The Stiff is Moving - Conjugate Direction Frank-Wolfe Methods with Applications to Traffic Assignment*

Maria Mitradjieva[1] and Per Olov Lindberg[2]

[1]Linköping University, SE-58183 Linköping

[2]Dept Transport Science, KTH, SE-100 44 Stockholm, Sweden,

e-mail: polin@kth.se

Abstract

We present versions of the Frank-Wolfe method for linearly constrained convex programs, in which consecutive search directions are made conjugate. Preliminary computational studies in a MATLAB environment applying pure Frank-Wolfe, Conjugate direction Frank-Wolfe (CFW), Bi-conjugate Frank-Wolfe (BFW) and "PARTANized" Frank-Wolfe methods to some classical Traffic Assignment Problems show that CFW and BFW compare favorably to the other methods. This spurred a more detailed study, comparing our methods to Bar-Gera's origin-based algorithm. This study indicates that our methods are competitive for accuracy requirements suggested by Boyce et al. We further show that CFW is globally convergent. We further point at independent studies by other researchers that show that our methods compare favourably with recent bush-based and gradient projection algorithms on computers with several cores.

Introduction

The Frank-Wolfe (FW) method (Frank & Wolfe, 1956) was originally suggested for quadratic programming problems, but the original paper also noted that it could be applied to linearly constrained convex programs. The main usage of the FW method has been in routing problems in the telecom and traffic areas, where it is usually attributed to (Fratta, Gerla, & Kleinrock, 1973) and (LeBlanc, 1973), respectively; see however (Bruynooghe, Gibert, & Sakarovitch, 1969) also. It still is the most popular method in these areas. According to the survey (Ouorou, Mahey, & Vial, 2000): "The reason for this is twofold. The method easily generates feasible solutions from shortest path calculations and enjoys fast convergence in the early iterations". However, to cite (Patriksson, 1994, p. 101): "The unsatisfactory performance of the Frank-Wolfe algorithm, in particular at the vicinity of an optimal solution, was observed quite early." The

*The title of the paper alludes to a sketch that was popular in Sweden in the 50's: Two men are having a beer in a bar. First man: "Hey, isn't that Fingal Olsson sitting over there?" Second man: " No way, he is dead." First man: "No, can't be. He is moving!"

underlying reason for the latter behavior is that the search directions of the FW-method tend to become orthogonal to the gradient of the objective function close to the optimum, leading to extreme zigzagging (e.g. (Patriksson, 1994, p. 102)). Hence, despite its wide usage in practice, the poor rate of convergence of the FW method has led methods oriented researchers to consider FW a "dead method". But as will be seen in the current paper, the FW method is not quite dead yet: *the stiff is in fact moving!* FW has further been a yardstick to measure new improved algorithms against. Wouldn't it be fair if FW was given the chance to shape up and fight back!

Since the deficiencies of the FW method were observed early, many attempts have been made to improve upon it, in particular finding better search directions than the "FW direction", (e.g. (Fuku-shima, 1984), (LeBlanc, Helgason, & Boyce, 1985), (Lupi, 1986)). The main proponent of these attempts is the PARTAN approach (LeBlanc et al., 1985), where one obtains better directions by taking PARTAN steps (e.g. (Luenberger, 1984, p. 255-257), (Shah, Buehler, & Kempthorne, 1964)) between pure Frank-Wolfe steps.

There are also other more complex extensions of the FW method, such as Restricted Simplicial Decomposition (RSD) (Hearn, Lawphongpanich, & Ventura, 1985) and, Disaggregate Simplicial Decomposition (DSD) (Larsson & Patriksson, 1992), where one solves auxiliary optimization problems to determine the next iteration point. In the current paper, however, we limit ourselves to simple modifications of the FW direction, not more complex than that they could be implemented as macros or scripts in standard software for Traffic Assignment.

To be specific we suggest a simple improvement of the FW directions: making them *conjugate*. Every textbook on nonlinear programming advocates conjugate gradients over pure gradient directions (i.e. steepest descent). In fact, PARTAN owns some of its motivation to conjugate directions. Indeed, applied to an unconstrained convex quadratic problem, PARTAN will generate conjugate directions (Luenberger, 1984, p. 255-257). We introduced conjugate direction FW methods in (Daneva & Lindberg, 2003a), and in (Daneva & Lindberg, 2003b) we extended them to nonconvex problems.

In 2002 Bar-Gera's origin-based algorithm (OBA) (Bar-Gera, 2002) was published, based on his thesis. OBA works with a "restricting" sub-network for each origin, to which arcs are added and deleted, and for which origin-based flows are updated in a quasi-Newton fashion. This was the situation in 2004, when the current paper was first submitted.[1]

After this there have appeared other, similar methods, inspired by OBA. In 2006 came Dial's algorithm B (Dial, 2006). There, path costs in the restricting sub-networks (now called bushes) are equilibrated using flow shifts. Dial reports impressive computational experience. The same idea was further developed by Nie (Nie, 2010). Another exponent is Gentile (Gentile, 2009). He also uses bushes, but reversed, with traffic heading for the same destination. In these bushes he recursively solves myopic optimization problems on how to route traffic to the neighbouring nodes, and then takes steps in that direction. He also reports impressive computational experience.

[1]The current paper was submitted to this journal in 2004. However, the editors wanted us to perform direct computational comparisons with Bar-Gera's OBA (Origin Based Algorithm) (Bar-Gera, 2002). We downloaded Bar-Gera's code, but could not make it run on our Unix system. We did however make some comparisons with DSD (Disaggregate Simplicial Decomposition, (Larsson & Patriksson, 1992)). After that Dr Mitradjieva (then Mrs Daneva) became pregnant and later she became ill, and after that she changed direction of studies, so the revision of the current paper was put on low heat. However, recently there has appeared results that might make a direct comparison with OBA unneccessary, (Caliper, 2010) and (Zhou, Brignone, & Clarke, 2010). Therefore we have submitted the current paper with only minor modifications, to care for questions by the referees and references to the latest computational developement.

Another, semiclassical, approach was followed by Florian et al (Florian, Constantin, & Florian, 2009), who use projected gradients in the space of path flows for a given OD-pair, also reporting impressive computational experience.

This year (2010) there have appeared two papers, that might make the direct computational comparisons, demanded by the editors (se footnote), superflous. Both Caliper Corp. (Caliper, 2010) and CityLabs (Zhou et al., 2010) have taken up our methods and made computational comparisons between FW, CFW, BFW, and other methods. Citilabs have compared with OBA and GP (Gradient Projection), and Caliper with Dial's algorithm B (termed OUE by them). They both show that BFW is competitive down to relative gaps of 10^{-4}. Further they both get BFW as the winner when using several cores, which now is common in modern PC's.

In the next section we review the FW method and some of its modifications. Section 2 is devoted to conjugate direction techniques and our own conjugate direction FW method (CFW) and bi-conjugate FW method (BFW). In particular we show that CFW is globally convergent. In Section 3 we apply MATLAB versions of pure FW, partanized FW (PFW), CFW and BFW to some classical traffic assignment problems. Measuring performance per major iteration, it is demonstrated that FW is clearly outperformed by PFW, which in turn is clearly outperformed by CFW and BFW. We further make direct comparisons of C-versions of our methods and DSD, which latter looses heavily except on the smallest problem. Finally we also make some indirect comparison of our C-versions with OBA, indicating that for levels of accuracy advocated by (Boyce, Ralevic-Dekic, & Bar-Gera, 2002), BFW might outperform Origin-Based methods. We also note that (Caliper, 2010) and (Zhou et al., 2010) suppport our findings.

1 The Frank-Wolfe Method and Modifications

Let $f : X \mapsto \mathbb{R}$ be a twice continuously differentiable convex function on the compact convex polyhedral set $X \subset \mathbb{R}^n$. The methods discussed in this paper apply to the problem of minimizing f over X:

$$\text{(P)} \qquad f^* \triangleq \min_{\mathbf{x}\in X} f(\mathbf{x}).$$

By continuity of f, (P) has a solution, unique if f is strictly convex.

The problem (P) is in principle well-behaved, but in the traffic and telecom applications we aim at, it is a convex cost multicommodity flow problem, which typically becomes very large. "Unfortunately, even for graphs of moderate sizes, the problem to be solved has exceedingly large dimensions. For real-life problems, the models may have tens or hundreds of thousands of constraints and hundreds of thousands or millions of variables" (Ouorou et al., 2000). Thus, one needs to utilize structure. The Frank-Wolfe algorithm allows efficient utilization of the multicommodity structure of (P). We briefly discuss it in the next section.

1.1 The Frank-Wolfe Method

The conventional FW algorithm (Frank & Wolfe, 1956) was introduced for solving general linearly constrained quadratic and convex problems. At iteration k, FW approximates f by linearizing at the current iterate $\mathbf{x}_k$, giving an *affine minorant* f_k to f:

$$f_k(\mathbf{x}) \triangleq f(\mathbf{x}_k) + \nabla f(\mathbf{x}_k)^T(\mathbf{x} - \mathbf{x}_k).$$

The *first* step in FW method is the *determination of the search direction*, which is done by minimizing f_k over X. This is a linear program

$$(\mathrm{LP}_k) \qquad f_k^* \triangleq \min_{\mathbf{x}\in X} f_k(\mathbf{x}),$$

the solution of which we denote by $\mathbf{y}_k^{FW}$. This *Frank-Wolfe point* $\mathbf{y}_k^{FW}$ is chosen as the *"point of sight"* in that the search direction is chosen to be $\mathbf{d}_k^{FW} = \mathbf{y}_k^{FW} - \mathbf{x}_k$, the *Frank-Wolfe direction*. Note that f_k^* is a *lower bound*, LBD_k, to f^* (since f_k underestimates f) a fact which may be used in termination criteria. In the telecom and traffic applications, (LP_k) decomposes into a set of shortest path problems, (e.g. (Patriksson, 1994, p. 97)).

The *second* step of the FW method is to perform a *line search* in the FW-direction, i.e., a one-dimensional minimization of $f(\mathbf{x})$, along the line segment between the current iterate $\mathbf{x}_k$ and the FW-point $\mathbf{y}_k^{FW}$. The point where this minimum is achieved (at least approximately) is chosen as the next iterate $\mathbf{x}_{k+1}$. Note that $f(\mathbf{x}_{k+1})$ is an *upper bound*, UBD_k, to f^*.

As already mentioned this algorithm is very easy to implement. Due to the slow asymptotic convergence, many modifications have been suggested. The line search can e.g. be modified in order to take predetermined (Powell & Sheffi, 1982) or longer steps (Weintraub, Ortiz, & González, 1985). In still other modifications the linear subproblem is modified in order to avoid generating extreme point solutions (Larsson, Patriksson, & Rydergren, 1997). In another group of papers, (Fuku-shima, 1984), (LeBlanc et al., 1985), (Lupi, 1986), the search direction is combined with previous ones. The current paper belongs to this class. Fukushima (Fuku-shima, 1984) chooses his point of sight $\mathbf{s}_k^F$ as a convex combination of previous FW-points $\mathbf{y}_i^{FW}$, i.e., $\mathbf{d}_k^F = \mathbf{s}_k^F - \mathbf{x}_k = \sum_{i=1}^k \lambda_i \mathbf{y}_i^{FW} - \mathbf{x}_k$, $\sum_{i=1}^{k-1} \lambda_i = 1$, $\lambda_i \geq 0$. In practice the combination is usually taken over the latest two FW-points $\mathbf{y}_{k-1}^{FW}$ and $\mathbf{y}_k^{FW}$, with $\lambda_{k-1} = \lambda_k = 0.5$. Lupi (Lupi, 1986) chooses his search direction $\mathbf{d}_k^L$ as a convex combination of the FW-direction $\mathbf{d}_k^{FW}$ and the previous direction $\mathbf{d}_{k-1}^L$, such that $\mathbf{d}_k^L$ and $\mathbf{d}_{k-1}^L$ are orthogonal. In the "modified Fukushima" approach Arezki (Arezki, 1986) chooses $\mathbf{s}_k^{MF}$ as a convex combination of $\mathbf{y}_{k-1}^{FW}$ and $\mathbf{y}_k^{FW}$, and chosen so that the direction $\mathbf{d}_k^{MF} = \mathbf{s}_k^{MF} - \mathbf{x}_k$ is orthogonal to the previous FW-direction $\mathbf{d}_{k-1}^{FW}$.

The method of parallel tangents (PARTAN) originates in attempts to avoid zig-zagging in gradient based methods for unconstrained optimization (Luenberger, 1984, p. 254), (Shah et al., 1964), by performing an extra line search at each iteration. In the PARTAN technique for unconstrained optimization, at the iterate $\mathbf{x}_k$, one first performs a line search in the gradient direction, giving the intermediate point $\mathbf{v}_k$. Then one applies an extra line search from $\mathbf{v}_k$ in the direction $\mathbf{d}_k^P = \mathbf{v}_k - \mathbf{x}_{k-1}$, giving the new iterate $\mathbf{x}_{k+1}$.

In the partanized FW method, the line search in the gradient direction is replaced by one in the FW direction $\mathbf{d}_k^{FW}$, giving the intermediate point $\mathbf{v}_k$. At $\mathbf{v}_k$, a "PARTAN" step is performed, i.e. a linesearch along the line connecting $\mathbf{v}_k$ and $\mathbf{x}_{k-1}$, giving the new iterate $\mathbf{x}_k$. An intrinsic difficulty with the PARTAN technique in the Frank-Wolfe context (for telecom and traffic problems) is the determination of appropriate bounds on the step lengths to stay feasible. In the early papers these bounds were determined individually for the first few steps. Later, (Arezki & van Vliet, 1990) and (Florian, Guelat, & Spiess, 1987) determined analytical recursion formulas for these bounds. In spite of improved convergence characteristics, the PARTAN approach has not superseded the Frank-Wolfe method in the telecom and transportation areas. The reason for this might be its complexity:

- it requires two line searches per major iteration – one FW step and one PARTAN step;

- it requires a somewhat complex recursive updating of the bounds for the PARTAN step lengths.

Arezki (Arezki, 1986) (Arezki, 1987) compares several search direction modifications, including the Fukushima (Fukushima, 1984), Lupi (Lupi, 1986), "modified Fukushima" (Arezki, 1986), and PARTAN (LeBlanc et al., 1985) approaches. He (Arezki, 1986) concludes that the best performance is given by "Heuristic PARTAN", which eliminates "backward" steps from partanized FW (i.e. steps on the $\mathbf{x}_{k-1}$ side of $\mathbf{v}_k$. For this reason, we will use Heuristic PARTAN as the modified FW-method to compare our methods with.

2 Conjugate Direction Frank-Wolfe Methods

In this section we describe our modifications of the FW method. In conjugate directions methods (e.g. (Luenberger, 1984, Ch. 8)) for unconstrained convex quadratic optimization, one performs line searches consecutively in a set of directions, $\mathbf{d}_1, \cdots, \mathbf{d}_n$, mutually conjugate with respect to the hessian H of the objective (i.e. fulfilling $\mathbf{d}_i^T H \mathbf{d}_j = 0$ for $i \neq j$). In $\mathbb{R}^n$ the optimum then is identified after n line searches (Luenberger, 1984, p. 241, Expanding Subspace Theorem).

In conjugate gradient methods, one obtains conjugate directions by "conjugating" the gradient direction with respect to the previous search direction; i.e., $\mathbf{d}_k \triangleq \nabla f(\mathbf{x}_k) + \beta_k \mathbf{d}_{k-1}$, with β_k chosen so that $\mathbf{d}_k$ is conjugate to $\mathbf{d}_{k-1}$. (In the quadratic case, $\mathbf{d}_k$ then in fact becomes conjugate to all previous directions $\mathbf{d}_1, \cdots, \mathbf{d}_{k-1}$, e.g. (Luenberger, 1984, p. 245, Conjugate Gradient Theorem)). A similar trick can be applied to the FW method.

2.1 The Conjugate Frank-Wolfe Method, CFW

Let $\mathbf{x}_k$ be the current iterate (see Figure 1), found by a line search with steplength τ_{k-1} in the direction $\mathbf{d}_{k-1}^{CFW} = \mathbf{s}_{k-1}^{CFW} - \mathbf{x}_{k-1}$, from $\mathbf{x}_{k-1}$ towards the point of sight $\mathbf{s}_{k-1}^{CFW} \in X$. At $\mathbf{x}_k$ we solve the linearized problem (LP_k) giving $\mathbf{y}_k^{FW} \in X$. In the FW method the new search direction is taken as $\mathbf{d}_k^{FW} = \mathbf{y}_k^{FW} - \mathbf{x}_k$. In the *conjugate direction FW*

method (CFW), we want to choose the search direction as

$$\mathbf{d}_k = \mathbf{d}_k^{FW} + \beta_k \mathbf{d}_{k-1}^{CFW}, \tag{1}$$

where β_k is chosen to make $\mathbf{d_k}$ conjugate to $\mathbf{d}_{k-1}^{CFW}$ with respect to the hessian H_k of f at $\mathbf{x}_k$. The problem with this choice of search direction is that it is hard to determine the maximal feasible step length. (This is an inherent problem for modifications of the FW direction, for traffic problems. This has to do with that the constraint set in the space of link flows is not given explicitly. Instead it is generated from the inside through the generation of the extreme points $\mathbf{y}_k^{FW}$.) However, we can equivalently (up to scaling) choose $\mathbf{d}_k^{CFW} = \mathbf{s}_k^{CFW} - \mathbf{x}_k$, where the point of sight

$$\mathbf{s}_k^{CFW} = \alpha_k \mathbf{s}_{k-1}^{CFW} + (1-\alpha_k)\mathbf{y}_k^{FW}, \quad 0 \le \alpha_k \le 1, \tag{2}$$

is a convex combination of $\mathbf{y}_k^{FW}$ and $\mathbf{s}_{k-1}^{CFW}$, both belonging to X. In this way the maximum step length (given only the information that $\mathbf{y}_k^{FW}$ and $\mathbf{s}_{k-1}^{CFW}$ both belong to X) is always 1 . Observe, in Figure 1, that the "residual" search direction $\overline{\mathbf{d}}_{k-1} \triangleq \mathbf{s}_{k-1}^{CFW} - \mathbf{x}_k$ also can be written $\overline{\mathbf{d}}_{k-1} = (1-\tau_{k-1})\mathbf{d}_k^{CFW}$

Using these expressions and (2), we have

$$\begin{aligned} \mathbf{d}_k^{CFW} &= \mathbf{s}_k^{CFW} - \mathbf{x}_k = \\ &\alpha_k \mathbf{s}_{k-1}^{CFW} + (1-\alpha_k)\mathbf{y}_k^{FW} - \mathbf{x}_k = \\ &\alpha_k(\mathbf{s}_{k-1}^{CFW} - \mathbf{x}_k) + (1-\alpha_k)(\mathbf{y}_k^{FW} - \mathbf{x}_k) = \\ &\alpha_k \overline{\mathbf{d}}_{k-1} + (1-\alpha_k)\mathbf{d}_k^{FW} = \\ &\mathbf{d}_k^{FW} + \alpha_k(\overline{\mathbf{d}}_{k-1} - \mathbf{d}_k^{FW}). \end{aligned} \tag{3}$$

Since $\overline{\mathbf{d}}_{k-1}$ is parallel to $\mathbf{d}_{k-1}^{CFW}$ we might as well make $\mathbf{d}_k^{CFW}$ conjugate to $\overline{\mathbf{d}}_{k-1}$ (unless the step length $\tau_{k-1} = 1$, in which case $\overline{\mathbf{d}}_{k-1} = 0$). We get, using the notation $H_k = H(\mathbf{x}_k)$,

$$0 = \overline{\mathbf{d}}_{k-1}^T H_k \mathbf{d}_k^{CFW} = \overline{\mathbf{d}}_{k-1}^T H_k [\mathbf{d}_k^{FW} + \alpha_k(\overline{\mathbf{d}}_{k-1} - \mathbf{d}_k^{FW})]. \tag{4}$$

We will later prove the global convergence of CFW. For this we need the mappings involved to be closed. Due to this, the determination of α_k has to be handled with some care. From (4) we get

$$\alpha_k = \frac{\overline{\mathbf{d}}_{k-1}^T H_k \mathbf{d}_k^{FW}}{\overline{\mathbf{d}}_{k-1}^T H_k (\mathbf{d}_k^{FW} - \overline{\mathbf{d}}_{k-1})}, \tag{5}$$

There are several problems with (5). First we need to have $\alpha_k \in [0,1]$. Second, we need to avoid $\alpha_k = 1$, because then $\mathbf{d}_k^{CFW} = \overline{\mathbf{d}}_{k-1}$ by (3), and we get no new direction. Third, there is a problem when the denominator is zero, which

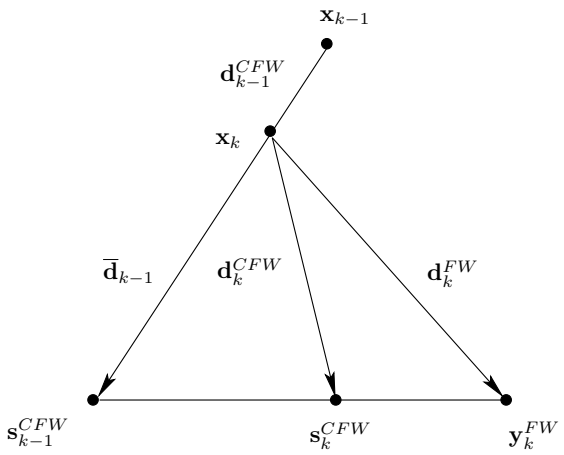

Figure 1: Determination of the search direction for the CFW algorithm

happens e.g. when $\tau_{k-1} = 1$. Further note, that for $\alpha_k = 0$, we get $\mathbf{d}_k^{CFW} = \mathbf{d}_k^{FW}$, i.e. the FW-direction, which is a conservative choice. Finally, we need to resolve these problems in a way that makes the resulting mapping closed, in order to be able to prove global convergence of CFW, which will be done in section 2.4.

Introducing N_k and D_k for the numerator and denominator in (5), we determine α_k as follows

$$\alpha_k = \overline{\alpha}(D_k, N_k) \triangleq \begin{cases} N_k/D_k, & \text{if } D_k \neq 0 \text{ and } N_k/D_k \in [0, 1-\delta], \\ 1-\delta, & \text{if } D_k \neq 0 \text{ and } N_k/D_k > 1-\delta, \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

for some given small $\delta > 0$.

This mapping is not closed as stated, but we will embed it in a closed point- to-set mapping $\overline{\mathbf{A}}$ by simply defining[2]

$$\overline{\mathbf{A}}(D_k, N_k) \triangleq [1-\delta] \tag{7}$$

Since $\overline{\mathbf{A}}$ is constant and closed-valued, it obviously is closed.

2.2 Outline of the CFW Algorithm

The CFW can be summarized in the following steps:

`Step 0.` (Initialization) Choose a tolerance $\varepsilon > 0$. Set iteration counter $k = 0$, and best lower bound $BLB_0 = -\infty$. Choose $\mathbf{x}_0 \in X$ to be a feasible solution for (P).

`Step 1.` (FW direction determination) At $\mathbf{x}_k$ compute $\mathbf{y}_k^{FW}$, and $\mathbf{d}_k^{FW} = \mathbf{y}_k^{FW} - \mathbf{x}_k$.

[2]This form of $\overline{\mathbf{A}}$ was suggested by one of the referees. It made the former complex proof of the closedness of $\overline{\mathbf{A}}$ unnecessary. Moreover, it allowed the present more natural form of α_k

`Step 2.` **(CFW direction determination)** *If* $k = 0$ or $\tau_{k-1} = 1$ set $\mathbf{d}_k^{CFW} = \mathbf{d}_k^{FW}$, *otherwise* determine α_k according to (6) and $\mathbf{s}_k^{CFW}$ according to (2). Set $\mathbf{d}_k^{CFW} = \mathbf{s}_k^{CFW} - \mathbf{x}_k$.

`Step 3.` (Line search) Find the step τ_k, which minimizes the objective $f(\mathbf{x}_k + \tau_k \mathbf{d}_k^{CFW})$ over $0 \leq \tau_k \leq 1$, giving an upper bound to (P): $UBD_k = f(\mathbf{x}_k + \tau_k \mathbf{d}_k^{CFW})$.

`Step 4.` (Update) Set $\mathbf{x}_{k+1} = \mathbf{x}_k + \tau_k \mathbf{d}_k^{CFW}$ and $k = k + 1$.

`Step 5.` (Convergence test) Update $BLB_k := \max\{BLB_{k-1}, f_k(\mathbf{y}_k^{FW})\}$ and terminate if the relative gap between the upper and the lower bound is smaller than ε. GOTO Step 1.

To make step 5 precise,we define the *gap* g_k at iteration k as $g_k = UBD_k - BLB_k$ and the *relative gap* as $g_k/|BLB_k|$.

Between the iterations the CFW algorithm needs to keep two n-vectors in memory, to be able to find the next conjugate direction $\mathbf{d}_k^{CFW}$, and has to perform only one line search to find the next iteration point.

The choice of a conjugate search direction in CFW, as opposed to other directions, might seem ad hoc, but it is in fact the best possible using available information. It follows namely from the above mentioned Expanding Subspace Theorem that if $\mathbf{d}_k^{CFW}$ is made conjugate to $\mathbf{d}_{k-1}^{CFW}$ then (in the quadratic case and assuming exact line searches) the line-search along $\mathbf{d}_k^{CFW}$ will pass through the optimum in the space spanned by $\mathbf{d}_{k-1}^{CFW}$ and $\mathbf{d}_k^{FW}$.

As the algorithm progresses, and we come closer and closer to the optimum, the objective gets successively better described by a quadratic. Thus the line in the search direction will pass successively closer to the optimum in said space, spanned by $\mathbf{d}_{k-1}^{CFW}$ and $\mathbf{d}_k^{FW}$. Our computer runs show that CFW (and BFW below) in fact take steps of size $\tau_k < 1$ in the majority of the iterations, and hence take close to optimal steps in said spaces.

The figures in section 3.2 bear witness of the efficiency of the steps of CFW. For instance in figure 3, FW and CFW seem to be equally efficient when the algorithms both have reached a linear development in the log-log-diagrams. But this is an optical illusion. We see in the figure that FW goes from a relative gap of 10^{-4} to 10^{-5} in the iterations 1 000 to 10 000 (approximately) i.e. in 9 000 iterations, whereas CFW makes the same progress in iterations 100 to 1 000, i.e. in 900 iterations, an improvement by a factor of 10.

So how about when we take a step of size $\tau = 1$? One way to explain the slow convergence of FW is that the current iterate $\mathbf{x}_k$ is a convex combination of all generated extreme points. And the weights of extreme points not belonging the optimal facet die out only slowly. If on the other hand, we take a step $\tau = 1$ in CFW (or BFW), the weights of extreme points with positive weight in $\mathbf{x}_k$, but zero weight in $\mathbf{s}_k^{CFW}$, get zeroed out. This will improve the possibilities to approximate the optimal facet. These two mechanism together explain the superiority of CFW (and BFW) over plain FW.

2.3 The Bi-Conjugate Frank-Wolfe Method, BFW

In order to put CFW on an equal footing with PFW concerning memory consumption, we have developed it one step further, namely to apply conjugation with respect to the last *two* directions instead of only the last one. The

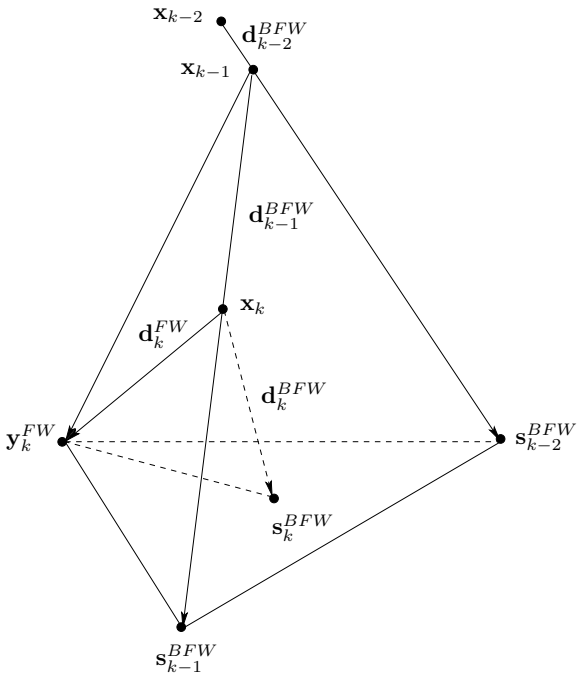

Figure 2: Determination of the search direction for the BFW algorithm

computations are slightly more complicated, but the differences in implementation are minor. As a matter of fact this modification outperforms CFW, at least for high iteration counts.

To describe the idea, let $\mathbf{x}_k$ be the current iterate (see Figure 2), found by a line search in the direction $\mathbf{d}_{k-1}^{BFW}$, from $\mathbf{x}_{k-1}$ towards $\mathbf{s}_{k-1}^{BFW} \in X$ ($\mathbf{x}_{k-1}$ in turn found by a line search in the direction $\mathbf{d}_{k-2}^{BFW}$ from $\mathbf{x}_{k-2}$ towards $\mathbf{s}_{k-2}^{BFW}$). At $\mathbf{x}_k$ we solve the linearized problem (LP_k) giving $\mathbf{y}_k^{FW} \in X$.

In the CFW method, the new search direction is $\mathbf{d}_k^{CFW} = \mathbf{s}_k^{CFW} - \mathbf{x}_k$, where the point of sight $\mathbf{s}_k^{CFW}$ is a convex combination of the FW point $\mathbf{y}_k^{FW}$ and the previous point of sight $\mathbf{s}_{k-1}^{FW}$. In the *bi-conjugate FW method* (BFW) we choose, in similarity, $\mathbf{d}_k^{BFW} = \mathbf{s}_k^{BFW} - \mathbf{x}_k$, where the point of sight $\mathbf{s}_k^{BFW}$ is a convex combination of $\mathbf{y}_k^{FW}$ and the *two* previous points of sight $\mathbf{s}_{k-1}^{BFW}$ and $\mathbf{s}_{k-2}^{BFW}$. Hence, for appropriate weights $\beta_k^i \geq 0$, with $\sum_{i=0}^{2} \beta_k^i = 1$,

$$\mathbf{d}_k^{BFW} = \mathbf{s}_k^{BFW} - \mathbf{x}_k = \beta_k^0 \mathbf{y}_k^{FW} + \beta_k^1 \mathbf{s}_{k-1}^{BFW} + \beta_k^2 \mathbf{s}_{k-2}^{BFW} - \mathbf{x}_k.$$

Or using $\mathbf{d}_k^{FW} = \mathbf{y}_k^{FW} - \mathbf{x}_k$,

$$\mathbf{d}_k^{BFW} = \beta_k^0 \mathbf{d}_k^{FW} + \beta_k^1 (\mathbf{s}_{k-1}^{BFW} - \mathbf{x}_k) + \beta_k^2 (\mathbf{s}_{k-2}^{BFW} - \mathbf{x}_k). \tag{8}$$

The coefficients β_k^i should be chosen to make $\mathbf{d}_k^{BFW}$ conjugate to the last two directions $\mathbf{d}_{k-1}^{BFW}$ and $\mathbf{d}_{k-2}^{BFW}$ with respect to the hessian H_k of f at $\mathbf{x}_k$, i.e.

$$(\mathbf{d}_k^{BFW})^T H_k \mathbf{d}_{k-1}^{BFW} = 0 \tag{9a}$$

$$(\mathbf{d}_k^{BFW})^T H_k \mathbf{d}_{k-2}^{BFW} = 0 \tag{9b}$$

The derivation of the formulas for $\beta_k^i,\ i = 0, 1, 2$ is given in Appendix A.

2.4 Convergence of the CFW Method

In this section we will prove the global convergence of CFW utilizing the *Convergence Theorem* (Zangwill, 1969, p. 91). The Convergence Theorem views an algorithm as an algorithmic (point-to-set) map $\mathbf{A}$, that maps the iteration point $\mathbf{x}_k$ to a set $\mathbf{A}(\mathbf{x}_k)$ to which $\mathbf{x}_{k+1}$ shall belong, $\mathbf{x}_{k+1} \in \mathbf{A}(\mathbf{x}_k)$. We state this theorem as given in (Zangwill, 1969, p. 91). Before that we need to give a definition:

Definition 1 *A point-to-set map* $\mathbf{A}$ *is closed at* $\mathbf{x}$ *if for all sequences* $\{\mathbf{x}_k\}_{k=1}^{\infty} \to \mathbf{x}$ *and* $\{\mathbf{y}_k\}_{k=1}^{\infty} \to \mathbf{y}$ *with* $\mathbf{y}_k \in \mathbf{A}(\mathbf{x}_k)$*, we have* $\mathbf{y} \in \mathbf{A}(\mathbf{x})$.

Theorem 1 (Convergence Theorem) *Let* $\mathbf{A}$ *be an algorithm on* X*, and suppose that, given* $\mathbf{x}_1$ *the sequence* $\{\mathbf{x}_k\}_{k=1}^{\infty}$ *is generated satisfying* $\mathbf{x}_{k+1} \in \mathbf{A}(\mathbf{x}_k)$*. Let a solution set* $\Gamma \subset X$ *be given and suppose*

i) all points $\mathbf{x}_k$ *are contained in a compact set* $S \subset X$

ii) there is a continuous function Z *on* X *such that*

a) if $\mathbf{x} \notin \Gamma$*, then* $Z(\mathbf{y}) < Z(\mathbf{x})$ *for all points* $\mathbf{y} \in \mathbf{A}(\mathbf{x})$

b) if $\mathbf{x} \in \Gamma$*, then* $Z(\mathbf{y}) \leq Z(\mathbf{x})$ *for all points* $\mathbf{y} \in \mathbf{A}(\mathbf{x})$

iii) the mapping $\mathbf{A(x)}$ *is closed at points outside* Γ.

Then the limit of any convergent subsequence of $\{\mathbf{x}_k\}$ *is a solution.*

In practice, one will use some form of inexact line-search. The standard stopping criteria, such as the Goldstein or Armijo criteria, e.g. (Luenberger, 1984, Section 7.5), do not go well along with conjugation. Therefore we will use a new condition, similar to the Wolfe condition.

Condition 1 (Acceptance condition for inexact line-search) At $\mathbf{x}_k$ the step $\tau_k \geq 0$ in the direction $\mathbf{d}_k$ is accepted if $\gamma \nabla f(\mathbf{x}_k)\mathbf{d}_k \leq \nabla f(\mathbf{x}_k + \tau_k \mathbf{d}_k)\mathbf{d}_k \leq 0$, where $\gamma \in [0, 1)$ is a given parameter.

This condition thus requires the slope in the search direction on the one hand to be nonpositive, on the other hand to be smaller in absolute value than a given fraction of the initial slope. Further note that for $\gamma = 0$ the condition gives exact line-search. Moreover the condition requires the direction $\mathbf{d}_k$ to be non-ascent at $\mathbf{x}_k$ to be meaningful, and in this case $f(\mathbf{x}_k + \tau_k \mathbf{d}_k) \leq f(\mathbf{x}_k)$.

We first prove that the inexact *line-search map* $S_\gamma(\mathbf{x}, \mathbf{d}) = \{\mathbf{y} | \mathbf{y} = \mathbf{x} + \tau\mathbf{d}, \gamma \nabla f(\mathbf{x})\mathbf{d} \leq \nabla f(\mathbf{x} + \tau\mathbf{d})\mathbf{d} \leq 0\}$, corresponding to condition 1, is closed.

Lemma 1 *The line-search map* $S_\gamma(\mathbf{x}, \mathbf{d})$ *is closed at* $(\mathbf{x}, \mathbf{d})$ *for* $\mathbf{d} \neq 0$

Proof: Assume that $\{\mathbf{x}_k\} \to \mathbf{x}$, $\{\mathbf{d}_k\} \to \mathbf{d} \neq 0$ and that $\mathbf{y}_k \in S_\gamma(\mathbf{x}_k, \mathbf{d}_k)$, with $\mathbf{y}_k \to \mathbf{y}$ (implying in particular that the $\mathbf{d}_k$ are non-ascent). Thus $\mathbf{y}_k = \mathbf{x}_k + \tau_k \mathbf{d}_k$ for appropriate τ_k. We get $\tau_k = \frac{|\mathbf{y}_k - \mathbf{x}_k|}{|\mathbf{d}_k|} \to \frac{|\mathbf{y} - \mathbf{x}|}{|\mathbf{d}|} := \tau$, giving $\mathbf{y} = \mathbf{x} + \tau\mathbf{d}$.

Since $\gamma\nabla f(\mathbf{x}_k)\mathbf{d}_k \leq \nabla f(\mathbf{x}_k + \tau_k\mathbf{d}_k)\mathbf{d}_k \leq 0$, we get by continuity of ∇f that $\gamma\nabla f(\mathbf{x})\mathbf{d} \leq \nabla f(\mathbf{x} + \tau\mathbf{d})\mathbf{d} \leq 0$, i.e. that $\mathbf{y} \in S_\gamma(\mathbf{x}, \mathbf{d})$. $\diamond$

Next we prove that the conjugate direction $\mathbf{d}_k^{CFW}$ of the CFW method is a descent direction provided that $\mathbf{x}_k$ is not optimal.

Lemma 2 *Assume that the sequence $\{\mathbf{x}_k\}$ is generated by CFW with inexact line-search according to Condition 1. Then, unless $\mathbf{x}_k$ is an optimal solution to (P), $\mathbf{d}_k^{CFW}$ is a descent direction for f at $\mathbf{x}_k$.*

Proof: We may assume that $\mathbf{x}_k$, and hence $\mathbf{x}_j, j = 0, .., k-1$ are nonoptimal. We must prove that the inner product $\nabla f(\mathbf{x}_k)^T\mathbf{d}_k^{CFW}$ is negative. According to (3) we have

$$\nabla f(\mathbf{x}_k)^T\mathbf{d}_k^{CFW} = \nabla f(\mathbf{x}_k)^T(\alpha_k\overline{\mathbf{d}}_{k-1} + (1-\alpha_k)\mathbf{d}_k^{FW}) = \alpha_k\nabla f(\mathbf{x}_k)^T\overline{\mathbf{d}}_{k-1} + (1-\alpha_k)\nabla f(\mathbf{x}_k)^T\mathbf{d}_k^{FW}$$

Since the Frank-Wolfe direction is a descent direction, unless $\mathbf{x}_k$ is optimal, and $\alpha_k \in [0, 1-\delta]$ (see 6), the second term is negative.

Concerning the first term, we will use induction. First note that $\mathbf{d}_0^{CFW} = \mathbf{d}_0^{FW}$, which is descent at $\mathbf{x}_0$. Assume inductively that $\mathbf{d}_j^{CFW}, j = 0, .., k-1$ are descent at $\mathbf{x}_j, j = 0, .., k-1$. Then by Condition 1, $\mathbf{d}_{k-1}^{CFW}$ is nonascent at $\mathbf{x}_k$, implying that $\nabla f(\mathbf{x}_k)^T\mathbf{d}_{k-1}^{CFW}$ and hence $\nabla f(\mathbf{x}_k)^T\bar{\mathbf{d}}_{k-1}$ are nonpositive, and hence $\mathbf{d}_k^{CFW}$ descent at $\mathbf{x}_k$ $\diamond$

The CFW algorithmic map $\mathbf{A}$ is composed of two maps. The first one is the direction finding map $\mathbf{D}$ that maps $(\mathbf{x}_k, \overline{\mathbf{d}}_{k-1})$ into $(\mathbf{x}_k, \mathbf{d}_k^{CFW})$ with $\mathbf{d}_k^{CFW} = \alpha_k\overline{\mathbf{d}}_{k-1} + (1-\alpha_k)\mathbf{d}_k^{FW}$, where $\alpha_k \in \overline{\mathbf{A}}(D_k, N_k)$. The other one is the line search map $\mathbf{S}_\gamma$ that maps $(\mathbf{x}_k, \mathbf{d}_k^{CFW})$ into $(\mathbf{x}_{k+1}, \overline{\mathbf{d}}_k)$ with $\mathbf{x}_{k+1} = \mathbf{x}_k + \tau_k\mathbf{d}_k^{CFW}$ and $\overline{\mathbf{d}}_k = (1-\tau_k)\mathbf{d}_k^{CFW}$, where τ_k is such that $y = \mathbf{x}_k + \tau\mathbf{d}_k^{CFW} \in \mathbf{S}_\gamma(\mathbf{x_k}, \mathbf{d_k^{CFW}})$. By Lemma 1, $\mathbf{S}_\gamma$ is closed.

Now we prove that the direction finding map $\mathbf{D}$ is closed at points which are not optimal.

Lemma 3 *The direction finding map $\mathbf{D}$ of CFW is closed at nonoptimal points.*

Proof: The direction finding map $\mathbf{D}$ is composed of two maps, the FW direction finding map $\mathbf{D}^{FW}$ that maps $\mathbf{x}_k$ into $(\mathbf{x}_k, \mathbf{d}_k^{FW})$, and the conjugation map $\mathbf{D}^{CFW}$, mapping $(\mathbf{x}_k, \mathbf{d}_k^{FW}, \overline{\mathbf{d}}_{k-1})$ into $(\mathbf{x}_k, \mathbf{d}_k^{CFW})$ with $\mathbf{d}_k^{CFW} = \alpha_k\overline{\mathbf{d}}_{k-1} + (1-\alpha_k)\mathbf{d}_k^{FW}$ and $\alpha_k \in \overline{\mathbf{A}}(D_k, N_k)$. Now our map can be written

$$\mathbf{D}(\mathbf{x}, \overline{\mathbf{d}}) = \mathbf{D}^{CFW}(\mathbf{D}^{FW}(\mathbf{x}), \overline{\mathbf{d}}) = \mathbf{D}^{CFW}(\mathbf{x}, \mathbf{d}^{FW}, \overline{\mathbf{d}}).$$

The closedness of the map $\mathbf{D}^{FW}(\mathbf{x})$ is proved in (Zangwill, 1969, Ch.8). The closedness of the conjugation map $\mathbf{D}^{CFW}$ is consequence of the closedness of the map $\overline{\mathbf{A}}$ and the fact that N_k and D_k are actually continuous functions of $\mathbf{x}_k$, $\mathbf{d}_k^{FW}$ and $\overline{\mathbf{d}}_{k-1}$. A detailed proof of the closedness of $\mathbf{D}^{CFW}$ is given in Appendix B. Hence the composite map $\mathbf{D}$ is also closed, since the feasible region X is compact (see e.g. (Zangwill, 1969, Corollary 4.2.1)). $\diamond$

Using the lemmas and the *Convergence Theorem* we can prove that the CFW algorithm presented above converges to a solution of the problem (P) or terminates at such a solution.

Theorem 2 (Global convergence of CFW) *Let the CFW algorithm be applied with inexact line search (according to Condition 1) and tolerance $\varepsilon = 0$. Then either the algorithm terminates at an optimal solution or every convergent subsequence of the iterates $\{\mathbf{x}_k\}$ converges to an optimal solution to (P).*

Proof: If the algorithm stops at $\mathbf{x}_k$ then $UBD_k = f(\mathbf{x}_k) = LBD_k$ by the convergence test (Step 5), implying that $\mathbf{x}_k$ is optimal. Otherwise suppose that the algorithm does not terminate at an optimal solution. Let algorithm $\mathbf{A}$ be the CFW approach. Let $\mathbf{x}_1 \in X$ be the starting point. We have to show that algorithm $\mathbf{A}$ satisfies the conditions of Theorem 1. Let Γ be the set of optimal solutions to problem (P). We consider a sequence of points $\{\mathbf{x}_k\}_{k=1}^{\infty}$ obtained by the rule $\mathbf{x}_{k+1} \in \mathbf{A}(\mathbf{x}_k)$. Now we check the three conditions of Theorem 1.

i) This condition is satisfied since every iteration point $\mathbf{x}_k$ lies in the feasible region X, which by assumption is compact.

ii) Let $Z = f$. Then a) follows since the conjugate direction $\mathbf{d}_k^{CFW}$ taken at every iteration is a descent direction at nonoptimal points according to Lemma 2. b) follows since $f(\mathbf{x}_{k+1}) = f(\mathbf{x}_k)$ when the stopping criterion is fulfilled.

iii) As already noted the algorithmic map $\mathbf{A}$ is composed of two maps, the direction finding map $\mathbf{D}(\mathbf{x}, \overline{\mathbf{d}})$ and the line search map $\mathbf{S}(\mathbf{x}, \mathbf{d}^{CFW})$, which are closed mappings. Since X is compact the composite map $\mathbf{A}$ is closed (see e.g. (Zangwill, 1969, Corollary 4.2.1)).

Thus the three conditions in the *Convergence Theorem* are fulfilled. $\diamond$

Corollary 1 *If the CFW algorithm is applied with inexact line search and a tolerance $\varepsilon > 0$, then it will terminate at a point $\mathbf{x}_k$ with relative gap between the upper and lower bounds less then ε, unless $f^* = 0$.*

Proof: First note that if the algorithm stops at $\mathbf{x}_k$ then the relative gap at $\mathbf{x}_k$ is less then ε. Thus assume that the algorithm does not terminate, i.e., the convergence test is never fulfilled. Then we would get the same sequence if we ran the algorithm with $\varepsilon = 0$.

Let $\{\mathbf{x}_k\}_{k=1}^{\infty}$ be the sequence of iterates. By compactness of X, $\{\mathbf{x}_k\}_{k=1}^{\infty}$ has a convergent subsequence, converging to some $\overline{\mathbf{x}} \in X$, which must be optimal according to Theorem 2. Therefore $f(\overline{\mathbf{x}}) = LBD(\overline{\mathbf{x}}) \triangleq min_{\mathbf{x} \in X}\{f(\overline{\mathbf{x}}) + \nabla f(\overline{\mathbf{x}})^T(\mathbf{x} - \overline{\mathbf{x}})\}$, implying that $\overline{\mathbf{x}}$ fulfills the tolerance test with $\varepsilon = 0$. But the gap $G(x) \triangleq f(\mathbf{x}) - LBD(\mathbf{x})$ is a continuous function of $\mathbf{x}$, and thus it tends to 0, whence the relative gap $G(x)/|f(x)|$ also tends to 0 unless $f^* = 0$, contradicting that the convergence test is never fulfilled. $\diamond$

Due to the superior convergence properties of BFW (see section 3), it would be interesting to prove global convergence of that method. But, since already the convergence proof of CFW is quite involved, and since BFW itself is at least one degree more complicated, we refrain from that. Please note however, that BFW can simply be made globally convergent, using the "spacer step technique", e.g. (Luenberger, 1984, Section 7.10), by introducing e.g. a pure FW step at iterations $k \dim(X), k = 1, 2, ..$, where $\dim(X)$ is the dimension of the feasible set X.

3 Applications to Traffic Assignment Problems

Of the dominating application areas for the FW method, we have chosen to test CFW and BFW on traffic assignment problems. In these, travellers between different origin-destination pairs in a congested urban transportation network, want to travel along their (in time) shortest routes (what is called the Wardrop user equilibrium condition). However, the travel times depend on the congestion levels, which depend on the route choices. The problem is to find the equilibrium traffic flows, where each traveller indeed travels along his shortest route. It is classical that (several versions of) this equilibrium problem can be stated as an optimization problem of the form (P), (see e.g. (Patriksson, 1994, Ch. 2)). We outline this formulation in the next subsection.

3.1 The Fixed Demand Traffic Assignment Problem

Consider a connected traffic *network* defined by sets of *nodes* $\mathcal{N}$ and directed *links* $\mathcal{A}$. Assume that the travel *demand* d_{pq} from each *origin* p to any *destination* q in the network is fixed and that the *traveltime* $t_a(x_a)$ on link $a \in \mathcal{A}$, depends only on the *flow* x_a in that link.

Let $\mathcal{R}_{pq}$ denote the non-empty set of *routes* (or *paths*) for the origin-destination (OD) pair (p,q) and $\mathcal{R} = \cup \mathcal{R}_{pq}$ the set of all routes. Let h_r denote the flow on route $r \in \mathcal{R}$. Let $\mathcal{C}$ be the set of OD pairs $(p,q) \in \mathcal{N}^2$.

The problem to find an assignment on the network which satisfies Wardrop's user equilibrium condition is called the Traffic Assignment Problem.
The Wardrop principle (Wardrop, 1952) can be stated as:

Definition 2 *(Wardrop's first principle) The route flow vector $h = (h_r)_{r\in\mathcal{R}}$ is an equilibrium if for any OD pair, the costs (i.e. times) of routes actually used (having $h_r > 0$) are equal and not larger than those of unused routes.*

Wardrop's conditions are equivalent to the first-order optimality conditions (e.g. (Patriksson, 1994, Ch. 2)) for the convex program:

$$(TAP)\quad \begin{cases} \min\limits_{h_r, x_a} & T(f) \triangleq \sum\limits_{a\in\mathcal{A}} \int_0^{x_a} t_a(s)\,ds \\ \text{subject to} & \sum\limits_{r\in\mathcal{R}_{pq}} h_r = d_{pq}, \quad (p,q)\in\mathcal{C}, & \text{(10a)} \\ & h_r \geq 0,\ r\in\mathcal{R}_{pq}, \quad (p,q)\in\mathcal{C}, & \text{(10b)} \\ & x_a = \sum\limits_{r\in\mathcal{R}} \delta_{ra} h_r, \quad a\in\mathcal{A}, & \text{(10c)} \end{cases}$$

where $\delta_{ra} = 1$ if route $r \in \mathcal{R}$ contains link a, and 0 otherwise. Conditions (10a) and (10b), which define the feasible region of (TAP), express that the route flows must be non-negative and fulfill the demands for every OD pair. Equation (10c) expresses the link flows x_a as functions of the route flows. We assume that the travel time t_a on link a, is a

Table 1: Test networks.

Network	origins	nodes	arcs	commodities
Sioux Falls	24	24	76	528
Barcelona	110	1020	2522	7922
Winnipeg	147	1052	2836	4344
Chicago Sketch	387	933	2950	93512
Chicago Regional	1790	12982	39018	2297945

continuously differentiable, positive and increasing function of the link flow x_a.

The problem (TAP) has a convex, separable, twice differentiable objective. Thus we can use the methods discussed in this paper to solve (TAP). To this end let $\mathbf{x} = (x_a)_{a\in\mathcal{A}}$ be the vector of link flows, and $X = \{\mathbf{x} = (x_a)_{a\in\mathcal{A}} |\ \exists \mathbf{h} = (h_r)_{r\in\mathcal{R}} \geq 0 : x_a = \sum_{r\in\mathcal{R}} \delta_{ra} h_r, \sum_{r\in\mathcal{R}_{pq}} h_r = d_{pq}\}$, the feasible set in the space of link flows. The linear subproblem we have to solve in Step 2. of the algorithm then turns out to be essentially the problem of finding the shortest paths from each origin to all destinations in the network, (see e.g. (Patriksson, 1994, p. 97)). To determine $\mathbf{y}_k^{FW}$ all demand is sent from origins to destinations along the shortest paths, what is termed an all-or nothing (AON) solution.

CFW and BFW share this framework. Once $\mathbf{y}_k^{FW}$ is computed, relevant vectors are computed through simple vector algebra, as described in subsections 2.1 and 2.3. This implies in particular that all computed vectors in the space of link flows will be convex combinations of previously generated $\mathbf{y}_k^{FW}$. As a consequence, a so called select link analysis can be performed in essentially the same manner as for plain FW.

3.2 Computational Experiments in a MATLAB Environment

The four methods discussed above (FW, PFW, CFW, BFW) have been implemented in MATLAB (The MathWorks, 2010). We tested them on the Sioux Falls and Winnipeg test networks (Table 1), frequently used in transportation planning (Bar-Gera, 2002). The computations were run on a 750 MHz Sun Blade 1000 Station with 8MB cache and 3GB Memory. We have previously presented preliminary MATLAB runs for CFW in (Daneva & Lindberg, 2003a).

In the experiments we start in an AON assignment using free flow costs. The main computational burden of the iterations, is the calculation of the shortest paths for the AON solution to the linear subproblems. The shortest path routine, LQUE from (Gallo & Pallottino, 1988), used in our MATLAB implementation, is implemented in C, using a MEX interface for communication between MATLAB and C. In the line search, we take a single Newton step i.e.:

$$\tau_k = -\frac{\nabla T(\mathbf{f}_k)\mathbf{d}_k}{(\mathbf{d}_k)^T H_k \mathbf{d}_k}.$$

Note that H_k is diagonal matrix, so that the computation of τ_k is not very time consuming. We could safeguard the steps by using Condition 1, or similar. However, it appears that simply taking a Newton step in practice performs better than taking a safeguarded step.

In figure 3 we display, in "log-log" diagram, the relative errors of all four methods against the iteration count for

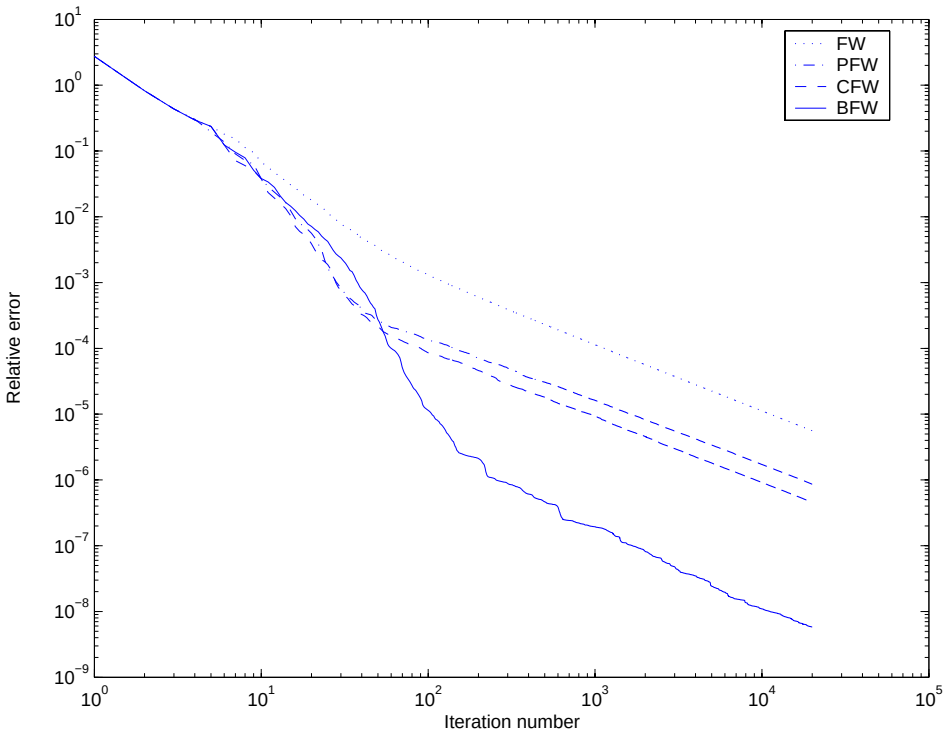

Figure 3: Comparison between FW, PFW, CFW, BFW for the Sioux Falls network.

the Sioux Falls case. Winnipeg shows a very similar behavior.

The relative errors (RE_k at iteration k) are computed as

$$RE_k = \frac{UBD_k - LBD}{LBD}, \tag{11}$$

where UBD_k and LBD are the current upper and a known lower bound, respectively. As LBD we use high (in Sioux Falls full) precision optimum values obtained by other methods. The iterations are continued much longer than what is typically done in practice, in order to show the asymptotic behavior, which is linear with slope 1 in the log-log diagram, corresponding to a relative error, RE_k, of the form

$$RE_k = const./k. \tag{12}$$

The seemingly small difference between PFW and CFW is illusive. Displaying the ratio of the relative errors of PFW and CFW in the Sioux Falls and Winnipeg cases, (figure 4), we see that the ratio stabilizes rather soon around 1.8 and 1.5, respectively. In view of (12), this implies that PFW asymptotically needs to take around 80% and 50% more iterations, respectively, to achieve the same accuracy as CFW. When comparing CFW with FW, the needed number of iterations of FW is around ten times that of CFW.

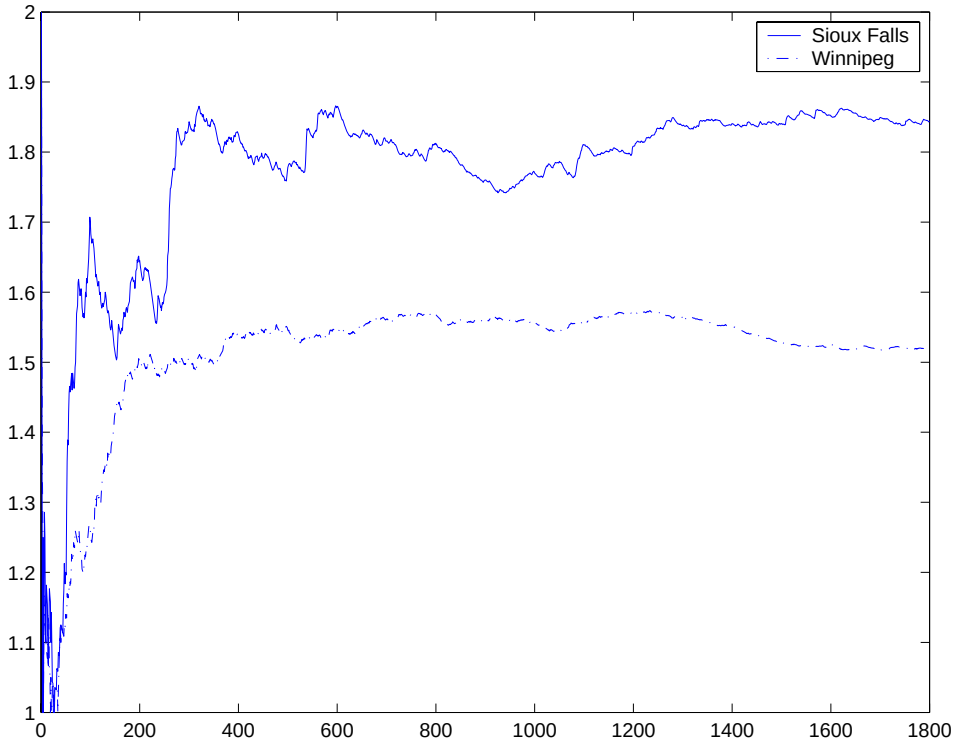

Figure 4: The ratio between the relative error for PFW and CFW.

From figure 3 it can be seen that CFW starts outperforming PFW around iterations 30-50, whereas BFW starts outperforming the other methods around iteration 100, reaching its full advantage around iteration 200-500. Similar numbers are valid in the Winnipeg case. These iterations counts may seem high, but are definitely feasible on present day powerful computers.

3.3 A C-based Comparison with OBA and DSD

The favorable performance of BFW makes it interesting to try to estimate how it compares with the recent powerful Origin-Based algorithms (OBA) described in (Bar-Gera, 2002). It is clear that OBA will be better asymptotically, since it seems to have first order convergence, as compared to the $1/n$ asymptotic relative error of CFW and BFW. However, in practice one will probably not run equilibrium computations to relative errors of e.g. 10^{-12}.

In (Boyce et al., 2002) Boyce et al. indicate, that a *relative gap* of 10^{-4} is an appropriate convergence criterion, e.g., when evaluating improvements to a given network. Since both we and Bar-Gera have studied the Winnipeg case, we made some tentative comparisons based on that case.

Since we have used the slower MATLAB environment, the comparison could not be based on CPU-times. It is well known, though, that for the Frank-Wolfe method, the main computational burden lies in the shortest path computations. The determination of the search direction is indeed somewhat heavier in our CFW or BFW than in

Table 2: CPU time needed to reach relative gap of 10^{-4}.

	BFW	CFW	FW	Bar-Gera's FW (scaled)	OBA (scaled)	DSD
Sioux Falls	0.05 s.	0.15 s.	0.76 s.	4.36 s.	0.07 s.	0.47s
Barcelona	2.66 s.	2.18 s.	3.77 s.	8.68 s.	6.7 s.	20.8s
Winnipeg	11.94 s.	17.40 s.	60.45 s.	46.71 s.	12.79 s.	7.18m
Chicago Sketch	3.69 s.	3.00 s.	4.18 s.	19.17 m.	0.52 m.	3.64m
Chicago Regional	14.37 m.	17.70 m.	42.12 m.	15.27 h.	1.05 h.	5.42h

Table 3: Iterations needed to reach relative gap of 10^{-4}.

	BFW	CFW	FW	DSD
Sioux Falls	124	357	1869	9
Barcelona	41	34	51	8
Winnipeg	163	243	838	7
Chicago Sketch	21	17	24	3
Chicago Regional	43	53	126	3

plain FW. However, timings in MATLAB of different parts of the code indicate that the extra overhead for CFW or BFW in comparison to FW is negligible (below 1%). Thus we may assume that running times for CFW and BFW are essentially the same as those for FW for a given number of iterations. Hence, we can use the FW as a yardstick. A comparison based on that yardstick indicated that BFW would be competitive with OBA for gaps of the order 10^{-4}.

These promising results spurred a more detailed investigation. Therefore we programmed our methods as well as FW in C, to allow for a more direct comparison with OBA. In figures 6 and 7 we display the relative errors of FW, CFW and BFW methods (in log-log diagrams) against the iteration count, for the Chicago Sketch and Chicago Regional networks. The Barcelona case behaves very similar to Siuox Falls. The LBD used in these cases is the LBD obtained after 5000 iterations with BFW. It should further be noted that detailed timings indicate that the overhead for CFW and BFW over FW is indeed negligible, for the larger networks. We have for instance, a timing of 0.334 min./iteration for all three methods on the Chicago Regional network.

Bar-Gera's results were obtained on a 333 MHz, 576MB, Sun Ultra 10 Station as compared to our 750 MHz Sun Blade 1000 Station. Since these workstations differ, not only in frequency but also in processor, we will use the SPEC CPU2000 benchmark results (SPEC, 2000) to compare them. There are two parts of these benchmarks, CINT2000 and CFP2000, which measure the compute intensive integer and floating point performance, respectively. Since the shortest path subproblems are integer intensive whereas the line search part is floating point intensive, we choose to take the average as our speed measure. That gives a relative speed of 129.5 for the Sun Ultra 10 as compared to 371.5 for Sun Blade 1000, giving a scale factor of approximately 0.349. Still, there are many other factors that may influence the timings, such as compiler settings, cache size, and concurrent workload. We do, however believe that this scale factor gives a comparison as fair as possible. The detailed comparison below indicate that the scaling is of the right order. In Table 2 we give our own timings, to reach a relative gap of 10^{-4}, as well as Bar-Gera's timings scaled to our

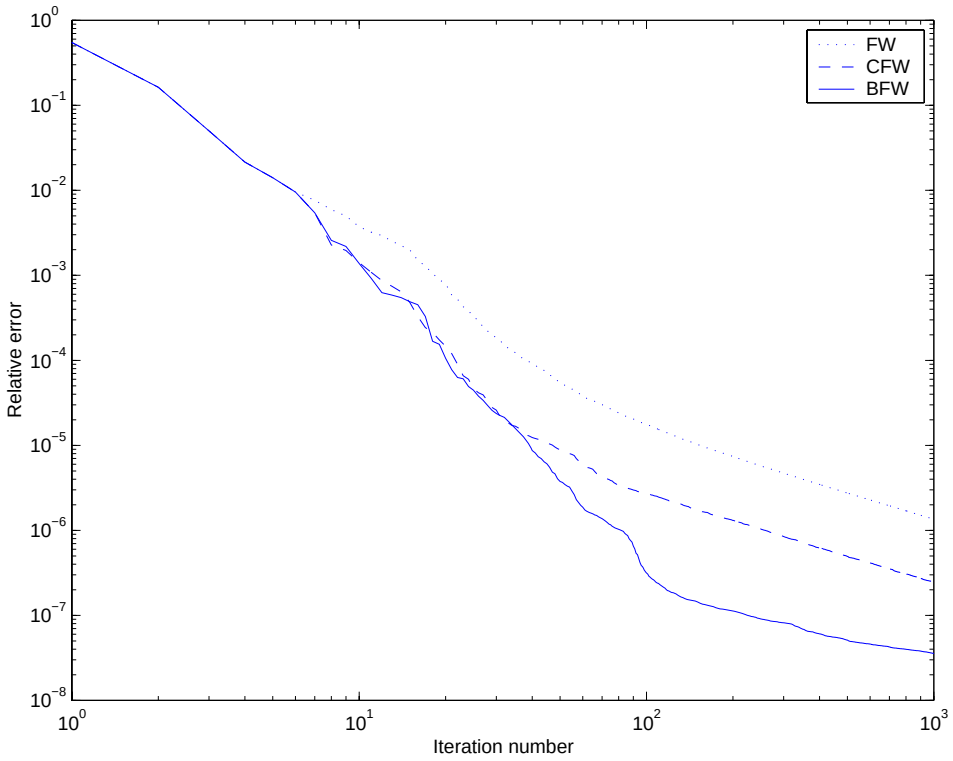

Figure 5: Comparison between FW, CFW, BFW for the Barcelona network.

computer.

(Remark: There are further two versions of the benchmark, the *base metric* which uses default settings of compilers and the *peak metric* which uses "optimized" settings. For Sun Ultra 10, only the base metric is given, whence we have based our computation on that. Should we, somewhat unfair, have compared the base metric for Sun Ultra 10 with the peak metric for Sun Blade 1000, the factor would have been 0.317, a scale factor that would not change the conclusions below.)

As said in the footnote in Section 1, the editor wanted us to perform direct computational comparisons with OBA, which we did not manage to do. As a substitute, we have performed direct comparison with the well-known DSD-method (Larsson & Patriksson, 1992), of which we had an in-house C-version. This code was developed in C by Dr C Rydergren at Linköping University from the original Fortran version of Larsson and Patriksson.

The results in Tables 2 and 3 are worth commenting. Let us first concentrate on the relation between our BFW, CFW and FW, Table 3. We see that the classical test problems, Sioux Falls and Winnipeg, are difficult for all three methods, whereas the more recent Barcelona, Chicago Sketch and Chicago Regional are easy. We further see that on the difficult problems BFW clearly outbeats CFW, whereas CFW might win with a small margin on the easy problems. The reason probably is that BFW is designed to be "correct" only in the limit, see appendix A. The recommendation is clear though, BFW, since it wins by a large margin on the difficult problems, but looses only with a small margin,

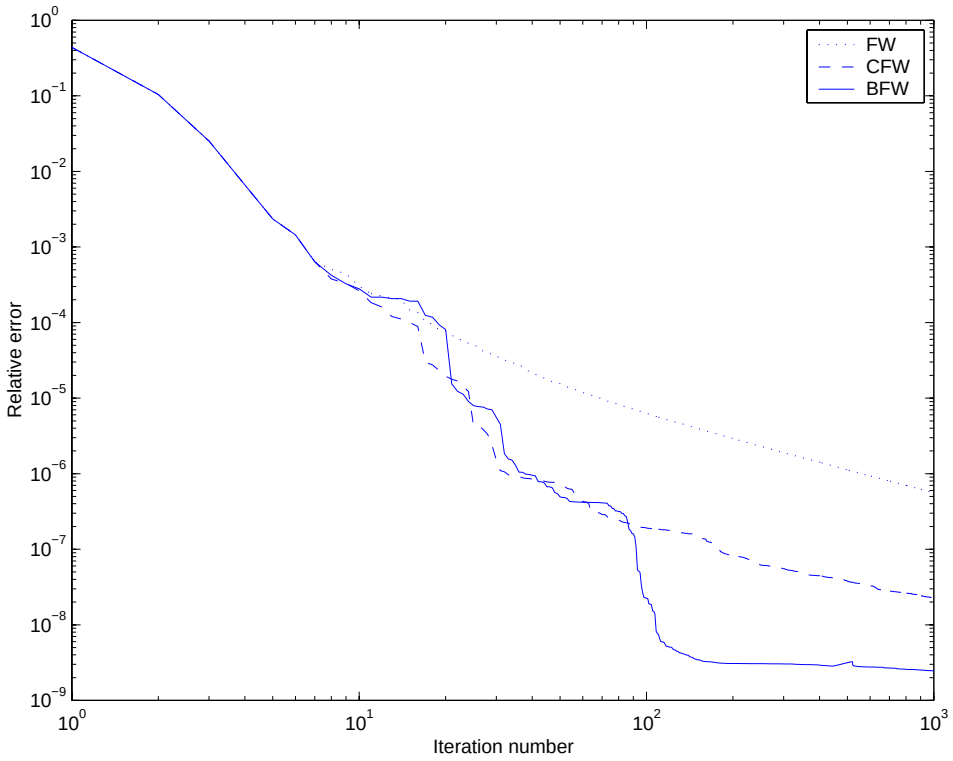

Figure 6: Comparison between FW, CFW, BFW for the Chicago Sketch network.

if at all, on the simple ones.

Concerning DSD, it obviously needs very few iterations, but it is also clear these may be forbiddingly heavy for the larger problems. In total DSD is no serious competitor for CFW or BFW.

Concerning the comparison with Bar-Gera's results see Table 2. We see that except for the Chicago problems his FW-timings are in line with ours and those for OBA with those for BFW. On the Chicago problems however his times are markedly higher. This may be due to his smaller cache memory (2MB, we believe, compared to our 8MB) or to that the Chicago problems have more OD pairs. In total one can say that BFW seems to be competitive with OBA for relative gaps of the order 10^{-4}.

These findings are confirmed by (Caliper, 2010) and (Zhou et al., 2010) who have programmed own versions of our methods and have BFW as a winner for relative gaps down to 10^{-4}. For smaller gaps other methods win; OUE for Caliper and GP for Citilabs. However, when one uses several cores, BFW comes out as winner again, due to that it parallelizes well, in contrast to its competitors.

It is informative to compare our relative timings of our methods vs OBA, with the corresponding of (Zhou et al., 2010) for the Chicago Regional network. In Table 4 we have normalized the times of OBA to 1 for our runs and for those of (Zhou et al., 2010). It is interesting to see that the relative times of BFW and CFW in particular are well in line with those of (Zhou et al., 2010), in spite of that the codes as well as the computers are different (Sun vs PC).

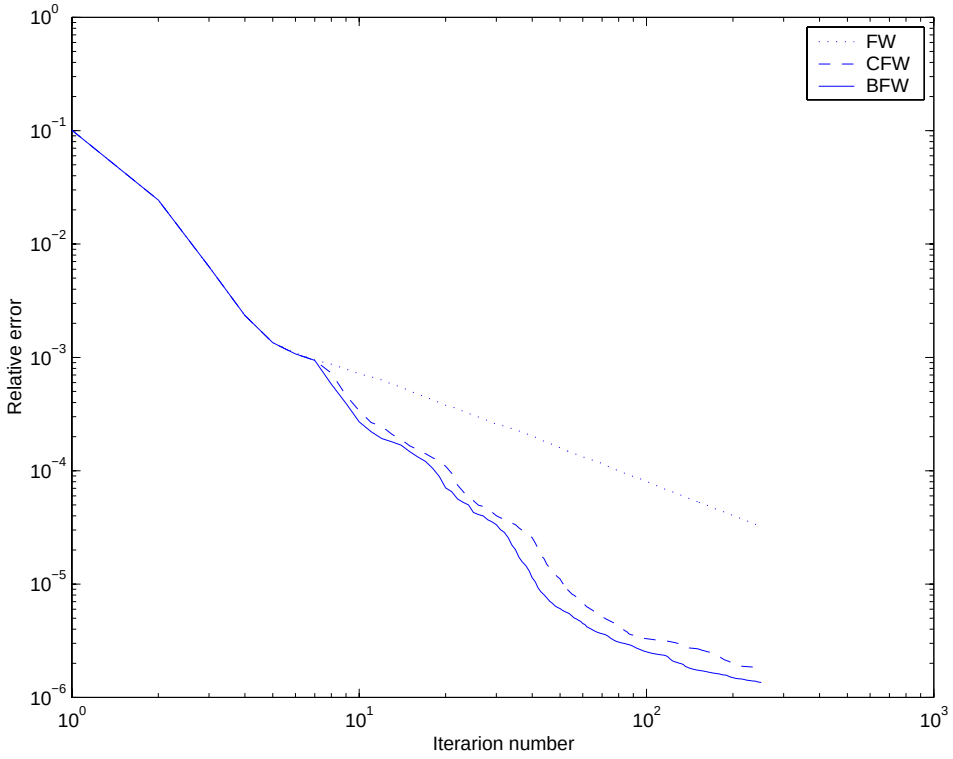

Figure 7: Comparison between FW, CFW, BFW for the Chicago Regional network.

Table 4: Relative timings on Chicago Regional.

Runs	BFW	CFW	FW	OBA
Ours	.228	.281	.669	1
Zhou et al	.240	.340	.972	1

3.4 On the Efficiency of onjugate FW-Methods

After our computational studies, we now have material to discuss the efficiency of conjugate FW-methods. As noted in section 2.2, under the assumption of quadratic costs and exact linesearch, the direction given by $\mathbf{d}_k^{CFW}$ will pass through the optimum in the space spanned by $\mathbf{d}_k^{FW}$ and $\mathbf{d}_{k-1}^{CFW}$. This explains the asymptotic stepwise efficiency of conjugate FW methods.

As the algorithm progresses, and we come closer and closer to the optimum, the objective gets successively better described by a quadratic. Thus the line in the search direction will pass successively closer to the optimum in said space, spanned by $\mathbf{d}_{k-1}^{CFW}$ and $\mathbf{d}_k^{FW}$. Also, a singel Newton step will become sucessively more exact. Our computer runs show that CFW (and BFW below) in fact take steps of size $\tau_k < 1$ in the majority of the iterations, and hence take close to optimal steps in said spaces.

The figures in section 3.2 bear witness of the efficiency of the steps of CFW. For instance in figure 3, FW and CFW seem to be equally efficient when the algorithms both have reached a linear development in the log-log-diagrams. But this is an optical illusion. We see in the figure that FW goes from a relative gap of 10^{-4} to 10^{-5} in the iterations 1000 to 10 000 (approximately) i.e. in 9 000 iterations, whereas CFW makes the same progress in iterations 100 to 1000, i.e. in 900 iterations, an improvement by a factor of 10. So asymptotically CFW is a factor 10 more efficient than FW, simply due to better search directions. Similarly, extending the linear trend of BFW in the diagram, it would take 7 iterations (from iteration 6 til 13) to get from 10^{-4} to 10^{-5}, an improvement by a factor more than 1000 over FW. Studying the (not dispayed) Winnipeg case in the same way we would get similar results.

As noted above Sioux Falls and Winnipeg are the most difficult problems for FW, and with the most marked difference between CFW and BFW. A similar rough analysis for Chicago regional would show CFW to be 100 times more efficient than FW per iteration asymptotically, whereas BFW is only marginally more efficient than CFW (by 30

So how about when we take a step of size $\tau = 1$? One way to explain the slow convergence of FW is that the current iterate $\mathbf{x}_k$ is a convex combination of all generated extreme points. And the weights of extreme points not belonging the optimal facet die out only slowly. If on the other hand, we take a step $\tau = 1$ in CFW (or BFW), the weights of extreme points with positive weight in $\mathbf{x}_k$, but zero weight in $\mathbf{s}_k^{CFW}$, get zeroed out. This will improve the possibilities to approximate the optimal facet. These two mechanism together explain the superiority of CFW (and BFW) over plain FW.

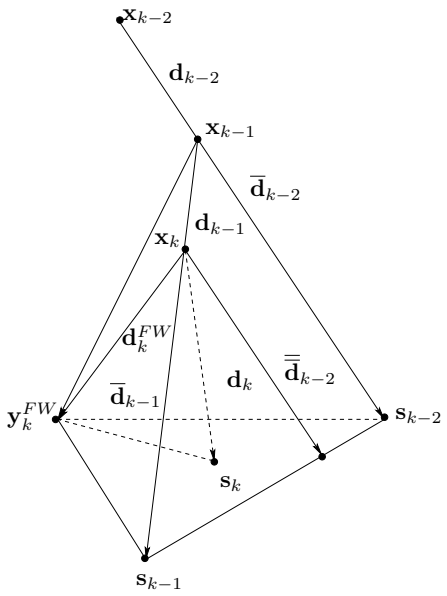

Figure 8: Determination of the coefficients in the BFW algorithm

APPENDIX

A Derivation of the Coefficients β_k^i in BFW

To increase readability we will drop the superscript BFW throughout this appendix. All directions and points, unless otherwise specified, pertain to the BFW method. In the deduction below, we assume that $\mathbf{d}_{k-1}$ and $\mathbf{d}_{k-2}$ are conjugate with respect to H_k, i.e., $(\mathbf{d}_{k-1})^T H_k \mathbf{d}_{k-2} = 0$. This is true only approximately, since $\mathbf{d}_{k-1}$ is made conjugate to $\mathbf{d}_{k-2}$ with respect to H_{k-1}. But since f is twice continuously differentiable, the error tends to zero as we converge. At the price of more cumbersome formulae we could have avoided this assumption. Under the assumption, the computation of $\mathbf{d}_k$ may be seen as first "conjugating" $\mathbf{d}_k^{FW}$ with respect to $\mathbf{d}_{k-1}$ and then conjugating the result with respect to $\mathbf{d}_{k-2}$.

Using equations (8), (9a) and (9b) we can determine all coefficients. Note that $\mathbf{d}_{k-1}$ and $\mathbf{d}_{k-2}$ need not be stored. They can be expressed in terms of $\mathbf{x}_k$, $\mathbf{y}_k^{FW}$, $\mathbf{s}_{k-2}$ and $\mathbf{s}_{k-1}$ by introducing (see figure 8):

$$\begin{aligned}
\overline{\mathbf{d}}_{k-1} &\triangleq (1-\tau_{k-1})\mathbf{d}_{k-1} = \mathbf{s}_{k-1} - \mathbf{x}_k \\
\overline{\overline{\mathbf{d}}}_{k-2} &\triangleq (1-\tau_{k-1})\overline{\mathbf{d}}_{k-2} = \\
&\quad (1-\tau_{k-1})(1-\tau_{k-2})\mathbf{d}_{k-2} = \\
&\quad \tau_{k-1}\mathbf{s}_{k-1} - \mathbf{x}_k + (1-\tau_{k-1})\mathbf{s}_{k-2}
\end{aligned}$$

where τ_k is the step length in the line search at iteration k. Hence, we might as well conjugate $\mathbf{d}_k$ with respect to

$\overline{\mathbf{d}}_{k-1}$ and $\overline{\overline{\mathbf{d}}}_{k-2}$ given that $\tau_{k-1}, \tau_{k-2} \neq 1$.

Note that

$\mathbf{s}_{k-2} - \mathbf{x}_k = (\mathbf{s}_{k-2} - \mathbf{s}_{k-1}) + (\mathbf{s}_{k-1} - \mathbf{x}_k) = \mathbf{s}_{k-2} - \mathbf{s}_{k-1} + \overline{\mathbf{d}}_{k-1}$.

Substituting this expression in (8) we get:

$$\mathbf{d}_k = \beta_k^0 \mathbf{d}_k^{FW} + (\beta_k^1 + \beta_k^2)\overline{\mathbf{d}}_{k-1} + \beta_k^2(\mathbf{s}_{k-2} - \mathbf{s}_{k-1}) \tag{13}$$

By this and (9b) we have

$0 = \overline{\overline{\mathbf{d}}}_{k-2}^T H_k \mathbf{d}_k = \overline{\overline{\mathbf{d}}}_{k-2}^T H_k[\beta_k^0 \mathbf{d}_k^{FW} + (\beta_k^1 + \beta_k^2)\overline{\mathbf{d}}_{k-1} + \beta_k^2(\mathbf{s}_{k-2} - \mathbf{s}_{k-1})]$,

which by the assumption $\overline{\mathbf{d}}_{k-1}^T H_k \overline{\overline{\mathbf{d}}}_{k-2} = 0$ gives:

$\beta_k^0 \overline{\overline{\mathbf{d}}}_{k-2}^T H_k \mathbf{d}_k^{FW} + \beta_k^2 \overline{\overline{\mathbf{d}}}_{k-2} H_k(\mathbf{s}_{k-2} - \mathbf{s}_{k-1}) = 0$.

Thus $\beta_k^2 = \mu_k \beta_k^0$, with $\mu_k = -\dfrac{\overline{\overline{\mathbf{d}}}_{k-2}^T H_k \mathbf{d}_k^{FW}}{\overline{\overline{\mathbf{d}}}_{k-2}^T H_k(\mathbf{s}_{k-2} - \mathbf{s}_{k-1})}$.

Moreover, $\mathbf{s}_{k-2} - \mathbf{s}_{k-1} = \overline{\mathbf{d}}_{k-2} - \mathbf{d}_{k-1} = \dfrac{\overline{\overline{\mathbf{d}}}_{k-2} - \overline{\mathbf{d}}_{k-1}}{1 - \tau_{k-1}}$.

By (13) and the requirement that $\overline{\mathbf{d}}_{k-1}$ is conjugate to $\mathbf{d}_k$, i.e., (9a), we have:

$\overline{\mathbf{d}}_{k-1}^T H_k[\beta_k^0 \mathbf{d}_k^{FW} + (\beta_k^1 + \beta_k^2)\overline{\mathbf{d}}_{k-1} + \dfrac{\beta_k^2(\overline{\overline{\mathbf{d}}}_{k-2} - \overline{\mathbf{d}}_{k-1})}{1 - \tau_{k-1}}] = 0$,

which leads to

$\beta_k^0 \overline{\mathbf{d}}_{k-1}^T H_k \mathbf{d}_k^{FW} + (\beta_k^1 - \dfrac{\tau_{k-1}\beta_k^2}{1 - \tau_{k-1}})\overline{\mathbf{d}}_{k-1}^T H_k \overline{\mathbf{d}}_{k-1} = 0$.

Using that $\beta_k^2 = \mu_k \beta_k^0$, we obtain $\beta_k^1 = \nu_k \beta_k^0$, where

$$\nu_k = -\frac{\overline{\mathbf{d}}_{k-1}^T H_k \mathbf{d}_k^{FW}}{\overline{\mathbf{d}}_{k-1}^T H_k \overline{\mathbf{d}}_{k-1}} + \frac{\mu_k \tau_{k-1}}{1 - \tau_{k-1}}$$

We thus get the following expressions for the coefficients in (8):

$$\beta_k^0 = \frac{1}{1 + \mu_k + \nu_k}, \quad \beta_k^1 = \nu_k \beta_k^0, \quad \beta_k^2 = \mu_k \beta_k^0,$$

The case of $\tau_{k-1} = 1$ and/or $\tau_{k-2} = 1$ is treated as in the CFW method, namely, taking the new direction $\mathbf{d}_k$ as $\mathbf{d}_k^{FW}$.

Like PFW, BFW needs to keep 4 n-vectors in memory, but it still needs only one line search per iteration.

B Closedness of the conjugation map $\mathbf{D}^{CFW}$

Lemma 4 *The conjugation map* $\mathbf{D}^{CFW}$ *is closed.*

Proof: Choose a convergent sequence $\{(\mathbf{x}_k, \mathbf{d}_k^{FW}, \overline{\mathbf{d}}_{k-1})\}_{k=1}^{\infty}$ that tends to $(\mathbf{x}, \mathbf{d}^{FW}, \overline{\mathbf{d}})$. For each k choose $\alpha_k \in \overline{\mathbf{A}}(D_k, N_k)$. Set

$$\mathbf{d}_k^{CFW} = \alpha_k \overline{\mathbf{d}}_{k-1} + (1 - \alpha_k)\mathbf{d}_k^{FW} \tag{14}$$

and assume that $\mathbf{d}_k^{CFW} \to \mathbf{d}^{CFW}$. To prove the closedness of the conjugation map $\mathbf{D}^{CFW}$, we must show that $\mathbf{d}^{CFW} \in \mathbf{D}^{CFW}((\mathbf{x}, \mathbf{d}^{FW}, \overline{\mathbf{d}})$.

Since $\alpha_k \in [0, 1 - \delta]$, we may assume that $\alpha_k \to \alpha$, going over to a subsequence if it is necessary. Note that N_k and D_k are continuous functions of $\mathbf{x}_k$, $\mathbf{d}_k^{FW}$ and $\overline{\mathbf{d}}_{k-1}$. Thus $\{N_k\}_{k=1}^{\infty} \to N$ and $\{D_k\}_{k=1}^{\infty} \to D$, where N and D are the values of these functions at $(\mathbf{x}, \mathbf{d}^{FW}, \overline{\mathbf{d}})$. By the closedness of $\overline{\mathbf{A}}$, we have $\overline{\alpha} \in \overline{\mathbf{A}}(D, N)$. Further taking the limit in (14), we get $\mathbf{d}^{CFW} = \alpha\overline{\mathbf{d}} + (1 - \alpha)\mathbf{d}^{FW}$, implying that $\mathbf{d}^{CFW} \in \mathbf{D}^{CFW}(\mathbf{x}, \mathbf{d}^{FW}, \overline{\mathbf{d}})$ as wanted. $\diamond$

Reference

Y. Arezki, "Comparison of some algorithms for equilibrium traffic assignment with fixed demands," *Proceedings 14th PTRC Summer Annual Meeting* (1986).

Y. Arezki, "Comparison of some algorithms for equilibrium traffic assignment," *3rd mini European Conf. in Transportation Problems, Herzeg Novi* (1987).

Y. Arezki and D. van Vliet, "A full analytical implementation of the PARTAN/Frank-Wolfe algorithm for equilibrium assignment," *Transportation Sci.* **24**(1), 58–62 (1990).

H. Bar-Gera, "Origin-based algorithms for the traffic assignment problem," *Transportation Sci.* **36**(4), 398–417 (2002).

H. Bar-Gera and D. Boyce, "Origin-based traffic assignment," in: *Transportation Planning*, Applied Optimization, Kluwer Acad. Publ., Dordrecht, 1–17, 2002.

D. Boyce, B. Ralevic-Dekic, and H. Bar-Gera, "Convergence of traffic assignments: How much is enough?," in: *16th Annual International EMME/2 Users' Group Conference* Albuquerque, NM, 2002.

M. Bruynooghe, A. Gibert, and M. Sakarovitch, "Une méthode d'affectation du trafic," in: *Proceedings of the 4th International Symposium on the Theory of Road Traffic Flow*, Bundesministerium für Verkehr, Bonn, Karlsruhe, 198–204, 1969.

Caliper Corporation, "What TransCAD Users Should Know about New Static Traffic Assignment Methods," Caliper Corporation Communication to Users (2010).

M. Daneva and P.O. Lindberg, "A Conjugate Direction Frank-Wolfe Method with Applications to the Trafic Assignment Problem ," in: *Operations Research Proceedings 2002*, Springer, 133–138, 2003.

M. Daneva and P.O. Lindberg, A Conjugate Direction Frank-Wolfe Method for Nonconvex Problems. Technical Report LiTH-MAT-R-2003-9, Dept. Mathematics, Linköping University, Linköping, Sweden, 2003.

R.B. Dial, "A path-based user-equlibrium traffic assignment algorithm that obviates path storage and enumeration," *Transportation Res. Part B* **40**, 917–936 (2006).

M. Florian, I. Constantin, and D. Florian, "A New Look at Projected Gradient Method for Equilibrium Assignment," *Transp Res Rec* **2090**, 10–16 (2009).

M. Florian, J. Guelat, 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. Frank and P. Wolfe, "An algorithm for quadratic programming," *Naval Res. Logist. Quart.* **3**, 95–110 (1956).

L. Fratta, M. Gerla, and L. Kleinrock, "The flow deviation method: An approach to store-and-forward communication network design," *Networks* **3**, 97–133 (1973).

M. Fukushima, "A modified Frank-Wolfe algorithm for solving the traffic assignment problem," *Transportation Res. Part B* **18**(2), 169–177 (1984).

G. Gallo and P. Pallottino, "Shortest path algorithms," *Annals of Operations Research* **13**, 3–79 (1988).

G. Gentile, Linear User Cost Equililbrium: a new algorithm for traffic ssignment Working Paper, Dipartimento di Ingegneria Civile, Edile e Ambientale, Sapienza Universit di Roma, Rome Italy IL, 2009. http://151.100.152.220/gentile/wp-content/uploads/2010/04/48-LinearUserCostEquilibrium-TRB2009.pdf

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

INRO, *Emme User's Guide*, Inro, Montreal, 2008.

T. Larsson and M. Patriksson, "Simplicial decomposition with disaggregated representation for the traffic assignment problem," *Transportation Sci.* **26**, 4–17 (1992).

T. Larsson, M. Patriksson, and C. Rydergren, "Applications of simplicial decomposition with nonlinear column generation to nonlinear network flows," in: P. M. Pardalos, D. W. Hearn, and W. W. Hager eds., *Network Optimization*, Springer, Berlin, 346–373, 1997.

L. J. LeBlanc, Mathematical programming algorithms for large scale network equilibrium and network design problems. Ph.D. thesis, IE/MS Dept, Northwestern University, Evanston IL, 1973.

L. J. LeBlanc, R. V. Helgason, and D. E. Boyce, "Improved efficiency of the Frank-Wolfe algorithm for convex network programs," *Transportation Sci.* **19**(4), 445–462 (1985).

D. G. Luenberger, *Linear and Nonlinear Programming*, Addison-Wesley, Reading, MA, 1984.

M. Lupi, "Convergence of the Frank-Wolfe algorithm in transportation network," *Civil Engineering Systems* **3**, 7–15 (1986).

MathWorks, *MATLAB User's Guide*, The MathWorks, Inc., Natick, MA, 1984-2010.

Y. Nie, "A class of bush-based algorithms for the traffic assignment problem ," *Transportation Res. B* **44**, 73–89 (2009).

A. Ouorou, P. Mahey, and J.-P. Vial, "A survey of algorithms for convex multicommodity flow problems," *Management Sci.* **46**(1), 126–147 (2000).

M. Patriksson, *The Traffic Assignment Problem - Models and Methods*, VSP, Utrecht, 1994.

W. Powell and Y. Sheffi, "The convergence of equilibrium algorithms with predetermined step sizes," *Transportation Sci.* **16**(1), 45–55 (1982).

B. V. Shah, R. J. Buehler, and O. Kempthorne, "Some algorithms for minimizing a function of several variables," *J. Soc. Indust. Appl. Math.* 12, 74–92 (1964).

SPEC CPU2000 V1.2 Documentation, Standard Performance Evaluation Corporation, http://www.spec.org/cpu2000/docs, 2000.

J. G. Wardrop, "Some theoretical aspects of road traffic research," *Proceedings of the Institute of Civil Engineers* **1**, Part II, 325–378 (1952).

A. Weintraub, C. Ortiz, and J. González, "Accelerating convergence of the Frank-Wolfe algorithm," *Transportation Res. Part B* **19**(2), 113–122 (1985).

Z. Zhou, A. Brignone, and M. Clarke, "Computational Study of Alternative Methods for Static Traffic Equilibrium Assignment," *Proceedings of the World Conference on Transport Research Society (WCTRS)* , Lisbon, Portugal (2010).

W. I. Zangwill, *Nonlinear programming: a unified approach*, Prentice-Hall Inc., Englewood Cliffs, N.J., 1969.