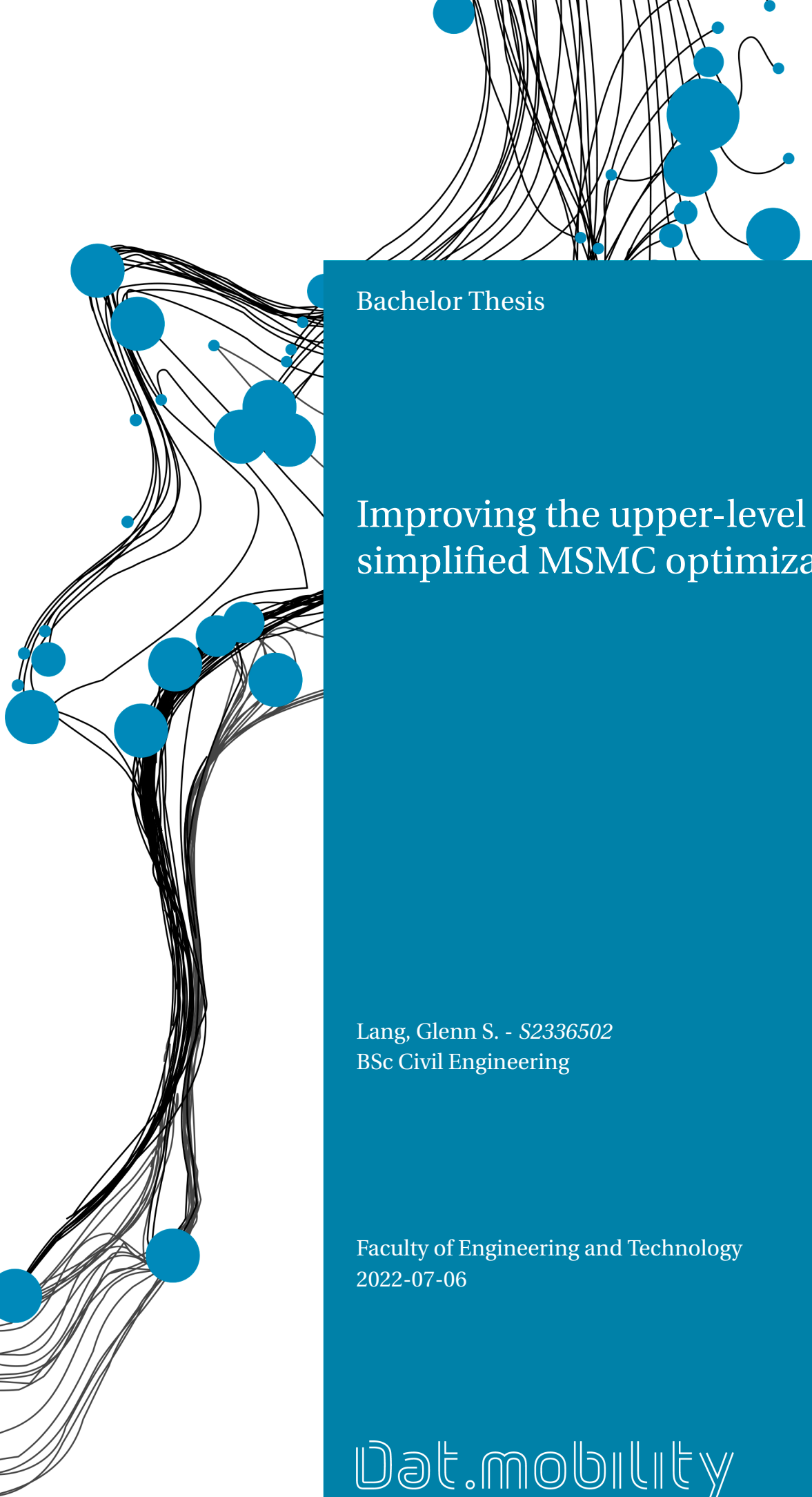


Bachelor Thesis

Improving the upper-level solver for simplified MSMC optimization problems

Lang, Glenn S. - *S2336502*
BSc Civil Engineering

Faculty of Engineering and Technology
2022-07-06

Dat.mobility
ADVANCING ANALYTICS

UNIVERSITY OF TWENTE.

Colophon

Title	Improving the upper-level solver for simplified MSMC optimization problems
Date	2022-07-06

Author	G.S. Lang (Glenn)
Student Number	S2336502
Email	g.s.lang@student.utwente.nl

Internal Supervisor	Dr. K. Gkiotsalitis (Konstantinos)
External Supervisor	L. Brederode (Luuk)
Second Assessor	Dr.ir. M. van den Berg (Marc)

Company	Dat.mobility

Preface

From here on lies my Bachelors Thesis, *"Improving the upper-level solver for simplified MSMC optimization problems"*. This paper marks the conclusion of my Civil Engineering Bachelor at the University of Twente, and reports the findings of my research, which was performed at the behest of Dat.mobility. But before moving on, I would like to thank all the people who supported me throughout this endeavor.

Firstly, I would like to express my gratitude to my internal supervisor, Kostas Gkiotsalitis. You consistently gave me good feedback, helped me structure the scope of my project, and were an integral part of keeping on top of organizational matters. Particularly your clear-headedness and sound view on my work were very important to me.

Secondly, I would like to thank my supervisor at Dat.mobility, Luuk Brederode. Truly, I have nothing but respect for your enthusiasm and the encouragement you have shown me throughout this research. You were always there to answer my questions, however trivial they may be, or give me feedback on my work, helping me clear my mind on many occasions.

Finally, I would also like to thank my friends and family, who supported me through this experience and showed me nothing but love and encouragement.

Glenn S. Lang
July 2022, Deventer

Summary

Travel demand origin-destination matrix estimation is a core, but expensive, aspect of transport modeling, especially in the context of large scale modeling. They are used to store, and subsequently generate the trips from an origin to a destination. In this research, the upper level of a bi-level and iterative matrix calibration method, known as multi source matrix calibration (MSMC), is improved upon. This upper-level consists of a convex-quadratic numerical optimization problem, which, by changing OD matrices, aims to minimize the differences between observed and modeled link flows, congestion patterns and delays. This upper-level is implemented in MATLAB and solved by its proprietary solver FMINCON, in combination with the interior-point algorithm. But, this solver shows sporadic, and thus unreliable, tendencies within the upper-level computation times. The first approach taken to improve upon this implementation is to find another, more dependable solver.

An initial list of ten solver-algorithm combinations is presented, determined by considering the characteristics of the problem (convex, quadratic and large-scale), as well their capability to integrate with MATLAB. These ten solvers are then evaluated on two separate networks. The first being the smaller academic network Sioux Falls, which is used to filter out the worst performing solvers, by comparing them to the current implementation. With these preliminary tests, this initial list reduced to only five. The second network of the province of Noord-Brabant, being more realistic, is used to assess the solvers for scalability and their performance in a real-world scenario. From this list, none were able to be applied, due to them running out of system memory.

After failing to apply these solvers to the Noord-Brabant network due to running out of system memory, a method is stipulated to reduce the problem size. This method makes use of the available gradients, by removing OD-pairs whose absolute gradient values show a low amplitude, as these pairs have a relatively minimal impact on the results of the calibration. Two separate gradient selection methods are then presented. The first is a static method, that selects the OD-pairs based on a set percentage. The second is a dynamic method, which chooses the OD-pairs based on the mean and standard deviation of the distribution of their gradients. This approach of reducing the problem size is shown to have similar convergence in tests on Sioux-Falls, with the dynamic method standing out as the best. Further, when applying this methodology on the network of Noord-Brabant, though the dynamic method shows comparatively worse convergence, the static method, when keeping 50% of the total problem size, is able to reduce the total computation time from 59 to 46 hours, without compromising the final convergence substantially, as the final objective function value is within 1% of the original implementation.

It is thus concluded that, with the original implementation, none of the alternative solvers are capable enough to perform in a realistic scenario. Additionally, exploratory results with regards to the problem size reduction are shown to be capable of solving the current issues reliably. Thus, this approach is deemed as an improvement upon the current implementation.

Table of Contents

1 Introduction

Transport infrastructure is an important part of modern civilization. Starting from when the Romans first built their roads to transport their goods throughout Europe to modern times, where they ensure that a person can go anywhere they wish to. Though, due to the constantly increasing population, the current infrastructure has failed to meet the demands it is subjected to (Saw et al., 2015), leading to cases of congestion and thus, travel delays. For this reason, predictive transport modeling has become a core aspect of infrastructure planning, as it, for example, aims to give insight into future traffic conditions for policy makers (Brederode et al., 2018), give an overview of a complex system, or explore impacts of planned interventions (Næss et al., 2014). To this end, many different traffic assignment models were developed and implemented, primarily divided between dynamic and static assignment models (Saw et al., 2015), as well as varying travel demand origin-destination (OD) matrix estimation methods (Hamerslag & Immers, 1988). These estimation methods are generally at the center of current and predictive traffic modeling, as an OD matrix is the defining factor when generating the trips that will be performed within a transportation network (Hamerslag & Immers, 1988).

Recently, Brederode et al. (2020) developed multi source matrix calibration (MSMC), a new OD matrix estimation method. This new method is comprised of an iterative calibration process, with two alternating levels, an upper and a lower level (Brederode et al., 2020). The upper level makes use of parameter values generated by the lower level and calibrates a travel demand OD matrix by means of a numerical optimization problem. The lower level then aims to translate the (calibrated) OD matrix into traffic flows via a static capacity constrained traffic assignment (SCCTA) model or a static capacity restrained traffic assignment (SCRTA) model (Brederode et al., 2020). The subject of this research is the solving of the upper level optimization problem.

The remainder of the report is structured as follows: The rest of section 1 aims to elaborate upon the context of the assignment, by introducing the involved parties, as well as the current issues related to the assignment. Additionally, section 1.2 explicitly states the research objective and associated questions. Section 2 aims to outline the criteria which demarcate the possible solvers, as well as their meaning. Furthermore, nonlinear solution algorithms are presented. This is followed by section 3, where the methodology followed to arrive at answers for the two research questions is explained. Section 4 then presents the results arrived at for the comparison of the solvers. The next section, section 5, goes into a different direction and presents the route taken to account for the results of the solver comparison. Following that is section 6, where the results associated with the method stipulated in the previous section are presented. Finally, conclusions (section 7), discussions (section 8) and recommendations 9) close off the report.

1.1 Context

1.1.1 Involved Parties

The primary party involved is Dat.mobility, the party who commissioned the project. Though the results of this project could have implications for many transport modeling agencies, as efficient

modeling is particularly important in modern times. Furthermore, indirectly implicated by this project are the parent company of Dat.mobility, Goudappel, and the Netherlands as a whole, since currently, about 75% of the Dutch traffic models are maintained within OmniTRANS, a toolbox developed by Dat.mobility, which will be utilizing MSMC in future. By utilizing the results of this research, more robust and reliable traffic prediction models can be developed, enabling the state to implement a more dependable road network, as well as corresponding policies, thus, ensuring all travelers a safe and timely journey.

Additionally, it must be noted that the current implementation of MSMC is as a prototype in MATLAB. Whereas, the final implementation of MSMC is likely to be either in Java or C++ and Dat.mobility thus also wishes for the solver, which is utilized in the upper level, to abide by other criteria, which fall outside the scope of this research. These include the type and cost of the license, as well the ability to seamlessly migrate the solver from the prototype to the final product.

1.1.2 OD Matrix Calibration

Travel demand matrix calibration is an important component of traffic modeling (Hamerslag and Immers, 1988; Willumsen, 1978), with a wide variety of methods having been developed. Such as the principle component analysis based method developed by Djukic et al. (2012), or the random forests based approach developed by Saadi et al. (2017). Although these methods may all differ in their execution, their ultimate aim is to minimize the differences between observed and modeled link flows, congestion patterns and delays, by changing an initial OD matrix (Brederode et al., 2020; Saadi et al., 2017).

1.1.3 Optimization Problem

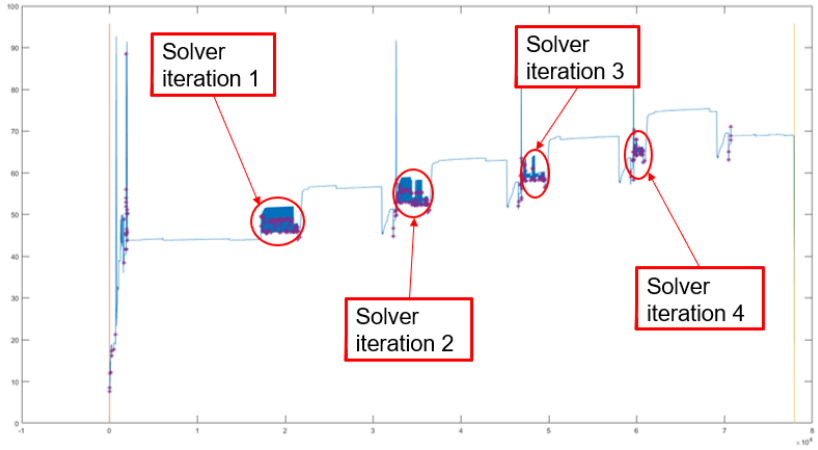
The upper-level of the MSMC approach is composed of a large-scale, linearly-constrained optimization problem, which calibrates the travel demand matrix ($\mathbf{D}^*$). It is shown in Equation 1 (Brederode et al., 2020). The aim is to minimize the objective function F, which is composed of the differences between observed (or prior) and current OD demand ($\mathbf{D}_0$ and $\mathbf{D}$), current and observed link flows ($\mathbf{y}(\mathbf{D})$ and $\widetilde{\mathbf{y}}$), as well as current and observed route queuing delays ($\boldsymbol{\tau}(\mathbf{D})$ and $\widetilde{\boldsymbol{\tau}}$). Furthermore, each of these differences can be weighted with the corresponding weight w_i, where $i \in \{1,2,3\}$. Finally, it must be noted that this program has been proven to be convex and smooth.

$$
\begin{aligned}
\mathbf{D}^* = \underset{\mathbf{D}}{\operatorname{argmin}}(\mathrm{F}) = \underset{\mathbf{D}}{\operatorname{argmin}} \left(w_1 \sum (\mathbf{D} - \mathbf{D}_0)^2 + w_2 \sum (\mathbf{y}(\mathbf{D}) - \widetilde{\mathbf{y}})^2 + w_3 \sum (\boldsymbol{\tau}(\mathbf{D}) - \widetilde{\boldsymbol{\tau}})^2 \right) \\
\text{Subject to:} \quad \mathbf{y}(\mathbf{D}) = \widehat{\boldsymbol{\alpha}} \boldsymbol{\Psi} \mathbf{D}_{k-1} + \frac{\partial \widehat{\boldsymbol{\alpha}}}{\partial \mathbf{D}} \boldsymbol{\Psi} \mathbf{D}_{k-1} (\mathbf{D} - \mathbf{D}_{k-1}) \\
\boldsymbol{\tau}(\mathbf{D}) = \frac{T}{2} \left(\frac{1}{\widetilde{\boldsymbol{\alpha}}(\mathbf{D}_{k-1})} - 1 \right) - \frac{T}{2} (\mathbf{D} - \mathbf{D}_{k-1})^T \left(\frac{\partial \widetilde{\boldsymbol{\alpha}}(\mathbf{D})}{\partial \mathbf{D}} \cdot \frac{1}{\widetilde{\boldsymbol{\alpha}}^2(\mathbf{D}_{k-1})} \right) \\
\mathbf{0} < \mathbf{D} \leq \bar{\mathbf{D}} \\
\chi_j \left(\sum_{i \in I_j} T_{ij}(\mathbf{D}) - \delta_j R_j \right) \leq 0 \; \forall \; j \in J_{\bar{\mathbf{D}}}
\end{aligned}
\tag{1}
$$

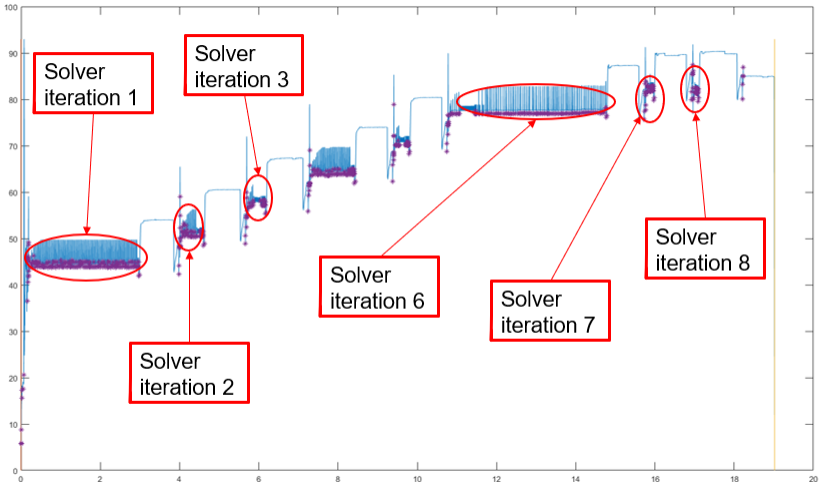
Where $\widehat{\boldsymbol{\alpha}}$ and $\boldsymbol{\Psi}$ are matrices containing acceptance factors for all route-link combinations and route choice probabilities, respectively, $\mathbf{D}_{k-1}$ is a prior solution for the travel-demand matrix, $\widetilde{\boldsymbol{\alpha}}$ is a vector containing flow acceptance factors, $\partial\widehat{\boldsymbol{\alpha}}/\partial\mathbf{D}$ and $\partial\widetilde{\boldsymbol{\alpha}}(\mathbf{D})/\partial\mathbf{D}$ are the sensitivity of $\widehat{\boldsymbol{\alpha}}$ and $\widetilde{\boldsymbol{\alpha}}$, respectively, T is the study period duration considered, $\bar{\mathbf{D}}$ is an arbitrary boundary defined by scaling $\mathbf{D}_0$, $\chi_j \in \{-1, 1\}$ depicts the constrained state of out-link j, I_j is the set of in-links for j, T_{ij} represents the response function for turn demands, R_j and δ_j are the supply of out-link j and the corresponding minimum deficit/surplus of supply, and $J_{\bar{\mathbf{D}}}$ is the set of out-links considered (Brederode et al., 2020).

1.1.4 Current Issues

Currently used to solve the optimization problem is the solver FMINCON, provided by MATLAB, with the default, "interior-point" solving algorithm (Brederode et al., 2020). This algorithm is a general-purpose nonlinear programming solution algorithm, which can utilize the available gradients of the objective function and constraints (Waltz et al., 2006).



(a) Without constraints (Time in s)



(b) With constraints (Time in $10^4 s$)

Figure 1: Solving time (x-axis) VS Computational requirements [%] (y-axis)

This algorithm (and MSMC) was previously assessed on the optimization problem under two conditions. The first test, depicted in Figure 1a, shows that the solving time required in the upper-level reduces with each iteration the solver is used. On the other hand, when the linear constraints are introduced, the computation time required for each upper level iteration becomes sporadic, as shown in Figure 1b. Specifically, during iterations 1 and 6, the solver struggles to find the solution within an acceptable time. Additionally, the total computation also grows substantially.

1.2 Research Objective

Problem Statement
The current solving method utilized to solve the constrained quadratic optimization problem is sub-optimal, as the number of steps required to converge is quite large, sometimes it does not even converge, and the total computation time is exceedingly long (see 1.1.4).

Research Objective
The objective of this research is to improve upon the current implementation of the upper-level solver, FMINCON, used for MSMC, or find another, more efficient and reliable one. To this end, various solvers will be statistically and qualitatively compared using performance indicators.

Definition of "*Improve Upon*"
Due to ambiguity of the words "*improve upon*", they must be further defined. In the context of this research, the definition is as follows: "to reliably speed up and stabilize the convergence rate of the solver whilst maintaining (or improving) its accuracy".

1.2.1 Research Questions

From the research objective, and the underlying context, two specific research question can be identified.

As depicted within the literature review (2.2), there are many different solvers, and their corresponding algorithms, that are applicable to the optimization problem of Equation 1.

Which of these solvers is the most efficient?

To answer the question posed above, answers to the following sub-questions are required.

- *How can the efficacy of a solver be determined?*
- *How can the concerned optimization problem be translated into a form which is accepted by the strictly quadratic solvers?*

The second question follows from section 1.1.4. As depicted, the computation time of the current solver changes to be more sporadic when the second constraint of the optimization problem is included.

Is the cause of these issues the solver utilized or the optimization problem?

2 Literature review

The aim of this research is, as depicted within the research objective (1.2), to improve upon the current implementation of the solver utilized to solve the constrained-quadratic optimization problem of Equation 1. To understand which solvers and their corresponding solution algorithms are suitable for the task, it is important to first re-frame the characteristics of the problem, as depicted in 1.1.3.

2.1 Key Characteristics

2.1.1 Constrained & Quadratic

The problem of interest is a linearly constrained and quadratic optimization problem, where the aim is to minimize the objective function F of Equation 1 (Brederode et al., 2020). This means that the solution to the optimization problem must satisfy the given conditions, i.e., being part of the set of feasible solutions $\boldsymbol{\Omega}$, shown in Equation 2 (Nocedal and Wright, 2006: Chapter 12).

$$\begin{gathered} \boldsymbol{\Omega} = \left\{ \mathbf{D} \,|\, \mathbf{0} < \mathbf{D} \leq \bar{\mathbf{D}} \,\wedge\, \Phi(\mathbf{D}) \right\} \\ \text{where} \quad \Phi(\mathbf{D}) = \chi_j \left(\sum_{i \in I_j} T_{ij}(\mathbf{D}) - \delta_j R_j \right) \leq 0 \,\forall\, j \in J_{\bar{\mathbf{D}}} \end{gathered} \tag{2}$$

2.1.2 Convex & Smooth

The convexity of the objective function F and constraints is one of the more important characteristics of this optimization problem. By it having such a characteristic, any local solution found within the feasible region is proven to be a global solution (Brederode et al., 2020; Nocedal and Wright, 2006: Chapter 1). Furthermore, the objective function and constraints are proven to be smooth (Brederode et al., 2020): Their gradients are explicitly available, which is an exploitable characteristic, as shown by the interior-point solution algorithm utilized by FMINCON (Brederode et al., 2020; Waltz et al., 2006).

2.1.3 Large-scale

The final, restraining characteristic of the optimization problem, is that it is large-scale (Brederode et al., 2018). A large-scale program is an optimization problem with more than thousands of decision variables (Nocedal and Wright, 2006: Chapter 7). Due to this, computation times can be quite substantial, and the efficiency of the chosen algorithm and solution process is particularly important, as computational resources are finite. In the case of this program, the networks associated with the travel demand matrix realistically comprise upwards of a thousand centroids, which is directly translated into the size of $\mathbf{D}$, the decision variable(s) (Brederode et al., 2020).

2.2 Possible Solution Algorithms

From these key characteristics, an overview of possible solution algorithms, specific to the concerned program, can be detailed. The following is an exhaustive list of applicable solution algorithms, both strictly quadratic and nonlinear, utilized by the solvers available in the YALMIP interface (Lofberg, 2004), a MATLAB toolbox featuring a multitude of solvers. A list of the solver-algorithm duos discussed below is shown in Table 1.

Table 1: Possible solvers with their corresponding algorithms and program characteristics

Solver	**Algorithm**	**Specifications**
OSQP	ADMM (2.2.1)	- Convex - Quadratic - Large-scale
MOSEK	IPM (2.2.3)	- Convex - Quadratic
GUROBI	Simplex (2.2.2) IPM (2.2.3)	- Quadratic - Large-scale
QUADPROG	Interior-point-convex (2.2.3)	- Convex - Quadratic - Large-scale
BARON	Branch-and-reduce (2.2.5)	- Nonlinear
SNOPT	SQP (2.2.4)	- Nonlinear - Smooth
KNITRO	IPM by Waltz et al. (2006) (2.2.3) IPM by Byrd et al. (1999) (2.2.3)	- Nonlinear - Large-scale
FMINCON	IPM by Waltz et al. (2006) (2.2.3) SQP (2.2.4)	- Nonlinear - Large-scale

2.2.1 Alternating Direction Method of Multipliers

The alternating direction method of multipliers (ADMM) is a recently developed first order, operator splitting algorithm. It is utilized by the solver OSQP and is shown to be applicable to large-scale and convex quadratic programs (Boyd et al., 2011; Stellato et al., 2020). The general form utilized by OSQP is shown in Equation 3. Important to note are the drawbacks of ADMM: it is heavily data dependent, and sensitive to the initial user-input (Stellato et al., 2020). Though, OSQP aims to combat these through a preconditioning phase that uses symmetric matrix equilibration (Stellato et al., 2020).

$$\begin{aligned} \underset{x\in\mathbb{R}^n}{\text{minimize}} \quad & \frac{1}{2}x^T Px + q^T x \\ \text{subject to} \quad & l \leq Ax \leq u \end{aligned} \tag{3}$$

Where $P \in S_+^n$ is a positive semi-definite matrix, $q \in \mathbb{R}^n$, $l, u \in \mathbb{R}^m$ and $A \in \mathbb{R}^{m\times n}$.

2.2.2 Simplex Method

The simplex method was developed to be applicable to linear programs. But, since the late 1950s, researchers have been expanding upon the method to apply it to quadratic programs (van de Panne

and Whinston, 1969; Wolfe, 1959) and it is currently utilized as the non-default algorithm for quadratic optimization problems in the GUROBI solver (Gurobi Optimization, 2022b). The general programs the algorithm is applicable to are linearly constrained convex quadratic programs (Wolfe, 1959). Though, it is not the first choice when tackling large-scale problems (Gurobi Optimization, 2022a).

2.2.3 Interior-Point

The interior-point method (IPM) is the default algorithm utilized by the solver FMINCON (MathWorks, 2022a), as well as GUROBI (Gurobi Optimization, 2022a), KNITRO (Byrd et al., 2006), MOSEK (MOSEK, 2022), and QUADPROG (MathWorks, 2022b). Furthermore, it is the family to which the algorithm currently used to solve the considered optimization problem belongs.

In the past, the IPM, also known as barrier method, was solely utilized to solve linear optimization problems. Though, recent technological advancements have been made to prove that this method is also efficiently applicable to nonlinear programming (Nocedal and Wright, 2006: Chapter 16) and specifically large-scale optimization problems (Gurobi Optimization, 2022a; Nocedal and Wright, 2006: Chapter 19). The basic nonlinear IPM is composed of two combined iterative processes. The outer iteration loop is responsible for setting the barrier parameter (μ_k), giving the method its second name, and stops when a test with regards to the nonlinear optimization problem is satisfied. The parameter μ_k further constrains the set of feasible solutions, reducing the time required to find an approximate solution. The interior loop is responsible for finding a feasible, approximate solution, until an error threshold is satisfied (Nocedal and Wright, 2006: Chapter 19).

From the basic algorithm, three specific implementations will be further explored. the algorithm utilized within FMINCON and KNITRO, developed by Waltz et al. (2006), the algorithm developed by Byrd et al. (1999), also utilized by KNITRO, and the algorithm known as "interior-point-convex" implemented in QUADPROG, the generic quadratic programming solver proprietary to MATLAB (MathWorks, 2022b). The algorithms utilized by GUROBI and MOSEK are not further explored, as no exact specifications are given (Gurobi Optimization, 2022a; MOSEK, 2022).

Interior-point-convex

The interior-point-convex (IPC) algorithm was specifically developed for large-scale convex-quadratic programs, subject to linear constraints (Altman and Gondzio, 1998, MathWorks, 2022b), with the generic form of shown in Equation 4. The general approach consists of a pre-solve phase, where the program is simplified as much as possible and redundancies are eliminated, followed by the predictor-corrector technique described by MathWorks (2022b).

$$\begin{aligned} \underset{x\in\mathbb{R}^n}{\text{minimize}} \quad & \frac{1}{2}x^T Hx + f^T x \\ \text{subject to} \quad & Ax \leq b \\ & A_{eq}x = b_{eq} \\ & l \leq x \leq u \end{aligned} \tag{4}$$

Where $f, l, u \in \mathbb{R}^n$, $b, b_{eq} \in \mathbb{R}^m$, $A, A_{eq} \in \mathbb{R}^{m\times n}$ and $H \in \mathbb{R}^{n\times n}$.

Interior-point by Waltz et al. (2006)
The IPM developed by Waltz et al. (2006) is a combination of the two main distinctions of the generic algorithm, the line-search IPM and the trust-region IPM (Nocedal and Wright, 2006: Chapter 19; Waltz et al., 2006). Equation 5 depicts the general program the algorithm is applied to.

$$\begin{aligned} \underset{x\in\mathbb{R}^n}{\text{minimize}} \quad & f(x) \\ \text{subject to} \quad & h(x) = 0 \\ & g(x) \leq 0 \end{aligned} \tag{5}$$

Where $f : \mathbb{R}^n \to \mathbb{R}$, $h : \mathbb{R}^n \to \mathbb{R}^l$ and $g : \mathbb{R}^n \to \mathbb{R}^m$.

Interior-point by Byrd et al. (1999)
The IPM developed by Byrd et al. (1999) aims to incorporate SQP into the trust-region IPM, with the intended application of efficiently solving large-scale, nonlinear programs. The use of the SQP algorithm is to inexpensively solve the barrier sub-problems generated by the interior-point algorithm.

2.2.4 Sequential Quadratic Programming

Sequential quadratic programming (SQP) was the nonlinear algorithm that dominated the industry before the interior-point algorithm became applicable to nonlinear optimization problems (Nocedal and Wright, 2006: Chapter 18). It is suitable for both small and large-scale programs, and is utilized by the solvers FMINCON, in its non-default state (MathWorks, 2022a), and SNOPT (Gill et al., 2021).

2.2.5 Branch-and-Reduce

The branch-and-reduce (BAR) algorithm, to the knowledge of the author, seems to be fairly unknown, only being employed by the solver BARON for nonlinear optimization problems (Sahinidis, 1996). Its typical application is non-convex problems (Sahinidis, 1996), though it is also proven to be applicable to nonlinear programs in general (Sahinidis, 2017). The algorithm is iterative, making use of the branch-and-bound and range reduction methods to find an approximate solution of the problem. The iterations conclude once a certain error threshold has been satisfied. Furthermore, on linearly constrained concave problems, exact solutions can be found (Sahinidis, 1996).

The generalization of the optimization problems the branch-and-reduce algorithm can solve is shown in Equation 6.

$$\begin{aligned} \underset{x\in X}{\text{minimize}} \quad & f(x) \\ \text{subject to} \quad & g(x) \leq 0 \end{aligned} \tag{6}$$

Where: $f : X \to \mathbb{R}$, $g : X \to \mathbb{R}^m$ and $X \subset \mathbb{R}^n$.

3 Research Methodology

To answer the research questions, and ultimately, fulfill the research objective, a methodology must be followed. Figure 2 shows the general framework the research will follow. Furthermore, sections 3.1 and 3.2 address the steps that will be followed for each research question in more detail.

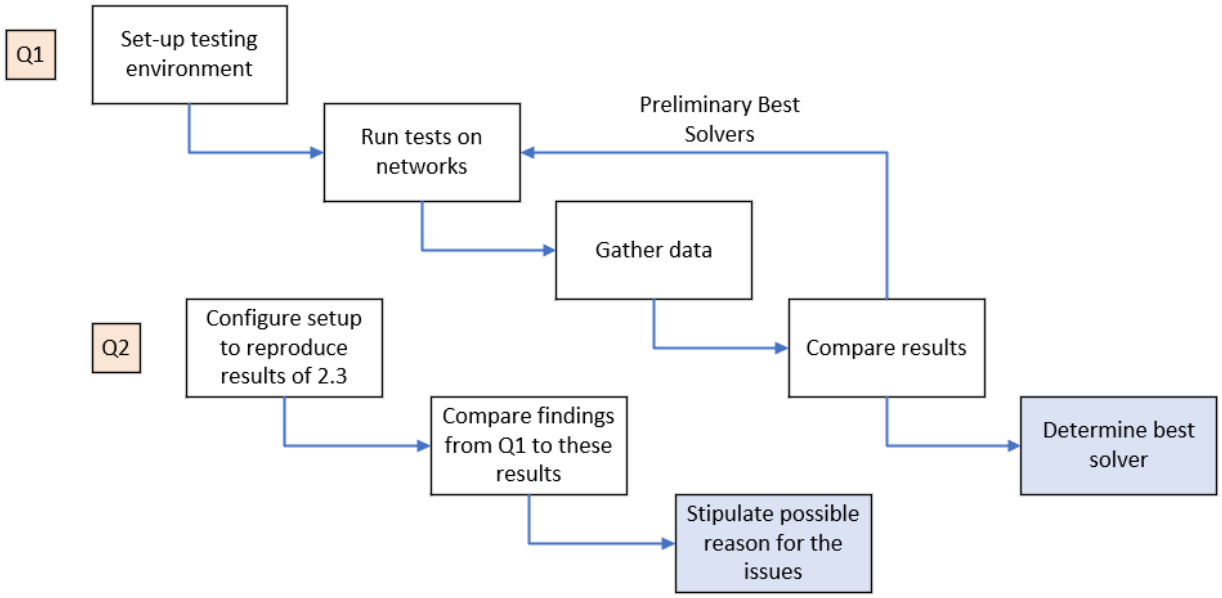


Figure 2: Proposed research methodology

The general framework is split between the two questions, demarcated by the orange squares. For each separate methodology, the final desired answer is depicted by a blue objective, while the steps needed to arrive there are in white.

3.1 Which of these solvers is the most efficient?

The research methodology proposed here makes use of the YALMIP interface (Lofberg, 2004) to integrate the available solvers into the current implementation of the model. This interface allows one to easily change which solver is being used, without having to change any other script details. Furthermore, the interface allows strictly quadratic solvers to not be restricted by the optimization problem parameters they must receive. This is especially important in this case, as the problem is not configured in accordance with Equations 3 and 4, which have an explicit matrix H or A. This is achieved by YALMIP processing the optimization problem before redirecting it to the selected solver (Lofberg, 2004). In addition to the YALMIP interface, nonlinear solvers that have a native interface, such as KNITRO or BARON, will be interfaced directly.

The first step consists of setting up the testing environment. The main point included under this bracket is familiarization with the current upper-level implementation, i.e., changing the networks and parameters to the desired ones. Furthermore, the production of the desired data and graphics

must be ensured. The form and type desired will be further discussed for the "gathering data" step. Additionally, this step also includes setting up YALMIP and the desired solvers.

3.1.1 Production of Data

The third step is running tests, which is split into two phases. The first phase consists of running the solvers on a smaller network, and the second phase, which is after the first results comparison, tests the preliminary "best" solvers on the larger network of the Noord-Brabant province, Netherlands. The network that will be utilized for preliminary testing is an adapted Sioux Falls network (Brederode et al., 2020), with Figure 3 depicting the general outline of the network. The adapted network consists of 24 centroids (origins/destinations), acting as nodes, and 35 links (Brederode et al., 2020). Each run on the Sioux Falls network will be set to continue until the MSMC criteria are met, with a maximum final limit of 50 iterations. For comparison, the reference solver-algorithm duo converges in four iterations, as shown in Figure 4.

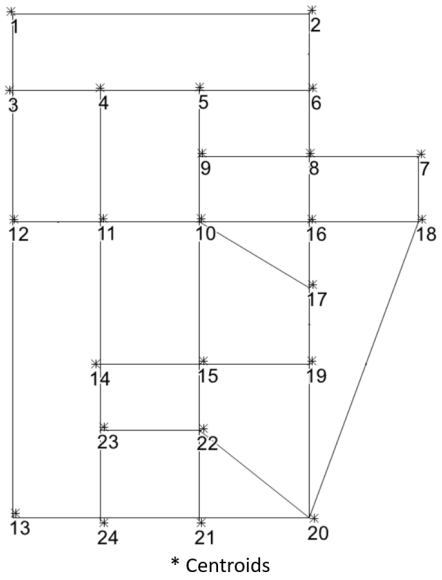


Figure 3: Sioux Falls network

During the second phase, the solvers will be tested on the Noord-Brabant transportation network, which consists of 1425 centroids, 145'269 links and 103'045 nodes. The aim is to perform tests on the scalability of the "best" solver, i.e., to see if it is a viable option in a realistic scenario. For each run, the maximum number of iterations performed will be 10, so long as the convergence criteria are not met beforehand.

The data generated through the tests will take two primary forms: solving/convergence time per iteration, with the associated RAM usage (as shown in section 1.1.4), and convergence graphs, such as the ones shown in Figure 4. Figure 4 is composed of 5 separate graphs. The four on the left (purple, red, green and blue) depict the objective function value (F) and the values of each component (associated with the weights w_1, w_2 and w_3). In addition to the value (depicted by the left axis of the graphs), a percentage is shown. For the Objective function value graph (purple) it

depicts the percentage of the final objective function value with respect to the initial objection function value. For the three other graphs, it depicts the average percentage error (sum squared error) between the modeled and observed values of that component. The final graph (yellow) depicts the violation of constraints. On the left axis is the total number of vehicles violating the constraints, and on the right axis is the number of links j for which the constraints were violated. It is important to note is that, for Sioux Falls, solely the convergence graphs will be utilized to gain insight, as the RAM usage and computation time are indistinguishable between solvers. For the Noord-Brabant network, on the other hand, the most suitable solvers will be distinguished by utilizing both data types, according to the details of the research objective.

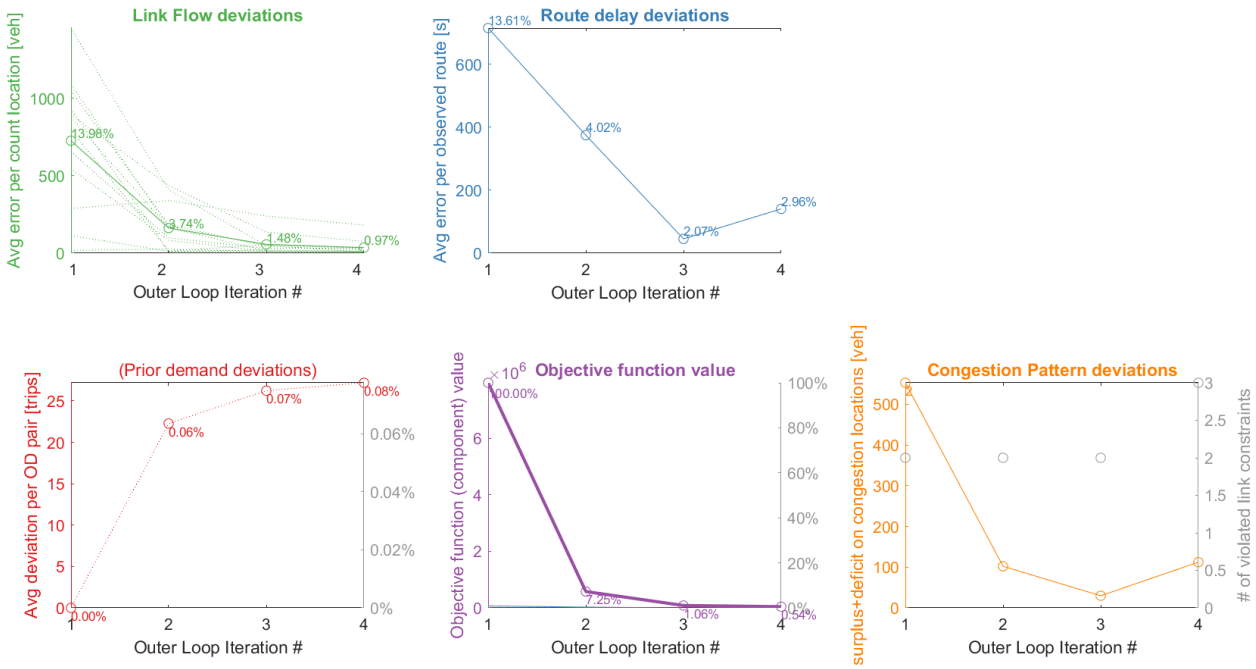


Figure 4: Example convergence graphs - Reference

3.2 Is the cause of these issues the solver utilized or the optimization problem?

The methodology proposed for the second question is straightforward. The first step taken to arrive at the desired goal will be to reproduce the results shown in section 1.1.4 with the current setup. Following that, the results gathered for the first research question will be utilized to gain insight into the possible reasons for the issues. Currently, two general possibilities are stipulated: firstly, if the behavior for all solvers is shown to be similar, then the issues most likely lie with the constraints themselves. Secondly, if this behavior is not shared, then the solver-algorithm duo was not suitable for the problem.

4 Results - Solver Comparison

An overview of the results for each solver is depicted in Table 2, for which descriptions and references can be found in Table 1. Each of the criteria mentioned follow from the research objective (1.2). Additionally, all mentioned results were retrieved on the Sioux Falls network, except for “Scalable”. The first criterion mentioned is the number of iterations. This criterion refers to the number of times the lower level was called during the full MSMC run, with the number of calls to the solver being one less. For this criterion, a low value is generally desirable, as that directly translates into a lower total run-time. The second criterion is the final convergence accuracy the real optimization problem achieves i.e., the quality of convergence. Here, the lower each component is, the more accurate the solution found is. It is split into three separate components and corresponds to the percentage associated with the objective function value graph (denoted by F), the average percentage error (sum squared error) of to the link flow deviations (denoted by w_2) and the average percentage error of the route delay deviations (denoted by w_3). Important to note is that the average percentage error of the prior demand deviations (associated with w_1) is not mentioned, as, though it is a criteria of convergence, the whole intent of MSMC is to change the original travel-demand matrix. The next criterion is the violation of constraints. This criterion is split into two components, the number of link constraints being violated and the total number of vehicles violating the constraints. In both cases, a lower value is desirable, though the latter is deemed more impactful. The penultimate criterion shows if the convergence of the objective function value is monotonously decreasing, i.e., not sporadic, depicted by "No". This is the desired behavior, as the expected behavioral pattern is thus more reliable, and it is guaranteed a solution will be found. The second possible value is "Yes", which means the solver showed sporadic convergence. The final criterion displayed here is the scalability of the solver, determined by utilizing the Noord-Brabant network. For this criterion, there are two possible values: "No", which means that the solver either gave conclusive insight into its applicability or was not tested for scalability due to the tests on Sioux Falls. The second value is "Yes", which means that the solver was applicable and produced adequate results.

Table 2: Results overview - solver comparison

Solver	**Interface**	**Algorithm**	**#i**	**Quality of convergence [%]**			**Constraint violations**		**Sporadic**	**Scalable**
				F	w_2	w_3	# links	# vehicles		
FMINCON	Native	IPM[1]	4	0.54	0.97	2.96	3	112	No	Yes
		SQP	4	0.31	0.63	2.79	0	0	No	No
	YALMIP	SQP	3	0.68	0.98	2.78	0	0	No	No
KNITRO	Native	Byrd IPM	4	0.47	0.80	4.19	1	137	No	No
		Waltz IPM	4	0.43	0.73	4.25	1	33	No	No
SNOPT	Native	SQP	4	0.50	0.82	4.36	2	153	No	No
BARON	Native	BAR	4	0.36	0.69	4.22	3	553	No	No
QUADPROG	YALMIP	IPC	22	0.25	0.62	1.52	0	0	Yes	No
GUROBI	YALMIP	IPM	3	0.36	0.91	3.01	0	0	Yes	No
OSQP	YALMIP	ADMM	> 50	n/a	n/a	n/a	n/a	n/a	Yes	No
MOSEK	YALMIP	IPM	n/a	n/a	n/a	n/a	n/a	n/a	n/a	No

[1] Reference solver-algorithm duo

4.1 Sioux Falls

For Sioux Falls, only the strictly quadratic solvers and FMINCON-SQP were tested via the YALMIP interface. The reason for which being memory problems with YALMIP, which is further elaborated upon in section 4.2. Furthermore, for this same reason, only the IPM of GUROBI is tested, and not the simplex method. Finally, for further viewing, a compilation of all relevant Sioux Falls convergence graphs, for the solver comparison, can be found in Appendix A.

4.1.1 FMINCON

Two different FMINCON-SQP variants were tested: variant 1 utilized the native interface and variant two was interfaced through YALMIP, with both showing promising results. Reasons for their diverging behavior could be the problem conversion through YALMIP or possibly differing settings. For the latter, as is shown in Table 2, a convergence of similar quality to the reference is achieved in one less iteration. Furthermore, the number of link constraints being violated is 0, meaning that the final solution reached was feasible. This behavior is also reflected in the prior iterations, as shown in Figure 5: the SQP algorithm implemented strictly abides by the constraints for each iteration.

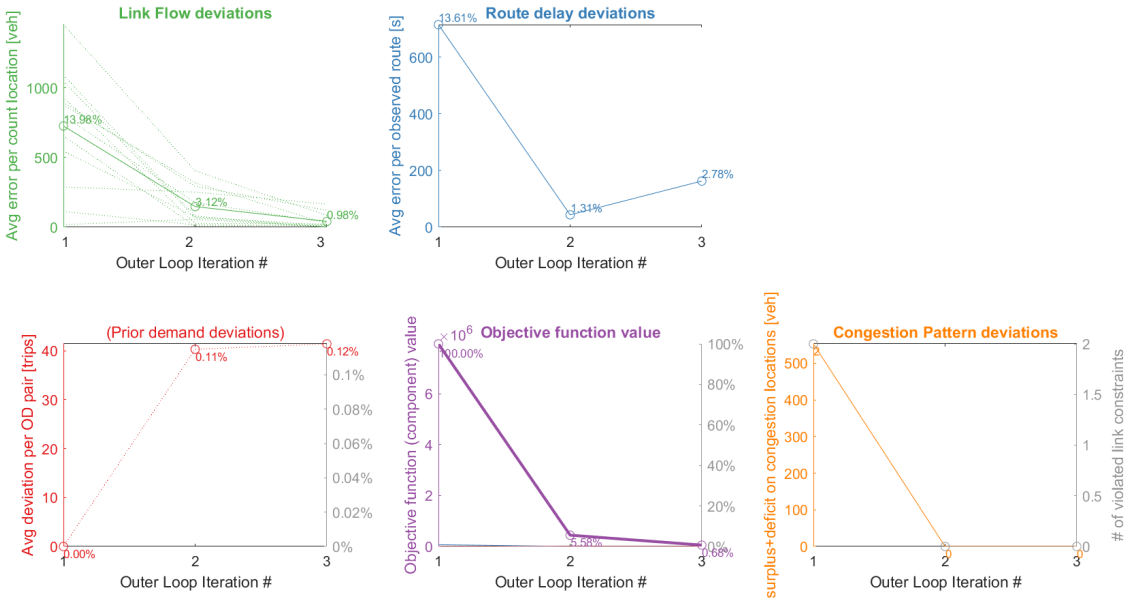


Figure 5: Sioux Falls - FMINCON-SQP + YALMIP convergence graphs

The results of the natively interfaced variant are also very promising, with it achieving a higher quality of convergence for the mentioned objective function components, as well as the objective function itself. Furthermore, this is achieved in the same number of iterations whilst strictly abiding by the set constraints, in the same manner the YALMIP interfaced variant does.

Considering these factors, FMINCON-SQP, in both interface variants, was chosen to be further tested for scalability.

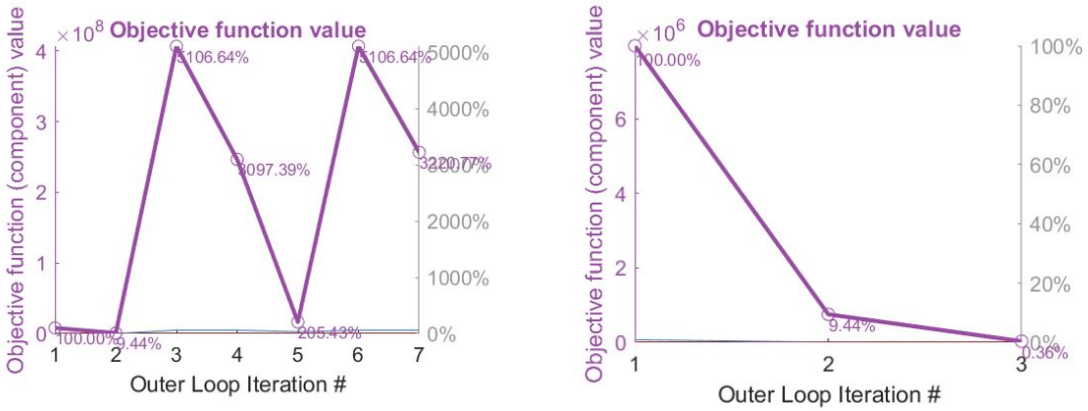
4.1.2 KNITRO and SNOPT

Both KNITRO variants, as well as SNOPT show very similar results. As shown in Table 2, all three duos achieve a lower objective function value (0.47, 0.41 and 0.5% VS 0.54%), as well as a lower average percentage error for the link flow deviations than the reference (0.8, 0.74 and 0.82% VS 0.97%). Furthermore, they achieve these results in the same number of iterations, whilst also violating less constraints. Though, as can be seen, the actual number of vehicles violating the constraints is still higher than the reference, for SNOPT and KNITRO-Byrd. Additionally, the gains made in F and w_2 came at a trade off for the average error of the route delay deviations (4.19, 4.11 and 4.36% VS 2.79%). Though, overall, their results are quite satisfactory, and all three were deemed suitable enough to be tested further for scalability.

4.1.3 GUROBI

As depicted within Table 2, the results for GUROBI are shown to be excellent across the board. For convergence speed, GUROBI requires a full iteration less to arrive at a lower objective function value (0.36% VS 0.54%) and a lower average percentage error for the link flow deviations (0.91% VS 0.97%). Additionally, this convergence is achieved with a very similar average percentage error for the route delay deviation (3.01% VS 2.96%). Finally, the solution found does not violate any of the link constraints.

On the other hand, GUROBI utilizes a solver parameter known as random "Seed", which acts as a perturbation (Gurobi Optimization, 2022c) and can very negatively impact the results of the calibration, as shown in Figure 6. The effects of changing the "Seed" is that, although GUROBI finds a solution during the first iteration, it violates the bound constraints, often leading to the problem becoming non-convex. For this reason, the GUROBI solver was deemed unsuitable, as the criterion of reliability was not met to a satisfying degree.



(a) "RandomSeed" 1

(b) "RandomSeed" 2

Figure 6: Sioux Falls - GUROBI Comparison between different "RandomSeed"s

4.1.4 BARON

BARON, although producing satisfactory results, was ultimately not applicable to the problem. Due to the nature of the implementation, it seems an issue of incompatibility arose with the way in which the link constraints ($\Phi(\mathbf{D})$) are defined in the MATLAB script, and subsequently supplied to BARON, though there is no concrete evidence. For this reason, BARON was never capable of running with these constraints enabled. The run shown in Figure 7 is thus without the constraints supplied, leading to the non-convergence of the congestion pattern deviations. In addition to this, BARON was never utilized to its full capability, instead, solving the optimization problem during it's pre-processing phase, where it makes use of a dynamic local search algorithm (Sahinidis, 2017).

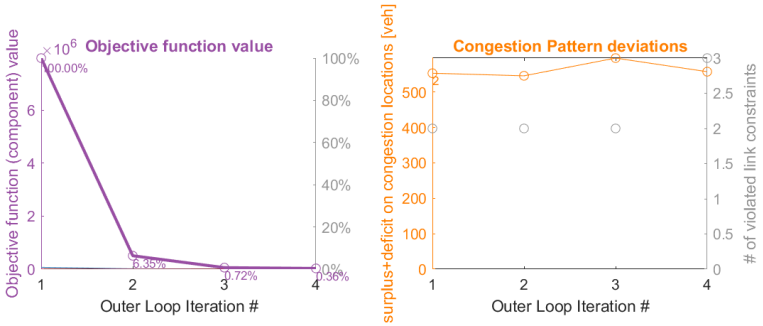


Figure 7: Sioux Falls - BARON convergence graphs

4.1.5 QUADPROG

As stated within Table 2, QUADPROG arrives at the best objective function value, overall. The same applies to the objective function components, as well as the constraints. On the other hand, 22 iterations are required for the run to converge. Furthermore, as shown in Figure 8, the objective function value is not monotonously decreasing. Highly suspicious is also the peak found at iteration 11, where the solution found does not at all seem to be reflective of the true optimization problem. These two named factors are the reason why QUADPROG was deemed unsuitable and will thus not be further tested for scalability.

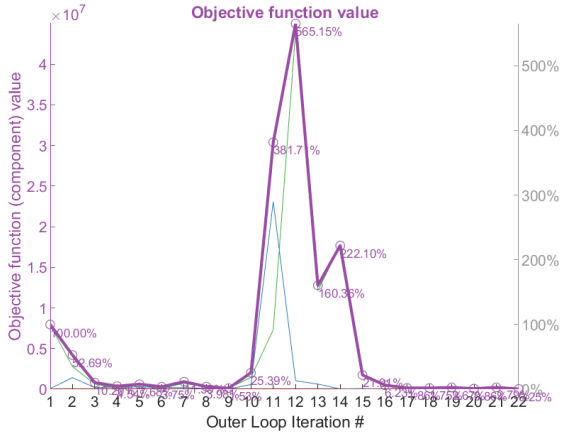


Figure 8: Sioux Falls - QUADPROG convergence graphs

4.1.6 OSQP and MOSEK

Though for different reasons, OSQP and MOSEK were deemed unsuitable for the problem. The ADMM algorithm of OSQP is quite imprecise in its approximations, and thus, it often violated the lower bound constraint ($\mathbf{0} < \mathbf{D}$) by introducing values smaller than zero. Unfortunately, a negative demand is nonsensical and the lower-level is unable to run if such a value is present. Thus, to ensure a successful run either way, the negative values of the solution found by OSQP were exchanged for machine-precision (10^{-16}) in a post-processing phase, leading to unreliable behavior. Zero was not chosen, as additional decision variables would be lost during a pre-processing phase of the MSMC (Brederode et al., 2020).

For MOSEK, the solver was simply unable to run due to returning an error signifying that the problem is non-convex, even though it has been proven to be (see section 2.1.2) and the "interior-point-convex" algorithm of QUADPROG runs without any problems. For these reasons, both OSQP and MOSEK were not further pursued as alternatives to the reference duo.

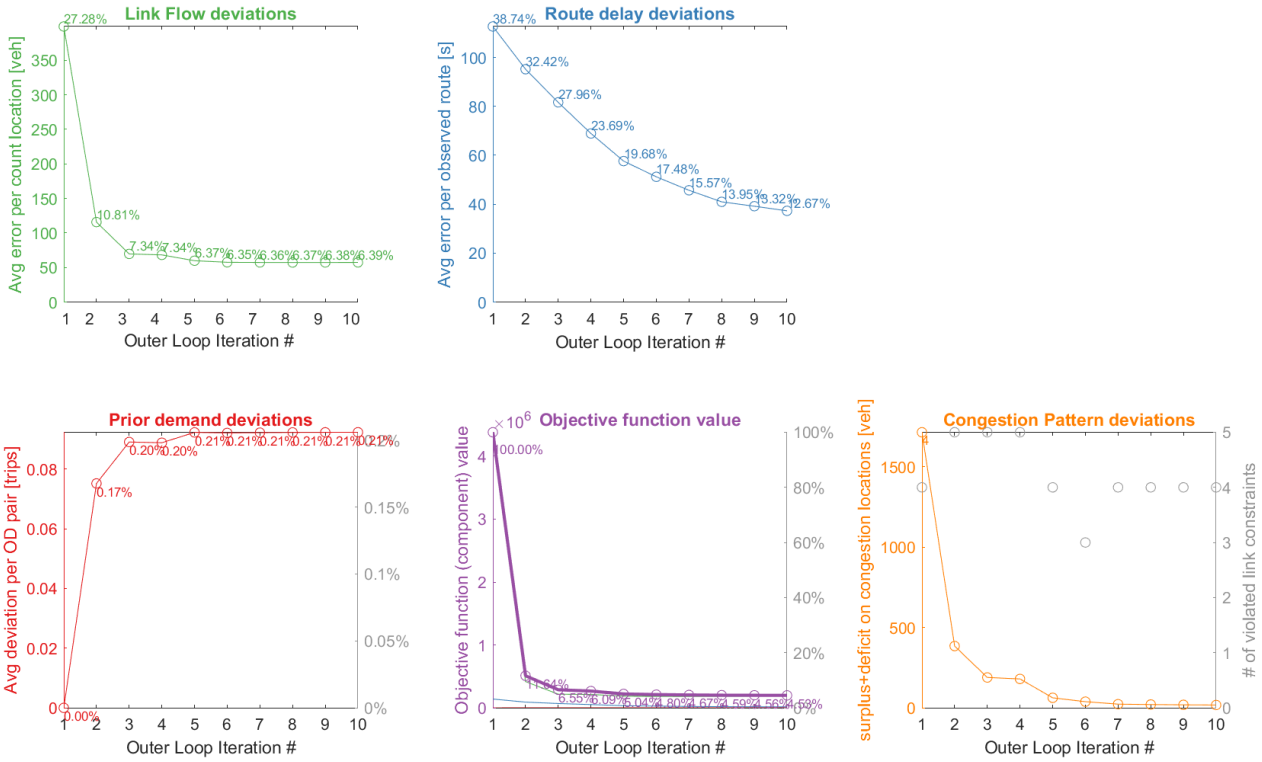
4.2 Noord-Brabant

From the original list of ten solver-algorithm duos (including FMINCON-SQP+YALMIP), five were deemed suitable and tested for scalability. These five are:

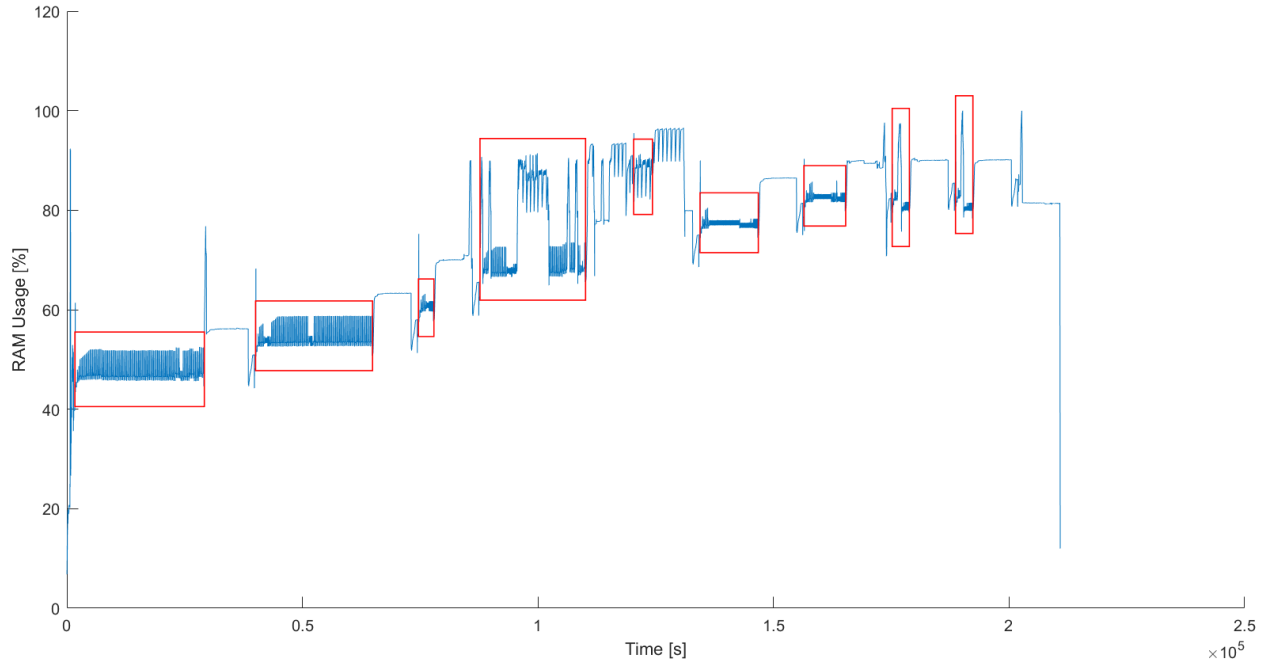
- FMINCON-SQP
- FMINCON-SQP+YALMIP
- KNITRO-Byrd
- KNITRO-Waltz
- SNOPT

Of these five duos, none of them were able to run on the Noord-Brabant transportation network. All of them, when called during the first iteration, crashed due to running out of system memory, which totaled 128GB of RAM. Important to note, is that for FMINCON-SQP interfaced through YALMIP, the solver was not called yet, and MATLAB ran out of memory when converting the problem to the format that YALMIP utilizes, which makes use of a large "object". For this reason, YALMIP was dropped from being utilized during the research.

In regards to the other four duos, all were called utilizing their native MATLAB interface, thus the issues must lie elsewhere. Though this is discussed more in-depth in section 8.1, the following depict initial, possible, reasons. A first reason would be the conversion from the MATLAB problem to the native language of the solver. As the FMINCON solver was implemented with only MATLAB in mind, the conversion from MATLAB to the a form which the solver can utilize may almost come at no cost. Whereas, this is not the case with the other solvers. Alternatively, the FMINCON implementation may also be more memory efficient, with the interface not being a problem at all. Finally, when considering FMINCON-SQP, without YALMIP, the algorithm is simply not suitable for the scale, as it requires more than 18TB of system memory. Thus, the only duo that successfully completed a run is the reference one, FMINCON-IPM. The results for its run are shown in Figure 9.



(a) Convergence graphs



(b) RAM usage and computation time (upper-level iterations are framed in red)

Figure 9: Noord-Brabant reference run - FMINCON-IPM

5 Problem Size Reduction

The results garnered on the Noord-Brabant network were the initiator for a new idea with regards to the approach taken to solve the current issues. This new approach consists of reducing the problem size, i.e., the number of OD-pairs that will be calibrated by the solver, by utilizing the available gradients. To the knowledge of the author, though this approach has been stipulated before (Leibovici, 2013), no extensive research on this approach has been carried out. An additional note, though, is that problem size reduction is a reoccurring idea, particularly for dynamic traffic assignment models, where they make use of principle component analysis (Prakash et al., 2017; Qurashi et al., 2020). But, this approach is not suitable to this application, as it exploits the periodic OD-vectors of dynamic traffic assignment models, which are combined into a single matrix.

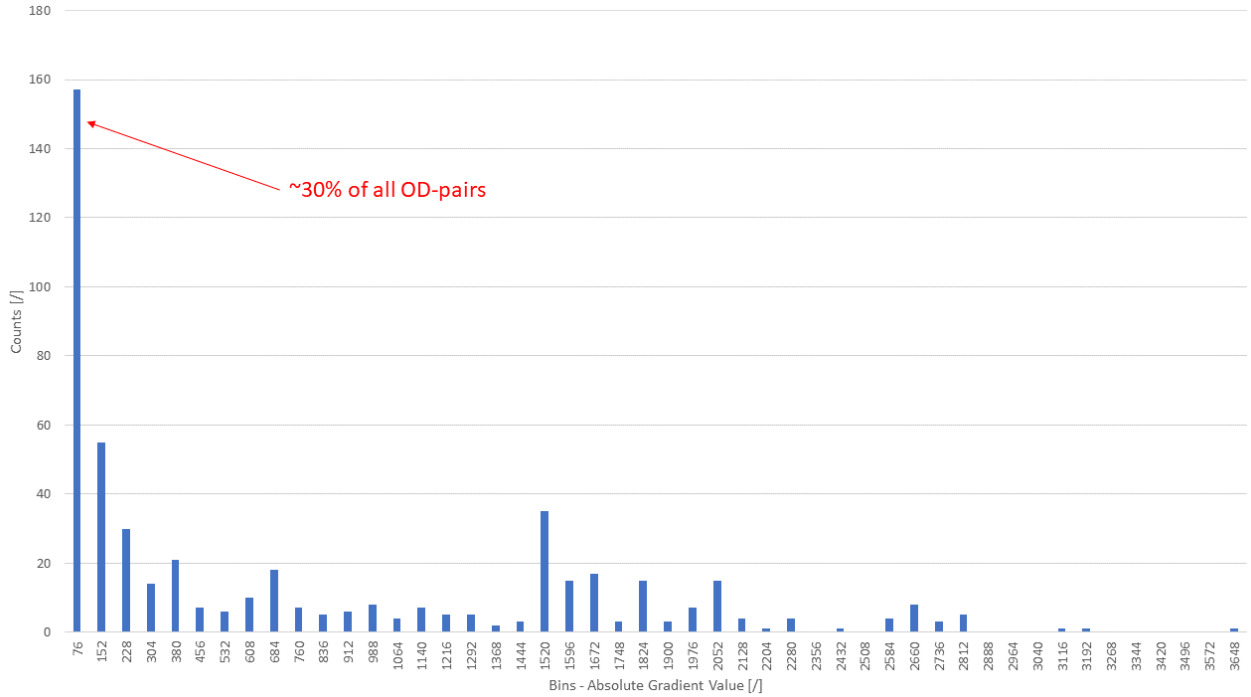


Figure 10: Histogram of the absolute gradient values per OD-pair - Sioux Falls

As is shown in Figure 10, the absolute value of the gradients per OD-pair, on Sioux Falls, as well as the Noord-Brabant network (Appendix B), form a right skewed histogram, with a very long tail. Due to this behavior, it is expected that most of these OD-pairs have a negligible contribution towards the final convergence, and can thus be removed from the problem. An example of what is meant by reducing the problem size is shown in Figure 11. As is depicted, the OD-pairs with low gradient values (x_2 and x_4) are removed from the problem ("Reduce" step) before the solver is applied ("Solve" step). After the solver is done and finds a solution, (s_1 and s_3), the removed OD-pair values are reintroduced into the vector ("Resize" step).

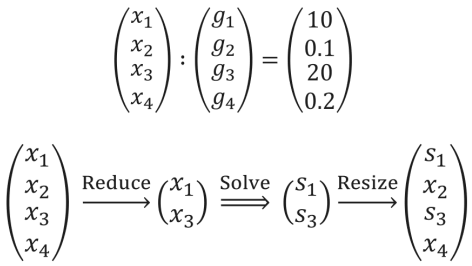


Figure 11: Example problem size reduction

5.1 Reducing the Problem Size

As stated, the method utilized to reduce the problem size will utilize the absolute gradient values. More specifically, the gradient values of the objective function F ($\delta F/\delta \mathbf{D}$). Furthermore, the gradients that will be used are produced with the current solution, i.e., the original travel demand during the first iteration, or the solution of a previous iteration, from iteration two onward.

After generating the gradients, the non-relevant OD-pairs must be removed from the problem. Two different methods were explored with this goal in mind, a static method, which sets a certain percentage and a dynamic method, which utilizes the attributes of the distribution of the absolute gradients. Important to note is that for each iteration, the relevant OD-pairs will be re-selected. The reason behind this is that the gradients of the OD-pairs will be changed with each iteration, and that previously suitable OD-pairs may now no longer be relevant enough to be re-chosen. In this manner, the most optimal solution possible, when reducing the problem size, can be arrived at.

Method 1: Static
The first method explored is the static method. In this case, a parameter is set that defines the percentage of OD-pairs that will be kept during the iteration. For example, 75%, where, for a vector with 500 OD-pairs, only the 375 with the highest gradients will be kept and altered by the solver.

Method 2: Dynamic
The aim of the second method is to make the problem size reduction more problem-agnostic, as the risk of utilizing a set percentage is that it is chosen wrongly for the network. Though, the same is applicable to the predefined scaling factor s used by the second method, so this problem isn't wholly circumvented. This method utilizes the mean (μ) and standard deviation (σ) of all OD-pair gradients to dynamically choose which OD-pairs to select. Using the mean and standard deviation, a threshold (t) is set, that defines the lowest relevant gradient value. Each chosen OD-pair d, with gradient g_d, must thus abide by the relation shown in Equation 7.

$$|g_d| \geq t = \mu - s * \sigma \tag{7}$$

Where $0 \leq s \leq 0.75 * \mu/\sigma$ is a predefined scaling factor.

Finally, it must be noted that the upper bound of the scaling factor s is determined during each iteration, to ensure that a problem size reduction takes place. This can lead to the factor s to be higher than the upper bound during that iteration. If that is the case, it is lowered to being equal to that upper bound.

5.1.1 Complete First Iteration

In addition to the reduction methods, another stipulation was proposed with regards to the utilization of the complete problem size during the first iteration. In other words, the problem size would only be reduced from iteration 2 onward. The aim of starting with a complete first iteration would be to reduce inconsistencies in data, as well as errors due to the linear approximation (Brederode et al., 2020). Thus, increasing stability, in this case, the duration of each upper level iteration, and the accuracy of convergence during the later iterations.

5.1.2 Testing

For testing the usability of the different problem size reduction methods, the same approach as the one taken for the solver comparison will be utilized, which is described in section 3.1.1. The variants will first be tested on the Sioux Falls network, utilizing different settings, before being tested on the Noord-Brabant network with the best performing parameters. Important to note is that, due to time constraints, solely the reference duo, FMINCON-IPM, will be utilized for these tests.

6 Results - Problem Size Reduction

6.1 Sioux Falls

The results for the problem size reduction on Sioux Falls presented here are spread out over three sections. The first two (6.1.1 and 6.1.2) aim to give insight into the results of the two stipulated methods, whilst the third section (6.1.3) presents the results when the first iteration is set to utilize to complete problem size. Furthermore, it must be noted that the congestion pattern deviations will not be presented, since they deviate too little from the reference to be a defining factor.

6.1.1 Static Method

Figure 12 shows the accuracy results of the first method, which are sorted by their total objective function value, with the reference being kept in the first position. As shown, for method 1, two parameters, 50% and 70%, showed promising results, by improving the accuracy across all three convergence criteria. From the remaining runs, though all are shown to be improving the objective function value and link flow deviation accuracy, it comes at the cost of diminishing the accuracy of the route delay deviations. Interesting to note is the sharp peak at 60% for the route delay deviations. A possible reason for this behavior is that, at 60%, when compared to 55%, some OD-pairs are introduced that have diverging gradients for the link flow and the route delay components (one is positive, the other is negative). And, when the percentage grows again, to 65%, OD-pairs are introduced, who have matching component gradients, combating this conflict.

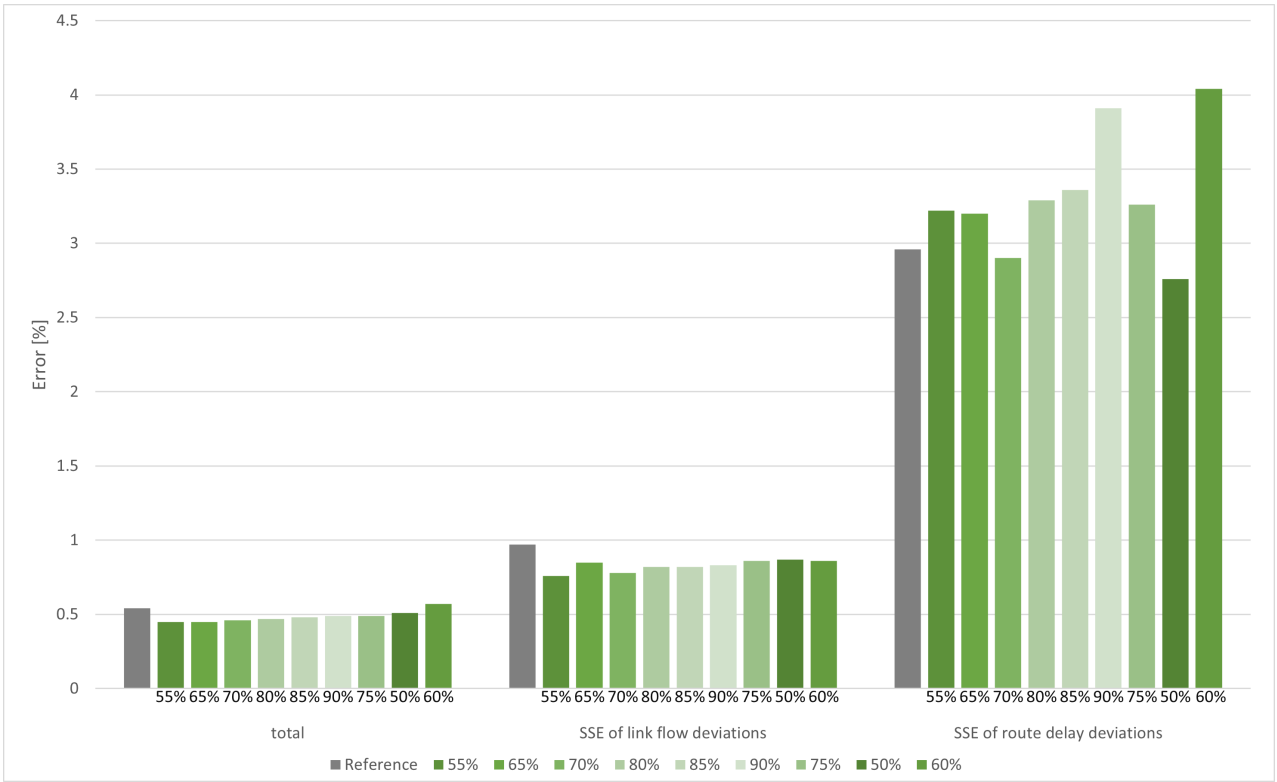
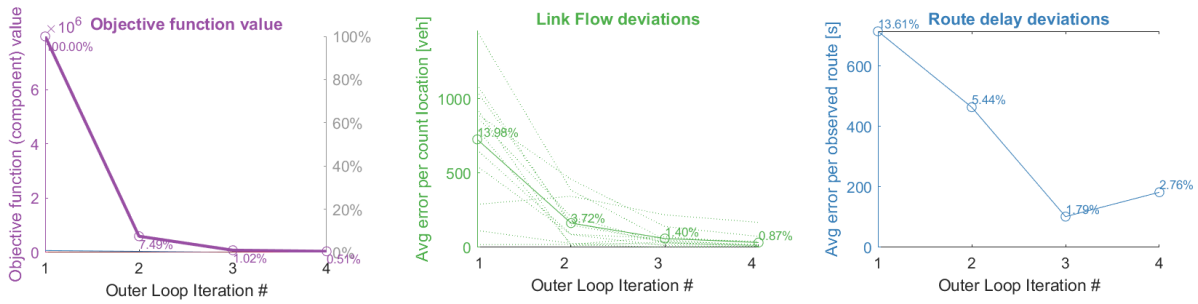
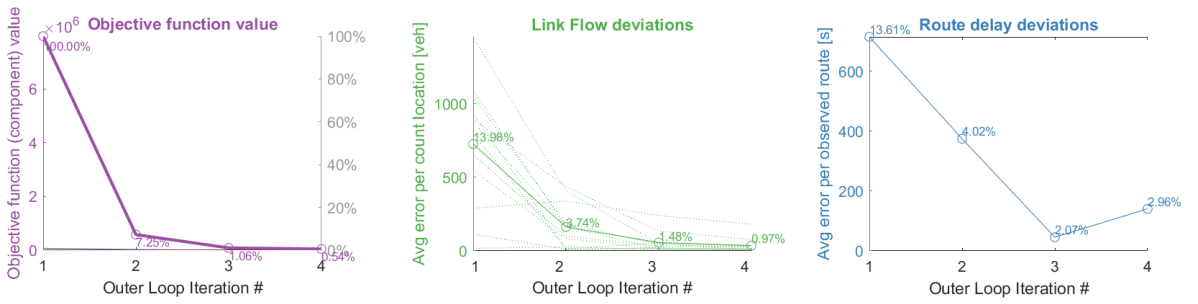


Figure 12: Accuracy results - Sioux Falls - Method 1

A particularity of the 50% run, depicted in Figure 13a, is the first iteration. As is shown, the route delay deviations after the first upper level iteration are substantially higher than in the reference (see Figure 13b). On the other hand, the objective function value and the link flow deviations do not diverge too substantially. A possible reason for this behavior is that, during the first iteration, more OD-pairs are left out that have a substantial impact on the route delay deviations. And then, in subsequent iterations these left out OD-pairs are instead considered.



(a) Reduced problem size - Convergence graphs



(b) Reference - Convergence graphs

Figure 13: Comparison - Method 1 50% VS Reference (Sioux Falls)

Finally, it must also be mentioned that, contrary to what one would expect, the accuracy of convergence is fully improved for the 50% and 70% variants. This is counter intuitive, as one would generally expect all components to suffer, since less data is being utilized. But this is not the case. The most likely cause for this is the same as stated prior: data inconsistencies are possibly present within the utilized data set, which are left out when less OD-pairs are considered.

6.1.2 Dynamic Method

Important to note, for the second method, is that during the first iteration, s can be set to produce the same result as method 1. But, from the second iteration onward, as the distribution of the gradients change, the number of gradients selected will change as well. Whereas, for the static method, the number of OD-pairs chosen stays the same.

Overall, the second method performed very well, as shown in Figure 14, consistently improving the accuracy of all three components, with only $s = 0.6$ and $s = 0.65$ reducing the accuracy of the route

delay deviations. Furthermore, setting $s = 0.5$ leads to the route delay deviations to be at the lowest value yet (for the reduced problem size). Thus, it would seem that, further than just making the reduction problem-agnostic, this dynamic selection also makes a better selection of the relevant OD-pairs, leading to better convergence overall compared to the static method.

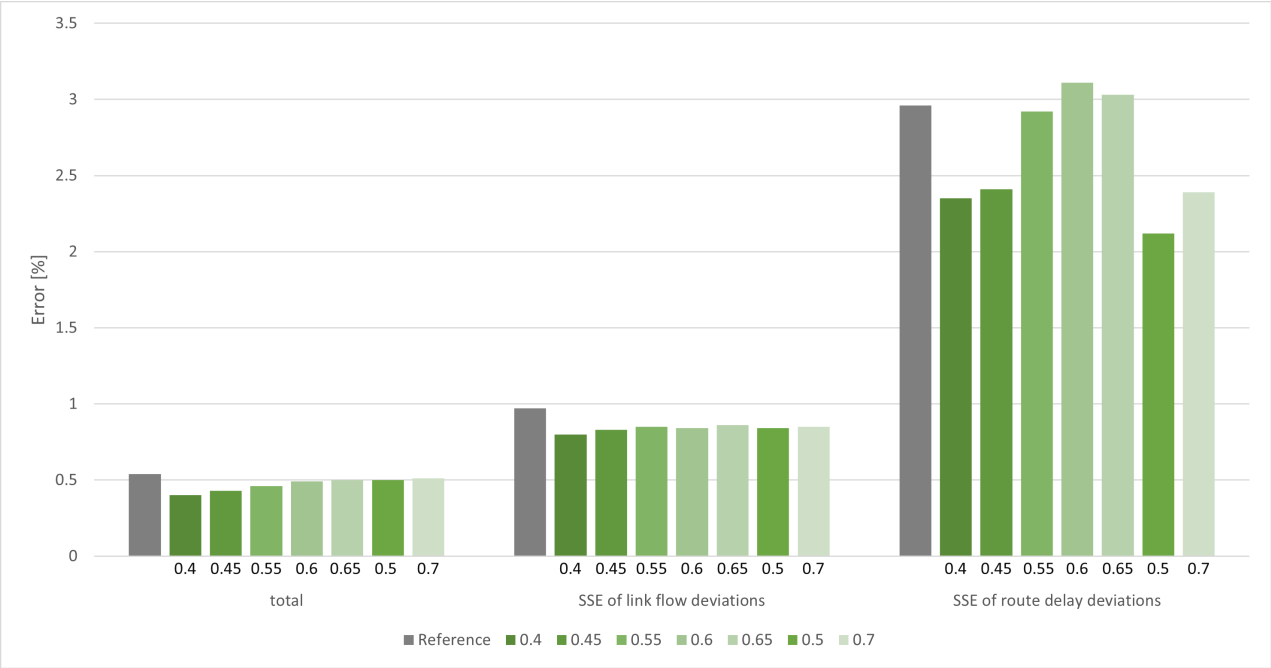


Figure 14: Accuracy results - Sioux Falls - Method 2

6.1.3 Complete First Iteration

When utilizing the complete problem size during the first iteration, the accuracy of the route delay deviations is consistently worsened, as shown in Figure 15. From this it can be seen that choosing to keep the complete problem size during the first iteration is not beneficial. A possibility for this behavior is that after the complete first iteration, the OD-pairs that have a high absolute gradient value do not have a significant impact on the route delay deviations.

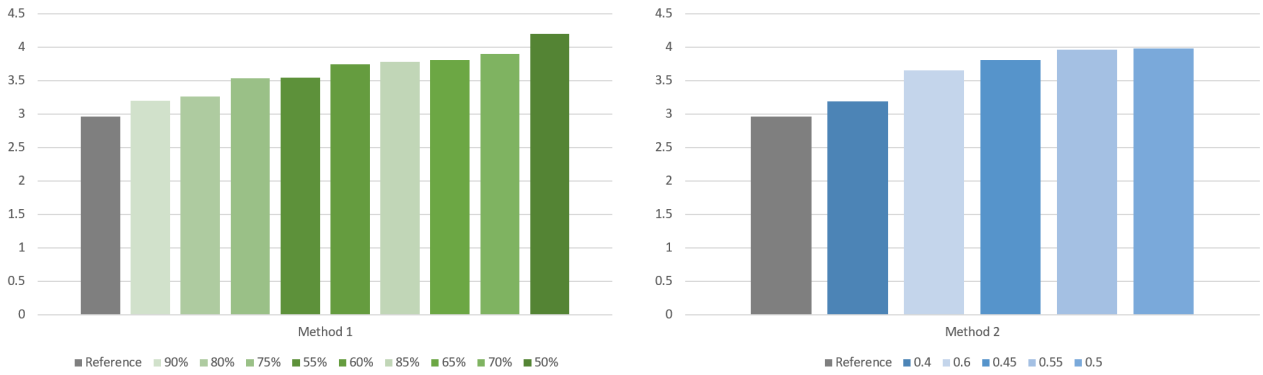


Figure 15: Route delay accuracy - Sioux Falls - Complete first iteration

6.2 Noord-Brabant

Two runs were successfully executed on the Noord-Brabant network, with the problem size reduction parameters shown in Table 3. The parameter choice was based on the results on Sioux Falls, with the best performing parameters of each method being selected. Of these runs, both runs were passably accurate in their convergence at the tenth iteration, as well as stable. Furthermore, all runs were able to reduce the total computation time substantially (up to 25%), as well as the violation of the link constraints.

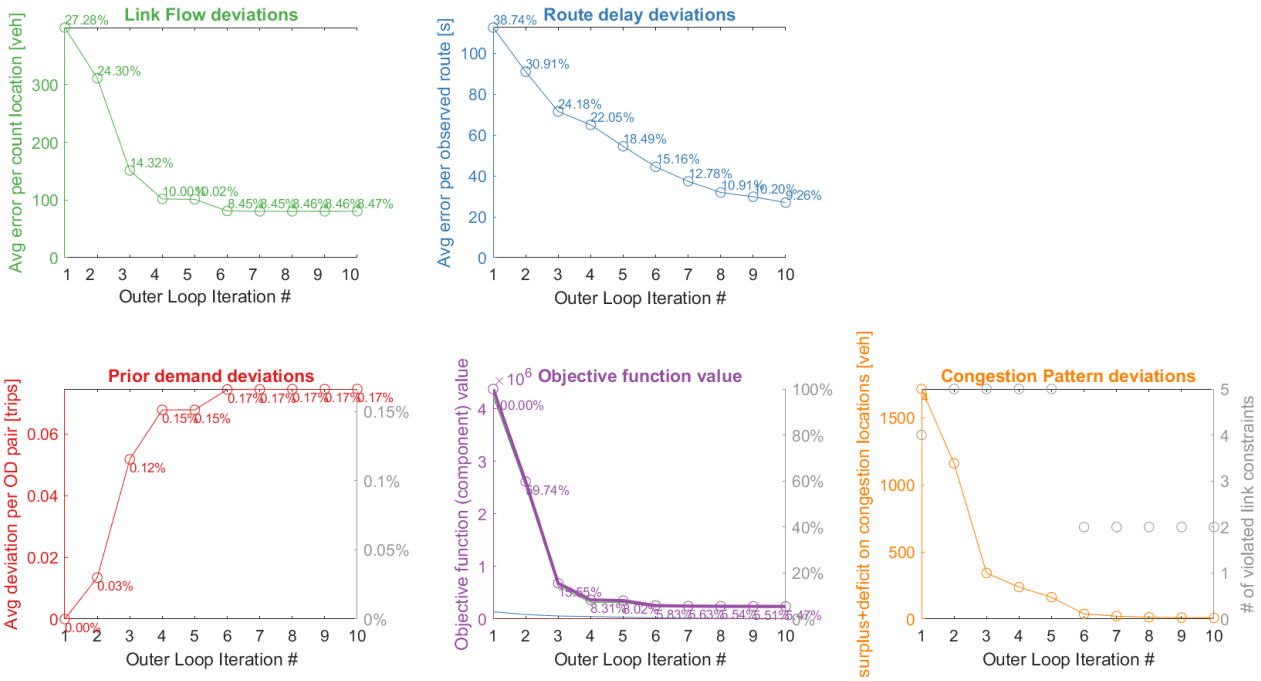
Table 3: Results Overview - Problem size reduction - Noord-Brabant

Method	**CFI**[1]	**Total** [%]	$\mathbf{w_2}$ [%]	$\mathbf{w_3}$ [%]	**# links**	**# vehicles**	**Total time** [hr]
Reference	n/a	4.53	6.39	12.67	4	50	59
1 [50%]	No	5.47	8.47	9.26	2	5	46
2 [$s = 0.5$]	No	5.87	8.36	15.26	2	25	44

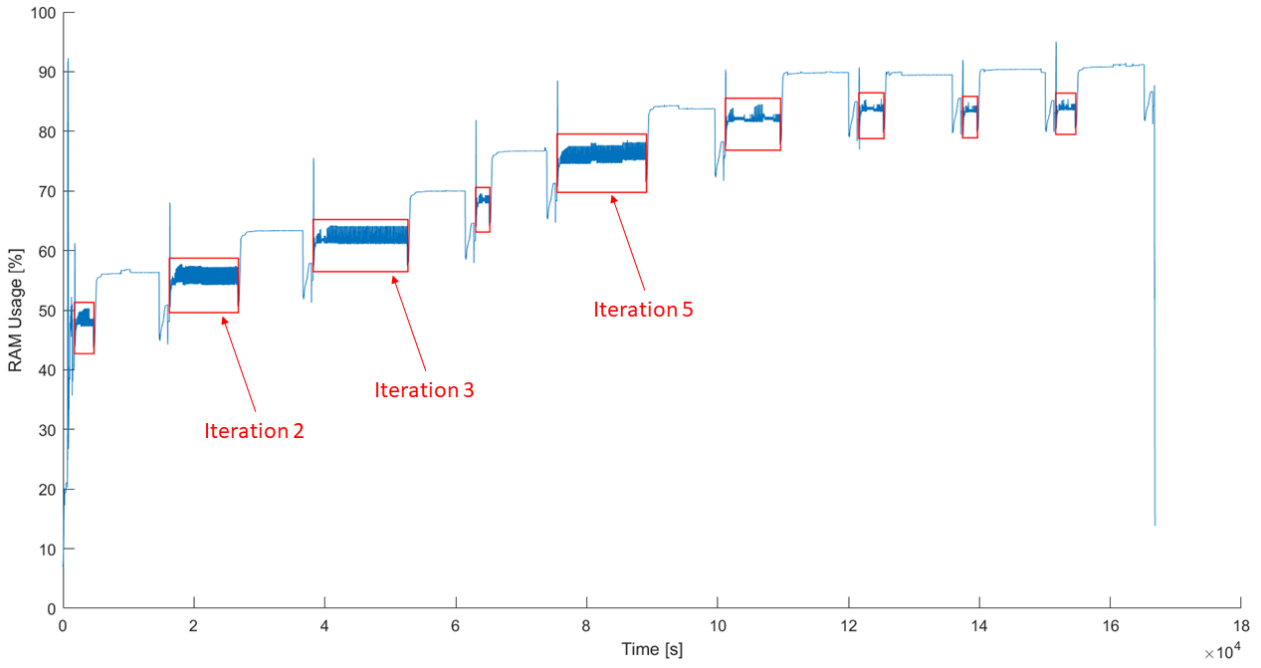
[1] Stands for "Complete first iteration"

As is depicted in Table 3, the first method reduced the total computation time by approximately 22%. Furthermore, the average percentage error for the route delay deviations was decreased. On the other hand, this comes at the cost of the total objective function value and link flow deviations being greater than the reference (by 1% and 2% respectively). A possible reason for this divergence is that the gradient selection chooses OD-pairs which do not have a significant impact on the link flow deviation. Additionally, it must be noted that, for this parameter and method combination, the length of each upper level iteration is still unstable, as shown in Figure 16b. In particular, iterations two (outer loop 2 -> 3), three (outer loop 3 -> 4) and five (outer loop 5 -> 6) stand out. When analyzing the convergence graph at these iterations, (Figure 16a), it can be seen that a significant change in the values of all objective function components occurs. Thus, a possible reason for this behavior it that the solver is searching for a longer time, as the objective function value can be substantially lowered.

Overall, the second, dynamic method performed worse than the static method, only improving upon the total computation time by two hours and the average percentage error of the link flow deviations. But, as is shown in Table 3, this improvement comes at the cost of the other convergence criteria suffering. This means that the method is not problem-agnostic. Additionally, as is shown Figure 17a, though the objective function value is shown to be monotonously decreasing, from outer loop iteration 6 to 7, the route delay deviations are shown to increase instead. This is very likely the cause of the corresponding upper level iteration 6 taking as long as is shown in Figure 17b. Though, it must also be noted that this upper level iteration, similarly to the first method, marks a somewhat significant change in the objective function value and the link deviations. Another possible reason for this worsened performance is the upper bound s is subjected to. During each iteration, it is possible that s was higher than $0.75 * \mu/\sigma$, thus being reduced to this upper bound, leading to a selection of too few OD-pairs.

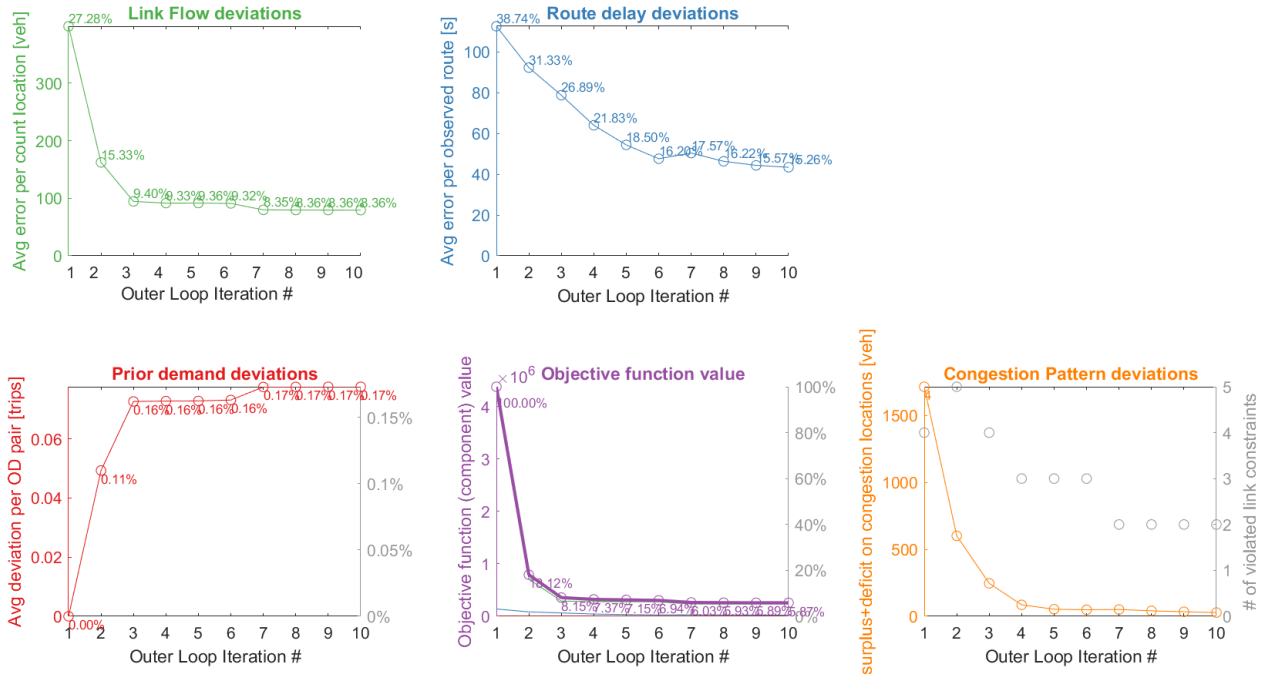


(a) Convergence graphs

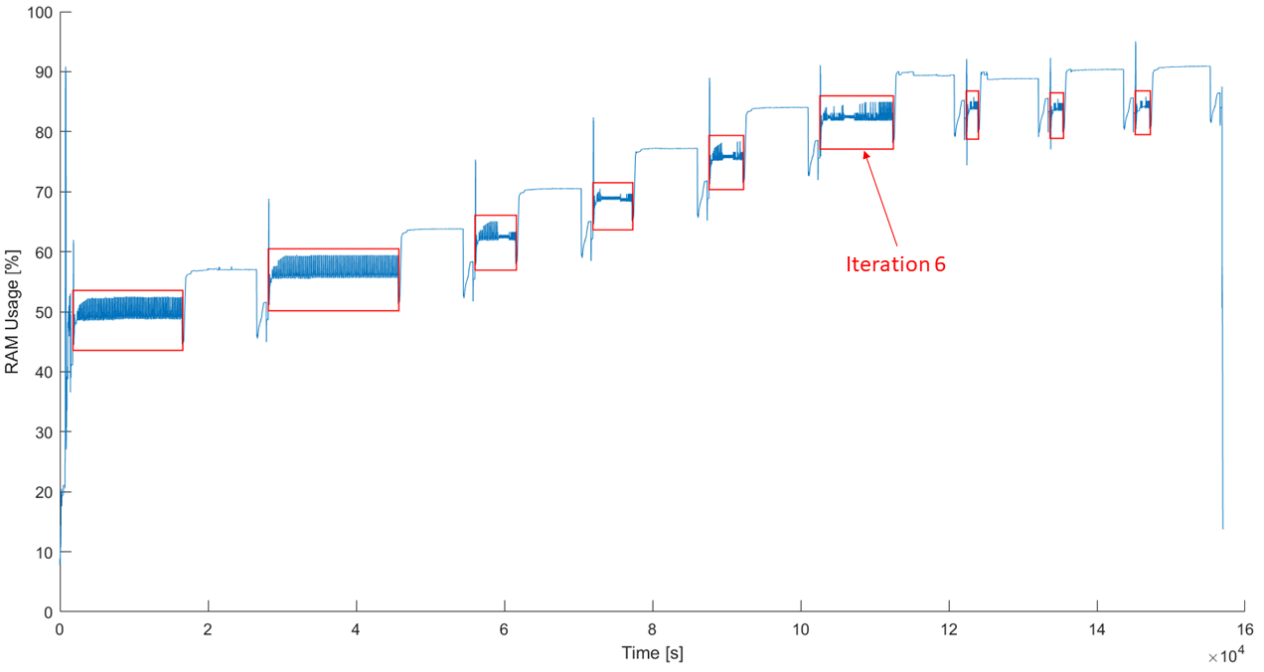


(b) RAM usage and computation time (upper-level iterations are framed in red)

Figure 16: BBMB - Method 1 (50%)



(a) Convergence graphs



(b) RAM usage and computation time (upper-level iterations are framed in red)

Figure 17: BBMB - Method 2 ($s = 0.5$)

7 Conclusions

As two distinct approaches evolved during the execution of this research, with regards to the first research question, two distinct conclusions followed. The first concludes the research with regards to the comparison of the solver-algorithm duos, with the second concluding the research into reducing the problem size. Furthermore, a short sentence is left with regards to the second research question.

7.1 Solver Comparison

During this research, ten solver-algorithm duos, including FMINCON-SQP interfaced through YALMIP, were evaluated on the Sioux Falls transportation network. Of these ten, five were deemed potentially viable and further tested for scalability on the Noord-Brabant transportation network. Overall, the chosen five showed very satisfactory results on Sioux Falls, violating less constraints than the reference implementation, and, in the case of FMINCON-SQP, taking the lead for all criteria. But, the final test for scalability showed a clear distinction between the potential solver-algorithm duos and the current implementation of FMINCON-IPM. FMINCON-IPM was the only duo that was able to successfully complete a run on the Noord-Brabant network, without failing due to running out of memory. For this reason, with regards to the first research question, it can be concluded that the current solver-algorithm-duo is the best for this application.

7.2 Problem Size Reduction

During the second part of this research, problem size reduction was stipulated as a solution to the "out of memory errors" occurring when utilizing solvers other than FMINCON-IPM. This initial idea was also backed by the fact that the absolute gradients per OD-pair showed to be strongly right skewed, with a long tail, meaning that many of the OD-pairs were not influential. From this initial stipulation, two methods were devised to determine which OD-pairs should be kept, a static and a dynamic method. Both methods were then tested on the Nood-Brabant network, in two configurations. Of the two methods, the static method showed better performance overall, by showing a similar quality of convergence compared to the reference, and reducing the total computation time by 22%. To conclude, though no evidence could be gathered with the goal of testing other solvers in mind, the problem size reduction proved to be effective, and thus a viable implementation going forward.

7.3 Is the cause of these issues the solver utilized or the optimization problem?

With regards to the second research question, no direct insight has been gained, as the original solver was the only one that successfully ran under the original configuration.

8 Discussions

8.1 Solver Comparison

Although the results conclude that the current implementation (FMINCON-IPM) is the most suitable for the application, this conclusion may strictly be tied to the way the problem is implemented and interfaced from within MATLAB. FMINCON is a solver native to the MATLAB toolbox, and thus is most likely highly optimized in its direct integration through the MATLAB interface, and memory management. And, though the other languages support being interfaced through MATLAB natively, that may not the way they are generally called. Due to this, it is unclear if the bottleneck lies within the conversion of the program from MATLAB to a form usable by the solvers, or within the solver itself. Though, for some of the solvers, KNITRO, for example, there seems to be precedents of memory issues, as Artelys (2021a) would suggest. A singular case where the unsuitability is clear, though, is FMINCON-SQP, as it is subject to the same optimization as the reference implementation. Thus, in this case, there is no discussion worth pursuing.

Another point of importance within this research is the usage of YALMIP and the possibility of it being a core bottleneck for the applicability of the tested solvers. Beyond just the high requirement of system memory, it could be the case that YALMIP falsely converts the problem before supplying it to the solver. This could be a possible reason for the behavior shown by GUROBI, for example. Though, for GUROBI, this issue of non-dependability is only present when the random "Seed" parameter is altered. On the other hand, this could be the cause for the mediocre performance of the other solvers interfaced through YALMIP, such as QUADPROG and OSQP.

A final point is the fact that, though the solvers were unable to be applied to the Noord-Brabant network, and were thus deemed unsuitable, some of them, notably GUROBI and FMINCON-SQP showed very promising results on Sioux Falls. Thus, it could be stated that the algorithms themselves actually perform better and are more suitable to the problem than the reference. If the testing machine were a more performing one, with larger amounts of system memory, the results could lead to these algorithms being deemed more suitable. But it must be mentioned that the Noord-Brabant network is only medium-sized, whereas large-sized networks have up to 9 times as many origins and a corresponding travel demand matrix that is up to 81 times as large.

8.2 Problem Size Reduction

For the dynamic method, it is shown that it performs much worse on the Noord-Brabant network than the static method, with a possible reason being the upper bound of s. Due to time constraints, this upper bound was scaled with an arbitrary value (0.75), which produced satisfactory results on Sioux Falls. But this upper bound adds a new layer of uncertainty. As this arbitrary scalar can also be changed and is possibly not applicable to all networks. One could consider leaving it out fully and simply keeping s as the only parameter that can be altered. But, as the desire is to always reduce the problem size, this is no longer a viable option. Furthermore, the aim of this method was to make the problem size reduction problem-agnostic, meaning that, if good results are gathered

on Sioux Falls with this configuration, good results should be gathered on any other network with the same parameter configuration.

Another point of discussion is the fact that this is solely possible because the full multi source matrix calibration is an iterative approach, the OD-pairs chosen change with each iteration, and the gradients are strongly right skewed. Fortunately, in the testing, both networks had gradients that shared the same properties. Thus, proving the approach taken here functions as intended. But, there is the possibility that some the properties are missing, which could render the approach less reliable. If, for example the gradients were more uniformly distributed, this approach may converge, as the chosen OD-pairs would change with every iteration, but the number of iterations may increase substantially. Though, considering the characteristics of a network, where there is a high density of origins/destinations, and a comparatively low number of count locations, not many OD-pairs should have a substantial influence. Thus, still leading to a strongly right-skewed distribution of absolute gradients.

Finally, though it is shown in section 6.2 that the total computation time can be reliably reduced, it can also be seen that, for method 1 at 50%, the duration of the upper level iterations does not stabilize (see Figure 16b). A possible reason stipulated for this is the large decrease in the objective function value and its components. But, as stated within section 1.1.4, this problem only occurs when the constraints are introduced. Thus, the initial problem has not been fully addressed.

9 Recommendations

In the following, recommendations for future research are described, in order of relevance by view of the author.

Firstly, as stated prior, a bottleneck for the utilization of other solvers may be MATLAB, and not the solvers themselves. For this reason, the optimization problem itself could be implemented in another programming language, such as AMPL, which was specifically designed for the solving of optimization problems (AMPL, 2021). Additionally, a language such as AMPL provides other alternative solvers, that were not tested in this research. Though, this advice does not only suit AMPL, many of the solver tested, such as GUROBI, KNITRO and SNOPT are callable through C++ as well (Artelys, 2021b; Gill et al., 2021; Gurobi Optimization, 2022a).

Secondly, with the current MATLAB implementation, one could try the solvers with the reduced problem size and observe if they are capable of running successfully on the Noord-Brabant network without running out of memory.

Thirdly, it is mentioned that there may be data points, for which the components of the objective function are conflicting (section 6.1.1). In other words, where, due to observation discrepancies, a positive link flow deviation gradient is paired with a negative route delay deviation gradient. To remedy these conflicts, an analysis based on the gradients of the components may give insight into the conflicting points, after which the user may resolve the conflict by removing the observation which is least trusted.

Fourthly, would be to do further research into the sensitivity of the parameters utilized. There are some discrepancies between the results on Sioux Falls and Noord-Brabant, particularly with the dynamic method. Thus, the chosen parameter does have an impact on the rate of convergence, as well as the quality of convergence. For this reason, if this method is to be further utilized, the sensitivity of the parameter should be further explored, by also applying the problem sized reduced MSMC on different networks, and seeing if a "universally" optimal value can be found. Or if not, how wide the feasible range of the parameter is.

Finally, specifically for Noord-Brabant. It is shown in Figures 16a and 17a that, after the fifth iteration for the first method and iteration six for the second method, there are no more changes in the link flow deviations, i.e., this component has converged to its optimum. On the other hand, the MSMC keeps going, as the route delay deviations are still decreasing. Instead of the continuing with the current method, the OD-pair selection could be constrained to only ones where absolute gradient of the route delay deviations is greater than 0. Alternatively, solely for the selection, one could increase the weight on this component.

10 References

Altman, A., & Gondzio, J. (1998). Regularized symmetric indefinite systems in interior point methods for linear and quadratic optimization. *Optimization Methods and Software, 11-12*, 275–302. https://doi.org/10.1080/10556789908805754

AMPL. (2021). *AMPL* [Accessed: 2022-06-16]. https://ampl.com/products/ampl/

Artelys. (2021a). *Tips and Tricks - Artely Knitro: Memory issues* [Accessed: 2022-06-14]. https://www.artelys.com/docs/knitro/2_userGuide/tips.html#memory-issues

Artelys. (2021b). *User Guide: Getting - Started* [Accessed: 2022-06-14]. https://www.artelys.com/docs/knitro/2_userGuide/gettingStarted.html

Boyd, S., Parikh, N., Chu, E., Peleato, B., & Eckstein, J. (2011). Distributed optimization and statistical learning via the alternating direction method of multipliers. *3*(1), 1–122. https://doi.org/10.1561/2200000016

Brederode, L., Pel, A., Wismans, L., de Romph, E., & Hoogendoorn, S. (2018). Static traffic assignment with queuing: Model properties and applications. *Transportmetrica A: Transport Science*, 1–36. https://doi.org/10.1080/23249935.2018.1453561

Brederode, L., Wismans, L., Rijksen, B., & Hoogendoorn, S. (2020). Travel demand matrix estimation for strategic road traffic assignment models with strict capacity constraints. https://doi.org/10.13140/RG.2.2.35478.98886

Byrd, R. H., Hribar, M. E., & Nocedal, J. (1999). An interior point algorithm for large-scale nonlinear programming. *SIAM Journal on Optimization*, *9*(4), 877–900. https://doi.org/10.1137/S1052623497325107

Byrd, R. H., Nocedal, J., & Waltz, R. A. (2006). Knitro: An integrated package for nonlinear optimization. In G. Di Pillo & M. Roma (Eds.), *Large-scale nonlinear optimization* (pp. 35–59). Springer US. https://doi.org/10.1007/0-387-30065-1_4

Djukic, T., Flötteröd, G., van Lint, H., & Hoogendoorn, S. (2012). Efficient real time od matrix estimation based on principal component analysis. *2012 15th International IEEE Conference on Intelligent Transportation Systems*, 115–121. https://doi.org/10.1109/ITSC.2012.6338720

Gill, P. E., Wong, E., Murray, W., & Saunders, M. A. (2021). *User's guide for SNOPT version 7.7: Software for large-scale nonlinear programming*. https://ccom.ucsd.edu/~optimizers/static/pdfs/sndoc7.pdf

Gurobi Optimization. (2022a). *Choosing the right algorithm* [Accessed: 2022-03-16]. https://www.gurobi.com/documentation/9.5/refman/choosing_the_right_algorit.html#:~:text=Gurobi%5C%20Optimizer%5C%20provides%5C%20two%5C%20main,is%5C%20also%5C%20more%5C%20numerically%5C%20sensitive.

Gurobi Optimization. (2022b). *Method* [Accessed: 2022-03-30]. https://www.gurobi.com/documentation/9.5/refman/method.html

Gurobi Optimization. (2022c). *Seed* [Accessed: 2022-06-29]. https://www.gurobi.com/documentation/9.5/refman/seed.html#parameter:Seed

Hamerslag, R., & Immers, B. H. (1988). Estimation of trip matrices: Shortcomings and possibilities for improvement. *Transportation Research Record*, 27–39. https://onlinepubs.trb.org/Onlinepubs/trr/1988/1203/1203-003.pdf

Leibovici, C. (2013). *Ideas for reducing problem dimension in optimization problem* [Accessed: 2022-06-14]. https://math.stackexchange.com/q/620296. https://math.stackexchange.com/q/620296

Lofberg, J. (2004). Yalmip : A toolbox for modeling and optimization in matlab. *2004 IEEE International Conference on Robotics and Automation (IEEE Cat. No.04CH37508)*, 284–289. https://doi.org/10.1109/CACSD.2004.1393890

MathWorks. (2022a). *Choosing the Algorithm* [Accessed: 2022-03-16]. https://www.mathworks.com/help/optim/ug/choosing-the-algorithm.html

MathWorks. (2022b). *Quadratic Programming Algorithms* [Accessed: 2022-03-29]. https://www.mathworks.com/help/optim/ug/quadratic-programming-algorithms.html#bsqspm_

MOSEK. (2022). *Overview* [Accessed: 2022-03-29]. https://docs.mosek.com/latest/intro/overview.html

Næss, P., Andersen, J., Nicolaisen, M., & Strand, A. (2014). Transport modelling in the context of the 'predict and provide' paradigm. *European Journal of Transport and Infrastructure Research*, *14*, 102–121. https://doi.org/10.18757/ejtir.2014.14.2.3020

Nocedal, J., & Wright, S. J. (2006). *Numerical optimization*. Springer New York. https://doi.org/10.1007/978-0-387-40065-5

Prakash, A. A., Seshadri, R., Antoniou, C., Pereira, F. C., & Ben-Akiva, M. E. (2017). Reducing the dimension of online calibration in dynamic traffic assignment systems. *Transportation Research Record*, *2667*(1), 96–107. https://doi.org/10.3141/2667-10

Qurashi, M., Ma, T., Chaniotakis, E., & Antoniou, C. (2020). Pc–spsa: Employing dimensionality reduction to limit spsa search noise in dta model calibration. *IEEE Transactions on Intelligent Transportation Systems*, *21*(4), 1635–1645. https://doi.org/10.1109/TITS.2019.2915273

Saadi, I., Mustafa, A., Teller, J., & Cools, M. (2017). A bi-level random forest based approach for estimating o-d matrices: Preliminary results from the belgium national household travel survey [World Conference on Transport Research - WCTR 2016 Shanghai. 10-15 July 2016]. *Transportation Research Procedia*, *25*, 2566–2573. https://doi.org/https://doi.org/10.1016/j.trpro.2017.05.301

Sahinidis, N. V. (2017). *BARON 22.2.12: Global Optimization of Mixed-Integer Nonlinear Programs,* User's Manual. http://www.minlp.com/downloads/docs/baron%5C%20manual.pdf

Sahinidis, N. V. (1996). BARON: A general purpose global optimization software package. *Journal of Global Optimization*, *8*(2), 201–205.

Saw, K., Katti, B. K., & Joshi, G. (2015). Literature review of traffic assignment: Static and dynamic. *International Journal of Transportation Engineering*, *2*(4), 339–347. https://doi.org/10.22119/ijte.2015.10447

Stellato, B., Banjac, G., Goulart, P., Bemporad, A., & Boyd, S. (2020). OSQP: An operator splitting solver for quadratic programs. *Mathematical Programming Computation*, *12*(4), 637–672. https://doi.org/10.1007/s12532-020-00179-2

van de Panne, C., & Whinston, A. (1969). The symmetric formulation of the simplex method for quadratic programming. *Econometrica*, *37*(3), 507–527. http://www.jstor.org/stable/1912797

Waltz, R. A., Morales, J. L., Nocedal, J., & Orban, D. (2006). An interior algorithm for nonlinear optimization that combines line search and trust region steps. *Mathematical Programming*, *107*(3), 391–408. https://doi.org/10.1007/s10107-004-0560-5

Willumsen, L. (1978). *Estimation of an o-d matrix from traffic counts ? a review.* (Working Paper Working Paper 99) [Copyright of the Institute of Transport Studies, University Of Leeds]. ARRAY(0x5599bd2a0028). Leeds, UK, Institute of Transport Studies, University of Leeds. https://eprints.whiterose.ac.uk/2415/

Wolfe, P. (1959). The simplex method for quadratic programming. *Econometrica*, *27*, 170.

A Solver Comparison - Sioux Falls - Additional Results

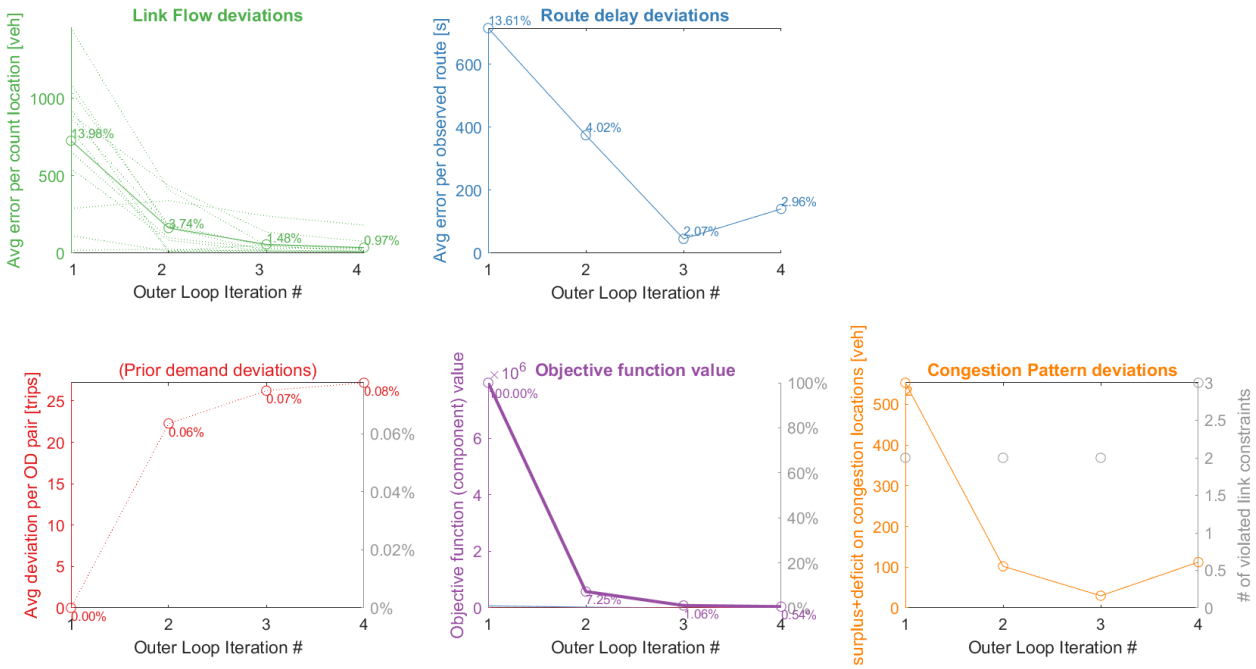


Figure 18: Sioux Falls - FMINCON-IPM convergence graphs

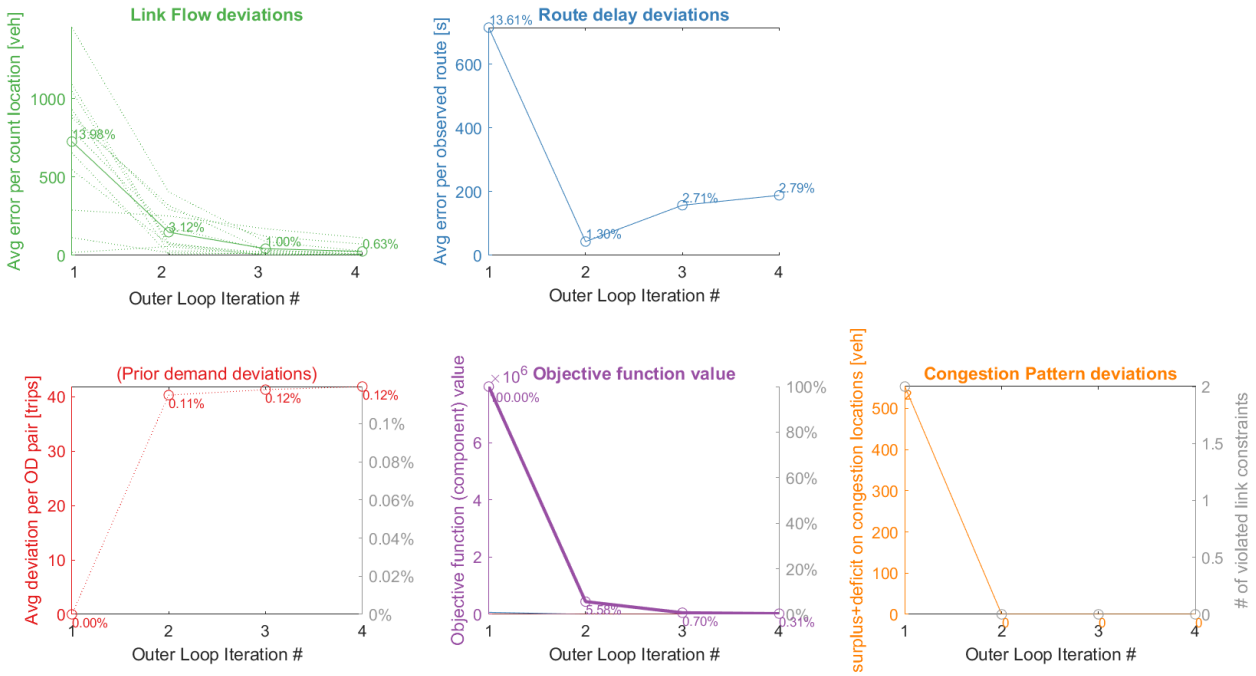


Figure 19: Sioux Falls - FMINCON-SQP convergence graphs

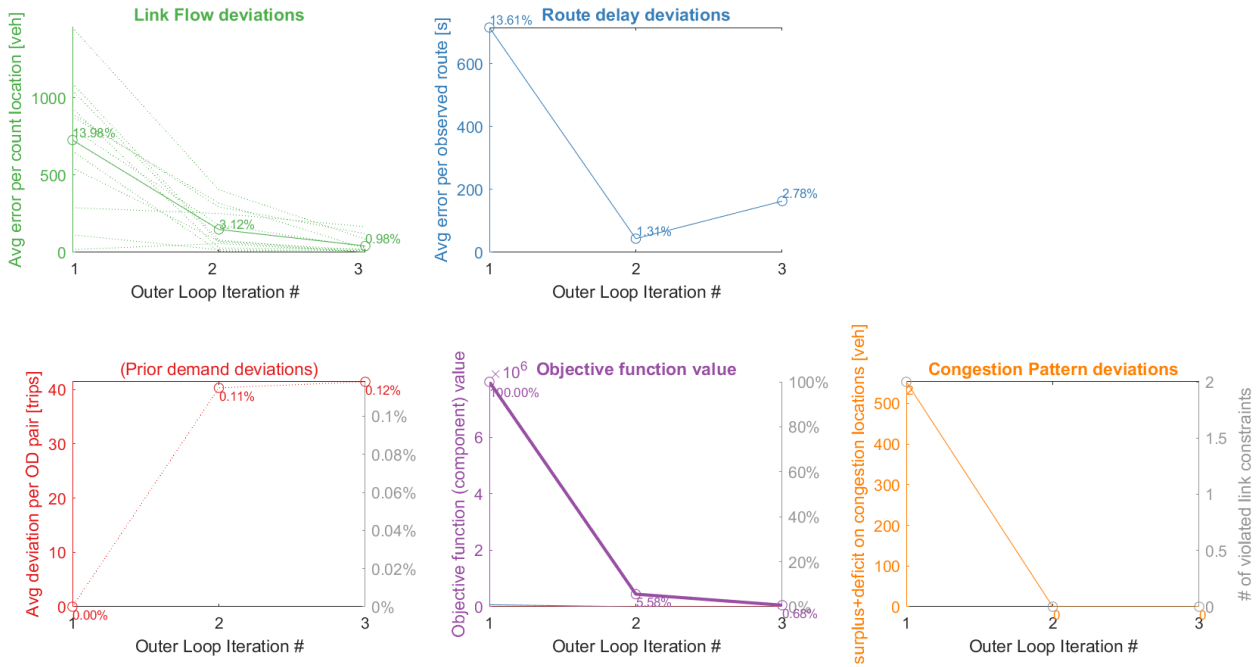


Figure 20: Sioux Falls - FMINCON-SQP + Yalmip convergence graphs

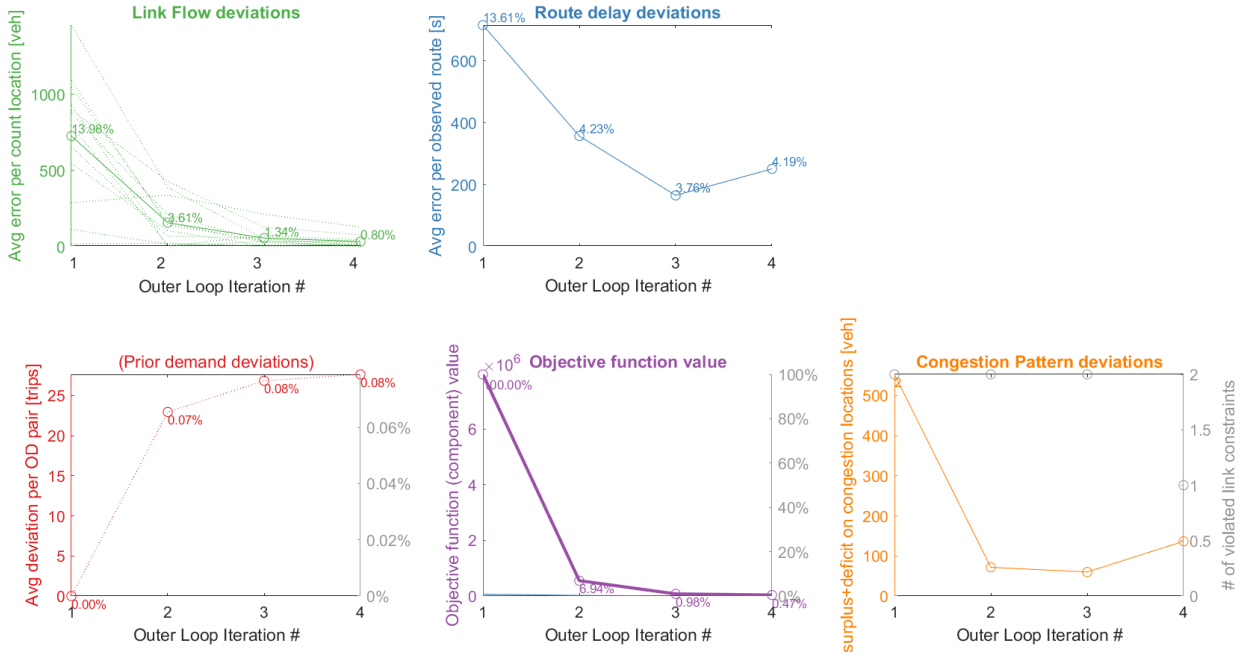


Figure 21: Sioux Falls - KNITRO-Byrd convergence graphs

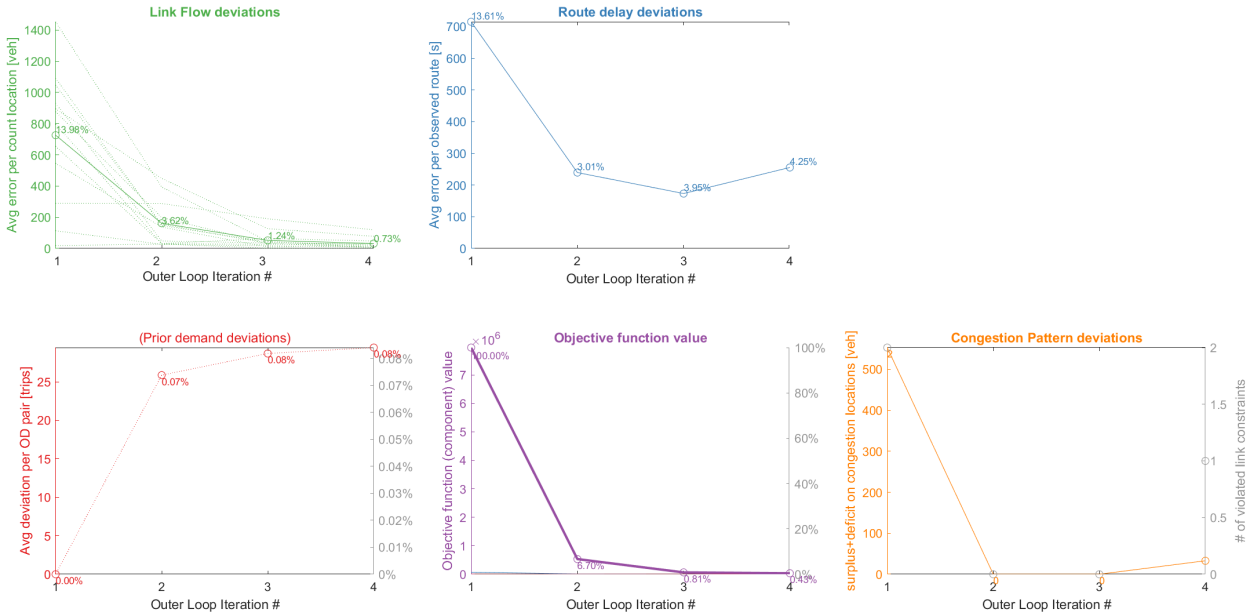


Figure 22: Sioux Falls - KNITRO-Waltz convergence graphs

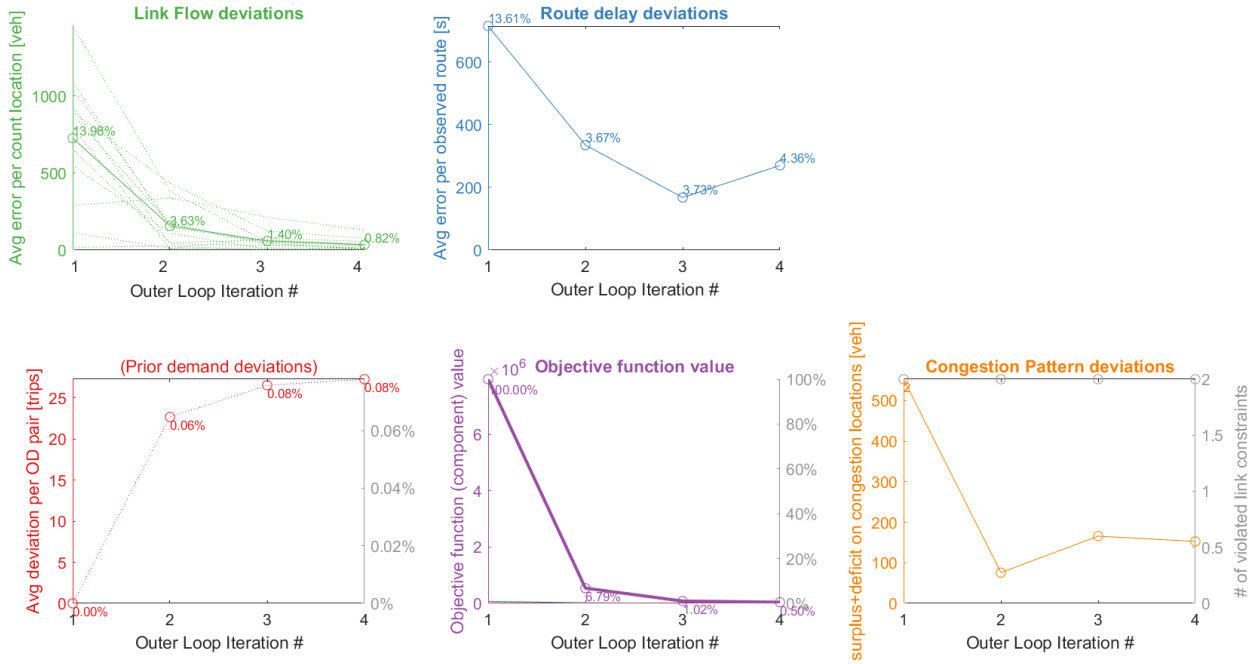


Figure 23: Sioux Falls - SNOPT convergence graphs

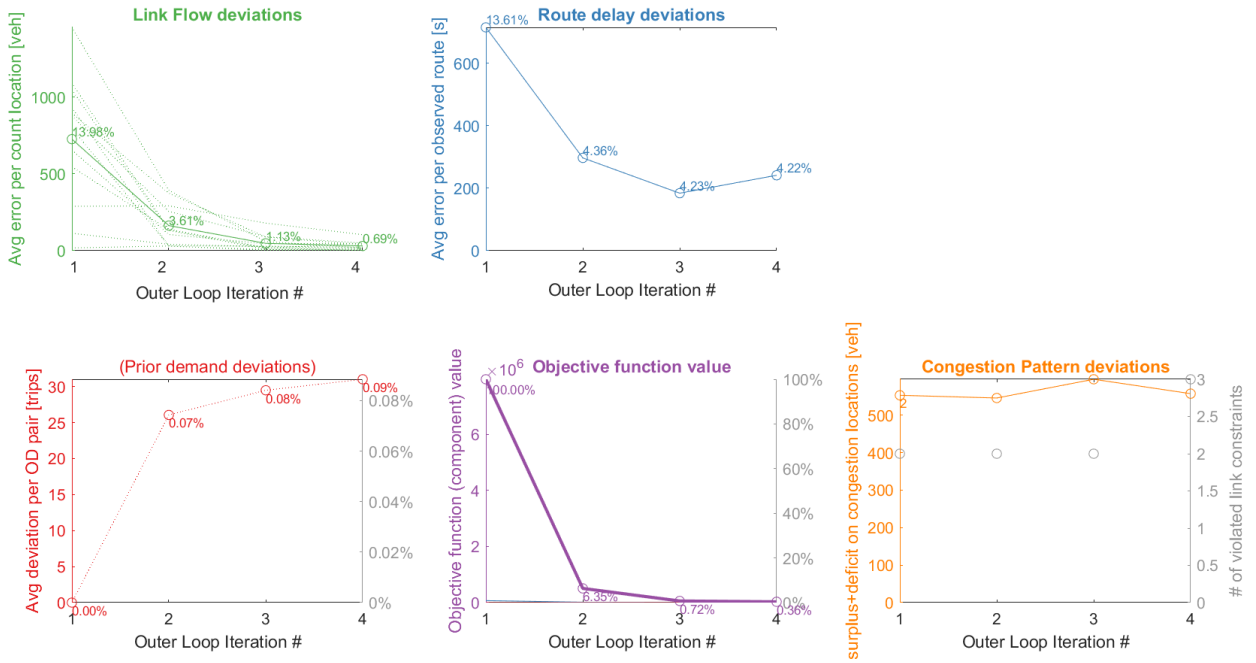


Figure 24: Sioux Falls - BARON convergence graphs

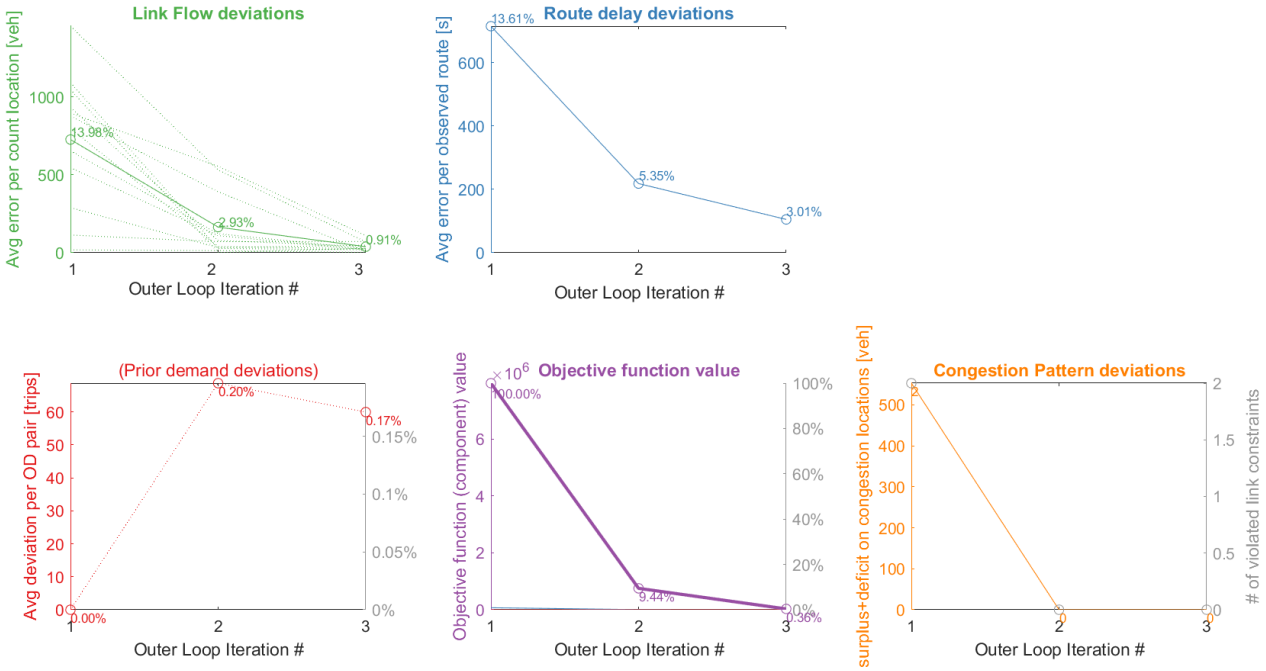


Figure 25: Sioux Falls - GUROBI convergence graphs

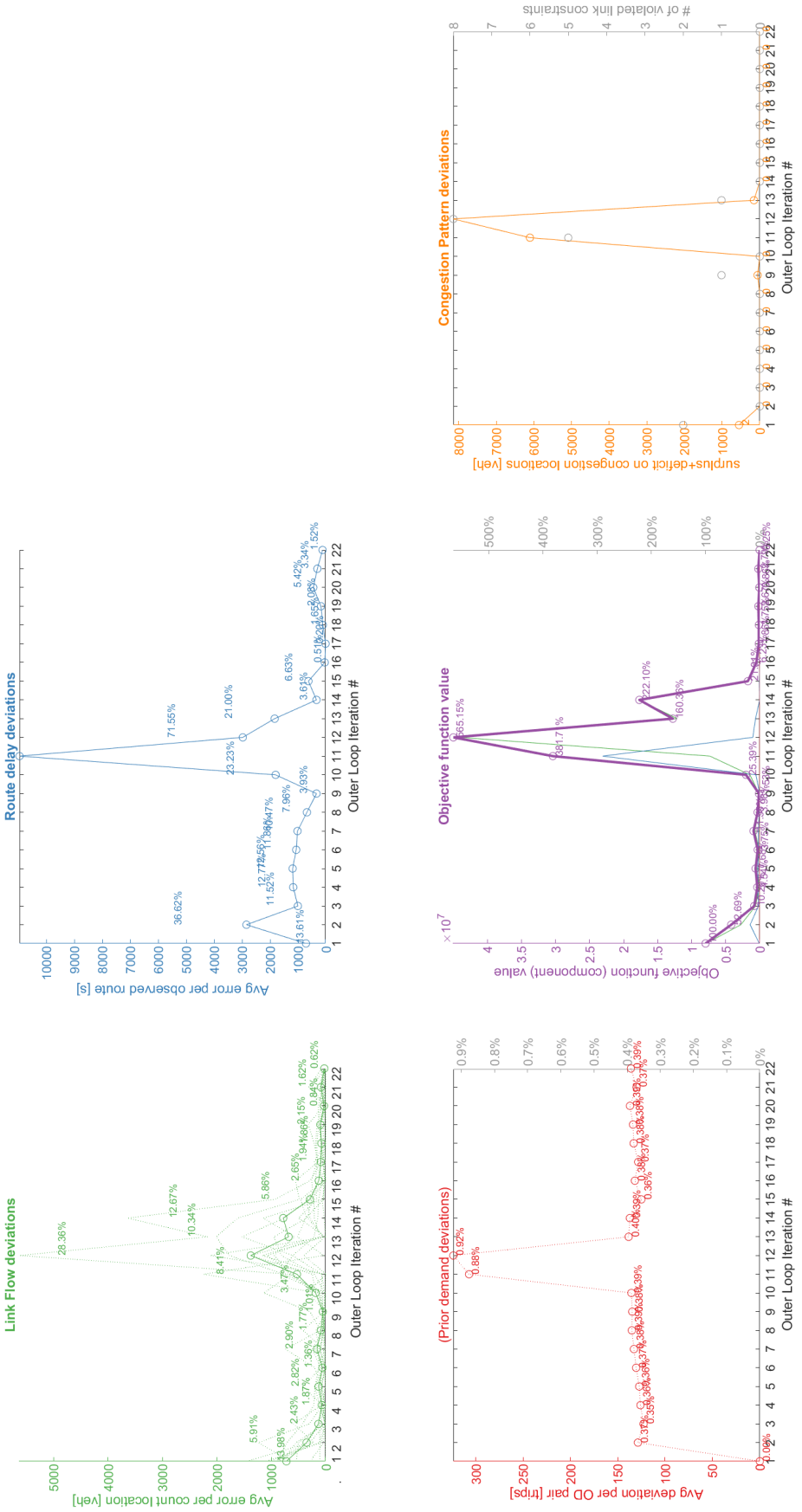


Figure 26: Sioux Falls - QUADPROG convergence graphs

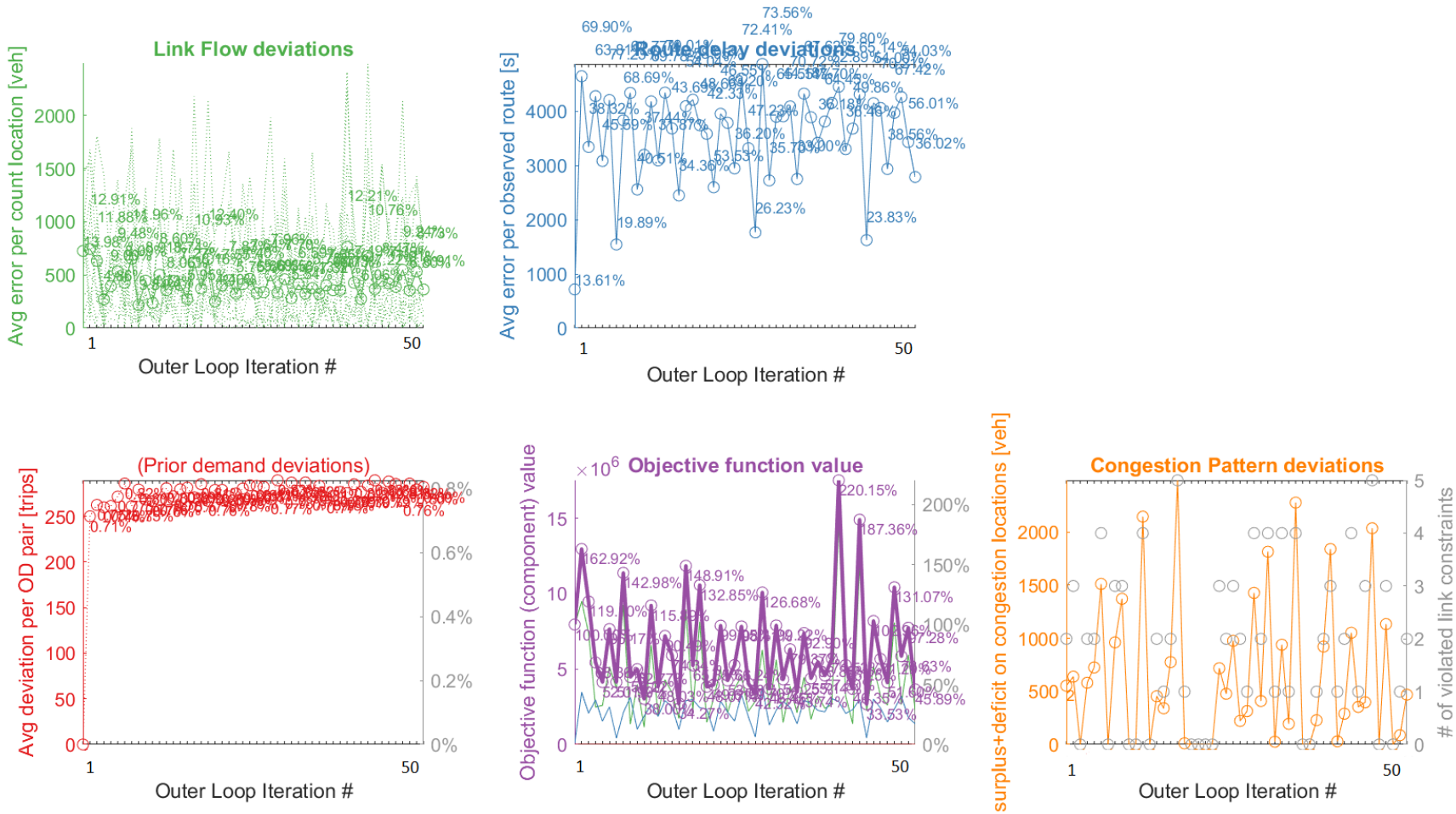


Figure 27: Sioux Falls - OSQP convergence graphs

B Absolute Gradients - Noord-Brabant

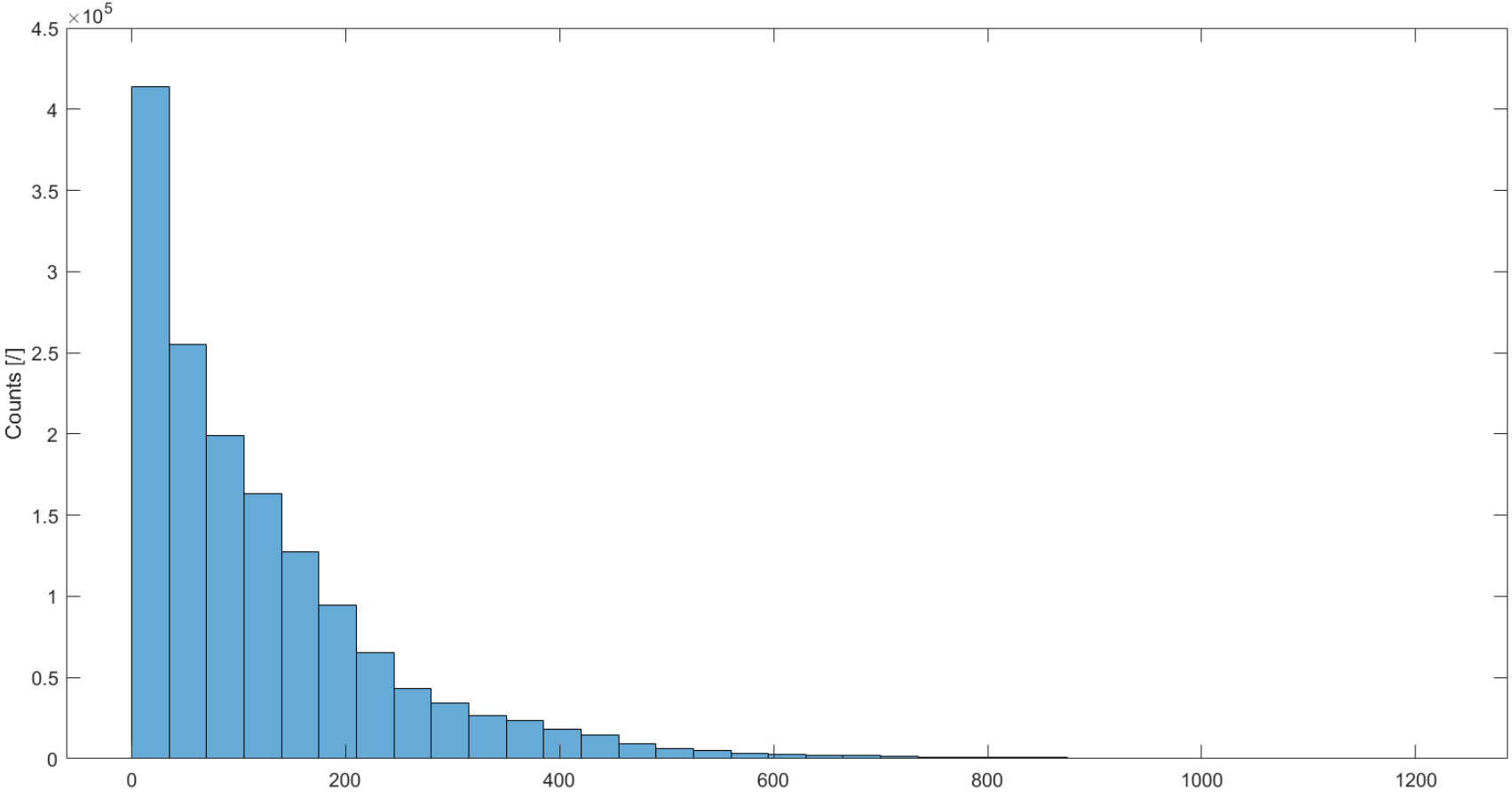


Figure 28: Histogram of the absolute gradient values per OD-pair - Noord-Brabant