Iterate averaging methods for solving non-linear programming problems

Applied to a transportation network equilibrium problem

R.T.J. Hiele
Delft, 2003
Delft University of Technology
Department of Civil Engineering and Geosciences
Transportation Planning and Traffic Engineering section

Preface

At the Delft University of Technology (DUT), faculty of Applied Mathematics, every student has to do a practical training of eight weeks (forty working days) during the study of five years.

The essence of this practical training is that the students of mathematics become familiar with:
1. A realistic working place for mathematicians.
2. Working at a project in a surrounding of engineers (where mathematics is used, but it is not necessary that mathematics is the most important topic) and to actively participate in different, real-life projects.
3. Working at another faculty or at a company near Delft.

After finishing above described practical training my tasks are to:
1. Write two reports, one for the resource team with whom I worked and another for the supervisor of the practical training. This report will be destined for the resource team I work with.
2. Present my experiences at a public meeting at the DUT.

My practical training takes place at one of the faculties of the DUT, Civil Engineering and Geosciences, Transportation Planning and Traffic Engineering section. I worked with the team of Prof. Dr. Ir. P.H.L. Bovy. My direct supervisor was Dr. M.C.J. Bliemer and I worked with two Ph.D. students Dusica Joksimovic and Dirk van Amelsfort. The department consists of forty-four persons. During my practical training I attended the staff meeting of the department (thirty-five persons were present) and I had a seminar on the paper 'A mathematical model and descent algorithm for bi-level traffic management' by Patriksson and Rockafeller (2002).

I mainly worked at the project of Dynamic Traffic Assignment (DTA) model and 'Road Pricing'. I fulfilled mathematical analyses on formulation in the existing models and I looked for improvements of the existing algorithms that are used for solving these models. For a wider dissemination of my results, the language used for the report is English.

Some methods for solving NLP problems are discussed in this report. Three of them are already found in literature and the remaining methods are new. The methods are compared with each other and some recommendations for further research are given.

I want to thank the resource team of Prof. Dr. Ir. P.H.L. Bovy for the pleasant collaboration. I want to thank Dr. M.C.J. Bliemer and Prof. Dr. Ir. P.H.L. Bovy for the warm welcome, the correction of the report and giving good comment for how to write a report. Further I want to thank Dr. M.C.J. Bliemer and Dirk van Amelsfort for the warm welcome and the explanation about the research. Finally, I want to thank Dusica Joksimovic for all the time she had for me (for the warm welcome, the explanation about the research, the correction of the report and giving good comment for how to write a report). During this practical training I learned much. Such like programming, the mathematics of iterate averaging methods and working with people of different countries.

Delft, January 2003

R.T.J. Hiele

Summary

Traffic congestion is an unresolved problem and it has effects not only to the transportation system, but also on other aspects of life (economic, spatial and social). The idea is that 'Road Pricing' can be used to solve this problem. There is a need for an appropriate tool for predicting the effects of 'Road Pricing'. Such a tool could be a traffic assignment model. Traffic is by nature dynamic and hence only dynamic models can describe traffic process adequately. It appears, however, that iterate averaging methods have not yet been applied to transportation network problems. In this research iterate averaging methods are investigated and also the possibility of applying these methods in transportation network problems.

Recently the Polyak method was introduced, which is supposed to have better convergence qualities than the method that is normally used, the Method of Successive Averages (MSA).

The three topics of this research of iterate averaging methods are:

- To find out how the Polyak method works, after which the Polyak method is implemented.
- To find out if the Polyak method indeed converges faster than MSA.
- To find out if there exist alternative methods that are faster in convergence than the Polyak method and MSA.

To find answers on these three topics the literature was studied. By reading the nature of the Polyak method is found out. When the Polyak method is understood, the method is implemented (and also MSA is implemented). That was needed for analysing the convergence of the methods. After the Polyak method is implemented research for alternative methods is done. Finally, all the described methods are compared and illustrations are given.

In order to satisfy the increasing demand for more accurate model outcomes and to be able to compute the effects of different traffic policies, new and improved traffic assignment models are needed. While Static Traffic Assignment models may provide basic insights, only dynamic assignment models are able capture the true dynamic nature of traffic and therefore provide the analyst with more accurate forecast. An iterative process is needed to solve the Dynamic Traffic Assignment (DTA) model. This is because network conditions may change after performing network loading. At all the iterations, the path flows are updated by combining the results from the current iteration with the previous iteration.

The 'classical' (e.g., derivative-based) fixed-point solution methods are often inappropriate for some problems. In such cases, the fixed-points are usually computed using one of the iterate averaging methods introduced by Robbins and Monro [3.1]. MSA, introduced by Sheffi and Powell [3.2], is probably the best-known and most widely-used instance of iterate averaging methods. In iterate averaging methods estimates for the fixed-point are found. These estimates are called design points. MSA computes each new design point by adding a part of the observation evaluated in the previous design point with a part of the previous design point.

MSA has the advantages of avoiding (potentially expensive) step size calculations, working directly with map outputs without requiring derivative calculations or other transformations, and being able to handle 'noisy' map evaluations (where the evaluation returns a value affected by a zero-mean disturbance). Other advantages of MSA are that it is simple to understand and that it is simple to implement. In many cases, however, the method's empirically observed convergence properties are disappointing: while it exhibits generally effective

performance in the initial iterations, this is followed by a pronounced 'tail' effect, resulting in overall slow convergence.

Approximately ten years ago, B.T. Polyak and J.A. Bather proposed two relatively minor modifications of iterate averaging methods which were rigorously shown to produce fixed-point estimates with asymptotically optimal properties.

The Polyak method is a two-pass method. The first pass resembles MSA except that the step sizes are larger; this allows the algorithm to explore the solution space more aggressively but leads to greater variability in the outputs. The second pass is carried out offline (i.e., without influencing the first pass); it calculates an average of iterates that are generated by the first pass. The average calculated by the second pass at termination is the fixed-point solution estimate.

A somewhat different approach was proposed by J.A. Bather. Here, the design point is derived from a combination of the average of previous design points with the average of previous evaluation results.

Apart from the Polyak method and the Bather method, alternative methods are proposed. In total eight methods are applied and presented in this report. They were all compared with different stop criterions.

MSA and the Polyak method were compared. To compare these methods and to compare also other methods a traffic problem with three cities and two routes is considered. This traffic problem is solved by using a DTA algorithm. For stopping this DTA algorithm there are different stop criteria.

The stop criterion that is a combination of the route costs and flows is the best stop criterion for stopping the DTA algorithm. This stop criterion is reached after 190 iterations of MSA and after 226 iterations of the Polyak method. The conclusion is that MSA is faster in convergence than the Polyak method for this stop criterion.

The Bather method, the Bliemer method and the Bliemer Moving method were compared. After 118 iterations of the Bather method, after 112 iterations of the Bliemer method and after 40 iterations of the Bliemer Moving method the stop criterion is reached. It can be concluded that the Bather method and the Bliemer method solve the problem in almost the same number of iterations. For different stop criterions the Bliemer Moving method much faster than is the four other methods.

The Bliemer Moving method is the fastest method, but by combining two methods it's possible to get a method that is even faster in convergence than the methods shown before. Therefore, the MSA-Bliemer method, the MSA-Bather method and the Bliemer-Bather method are compared. The fastest method in convergence, for the stop criterion we chose, is the Bliemer-Bather method.

The conclusion is that there are alternative methods that are much faster in convergence than the Method of Successive Averages and the Polyak method. The best alternative methods are the MSA-Bather method and the Bliemer-Bather method.

The recommendation is to use the Bliemer-Bather method for solving Non-Linear Programming (NLP) problems in transportation networks and to do further research how the values of the variables used in the Bliemer-Bather method have to be chosen.

Table of Contents

1. Introduction

Traffic congestion is an unresolved problem and it has effects not only to the transportation system, but also on other aspects of life (economic, spatial and social). The idea is that 'Road Pricing' can be used to solve this problem. There is a need for an appropriate tool for predicting the effects of road pricing. Such a tool could be a traffic assignment model. Traffic is by its nature dynamic and hence only dynamic models can describe a traffic process in a realistic way. It appears, however, that iterate averaging methods have not yet been applied to transportation network problems. In this research iterate averaging methods are investigated and also the possibility of applying these methods in transportation network problems.

The Dynamic Traffic Assignment (DTA) model used in the Traffic department is a complex variational inequality problem [1.1]. The solution of a complex variational inequality problem can be found by solving iteratively a Non-Linear Programming (NLP) problem. Solving a NLP problem is at this moment done by a steepest descent technique, where the step size is calculated very simply, using the Method of Successive Averages (MSA). The convergence of this algorithm is very slow. Recently the Polyak method was introduced, which is supposed to have better convergence qualities than MSA. With a small modification of MSA some researchers have gained an enormous improvement in velocity of calculation. For more detail about MSA and the Polyak method see [1.2].

The prototypical simple iterate averaging method is due to Robbins and Monro [3.1]. Let $\Re$ be the field of reals, and $T(.): \Re \to \Re$ be a map for which a root x^* is to be found (so that $T(x^*) = x^*$). Suppose that we are free to select the points x^k at which to evaluate $T(.)$ during the iterative search for the root – these are called the design points. However, each such evaluation returns a result that is affected by noise: the result is $T(x) = T(x) + \varepsilon$, where ε is a random zero-mean noise vector.

Robbins and Monro proposed the following iterative procedure for choosing the design points (x^{k+1}):

$$x^{k+1} = x^k + \alpha^k \cdot T(x^k), \text{ with } x^0 \in \Re \text{ and k=1,2,3,...,} \qquad (1.1)$$

with the design point x^k, the evaluation $T(x^k)$ and where the sequence α^k the step size is.

It is characteristic of iterate averaging methods such as MSA that the successive design points generated to explore the feasible space are also taken to be the successive estimates of the fixed-point equation solution. This was noted by Frees and Ruppert [5.3], who pointed out the potential advantages of using one method to select the design points, and a different method to estimate the solution. Use of a distinct method for each purpose could allow, on the one hand, a more aggressive exploration of the feasible space and, on the other, a more effective exploitation of the information generated during that exploration in order to estimate a solution.

My tasks are the following:
The DTA model that is used by the Traffic section solves a complex variational inequality problem. The solution of a complex variational inequality problem can be found by solving iteratively a Non Linear Programming (NLP) problem. Solving this NLP problem is at this moment done by a steepest descent technique, where

the step size is calculated very simply, using the Method of Successive Averages (MSA). This algorithm converges very slowly. Recently another technique was proposed. It is called the Polyak method, which is supposed to have better convergence qualities than MSA. For the Polyak method just a little is modified in MSA that will lead to an enormous improvement in computation time. MSA and the Polyak method are implemented as well as improving of the convergence of the algorithm.

The three topics of this research of iterate averaging methods are:

- To find out how the Polyak method works and to implement the Polyak method.
- To find out if the Polyak method convergences faster than the Method of Successive Averages.
- To find out if there exist some alternative methods those are faster in convergence than the Polyak method and MSA.

To find answers on these three topics the relevant literature is read, especially 'Accelerated Averaging Methods for Fixed Point Problems in Transportation Analysis and Planning' by J. Bottom and I. Chabini. By reading these articles the nature of the Polyak method is found out. When the Polyak method is understood, the method is implemented (and also the Method of Successive Averages is implemented) in the software package 'Matlab version 6.0.0.88 release 12'. That was needed for analysing the convergence of the methods. After the Polyak method is implemented, research for alternative methods is done, such as the Bather method, the Bliemer method, the Bliemer Moving method, and combinations of methods described in this report (the MSA-Bather method, the MSA-Bliemer method and the Bliemer-Bather method). Finally, all the previous described methods are compared and illustrations are given.

The research is done in the following way:
Given is the transport network *problem*. Of this problem is made a *model* (a DTA model). To solve this model *methods* are considered. If a method is implemented an *algorithm* is vested.

The structure of this report is as follows. In chapter 2 the DTA model is explained. In this model an iterative stochastic algorithm is used to solve the fixed-point problem. In chapter 3 the mathematical background of iterate averaging methods is discussed. The topics discussed in chapter 3 are: simple iterate averaging methods, convergence of iterate averaging methods, the stop criterion for DTA algorithms and the step size for iterate averaging algorithms. In chapter 4 the iterate averaging method of Robbins and Monro is shown and Sheffi and Powell's MSA is introduced. In chapter 5 the Polyak method and the Bather method are described and numerical examples for both are given. In chapter 6 a research on alternative methods is done and the Bliemer method, the Bliemer Moving method, the MSA-Bather method, the MSA-Bliemer method and the Bliemer-Bather method are introduced. Numerical examples for these methods are also given in this chapter. In chapter 7 the methods are discussed and compared to each other mathematically and computationally. In chapter 8 the conclusion is drawn what the best iterate averaging method is for solving a transportation network problem and recommendations for further research are given. The implementation of the methods, as discussed in above-mentioned chapters, can be found in the appendices.

2. Dynamic Traffic Assignment (DTA) model

To be able to make forecasts about future traffic conditions on transport networks, to compare scenarios of different infrastructure investments, or to estimate effects of traffic management measures, policy analysts rely on tools such as a traffic assignment model. In order to satisfy the increasing demand for more accurate model outcomes and to be able to compute the effects of different traffic policies, new and improved traffic assignment models are needed. While static traffic assignment models may provide basic insights, only dynamic assignment models are able to capture the true dynamic nature of traffic and therefore provide the analyst more accurate forecast. In the recent studies Dynamic Traffic Assignment (DTA) models have gained increasing attention by many researchers. In this chapter one of the DTA models is explained.

2.1 A common structure of DTA models

Most existing DTA models share a common structure. In the literature, this common structure is often not explicitly stated in model formulation, but it can be extracted from those models. This common structure can be viewed as a high-level abstraction of the proposed modelling framework found in the work of Y. He [2.1] or see M.C.J. Bliemer [1.1]. This common structure consists of the following components:

1. a demand model
2. a supply model
3. a supply/demand interaction mechanism

This structure is depicted in figure 2.1.

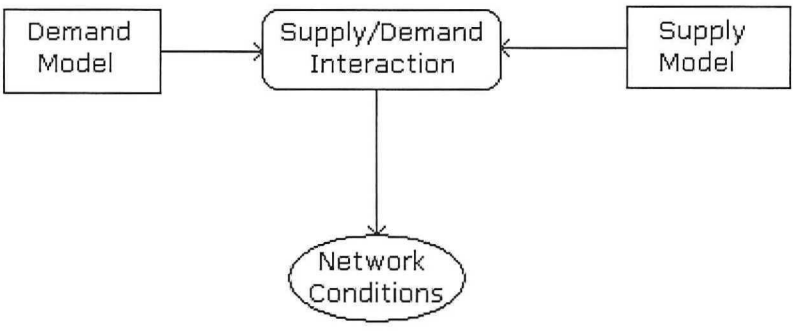

Figure 2.1: A common structure of DTA models

The demand model component represents the demand for the transportation system. The demand is usually given by a set of time-dependent Origin-Destination (OD) flows and path flows. The set of OD flows and path flows generated by the demand model often satisfy certain conditions such as system optimal and user optimal conditions. It should be noted that these two optimal conditions do not generally coincide. To achieve a system optimum, users must behave according to the system optimal conditions instead of following their own behaviours such as departure time choice, mode choice and route choice. On the other hand, if the demand model represents users' behaviours, a user optimum is attained.

The supply model represents the network and the flow progression in the network. A network is described as a directed and connected graph consisting of links and nodes and travel costs are associated with each link. The supply model generates the network performance in response to a given demand.

The supply/demand interaction mechanism represents how the supply model and the demand model interact. The interaction produces certain network conditions such as link or path flows and link or path travel times. The network conditions must satisfy both the demand model and the supply model.

2.2 A framework for the DTA model

A modelling framework for the DTA problem by Y. He [2.1] or M.C.J. Bliemer [1.1] is shown in figure 2.2. The framework contains the following components:

1. a users' behaviour model component
2. a dynamic network loading model component
3. a link performance model component

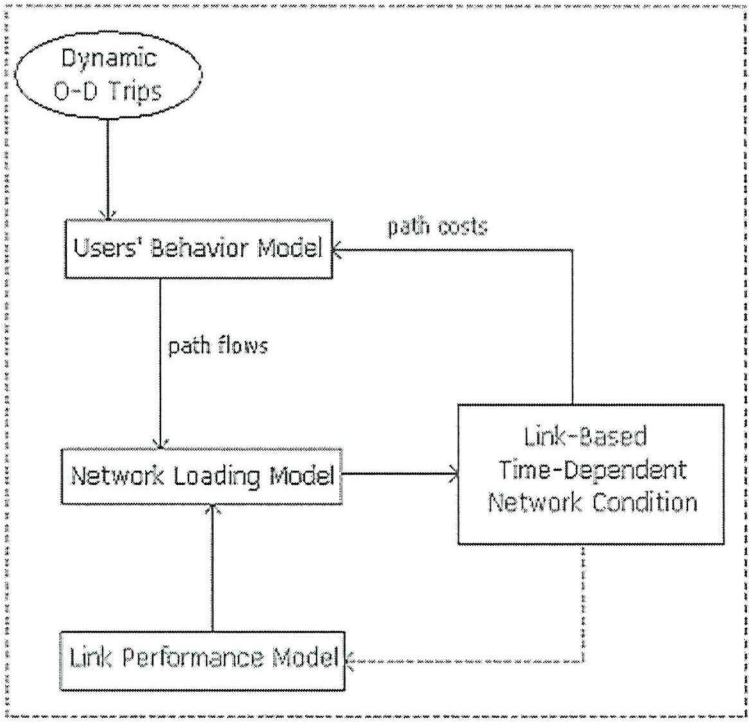

Figure 2.2: A Framework for DTA models

The users' behaviour model component takes as input the dynamic OD trips and a subset of paths between each OD pair. The dynamic OD trips are the time-dependent traffic demand for each OD pair. In the continuous time horizon, the dynamic OD trips are given as departure flow rates at each origin and each time instant. In discrete time representation, they are given as number of trips during a time interval. These dynamic OD trips can be predicted and are treated in the DTA model as input.

The subset of paths between each OD pair is assumed to be the set of routes, which the users choose when they depart from their origins. These subsets of paths can be dynamically augmented by using a path generation module based on certain criteria. The users' behaviour model component assigns the dynamic OD trips among the subset of paths according to the users' route choice behaviours. This results in a set of time-dependent path flows.

The network loading model takes the path flows from the users' behaviour model as input and uses link performance models to generate the resulting link-based network conditions such as time-dependent link volumes and link travel times. The link-based network conditions serve two purposes. Firstly, they are used to compute path travel times. The path travel times are then used by the users' behaviour model to assign OD trips. Secondly, the network conditions are

input to the path generation module to come up with a subset of new paths for each OD pair.

The proposed framework has a modularised structure and the components interrelate through specified inputs and outputs. The framework provides flexibility in both model formulations and computer implementations because one model can be changed without affecting others.

The users' behaviour model corresponds to the demand model in the common structure, the dynamic network loading model and link performance model together correspond to the supply model. The interaction between the three model components in the framework represents a supply/demand interaction mechanism.

2.3 Solution algorithm for the DTA model

An iterative process is needed to solve the DTA model. This is because network conditions may change after performing network loading. This results in a set of new path travel times and thus a set of new path flows. The set of new path flows is not necessarily equal to the set of path flows used in the previous network loading procedure.

The idea of the solution algorithm is to find a solution to the DTA model by an iterative process on path flows/costs or on unit flows/times. At all the iterations, the path flows/costs or the unit flows/times are updated by combining the results from the current iteration with the previous iteration. The 'Method of Successive Averages' (see chapter 4) is used to update path flows/costs or unit flows/times. The DTA algorithm with MSA included is outlined by Y. He [2.1] or M.C.J. Bliemer [1.1] in figure 2.3.

Step 0 (Initialisation):

- K = maximum number of iterations.
- Compute initial path flows $\left\{T_p^{rs(0)}(t)\right\}$ from free-flow path travel times.
- k=1.

Step 1 (Main loop):

- Perform dynamic network loading procedure.
- Compute auxiliary path flows $g_p^{rs}(t)$ by the route choice algorithm.
- Update path flows:

$$T_p^{rs(k+1)}(t) = T_p^{rs(k)}(t) + \alpha^{(k)} \cdot \left[g_p^{rs}(t) - T_p^{rs^k}(t)\right], \text{ with}$$

$$\alpha^{(k)} = \frac{1}{k}.$$

Step 2 (Stop criterion):
If k = K, then stop, else k=k+1 and go to step 1.

Figure 2.3: The DTA algorithm with MSA included

Explanation of the variables and functions used in the DTA algorithm with MSA included (figure 2.3):

- $T_p^{rs(k)}(t)$: calculated path flow rate on path *p* from origin *r* toward destination *s* at time *t* in iteration *k*
- $g_p^{rs}(t)$: auxiliary path flow for path *p* from origin *r* toward destination *s* starting at time *t*
- $\alpha^{(k)}$: adjustment parameter with value between zero and one in iteration *k*

For the implementation and examination of the different iterative averaging methods the DTA model is used (see figure 2.3). The stop criterion in figure 2.3 is a simple to implement stop criterion. Different kinds of stop criterions are discussed in section 3.3 and one of them is chosen for implementation.

3. Mathematical background of iterate averaging methods

Many important problems in transportation analysis and planning can be formulated as fixed-point problems; the fixed-point property generally translates a consistency constraint on model solutions such as, for example, equilibrium between the supply and demand relationships. Because of the typically large size of problem instances in transportation applications, the frequent absence of analytical forms for some of the involved maps, and the prevalent use of probabilistic maps requiring stochastic sampling or simulation methods for evaluation, 'classical' (e.g., derivative-based) fixed-point solution methods are often inappropriate for these problems. In such cases, the fixed-points are usually computed using one of the iterate averaging methods introduced by Robbins and Monro [3.1]. The Method of Successive Averages (MSA) was introduced by Sheffi and Powell [3.2] is probably the best-known and most widely-used instance in the transportation field (see also [3.3]).

This chapter will be an introduction to iterate averaging methods and the mathematical background (like convergence, stop criterions and step sizes).

3.1 Simple iterate averaging methods

The prototypical simple iterate averaging method is due to Robbins and Monro [3.1]. Let $\Re$ be the field of reals, and $T(.): \Re \to \Re$ be a map for which a root x^* is to be found (so that $T(x^*) = x^*$). Suppose that we are free to select the successive points x^k at which to evaluate $T(.)$ during the iterative search for the root – these are called the design points. However, each such evaluation returns a result that is affected by noise: the result is $T(x) = T(x) + \varepsilon$, where ε is a random zero-mean noise vector.

Robbins and Monro proposed the following iterative procedure for choosing the design points (x^{k+1}):

$$x^{k+1} = x^k + \alpha^k \cdot T(x^k), \text{ with } x^0 \in \Re \text{ and k=1,2,3,...,} \tag{3.1}$$

with the design point x^k, the evaluation $T(x^k)$ and where the sequence α^k is chosen so that the summability conditions are given in (3.2) and (3.3):

$$\sum_k \alpha^k \text{ diverges,} \tag{3.2}$$

$$\sum_k (\alpha^k)^2 \text{ converges.} \tag{3.3}$$

It is proved that under mild conditions the sequence x^k generated by this procedure converges in probability to a root x^* of T. Blum [3.4][3.5] extended this result under more stringent conditions to multidimensional maps. $T(.): \Re^n \to \Re^n$ and almost sure (a.s.) convergence. Clearly, these methods can find the fixed-point of a noisy map

by applying them to solve for the root of the transformed map $[T-I](.)$; the procedure is then:

$$x^{k+1} = x^k + \alpha^k \cdot \left[T\left(x^k\right) - x^k\right], \text{ with } x^0 \in \Re^n \text{ and k=1,2,3,....} \quad (3.4)$$

It is possible to write (3.4) in another way:

$$x^{k+1} = \left(1-\alpha^k\right)\cdot x^k + \alpha^k \cdot T\left(x^k\right), \text{ with } x^0 \in \Re^n \text{ and k=1,2,3,....} \quad (3.5)$$

3.2 Convergence of iterate averaging methods

Since the work of Robbins and Monro, much effort has been devoted to understanding and improving the convergence properties of iterate averaging methods. Because of the noise affecting the map evaluations, general discussion of the convergence behaviour of these algorithms is frequently expressed in terms of the statistical properties of the fixed-point estimate. The asymptotic distribution of $k \cdot \sqrt{x^k - x^*}$ was derived for certain classes of problems, and from this a formula for determining the optimal (i.e., asymptotic variance minimising) step size sequence α^k was obtained; unfortunately, it depends on generally unknowable quantities such as the value of the fixed point x^* itself. However, these results provide bounds against which the performance of other methods can be compared. The convergence of MSA, the Polyak method and the Bather method are discussed below.

For MSA (section 4.1) the convergence properties are typically disappointing: while it exhibits generally effective performance in the initial iterations, this is followed by a pronounced 'tail' effect, resulting in overall slow convergence.

For the Polyak method Polyak [3.6] and Polyak and Juditsky [3.7] showed that if $\alpha^k \to 0$ more slowly than MSA rate (specially, if), then the resulting asymptotic distribution of $k \cdot \sqrt{x^k - x^*}$ attains the minimum possible variance. The larger step sizes tend to prevent the algorithm from getting stuck at an early stage, while the off-line averaging takes care of the increased noise that the larger step sizes produce. It has been known for a long time that if $\alpha^k = O\left(1/k\right)$, then the asymptotic behaviour of mean is no better than the design points and can be worse in the sense of rate of convergence. Remarkably, therefore, this easy-to-implement procedure equals or surpasses the theoretical asymptotic performance of any possible iterate averaging method.

Schwabe and Walk [3.8] have shown that the Bather method has the same asymptotically optimal convergence properties as the Polyak method (i.e., it converges to a solution with minimal asymptotic variance), but that it may be less sensitive to the choice of initial value; consequently, it may exhibit superior properties for small numbers of iterations.

3.3 Stop criterion for DTA algorithms

For stopping the DTA algorithm there are several possibilities.

1. One of them is already shown in figure 2.3. After K iterations the DTA algorithm is stopped (3.6).

   If k = K then stop. (3.6)

2. The Equilibrium State is reached if the solution does not change anymore. A second possibility for a stop criterion is shown in (3.7).

   If $1-\varepsilon < \frac{c_{tr}^{k}}{c_{tr}^{k+1}} < 1+\varepsilon \ \forall t,r$ then stop, with ε small enough. (3.7)

3. We want to achieve an Equilibrium State, using the DTA algorithm. This Equilibrium State is reached if the costs of one route are the same as the costs of the other route ($c_{t1} = c_{t2} \ \forall t$) or if the route costs are not the same all the traffic should chose the cheapest route.

   If $gap = \left| c_{tr} - c_{tr} \right| < \varepsilon \ \forall t,r$ then stop, with ε small enough. (3.8)

4. Define the normalised duality gap (ng) as:

$$\text{ng} = \frac{\sum_t \sum_r \left| \min\{c_{tr}\} - c_{tr} \right| \cdot x_{tr}}{\sum_t \sum_r c_{tr} \cdot x_{tr}} \tag{3.9}$$

   We stop is ng < ε, with ε small enough. This stop criterion is used for solving the small traffic problem, because this criterion explains the best the state we want to achieve (explained in more detail in subsection 4.1.1, with four time periods and two routes).

Explanation of the variables used in the stop criteria:

c_{tr} : the route cost *c* of route *r* in time period *t*

x_{tr} : the route flow *x* of route *r* in time period *t*

3.4 Step size for iterate averaging algorithms

The difficulty of selecting a good step size sequence $\{\alpha^k\}$ has been a serious handicap in applications. In a fundamental paper, Polyak and Juditsky [3.7] showed that if α^k goes to zero slower than $O\left(1/k\right)$, the averaged sequence $\frac{1}{k} \cdot \sum_{i=1}^{k} x^i$ converges to its limit at an optimum rate. This result implies that we should use larger than usual gains and let the off-line averaging take care of the increased noise effects (due to the larger step size), with substantial overall improvement. The basic stochastic approximation algorithm tends to be more robust with a larger step size, therefore it is less likely to get stuck at an early stage and more likely to have a faster initial convergence.

The usual idea is to select the step sizes in such a way that an appropriate measure of the rate of convergence is maximised. Typically, the step sizes are required to be chosen as

$$\alpha^k = p \cdot k^{-\beta}, \text{ with } 0.5 < \beta \leq 1 \text{ and } p > 0, \tag{3.10}$$

so that α^k is sufficiently small for k=1,2,3,...; this is sufficient to allow the algorithm to converge beginning in an arbitrary starting point, while ensuring that the variance of the successive iterates decreases to zero so that the sequence converges to a single value.

3.4.1 Step size illustration

The step size changes of MSA, the Polyak method and the Bather method are illustrated in figure 3.1. These step sizes are used when the numerical example in subsection 4.1.1. is solved.

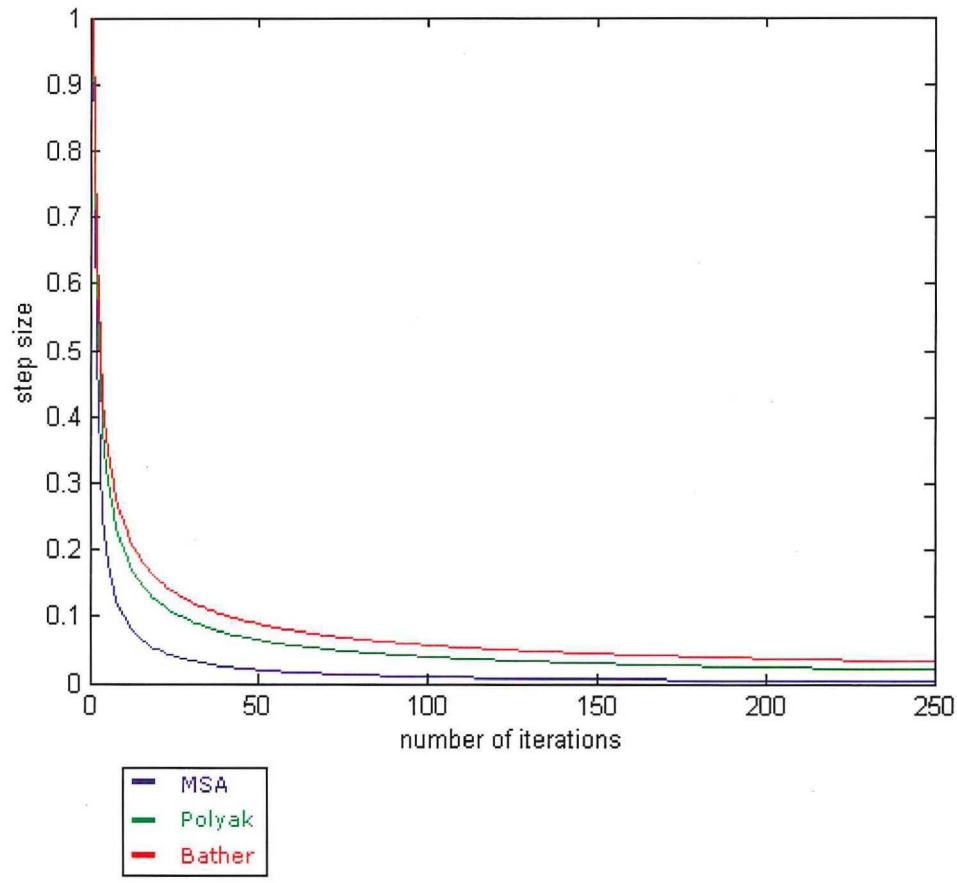

Figure 3.1: The step sizes of MSA, the Polyak method and the Bather method at increasing number of iterations

	p	β
MSA	1.00	1.00
Polyak	1.00	0.70
Bather	1.00	0.62

Table 3.1: Parameters of the step sizes of MSA, the Polyak method and the Bather method (see (3.10)).

In figure 3.1 it can be seen that the step sizes of the Bather method go slower to zero than the step sizes of the Polyak method. The step sizes of these two methods go both slower to zero than the step sizes of MSA. MSA has reached the stop criterion after 190 iterations, the Polyak method has reached the stop criterion after 226 iterations and the Bather method has reached the stop criterion after 118 iterations.

4. Method of Successive Averages (MSA)

The Method of Successive Averages (MSA) has the advantages of avoiding (potentially expensive) step size calculations, working directly with map outputs without requiring derivative calculations or other transformations, and being able to handle 'noisy' map evaluations (where the evaluation returns a value affected by a zero-mean disturbance). In many cases, however, the method's empirically observed convergence properties are disappointing: while it exhibits generally effective performance in the initial iterations, this is followed by a pronounced 'tail' effect, resulting in overall slow convergence.

In this chapter MSA is explained and an example of solving a transportation problem using MSA is given.

4.1 The Method of Successive Averages

The classical application of iterate averaging to a transportation problem is introduced by Sheffi and Powell [3.2], who used it to minimise a twice continuously differentiable functional (the objective function of the unconstrained convex optimisation formulation of the Stochastic User Equilibrium problem). In their application, the function $T(.)$ provided a noisy descent direction while the stochastic potential function was provided by the objective function itself. Although any step size sequence α^k satisfying the summability conditions (3.2) and (3.3) (section 3.1) could be used in an iterate averaging method, a particular sequence $\alpha^k = 1/k$ (formule (3.10) with p=1 and β=1) was proposed by Sheffi and Powell [3.2] as the basis for their solution algorithm for the static stochastic user equilibrium problem. Because each successive design point generated by this method is the average of the preceding map evaluation results, they called this MSA (see (4.1)).

$$x^{k+1} = x^k + \alpha^k \cdot \left[T\left(x^k\right) - x^k\right],\ x^0 \in \Re^n \text{ and k=1,2,3,....} \tag{4.1}$$

$$\alpha^k = 1/k, \qquad \text{k=1,2,3,....} \tag{4.2}$$

MSA is attractive because it inherits the robust convergence properties of iterate averaging methods while requiring only a trivial step size calculation.

MSA has been applied to a wide variety of problems that arise in transportation analysis and planning. In some applications, it can be rigorously proven to converge to a fixed-point, whereas in others it is used as a heuristic that has been found to give good results in practice. It avoids (potentially expensive) step size calculations, and it works directly with model outputs without requiring derivative calculations or other transformations.

As an example, Cascetta and Postorino [4.2] observed that in MSA an iteration's estimate is affected by the results from all prior iterations, including those from early iterations that are presumably far from the solution. Furthermore, later iterations, which are presumably closer to the solution, receive smaller weights when computing a new estimate.

4.1.1 A numerical example of MSA

To illustrate MSA, and also the next algorithms that are described in this report a small example is given. There are three cities and three links connecting these cities (see figure 4.1). Link 1 forms route 1 and link 2 and link 3 form together route 2. Travellers want to travel from A to B and want to take the cheapest route. Four time periods are considered. For each time period there is a demand *d* (4.2).

$$d = [15 \;\; 20 \;\; 15 \;\; 20] \tag{4.3}$$

The route costs of two routes are computed by the costs of the links. Each link has costs *c*:

$$c = 1 + 0.01 \cdot u^2 + 0.01 \cdot v^2 \tag{4.4}$$

Where u is the number of travellers entering the road and v is the density on the road.

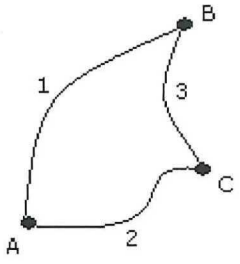

Figure 4.1: The cities and routes of the small problem described

The results of MSA after 100.000 iterations are shown in table 4.1. In the first column the route costs and number of travellers in the attained Equilibrium State of route 1 are shown and in the second column the route costs and number of travellers in the attained Equilibrium State of route 2 are shown.

	MSA	
time periods	*route 1*	*route 2*
	route costs	
1	2,2750	2,2750
2	3,5915	3,5915
3	6,2796	6,2795
4	7,1277	7,5794
	route flows	
1	11,2917	3,7083
2	11,4736	8,5264
3	3,1140	11,8860
4	19,9992	0,0008

Table 4.1: Route costs and flows of the problem solved with MSA in the DTA algorithm.

Now the stop criterion is set to 0.001 and MSA is taken to solve the previous described problem. The outcomes are given in figure 4.2 and table 4.1.

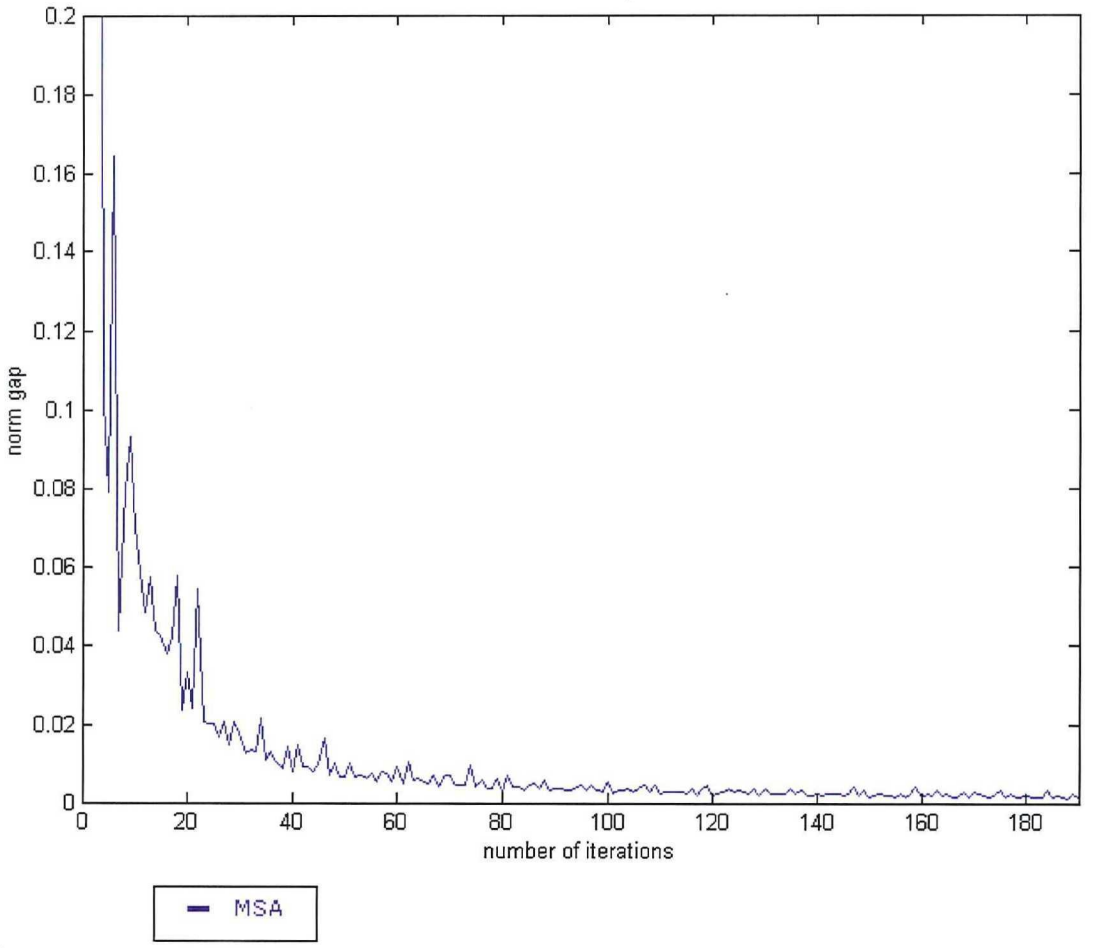

Figure 4.2: The normalised duality gap value with increasing number of iterations when the problem is solved with MSA in the DTA algorithm

	MSA	
time periods	*route 1*	*route 2*
	route costs	
1	2,2745	2,2754
2	3,5910	3,5916
3	6,2813	6,2587
4	6,9742	7,5529
	route flows	
1	11,2895	3,7105
2	11,4737	8,5263
3	3,1579	11,8421
4	19,5789	0,4211

Table 4.2: Route costs and flows of the problem solved with MSA in the DTA algorithm. The stop criterion is set to 0.001 (see figure 4.2).

stop criterion	0.001

Tabel 4.3: If de ng is smaller than de stop criterion the algorithm is stopped.

The normalised gap value for every iteration step of the solution algorithm is shown in figure 4.2. The normalised gap goes to zero and when the normalised gap is smaller than the stop criterion the program stops. MSA requires 190 iterations to reach the stop criterion. The values of the route costs and flows of the two routes are shown in table 4.1. The first column shows the route costs and number of travellers in the Equilibrium State of route 1 while the second column shows the route costs and number of travellers in the Equilibrium State of route 2. It appears that the route costs of the first three time periods are almost identical. In the fourth time period the route costs differ substantially. This is caused by the fact that the value for the most expensive route in the last time period in the equilibrium goes to zero.

5. Polyak and Bather methods

5.1 Introduction to the Polyak and Bather method

Approximately ten years ago, Polyak [5.1] and Bather [5.2] proposed two relatively minor modifications of the iterate averaging method which were rigorously shown to produce fixed-point estimates with asymptotically optimal properties. These 'accelerated' methods have not been widely discussed in the transportation literature. However, in initial computational experiments where the Polyak method was applied to the DTA model and the anticipatory route guidance generation problems, the method often exhibited convergence rates four or more times faster than MSA [1.2]. In view of their attractive theoretical properties and these encouraging preliminary empirical results, the two methods would seem to merit serious consideration by the transportation community.

It is characteristic of iterate averaging that the design points generated by successive iterations of the algorithm are also taken to be the successive estimates of the equation solution. This was also noted by Frees and Ruppert [5.3], who pointed out the advantages of using one method to select the design points, and a different method to estimate the solution. Use of methods adapted to each purpose could allow, for example, a more aggressive exploration of the feasible space and a more effective exploitation of the results generated during that exploration to estimate a solution. One family of methods that exploits this idea is called iterate averaging.

5.2 The Polyak method

The Polyak method is a two-pass method. The first pass resembles MSA except that the step sizes are larger; this allows the algorithm to explore the solution space more aggressively but leads to greater variability in the outputs. The second pass is carried out offline (i.e., without influencing the first pass); it calculates an average of iterates that are generated by the first pass. The average calculated by the second pass at termination is the fixed-point solution estimate.

One method that implements this idea is due to Polyak [5.1]. Let the equation

$$x^{k+1} = x^k + \alpha^k \cdot \left[T\left(x^k\right) - x^k\right], \text{ with } x^0 \in \Re^n \text{ and k=1,2,3,....} \qquad (5.1)$$

be a simple averaging process and suppose that the process converges to a fixed-point x^*. In the Polyak method one also computes, 'in parallel' with and independently of the simple averaging process, a running average of the design points x^k that is generated, say

$$\bar{x}^k = \frac{1}{k} \cdot \sum_{i=1}^{k} x^i \text{ , with } x^0 \in \Re^n \text{ and k=1,2,3,....} \qquad (5.2)$$

The sequence $\bar{x}^k$ also converges to the limit x^*.

5.2.1 A numerical example of the Polyak method

The traffic problem with three cities and two routes described in subsection 4.1.1. is considered. In the Polyak method the adjustment parameter α is computed as follows:

$$\alpha^k = p \cdot k^{-\beta}, \tag{5.3}$$

where k is the number of iterations, β can be chosen between values of 0.5 and 1.0 and p>0. In the rest of this report p will be one. Figure 5.1 shows the number of iterations with which the problem is solved using the Polyak method for a given value of β. Therefore, the best value for β can be chosen for the calculation. According to figure 5.1 the best value for β is 0.70.

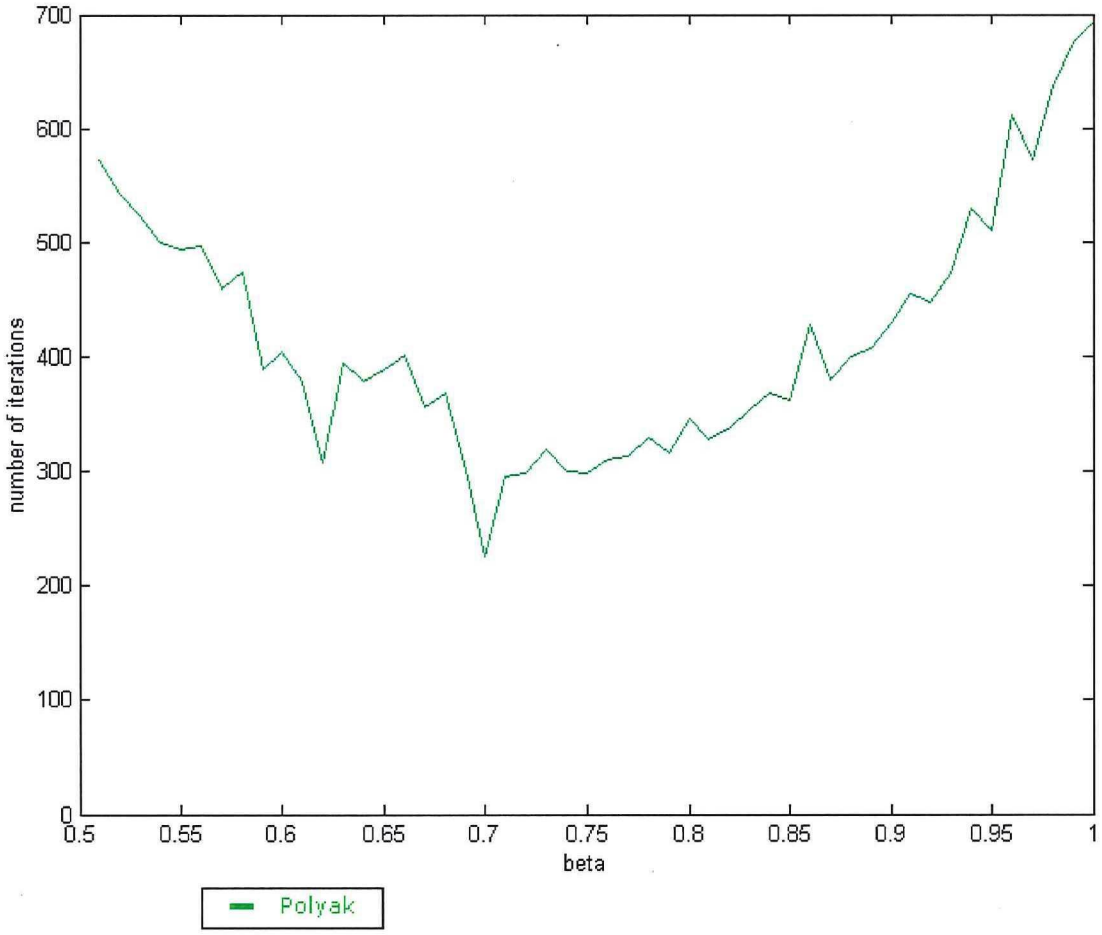

Figure 5.1: Required number of iterations for the Polyak method for given values of β

stop criterion	0.001

Tabel 5.1: If de ng is smaller than de stop criterion the algorithm is stopped.

To solve the given assignment problem using the Polyak method, the stop criterion is set to 0.001, p to 1 and β to 0.70. The outcomes are given in figure 5.2 and table 5.2.

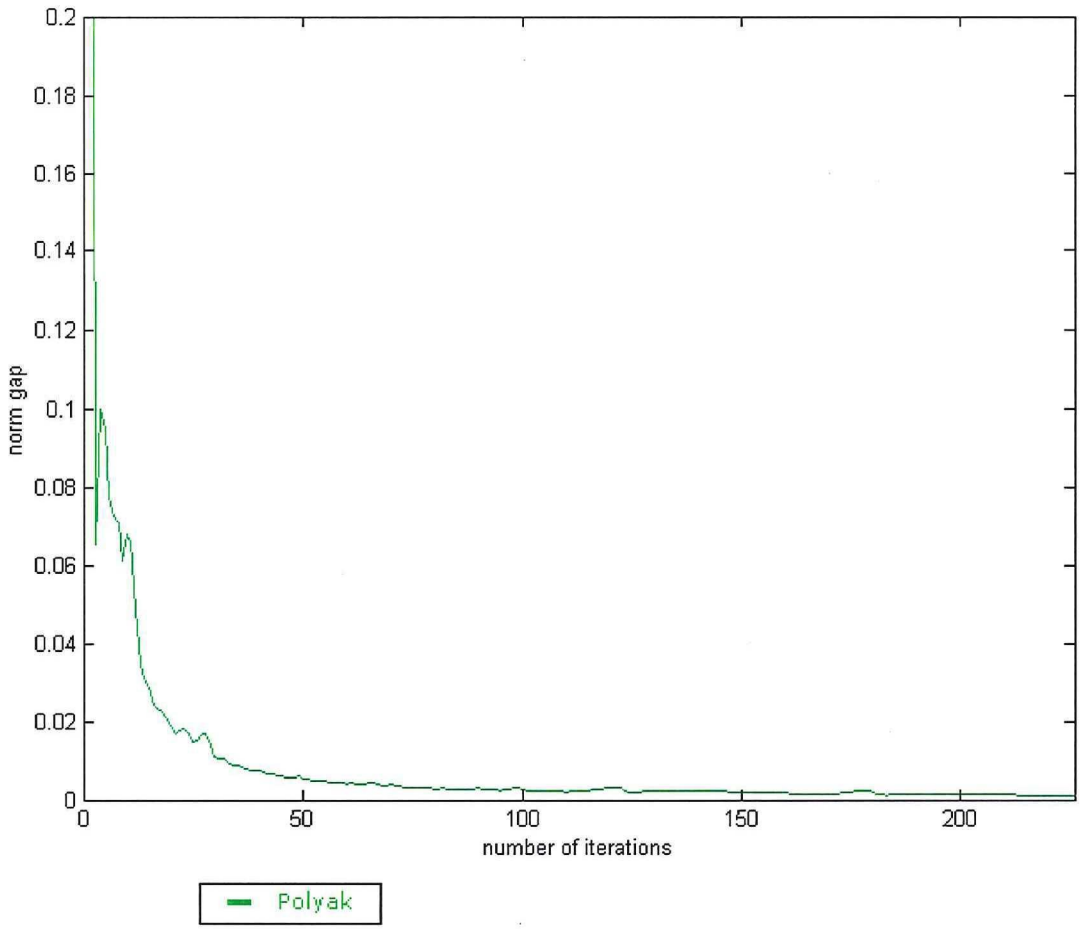

Figure 5.2: The normalised duality gap value with increasing number of iterations when the problem is solved with the Polyak method in the DTA algorithm

	Polyak	
time periods	*route 1*	*route 2*
	route costs	
1	2,2515	2,2908
2	3,5795	3,5823
3	6,2558	6,2553
4	7,0240	7,5502
	route flows	
1	11,1871	3,8129
2	11,5238	8,4762
3	3,1291	11,8709
4	19,6898	0,3102

stop criterion	0.001
β	0.70

Tabel 5.3: Variables used for figure 5.2.

Table 5.2: Route costs and flows of the problem solved with the Polyak method in the DTA algorithm (see figure 5.2).

The normalised gap value for every iteration of the solution algorithm is shown in figure 5.2. The normalised gap goes to zero and when the normalised gap is smaller than the stop criterion the program stops. The Polyak method requires 226 iterations to reach the stop criterion. The values of the route costs and flows of the two routes are shown in table 5.1. The first column shows the route costs and number of travellers in the Equilibrium State of route 1 while the second column the route costs and number of travellers in the Equilibrium State of route

2 shows. It appears that the route costs of the first three time periods are almost identical. In the fourth time period the route costs differ. This is caused by the fact that the value for the most expensive route in the last time period in the equilibrium goes to zero.

5.3 The Bather method

A somewhat different approach was proposed by Bather [5.2]. Here, the design point in each iteration is derived from a combination of the average of previous design points with the average of previous evaluation results:

$$x^{k+1} = \bar{x}^{k} - k \cdot \alpha^{k} \cdot \left(\tau^{k} - \bar{x}^{k}\right), \text{ with } x^{0} \in \Re^{n} \text{ and k=1,2,3,...,} \tag{5.4}$$

$$\alpha^{k} = p \cdot k^{-\beta}, \text{ with } 0.5<\beta\leq 1 \text{ and } p>0, \tag{5.5}$$

where $\bar{x}^{k}$ is as before, the running average of the design points (x^{k}) selected in previous iterations, while

$$\tau^{k} = \frac{1}{k}\sum_{i=1}^{k} T\left(x^{i}\right), \text{ with } x^{0} \in \Re^{n} \text{ and k=1,2,3,...,} \tag{5.6}$$

is the running average of the corresponding function evaluation results minus the running average of the design points. As in the Polyak method, the function evaluations are made at the design points (x^{k}) while the fixed point is estimated by $\bar{x}^{k}$. Since,

$$(k+1)\bar{x}^{k+1} = k \cdot \bar{x}^{k} + x^{k+1}, \tag{5.7}$$

Bather's recursion can also be expressed as

$$\bar{x}^{k+1} = \bar{x}^{k} + \frac{k}{k+1} \cdot \alpha^{k} \cdot \left(\tau^{k} - \bar{x}^{k}\right), \text{ with } x^{0} \in \Re^{n} \text{ and k=1,2,3,...,} \tag{5.8}$$

$$\alpha^{k} = p \cdot k^{-\beta}, \text{ with } 0.5<\beta\leq 1 \text{ and } p>0, \tag{5.9}$$

$$\tau^{k} = \frac{1}{k}\sum_{i=1}^{k} T\left(x^{i}\right), \text{ with } x^{0} \in \Re^{n} \text{ and k=1,2,3,...,} \tag{5.10}$$

thus it resembles the basic Robbins-Monro procedure with the design points replaced by their averages.

5.3.1 A numerical example of the Bather method

The traffic problem with three cities and two routes described in subsection 4.1.1. is considered. In the Bather method the adjustment parameter α is computed as follows:

$$\alpha^k = p \cdot k^{-\beta}, \tag{5.11}$$

where k is the number of iterations and b can be chosen between values of 0.5 and 1.0. Figure 5.3 shows the required number of iterations in which the problem is solved using the Bather method for given values of β (and p=1). Therefore, the best value for β can be chosen. According to figure 5.3 the best value for β is 0.62.

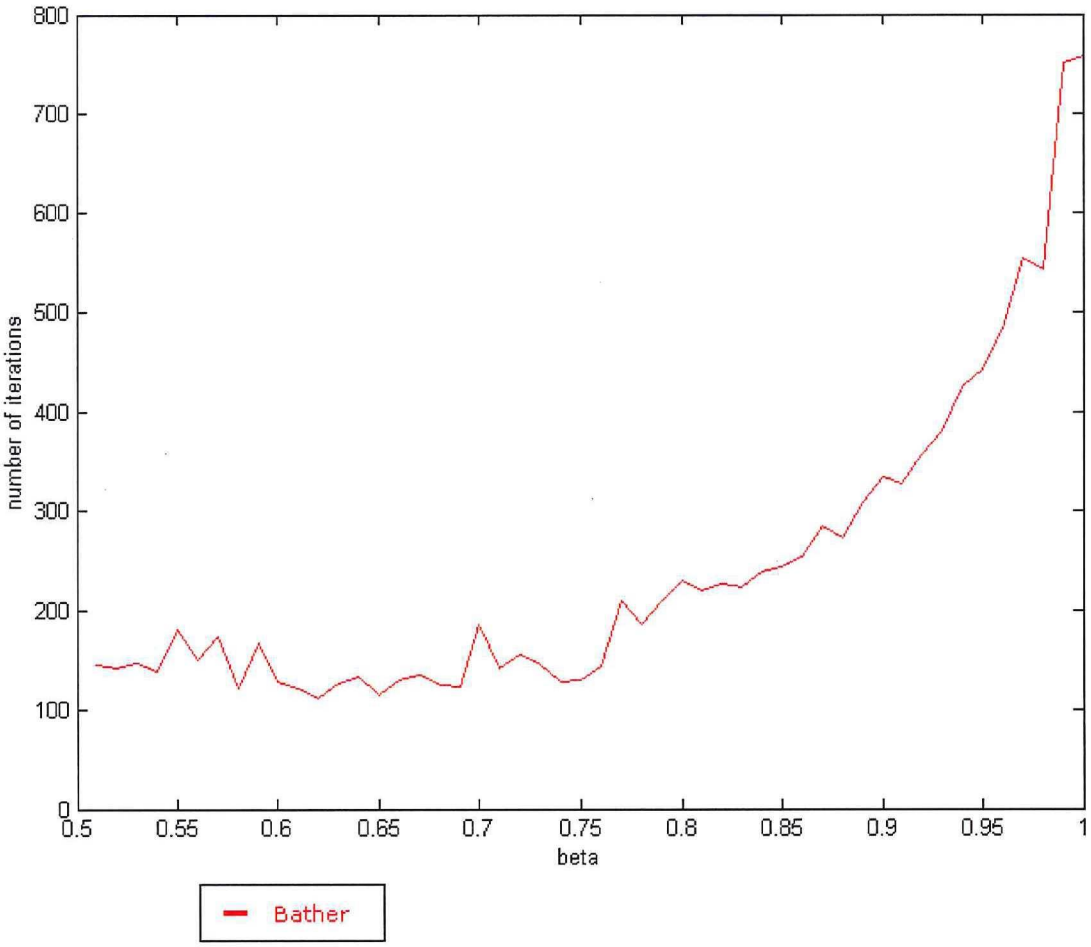

Figure 5.3: Required number of iterations for the Bather method for given values of β

stop criterion	0.001

Tabel 5.4: If de ng is smaller than de stop criterion the algorithm is stopped.

To solve the given assignment traffic problem using the Bather method, the stop criterion is set to 0.001, p to 1 and β to 0.62. The outcomes are given in figure 5.4 and table 5.5.

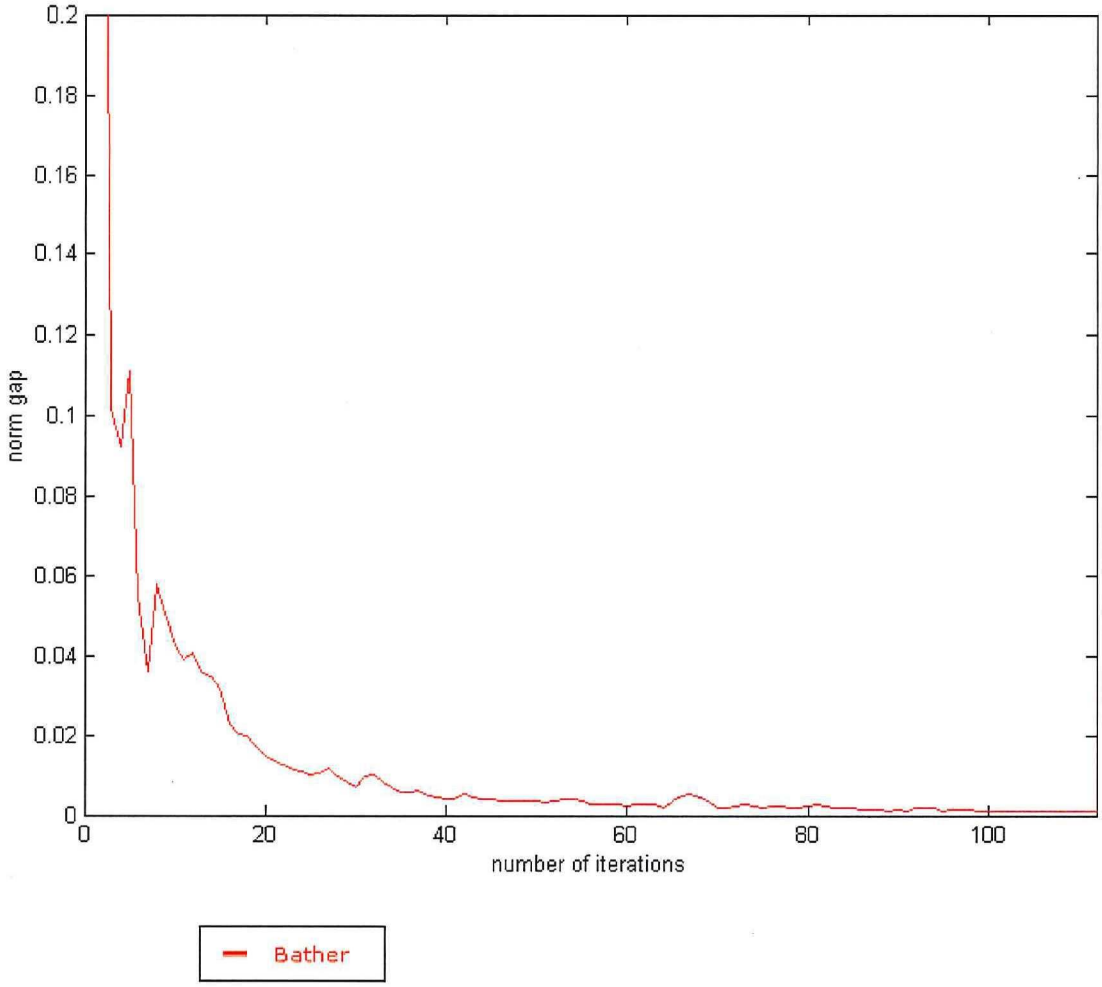

Figure 5.4: The normalised duality gap value with increasing number of iterations when the problem is solved with the Bather method in the DTA algorithm

	Bather	
time periods	*route 1*	*route 2*
	route costs	
1	2,2710	2,2777
2	3,5888	3,5908
3	6,2752	6,2693
4	6,9636	7,5677
	route flows	
1	11,2740	3,7260
2	11,4794	8,5205
3	3,1313	11,8687
4	19,5674	0,4326

Table 5.5: Route costs and flows of the problem solved with the Bather method in the DTA algorithm (see figure 5.4).

stop criterion	0.001
β	0.62

Tabel 5.6: Variables used for figure 5.4.

The normalised gap value for every iteration of the solution algorithm is shown in figure 5.4. The normalised gap goes to zero. When the normalised gap is smaller than the stop criterion the program is stopped. The Bather method requires 112 iterations to reach the stop criterion. The values of the route costs and flows of the two routes are shown in table 5.5. The first column shows the route costs and number of travellers in the Equilibrium State of route 1 while the second column shows the route costs and number of travellers in the Equilibrium State of route 2. It appears that the route costs of the first three time periods are almost identical. In the fourth time period the route costs differ. This is caused by the fact that the value for the most expensive route in the last time period in the equilibrium goes to zero.

6. Alternative methods

In this chapter two new methods are introduced. The Bliemer method is proposed by Dr. M.C.J. Bliemer and after that the Bliemer Moving method is proposed by R.T.J. Hiele. The other alternative methods introduced in this chapter are combinations of earlier introduced methods.

6.1 The Bliemer method

Let

$$x^{k+1} = x^k + \alpha^k \cdot \left[T\left(x^k\right) - x^k\right], \text{ with } x^0 \in \Re^n \text{ and k=1,2,3,....} \quad (6.1)$$

be a simple averaging process and suppose that the process converges to a fixed point x^*. In the Polyak method one also computes, 'in parallel' with and independently of the simple averaging process, a running average of the design points (x^k) that is generated, say

$$\bar{x}^k = \frac{1}{k} \cdot \sum_{i=1}^{k} x^i \text{ , with } x^0 \in \Re^n \text{ and k=1,2,3,....} \quad (6.2)$$

The Bliemer method is almost the same as the Polyak method, with the difference that the observations ($T(.)$) are evaluated in the average of the previous design points ($\bar{x}^k$) and not at the previous design point(x^k):

$$x^{k+1} = x^k + \alpha^k \cdot \left[T\left(\bar{x}^k\right) - x^k\right], \text{ with } x^0 \in \Re^n \text{ and k=1,2,3,....} \quad (6.3)$$

$$\alpha^k = p \cdot k^{-\beta}, \text{ with p>0, } 0.5<\beta\leq 1.0 \text{ and k=1,2,3,....} \quad (6.4)$$

6.1.1 A numerical example of the Bliemer method

The traffic problem with three cities and two routes described in subsection 4.1.1. is considered. In figure 6.1 the number of iterations in which the problem is solved using the Bliemer method is expanded against β. In this way, the best β for the calculation can be found. According to figure 6.1 the best value for β is 0.54.

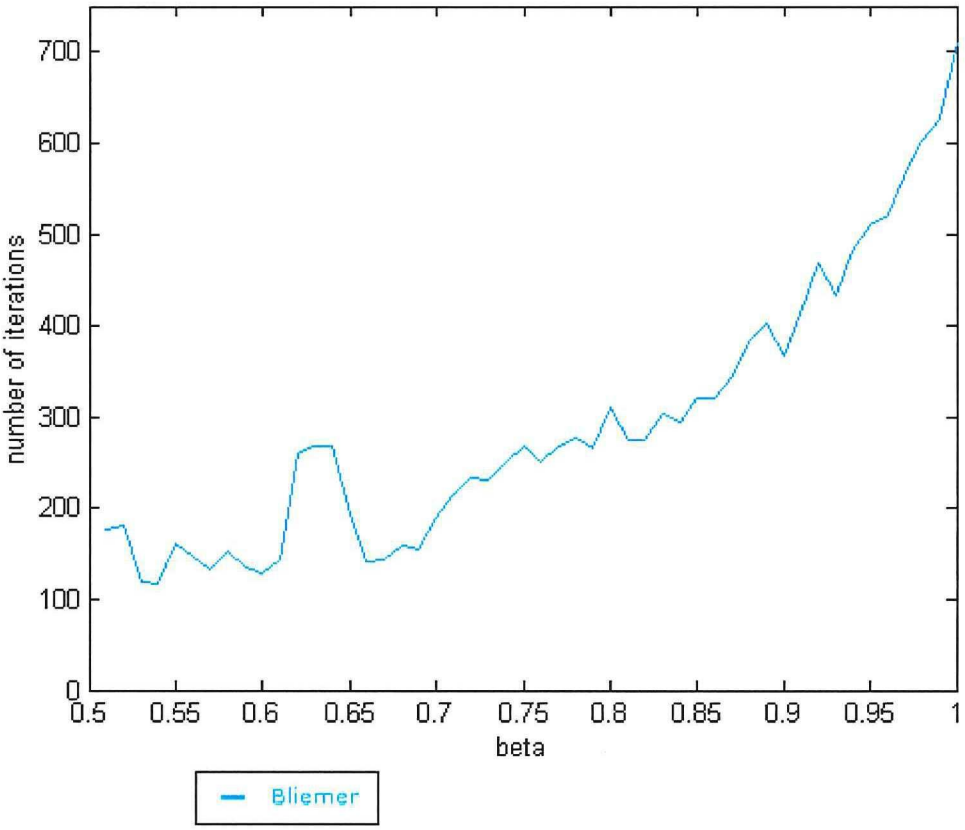

Figure 6.1: Required number of iterations for the Bliemer method for given values of β

stop criterion	0.001

Tabel 6.1: If de ng is smaller than de stop criterion the algorithm is stopped.

To solve the given traffic problem using the Bliemer method the stop criterion is set to 0.001, p to 1 and β to 0.54. The outcomes are given in figure 6.2 and table 6.2.

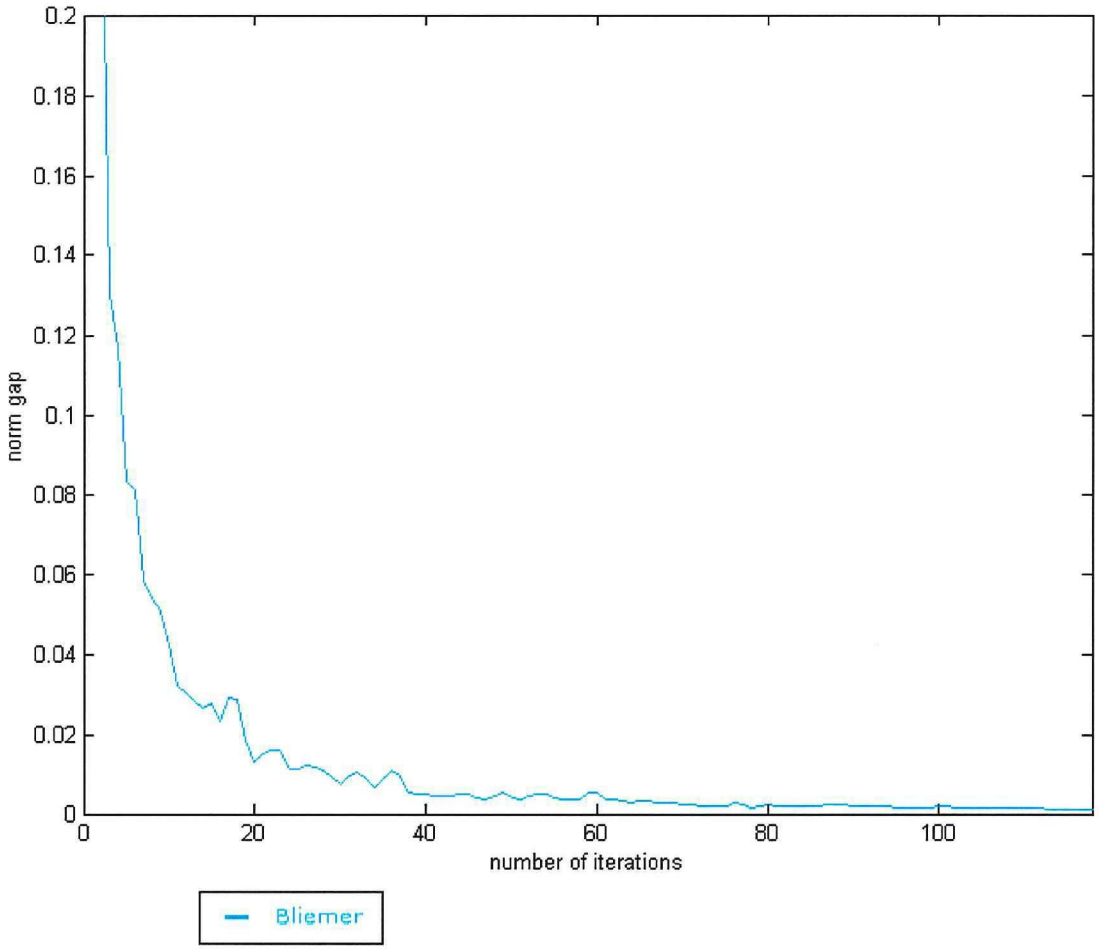

Figure 6.2: The normalised gap value with increasing number of iterations when the problem is solved with the Bliemer method in the DTA algorithm

	Bliemer	
time periods	*route 1*	*route 2*
	route costs	
1	2,2748	2,2752
2	3,5917	3,5910
3	6,2803	6,2760
4	6,9426	7,5769
	route flows	
1	11,2909	3,7091
2	11,4753	8,5247
3	3,1203	11,8797
4	19,5251	0,4749

Table 6.2: Route costs and flows of the problem solved with the Bliemer method in the DTA algorithm (see figure 6.2).

stop criterion	0.001
β	0.54

Tabel 6.3: Values of the variables used for figure 6.2

The normalised gap value for every iteration of the solution algorithm is shown in figure 6.2. The normalised gap goes to zero and when the normalised gap is smaller than the stop criterion the program is stopped. Using the Bliemer method there are 118 iterations needed to reach the stop criterion. The values of the route costs and the values of the equilibrium of the two routes are shown in table 6.1. In the first column the route costs and number of travellers in the Equilibrium State of route 1 are shown and in the second column the route costs

and number of travellers in the Equilibrium State of route 2 are shown. It appears that the route costs of the first three time periods are almost identical. In the fourth time period the route costs differ. This is caused by the fact that the value for the most expensive route in the last time period in the equilibrium goes to zero.

6.2 The Bliemer Moving method

The Bliemer Moving method is almost the same as the Bliemer method (section 6.1). The Bliemer Moving method is the Bliemer method with a moving average:

$$x^{k+1} = x^k + \alpha^k \cdot \left[T\left(\hat{x}^k\right) - x^k \right], \text{ with } x^0 \in \Re^n \text{ and k=1,2,3,...,} \tag{6.5}$$

$$\hat{x}^k = {}^1\!/_M \sum_{i=k-M+1}^{k} x^i \text{ , with } x^0 \in \Re^n \text{ and k=1,2,3,...,} \tag{6.6}$$

$$\alpha^k = p \cdot k^{-\beta} \text{, with p>0, } 0.5<\beta\leq 1.0 \text{ and k=1,2,3,....} \tag{6.7}$$

where M is a fixed constant.

6.2.1 A numerical example of the Bliemer Moving method

The traffic problem with three cities and two routes described in subsection 4.1.1. is considered. In figure 6.3 the number of iterations in which the problem is solved using the Bliemer Moving method with moving equal to 30 is expanded against β. M is the number of design points taken for the moving average. Therefore, the best value for β can be chosen for the calculation. According to figure 6.3 the best value for β is 0.54 (or 0.55 or 0.56).

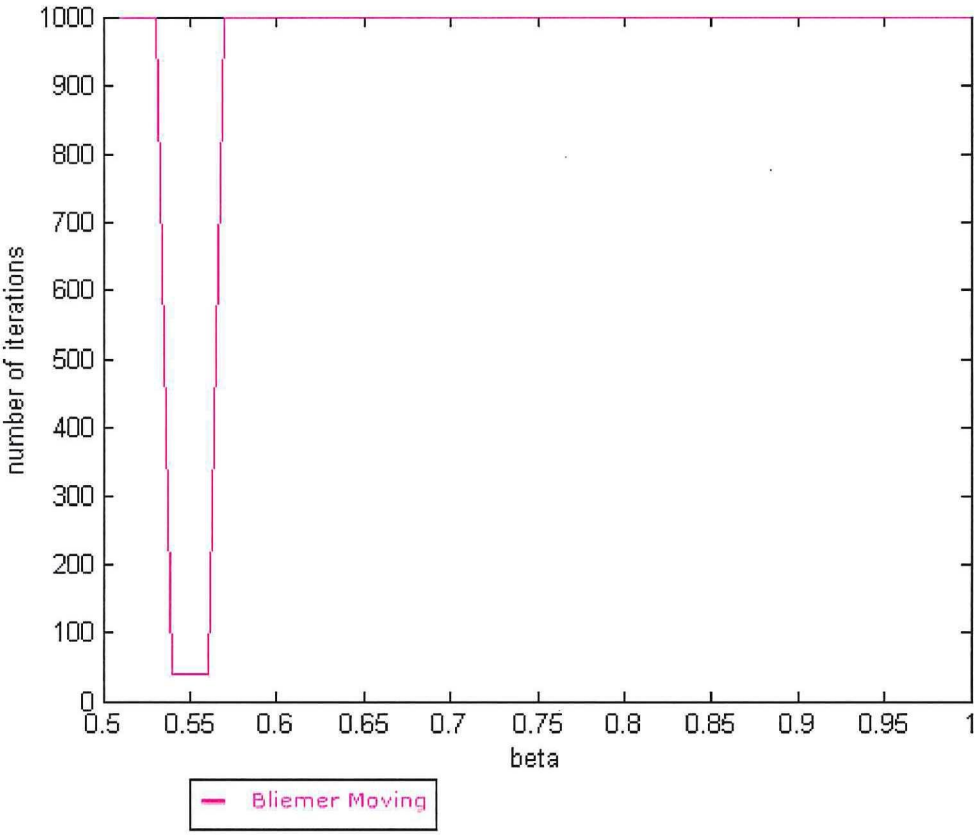

Figure 6.3: Required number of iterations for the Bliemer Moving method for given values of β

stop criterion	0.001
moving	30

Tabel 6.4: Values of the variables used for figure 6.3.

To solve the given traffic problem using the Bliemer Moving method the stop criterion is set to 0.001, p to 1 and β to 0.54. The outcomes are given in figure 6.4 and table 6.5.

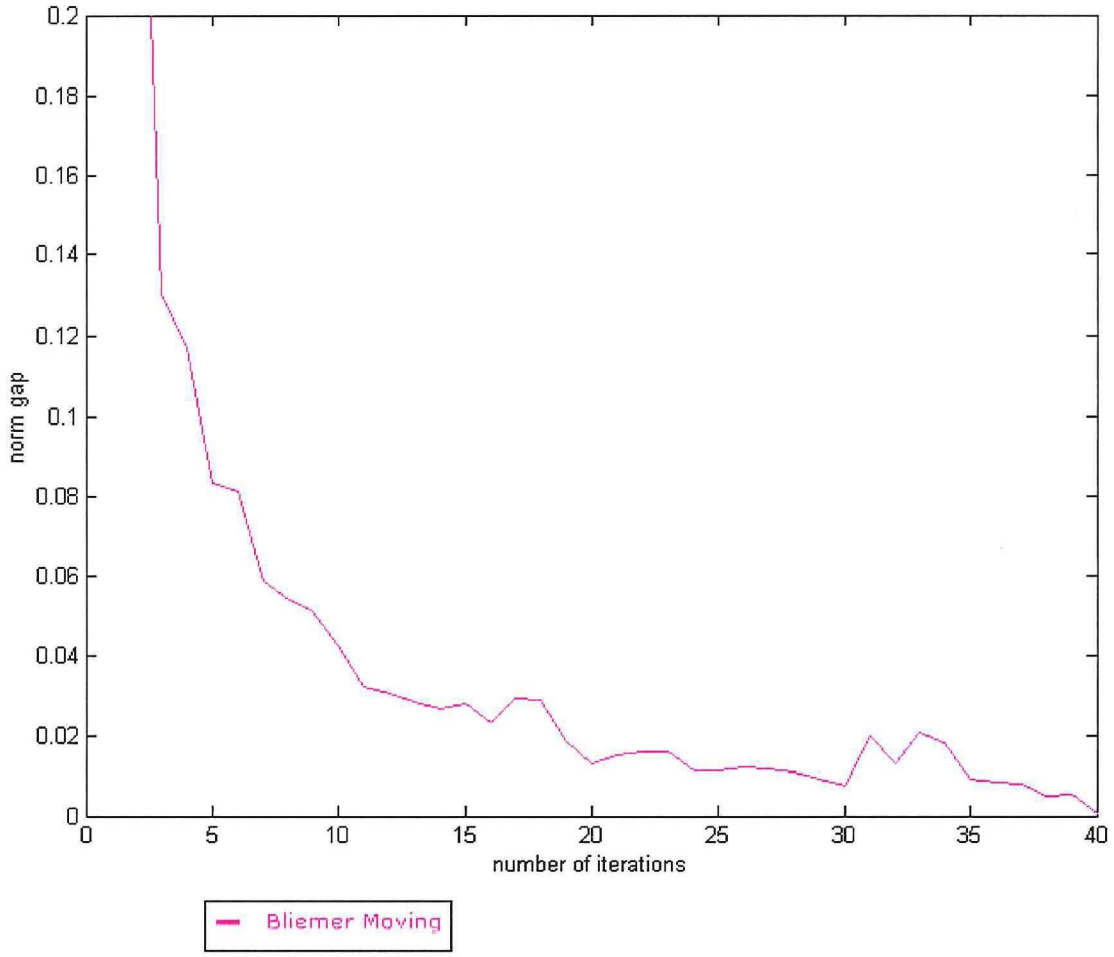

Figure 6.4: The normalised gap value with increasing number of iterations when the problem is solved with the Bliemer Moving method in the DTA algorithm

	Bliemer	**Moving**
time periods	*route 1*	*route 2*
	route costs	
1	2,2740	2,2757
2	3,5943	3,5861
3	6,2871	6,2594
4	7,1123	7,5530
	route flows	
1	11,2872	3,7128
2	11,4905	8,5095
3	3,1443	11,8557
4	19,9261	0,0739

Table 6.5: Route costs and flows of the problem solved with the Bliemer Moving method in the DTA algorithm (see figure 6.4).

stop criterion	0.001
β	0.54
moving	30

Tabel 6.6: Values of the variables used for figure 6.4.

The normalised gap for every iteration of the solution algorithm is shown in figure 6.4. The normalised gap goes to zero and when the normalised gap is smaller than the stop criterion the program is stopped. Using the Bliemer Moving method there are 40 iterations needed to reach the stop criterion. The values of the route costs and flows of the two routes are shown in table 6.5. In the first column the route costs and number of travellers in the Equilibrium State of route 1 are shown and in the second column the route costs and number of travellers in the

Equilibrium State of route 2 are shown. It appears that the route costs of the first three time periods are almost identical. In the fourth time period the route costs differ. This is caused by the fact that the value for the most expensive route in the last time period in the equilibrium goes to zero. Observe that the flow on route 2 in the last time period is much closer to zero when the Bliemer Moving method is used. The Bliemer Moving algorithm makes sure that bad design points are omitted.

6.3 The MSA-Bliemer method

This method first starts with MSA (see section 4.1) and after a certain number of iterations there is a switch to the Bliemer method (see section 6.1). This method is derived from Kushner's proposal. For more detail see [6.1].

6.3.1 A numerical example of the MSA-Bliemer method

The traffic problem with three cities and two routes described in subsection 4.1.1. is considered. In the MSA-Bliemer method the adjustment parameter α is computed as follows:

$$\alpha^k = p \cdot k^{-\beta}, \tag{6.8}$$

where k is the number of iterations and β can be chosen between values of 0.5 and 1.0. In figure 6.5 the number of iterations in which the problem is solved using the MSA-Bliemer method with n-switch equal to 33 is expanded against β. Therefore, the best value for β can be chosen for the calculation. According to figure 6.5 the best value for β is 0.54.

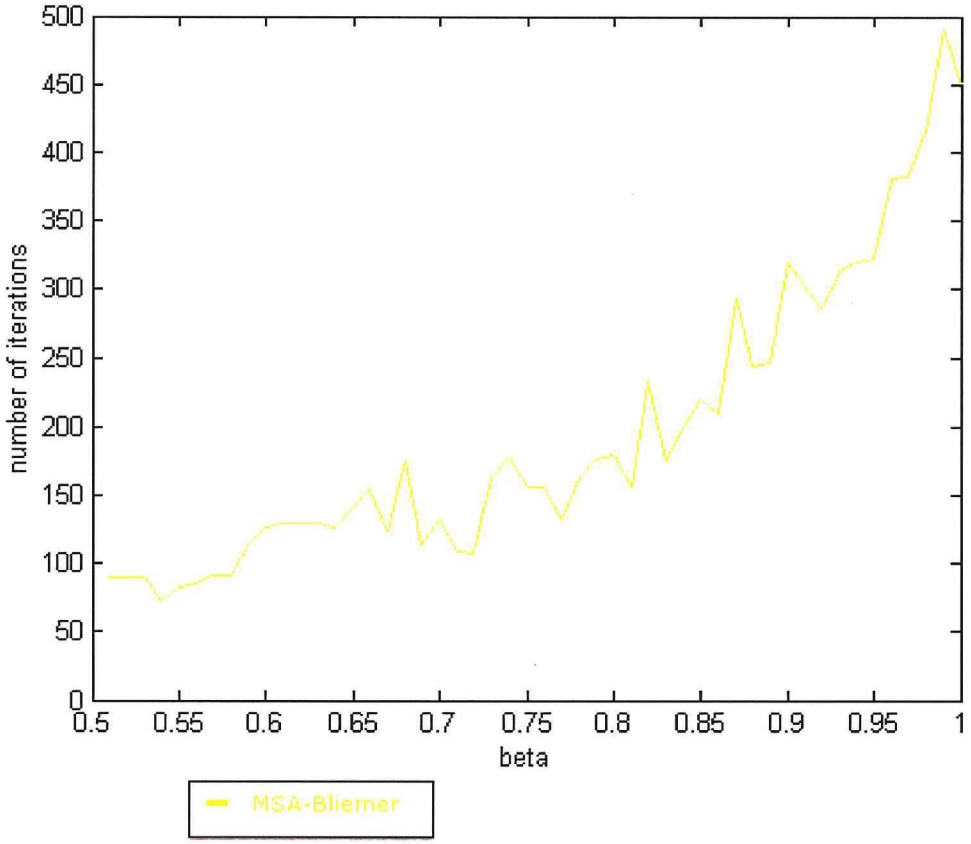

Figure 6.5: Required number of iterations for the MSA-Bliemer method for given values of β

stop criterion	0.001
n-switch	33

Tabel 6.7: Values of the variables used for figure 6.5.

To solve the given traffic problem using the MSA-Bliemer method the stop criterion is set to 0.001, p to 1 and β to 0.54. The outcomes are given in figure 6.6 and table 6.8.

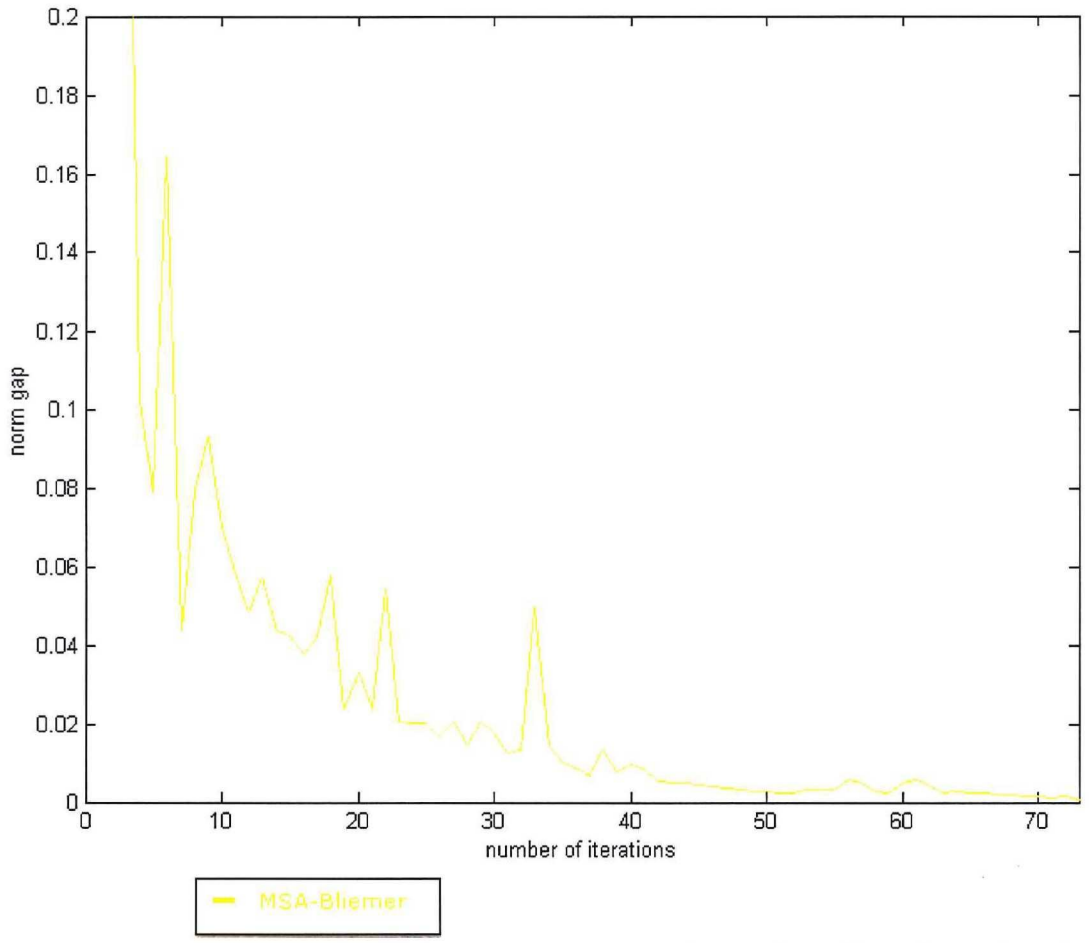

Figure 6.6: The normalised gap value with increasing the number of iterations when the problem is solved with the MSA-Bliemer method in the DTA algorithm

	MSA- Bliemer	
time periods	*route 1*	*route 2*
	route costs	
1	2,2738	2,2758
2	3,5909	3,5910
3	6,2790	6,2739
4	6,9825	7,5734
	route flows	
1	11,2863	3,7137
2	11,4764	8,5236
3	3,1238	11,8762
4	19,6236	0,3764

stop criterion	0.001
β	0.54
n-switch	33

Tabel 6.9: Values of the variables used for figure 6.6.

Table 6.8: Route costs and flows of the problem solved with the MSA-Bliemer method in the DTA algorithm (see figure 6.6).

The normalised gap value for every iteration of the solution algorithm is shown in figure 6.6. The normalised gap goes to zero and when the normalised gap is smaller than the stop criterion the program is stopped. Using the MSA-Bliemer method there are 73 iterations needed to reach the stop criterion. The values of the route costs and the values of the equilibrium of the two routes are shown in table 6.8. In the first column the route costs and number of travellers in the Equilibrium State of route 1 are shown and in the second column the route costs and number of travellers in the Equilibrium State of route 2 are shown. It appears that the route costs of the first three time periods are almost identical. In the fourth time period the route costs differ. This is caused by the fact that the value for the most expensive route in the last time period in the equilibrium goes to zero.

6.4 The MSA-Bather method

This method first starts with MSA (see section 4.1) and after a certain number of iterations there is a switch to the Bather method (see section 5.3).

6.4.1 A numerical example of the MSA-Bather method

The traffic problem with three cities and two routes described in subsection 4.1.1. is considered. In the MSA-Bather method the adjustment parameter α is computed as follows:

$$\alpha^k = p \cdot k^{-\beta}, \tag{6.9}$$

where k is the number of iterations and β can be chosen between values of 0.5 and 1.0. In figure 6.7 the number of iterations in which the problem is solved using the MSA-Bather method with n-switch equal to 10 is expanded against β. Therefore, the best value for β can be chosen for the calculation. According to figure 6.7 the best value for β is 0.67.

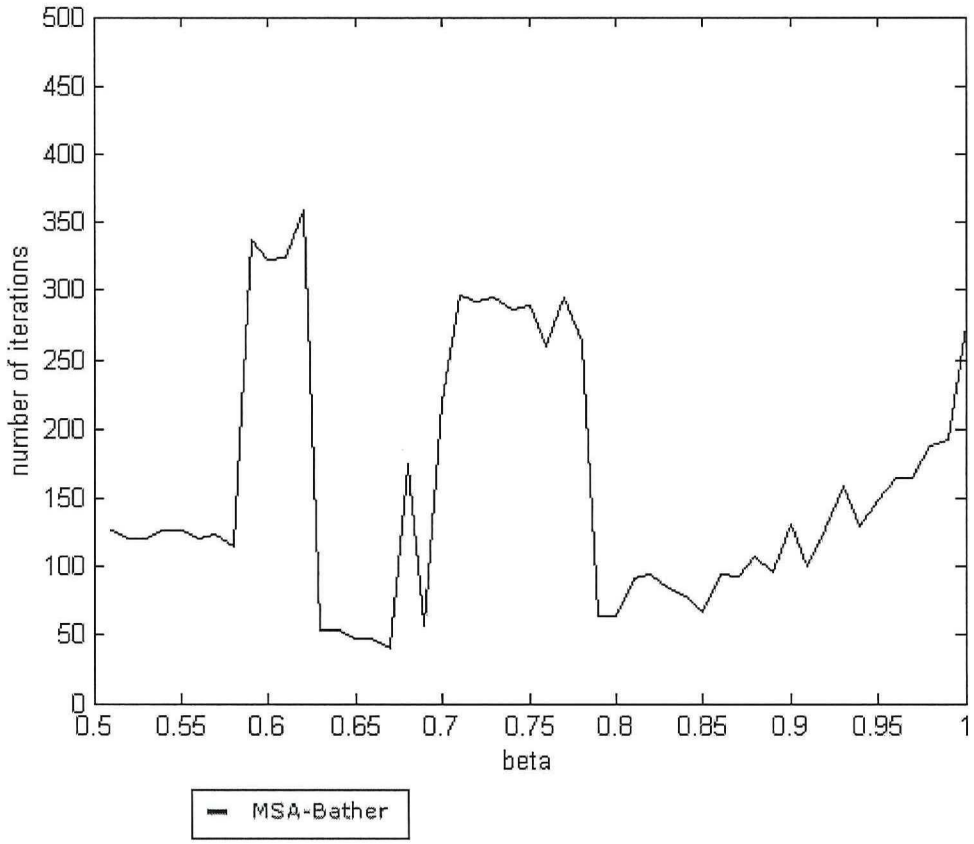

Figure: 6.7: Required number of iterations for the MSA-Bather method for given values of β

stop criterion	0.001
n-switch	10

Tabel 6.10: Values of the variables used for figure 6.7.

To solve the given traffic problem using the MSA-Bather method the stop criterion is set to 0.001, p to 1 and β to 0.67. The outcomes are given in figure 6.8 and table 6.11.

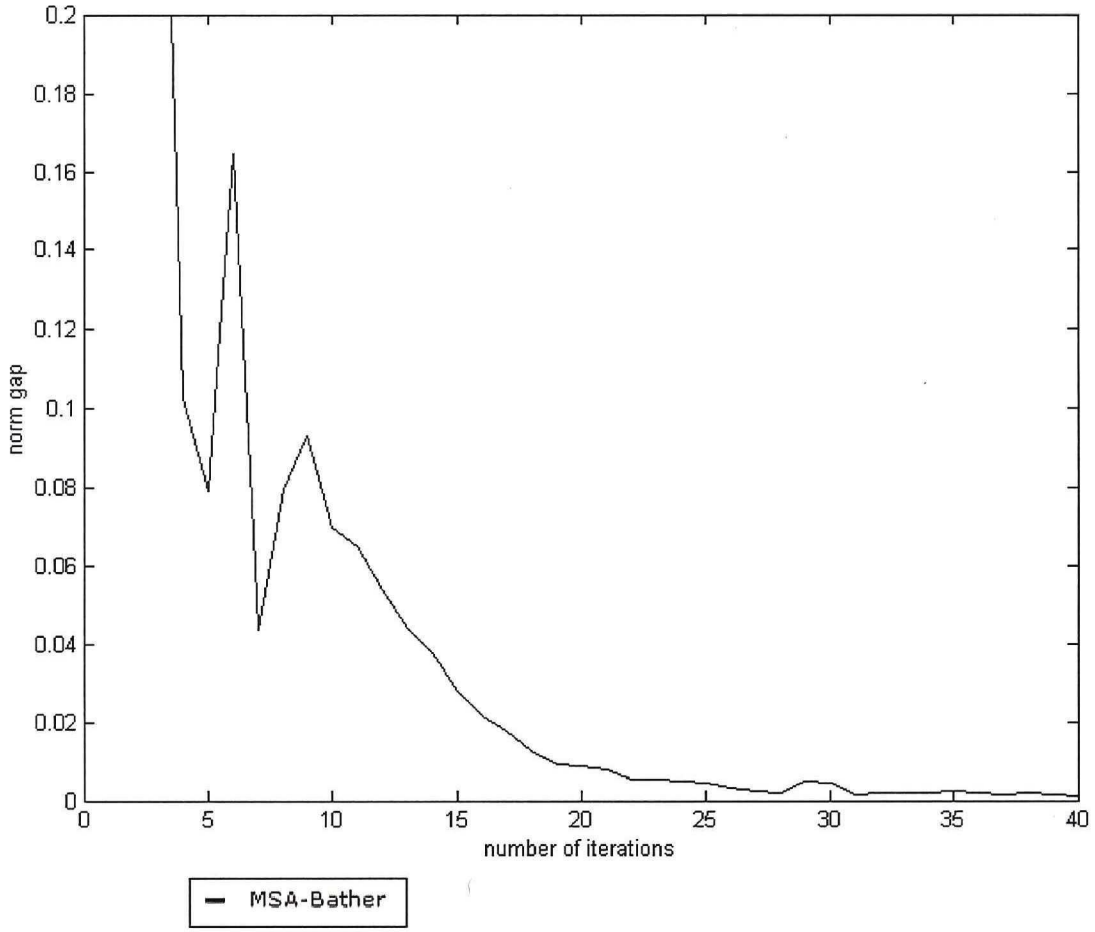

Figure 6.8: The normalised gap value with increasing number of iterations when the problem is solved with the MSA-Bather method in the DTA algorithm

	MSA-	**Bather**
time periods	*route 1*	*route 2*
	route costs	
1	2,2758	2,2745
2	3,5901	3,5944
3	6,2827	6,2383
4	7,0594	7,5234
	route flows	
1	11,2950	3,7050
2	11,4643	8,5357
3	3,2077	11,7923
4	19,7654	0,2346

Table 6.11: Route costs and flows of the problem solved with the MSA-Bather method in the DTA algorithm (see figure 6.8).

stop criterion	0.001
β	0.67
n-switch	10

Tabel 6.12: Values of the variables used for figure 6.8.

The normalised gap value for every iteration of the solution algorithm is shown in figure 6.8. The normalised gap goes to zero and when the normalised gap is smaller than the stop criterion the program is stopped. Using the MSA-Bather method there are 40 iterations needed to reach the stop criterion. The values of the route costs and flows of the two routes are shown in table 6.4. In the first column the route costs and number of travellers in the Equilibrium State of route 1 are shown and in the second column the route costs and number of travellers in the Equilibrium State of route 2 are shown. It appears that the route costs of the

first three time periods are almost identical. In the fourth time period the route costs differ. This is caused by the fact that the value for the most expensive route in the last time period in the equilibrium goes to zero.

6.5 The Bliemer-Bather method

This method first starts with the Bliemer method (see section 6.1) and after a certain number of iterations there is a switch to the Bather method (see section 5.3).

6.5.1 A numerical example of the Bliemer-Bather method

The traffic problem with three cities and two routes described in subsection 4.1.1. is considered. In the Bliemer-Bather method the adjustment parameter α is computed as follows:

$$\alpha^k = p \cdot k^{-\beta}, \tag{6.10}$$

where k is the number of iterations and β can be chosen between values of 0.5 and 1.0. In figure 6.9 the number of iterations in which the problem is solved using the Bliemer-Bather method with n-switch equal to 13 is expanded against β. Therefore, the best value for β can be chosen for the calculation. According to figure 6.9 the best value for β is 0.61.

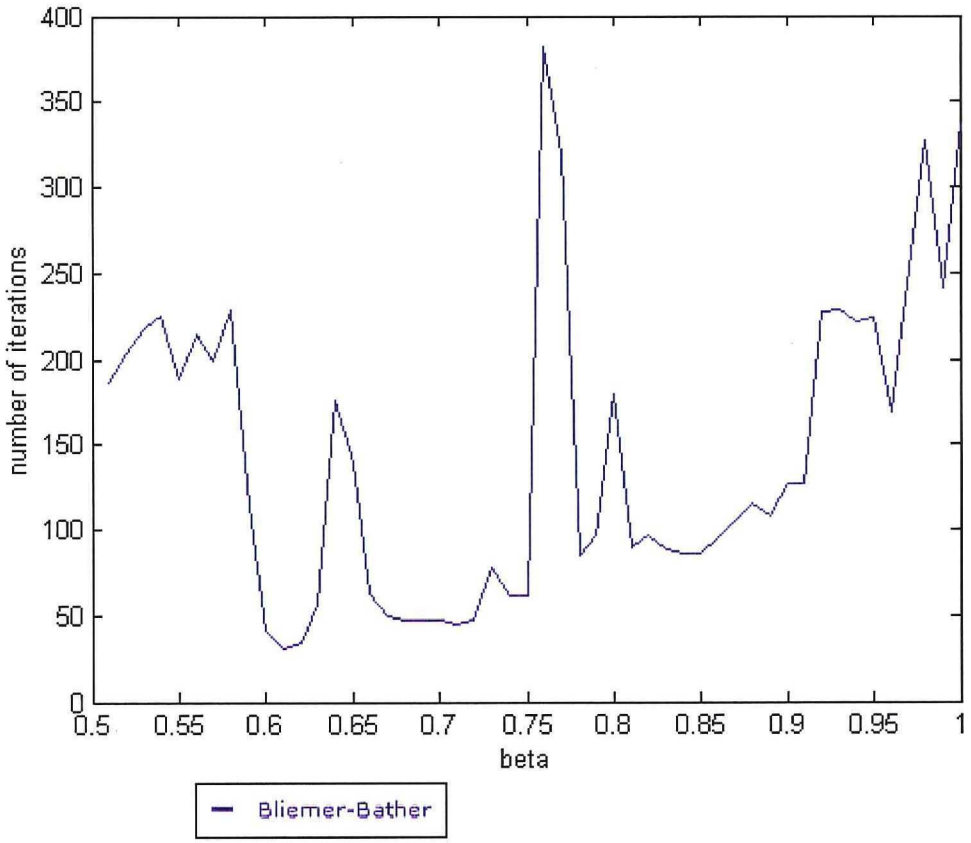

Figure 6.9: Required number of iterations for the Bliemer-Bather method for given values of β

stop criterion	0.001
n-switch	13

Tabel 6.13: Values of the variables used for figure 6.9.

To solve the given traffic problem using the Bliemer-Bather method the stop criterion is set to 0.001, p to 1 and β to 0.61. The outcome can be found in figure 6.10 and table 6.14.

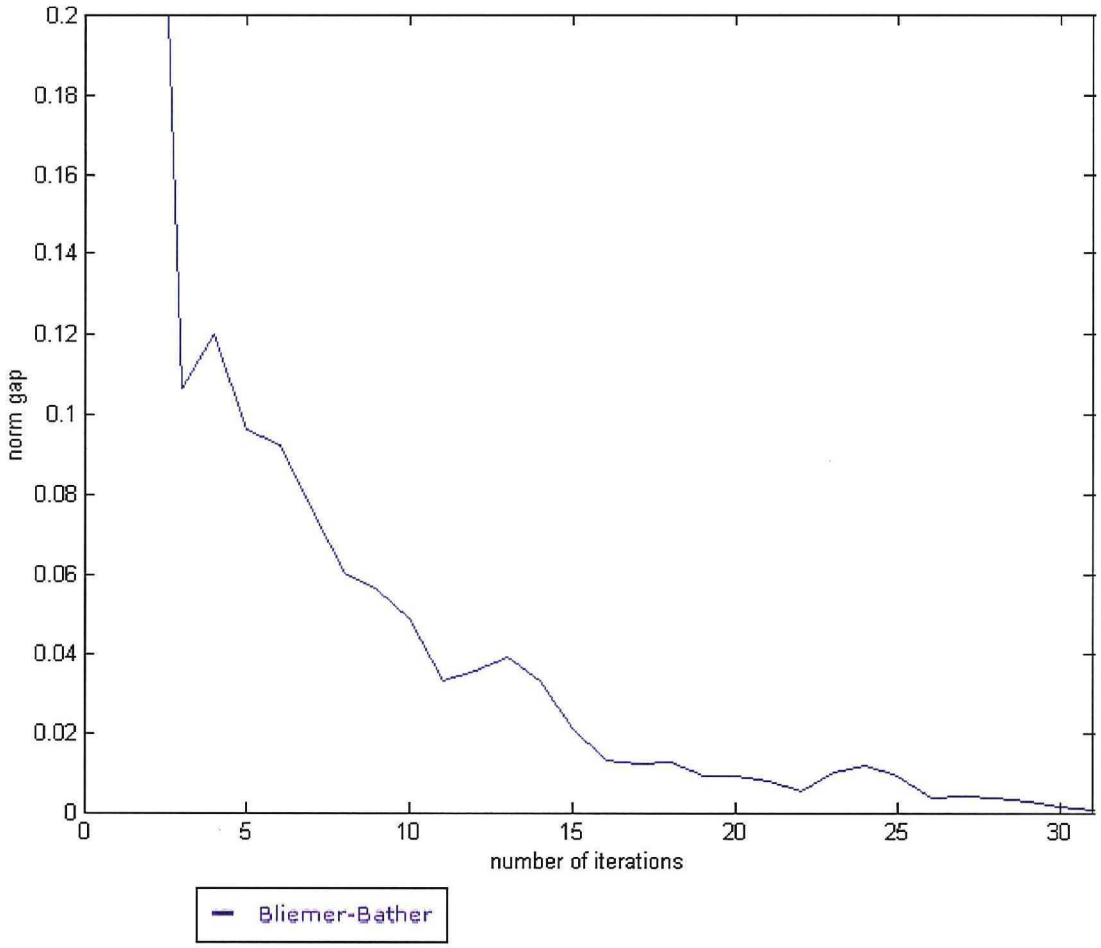

Figure 6.10: The normalised gap value with increasing number of iterations when the problem is solved with the Bliemer-Bather method in the DTA algorithm

	Bliemer-	**Bather**
time periods	*route 1*	*route 2*
	route costs	
1	2,2730	2,2763
2	3,5911	3,5897
3	6,2794	6,2728
4	7,0052	7,5717
	route flows	
1	11,2829	3,7171
2	11,4808	8,5192
3	3,1230	11,8770
4	19,6787	0,3213

Table 6.14: Route costs and flows of the problem solved with the Bliemer-Bather method in the DTA algorithm (see figure 6.10).

stop criterion	0.001
β	0.61
n-switch	13

Tabel 6.15: Values of the variables used for figure 6.10.

The normalised gap value for every iteration of the solution algorithm is shown in figure 6.10. The normalised gap goes to zero and when the normalised gap is smaller than the stop criterion the program is stopped. Using the Bliemer-Bather method there are 31 iterations needed to reach the stop criterion. The values of the route costs and flows of the two routes are shown in table 6.5. In the first column the route costs and number of travellers in the Equilibrium State of route 1 are shown and in the second column the route costs and number of travellers in the Equilibrium State of route 2 are shown. It appears that the route costs of the first three time periods are almost identical. In the fourth time period the route costs differ. This is caused by the fact that the value for the most expensive route in the last time period in the equilibrium goes to zero.

7. Comparison of developed methods

In this chapter the various methods presented before (MSA, Polyak, Bather, Bliemer, Bliemer Moving, MSA-Bliemer, MSA-Bather and Bliemer-Bather method) are compared. First MSA will be compared with the Polyak method (section 7.1). In section 7.2 the Polyak method will be compared with the Bather method. In section 7.3 the Bather method is compared with the Bliemer method and the Bliemer Moving method. In chapter 6 combinations of method are made, such as the MSA-Bliemer method (section 6.3), the MSA-Bather method (section 6.4) and the Bliemer-Bather method (section 6.5). These methods will be compared in section 7.4. In section 7.5 all the methods are shown at decreasing values of the stop criterion.

7.1 Comparison of MSA and the Polyak method

The traffic problem with three cities and two routes described in subsection 4.1.1 is also considered for comparison of the methods described in this report. To compare MSA and the Polyak method the stop criterion is set to 0.001, p to 1 and the value of β for the Polyak method is set to 0.70. The outcome can be found in figure 7.1 and table 7.1.

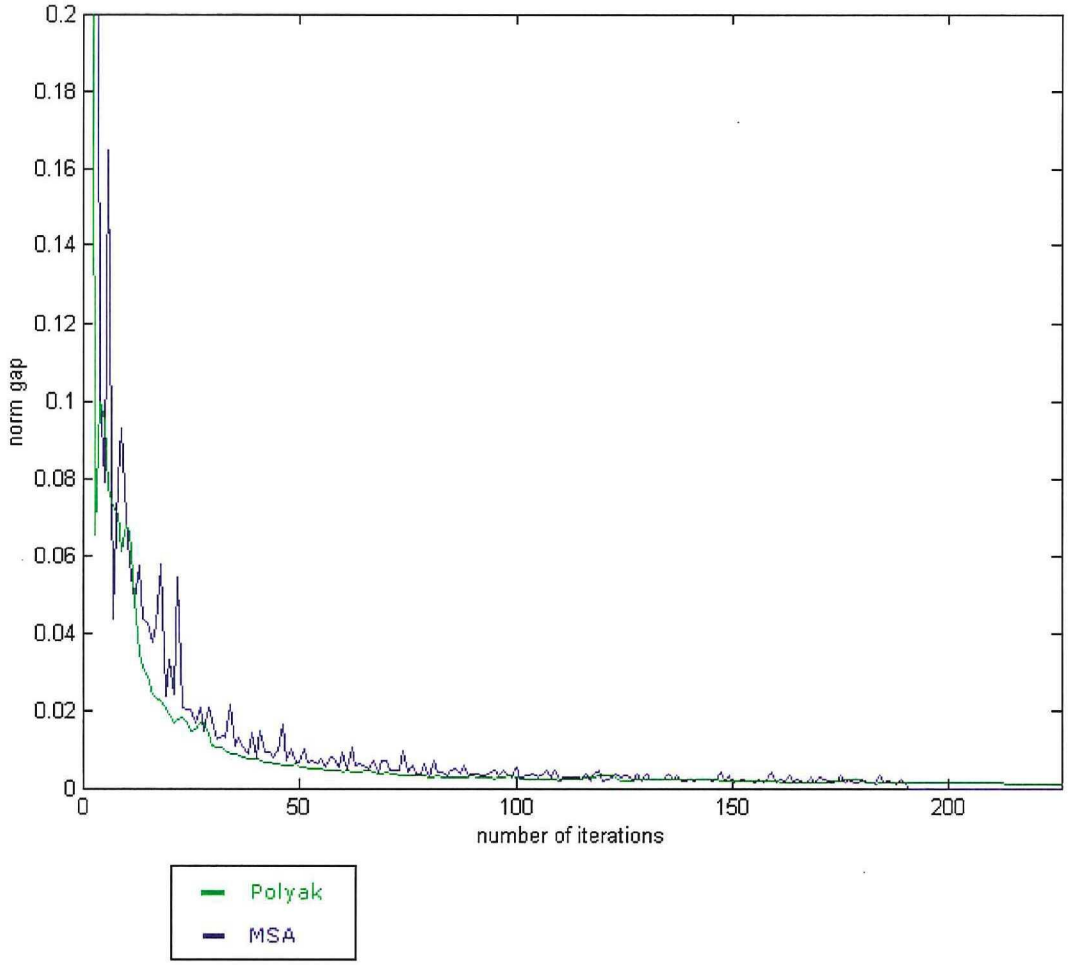

Figure 7.1: Comparison of the convergence of MSA and the Polyak method

	MSA			**Polyak**	
time periods	*route 1*	*route 2*		*route 1*	*route 2*
	route costs			*route costs*	
1	2,2745	2,2754		2,2515	2,2908
2	3,5910	3,5916		3,5795	3,5823
3	6,2813	6,2587		6,2558	6,2553
4	6,9742	7,5529		7,024	7,5502
	route flows			*route flows*	
1	11,2895	3,7105		11,1871	3,8129
2	11,4737	8,5263		11,5238	8,4762
3	3,1579	11,8421		3,1291	11,8709
4	19,5789	0,4211		19,6898	0,3102

Table 7.1: Route costs and flows of the problem solved with MSA and the Polyak method in the DTA algorithm (see figure 7.1).

stop criterion	0.001
β	0.70

Tabel 7.2: Variables used for figure 7.1.

After 190 iterations of MSA and after 226 iterations of the Polyak method the stop criterion is reached. The values of the route costs and the values of the route flows of the two routes for MSA and the Polyak method are shown in table 7.1. From table 7.1 one can see that the results are almost identical. In this case the Polyak method is not faster than MSA.

7.2 Comparison of the Polyak and Bather method

The traffic problem with three cities and two routes described in subsection 4.1.1 is also considered for the comparison of the methods described in this report. To compare the Polyak method and the Bather method the stop criterion is set to 0.001, p to 1, the value of β for the Polyak method is set to 0.70 and the value of β for the Bather method is set to 0.62. The outcome can be found in figure 7.2 and table 7.3.

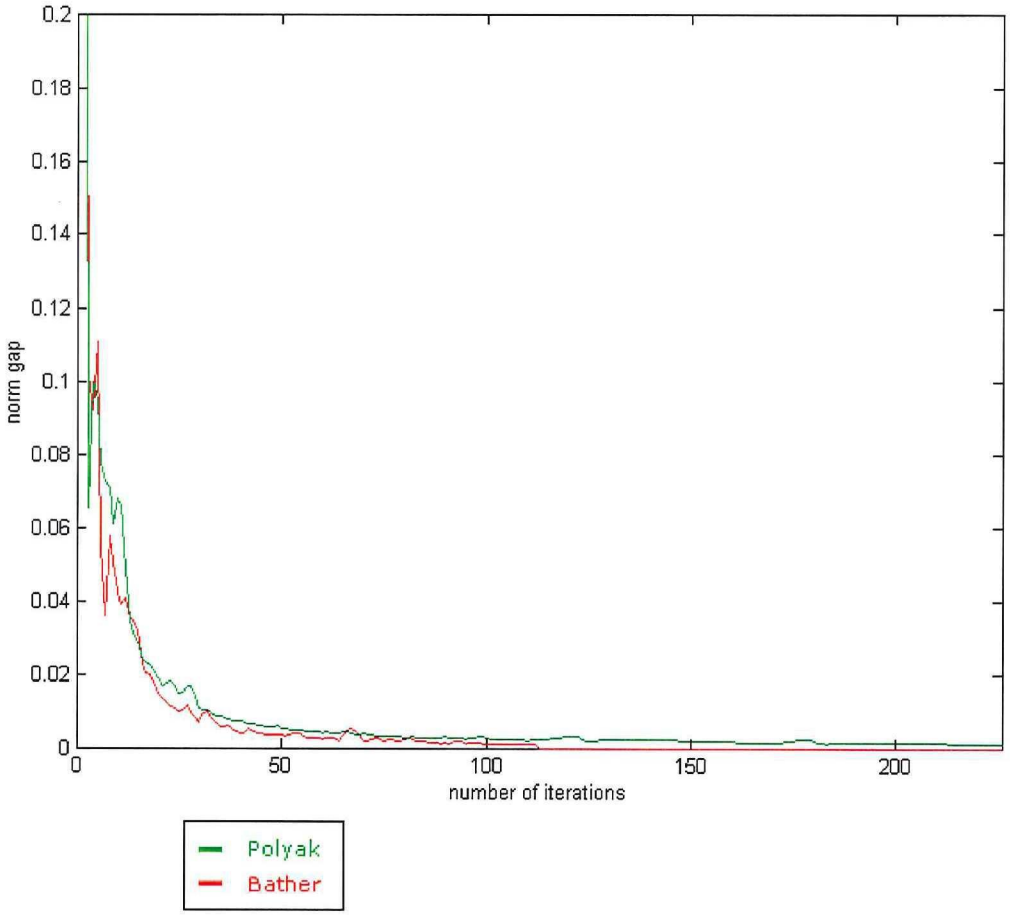

Figure 7.2: Comparison of the convergence of the Polyak method and the Bather method

	Polyak			**Bather**	
time periods	*route 1*	*route 2*		*route 1*	*route 2*
	route costs			*route costs*	
1	2,2515	2,2908		2,2710	2,2777
2	3,5795	3,5823		3,5888	3,5908
3	6,2558	6,2553		6,2752	6,2693
4	7,0240	7,5502		6,9636	7,5677
	route flows			*route flows*	
1	11,1871	3,8129		11,2740	3,7260
2	11,5238	8,4762		11,4794	8,5205
3	3,1291	11,8709		3,1313	11,8687
4	19,6898	0,3102		19,5674	0,4326

Table 7.3: Route costs and flows of the problem solved with the Polyak method and the Bather method in the DTA algorithm (see figure 7.2).

stop criterion	0.001
β (polyak)	0.70
β (bather)	0.62

Tabel 7.4: Variables used for figure 7.2.

After 226 iterations of the Polyak method and after 118 iterations of the Bather method the stop criterion is reached. The values of the route costs and the values of the route flows of the two routes for the Polyak method and the Bather method are shown in table 7.3. From table 7.3 one can see that the results are almost identical.

7.3 Comparison of the Bather, Bliemer and Bliemer Moving method

The traffic problem with three cities and two routes described in subsection 4.1.1 is also considered for the comparison of the methods described in this report. To compare the Bather method, the Bliemer method and the Bliemer Moving method the stop criterion is set to 0.001, p to 1, the value of β of the Bather method is set to 0.62, the value of β of the Bliemer (Moving) method is set to 0.54 and M is set to 30. The outcome can be found in figure 7.3 and table 7.5.

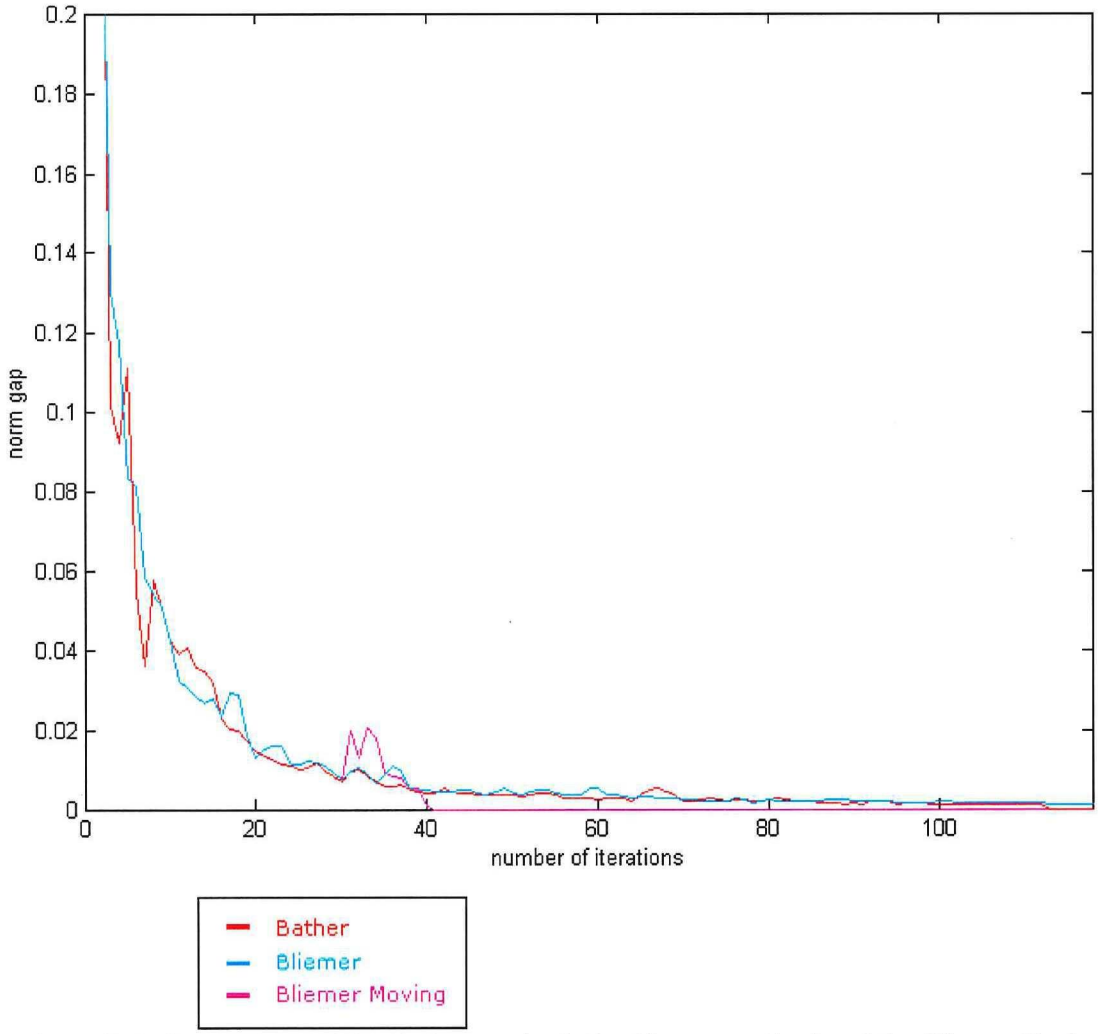

Figure 7.3: Comparison of the Bather method, the Bliemer method and the Bliemer Moving method

	Bliemer			**Bliemer Moving**	
time periods	*route 1*	*route 2*		*route 1*	*route 2*
	route costs			*route costs*	
1	2,2748	2,2752		2,2740	2,2757
2	3,5917	3,5910		3,5943	3,5861
3	6,2803	6,2760		6,2871	6,2594
4	6,9426	7,5769		7,1123	7,5530
	route flows			*route flows*	
1	11,2909	3,7091		11,2872	3,7128
2	11,4753	8,5247		11,4905	8,5095
3	3,1203	11,8797		3,1443	11,8557
4	19,5251	0,4749		19,9261	0,0739

Table 7.5: Route costs and flows of the problem solved with the Bliemer method and the Bliemer Moving method in the DTA algorithm (see figure 7.3).

stop criterion	0.001
β (bather)	0.62
β (bliemer (moving))	0.54
moving	30

Tabel 7.6: Variables used for figure 7.3.

After 118 iterations of the Bather method, after 112 iterations of the Bliemer method and after 40 iterations of the Bliemer Moving method the stop criterion is reached. The values of the route costs and the values of the route flows of the two routes for the Bather method are shown in table 7.3 and the values for the Bliemer method and the Bliemer Moving method are shown in table 7.5. From table 7.3 and table 7.5 one can see that the results are almost identical.

7.4 Comparison of combined methods

The traffic problem with three cities and two routes described in subsection 4.1.1 is also considered for the comparison of the methods described in this report. To compare the MSA-Bliemer method, the MSA-Bather method and the Bliemer-Bather method the stop criterion is set to 0.001, p to 1, the value of β of the MSA-Bliemer method is set to 0.54, the value of β of the MSA-Bather method is set to 0.67 and the value of β of the Bliemer-Bather method is set to 0.61. In these three methods there is another variable called n-switch. A value for n-switch of MSA-Bliemer (33), a value for n-switch of MSA-Bather (10) and a value for n-switch of Bliemer-Bather (13) are also taken. The outcome can be found in figure 7.4 and table 7.7.

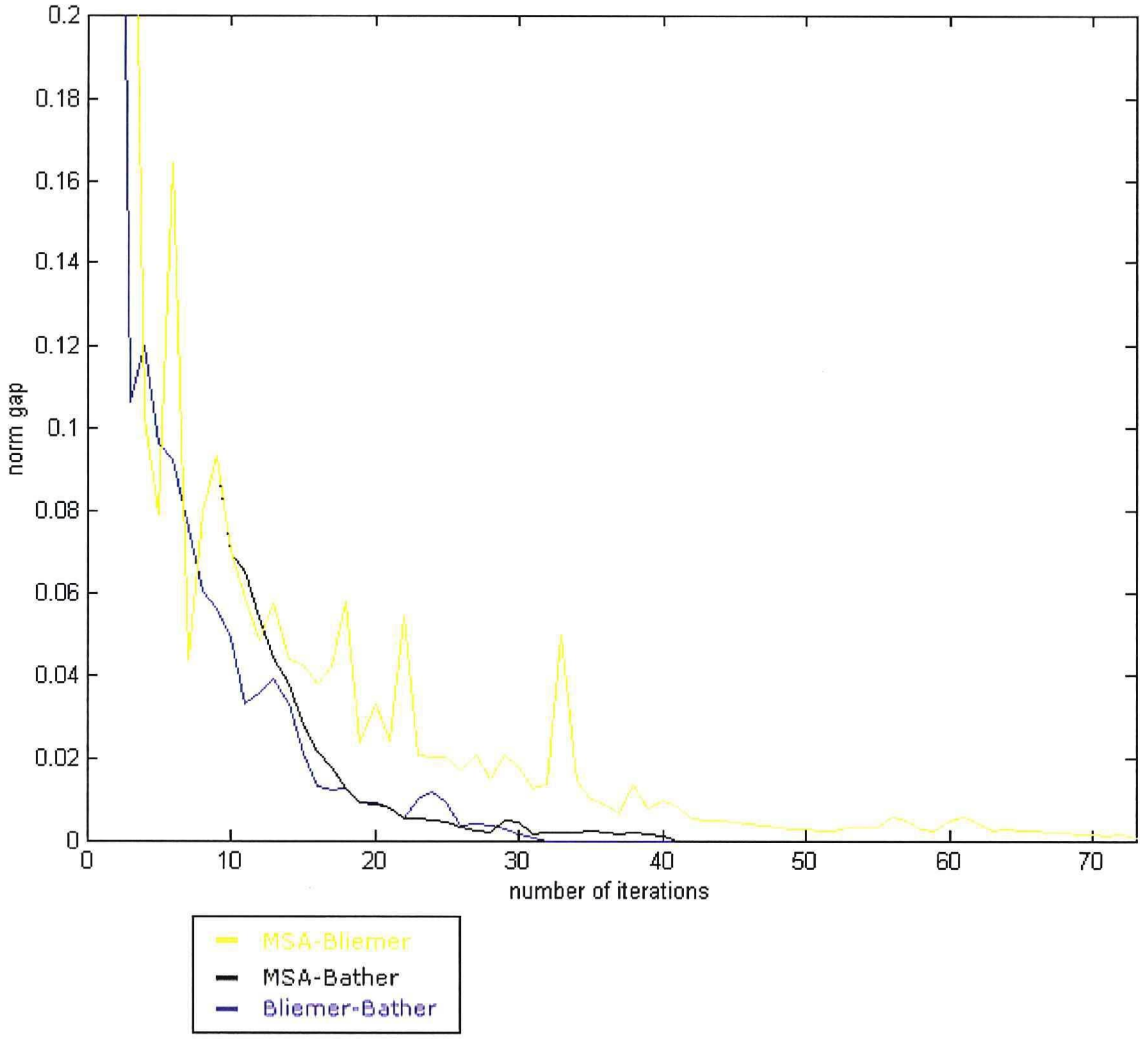

Figure 7.4: Comparison of the MSA-Bliemer method, the MSA-Bather method and the Bliemer-Bather method

	MSA-	**Bliemer**		**MSA-**	**Bather**		**Bliemer-**	**Bather**
time periods	*route 1*	*route 2*		*route 1*	*route 2*		*route 1*	*route 2*
	route costs			*route costs*			*route costs*	
1	2,2738	2,2758		2,2758	2,2745		2,2730	2,2763
2	3,5909	3,5910		3,5901	3,5944		3,5911	3,5897
3	6,2790	6,2739		6,2827	6,2383		6,2794	6,2728
4	6,9825	7,5734		7,0594	7,5234		7,0052	7,5717
	route flows			*route flows*			*route flows*	
1	11,2863	3,7137		11,2950	3,7050		11,2829	3,7171
2	11,4764	8,5236		11,4643	8,5357		11,4808	8,5192
3	3,1238	11,8762		3,2077	11,7923		3,1230	11,8770
4	19,6236	0,3764		19,7654	0,2346		19,6787	0,3213

Table 7.7: Route costs and flows of the problem solved with the MSA-Bliemer method, the MSA-Bather method and the Bliemer-Bather method in the DTA algorithm (see figure 7.4).

stop criterion	0.001
β (msa-bliemer)	0.54
β (msa-bather)	0.67
β (bliemer-bather)	0.61
n-switch (msa-bliemer)	33
n-switch (msa-bather)	10
n-switch (bliemer-bather)	13

Tabel 7.8: Variables used for figure 7.4.

After 73 iterations of the MSA-Bliemer method, after 40 iterations of the MSA-Bather method and after 31 iterations of the Bliemer-Bather method the stop criterion is reached. The values of the route costs and the values of the equilibrium of the two routes for the MSA-Bliemer method, the MSA-Bather method and the Bliemer-Bather method are shown in table 7.7. From table 7.7 one can see that the results are almost identical.

7.5 Comparison of all previous described methods (considering the stop criterion)

The traffic problem with three cities and two routes described in subsection 4.1.1 is also considered for the comparison of the methods described in this report with a different stop criterion. The stop criterions taken are 0.05, 0.01, 0.001 and 0.0005. For the outcome see figure 7.5 and table 7.9.

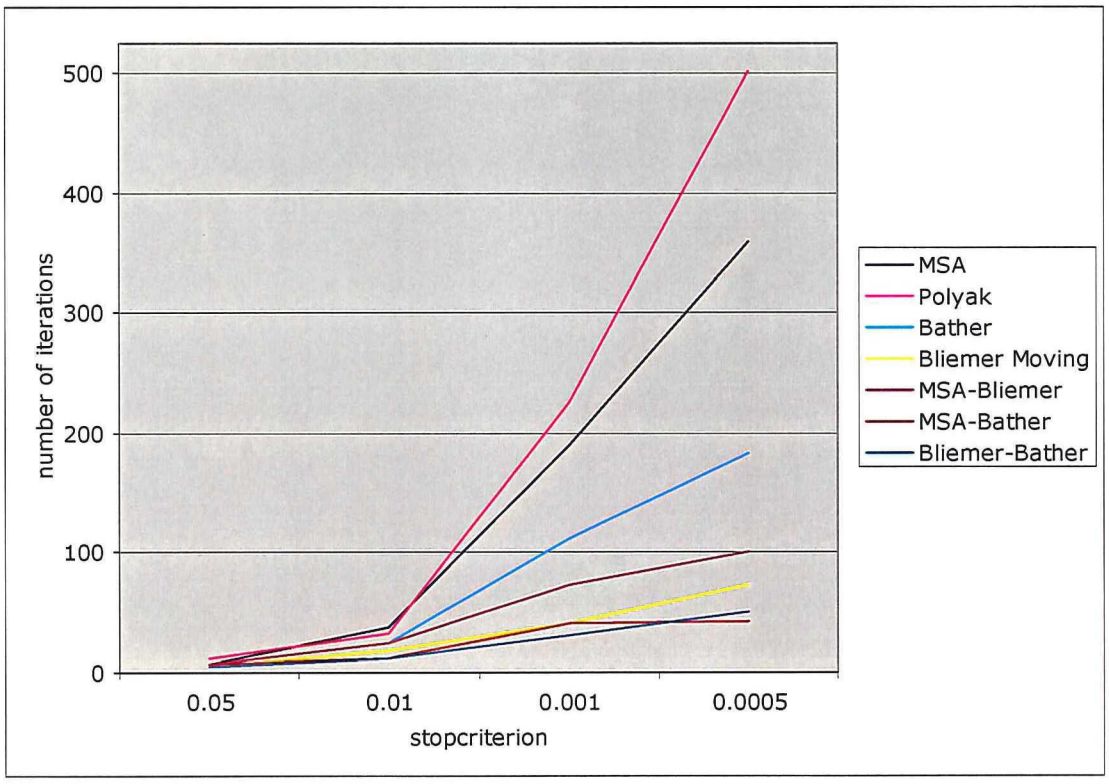

Figure 7.5: All methods (MSA, Polyak, Bather, Bliemer, Bliemer Moving, MSA-Bliemer, MSA-Bather and Bliemer-Bather method) at decreasing values of the stop criterion

Method / SC	*SC = 0.05*	*SC = 0.01*	*SC = 0.001*	*SC = 0.0005*
MSA	7	38	190	359
Polyak	12	33	226	503
Bather	6	25	112	184
Bliemer	10	27	118	212
Bliemer Moving	5	18	40	73
MSA-Bliemer	6	25	73	101
MSA-Bather	6	12	40	43
Bliemer-Bather	5	12	31	50

Table 7.9: Number of iterations in which the problem is solved for different methods and different stop criterions (SC) (see figure 7.5).

According to figure 7.5 it seems that there is no big difference in the number of iterations of the different methods considering a stop criterion of 0.05. But it's better not to use the Polyak method or the Bliemer method, because they converge slowly. The number of iterations of these methods are respectively 12 and 10 and for the other methods the number of iterations is 5, 6 or 7. For the stop criterion of 0.01 the two methods which converge as fastest are the MSA-Bather method (12 iterations) and the Bliemer-Bather method (12 iterations). The Bliemer Moving method has also good performance for a stop criterion of 0.01, but MSA converges very slowly. For the stop criterion of 0.001 the method which converges as fasts is the Bliemer-Bather method (31 iterations). The Bliemer Moving method (40 iterations) and the MSA-Bather method (40 iterations) also have good performances. The method which needs the greatest number of iterations is again the Polyak Method (226 iterations). The method which converges as fasts for the smallest stop criterion (0.0005) is the MSA-Bather method (43 iterations) and the method with the worst performance is the Polyak method (503 iterations). The Bliemer-Bather method (50 iterations) has good performance.

The conclusion is that the best methods are: the Bliemer Moving method, the MSA-Bliemer method, the MSA-Bather method and the Bliemer-Bather method. These methods are shown again in figure 7.6.

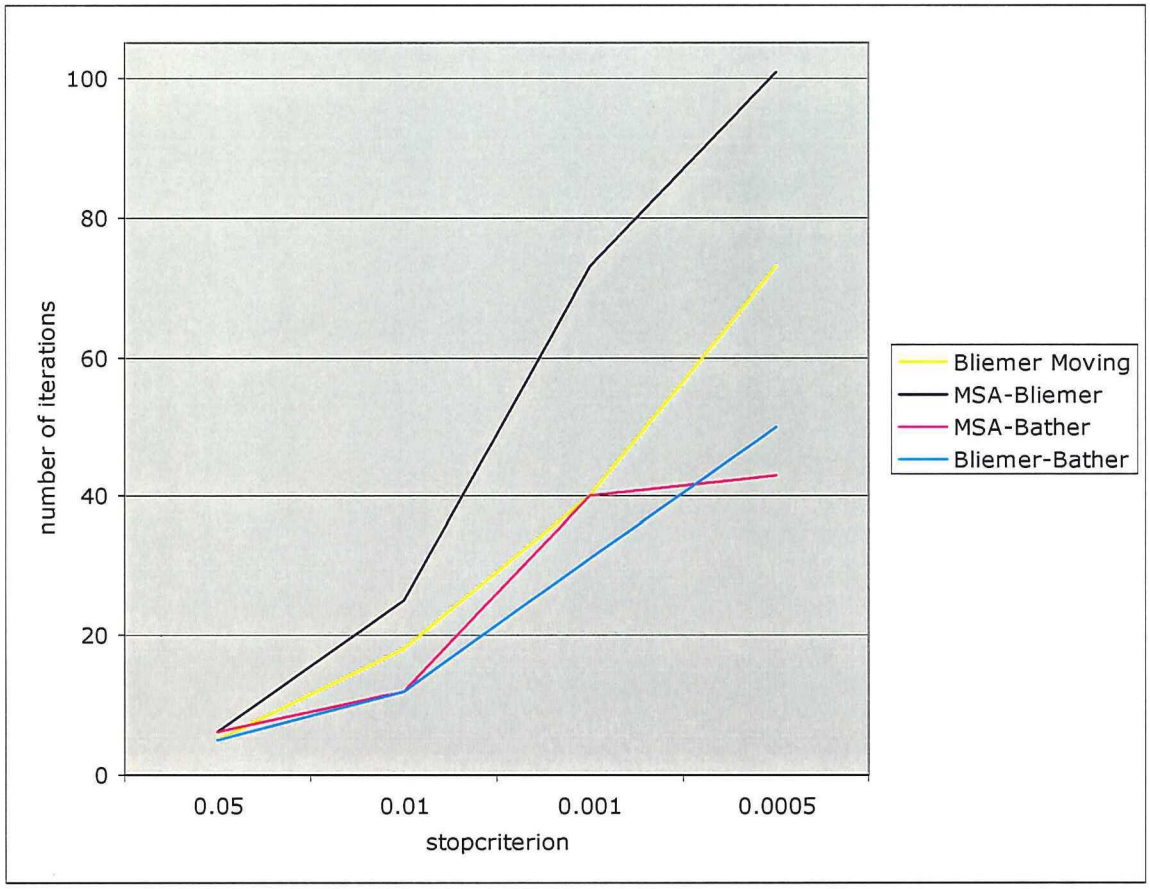

Figure 7.6: The best methods shown of figure 7.5 (Bliemer Moving, MSA-Bliemer, MSA-Bather and Bliemer-Bather method) at decreasing value of the stop criterion

According to figure 7.6 one conclusion can be derived. The methods with the best performance are: the MSA-Bather method and the Bliemer-Bather method. But, these methods are tested on a very simple traffic problem and for proving which method is the best, more extended research is needed using realistic networks of larger size.

After more insight in the algorithms of the methods the Bliemer-Bather method will also be a good candidate to do further research with. That's because of the variables used in the Bliemer-Bather method. The variables can be more diversified. By diversifying the variables the Bliemer-Bather method can become faster in convergence.

After further research on the values of the variables used in the Bliemer-Bather method is done, the recommendation may be changed to using the Bliemer-Bather method for solving NLP problems.

8. Conclusion and recommendations

8.1 Summary of findings

Methods for iteratively solving Non Linear Programming (NLP) problems with descent approaches are presented in this report. The methods are: the Method of Successive Averages (MSA), the Polyak method, the Bather method, the Bliemer method, the Bliemer Moving method, the MSA-Bliemer method, the MSA-Bather method and the Bliemer-Bather method.

The 'classical' fixed-point solution methods are often inappropriate for some problems. In such cases, the fixed-points are usually computed using one of the iterate averaging methods introduced by Robbins and Monro [3.1]. MSA, introduced by Sheffi and Powell [3.2] is probably the best-known and most widely-used instance. MSA computes each new design point by adding a part of the observation evaluated in the previous design point with a part of the previous design point.

B.T. Polyak and J.A. Bather proposed two relatively minor modifications of the iterate averaging method which were rigorously shown to produce fixed-point estimates with asymptotically optimal properties.

The Polyak method is a two-pass method. The first pass resembles MSA except that the step sizes are larger; this allows the algorithm to explore the solution space more aggressively but leads to greater variability in the outputs. The second pass is carried out offline (i.e., without influencing the first pass); it calculates an average of the iterates that are generated by the first pass. The average calculated by the second pass at termination is the fixed-point solution estimate.

A somewhat different approach was proposed by Bather [5.2]. Bather derives the new design point from a combination of the average of previous design points and the average of previous evaluation results.

To compare the performance of the various methods a traffic assignment problem with three cities and two routes is considered. The conclusion is that MSA is faster in convergence than the Polyak method for the relative gap stop criterion.

The other five iterate averaging methods are new alternative methods. One of those is proposed by Dr. M.C.J. Bliemer. The Bliemer method is almost similar to the Polyak method, with the difference that the observations are evaluated in the average of the previous design points and not at the previous design point as in the Polyak method. The Bliemer Moving method is the Bliemer method with a moving average.

The other three methods are combinations of MSA, the Bather method and the Bliemer method. The first combination is the MSA-Bliemer method, which first starts with MSA and after a certain number of iterations switches to the Bliemer method. The second method is the MSA-Bather method, which first starts with MSA and after a certain number of iterations switches to the Bather method. The Bliemer-Bather method first starts with the Bliemer method and after a certain number of iterations switches to the Bather method.

The Bather method, the Bliemer method and the Bliemer Moving method are compared. It appears that the Bather method and the Bliemer method solve the problem in almost the same number of iterations.

Figure 8.1 shows in how many iterations all the eight methods are solved at decreasing values of the stop criterion.

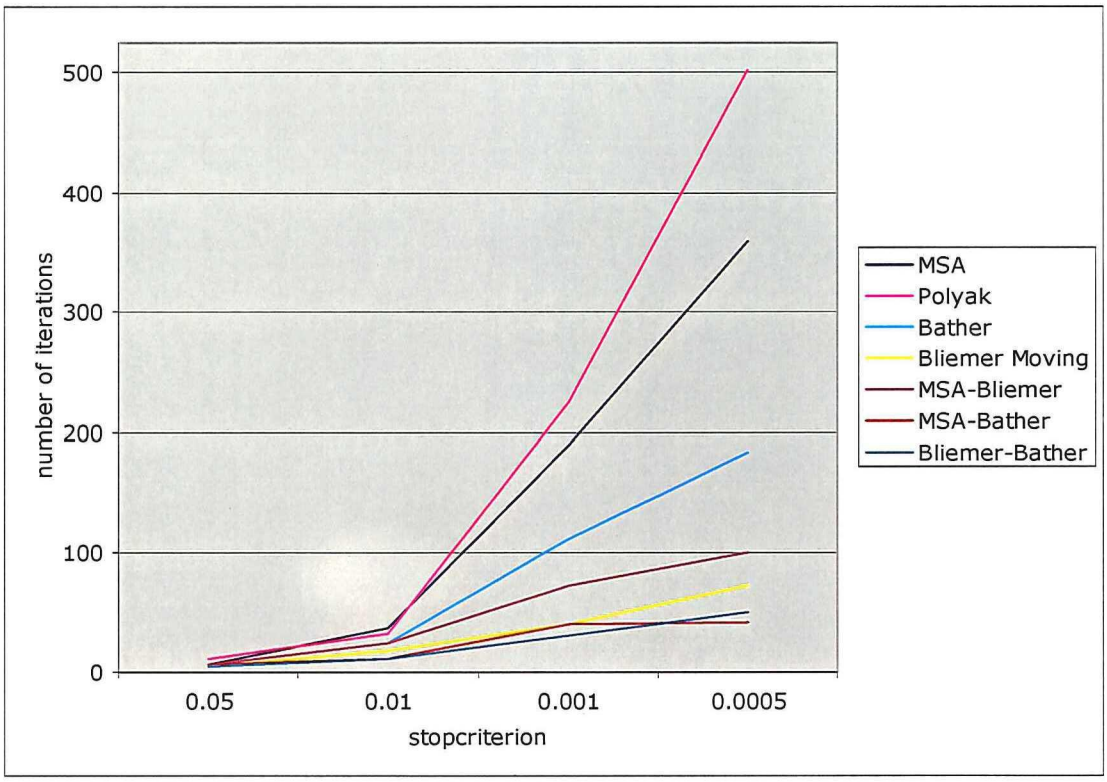

Figure 8.1: All methods (MSA, Polyak, Bather, Bliemer, Bliemer Moving, MSA-Bliemer, MSA-Bather and Bliemer-Bather method) at decreasing values of the stop criterion

8.2 Conclusions

On the whole, also for different values of stop criteria, the Bather method is faster in convergence than the Bliemer method. However the Bliemer Moving method is much faster than the four methods presented before.

By composing two methods it appears possible to get an even faster convergence. The fastest methods in convergence, for the stop criterion we chose, are the Bliemer-Bather combined method and the MSA-Bather combined method.

The conclusion is that there are alternative methods that are much faster in convergence than MSA and the Polyak method. The best alternative methods are the MSA-Bather method and the Bliemer-Bather combined method (see figure 8.2). This conclusion has to be verified for multiple realistic-size transport networks.

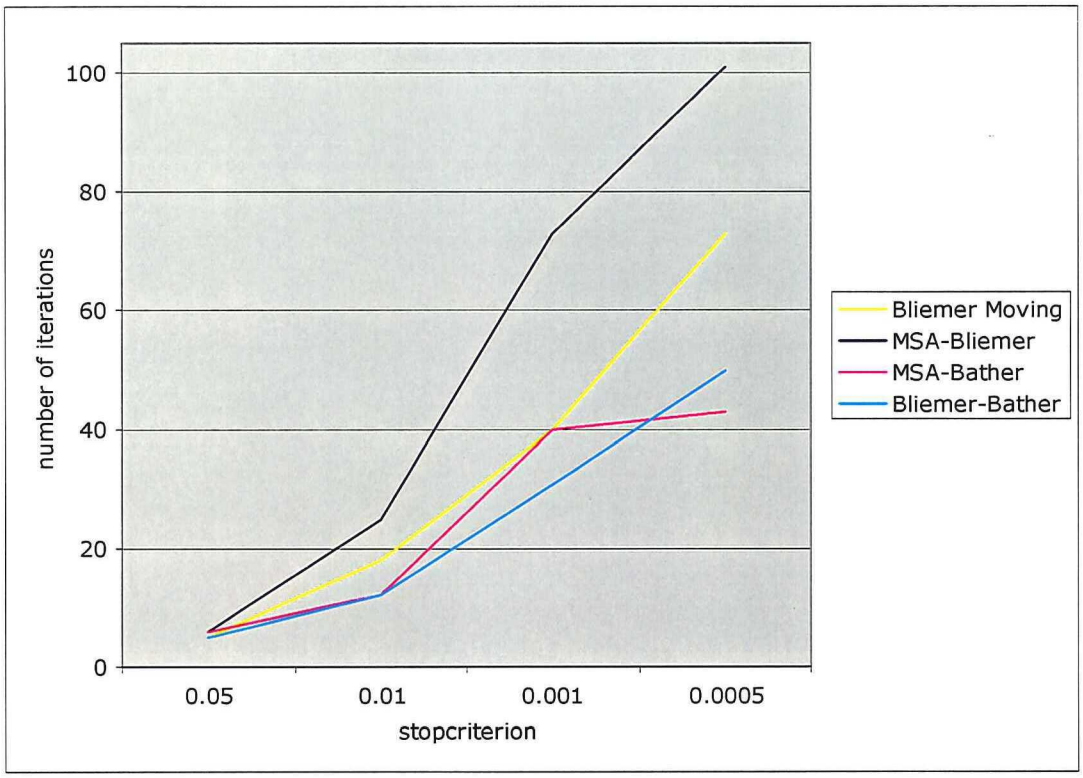

Figure 8.2: The best methods shown of figure 8.1 (Bliemer Moving, MSA-Bliemer, MSA-Bather and Bliemer-Bather method) at decreasing value of the stop criterion

8.3 Recommendations for further research

At this point the MSA-Bather method and the Bliemer-Bather method are recommended for solving NLP problems.

Further research is necessary to test these procedures on a more varied set of real-sized transport networks. The following questions emerge for further research:

- Do the demand and route costs have influence on the number of iterations?
- The programs have to be run again, but then with a different demand or different cost functions. As proposition for the demand and cost functions see (8.1) and (8.2).

  $d = [25\ 19\ 25\ 5]$ (8.1)

  $c = 1 + 0.03*u + 0.1*v$ (8.2)
- How to choose the stop criterion? Make it dependent of the demand or number of time periods.
- Why are the alternative methods that are faster than MSA?
- How to choose β? What is the best step size?
- How to choose the n-switch?
- How to choose the number of iterations used in the moving average?
- What is the best starting point for the algorithms?
- Is it good to choose always an 'All Or Nothing' (AON) assignment? Maybe start with the LOGIT assignment. The LOGIT assignment is defined in (8.3).

  $\text{LOGIT}(x) = \ln[\, x / (1-x)\,]$ (8.3)

- It's also possible to use the Frank-Wolfe assignment (see figure 8.3) [8.1].

Step 1 (Initialization):
- Take k=1, and perform an AON assignment based on $t_a=t_a(0)$. This yields flow vector $q^{(k)}_a$

Step 2 (Update link travel times):
- Compute $t_a=t_a(q^{(k)}_a)$ $\forall a$.

Step 3 (Determine descent direction):
- Perform an AON assignment based on t_a. This yields the auxiliary flow vector $w^{(k)}_a$.

Step 4 (Determine step size):
- Find $\alpha^{(k)}$ that solves:

$$\min \sum_a \int_0^{q_a^{(i)}+\alpha\left(w_a^{(i)}-q_a^{(i)}\right)} t_a(x)dx \qquad (8.4)$$

A simpler approach to this step is choosing $a^{(k)} = 1/k$. This approach is called MSA.

Step 5 (Move):
- Set $q^{(k+1)}_a = q^{(k)}_a + \alpha^{(k)}(w^{(k)}-q^{(k)}_a)$ $\forall a$.

Step 6 (Convergence test):
- If a certain predetermined convergence criterion is met, then stop. Otherwise, set k=k+1 and return to step 2.

Figure 8.3: The Frank-Wolfe assignment

- The iterate averaging methods can also be tested on a static transportation network problem.
- Take for all the methods the same step sizes and see what the best method is.

References

[5.2] Bather, J.A., *Stochastic approximation: A generalisation of the Robbins-Monro procedure*. In P. Mandl and M. Huskova, editors, Proceedings of the Fourth Prague Symposium on Asymptotic Statistics, pages: 13-27. Charles University, 1989.

[1.1] Bliemer, M.C.J., *Analytical dynamic traffic assignment with interacting user-classes: Theoretical Advances and Applications using a Variational inequality Approach*, T2001/1, January 2001, TRAIL Thesis Series, Delft University Press, The Netherlands.

[3.4] Blum, J.R., *Approximation methods which converge which probability one*. Annals of Mathematical Statistics, 25(2): 382-386, June 1954a.

[3.5] Blum, J.R., *Multidimensional stochastic approximation methods*. Annals of Mathematical Statistics, 25(4): 737-744, December 1954b.

[3.3] Bottom, J., *Consistent anticipatory route guidance*. PhD Thesis, Department of Civil and Environmental Engineering, Massachusettes Institute of Technology, Cambridge MA, USA, 2000.

[1.2] Bottom, J., and I. Chabini, *Extended abstract: Accelerated averaging methods for fixed-point problems in transportation analysis and planning*. USA.

[8.1] Bovy, P.H.L., and M.C.J. Bliemer. *Transportation Modelling*, Lecture Notes, Faculty of Civil Engineering and Geosciences, Transportation Planning and Traffic Engineering Section, Delft University of Technology, August 2002.

[4.2] Cascetta, E. and M.N. Postorino. *Fixed-point models for the estimation of OD matrices using traffic counts on congested networks*. Submitted to Transportation Science, 1998.

[5.3] Frees, E.W., and David Ruppert. *Estimation following a sequentially designed experiment*. Journal of the American Statistical Association, 85(412): 1123-1129, December 1990.

[2.1] He, Y., A flow-based approach to the dynamic traffic assignment problem: *Formulations, algorithms and computer implementations*. PhD thesis for the degree of Master of Science in Transportation, Massachusetts Institute of Technology, MIT Press, 1997.

[6.1] Kushner, H.J., and G.G. Yin. *Stochastic approximation algorithms and applications*. Number 35 in Applications of Mathematics – Stochastic Modelling and Applied Probability. Springer-Verlag, 1997.

[3.6] Polyak, B.T., *New method of stochastic approximation type*. Automation and Remote Control, 51(7): 937-946, July 1990.

[3.7] Polyak, B.T., and A.B. Juditsky. *Acceleration of stochastic approximation by averaging*. SIAM Journal of Control and Optimisation, 30(4): 838-855, July 1992.

[3.1] Robbins, H., and S. Monro. *A stochastic approximation method.* Annals of mathematical statistics, 22(3): 400-407, 1951.

[3.8] Schwabe, R., and Harro Walk. *On a stochastic approximation procedure based on averaging*. Metrika, 44(2): 165-180, 1996.

[3.2] Sheffi, Y. and W.B. Powell. *An algorithm for the equilibrium assignment problem with random link times*. Networks, 12:191-207, 1982.

A1. Implementation of the MSA algorithm

1.1 Description of the MSA algorithm

MSA computes the new design point (x^{k+1}) by adding a part of the previous design point (x^k) with a part of the observation evaluated in the previous design point ($T(x^k)$).

1.2 Formulation and solution algorithm of the MSA algorithm

To formulate the MSA algorithm the following variables and function are used:

- α the step size ($0<\alpha<1$)
- AON the 'All Or Nothing' assignment
- c the route costs
- d the demand
- k the iteration number
- ng the normalised duality gap
- s the stop criterion
- t the number of time periods
- T the observation
- x the route flows (link traversal time trajectories)

The mathematical formulation of MSA is:

$$x^{k+1} = x^k + \alpha^k \cdot \left[T(x^k) - x^k\right], \text{ with } x^0 \in \Re^n \text{ and k=1,2,3,...,} \quad \text{(A1.1)}$$

$$\alpha^k = p \cdot k^{-\beta}, \text{ with } 0.5<\beta\leq 1 \text{ and } p>0, \quad \text{(A1.2)}$$

The DTA algorithm with the MSA algorithm is described as follows (see figure A1.1):

Step 0 (Initialisation):
Initialisation of the variables: c, d, k, t and x.

Step 1 (Main loop):
k = k + 1
T = AON(c,d,t)
$\alpha = 1/k$
$x^{k+1} = x^k + \alpha^k \cdot \left[T(x^k) - x^k\right]$
$c(x^{k+1})$

Step 2 (Stop criterion):

$$ng = \frac{\sum_r \sum_i \left| \left[\left(\min\left\{ c_{ir}^r \left(x_{ir}^{i+1} \right) \right\} \right)^r \left(\min\left\{ c_{ir}^r \left(x_{ir}^{i+1} \right) \right\} \right)^r \right] - c_{ir}\left(x_{ir}^{i+1} \right) \right| \cdot x_{ir}^{i+1}}{\sum_r \sum_i c_{ir}\left(x_{ir}^{i+1} \right) \cdot x_{ir}^{i+1}}$$

If ng < s, then stop, else go to step 1.

Figure A1.1: The DTA algorithm with the MSA algorithm included

A2. Implementation of the Polyak algorithm

2.1 Description of the Polyak algorithm

The Polyak method computes the new design point (x^{k+1}) by adding a part of the previous design point (x^k) with a part of the observation evaluated in the previous design point ($T(x^k)$) just like MSA. But the Polyak method also computes 'in parallel' with and independently of the iterate averaging process the average of the design points, say $\bar{x}^k = \frac{1}{k} \cdot \sum_{i=1}^{k} x^i$. Here is meant that the computed solution estimates do not influence the determination of the design points. The sequence $\bar{x}^k$ also converges to the limit x^*. So this average is your estimate of the fixed-point solution.

2.2 Formulation and solution algorithm of the Polyak algorithm

To formulate the Polyak algorithm the following variables and function are used:

- α the step size ($0<\alpha<1$)
- AON the 'All Or Nothing' assignment
- c the route costs
- d the demand
- β the exponent ($0.5<\beta\leq 1$)
- k the iteration number
- ng the normalised duality gap
- s the stop criterion
- t the number of time periods
- T the observation
- x the route flows (link traversal time trajectories)
- $\bar{x}$ the average over all the previous route flows

The mathematical formulation of the Polyak method is:

$$x^{k+1} = x^k + \alpha^k \cdot \left[T(x^k) - x^k\right], \text{ with } x^0 \in \Re^n \text{ and k=1,2,3,...,} \quad \text{(A2.1)}$$

$$\bar{x}^k = \frac{1}{k} \cdot \sum_{i=1}^{k} x^i, \text{ with } x^0 \in \Re^n \text{ and k=1,2,3,...,} \quad \text{(A2.2)}$$

$$\alpha^k = p \cdot k^{-\beta}, \text{ with } 0.5<\beta\leq 1 \text{ and } p>0, \quad \text{(A2.3)}$$

The DTA algorithm with the Polyak algorithm is described as follows (see figure A2.1):

Step 0 (Initialisation):
Initialisation of the variables: c, d, β, k, t, x and X.

Step 1 (Main loop):
$k = k + 1$
$T = AON(c,d,t)$
$\alpha = k^{-\beta}$
$x^{k+1} = x^k + \alpha^k \cdot \left[T\left(x^k\right) - x^k\right]$
$\bar{x}^{k+1} = \bar{x}^k \cdot (k-1)/k + x^{k+1}/k$
$c(x^{k+1})$
$c(\bar{x}^{k+1})$

Step 2 (Stop criterion):

$$ng = \frac{\sum_r \sum_i \left[\left[\min\left\{c_{ir}^r\left(\bar{x}_{ir}^{i+1}\right)\right\}\right]^r \left[\min\left\{c_{ir}^r\left(\bar{x}_{ir}^{i+1}\right)\right\}\right]^r\right] - c_{ir}\left(\bar{x}_{ir}^{i+1}\right) \cdot \bar{x}_{ir}^{i+1}}{\sum_r \sum_i c_{ir}\left(\bar{x}_{ir}^{i+1}\right) \cdot \bar{x}_{ir}^{i+1}}$$

If ng < s, then stop, else go to step 1.

Figure A2.1: The DTA algorithm with the Polyak algorithm included

A3. Implementation of the Bather algorithm

3.1 Description of the Bather algorithm

The design point (x^{k+1}) is a part of the average of previous design points ($\bar{x}^{k}$) minus a part of the average of previous observations (τ^{k}). The function evaluations are made at the design points (x^{k}) while the fixed-point is estimated by $\bar{x}^{k}$.

3.2 Formulation and solution algorithm of the Bather algorithm

To formulate the Bather algorithm the following variables and function are used:

- α the step size ($0<\alpha<1$)
- AON the 'All Or Nothing' assignment
- c the route costs
- d the demand
- β the exponent ($0.5<\beta\leq 1$)
- k the iteration number
- ng the normalised duality gap
- s the stop criterion
- τ the average of the previous observations
- t the number of time periods
- T the observation
- x the route flows (link traversal time trajectories)
- $\bar{x}$ the average over all the previous route flows

The mathematical formulation of the Bather method is:

$$x^{k+1} = \bar{x}^{k} - k \cdot \alpha \cdot \left(\tau^{k} - \bar{x}^{k}\right), \text{ with } x^{0} \in \Re^{n} \text{ and k=1,2,3,...,} \quad \text{(A3.1)}$$

$$\bar{x}^{k} = \frac{1}{k} \cdot \sum_{i=1}^{k} x^{i} \text{ , with } x^{0} \in \Re^{n} \text{ and k=1,2,3,...,} \quad \text{(A3.2)}$$

$$\tau^{k} = \frac{1}{k} \cdot \sum_{i=1}^{k} T\left(x^{i}\right), \text{ with } x^{0} \in \Re^{n} \text{ and k=1,2,3,...,} \quad \text{(A3.3)}$$

$$\alpha^{k} = p \cdot k^{-\beta} \text{ , with } 0.5<\beta\leq 1 \text{ and p>0,} \quad \text{(A3.4)}$$

Or, with $(k+1)\cdot \bar{x}^{k+1} = k\cdot \bar{x}^{k} + x^{k+1}$:

$$\bar{x}^{k+1} = \bar{x}^{k} + {}^{k}\!/_{k+1}\cdot \alpha^{k}\cdot\left(\tau^{k} - \bar{x}^{k}\right), \text{ with } x^{0} \in \Re^{n} \text{ and k=1,2,3,...,} \quad \text{(A3.5)}$$

$$\text{and } \bar{x}^{k} = \frac{1}{k}\cdot\sum_{i=1}^{k} x^{i}, \text{ with } x^{0} \in \Re^{n} \text{ and k=1,2,3,...,} \quad \text{(A3.6)}$$

$$\text{and } \tau^{k} = \frac{1}{k}\cdot\sum_{i=1}^{k} T\left(x^{i}\right), \text{ with } x^{0} \in \Re^{n} \text{ and k=1,2,3,...,} \quad \text{(A3.7)}$$

$$\alpha^{k} = p\cdot k^{-\beta}, \text{ with } 0.5<\beta\leq 1 \text{ and } p>0, \quad \text{(A3.8)}$$

The DTA algorithm with the Bather algorithm is described as follows (see figure A3.1):

Step 0 (Initialisation):

Initialisation of the variables: c, d, β, k, τ, t and $\bar{x}$.

Step 1 (Main loop):

k = k + 1

T = AON(c,d,t)

$\tau^{k+1} = \tau^{k}\cdot {}^{(k-1)}\!/_{k} + T^{k+1}/_{k}$

$\alpha = k^{-\beta}$

$\bar{x}^{k+1} = \bar{x}^{k} + {}^{k}\!/_{(k+1)}\cdot \alpha^{k}\cdot\left[\tau^{k} - \bar{x}^{k}\right]$

c ($\bar{x}^{k+1}$)

Step 2 (Stop criterion):

$$ng = \frac{\sum_{c}\sum_{x}\left[\left[\min\left(c, \bar{x}^{k+1}\right)^{T}\left(\min\left(c, \bar{x}^{k+1}\right)\right)\right] - c\left(\bar{x}^{k+1}\right)\right]\cdot \bar{x}^{k+1}}{\sum_{c}\sum_{x} c\left(\bar{x}^{k+1}\right)\cdot \bar{x}^{k+1}}$$

If ng < s, then stop, else go to step 1.

Figure A3.1: The DTA algorithm with the Bather algorithm included

A4. Implementation of the Bliemer algorithm

4.1 Description of the Bliemer algorithm

The Bliemer method computes the new design point (x^{k+1}) by adding a part of the previous design point (x^k) with a part of the observation evaluated in the average of the previous design points ($T(\bar{x}^k)$). So the function evaluations are made at the average of the design points ($\bar{x}^k$) and the fixed-point is estimated by $\bar{x}^k$.

4.2 Formulation and solution algorithm of the Bliemer algorithm

To formulate the Bliemer algorithm the following variables and function are used:

- α the step size ($0<\alpha<1$)
- AON the 'All Or Nothing' assignment
- c the route costs
- d the demand
- β the exponent ($0.5<\beta\leq 1$)
- k the iteration number
- ng the normalised duality gap
- s the stop criterion
- t the number of time periods
- T the observation
- x the route flows (link traversal time trajectories)
- $\bar{x}$ the average over all the previous route flows x

The mathematical formulation of the Bliemer method is:

$$x^{k+1} = x^k + \alpha^k \cdot \left[T(\bar{x}^k) - x^k\right], \text{ with } x^0 \in \Re^n \text{ and k=1,2,3,...,} \quad \text{(A4.1)}$$

$$\text{and } \bar{x}^k = \frac{1}{k} \cdot \sum_{i=1}^{k} x^i \text{, with } x^0 \in \Re^n \text{ and k=1,2,3,...,} \quad \text{(A4.2)}$$

$$\alpha^k = p \cdot k^{-\beta} \text{, with } 0.5<\beta\leq 1 \text{ and } p>0, \quad \text{(A4.3)}$$

The DTA algorithm with the Bliemer algorithm is described as follows (see figure A4.1):

Step 0 (Initialisation):

Initialisation of the variables: c, d, β, k, t, x and $\bar{x}$.

Step 1 (Main loop):

$k = k + 1$

$T = AON(c,d,t)$

$\alpha = k^{-\beta}$

$x^{k+1} = x^{k} + \alpha^{k} \cdot \left[T(\bar{x}^{k}) - x^{k} \right]$

$\bar{x}^{k+1} = \bar{x}^{k} \cdot (k-1)/k + x^{k+1}/k$

$c(\bar{x}^{k+1})$

Step 2 (Stop criterion):

$$ng = \frac{\sum_{r}\sum_{s}\left[\left[\min_{k}\left\{c_{k}(\bar{x}^{k+1})\right\}\right]^{T} - \left[\min_{k}\left\{c_{k}(\bar{x}^{k+1})\right\}\right]\right] - c_{rs}(\bar{x}^{k+1}) \cdot \bar{x}^{k+1}_{rs}}{\sum_{r}\sum_{s} c_{rs}(\bar{x}^{k+1}) \cdot \bar{x}^{k+1}_{rs}}$$

If ng < s, then stop, else go to step 1.

Figure A4.1: The DTA algorithm with the Bliemer algorithm included

A5. Implementation of the Bliemer Moving algorithm

5.1 Description of the Bliemer Moving algorithm

The Bliemer Moving method computes the new design point (x^{k+1}) by adding a part of the previous design point (x^k) with a part of the observation evaluated in the moving average of the previous design points ($T(\hat{x}^k)$). So the function evaluations are evaluated in the moving average of the design points ($\hat{x}^k$) and the fixed-point is estimated by $\hat{x}^k$.

5.2 Formulation and solution algorithm of the Bliemer Moving algorithm

To formulate the Bliemer Moving algorithm the following variables and function are used:

- α the step size ($0<\alpha<1$)
- AON the 'All Or Nothing' assignment
- c the route costs
- d the demand
- β the exponent ($0.5<\beta\leq 1$)
- k the iteration number
- M the number of design points taken for the moving average
- ng the normalised duality gap
- s the stop criterion
- t the number of time periods
- T the observation
- x the route flows (link traversal time trajectories)
- $\hat{x}$ the moving average over M of the previous route flows

The mathematical formulation of the Bliemer Moving method is:

$$x^{k+1} = x^k + \alpha^k \cdot \left[T(\hat{x}^k) - x^k\right], \text{ with } x^0 \in \Re^n \text{ and k=1,2,3,...,} \quad \text{(A5.1)}$$

$$\hat{x}^{k+1} = \frac{1}{M} \cdot \sum_{i=k-M+1}^{k} x^i, \quad \text{with } x^0 \in \Re^n, \text{ k=1,2,3,... and M=1,2,3,...,} \quad \text{(A5.2)}$$

$$\alpha^k = p \cdot k^{-\beta}, \text{ with } 0.5<\beta\leq 1 \text{ and } p>0, \quad \text{(A5.3)}$$

The DTA algorithm with the Bliemer Moving algorithm is described as follows (see figure A5.1):

Step 0 (Initialisation):

Initialisation of c, β, k, M, x and $\bar{x}$.

Step 1 (Main loop):

$k = k + 1$

$T = AON(c,d,t)$

$\alpha = k^{-\beta}$

$x^{k+1} = x^k + \alpha^k \cdot \left[T(\hat{x}^k) - x^k\right]$

If $k < M$, then $\hat{x}^{k+1} = \hat{x}^k \cdot (k-1)/k + x^{k+1}/k$, else $\hat{x}^{k+1} = \frac{1}{M} \cdot \sum_{k=k-M+1} x^k$

$c(\hat{x}^k)$

Step 2 (Stop criterion):

$$ng = \frac{\sum\sum\left|\left[\min\left\{c_a(\hat{x}_a^{k+1}), c_a(\hat{x}_a^{k+1})\right\} - c_a(\hat{x}_a^{k+1})\right| \cdot \hat{x}_a^{k+1}}{\sum\sum c_a(\hat{x}_a^{k+1}) \cdot \hat{x}_a^{k+1}}$$

If ng < s, then stop, else go to step 1.

Figure A5.1: The DTA algorithm with the Bliemer Moving algorithm included

A6. Implementation of combinations of algorithms

6.1 The MSA-Bliemer algorithm

If k < ks, then the MSA algorithm (appendix 1) is used in the DTA algorithm, else the Bliemer algorithm (appendix 4) is used.

To formulate the MSA-Bliemer algorithm the following variable is used:

- ks the iteration step after which the MSA algorithm is replaced by the Bliemer algorithm

6.2 The MSA-Bather algorithm

If k < ks, then the MSA algorithm (appendix 1) is used in the DTA algorithm, else the Bather algorithm (appendix 3)

To formulate the MSA-Bather algorithm the following variable is used:

- ks the iteration step after which the MSA algorithm is replaced by the Bather algorithm

6.3 The Bliemer-Bather algorithm

If k < ks, then the Bliemer algorithm (appendix 4) is used in the DTA algorithm, else the Bather algorithm (appendix 3)

To formulate the Bliemer-Bather algorithm the following variable is used:

- ks the iteration step after which the Bliemer algorithm is replaced by the Bather algorithm