Australasian Transport Research Forum 2013 Proceedings
2 - 4 October 2013, Brisbane, Australia
Publication website: http://www.patrec.org/atrf.aspx

A Computational Study of Traffic Assignment Algorithms

Olga Perederieieva[1], Matthias Ehrgott[2], Judith Y. T. Wang[3], Andrea Raith[1]

[1]The University of Auckland, Auckland, New Zealand

[2]Lancaster University, Lancaster, UK

[3]The University of Leeds, Leeds, UK

Email for correspondence: o.perederieieva@auckland.ac.nz

Abstract

Traffic congestion is an issue in most cities worldwide. One way to model and analyse the effect of congestion and other factors on route choice behaviour and to predict the impact of traffic management projects and transport policies is traffic assignment (TA). The most commonly used TA model is known as user equilibrium (UE), which is based on the assumption that all drivers want to minimise their travel time or generalised cost. As a result, an equilibrium is achieved when no one has an incentive to switch to another route.

Although the conventional mathematical model of TA belongs to the convex optimisation domain and, hence, is relatively easy to solve, efficient algorithms are required in order to be able to solve TA in a reasonable amount of time for realistic transport networks. This motivates researchers to propose numerous methods and algorithms to solve this problem in the literature. However, there is no comprehensive empirical study that compares the performance of different approaches on benchmark instances. In this study, our objective is to fill this gap.

We provide a literature review of the most promising methods. We classify algorithms according to the way the solution is represented, namely, link-based (solution is represented by link flows), path-based (solution is represented by path flows) and origin-based (solution is represented by link flows corresponding to each origin), and implement the most representative algorithms in each group. We perform numerical tests on benchmark instances of various sizes, compare the algorithms and analyse the impact of their main components on their running time. We also study the convergence behaviour of the methods with respect to different levels of solution accuracy.

1. Introduction and Motivation

Due to the fast development of cities and road networks the role of transportation planning grows every day since it provides tools for developing effective transportation spending and policies. As presented in Ortúzar and Willumsen (2001), transportation planning allows to: analyse the current usage of a road network; predict the impact of potential projects and policies; control traffic (level of congestion, emissions, toll revenue etc). In order to achieve these goals, a model that can realistically describe travel decisions made by drivers is required. However, in order to create such a model one should know how many people are travelling, from where to where they are travelling, which travel mode they choose (car, bus, bicycle etc) and which routes they prefer. Since it is very difficult to predict how a particular individual will travel, the conventional approach to tackle this difficulty is to make some assumptions on how people choose routes and to find a flow pattern satisfying these assumptions. The most well-known such assumptions are the ones following Wardrop's first principle that is also called *user equilibrium condition*, (Wardrop, 1952): *The journey times on all the routes actually used are equal, and less than those which would be experienced by a single vehicle on any unused route.*

This principle models the behaviour of travellers by assuming that all drivers are selfish and they tend to choose the fastest routes going from their origin to their destination. As a result an *equilibrium state* is achieved, when no one has an incentive to switch to another route.

This principle allows different mathematical formulations of the problem based on different additional assumptions. The classical model that is commonly used in practice is a static deterministic traffic assignment (TA) and is the subject of this paper.

This classical model was developed in the 1950s and since then various algorithms were proposed to solve it. The wide research interest in this problem has several reasons. First, TA is a challenging problem that arises in different practical applications. Second, due to the fast development of hardware and software, the existing transportation models continue to grow. For example, in 2006 the ART model (Auckland Regional Transport Model) included 202 zones, whereas in 2008 it contained 512 zones, (Davies et al., 2009). Third, high accuracy of the solution is required for the select link analysis[1] and for consistent comparison between design scenarios, as presented in Slavin et al. (2010) and Gentile (2009). Therefore, we can conclude that there is a growing need for efficient algorithms able to solve TA problems of realistic size with high accuracy.

Regardless the number of proposed algorithms in the literature, there is no comprehensive survey study comparing them. According to our knowledge, only the paper of Inoue and Maruyama (2012) analyses many different methods under the same computational environment. The authors implemented 11 algorithms for traffic assignment. However, all implementation details are omitted in the paper. It is not clear if the authors used the same framework and how carefully they followed the descriptions of the algorithms available in the literature.

The aim of our research is to analyse and compare the most promising approaches for solving TA in the context of a *framework* that can be shared by different algorithms. We want to compare them without considering any special implementation details that may have been used in commercial or research software. Instead we focus on the use of common code wherever possible. This will allow testing general ideas of the methods without giving an advantage to any of them. We do not claim the best possible implementations, but rather concentrate on the code that can be shared by as many algorithms as possible and compare the programming blocks that are different for each particular method.

Another motivation of this study is to identify the advantages and disadvantages of different *groups* of algorithms (the classification is presented in Section 3). Early on only link-based approaches were available in software because of memory limitations. In recent years, however, other types of algorithms have been implemented in many commercial software packages used by practitioners. One of the main advantages of new algorithms is that they can provide the path choice information which is necessary to evaluate effects of schemes such as congestion pricing without re-running the algorithm, (Florian et al., 2009). Therefore, it becomes important to analyse them and to compare them with classical link-based approaches.

The rest of the paper is organised as follows. Section 2 states the deterministic static traffic assignment problem. Section 3 is devoted to the literature overview and the description of our choice of the algorithms. In Section 4 various methods for solving traffic assignment are described. Section 5 discusses the computational study and comparison of the algorithms. Finally, Section 6 presents conclusion and future work.

2. Problem Formulation

In order to formulate the TA problem the following input data is provided:

[1] Select link analysis provides routeing analysis (i.e. a picture of where traffic is coming from and going to) for assigned vehicles at selected links (and combination of links) throughout the modelled network, see http://www.transportscotland.gov.uk/analysis/LATIS/data/Model-data/Operational-Analysis.

- A transportation network is defined as a directed graph $G(A,N)$ where N is a set of nodes and A is a set of links;
- D_p is the demand of origin-destination (O-D) pair $p \in Z$ (*demand* represents how many vehicles are travelling from an origin to a destination), Z is the set of all O-D pairs;
- $c_a(\vec{f})$ is a function that describes travel time on link a, it depends on link flows $\vec{f} = (f_1, f_2, \ldots, f_{|A|})$, i.e. the number of vehicles per time unit on each link.

Let $\vec{\mathrm{F}} = (\mathrm{F}_1, \mathrm{F}_2, \ldots, \mathrm{F}_{|\mathrm{K}|})$ denote a vector of path flows, where K is the set of all simple paths of graph $\mathrm{G}(\mathrm{A},\mathrm{N})$. They are related to link flows by the expression $\mathrm{f_a} = \sum_{\mathrm{p}\in\mathrm{Z}} \sum_{\mathrm{k}\in\mathrm{K_p}} \delta_{\mathrm{a}}^{\mathrm{k}} \mathrm{F_k}$, where $\delta_{\mathrm{a}}^{\mathrm{k}}\mathrm{F_k}$ equals one if link a belongs to path k, and zero otherwise; $\mathrm{K_p} \subseteq \mathrm{K}$ is the set of paths between O-D pair p. Let $\mathrm{C_k}(\vec{\mathrm{F}})$ denote the travel time on path k, it is also called path cost function.

The conventional model of the traffic assignment problem is based on two important assumptions that allow to formulate and solve it as a mathematical program. These assumptions are stated as follows:

- *Additivity* of path cost functions: travel time of each path is the sum of travel times of links belonging to this path, i.e. $C_k(\vec{F}) = \sum_{a\in A} \delta_a^k \, c_a(\vec{f})$;
- *Separability* of link cost functions: travel time of each link depends only on flow on this link, i.e. $c_a(\vec{f}) = c_a(f_a)$.

If these assumptions are satisfied, solving the following optimisation problem results in the link flows satisfying the user equilibrium condition, (Sheffi, 1985):

$$\begin{aligned} &\min \sum_{\mathrm{a}\in\mathrm{A}} \int_0^{\mathrm{f_a}} \mathrm{c_a}(\mathrm{x})\mathrm{dx} \\ &\sum_{\mathrm{k}\in\mathrm{K_p}} \mathrm{F_k} = \mathrm{D_p}, && \forall \mathrm{p} \in \mathrm{Z}, \\ &\mathrm{F_k} \geqslant 0, && \forall \mathrm{k} \in \mathrm{K_p}, \forall \mathrm{p} \in \mathrm{Z}, \\ &\mathrm{f_a} = \sum_{\mathrm{p}\in\mathrm{Z}} \sum_{\mathrm{k}\in\mathrm{K_p}} \delta_{\mathrm{a}}^{\mathrm{k}} \mathrm{F_k}, && \forall \mathrm{a} \in \mathrm{A}. \end{aligned} \tag{2.1}$$

In order to ensure the existence of a solution of TA all path cost functions $\mathrm{C_k}(\vec{\mathrm{F}})$ must be *positive and continuous*, and to ensure the uniqueness of the solution of TA in terms of link flows $\vec{\mathrm{f}}$ all path cost functions $\mathrm{C_k}(\vec{\mathrm{F}})$ must be *strictly monotone* (Florian and Hearn, 1995). In the following it is assumed that these requirements are satisfied.

3. Literature Overview

One of the possible ways to classify traffic assignment algorithms is according to how the solution is represented: link-based (solution variables are link flows), path-based (solution variables are path flows) and origin-based (solution variables are link flows coming from a particular origin), see (Zhou and Martimo, 2010).

Historically the first algorithms developed for solving the traffic assignment problem were link-based. The most well-known such algorithm is Frank-Wolfe (FW), (Frank and Wolfe, 1956). Due to its simplicity and low memory requirements it is used even now and is implemented in different commercial software packages. However, this algorithm and other link-based methods are known to tail badly in the vicinity of the optimum and usually cannot be used to achieve highly precise solutions as is demonstrated by numerous numerical studies, see Lee et al. (2002), Florian et al. (2009) and Inoue and Maruyama (2012).

Many improvements of FW were proposed in the literature. As explained in Zhou and Martimo (2010), some of them try to improve the FW search direction (for example, see Fukushima (1984), Florian et al. (1987), Mitradjieva and Lindberg (2012)) or step size (see Powell and Sheffi (1982), Weintraub et al. (1985)). Zhou and Martimo (2010) also categorise restricted simplicial decomposition (RSD), (Hearn et al., 1985) and nonlinear simplicial decomposition (NSD), (Larsson et al., 1997) as link-based methods. These algorithms apply a more complicated structure than FW, however they use FW as a special case. In both of them the link flow solution is represented by a linear combination of extreme points. The algorithms of this type usually have two main routines: generating new extreme points and optimising the so-called restricted master problem with respect to previously generated extreme points.

The first path-based algorithm was proposed in Dafermos and Sparrow (1969). It is called path equilibration (PE). Its main idea is to shift flow from the path with maximum cost to the path with minimum cost. However, in the years after the PE algorithm was published, path-based methods were considered impractical because of memory limitations. As a result, the development in this field started again only in recent years. Larsson and Patriksson (1991) proposed disaggregated simplicial decomposition (DSD) which is similar to RSD in the way that it also uses a linear combination of extreme points to represent the solution. However, the extreme points are in the space of path flows instead of link flows. In Jayakrishnan et al. (1994) the gradient projection (GP) method was proposed and further studied in Chen and Jayakrishnan (1998). It is similar to PE, but O-D flow is moved from several non-shortest paths to the shortest path. Another path-based algorithm was proposed in Florian et al. (2009). It is called projected gradient (PG) and is based on the idea of moving flow from the set of paths with cost greater than the average path cost to the set of paths with cost less than the average path cost. Another similar approach called improved social pressure (ISP) algorithm was developed in Kumar and Peeta (2011). This method also shifts flow from the set of costlier paths to the set of cheaper paths, but a more complicated strategy of flow distribution is applied.

The origin-based algorithms represent a more recent development in this field. Their main idea is to decompose the problem into a sequence of subproblems that operate on acyclic subnetworks of the original transportation network (Nie, 2010). In general, for this group of algorithms, the flow shifts are restricted to subproblems and are usually similar to the ideas presented earlier for path- and link-based approaches. The first algorithm of this type applied to the traffic assignment problem was proposed by Bar-Gera (2002). It is called origin-based algorithm (OBA). Nie (2011) presented some corrections to OBA, and Nie (2010) compared different origin-based algorithms (corrected OBA (COBA) and modified OBA (MOBA)) based on OBA. Other developments in this field include the following methods: algorithm B (B) proposed by Dial (2006), some modifications of it (iB), see Inoue and Maruyama (2012) and Tianran et al. (2010), linear user cost equilibrium (LUCE) developed by Gentile (2009) and traffic assignment by pairs of alternative segments (TAPAS) implemented by Bar-Gera (2010).

In order to select algorithms for implementation we analyse different empirical studies from the literature. During such analysis it is important to pay attention to how the algorithms were compared (re-implemented by the authors of the study or using existing software), what instances were used and how precise the obtained solutions were.

In the majority of the studies relative gap is used as a convergence measure. It is calculated as follows:

$$\mathrm{RGAP} = 1 - \frac{\sum_{p \in Z} D_p \, C_{min}^p}{\sum_{a \in A} f_a \, c_a} \tag{3.1}$$

where $C_{min}^{p} = \min_{k \in K_p} C_k$ is the shortest path of O-D pair p. We divide the existing numerical studies into two main groups (low and high precision) corresponding to the accuracy of the solution. The algorithms from the low precision group are stopped when the relative gap is in the interval $[10^{-4}, 10^{-7}]$, and for the high precision group the interval is $[10^{-10}, 10^{-14}]$. Table 1 presents the first group and Table 2 the second one. In each table one algorithm from each study is highlighted in bold, which means that it showed the best performance on the majority of the tested instances. All other algorithms in the same study are written in the order of decreasing performance. If in a particular study the existing executable of the algorithm or commercial software was used, it is highlighted in grey. If comparison of the algorithms was made indirectly, i.e. based on the running times reported in other papers, it is highlighted in light grey. We also try to align the algorithms that were compared in different studies. However, since sometimes contradictory conclusions are made in different papers, this alignment is very approximate. The last line of each table summarises the algorithms that seem to be the most promising, namely BFW (bi-conjugate Frank-Wolfe (Mitradjieva and Lindberg, 2012)), GP, PG, ISP, B, LUCE, TAPAS.
Therefore, we choose to implement the following algorithms:

- *Link-based*: FW (FW is reported to have the worst performance. However, it is the basis for CFW (conjugate Frank-Wolfe (Mitradjieva and Lindberg, 2012)) and BFW, that is why it is also considered in our study), CFW, BFW;
- *Path-based*: PE (this algorithm was not compared in the studies presented above. However, we choose to implement it as well due to its simplicity and ideas similar to other path-based approaches), GP, PG and ISP;
- *Origin-based*: B.

The most recent algorithms that also show very promising performance, namely LUCE and TAPAS, are subject of our future research.

Table 1: Summary of existing empirical studies. Low precision: $RGAP \in [10^{-4}, 10^{-7}]$

Mitradjieva and Lindberg (2012)	Zhou and Martimo (2010)	Slavin et al. (2006)	Dial (2006)	Florian et al (2009)	Gentile (2009)	Tianran et al. (2010)	Kumar and Peeta (2011)
					LUCE		**ISP**
				PG	PG		SP (older version of ISP)
		B	**B**		B	**iB**, B	
		OBA	OBA	OBA	OBA		
		FW	FW	FW	FW	FW	
	GP	GP				OBA	
BFW	BFW						
CFW	CFW						
FW	FW						
OBA	OBA						
DSD							
Best: BFW, GP, PG, ISP, B, LUCE							

Table 2: Summary of existing empirical studies. High precision: $RGAP \in [10^{-10}, 10^{-14}]$

Bar-Gera (2002)	Nie (2010)	Bar-Gera (2010)	Inoue and Maruyama (2012)
		TAPAS	**TAPAS**
	B		B, iB
	MOBA		
	COBA		
OBA	OBA	OBA	OBA, DSD, MOBA, LUCE, ASD
FW		FW	FW
Best: B, TAPAS			

4. Algorithms

One of the reasons for the development of various traffic assignment algorithms is a specific problem structure that can be exploited by solution methods in many different ways (Patriksson, 1994).

The constraints of the TA problem represent a polyhedral set and the objective function is convex. As a result feasible direction methods can be applied as a solution technique (Andréasson et al., 2007). The algorithms described later in this section belong to this type of nonlinear optimisation methods. The main idea behind this type of methods is as follows: starting from a feasible solution, a feasible direction of descent is calculated and the solution is moved along this direction in such a way that the strict decrease of the objective function is guaranteed.

Another property of traffic assignment that can be exploited is the O-D pair separability: the flow conservation constraints of one O-D pair do not affect those of any other pair. This allows applying decomposition algorithms, that are based on the idea of decomposing the problem into smaller sub-problems that can be solved one after another or in parallel, see Patriksson (1994). This section discusses different decomposition approaches and algorithms. In particular we focus on the algorithms that were chosen for our numerical study, see Section 3.

4.1. Link-based Algorithms

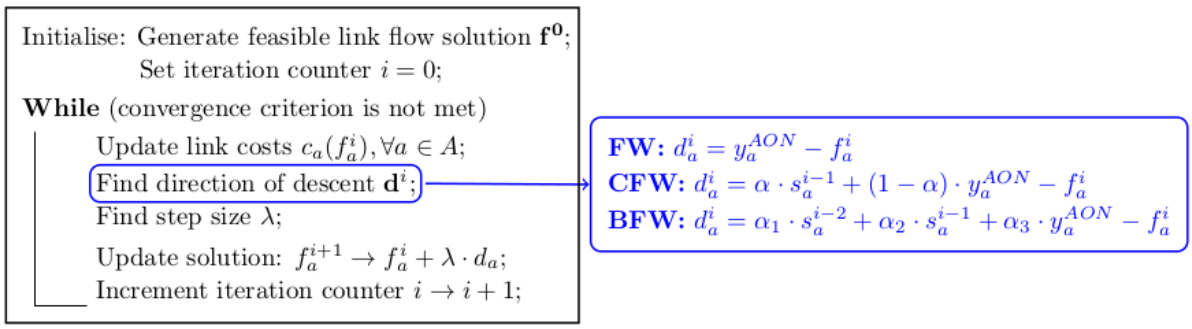
This Section presents Frank-Wolfe and two of its modifications: conjugate and bi-conjugate Frank-Wolfe methods. These algorithms can be described using the framework presented in Figure 1. As can be seen from the framework the methods differ only in the way the direction of descent is defined.

Each algorithm starts with an initial feasible link flow pattern. It is usually generated using *all-or-nothing* (AON) assignment: each link flow is initialised with zero and corresponding link travel times are calculated, then shortest paths for each O-D pair are found and corresponding demand is assigned to each shortest path. All these path flows are then projected on all links resulting in the initial feasible link flows.

FW further exploits the AON procedure. In order to generate a feasible direction of descent it performs AON assignment at each iteration and gets corresponding link flows $y_a^{AON}, \forall a \in A$. Since the AON link flow solution is feasible, given a current link flow solution $f_a^i, \forall a \in A$, vector

$d_a^i = y_a^{AON} - f_a^i, \forall a \in A$ is a feasible direction. Moreover, this direction is also a direction of descent (Sheffi, 1985).

Figure 1: Framework for link-based algorithms



CFW and BFW algorithms use information about previously generated directions of descent in order to find a new one. In particular, the new search direction is constructed via mutually conjugate directions with respect to the Hessian of the objective: CFW takes into account the usual FW direction and the direction from the previous iteration, whereas BFW in addition considers one more direction from iteration $i-2$ where i is the current iteration. Both algorithms operate on auxiliary variables $s_a, \forall a \in A$ called points of sight that store information about previously generated directions of descent. For details see Mitradjieva and Lindberg (2012).

After finding a direction of descent each algorithm proceeds to step size calculation which represents how far away the current solution must be moved in the direction of descent. The step size can be found by solving exactly or approximately the following optimisation problem (Florian and Hearn, 1995):

$$\min_{0 \leqslant \lambda \leqslant 1} \sum_{a \in A} \int_0^{f_a^i + \lambda d_a^i} c_a(\omega) d\omega \qquad (4.1)$$

There are various methods available for solving the one-dimensional optimisation problem (4.1), see for example Andréasson et al. (2007). In this study we applied quadratic interpolation, i.e. the objective function along the direction of descent is approximated by a quadratic function and minimised using the analytic solution:

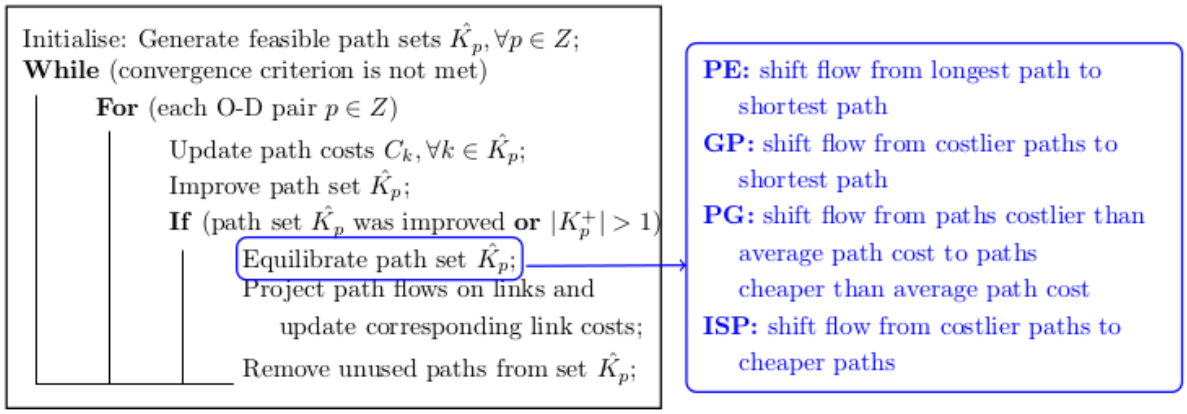
$$\lambda = -\frac{\sum_{a \in A} c_a(f_a) d_a}{\sum_{a \in A} \frac{\partial c_a(f_a)}{\partial f_a} d_a^2}. \qquad (4.2)$$

3.4. Path-based Algorithms

Path-based methods exploit the O-D pair separability of the TA problem. At each iteration, the flows are moved only within one O-D pair and path flows of the other O-D pairs are fixed. In order to achieve this, paths must be stored. Let $\widehat{K_p}, \forall p \in Z$ denote the corresponding sets of paths. A general framework of this group of algorithms is presented in Figure 2.

In order to prevent storing all possible simple paths for each O-D pair a *column generation* approach is usually applied, which consists in generating new paths when needed (Patriksson, 1994). In particular, it is performed as follows: for a given O-D pair p find the shortest path and add it to $\widehat{K_p}$ if it is shorter than the current shortest path contained in this set. This step corresponds to "*Improve path set* $\widehat{K_p}$" of the framework. Also, in order to keep only promising paths in $\widehat{K_p}$, the paths that do not carry flow anymore are removed.

Figure 2: Framework for path-based algorithms



As in the case of link-based methods, initialisation is performed by AON assignment. In addition to a link flow solution each set $\widehat{K_p}, \forall p \in Z$ of paths must also be initialised. This can be done by adding the shortest path corresponding to O-D pair p to $\widehat{K_p}$.

All algorithms of this group differ in how path set $\widehat{K_p}$ is equilibrated. Equilibration of the set aims at equalising some or all of the path costs of set $\widehat{K_p}$. The remainder of this section discusses different algorithms belonging to this group.

The PE algorithm equalises the costs of the current longest path $l \in \widehat{K_p}$ with positive flow and the cost of the current shortest path $s \in \widehat{K_p}$. This is equivalent to solving a non-linear equation. We apply Newton's method in order to solve it, namely we calculate the Newton step $\Delta F = \frac{C_l - C_s}{\sum_{a \in A_{s,l}} \frac{\partial c_a}{\partial f_a}}$, where $A_{s,l}$ is the set of links that belong to path l and path s without links common to both these paths, and update path flows in the following way:

$$\begin{aligned} F_l &= F_l - \min\{F_l, \Delta F\}, \\ F_s &= F_s + \min\{F_l, \Delta F\}. \end{aligned} \tag{4.3}$$

Expression $\min\{F_l, \Delta F\}$ is necessary, because the calculated flow shift ΔF might be infeasible causing F_l to become negative.

The GP algorithm considers several paths at each iteration. In particular, it moves flow to the current shortest path $s \in \widehat{K_p}$ from all other paths of set $\widehat{K_p}$. First, an appropriate flow shift is calculated:

$$\Delta F_k = \frac{C_k - C_s}{\sum_{a \in A_{s,k}} \frac{\partial c_a}{\partial f_a}}, \qquad \forall k \in \widehat{K_p}, k \neq s. \tag{4.4}$$

Second, a new solution is projected onto the feasible set as follows:

$$\begin{aligned} F_k &= F_k - \min\{F_k, \alpha \Delta F_k\}, \qquad \forall k \in \widehat{K_p}, k \neq s, \\ F_s &= D_s - \sum_{k \in \widehat{K_p}, k \neq s} F_k, \end{aligned} \tag{4.5}$$

where α is a predefined constant that must be small enough in order to guarantee convergence of the algorithm (Jayakrishnan et al., 1994). In our study it was set to 0.25 because this value allows all tested instances to converge (Jayakrishnan et al. (1994) recommend to set it to 1).

The main idea of the PG algorithm is to move flow from the paths that have cost greater than the current average path cost to the paths that have cost less than the average value. It is equivalent to defining the following direction of descent $\vec{d}$:

$$d_k = \bar{C}_p - C_k, \forall k \in \widehat{K_p} \tag{4.6}$$

where $\bar{C}_p = \frac{\sum_{k \in \widehat{K_p}} C_k}{|\widehat{K_p}|}$ is the average cost of the paths of O-D pair p. In order to find the appropriate amount of flow to move line search is applied along direction $\vec{d}$. As a result, step size λ is calculated. We used quadratic interpolation for this purpose, see Section 4.1. The path flows are updated accordingly: $F_k = F_k + \lambda d_k, \forall k \in \widehat{K_p}$.

The ISP algorithm is based on the idea of "*social pressure*" which is defined as the difference between a path cost and the cost of the shortest path. All the paths are divided into two groups $\underline{P_p} \subset \widehat{K_p}$ (paths with the cost less than or equal to some value π) and $\overline{P_p} \subset \widehat{K_p}$ (paths with the cost greater than π) such that $\underline{P_p} \cup \overline{P_p} = \widehat{K_p}$. π is determined at each iteration and is defined as follows: $\pi = C_s + \delta(C_l - C_s)$, where C_s and C_l are the costs of current shortest and longest paths, δ is a predefined constant that was set to 0.15 as suggested in the paper of Kumar and Peeta (2011). Flow is shifted from the paths belonging to set $\overline{P_p}$ to the paths belonging to set $\underline{P_p}$. This is also equivalent to defining a direction of descent $\vec{d}$ and moving the solution along this direction. Direction $\vec{d}$ is as follows:

$$\begin{aligned} d_k &= C_s - C_k, && \forall k \in \overline{P_p}, \\ d_l &= \frac{\sum_{k \in \overline{P_p}} \Delta F_k}{s_l(F_l) \sum_{m \in \underline{P_p}} \frac{1}{s_m(F_m)}}, && \forall l \in \underline{P_p}, \end{aligned} \tag{4.7}$$

where $s_m(F_m)$ is the first derivative of the cost function of the path m with respect to path flow: $s_m(F_m) = \sum_{a \in m} \frac{\partial c_a(f_a)}{\partial f_a}$. The step size λ is calculated as presented in Section 4.1 and path flows are updated as follows: $F_k = F_k + \lambda d_k, \forall k \in \widehat{K_p}$.

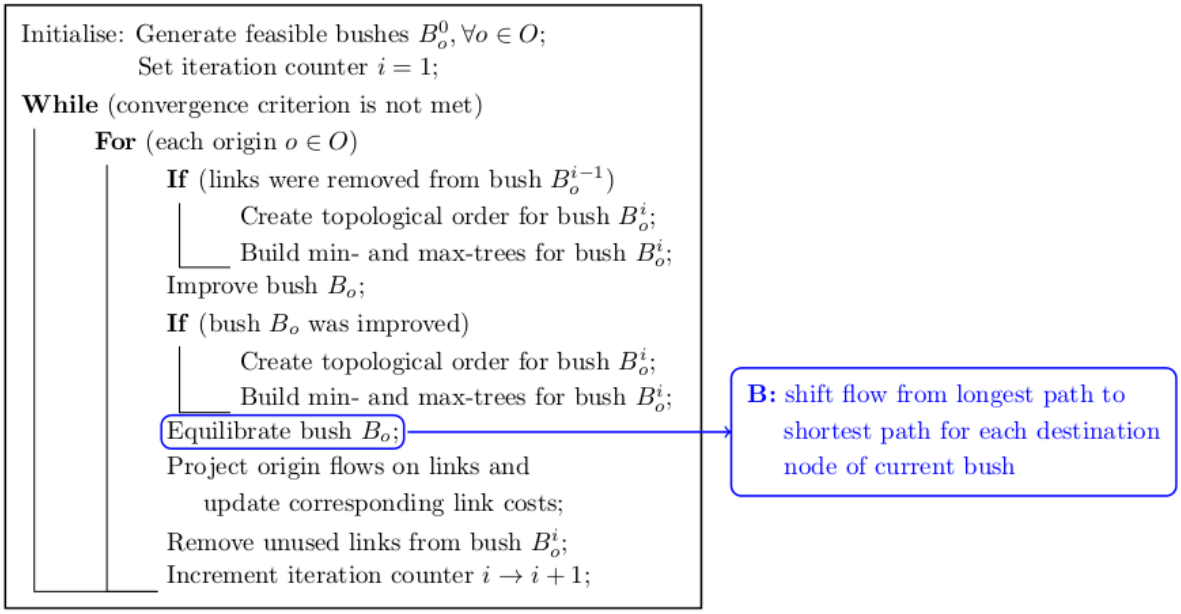
4.3 Origin-based Algorithms

Origin-based algorithms also exploit the O-D pair separability of the TA problem. But instead of decomposing the problem into sub-problems corresponding to each O-D pair, the algorithms of this type decompose the problem into sub-problems corresponding to each origin. As a result, at each iteration flows are moved within one origin using a special data structure called *bush*. A bush is a directed sub-network of the original network $G(A, N)$ rooted at a given origin o such that it is (Nie, 2010):

- *Connected*, i.e. using only links in the bush it is possible to reach every node that was reachable in the original network;
- *Acyclic*, i.e. the bush does not contain directed cycles.

We denote a bush by $B_o(N, A_o) \subset G(A, N)$, where $A_o \subset A$. Let $O \subseteq N$ denote the set of origins. Then, at each iteration only one bush $B_o(N, A_o)$ is considered. The current link-based solution $\vec{f}$ is the sum of $|O|$ link flows of $B_o(N, A_o), \forall o \in O$, i.e. $f_a = \sum_{o \in O} f_a^o, \forall a \in A$, where f_a^o denotes the flow on link a of the bush $B_o(N, A_o)$ (Dial, 2006).

Figure 3: Framework for origin-based algorithms



The motivation to decompose the traffic assignment problem by origins consists in the *acyclicity of user equilibrium*: for the single-origin formulation of the traffic assignment problem, links that have positive flow at user equilibrium never form a directed cycle (Nie, 2010). This property gives several interesting advantages from a practical point of view since different operations such as shortest path calculations can be performed more efficiently on acyclic structures (such as bushes) than on general networks (Nie, 2010).

All algorithms that decompose the TA problem by origins share a general framework presented in Figure 3.

Each bush $B_o(N, A_o)$ is usually initialised with a shortest path tree rooted at origin o and the link flows are initialised by AON assignment.

As in the case of path-based methods, bushes are constructed iteratively by adding promising links and removing unused ones. The removal of links is performed in the following way: links that carry zero flow are dropped if the connectivity of the bush is retained. The addition of new links must be performed with care in such a way that directed cycles are not created. The reader is referred to Nie (2010) for a detailed explanation of this algorithmic step.

In our study we consider only one origin-based algorithm called algorithm B, see Dial (2006). Its main idea is similar to PE (see Section 4.2), namely the flow is shifted from the longest used path to the shortest path, but within a given bush. We implemented the same approach as in the case of the PE algorithm: the Newton step is applied to equalise the path costs. It must be noticed that the explicit storage of paths is not required anymore, because the calculation of the longest path can be done in linear time on the bush due to its acyclicity. In the case of path-based algorithms the paths must be stored because the longest path calculation is NP-complete on general graphs (Patriksson, 1994).

5. Computational Study

5.1 Environment and Problem Instances

This section summarises our empirical study. All algorithms were implemented in the C++ programming language and compiled using g++ 4.7.3 (Ubuntu/Linaro 4.7.3-1ubuntu1). All runs were performed under the following environment:

- *OS*: Ubuntu Release 13.04 64-bit;
- *CPU*: Intel Core i5-2500 CPU, 4 Core, 3.30GHz;
- *RAM*: 7.7 GB.

The tests were performed on the instances available at the web-site: http://www.bgu.ac.il/~bargera/tntp/. The main characteristics of these instances are presented in Table 3.

Table 3: Problem instances

Instance name	Nodes	Links	Zones	O-D pairs
Sioux-Falls	24	76	24	528
Anaheim	416	914	38	1406
Barcelona	1020	2522	110	7922
Winnipeg	1052	2836	147	4344
Chicago Sketch	933	2950	387	93135

We use the relative gap RGAP (see Section 3) as a convergence criterion for two reasons: it is a common measure of convergence, see Slavin et al. (2006), Florian et al. (2009), Dial (2006), Nie (2010), and it can be calculated for all tested algorithms. The required accuracy of the solution was set to $\epsilon = 10^{-14}$, i.e. the algorithms were stopped after the relative gap was less than ϵ.

We have also performed one more numerical test on the Chicago Regional network that is much larger than the instances presented in Table 3. It has 1791 nodes, 39018 links, 1790 zones and 2296227 O-D pairs. For this test we set a time limit of 8 hours for every algorithm. The results are discussed in the following sections.

5.2 Implementation

All algorithms were implemented in terms of the frameworks presented in Section 4. This ensures the usage of common code wherever possible.

Each TA method requires solving the shortest path problem multiple times. Link-based approaches need a single-source shortest path algorithm for general graphs. The same algorithm is also required for the relative gap calculation. For this purpose we implemented the label correcting algorithm presented in Sheffi (1985). Path-based approaches, on the other hand, require a point-to-point shortest path method since at each iteration only one O-D pair is considered. For this case we used the A* algorithm[2], see Goldberg et al. (2006). For origin-based algorithm B we implemented the conventional shortest path algorithm for directed acyclic graphs that uses topological order, see Dasgupta et al. (2006). It should also be noticed that each bush is constructed iteratively. This means that topological sorting can be performed dynamically (Pearce and Kelly, 2006). However, our current implementation does not exploit this fact and the topological order is computed from scratch each time the bush topology changes.

In Florian et al. (2009) it is mentioned that when the solution is close to the optimal one, many O-D pairs are equilibrated and there is no need to change their path flows and improve path sets. Our implementation does not take into account this empirical fact. Consequently, some algorithmic steps are performed even in the case when this is not necessary. For

[2] An implementation of the A* algorithm was kindly provided by Boshen Chen.

example, the improvement of path set $\widehat{K_p}$ which involves calculation of the shortest path is performed at each iteration regardless how close to equilibrium O-D pair p is.

Also our current implementation is not optimised with respect to memory usage. For all algorithms we used extended floating point precision (C++ long double type).

5.3 Results

Our first numerical test was performed on the small and medium size instances presented in Table 3. All link-based methods were not able to reach the required accuracy of relative gap in a reasonable amount of time (execution time of all algorithms was limited to 8 hours). The PG algorithm showed unstable numerical behaviour consisting in the loss of flow at each iteration. Basically, because of rounding errors in the computation of the average path cost, the amount of flow lost at each iteration prevents this algorithm from converging when the required precision is approximately less than 10^{-7} (this value depends on the problem instance).

Figure 4 presents running times of the algorithms that were able to reach the required precision. According to these results, algorithm B is usually significantly faster than other methods. PE shows the best performance among path-based algorithms followed by ISP and GP. GP is usually slower because its performance depends on parameter α. In our numerical tests we didn't adjust this parameter to improve performance of the algorithm since it is instance-dependent. We think that ISP is slower than PE mainly because it requires additional computations at each step, see Section 4.2.

Figures 5 and 7 show convergence of the algorithms on the Barcelona and Chicago Sketch instances, respectively, whereas Figures 6 and 8 show early stages of the same convergence. After analysing these results, we can conclude that link-based methods converge fast on early iterations, but start tailing in the vicinity of the optimum. Usually, if the required level of precision is 10^{-5} of relative gap, link-based methods represent a reasonable alternative to other approaches. Among the algorithms of this type BFW demonstrated the best performance on the majority of the tested instances. If the required precision is stricter, then algorithm B is a good choice in terms of both memory usage and running time compared to path-based approaches.

Different convergence patterns of different groups of algorithms occur in Figures 5 and 7. In particular, all link-based methods tail in the vicinity of optimum. As mentioned in Patriksson (1994), Jayakrishnan et al. (1994) and Mitradjieva and Lindberg (2012), this happens because the search direction is perpendicular to the gradient of the objective function when the algorithm approaches the optimal solution. This does not happen for path- and origin-based algorithms. In Figure 5 the relative gap of algorithm B and PE drops by a factor of 10 after reaching a value of 10^{-8}. This behaviour is not common (for example, see Figure 7). One possible explanation of this phenomenon is the fact that the relative gap is measured after all O-D pairs are considered and not after each flow move. Therefore, for this particular instance, the relative gap might reduce by a factor of 10 after flows are shifted within all O-D pairs.

Our second numerical test was performed on a large instance of the Chicago Regional network. The required precision was set to 10^{-5} of relative gap and time of execution was limited to 8 hours. All path-based algorithms were not able to reach the required precision because of the lack of memory. One obvious reason is absence of memory usage optimisation of the current implementation. The convergence of the algorithms is presented in Figure 9, with a few first iterations of path-based algorithms before they run out of memory (the PG algorithm is not presented because it ran out of memory after the first iteration). As can be seen from the results, the fastest algorithm was BFW. Algorithm B starts dominating FW and CFW when the relative gap is less than 10^{-4}. However, on this level of solution accuracy it is still slower than BFW.

5.4 Summary

Each type of algorithm presented above has its advantages and disadvantages that must be taken into account when solving the TA problem. Depending on the particular requirements on the solution, a problem instance and available resources, preference must be given to a particular algorithm. Table 4 presents some potential trade-offs of different approaches.

6. Conclusion and Future Work

This work starts with a literature overview of existing traffic assignment algorithms, based on which we selected and implemented the most promising ones. The main aim of our numerical study is to compare the algorithms under the same computational environment and in terms of a framework that insures that common code is used wherever possible. We implemented and analysed several algorithms belonging to link-, path- and origin-based methods, namely FW, CFW, BFW, PE, GP, PG, ISP and B.

Based on the performed computational study we can conclude that link-based algorithms represent a good alternative to other methods when the required level of precision is low. Path-based algorithms must be implemented with care in terms of memory usage, especially if the problem instance is large. Origin-based algorithm B is capable to achieve a high level of precision with moderate memory consumption and is preferable when high precision of the solution is needed. Numerical stability of the algorithms is also an important issue especially if high accuracy of the solution is necessary and must be taken into account.

The future development of this study consists in implementing the most recent algorithms such as LUCE and TAPAS and in improving the current code by reducing memory usage of path-based algorithms, adding on-line topological sorting and other TA specific features that can be exploited by different methods. Also we plan to study the performance of algorithms with respect to the representative operation counts in order to identify and analyse the asymptotic bottleneck operations.

Figure 4: Running time of algorithms

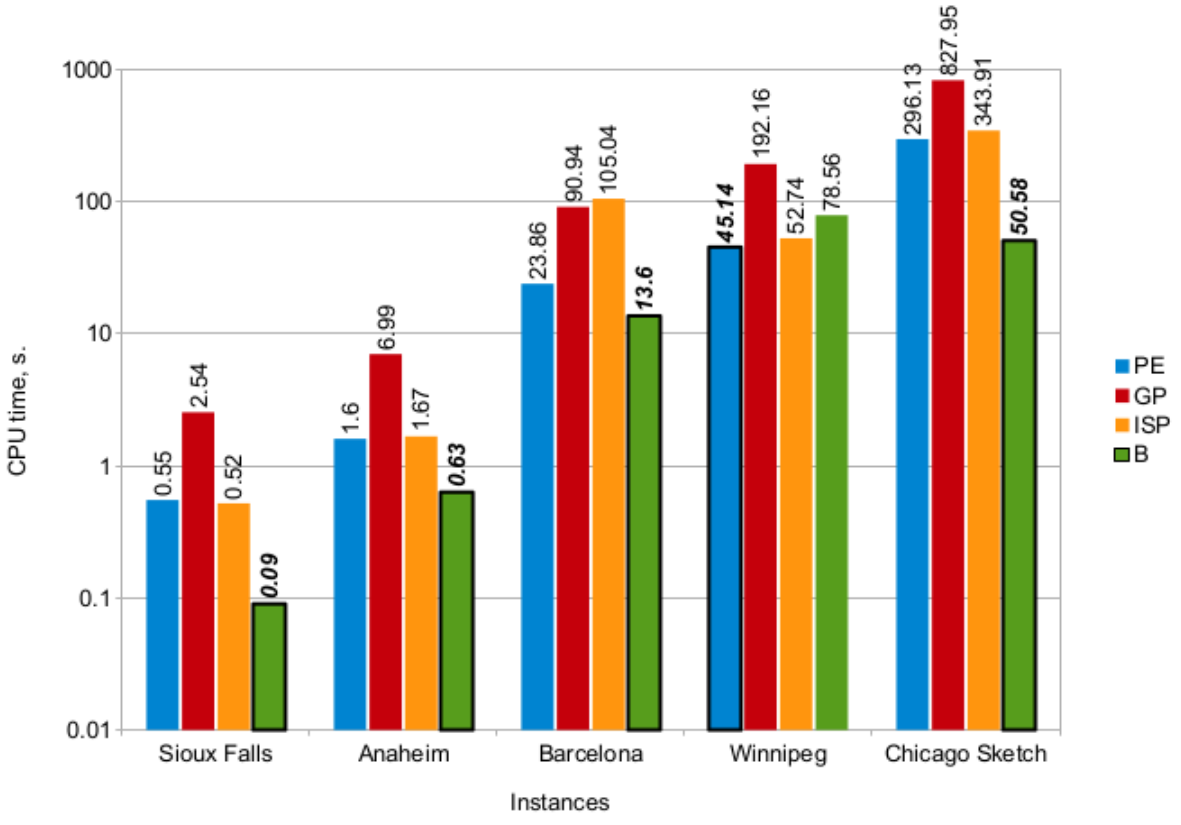


Table 4: Trade-offs of different TA algorithms

Factor	Link-based	Path-base	Origin-based
Memory requirements	Low: $O(\|A\|)$	High: potentially exponential (number of paths in graph is exponential)	Moderate: $O(\|O\|\|A\|)$
Solution precision	Low: usually cannot achieve high precision in reasonable amount of time	High	High
Shortest path algorithm	Single-source shortest path on general graph	Point-to-point shortest path on general graph	Single-source shortest path on directed acyclic graph
Flow moves at each iteration	Flows are moved within entire network	Flows are moved within path set	Flows are moved within bush
Convergence behaviour	Quickly converges in the beginning, tails near vicinity of optimum	Do not tail in the vicinity of optimum	

Figure 5: Convergence of algorithms. Barcelona instance

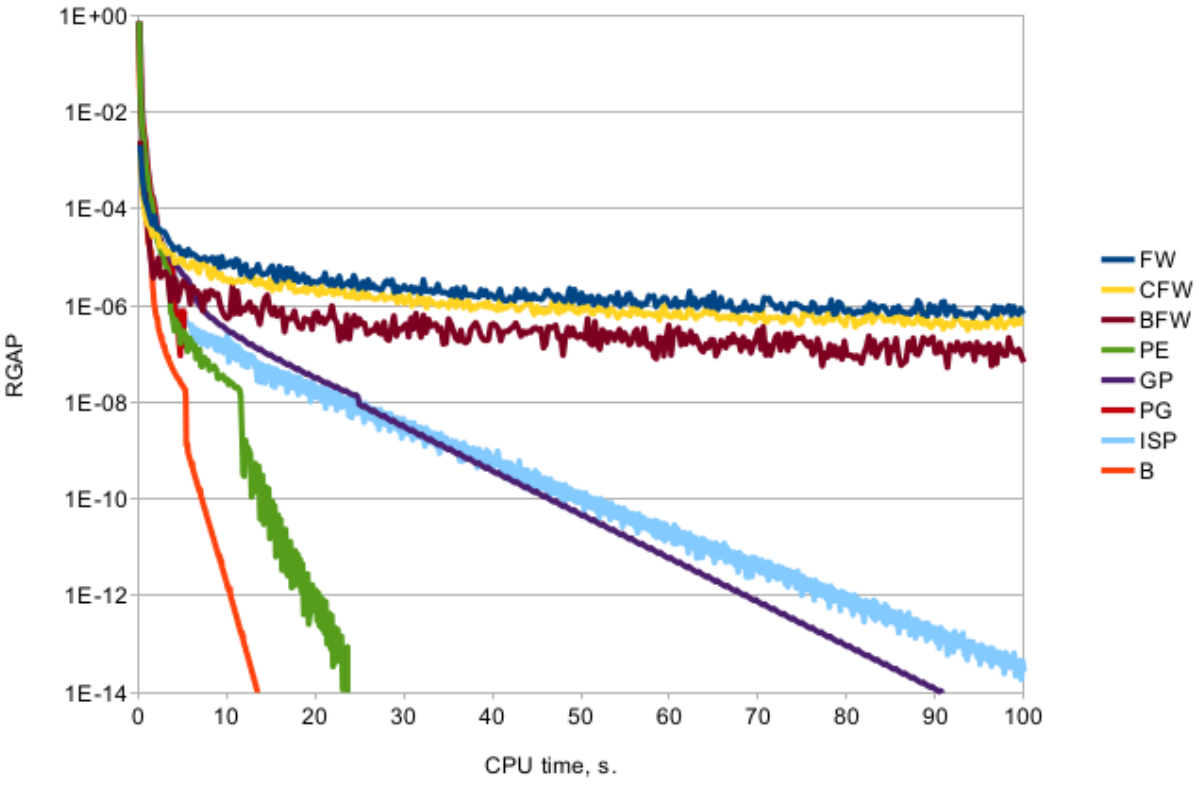


Figure 6: Convergence of algorithms on early stages. Barcelona instance

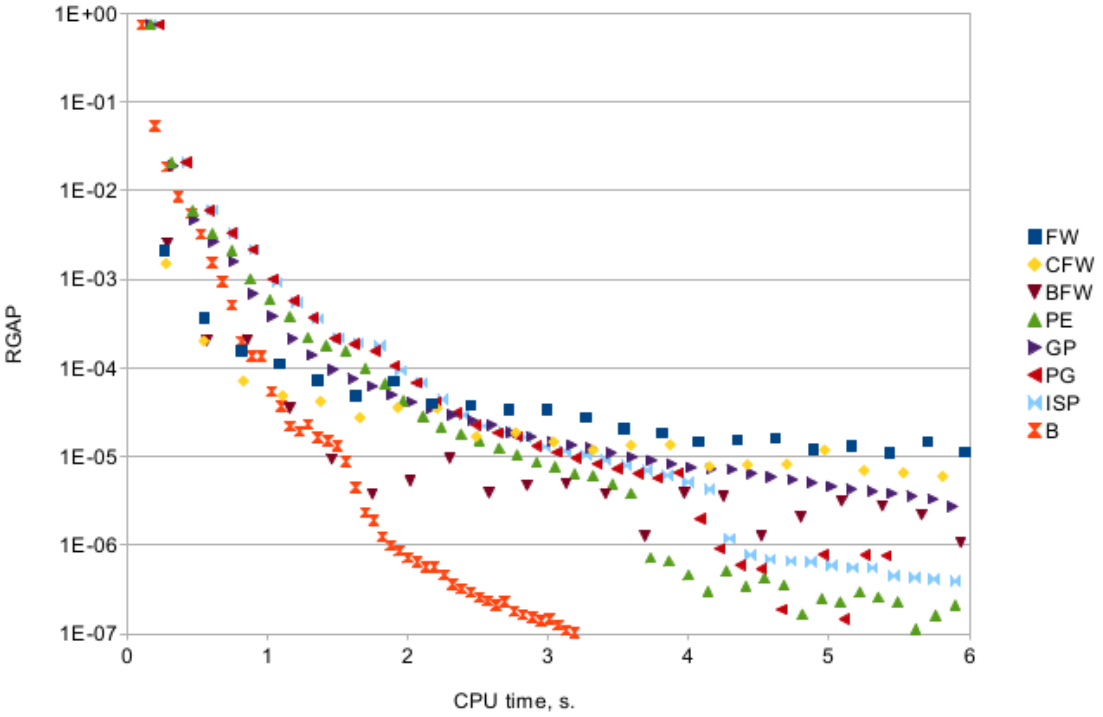


Figure 7: Convergence of algorithms. Chicago Sketch instance

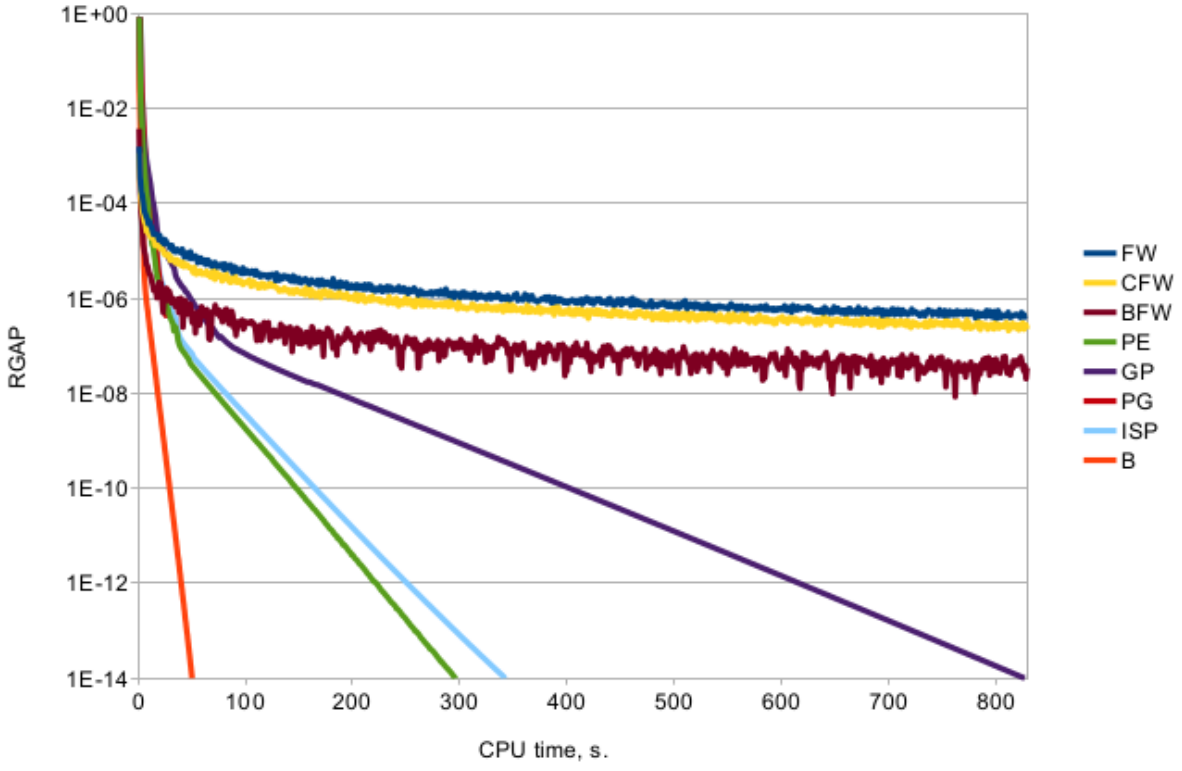


Figure 8: Convergence of algorithms on early stages. Chicago Sketch instance

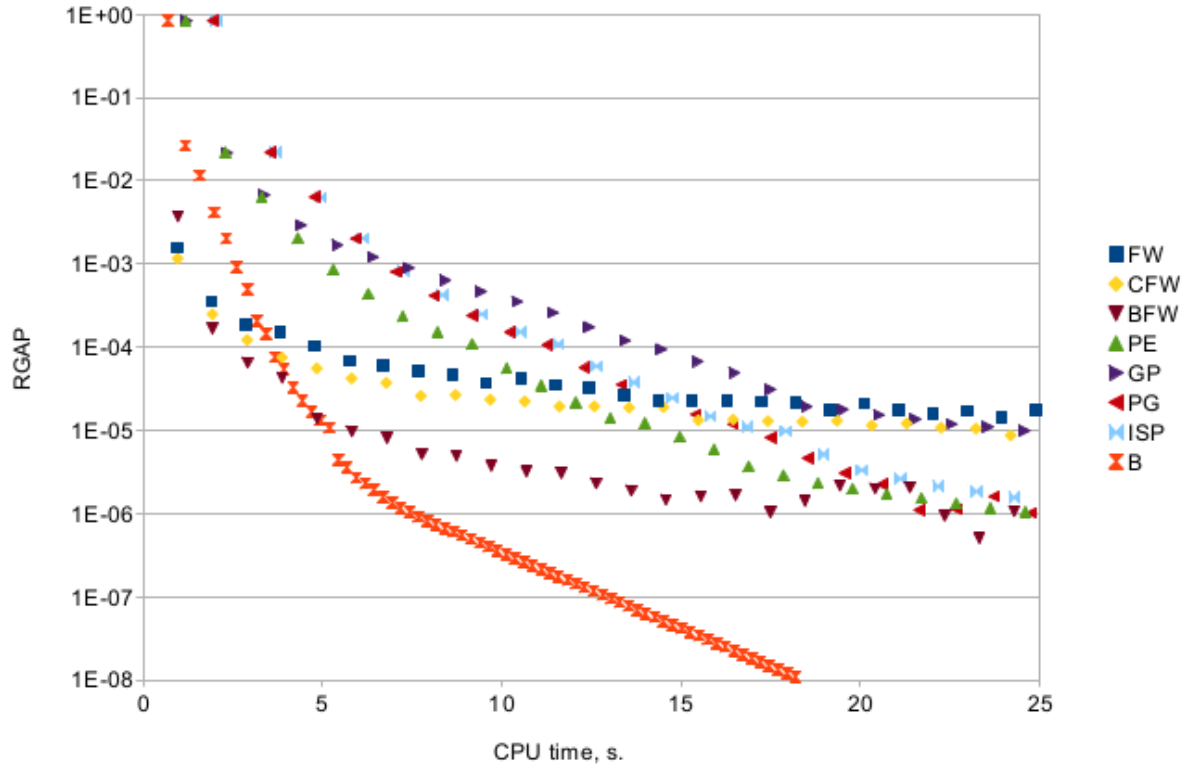
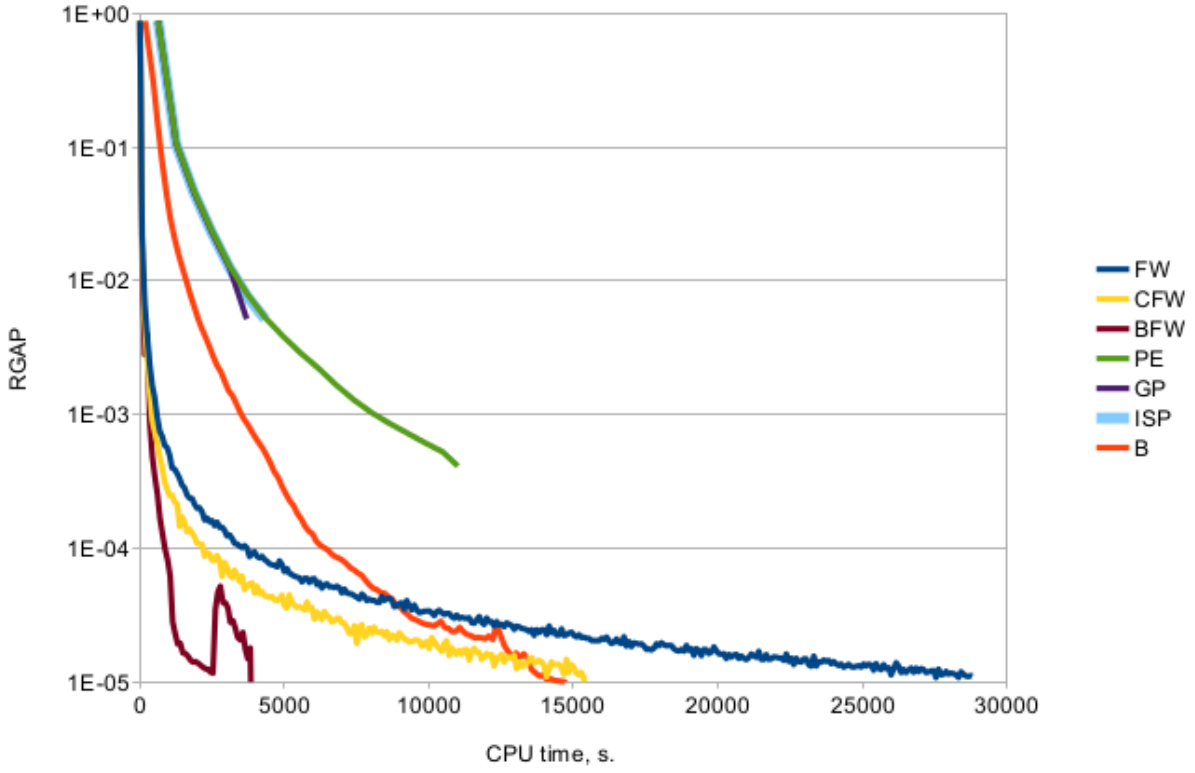


Figure 9: Convergence of algorithms. Chicago Regional instance



References

N. Andréasson, A. Evgrafov, and M. Patriksson. An Introduction to Continuous Optimization. Studentlitteratur, 2007.

H. Bar-Gera. Origin-based algorithm for the traffic assignment problem. *Transportation Science*, 36(4):398–417, 2002.

H. Bar-Gera. Traffic assignment by paired alternative segments. *Transportation Research Part B: Methodological*, 44(8-9):1022 – 1046, 2010.

A. Chen and R. Jayakrishnan. A path-based gradient projection algorithm: effects of equilibration with a restricted path set under two flow update policies. In *Transportation Research Board Annual Meeting*, 1998.

S. C. Dafermos and F.T. Sparrow. The traffic assignment problem for a general network. *Journal of research of the National Bureau of Standards. B, Mathematical sciences*., 73B:91–118, 1969.

S. Dasgupta, C.H. Papadimitriou, and U.V. Vazirani. Algorithms. McGraw-Hill Education (India) Pvt Limited, 2006.

J. Davies, J. Valero, and D. Young. The Auckland transport models project: Overview and use to date. In *Australasian Transport Research Forum 2009 Proceedings*, 2009.

R. Dial. A path-based user-equilibrium traffic assignment algorithm that obviates path storage and enumeration. *Transportation Research Part B: Methodological*, 40(10):917–936, 2006.

M. Florian and D. Hearn. Network equilibrium models and algorithms. In C.L. Monma M.O. Ball, T.L. Magnanti and G.L. Nemhauser, editors, Network Routing, volume 8 of Handbooks in *Operations Research and Management Science*, chapter 6, pages 485 – 550. Elsevier, 1995.

M. Florian, J. Gúelat, and H. Spiess. An efficient implementation of the PARTAN variant of the linear approximation method for the network equilibrium problem. *Networks*, 17:319–339, 1987.

M. Florian, I. Constantin, and D. Florian. A new look at projected gradient method for equilibrium assignment. *Transportation Research Record: Journal of the Transportation Research Board*, 2090:10–16, 2009.

M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3(1-2):95–110, 1956. ISSN 1931-9193.

M. Fukushima. A modified frank-wolfe algorithm for solving the traffic assignment problem. *Transportation Research Part B: Methodological*, 18(2):169–177, 1984.

G. Gentile. Linear user cost equilibrium: a new algorithm for traffic assignment. Working paper, April 2009.

A. V. Goldberg, H. Kaplan, and R. F. Werneck. Reach for A: Efficient point-to-point shortest path algorithms. In *Workshop on Algorithm Engineering & Experiments*, pages 129–143, 2006.

D. W. Hearn, S. Lawphongpanich, and J. A. Ventura. Finiteness in restricted simplicial decomposition. *Oper. Res. Lett*., 4(3):125–130, October 1985. ISSN 0167-6377.

S. Inoue and T. Maruyama. Computational experience on advanced algorithms for user equilibrium traffic assignment problem and its convergence error. *Procedia - Social and Behavioral Sciences*, 43(0):445 – 456, 2012. ISSN 1877-0428. 8th International Conference on Traffic and Transportation Studies (ICTTS 2012).

16R. Jayakrishnan, W. K. Tsai, J. Prashker, and S. Rajadhyaksha. A faster path-based algorithm for traffic assignment. *Transportation Research Record*, 1443:75–83, 1994.

A. Kumar and S. Peeta. An improved social pressure algorithm for the static deterministic user equilibrium traffic assignment problem. *Transportation Research Board of the National Academics*, Washington, D.C., 2011.

T. Larsson and M. Patriksson. Simplicial decomposition with disaggregated representation for the traffic assignment problem. *Transportation Science*, 26:4–17, 1991.

T. Larsson, M. Patriksson, and C. Rydergren. Applications of simplicial decomposition with nonlinear column generation to nonlinear network flows. In of *Lecture Notes in Economics and Mathematical Systems*, pages 346–373. Springer-Verlag, 1997.

D.-H. Lee, Y. Nie, A. Chen, and Y. C. Leow. Link- and path-based traffic assignment algorithms: Computational and statistical study. *Transportation Research Record: Journal of the Transportation Research Board*, 1783:80–88, 2002.

M. Mitradjieva and P. O. Lindberg. The stiff is moving - conjugate direction Frank-Wolfe methods with applications to traffic assignment. *Transportation Science, Articles in Advance*, INFORMS, pages 1–14, 2012.

Y. Nie. A class of bush-based algorithms for the traffic assignment problem. *Transportation Research*, 44:73–89, 2010.

Y. Nie. A note on Bar-Gera's algorithm for the origin-based traffic assignment problem. *Transportation Science*. Under review, 2011.

J. D. Ortúzar and L.G. Willumsen. *Modelling Transport*. J. Wiley, Chichester New York, 2001.

M. Patriksson. The traffic assignment problem: models and methods. *Topics in transportation*. VSP, 1994.

D. J. Pearce and P. H. J. Kelly. A dynamic topological sort algorithm for directed acyclic graphs. *J. Exp. Algorithmics*, 11:2006, 2006.

W. B. Powell and Y. Sheffi. The Convergence of Equilibrium Algorithms with Predetermined Step Sizes. *Transportation Science*, 16(1), February 1982.

Y. Sheffi. Urban transportation networks: equilibrium analysis with mathematical programming methods. Prentice-Hall, Englewood Cliffs, N.J., 1985.

H. Slavin, J. Brandon, and A. Rabinowicz. An empirical comparison of alternative user equilibrium traffic assignment methods. In *European Transport Conference*, 2006.

H. Slavin, P. Ricotta, J. Brandon, A. Rabinowicz, and S. Sundaram. A new traffic assignment method for small and medium communities. In *Tools of the Trade Conference*, September 2010.

Z. Tianran, Y. Chao, and C. Dongdong. Computational study of origin-based user equilibrium traffic assignment algorithms. In *Intelligent Computation Technology and Automation* (ICICTA), 2010 International Conference on, volume 1, pages 1002 –1006, may 2010.

J.G. Wardrop. Some theoretical aspects of road traffic research. Institution of Civil Engineers, 1952.

A. Weintraub, C. Ortiz, and J. Gonzalez. Accelerating convergence of the Frank-Wolfe algorithm. *Transportation Research Part B: Methodological*, 19(2):113–122, April 1985.

Z. Zhou and M. Martimo. Computational study of alternative methods for static traffic equilibrium assignment. In *Proc. World Conf. Transport* Res. Soc. WCTRS, Lisbon, Portugal, 2010.