C

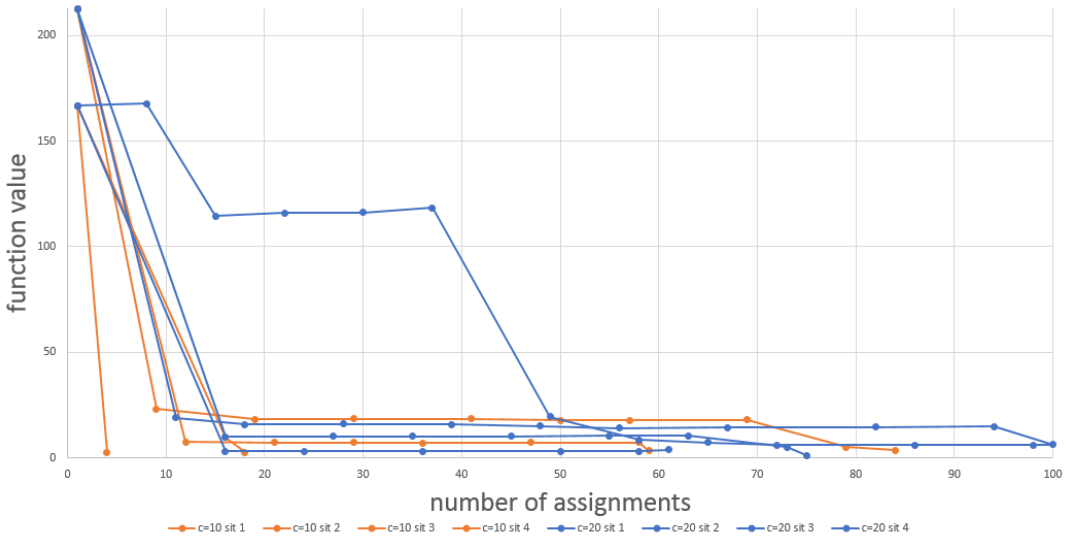


Figure 25: Tests of C with $C = 10$ and $C = 20$

In these examples, the theoretical idea can be verified. Here the choice of $C = 5$ gives the best results followed by $C = 10$. $C = 1$ gives worse results than $C = 10$. This shows that a too small value for C is also not a good choice. $C = 20$ gives the worst situation. Here C is chosen too large for a good convergence. For this reason, only $C = 5$ and $C = 10$ are further considered.

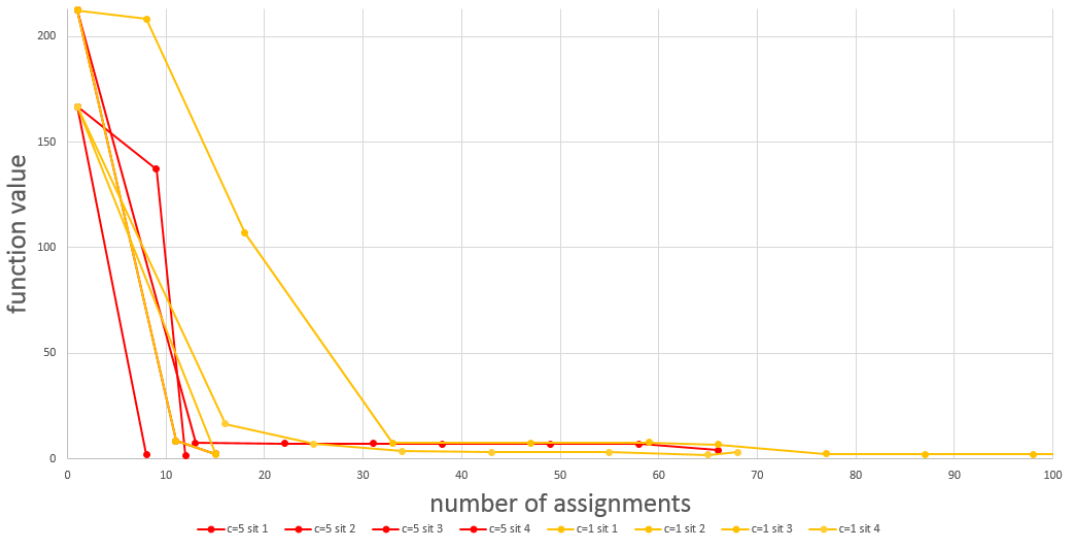


Figure 26: Tests of C with $C = 1$ and $C = 5$

5.4.2 The value maxstep

In this section the following values of maxstep are tested in combination with $C = 10$ and $C = 5$: maxstep = 30, maxstep = 50, maxstep = 100 and maxstep = 10.

All these situations are now compared in table 6.

C	maxstep	sit 1	sit 2	sit 3	sit 4	avg
10	30	59	84	4	18	41.25
10	50	69	110	6	>300	121.25
10	100	95	99	8	39	60.25
10	10	37	55	24	34	37.5

C	maxstep	sit 1	sit 2	sit 3	sit 4	avg
5	30	66	15	8	12	25.25
5	50	89	9	10	170	69.5
5	100	96	9	13	63	45.25
5	10	44	46	8	35	33.25

Table 6: Tests for maxstep

The choices of maxstep = 30 and maxstep = 10 give for both $C = 10$ and $C = 5$ the best performance with convergence in all four situations, within less than 100 assignments for all of them. Hence the following four combinations are used for the remaining two parameter tests:

- $C = 10$ and maxstep = 30
- $C = 10$ and maxstep = 10
- $C = 5$ and maxstep = 30
- $C = 5$ and maxstep = 10

5.4.3 Number of gradient approximations m

The number of gradient approximations m determines how many computations of the gradient approximation are used to average the final gradient, that is used for the step to the next solution.

$$\hat{g}_k = \frac{1}{m} \sum_{1}^{m} \hat{g}_{k_l} \tag{40}$$

A high value decreases the chance of taking the step in an undesired direction. However, at the same time, the number of assignments increases since for every gradient approximation an assignment needs to be performed. So the choice of the number of approximations needs to be taken with care. An analysis is performed to investigate whether a low value of this parameter is enough for a good performance or whether a higher value improves the performance.

The investigated values for the number of gradient approximations are chosen based on the number of OD pairs to be optimized. For the CHARM network there are 89 non-zero OD pairs. From this amount 5%, 10% and 20% are chosen. These percentages are also used in the paper of Cipriani (2011) [4]. This results in $89 * 0.05 = 4.45 \approx 4$, $89 * 0.1 = 8.9 \approx 9$ and $89 * 0.2 = 17.8 \approx 18$ gradient approximations per iteration. For comparison, also only 1 approximation and 2 approximations per iteration are done.

The results of these situations, with $\gamma = 0.17, c = 10$ and maxstep = 30 as in the reference case, are shown in table 7.

m	sit 1	sit 2	sit 3	sit 4	avg
1	25	4	7	84	30
2	59	84	4	18	41.25
4	36	82	6	159	70.75
9	23	46	11	101	45.25
18	36	177	20	69	75.5

Tests for $C = 10$ and maxstep $= 30$

m	sit 1	sit 2	sit 3	sit 4	avg
1	25	4	7	60	24
2	66	15	8	12	25.25
4	66	108	10	127	77.75
9	308	246	15	69	159.5
18	194	300	24	300	204.5

Tests for $C = 5$ and maxstep $= 30$

m	sit 1	sit 2	sit 3	sit 4	avg
1	51	101	35	9	49
2	37	55	24	34	37.5
4	37	300	33	55	106.25
9	115	289	28	20	113
18	291	120	38	43	123

Tests for $C = 10$ and maxstep $= 10$

m	sit 1	sit 2	sit 3	sit 4	avg
1	58	101	33	22	53.5
2	44	46	8	35	33.25
4	37	181	10	167	98.75
9	288	95	15	20	104.5
18	93	117	24	32	66.5

Tests for $C = 5$ and maxstep $= 10$

Table 7: Tests for the different situations

It can be seen that for $m = 18, m = 9$ and $m = 4$ the number of required assignments until convergence is higher than in the cases where $m = 1$ and $m = 2$. Also it can be seen that in almost all situations the choice of maxstep $= 30$ is better than the choice of maxstep $= 10$. For this reason we further choose to use the following sets of parameters to investigate the influence of the value of γ:

- $C = 5$, maxstep $= 30$ and $m = 1$
- $C = 5$, maxstep $= 30$ and $m = 2$
- $C = 10$, maxstep $= 30$ and $m = 1$
- $C = 10$, maxstep $= 30$ and $m = 2$

5.4.4 Gamma

The literature gives several conditions for the values of α and γ and also suggests different values. Spall gives in his work (1997) [10] the following condition for these parameters:

$$\alpha - 2\gamma > 3\gamma - \alpha/2 \geq 0 \tag{41}$$

Most literature uses the values $\alpha = 0.602$ and $\gamma = 0.101$ or $\alpha = 1$ and $\gamma = 1/6 \approx 0.17$.

Since line search is used instead of the gain sequence a_k, the value of α is not important for this situation. Next to the often used values $\gamma = 0.17$ and $\gamma = 0.101$, also some tests are performed with higher values for γ. The considered values for γ are 1 and 0.5.

Now the different values for γ are tested and the required assignments for convergence are shown in table 8.

m	γ	sit 1	sit 2	sit 3	sit 4	avg
2	0.17	59	84	4	18	41.25
1	0.17	25	4	7	84	30
2	0.101	62	94	4	18	44.5
1	0.101	27	4	7	87	31.25
2	0.5	31	65	4	18	29.5
1	0.5	18	4	7	300	82.25
2	1	54	66	4	20	36
1	1	62	4	7	52	31.25

Tests for $C = 10$

m	γ	sit 1	sit 2	sit 3	sit 4	avg
2	0.17	66	15	8	12	25.25
1	0.17	25	4	7	60	24
2	0.101	76	14	8	11	27.25
1	0.101	27	4	7	63	25.25
2	0.5	33	14	8	11	16.5
1	0.5	27	4	7	248	71.5
2	1	32	14	8	11	16.25
1	1	62	4	7	54	31.75

Tests for $C = 5$

Table 8: Tests for $C = 10$ and $C = 5$

Here it can be seen that the tests with $C = 10$ are most times worse than the tests for $C = 5$. To get more secure about the situations at hand, another run for every matrix is performed, denoted by situations 3 and 6, resulting in table 9.

m	γ	sit 1	sit 2	sit 3	sit 4	sit 5	sit 6	avg
2	0.17	66	15	80	8	12	122	50.5
1	0.17	25	4	44	7	60	51	31.83
2	0.101	76	14	82	8	11	124	52.5
1	0.101	27	4	37	7	63	118	42.67
2	0.5	33	14	80	8	11	102	41.33
1	0.5	27	4	64	7	300	45	74.5
2	1	32	14	181	8	11	81	54.5
1	1	62	4	300	7	54	83	85

Table 9: Tests for $C = 5$

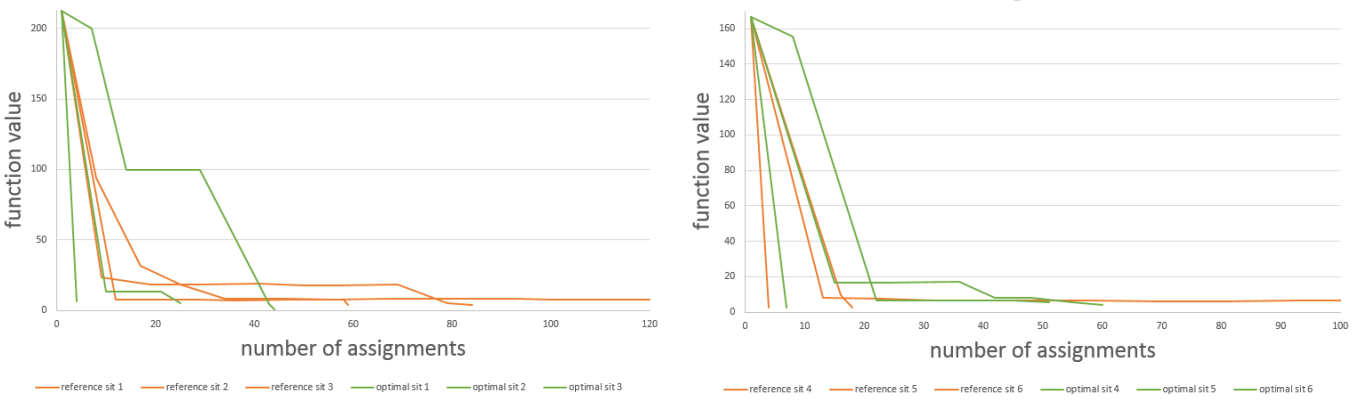
The best combination is chosen, resulting in the following set of parameters:

$$C = 5, \text{maxstep} = 30, m = 1 \text{ and } \gamma = 0.17$$

The chosen value for γ is thus 0.17, which is one of the two most used values for this parameter. So this should be a good choice.

5.4.5 The optimal found set of parameters

The optimal found set of parameters is given by $C = 5, \text{maxstep} = 30, m = 1$ and $\gamma = 0.17$. Figure 27 gives a comparison between the performance of the optimal set of parameters and the reference case.



(a) Start matrix 1

(b) Start matrix 2

Figure 27: Comparison between the reference set of parameters (orange) and the optimal found set (green)

We choose to use this optimal set of parameters for the tests in the next section. However, it might be better to use another set of parameters for other networks and other start matrices since the algorithm is unreliable concerning the number of assignments. Moreover, since we only investigated the parameters one by one and not simultaneous, there might be a combination of parameters that works a lot better than the found optimal set. Nevertheless, as can be seen, it is possible to do some tests to find a set of parameters that performs relatively good and some more research can make the performance of the algorithm even better.

5.5 Sensitivity analysis of the input data for SPSA

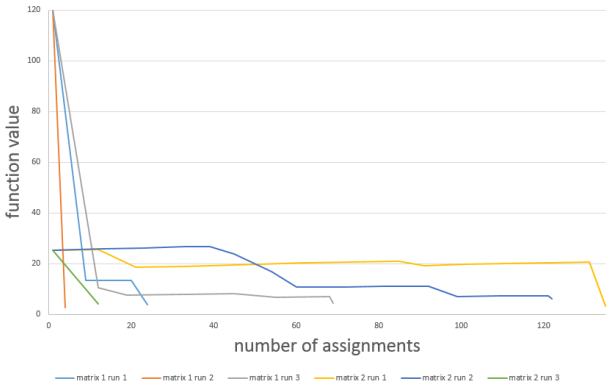
To test the algorithm and the set of parameters further, other different situations for the CHARM network are investigated.

To start with the general tests, a small number of measurements is chosen and then this number is increased stepwise. There is a set of 68 real measurement locations available. From this set a certain part is used for the tests. To start with, only 20% of the measurements are used. The resulting 14 measurements are chosen randomly from the set. Next more measurement points are added to the network to see how the algorithm can handle a larger number of measurements. The number of measurement points is increased from 20% until 100%. For the tests two different combinations of measurement points are tested for every value of percentage.

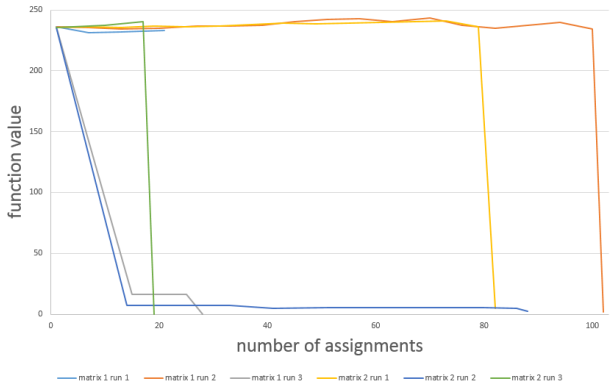
The original matrix is perturbed by random factors within the interval $[0.8, 1.2]$. This is done two times, resulting in two different start matrices, named matrix 1 and matrix 2. These matrices are given in figure 35. Then the algorithm is run three times for each matrix. Figures 28, 29 and

30 give the resulting graphs of the objective function value versus the number of assignments with STAQ. Here the maximum number of assignments is chosen to be 300, so if the graphs reach more than 300, the algorithm did not converge within this number of assignments.

20 percent

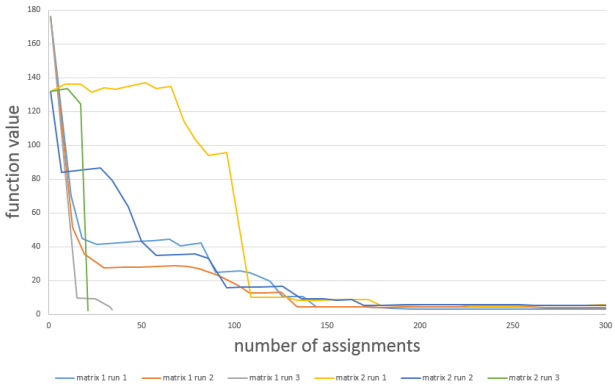


(a) Situation 1

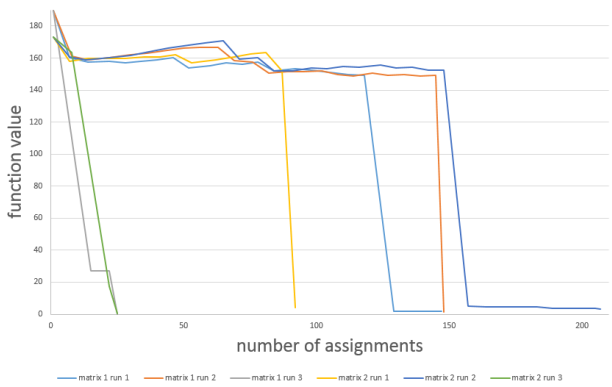


(b) Situation 2

40 percent



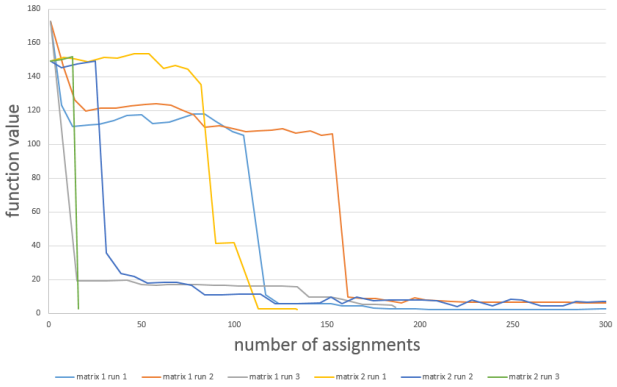
(c) Situation 1



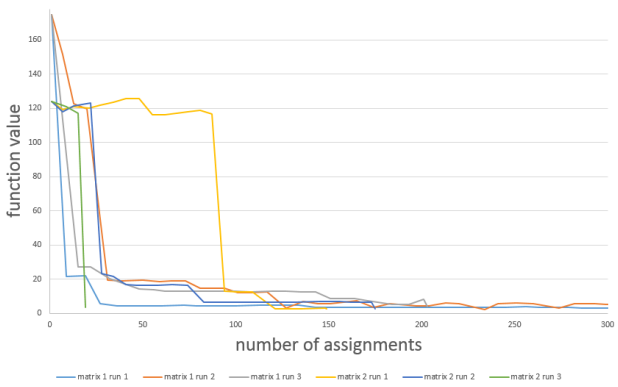
(d) Situation 2

Figure 28: Plots of function value versus number of assignments for the different situations with 20 and 40 percent of the measurement points

60 percent

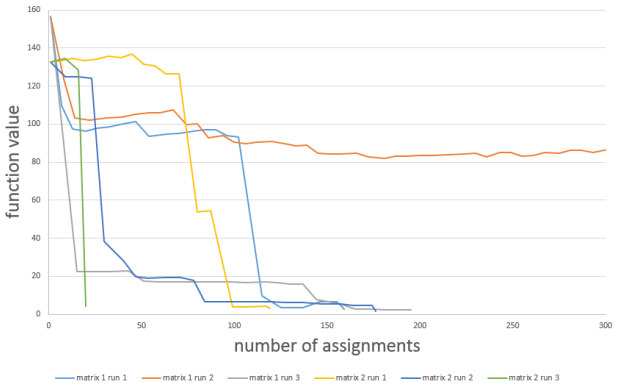


(a) Situation 1

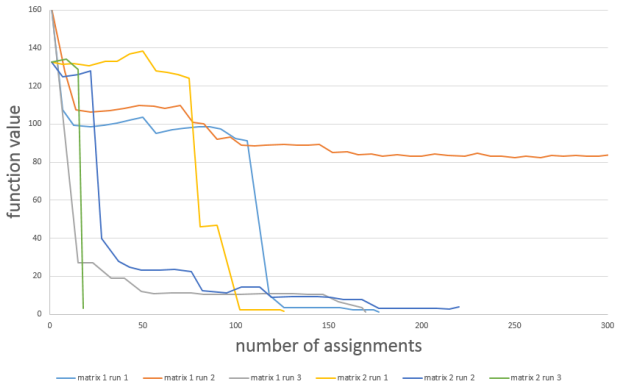


(b) Situation 2

80 percent



(c) Situation 1



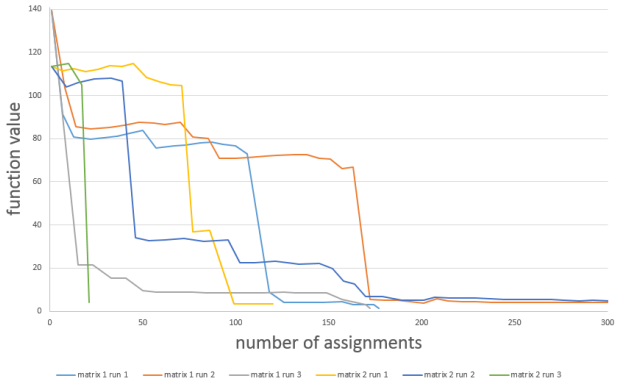
(d) Situation 2

Figure 29: Plots of function value versus number of assignments for the different situations with 60 and 80 percent

It can be seen that there are mostly high differences in the number of assignments for the three different runs. Also there are high differences in the performances for different percentages of the measurements. Here the random characteristic of SPSA gets visible. This implies that beforehand it is almost unknown how good the algorithm performs.

However, in more than 74% of the situations investigated, the algorithm was able to find a solution that fulfills the convergence criteria in less than 300 assignments of STAQ. Most of the remaining 26% might converge a little later, due to the small objective function value after 300 assignments of STAQ, see e.g. figure 28c. Only for the case of matrix 2 and run 2 with 80 percent of the measurements, it looks like that the algorithm has problems to converge, see figures 29c and 29d

all measurement points



(a) all measurements

Figure 30: Plot of function value versus number of assignments for all measurement points

5.6 Conclusion

In this section we tested the new method on a small real network, called the CHARM network. For this we first defined the used measures of performance, followed by a new modification of the method to make the convergence faster. Then tests for the parameters m, γ, C and maxstep are performed and the optimal set with $m = 1, \gamma = 0.17, C = 5$ and maxstep $= 30$ is used for some general tests of the situation. Here in more than 74% of the situations the algorithm is able to find a solution within 300 assignments of STAQ.

Note that the OD matrix calibration problem is mostly an underspecified problem, which means there is not only one unique solution, but several different OD matrices that reproduce the same values for speed (and flow). The problem is underspecified, because e.g. in this example there are 89 OD pairs that can be calibrated, but there are at most 68 measured values of the speed. So there are 21 more variables than constraints. However, the OD matrix calibration problem does not deal with finding one optimal matrix, but the aim is to find a matrix that reproduces the measured values as good as possible, which is already stated in section 3.2.

6 Conclusion and Recommendations

6.1 Conclusion

In this thesis the origin-destination matrix calibration problem is investigated for the application of the quasi-dynamic assignment model STAQ. The OD matrix calibration problem can be seen as a bi-level optimization problem that acts like a Stackelberg game. The follower, the lower level, reacts on the actions of the leader, the upper level. This requires the optimization to find the solution iteratively. For this purpose the stochastic optimization algorithm SPSA (Simultaneous Perturbation Stochastic Approximation) is used. This algorithm performs a gradient approximation in every iteration with only two objective function values. With this gradient the next solution is found iteratively.

During the literature review some promising modifications of the SPSA algorithm have been found, where some of them are applied during this thesis. The others are given in section 6.2. Also some problems that might come up with the SPSA algorithm are found in the literature.

After some background information about the four step model and the different parts of the OD matrix calibration problem, a main problem of the classic SPSA algorithm for the calibration is investigated, the problem of local minima. After some research, it becomes clear that especially the fact that the flow on a link is not bijective with respect to the OD demand causes all of the local minima that we found in the test situations. However, it is not clear if these include all the local minima present in the objective function. To prevent these local minima, the objective function is chosen such that it only includes the difference in speed on the links. The convergence criterion is now only dependent on the speed and convergence is reached if the absolute differences in speed of all measured links are small enough. Also line search is used instead of a sequence of step sizes in order to make sure that the step is only taken as far as the function value decreases. For this purpose the golden section search is applied.

Further bounds to prevent the algorithm from infeasible solutions are investigated. Since including the initial matrix in the objective function might cause local minima as well, a penalty is used whenever the beforehand chosen bounds are crossed. In general it can be said that it is advisable to keep the objective function as simple as possible to avoid local minima. Using only the speed differences makes the objective function bijective. Together with the assumption that all measurement combinations of flow and speed lie on the objective function, this gives that only minimizing the speed differences also minimizes the flow differences.

After the application of the modifications on some test networks, it becomes clear that the algorithm performs quite succesful for these networks. So in the next section the modified algorithm is used for a larger real network, the CHARM network.

Some more tests show that the use of the asymmetric design, where only one objective function value is required for every gradient approximation, needs less assignments to converge. Next several tests for the parameters are done to find a relatively good set of parameters. This set is then used in a concluding test to see how good the performance of the algorithm is. During these tests it is found that in more than 81% of the situations the algorithm is able to converge within 300 assignments of STAQ. For the other situations more assignments are required.

The modifications make the SPSA algorithm for the OD matrix calibration a lot faster and more reliable. However, in general it is not possible to give a guarantee for the converge or to know how many assignments with STAQ are required beforehand. A lot depends on the situation at hand, the initial matrix and the randomness of the Δ parameter of the algorithm. In most

cases where the algorithm has problems to converge or needs a lot of computations, it helps to run the algorithm again from the start, since then the random values are chosen different and the algorithm might be able to perform better. Despite this, the algorithm is capable of most situations and additional modifications might improve its performance further.

6.2 Recommendations

Further research can be done to make the algorithm more reliable and faster. During this thesis a lot of choices needed to be made, for example the form of the objective function, the convergence criteria or the choice of parameters. All of these can be chosen different and can cause the algorithm to perform better.

Although the SPSA algorithm gives a good performance for the tested situations, other optimization algorithms can be used as well and might perform better and especially more reliable. However, there are more possible modifications of the SPSA algorithm found in the literature. Some of them are given now. Since this thesis only dealed with the standard version and the enhanced method to avoid local minima, these modifications need to be tested in further research. Some possible modifications are the following:

- **Polynomial Interpolation**. Cipriani (2011,2013) named this modification together with the use of the asymmetric design. The polynomial interpolation can replace the golden section search. Here a third degree polynomial is calculated which fits calculated function value. Then the minimum of this polynomial is found and this gives the next iterate of the algorithm.
- **Second order SPSA** (see Cipriani et al. (2013)). Here an estimation of the Hessian matrix H of $z(D)$ is required. This approximated matrix is then used together with the gradient approximation to find the new solution.
- **c-SPSA and w-SPSA** (see Lu et al.(2015) and Tympakianaki et al.(2015)). These modifications use further information about the correlations between the OD pairs to improve the performance of the algorithm.

There are a lot possibilities for further research to improve the reliability and the performance of the SPSA algorithm in order to solve the OD-matrix calibration problem more efficient.

References

[1] Michiel C. J. Bliemer, Mark P. H. Raadsen, Luuk J. N. Brederode, Michael G. H. Bell, Luc J. J. Wismans, Mike J. Smith, *Genetics of traffic assignment models for strategic transport planning*, Transport Reviews, pages 1-23, DOI: 10.1080/01441647.2016.1207211.

[2] James C. Spall, *An Overview of the Simultaneous Perturbation Method for Efficient Optimization*, Johna Hopkins APL Technical Digest, Volume 19, Number 4, 1998.

[3] Ernesto Cipriani, Andrea Gemma, Marialisa Nigro, *A bi-level gradient approximation method for dynamic traffic demand estimation: sensitivity analysis and adaptive approach*, 16th International IEEE Annual Conference on Intelligent Transportation Systems, 2013, DOI: 10.1109/ITSC.2013.6728539.

[4] Ernesto Cipriani, Michael Florian, Michael Mahut, Marialisa Nigro, *A gradient approximation approach for adjusting temporal origin-destination matrices*, Transportation Research Part C, Elsevier, 2013, DOI: 10.1016/j.trc.2010.05.013.

[5] James C. Spall, *Multivariate Stochastic Approximation Using a Simultaneous Perturbation Gradient Approximation*, IEEE Transactions on automatic control, Volume 17, 1992, DOI: 10.1109/9.119632.

[6] Rodric Frederix, *Dynamic origin-destination matrix estimation in large-scale congested networks*, PhD Thesis KU Leuven, 2012.

[7] David W. Hutchinson, *On an Efficient Distribution of Perturbations for Simulation Optimization using Simultaneous Perturbation Stochastic Approximation*, Proceedingsof the IASTED International Conference APPLIED MODELLINF AND SIMULATION, Cambride, MA, USA, pages 440-445, November 2002.

[8] L.Lu, Y. Xu, C. Antoniour, M. Ben-Akiva, *An enhanced SPSA algorithm for the calibration of Dynamic Traffic Assignment models*, Transportation Research Part C: Emerging Technologies, Volume 51, February 2015, pages 149-166, DOI: 10.1016/j.trc.2014.11.006.

[9] A. Tympakianaki, H.N. Koutsopoulos, E. Jenelius, *c-SPSA: Cluster-wise simultaneous perturbation stochastic approximation algorithm and its application to dynamic origin-destination matrix estimation*, Transportation Research Part C: Emerging Technologies, Volume 55, June 2015, pages 231-245, DOI: 10.1016/j.trc.2015.01.016.

[10] James C. Spall, *Implementation of the simultaneous perturbation algorithm for stochastic optimization*, IEEE Transactions on Aerospace and Electronic Systems, Volume 34, Issue 3, pages 817-823, July 1998, DOI: 10.1109/7.705889.

Terminology

In this section some terminology is explained in more detail.

- **Demand**. During this thesis the term traffic demand, OD demand or just demand gives the amount of traffic that wants to travel from an origin to a destination. For every pair of origin and destination, the demand gives the number of vehicles that want to make this trip.
- **Flow**. The flow or flow rate gives the amount of traffic that actually passes a specific point in some defined time interval. This includes that the flow can be at most equal to the total demand on that link, but if congestion occurs, the flow might be lower than the demand. The flow is measured in vehicles per hour.
- **Speed**. The value of speed gives the average speed of all vehicles passing that link. Every link has a free flow speed, which is the possible speed if no congestion occurs, and a speed at capacity, which gives the average speed whenever the capacity is reached. From the speed it is possible to see whether a link is congested or not. If the average speed is lower than the speed at capacity, the link is congested, if it lies above the speed at capacity, then the link is in free flow.
- **Bottleneck**. A link is a potential bottleneck when its capacity is lower than the sum of the capacities of the inlinks to that link. The inlinks are all links that are directed to and connected with this link. A potential bottleneck will only become an active bottleneck when the sum of the outflows of the inlinks, so the amount of flow that arrives at that link, is greater than the capacity of this link.

List of acronyms

SPSA Simultaneous Perturbation Stochastic Approximation

STAQ Static Traffic Assignment with Queueing

OD Origin-destination

Appendix

A SPSA in detail

In this section we give a summary of the SPSA- (Simultaneous Perturbation Stochastic Approximation) method for solving the unconstrained minimization problem

$$\min_{x \in \mathbb{R}^p} f(x) \tag{42}$$

where $f \in C^1$ on $\mathbb{R}^p$. Denote $\nabla f(x)$ by $g(x)$. The well-known steepest descent method proceeds as follows:

Starting at x_0 the method generates iterates

$$x_{k+1} = x_k - a_k g(x_k) \qquad\qquad k = 0, 1, ... \tag{43}$$

where the stepsizes a_k are computed by solving line minimization

$$a_k = \arg\min_{a>0} f(x_k - a g(x_k)) \tag{44}$$

either "exactly" or by some "inexact version". Often we are faced with a situation where the gradient g or even the function f is not explicitly available. Thus the value of g has to be approximated numerically.

From numerical analysis we consider the following formula for approximating the derivative $h'(t)$ of a C^2 or C^3 function $h : \mathbb{R} \to \mathbb{R}$ at a point t based on stepsize $c > 0$:

$$\frac{h(t+c) - h(t-c)}{2c} = h'(t) + O(c^2) \tag{45}$$

This leads to the following approximation $\tilde{g}$ of gradient g.

$$\tilde{g}(\bar{x}) \text{ given by } \tilde{g}_i(\bar{x}) = \frac{f(\bar{x} + ce_i) - f(\bar{x} - ce_i)}{2c} \tag{46}$$

Here $g_i(\bar{x})$ denotes the i^{th} component of $g(\bar{x})$, i.e. $g_i(\bar{x}) = \frac{\partial}{\partial x_i} g(\bar{x})$.

For C^2 or C^3 function $f : \mathbb{R} \to \mathbb{R}$ formula 45 directly gives the error bounds (for $c \to 0$)

$$\tilde{g}(\bar{x}) = g(\bar{x}) + O(c^2) \tag{47}$$

According to equation 46 the computation of $\tilde{g}$ requires (assuming $f(\bar{x})$ is known) $2p$ function evaluations of f. In cases where p is large and function evaluations of f are expensive, this approximation might be too expensive.

Here now comes the idea of SPSA into play. It defines a type of probabilistic approximation for $g(\bar{x})$ which (independent of p) only requires two function evaluations as follows:

Given a direction $\Delta \in \mathbb{R}^p$ and a stepsize c, the corresponding vector $\tilde{g}(\bar{x})$ is defined componentwise as

$$\tilde{g}_i(\bar{x}) = \frac{f(\bar{x} + c\Delta) - f(\bar{x} - c\Delta)}{2c\Delta_i} \tag{48}$$

Clearly we have to assume $\Delta_i \neq 0$, otherwise $\tilde{g}$ is not defined. Note that actually $\tilde{g}(\bar{x}) = \tilde{g}(\bar{x}, \Delta)$.

Clearly, we can't expect $\tilde{g}(\bar{x})$ to be a good approximation of $g(\bar{x})$. But, as we will see, if we take (many) different directions $\Delta^1, ..., \Delta^k$ (randomly), compute the corresponding vector $\tilde{g}^j(\bar{x}) = \tilde{g}(\bar{x}, \Delta^j)$ and take the mean

$$\hat{g}(\bar{x}) = \frac{1}{k} \sum_j \tilde{g}^j(\bar{x}) \tag{49}$$

then (under some assumptions given later) we can expect $\tilde{g}(\bar{x})$ to be a reasonable approximation for $g(\bar{x})$.

We now introduce the SPSA approximation in more detail. For easyness we describe a version, where no noise (error in evaluation of $f(x)$) is present.

Let $\mathbb{D}$ be a given set of vectors Δ

$$\mathbb{D} = \{\Delta \in \mathbb{R}^p \quad | \quad \alpha_0 \leq |\Delta_i| \leq \alpha_1\} \tag{50}$$

for some numbers $0 < \alpha_0 \leq \alpha_1$.

We could for example choose $\mathbb{D} = \{\Delta \in \mathbb{R}^p \quad | \quad \Delta_i \in \{-1, 1\} \quad \forall i\}$.

Let $\Delta \in \mathbb{D}$ be randomly generated such that the expected value $E[\Delta] = 0$. Then $\tilde{g}(\bar{x})$ becomes a random variable. Then the following holds:

L_1. Let f be three times continuously differentiable in a neighbourhood of $\bar{x}$. Let $\Delta \in \mathbb{D}$ be generated randomly as described above.

Then for the expected value of $\tilde{g}(\bar{x})$ we have for $c \to 0$:

$$E[\tilde{g}(\bar{x}) - g(\bar{x})] = O(c^2) \tag{51}$$

Proof see Spall (1992).

L_1 states that in the mean for c small $\tilde{g}(\bar{x})$ can be seen as a reasonable approximation of gradient $g(\bar{x})$.

The SPSA algorithm for solving equation 45 now is as follows: Starting with x_0 and fixed sequence of stepsizes $a_k, c_k, k \geq 0$ such that $a_k \to 0, c_k \to 0$ and

$$\sum a_k = \infty \qquad \sum \frac{a_k^2}{c_k^2} < \infty \tag{52}$$

we generate a sequence $\Delta \in \mathbb{D}$ randomly as described above.

Let Ω denote the corresponding space of sequences

$$\Omega = \{\omega = (\Delta^0, \Delta^1, ...)\} \tag{53}$$

We then compute the corresponding SPSA iterates: starting with x_0 compute

$$x_{k+1} = x_k - a_k \hat{g}(x_k, \Delta^k) \qquad k = 0, 1, ... \tag{54}$$

Let further x^* be a local minimum of f (in particular $g(x^*) = 0$).

Then under strong conditions on x_0 (see conditions A3-A5 in Spall (1992)) for the iteration given in equation 54 we have

$$x_k \to x^* \qquad \text{for almost all } \omega \in \Omega \tag{55}$$

B Figures

```
ak, ck given
For k=1:n
delta=2*round(rand(p,1))-1;
thetaplus=theta+ck*delta;
thetaminus=theta-ck*delta;
Lplus=loss(thetaplus);
Lminus=loss(thetaminus);
ghat=(Lplus-Lminus)./(2*ck*delta);
theta=theta-ak*ghat;
end
theta
```

Figure 31: A short implementation of SPSA

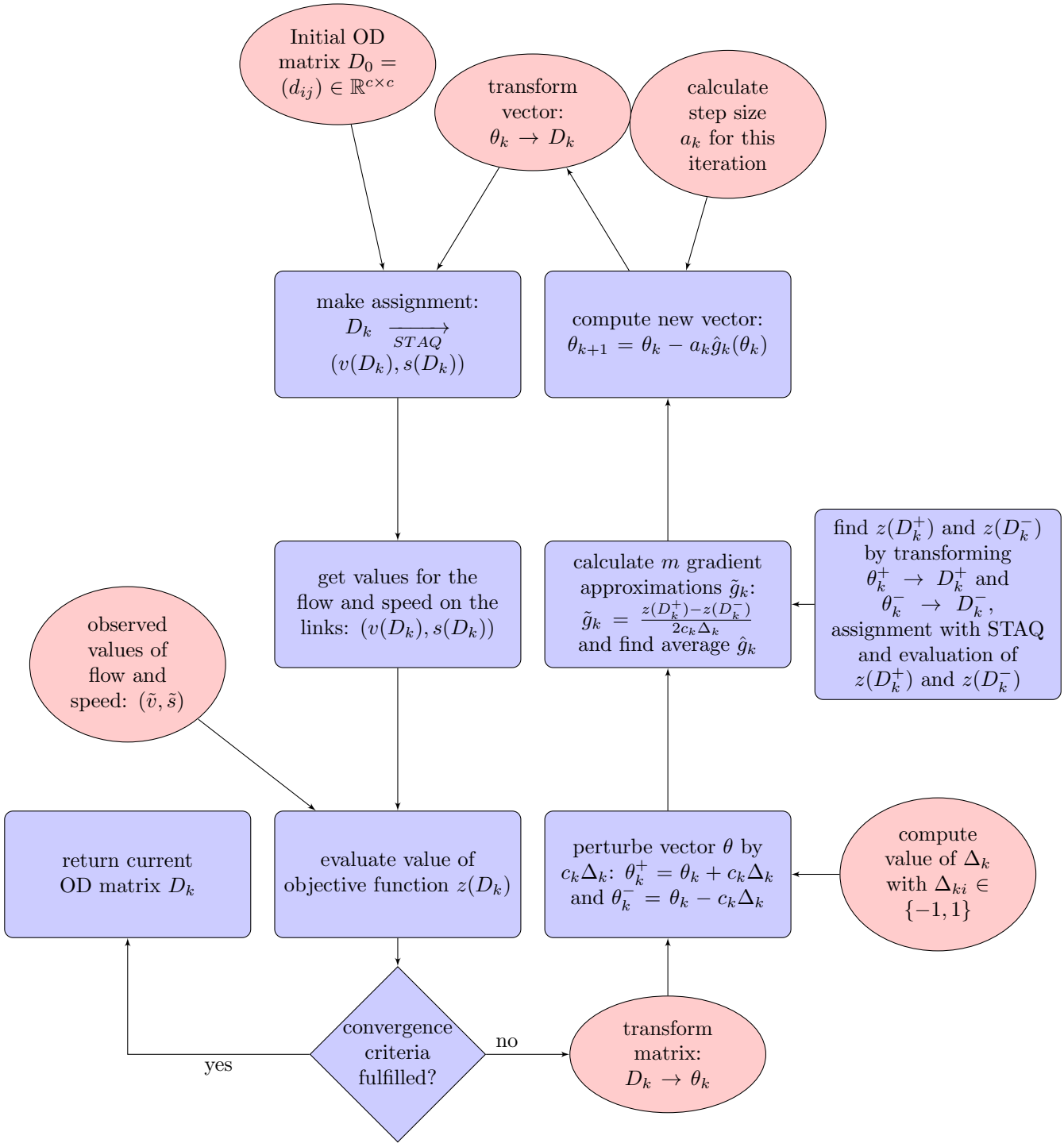
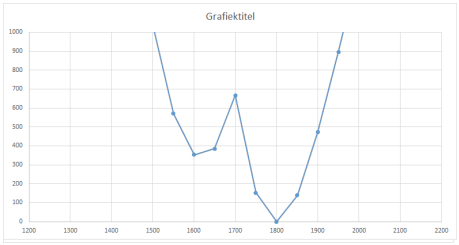
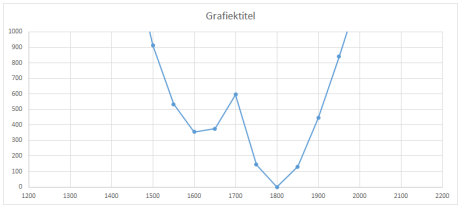


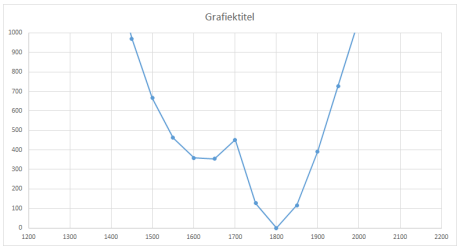
Figure 32: Scheme of the procedure of the OD matrix calibration



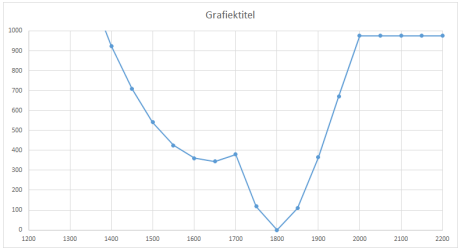
(a) Weight flow 0.05



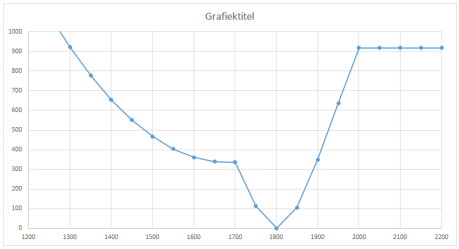
(b) Weight flow 0.04



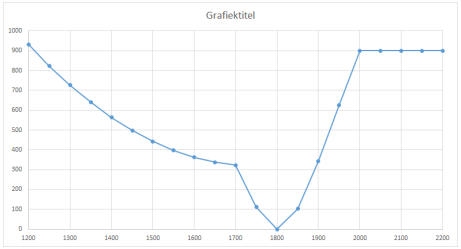
(c) Weight flow 0.02



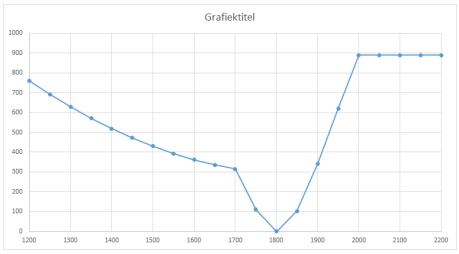
(d) Weight flow 0.01



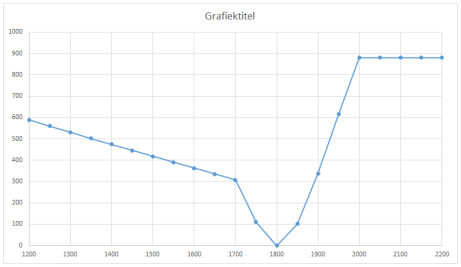
(e) Weight flow 0.004



(f) Weight flow 0.002

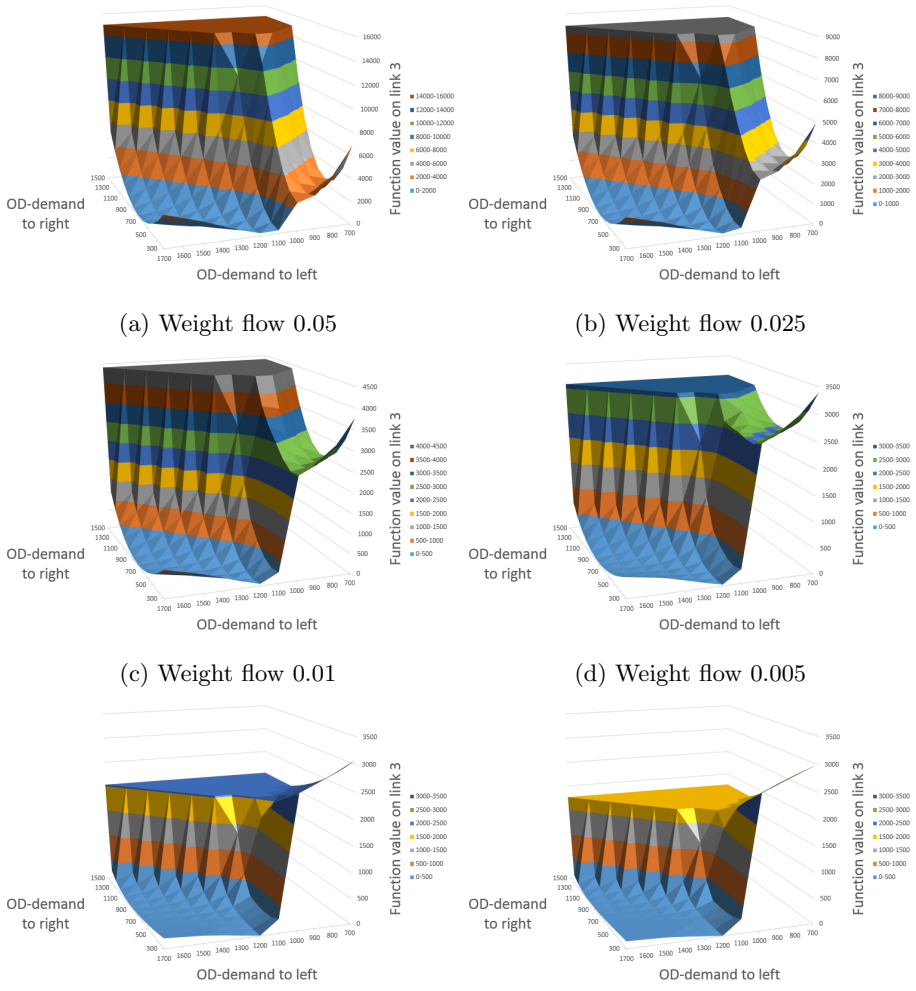


(g) Weight flow 0.001



(h) Weight flow 0

Figure 33: Function values for a measured demand of 1800 on link 2 of the matchstick network with different weights on flow



(a) Weight flow 0.05

(b) Weight flow 0.025

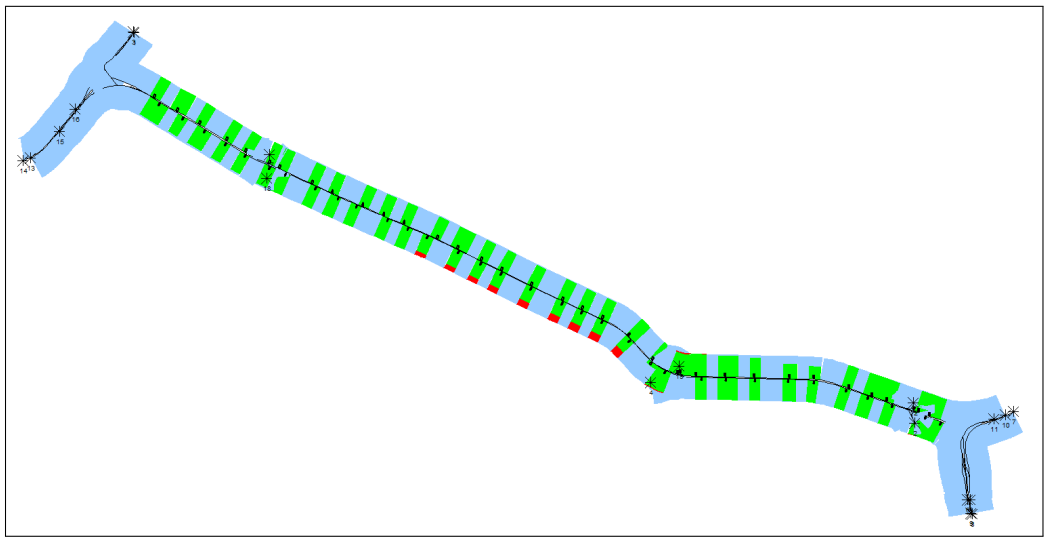
(c) Weight flow 0.01

(d) Weight flow 0.005

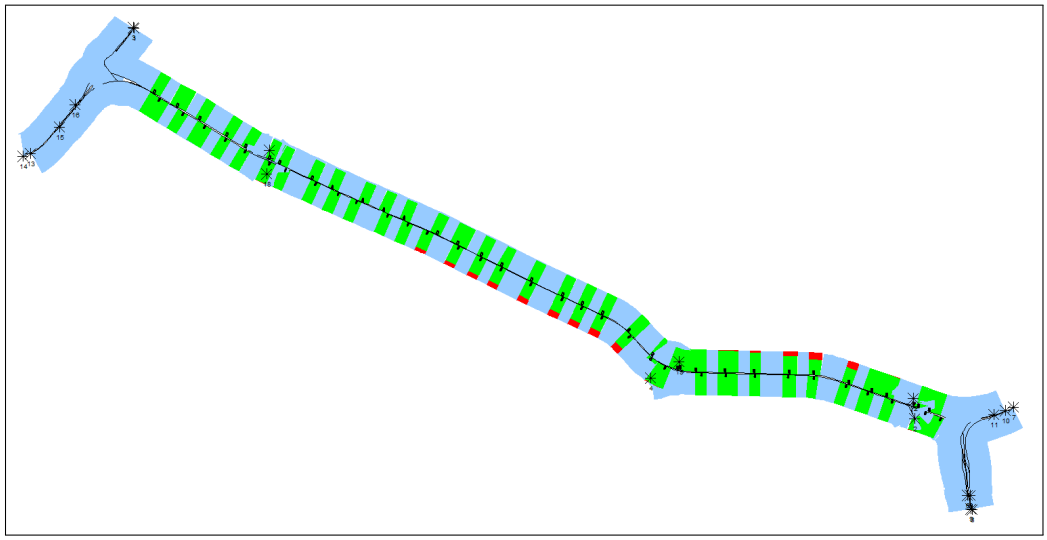
(e) Weight flow 0.001

(f) Weight flow 0

Figure 34: Function values for a measured flow of 1284 and a measured speed of 24 on link 3 of network 2 with different weights on flow

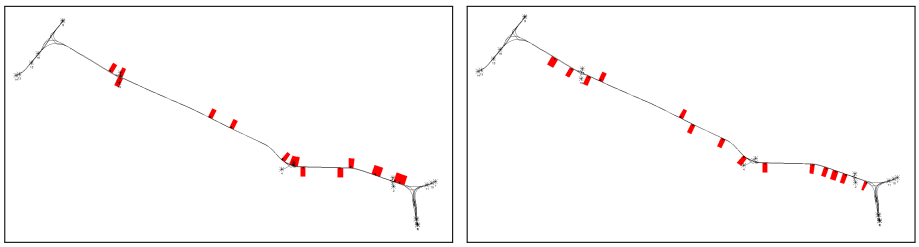


(a) Start matrix 1



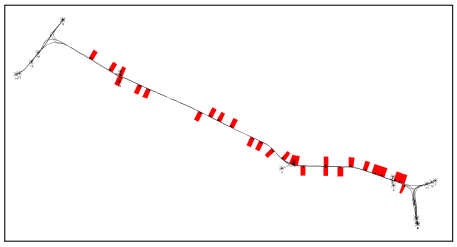
(b) Start matrix 2

Figure 35: The start matrices 1 and 2 compared to the measured values for all measurements-green indicates the equal part, red and blue indicate the unequal parts

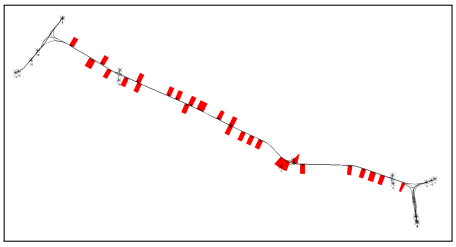


(a) 20 percent situation 1

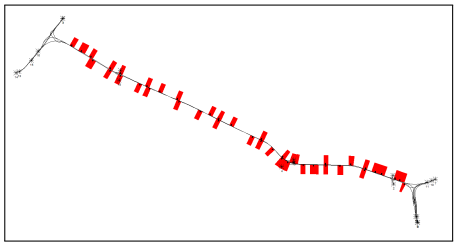
(b) 20 percent situation 2



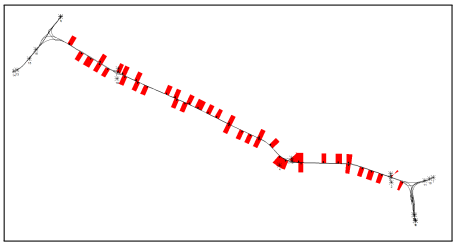
(c) 40 percent situation 1



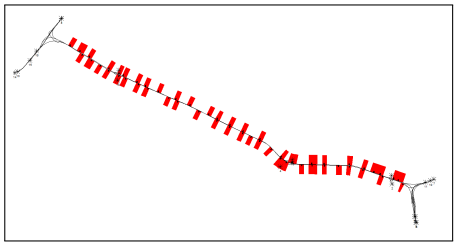
(d) 40 percent situation 2



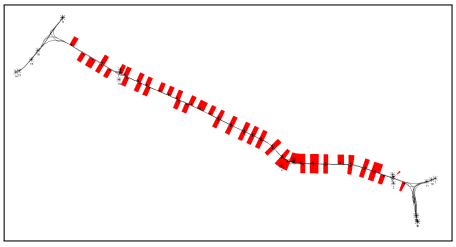
(e) 60 percent situation 1



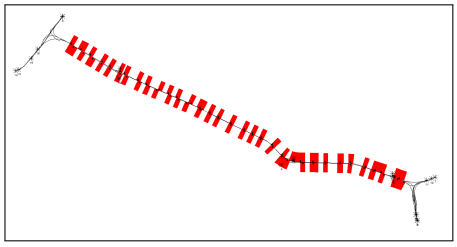
(f) 60 percent situation 2



(g) 80 percent situation 1



(h) 80 percent situation 2



(i) all measurements

Figure 36: count locations for the different situations of section 5.5