International Journal of Pure and Applied Mathematics
Volume 50 No. 2 2009, 157-164

ON METHODS FOR THE CONVEX MULTI-COMMODITY FLOW PROBLEM

P.O. Lindberg §

Department of Mathematics
Linköping University
Linköping, 58330, SWEDEN
e-mail: polin@mai.liu.se
and
Department of Transport and Economics
KTH – Royal Institute of Technology
Stockholm, SWEDEN

Abstract: The Convex Multi-Commodity Flow Problem is a central routing problem in road and telecom traffic. We show that the traditional method for these problems, the Frank-Wolfemethod, still has a lot to give, through better search directions, and other amendments. In the end it can beat recent competitors, for accuracies of practical interest.

AMS Subject Classification: 90-02
Key Words: routing, telecom, traffic, optimization, math. programming

1. Introduction

The Convex Multi-Commodity Flow Problem (CMFP) is a central routing problem in telecom and road traffic engineering. It concerns routing of traffic through networks with congestion effects. In these applications traffic has to be routed between given *Origin-* and *Destination* points (*OD-pairs*) in a network to minimize some objective, which in the telecom case is the total traffic delay. In the road traffic case the objective is a synthetic one, that makes the optimum fulfill certain equilibrium conditions (the Wardrop conditions, see [7]).

Received: August 14, 2008

§Correspondence address: Department of Mathematics, Linköping University, Linköping, 58330, SWEDEN

Network	origins	nodes	arcs	OD-pairs	variables	constr's
Sioux Falls	24	24	76	528	1824	528
Barcelona	110	1020	2522	7922	244634	7 922
Chic. Reg.	1790	12982	39018	2297945	$69.1 * 10^6$	$.6 * 10^6$

Table 1: Test networks

For space reasons we cannot give a complete survey of these areas, but direct the readers on the first hand to the survey by Ourouro et al [6], on the second hand to Patriksson [7] for the road traffic area.

The CMFP is a mathematical program, which typically is very large, with up to hundreds of thousand constraints and millions of variables, see [6].

I have worked mainly in the traffic area. Some wellknown test cases in this area, and that I will use as test cases below, are *Siuox Falls*, *Barcelona*, and *Chicago Regional* with characteristics according to Table 1. The number of variables and constraints given are for the *node-arc* formulation of the problem. As can be seen, even small networks give rize to nontrivial nonlinear programs.

2. Mathematical Statement of the CMFP

We will give the "traffic version" of CMFP, which is a *link-path* formulation (the node-arc formulation follows easily). We give a network with sets of *nodes* $\mathcal{N}$ and (directed) *links* $\mathcal{A}$. We also have sets of *origins* $\mathcal{P} \subset \mathcal{N}$ and *destinations* $\mathcal{Q} \subset \mathcal{N}$. For every OD-pair $(p,q) \in \mathcal{P} \times \mathcal{Q}$ there is a traffic demand $d^{pq} \geq 0$ (in veh./h). Let further $\mathcal{R}^{pq}$ be the set of simple (i.e. nonrepeating) routes connecting OD-pair (p,q), $\mathcal{R} = \cup \mathcal{R}^{pq}$ the set of all routes, and $\mathcal{R}_a$ the subset of routes passing link a. Let $t_a(f_a)$ be the travel time in link a, assumed nondecreasing and only depending on the total flow f_a in link a. Then the traffic version of CMFP can be stated as follows.

$$\min_{h_r \geq 0, f_a} \quad T(f) =_{df} \sum_{a \in \mathcal{A}} \int_0^{f_a} t_a(s)\, ds$$

$$\text{subject to} \quad \sum_{r \in \mathcal{R}_{pq}} h_r = d_{pq}, \quad (p,q) \in \mathcal{P} \times \mathcal{Q} \tag{1a}$$

$$f_a = \sum_{r \in \mathcal{R}_a} h_r, \quad a \in \mathcal{A}, \tag{1b}$$

In telecom applications the objective $T(f)$ is exchanged for a measure of the total traffic delay.

3. The Frank-Wolfe Algorithm and Competitors

The dominating method for the CMFP has been the Frank-Wolfe (FW) method, devised already in the 50'ies by Frank and Wolfe for linearly constrained convex optimization problems, see [7]. It was first applied to CMFP:s in 1973 by LeBlanc (traffic) and Fratta et al (telecom) (see [6]). It utilizes the structure of CMFP:s beautifully. It works as follows for a general linearly constrained convex optimization problem,

$$(P_{FW}) \quad f_{\min} =_{df} \min_{x \in C} f(x),$$

where $C \subset \Re^n$ is a convex polyhedral set and f is a smooth function on C.

The FW-Algorithm

1. **Start** at a feasible $x^{(0)} \in C$. Until convergence repeat 2-3.

2. Given the iterate $x^{(i)}$ in iteration i, linearize the objective to $\underline{f}^{(i)}(x) =_{df} f(x^{(i)}) + \nabla f(x^{(i)})(x - x^{(i)})$, and solve the *FW-subproblem*

$$(\underline{P}^{(i)}_{FW}) \quad \underline{f}^{(i)}_{\min} =_{df} \min_{x \in C} \underline{f}^{(i)}(x),$$

giving the solution $y^{(i)}_{FW}$, the *FW-point* .

3. Make a line search from $x^{(i)}$ in the *(FW-)direction* towards $y^{(i)}_{FW}$, giving the next iterate $x^{(i+1)}$ **end**.

Note that $\underline{f}^{(i)}_{min}$ is an underestimate of $f_{\min}$, since $\underline{f}^{(i)}$ is an underestimate of f. On the other hand, $f(x^{(i)})$ is an overestimate of $f_{\min}$. Hence, we can compute a *gap* $\Delta =_{df} f(x^{(i)}) - \underline{f}^{(i)}_{min}$, and a *relative gap* $\delta =_{df} \Delta / f(x^{(i)})$, that can be used for termination criteria.

The FW-algorithm utilizes the structure of CMFP beautifully: The subproblem (P_{FW}) decomposes into a set of shortest path problems, one for each origin. Sending the demands along these paths gives the FW-point. The line search can be done completely in the space of total link flows. And only these total link flows need to be stored to the next iteration.

The FW algorithm has a rather fast initial convergence, but asymptoti-

cally it is quite slow, the relative gap converging as $1/i$, where i is the iteration number. Due to the slow convergence, there have been many suggestions for improvements for the FW-algorithm. In later years one has concentrated on new (first order) methods such as DSD (Disaggregate Simplicial Decomposition, see [7]), OBA (Origin Based Algorithm, [1]) in the traffic area, and ACCPM (Analytic Centre Cutting Plane Method), and other methods in the telecom area, see [6]. Typically these new methods compare themselves with (not necessarily well implemented) versions of the FW-method.

Since these new methods typically are first order methods, they should for sufficiently high accuracies beat the FW- method. So what is an appropriate accuracy? Boyce et al [2] advocate a relative gap of 10^{-4} in the traffic area. Traffic engineers usually are satisfied with less stringent demands, such as rel. gaps of 10^{-2}.

We will outline some improvements of the FW-method that makes it competitive the above methods for the required accuracies and higher. We will compare ourselves with the leader in the traffic area, OBA [1].

4. Improvements of the FW-Algorithm

Since OBA has been run on a different computer, we cannot make direct timing comparisons. We can however use the Spec CPU2000 benchmarks [8] to scale the running times of OBA (and the FW-algoritm of [1], which we will term *BGFW*) to our computer (for details see [4]).

4.1. The Conjugate Frank-Wolfe Method, CFW

Conjugation is a classical device to derive improved search directions. The main problem in the CMFP setting is to derive feasible bounds on the step lengths in the line search in a conjugate direction. Such bounds were derived by Daneva (in her Master's thesis, see [3], [4]). She did this also for conjugation with respect to the *two* previous search directions (*Biconjugate FW, BFW*).

Comparing the (scaled) running times of OBA with our running times for BFW for relative gaps of 10^{-4}, BFW outperforms OBA by a factor of four or more, except for the smallest problem. BGFW is beaten by our FW-algorithm by factors of 10-20, indicating that probably is not the best implementation.

4.2. Speeding up the Subproblem Solution

The next natural the FW-improvement, is to speed up the subproblem solution. We have applied two approaches, *updating shortest path calculations*, and *bucketed shortest path calculations*. These are reported in Holmgren [5].

4.2.1. Updating Shortest Path Calculations (USPC)

The main computational burden of the FW-method lies in the shortest path calculations, more than 95% even for moderate problems. Thus, speeding up the shortest path calculations will have a direct impact. When the total link flows stabilize, the shortest path subproblems will experience only small cost changes. Thus it is natural to *update* the shortest path solution rather than to solve from scratch. To update the shortest path solutions, we employ classical labeling techniques from network flows. In particular we employ *predecessor* and *thread pointers* and *depth* indices. Furthermore, we introduce a new way to scan the arcs, *thread-following*. A further improvement is that USPC also allows for updating of the FW-point (rather than recomputing it). Implementing USPC gives timing reductions from 35% (Chicago) to 50-60% (Barcelona and Sioux Falls), compared to plain FW (without USPC).

4.2.2. Bucketed Shortest Path Calculations (BSPC)

In USPC we have to scan the arcs for arcs of negative reduced costs. For large networks this may be quite time-consuming. At the same time, only a subset of the arcs will be interesting in the long run. In BSPC we try to identify the set of interesting arcs, putting them in a bucket of interest, one for each origin. We update the bucket intermittently. In most of the shortest path calculations we only scan the bucket for arcs of negative reduce costs, thus gaining speed. The drawback is that we do not get an underestimate of the optimum in this way, due to that there may be arcs of negative reduced cost outside of the bucket. Therefore we scan all arcs at regular intervals. Implementing USPC with BSPC we are competitive with OBA down to relative gaps of 10^{-5}, with timing reductions (over OBA) from factors of 4 (Barcelona) to 6.5 (Chicago).

4.3. Partial Updating

In similarity to BSPC, we can choose to solve only a subset of the shortest path (SP) subproblems under an iteration (then termed *mini-iteration*). The solving of the subproblems and the concomitant line-search has two goals: to get a lower bound (LBD) of the optimum value and to improve the objective. If we only solve a subset of the SP-subproblems we do not an LBD, but we get a (hopefully) improved objective at a lower computational cost. To get an LBD, we have to solve all sub-problems every now and then (a *major* iteration).

Implementing this for pure USPC with plain (non-conjugate) FW we get timing reductions around 50% when we solve around on tenth of the SP-subproblems in each mini-iteration, and doing a major iteration every tenth iteration or so. Partly due to a computer breakdown we have only been able to compare with USPC. Comparison with BSP and conjugate FW demands more intricate changes to these codes.

4.4. Away Steps

Part of the convergence problems of the FW-algorithm lies in that it zigzags towards the optimal point, without ever reaching the optimal facet. To overcome this zigzagging already Wolfe suggested taking an "*away* step", i.e. *first* computing a new "*away* point", by using an objective with an oppositely directed gradient in the subproblem, and *then* making a line-search from the current iterate in the direction away from the away point.

This strategy has two problems in the CMFP setting. Firstly if we negate all costs in the SP-subproblems, we get NP-complete problems, which cannot be solved in reasonable time. Secondly, even if we could compute the away point, it would be hard to determine feasibility bounds on the step-lengths.

We can however use any other feasible point as away point, e.g. the initial solution $x^{(0)}$. We can then recursively compute the weights μ_i to represent the current point $x^{(i)}$ as a convex combination of the chosen away point and another feasible point $z^{(i)}$, the "*approach* point" which in turn is a convex combination of the FW-point $y_{FW}^{(i)}$ and the previous approach point $z^{(i-1)}$. The updating of the multiplier μ_i is quite simple:

Let λ_i be the step-length from $x^{(i)}$ towards $y_{FW}^{(i)}$ to get to $x^{(i+1)}$, then $\mu_{i+1} = (1 - \mu_i)\lambda_i + \lambda_i$.

Implementing this, it turns out that for the high accuracies that we aspire

(relative gaps 10^{-4} and smaller) we get some problems. Probably due to rounding errors, we arrive at points slightly outside the feasible set, and get objective values lower than the LBD:s of other approaches.

To overcome these problems one can store and update the approach point in each iteration (at extra computing costs and storage requirements, though). Implementing this new approach, (and comparing, as before, with plain FW with USPC) we get, as for Partial Updating, timing reductions on the order of 50%.

In total, the last two improvements of the FW-method (away steps and partial updating) show promise to deliver further improvements when combined with the first two improvements. It is not to be expected that the timing reductions will be on the order of 50% each, as above, but 25% seems reasonable.

5. Conclusions

In summary we have showed that there is a large potential for improving the FW-method, and indeed making it competitive with the best methods of today for accuracies advocated by some of the proponents of these new methods. The competition from these new methods has spurred the FW-method to improve, Hopefully the improved FW-method will conversely spur further improvements in these new methods and in other methods still to appear.

References

[1] H. Bar-Gera, Origin-based algorithms for the traffic assignment problem, *Transportation Sci.*, **36** (2002), 398-417.

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

[3] M. Daneva, P.O. Lindberg, A conjugate Frank-Wolfe method with applications to traffic assignment, In: *Operations Research Proceedings, 2002*, Springer, Berlin (2003), 133-138.

[4] M. Daneva, P.O. Lindberg, The stiff is moving - conjugate direction Frank-Wolfe methods with applications to traffic assignment, *Working Paper*, Department of Mathematics, Linköping University (2004).

[5] J. Holmgren, *Efficient Updating Shortest Path Calculations for Traffic Assignment*, Master's Thesis, Linköping University (2004), http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-2573

[6] A. Ouorou, P. Mahey, J.-P. Vial, A survey of algorithms for convex multi-commodity flow problems, *Management Sci.*, **46** (2000), 126-147.

[7] M. Patriksson, *The Traffic Assignment Problem - Models and Methods*, VSP, Utrecht (1994).

[8] *SPEC CPU2000 V1.2 Documentation*, Standard Performance Evaluation Corporation, http://www.spec.org/cpu2000/docs (2000).