A NEW IMPLEMENTATION OF ORIGIN-BASED ASSIGNMENT ALGORITHM IN SATURN

Yanling Xiang and Ian Wright, Atkins Limited
Dirck Van Vliet
Hillel Bar-Gera, Purdue University, and Ben-Gurion University of the Negev, Israel
David Boyce, Northwestern University

1. INTRODUCTION

The most common traffic assignment algorithm used in practice is the link-based Frank-Wolfe algorithm (Frank and Wolfe, 1957), which has been the principal assignment algorithm (with various modifications) used in the SATURN highway assignment suite since its first release in 1981 (Van Vliet et al, 1980). Whilst the Frank-Wolfe (FW) algorithm is theoretically proven to produce a converged solution (when the problem has a convex optimisation formulation) it fails to achieve accurate solutions within any practical computation timeframe.

The Origin-Based Assignment (OBA) algorithm (Bar-Gera, 2002) was developed by Prof. Hillel Bar-Gera, while a PhD student at the University of Chicago in the late 1990's. The key difference of OBA from the link-based FW algorithm is that it stores the link flows as generated by each individual origin. In a certain sense, OBA is an intermediate between link-based and path-based algorithms but without requiring excessive RAM or CPU as typically characterized by the latter. The principle of OBA is that, by considering origin-based link flows from the a-cyclic subnetwork, it provides a computationally efficient route-building process as well as enabling the elimination of residual flows (i.e. small flows on sub-optimal routes) that have a detrimental impact on algorithm convergence.

The algorithm revolutionised traffic assignment in that it provided a Wardrop Equilibrium solution whose numerical accuracy was restricted only by the numerical accuracy of the computer, i.e. it converged exactly within reasonable CPU times. OBA first became available with the release of SATURN Version 10.5 in 2005 for single user class (SUC) assignments only - this restriction precluded its practical application.

A secondary benefit of the OBA algorithm is that the secondary analysis that requires route information (e.g. select-link analysis and cost skimming) may be readily extracted from the route information stored during the assignment rather than having to be subsequently re-built from the link information stored under FW.

This paper presents the subsequent developments of OBA within SATURN: (i) the extension of the algorithm to handle multiple user class (MUC) assignments; and, more recently, (ii) a hybrid algorithm combining the speed of Frank-Wolfe algorithm with the accuracy of OBA. This development work continues and the latest results presented show its potential. The recent release of multi-threaded implementation of the existing FW algorithm (SATURN Multi-Core) provides further exciting opportunities to combine the speed of FW with the subsequent accuracy of OBA MUC.

This paper is organised as follows: Section 2 presents an overview of the OBA algorithm (applicable to both SUC and MUC implementations) whilst Section 3 describes the extension of the existing OBA algorithm for the MUC problem as well before providing a practical comparison of FW and OBA algorithms. Section 4 describes the hybrid algorithm with latest results whilst Section 5 provides a summary of the findings.

2. AN OVERVIEW OF ORIGIN-BASED ASSIGNMENT

A-CYCLIC SUBNETWORKS

Unlike the link-based FW algorithm, the routes created by the OBA path-building process connecting the origin to all other origins cannot pass through the same node more than once – the resulting subnetwork is described as '*a-cyclic*'. The work undertaken by Bar-Gera (2002) proved that there is **always** an equilibrium solution by origin for a-cyclic subnetworks and the restriction to using only a-cyclic subnetworks will not prevent the algorithm from converging to the true equilibrium solution.

APPROACH PROPORTIONS AND ROUTE FLOWS

The main solution variable in the OBA algorithm is the calculation of the origin-based approach proportions for every origin and every link in the subnetwork, such that for every origin p and node i contained therein:

- the sum of origin-based approach proportions, for each origin p over all subnetwork links ending at node i is equal to one; and

- the approach proportions for origin p of links that are not included in the subnetwork are restricted to zero.

Using origin-based approach proportions, route proportions are determined by the product of the individual approach proportions of all the subnetwork links along the route. The resulting route flows are simply determined by the product of OD flow and route proportion.

Advantages

The representation of the solution by origin-based approach proportions allows storing a complete description of the route flows very efficiently. This is a major difference from many alternative solution procedures, including the most common FW algorithm, which store only total link flows during the iterative process. The use of a-cyclic subnetworks allows a definition of a topological order of the nodes, which is an origin-specific ordering of the nodes, such that every link in the restricting subnetwork goes from a node of lower topological order to a node of higher topological order.

Most computations in the proposed algorithm are done in a single pass over the nodes, either in ascending or descending topological order. The time required by such computations is a linear function of the number of links in the network. This computational efficiency is a major advantage of the restriction to a-cyclic solutions.

TRAFFIC ASSIGNMENT

Using the A-Cyclic Subnetwork

In solving the traffic assignment problem, the algorithm starts with trees of minimum cost routes as restricting subnetworks, leading to an all-or-nothing assignment. Then, the algorithm considers all origins in a sequential order. For each origin the restricting subnetwork is updated, and the origin-based approach proportions are adjusted within the given restricting subnetwork.

To update a restricting subnetwork, unused links are removed. Once a new restricting subnetwork is found, several computationally intensive steps are needed including a reorganization of the data structure. However, the structures of the restricting subnetworks tend to stabilise fairly quickly. Therefore, it is useful to update origin-based approach proportions while keeping the restricting subnetworks fixed. This is done by introducing "inner iterations" as described in the flow chart, presented below in Figure 1.

To update origin-based approach proportions within a given restricting subnetwork a search direction based on shifting flow from high cost alternatives to low cost alternatives is used. In addition to current costs, estimates of cost derivatives are used to improve the search direction in a quasi-Newton fashion. When two independent routes are considered, the amount of flow shifted by the search direction equals the difference between route costs divided by the sum of route cost derivatives – the same result as would be obtained by considering an objective function second-order approximation.

Eliminating Residual Flows

The second-order search direction described above is viewed as desirable flow shifts. These are scaled by a step size between zero and one, and then truncated to guarantee feasibility, a process referred as the '*boundary search procedure*'. This technique is somewhat different than conventional line search techniques, where shifts are first truncated to guarantee feasibility and only then scaled by a step size. The importance of the boundary search for OBA is that it is effective in eliminating residual flows, i.e. small flows on sub-optimal routes. The elimination of residual flows is critical for algorithm convergence. See (Bar-Gera, 2002) for details.

Achieving Convergence using 'Social Pressure'

In order to guarantee descent of the objective function, and convergence of the algorithm, the search considers various step size values of $1/n$. The stopping condition is based on the concept of '*social pressure'* introduced by Kupsizewska and Van Vliet (1999). The principle of social pressure is that every traveller shifted from route r_1 to r_2 places pressure (positive or negative), which is equal to their gain (or loss) as a result from the shift, that is according to the difference in route costs. The total social pressure is the sum of the pressure from all the travellers.

The search direction is beneficial in the sense that it always enjoys positive social pressure for small step sizes but, as the step size increases, the social pressure decreases, and eventually it may become negative. The objective is to determine the largest step size with positive social pressure - this social pressure principle is similar to the stopping condition of the line-search in the Frank-Wolfe algorithm.

3.   EXTENDING OBA TO MULTI-USER CLASS PROBLEMS

OVERVIEW

The OBA algorithm was originally implemented for single user class problems but the principle restriction to SUC related to the implementation rather than any theoretical considerations. By its nature, OBA creates multiple copies of the same network via its use of a-cyclic subnetworks and provides a flexible framework that may be readily expanded.

Within OBA, the origin-based link flows are independent between origins and the revisions to handle MUC problems naturally extends that principle such that the origin-based link flows are independent by origin **and** user class. In theory, the extension to MUC problems is simply achieved by considering the MUC network as an enlarged SUC network, whereby the origins for each user class are arranged sequentially in turn as if they were consecutive origins with the total number of *virtual* origins equal to the number of user classes (UC) multiplied by the number of zones.

PROGRAMMING

In practice, however, there was a substantial volume of re-coding of the existing data structures and algorithm to handle specific user-class components such as demand, generalised costs, banned turns, and penalties as well as ensuring consistency with the existing functionality of the SATURN suite.

The MUC implementation of OBA in SATURN is a natural extension of SUC OBA based on its multi-copy property. The theoretical framework of OBA is theoretically scalable across any number of user classes in essence and all the properties of SUC OBA hold equally for MUC.

The existing OBA was coded in the C language and linked to the existing SATURN FORTRAN through object libraries. Whilst the majority of the changes were (relatively) straightforward, the principle complication occurred in the calculation of link costs for MUC networks. For the existing FW assignments implemented in SATURN, SUC and MUC link costs are calculated differently – for MUC, link costs are the summation of:

- UC independent congested flow-dependent link costs (delays); and

- UC dependent free-flow link costs and fixed penalties (including bans or tolls).
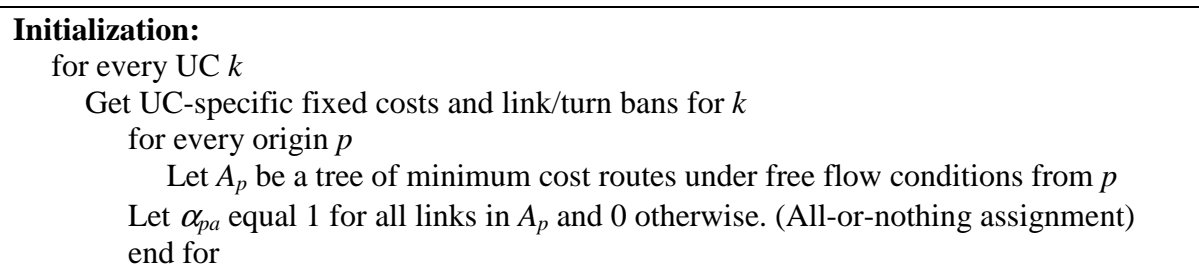
The cost function in the C programming for MUC OBA was implemented in a similar way as that in FORTRAN for MUC FW assignments. The handling of banned links in C was actually a translation of equivalent FORTRAN code. For the calculation of cost derivatives, no special treatment was needed as they are a function of congested link flows which are UC independent.

The following summarises key techniques and methods that were adopted for the development of the MUC OBA in SATURN with C programming:

- New data structures were created to store origin-based networks by UC and origin (plus various new structures and arrays for MUC OBA-specific components);

- FORTRAN COMMON blocks already in SATURN were shared wherever possible to reduce computer memory requirements;

- The MUC OBA assignment initialisation was based on the same principle as that of SUC OBA - an All-Or-Nothing (AON) shortest path calculation. The only difference was that, in addition to the UC-independent free flow costs, the UC-specific fixed costs formed the basis for MUC route initialisation; and

- MUC OBA assignments were undertaken by UC and by origin in a sequential order - whenever user classes are switched from one to another during the assignments, UC-specific fixed costs and link bans are updated accordingly and global structures refreshed for the next UC assignment.

Figure 1 below presents the flow chart of the implementation of MUC OBA algorithm in SATURN.

Figure 1 – Flow Chart showing the OBA MUC Algorithm

Initialization:
  for every UC k
    Get UC-specific fixed costs and link/turn bans for k
      for every origin p
        Let A_p be a tree of minimum cost routes under free flow conditions from p
      Let α_{pa} equal 1 for all links in A_p and 0 otherwise. (All-or-nothing assignment)
        end for

```
    end for
Main loop:
    for n=1 to number of main iterations
        for every UC k
            Get fixed costs and link/turn bans for UC k
                for every origin p
                    update restricting subnetwork $A_p$
                    update origin-based approach proportions $\alpha_{pa}$
            end for
        end for
        for m=1 to number of inner iterations
            for every UC k
                Get fixed costs and link/turn bans for UC k
                    for every origin p
                        update origin-based approach proportions $\alpha_{pa}$
                end for
            end for
        end for
    end for

Update restricting subnetwork for UC k and origin p:
    remove unused links from $A_p$
    for every node i compute the maximum cost $\nu_i$ from p to i
    for every link a=[i,j]
        if $\nu_i < \nu_j$ add link a to $A_p$
    find new topological order for new $A_p$
    update data structures

Update origin-based approach proportions for UC k and origin p:
    compute average costs and Hessian approximations
        for step size 1,1/2,1/4,1/8…
            compute flow shifts and scale by step size
            project and aggregate flow shifts
            if social pressure is positive then stop
        end for
    apply flow shifts
    update total link flows and link costs
```

PERFORMANCE OF THE OBA MUC ALGORITHM

The performance of the new OBA MUC algorithm against the existing FW-implementation in SATURN is shown below for two real-life models:

- Model 1 was a relatively small SATURN assignment model with **six** user classes, 216 zones, 3001 assignment nodes, and 4545 assignment links; whilst

- Model 2 was a much larger SATURN model with **nine** user classes, 453 zones, 7063 assignment nodes, and 10285 assignment links.

The comparisons illustrate %GAP against total CPU time in hours. %GAP is a normalised convergence indicator for Wardrop equilibrium - in SATURN it measures the degree to which the routes assigned are minimum cost routes *after* the simulation stage.

Figure 2 shows the convergence for the OBA MUC and FW algorithms for Model 1 whilst Figure 3 provides the same comparison for Model 2. For both models, whilst the FW algorithm was more efficient in early stages, it did not reach the same low level of convergence that OBA subsequently achieved within the equivalent CPU time.

Figure 2 - SATURN Convergence (Model 1)
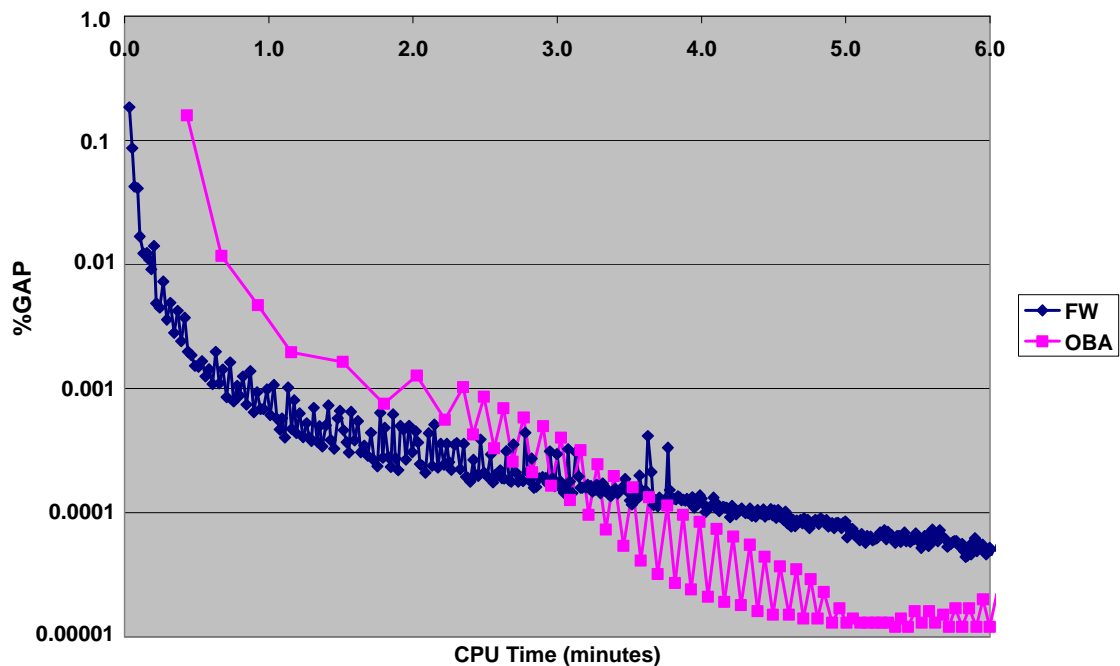
Figure 3 - SATURN Convergence (Model 2)

7
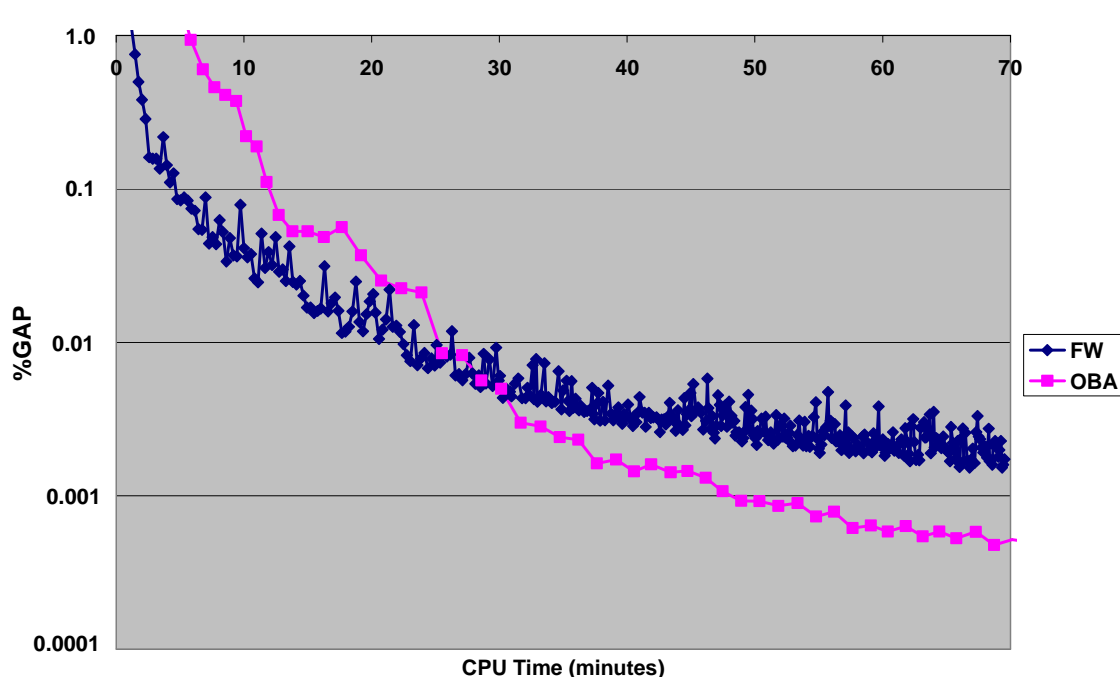
SUMMARY

The analysis shows that the OBA MUC algorithm was able to achieve higher levels of convergence than the FW algorithm within similar CPU expenditure. Whilst the existing FW algorithm was much faster in the initial stages it eventually became 'stuck' and unable to reduce the convergence error within its assigned solution.

4. HYBRID FW-OBA ALGORITHM

More recently, a new hybrid algorithm - combining the speed of Frank-Wolfe with the accuracy of SATURN-OBA – has been developed to enable very highly converged highway assignment solutions to be achieved without a significant increase in CPU overheads. The current development work is focussing on the optimal strategy for balancing the CPU time required and the level of convergence achieved.
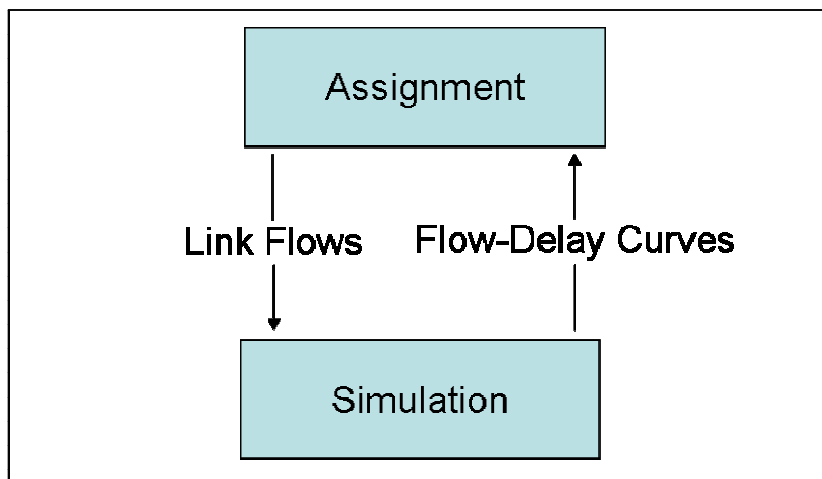
SATURN ASSIGNMENT

The hybrid algorithm initially starts off with the FW algorithm before transferring to the OBA MUC algorithm. The transfer point is determined by the level of convergence between the assignment-simulation loops in SATURN.

Assignment-Simulation Loops

The SATURN assignment consists of two modules (Figure 4): (i) a pure assignment module based on "separable" link cost-flow functions; and (ii) a simulation stage which accounts for non-separable interactions between traffic streams at junctions. The assignment provides link (and turn) flows to the simulation which, in turn, produces an updated set of separable cost-flow functions based on the junction interactions for use in the next assignment. These two modules are run in an iterative loop until: (i) both modules have converged internally; and (ii) the changes in the cost-flow functions and/or link flows are very small.

Figure 4 - SATURN Assignment and Simulation Loop

Switching Between Algorithms

A key feature of the early assignment-simulation loops is the large changes in the cost-flow functions between successive loops. These occur because of the large imbalance in the assigned flows estimated in the current assignment and those derived from the previous loop and used in the simulation. Consequently, at this early stage, it is not necessary to estimate a highly converged assignment and it is more efficient, in terms of CPU required, to use FW. However, once the convergence between the assignment and simulation has begun to stabilise, it is advantageous to have a highly convergent assignment available with OBA.

PERFORMANCE OF THE HYBRID ALGORITHM

The performance of the hybrid algorithm against both the FW and OBA MUC implementation in SATURN is shown below for three more real-life models using the same comparisons previously shown in Section 3 with (Log) %GAP plotted against total CPU times. The Hybrid algorithm was also tested using the earlier models and the performance of the algorithm is also presented. The three new models were:

- Model 3 was a SATURN model with **nine** user classes, 321 zones, 5678 assignment nodes, and 8354 assignment links;

- Model 4 was a larger SATURN model with **five** user classes, 510 zones, 15875 assignment nodes, and 22113 assignment links; and

- Model 5 is a SATURN model with a **single** user class, 846 zones, 15599 assignment nodes, and 26849 assignment links.

Different convergence targets were used depending on the overall size of the model. A convergence target of %GAP < 0.0001 for four consecutive loops was set for Model 3 whilst more relaxed targets of %GAP < 0.001 and %GAP < 0.01 (for four consecutive loops) was selected for the larger Models 4 and 5 respectively.

Figure 5 compares the profiles for the three algorithms for Model 3 with the assignment terminated when the convergence criteria was met. As previously described in Section 3, the FW algorithm was quicker in the early stages despite the oscillations in performance. The MUC OBA algorithm was much slower to achieve similar levels of convergence but less prone to oscillation as well as having a more linear descent profile. If the assignment was continued beyond the pre-defined convergence target, the algorithm would have converged to a lower %GAP value than would be achieved by FW.

The hybrid algorithm combined the strengths of the two algorithms: the initial rapid improvements in convergence from FW followed by a steady decent profile of OBA MUC. Within these tests, the switch to MUC OBA was undertaken at an early stage (i.e. %GAP < 0.1) – empirical testing showed that switching to OBA MUC at this level of convergence enabled lower %GAP values to be subsequently achieved. Further optimisation work is underway in this area.
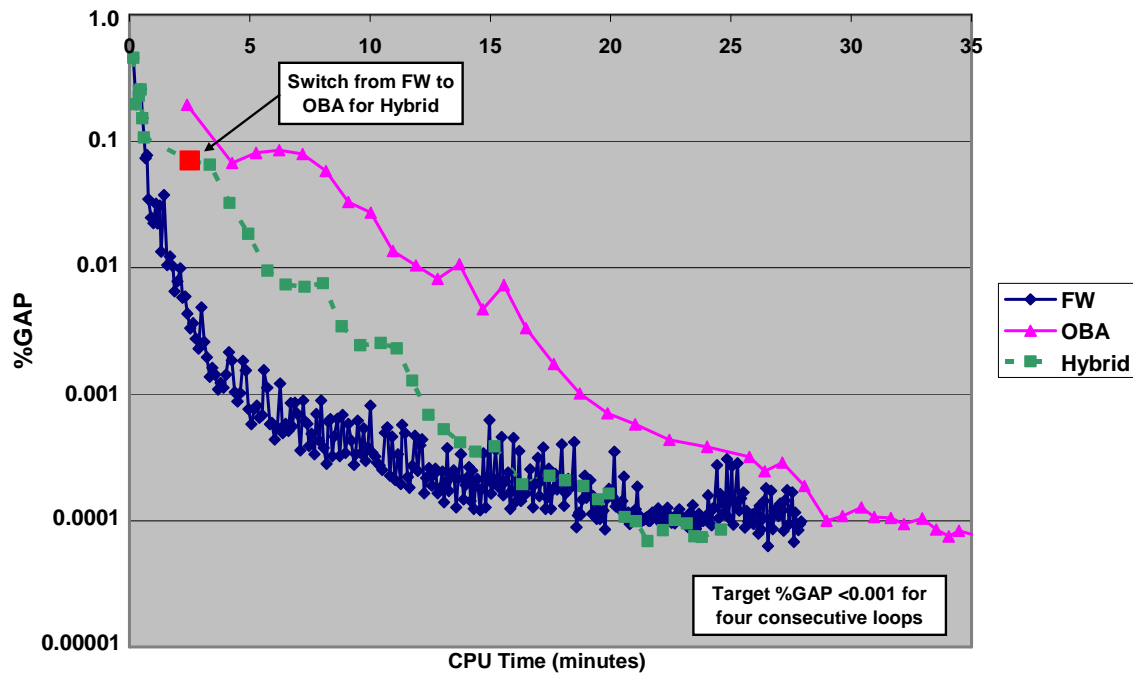
Figure 5 - SATURN Convergence (Model 3)

Figure 6 provides a similar comparison for the larger Model 4 with the assignment also terminated when the convergence criteria (%GAP < 0.001) was achieved. The MUC OBA and Hybrid algorithms performed less well with FW achieving the convergence target much more quickly. The MUC OBA algorithm eventually achieved a lower %GAP value but a considerable extra CPU overhead was incurred. Conversely, the Hybrid algorithm incurred some oscillations in the convergence caused by variations in flows and cost-delay functions in the simulation between successive loops.
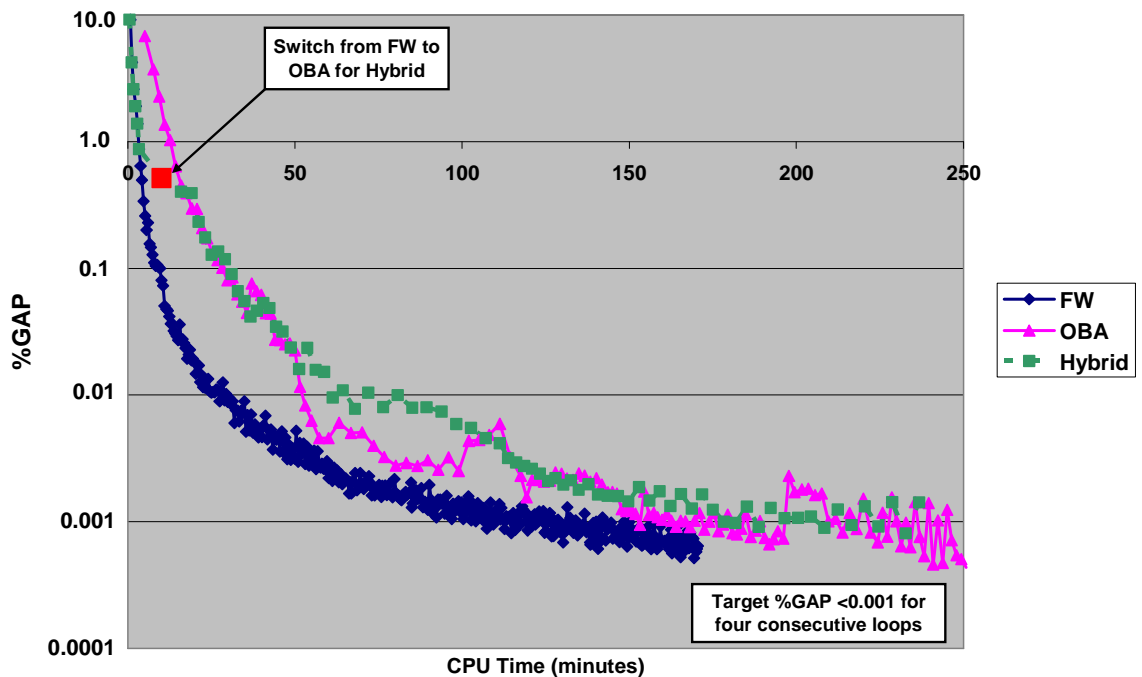
Figure 6 - SATURN Convergence (Model 4)

Figure 7 provides the same comparison for the Model 5 with the assignment terminated when the convergence criteria (%GAP < 0.01) was achieved. Whilst the MUC OBA converges much more slowly, the Hybrid algorithm is comparable to the FW algorithm in terms of CPU expenditure to reach the same convergence.

Figure 7 - SATURN Convergence (Model 5)
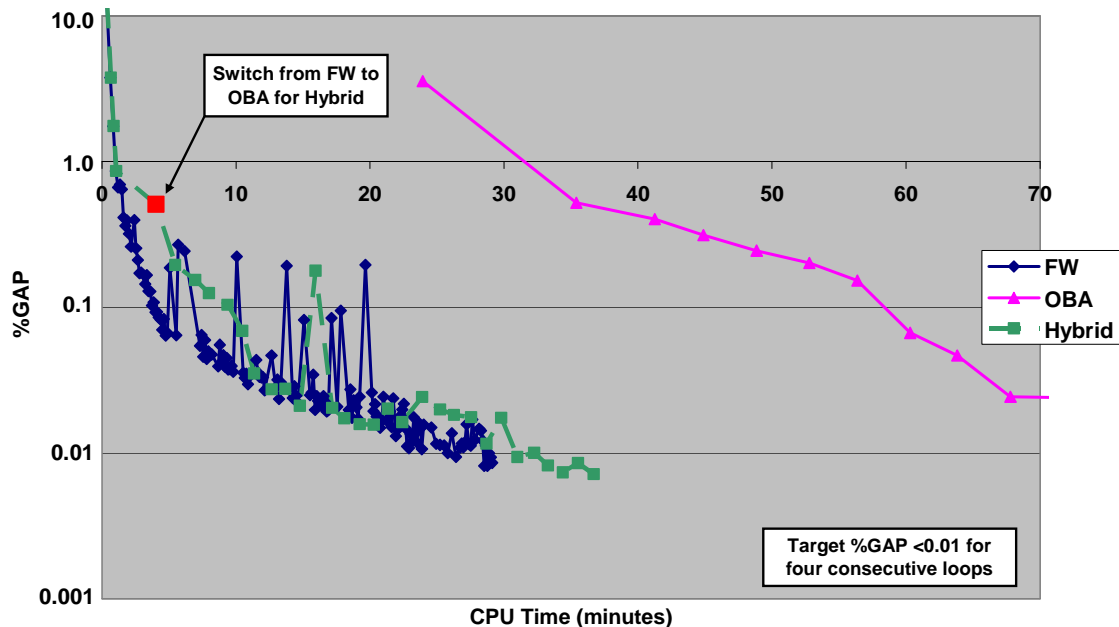
For completeness, Figure 8 shows the performance of the Hybrid model for Model 2 previously tested in Section 3. As with the other models, the Hybrid algorithm shared the initial characteristics of the FW algorithm before providing an OBA solution to a higher level of accuracy.

Figure 8 - SATURN Convergence (Model 2)

11
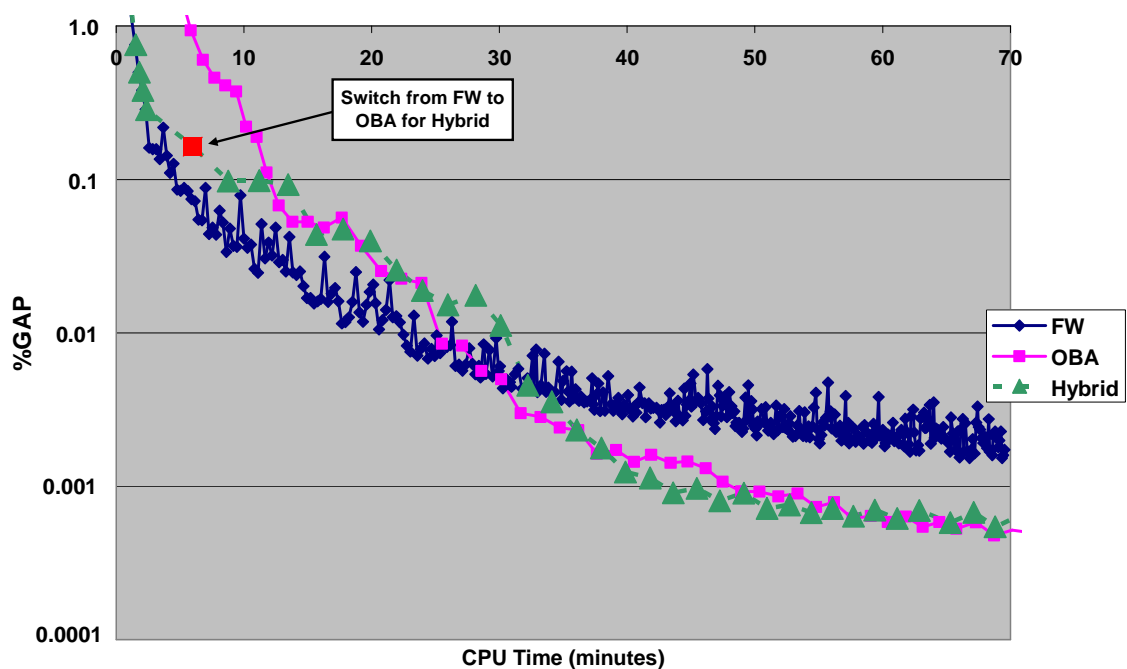
SUMMARY

By combining the rapidity of the FW algorithm in the early stages and subsequently switching over to OBA MUC algorithm, the Hybrid algorithm has shown performance advantages and its potential in solving real-life traffic assignment problems:

- It is (generally) faster than OBA MUC alone but also more accurate than FW in assignment convergence; and

- By using (and storing) route proportions, any secondary analysis may be undertaken without needing to re-build the paths used by FW.

5. CONCLUSIONS

The development of the SATURN-based **OBA MUC algorithm** has been presented in this paper. The performance of OBA MUC has been compared against the existing FW algorithm for four real-life applications and its theoretical superiority in achieving higher levels of convergence has been demonstrated.

These **higher levels of convergence** now achievable in SATURN will provide practical benefits to other models that are sensitive to convergence including demand models (e.g. DIADEM) and cost-benefits models (e.g. TUBA) for example.

The **hybrid FW-OBA MUC algorithm** brings together the advantages of FW and OBA MUC and has led to improvements in computer runtimes and/or assignment accuracy. The switch to OBA guarantees greater accuracy in the final equilibrium assignment such that the impact of assignment noise is significantly reduced (compared to existing FW-based processes) and high levels of convergence.

On a more practical-side, the hybrid algorithm has offered **reduced model runtimes** in some networks and **further efficiencies in secondary analysis** as the route information is stored during the assignment. The hybrid algorithm has also recently been combined with the multi-threaded implementation of FW available in SATURN Multi-Core and this has demonstrated further reductions in CPU required – however, this discussion is beyond the scope of this paper.

Development work continues on the hybrid method already available in the latest SATURN Beta versions with work focussing on further optimisation of the switch-over strategy and associated controls.

The current OBA MUC development work has also reinforced the importance of good quality network coding. Convergence within SATURN is sensitive to the interactions between the assignment and simulation loops. New OBA algorithms are able to achieve very high levels of convergence **but** only if the network coding enables robust and stable estimates of flow and travel costs to be calculated.

6.   REFERENCES

Bar-Gera, H and Van Vliet, D (2003) The application of Origin-based assignment to SATURN networks with asymmetric cost functions, European Transport Conference 2003.

Bar-Gera, H. and Boyce, D. (2003) Origin-based algorithms for combined travel forecasting models, Transportation Research **37B** (4), pp. 405-422.

Bar-Gera, H. (2002) Origin-Based Algorithm for the Traffic Assignment Problem, Transportation Science 36 (4), pp 398-417.

Bar-Gera, H. (1999) Origin-Based Algorithm for Transportation Network Modeling, National Institute of Statistical Sciences (NISS), Technical Report Number 103.

Kupsizewska, D and Van Vliet, D. (1999) UK 101 uses for path-based assignment, European Transport Conference 1999.

Van Vliet, D. Hall, M.D. and Willumsen, L.G. (1980) SATURN - A Simulation-Assignment Model for the Evaluation of Traffic Management Schemes, Traffic Engineering & Control, 21, 168-176, April 1980.

Frank, M and Wolfe, P (1956), An algorithm for quadratic programming, Naval Research Logistics Quarterly, No. 3 1956, pp. 95-110.