Comparison of Methods for Increasing the Performance of a DUA Computation

Michael Behrisch, Daniel Krajzewicz, Peter Wagner and Yun-Pang Wang
Institute of Transportation Systems, German Aerospace Center, Germany

Abstract

Computing realistic routes for a given road network and a known demand of vehicles is one of the most important steps when preparing a road traffic simulation. The approach developed by Christian Gawron in 1998 which we use for this purpose computes a dynamic user equilibrium by iteratively performing the simulation and computing new vehicle routes. The results are valid, but the computation is very time consuming due to the need to perform both the complete simulation and rerouting of all vehicles within each iteration step. Herein, we want to discuss some approaches to reduce the needed time and memory consumption. The results show that this can be achieved without reducing the algorithm's quality.

Keywords: microscopic simulation, traffic assignment, SUE, DUA

1 Motivation

The dynamic user assignment method developed by Gawron [1] was implemented as the default assignment method for the open source microscopic traffic simulation package SUMO [2-5] from the very begin on. Within the past few years, SUMO has been used in several large-scale projects where whole cities had to be simulated [6] and Gawron's algorithm has proven to be a valid tool, which allows using a demand based on OD-matrices in the microscopically simulated road network. Because of using travel time information from the simulation and iteratively adjusting vehicles' routes to this, jams are solved automatically and the generated route set results in realistic simulation.
Nonetheless, the preparation of routes for the simulated scenarios was often painful, due to the large computation time of up to several days the algorithm needs. Since the algorithm has to be rerun even after small changes in the network, waiting such a long time for the results is often unacceptable.
Furthermore, if dealing with even larger scenarios, which include major cities with their surrounding area and over one million vehicle journeys, one further issue arises: as the algorithm retrieves the edge travel times from the simulation in each iteration step, and initially all vehicles are using the shortest route in the empty network, jams and network deadlocks are quite common within the first iteration steps. The simulation is then filled with several hundreds of thousands vehicles which causes severe memory problems.

Due to the above mentioned problems, two approaches for speeding up the computation were implemented and evaluated. In this publication, Gawron's DUA algorithm will be presented first, followed by a description of the observed reasons for large computation times and simulation failures. After this, two modifications are introduced, followed by a comparison of the original and the modified algorithms. The publication is closed with a conclusion and ideas for future improvements.

2 Original Algorithm and Proposed Modifications

The dynamic user assignment algorithm developed by C. Gawron is a microscopic approach meaning that each vehicle has its own route and knows its own travel time through the network when using it. The basic procedure is as follows:
Step 1: Initialize the process by computing the fastest route through the empty network for each vehicle to simulate. Then add this route to the driver's list of known routes, set its probability to be chosen to 1, and choose it as the one to use.
Step 2: Perform the simulation using the currently chosen routes in order to obtain the edges' travel times over simulation time.
Step 3: Compare the mean travel times to the last run (if any) and quit if the algorithm converges, i.e. if the mean travel time reduction falls below a given threshold.
Step 4: Compute new routes for vehicles using the current travel times within the network for all drivers. Then add new fastest routes to the respective driver's route list and update all routes' travel times as estimated by the driver and the route choice probabilities for all routes the driver knows. After that, choose one route regarding the route choice probabilities and continue with step 2

In the following, we focus on the algorithm that generates new vehicular routes (step 4) and do not put emphasis on the use of a particular simulation model needed to compute the vehicle's and edges' travel times.
At first, the travel times for the routes known by a driver are adapted to the travel times obtained from the simulation:

$$\tau'_d(x) = \begin{cases} \tau_s(x) & x = last\_chosen\_route \\ \beta\tau_r(x) + (1-\beta)\tau_d(x) & otherwise \end{cases} \quad (1)$$

where $\tau_d(x), \tau_s(x), \tau_r(x)$ are route x's prior travel times as estimated by the driver, retrieved from the simulation, and reconstructed from the edge travel times that were determined by the simulation, respectively

$\tau'_d(x)$ is driver d's new estimation of the duration of route x

β is a factor affecting the speed of adapting remembered travel times to the current

The choice probability of each route is then recomputed. The probability for each unused route known by the driver is recomputed by a function that compares its travel time with the travel time of the route used in the last simulation step. The later route's probability is also adjusted, herein. This is done using the following equations:

$$p'_d(r) = \frac{p_d(r)(p_d(r) + p_d(s))\exp(\frac{\alpha\delta_{rs}}{1-\delta_{rs}^2})}{p_d(r)\exp(\frac{\alpha\delta_{rs}}{1-\delta_{rs}^2}) + p_d(s)} \quad (2a)$$

$$p'_d(s) = p_d(r) + p_d(s) - p'_d(r) \quad (2b)$$

where $p_d(x)$ is the prior probability to use route x by driver d

$p'_d(x)$ is the new probability to use route x by driver d

r is the route used in the last simulation run, s another route from the list of known routes

δ_{rs} is the relative costs difference between routes r and s, computed as:

$$\delta_{rs} = \frac{\tau_d(s) - \tau_d(r)}{\tau_d(s) + \tau_d(r)} \quad (3)$$

where $\tau_d(x)$ is the travel time for driver d to complete route x.

As described before, the list of routes known by a driver is not computed initially. Instead, a new fastest route is computed in each iteration step and added to the list of known routes, if it was not known before.

Within the investigations described herein, edge travel times were collected and aggregated within the simulation for intervals of 900 seconds. These time series were read by the router and used for computing new routes and the travel times for already known routes. Herein, each edge's travel time was determined by looking up in the corresponding time-series generated by the simulation for the interval that matched interval begin <= entry time < interval end. If the travel time for an edge is asked for which no matching interval exists, which can only be the case if asking for times later than the simulation's end, the last interval's value was used. α was set to 0.5, and β to 0.9. These values were found to be proper within previous projects and generate a valid route set, where "valid" means in this case that using this route set, all vehicles are simulated, manage to end their route at the end of the simulated day, and neither large jams nor network deadlocks occur.

3 Observed Bottlenecks

Two processes are responsible for computing the DUA, the first is the routing application which computes new vehicle routes, and the second is the simulation which determines the edge travel times using the new vehicle routes. The major cause of computation time is easily identified if the execution times of both applications are plotted along the iterations as seen in Figure 1.

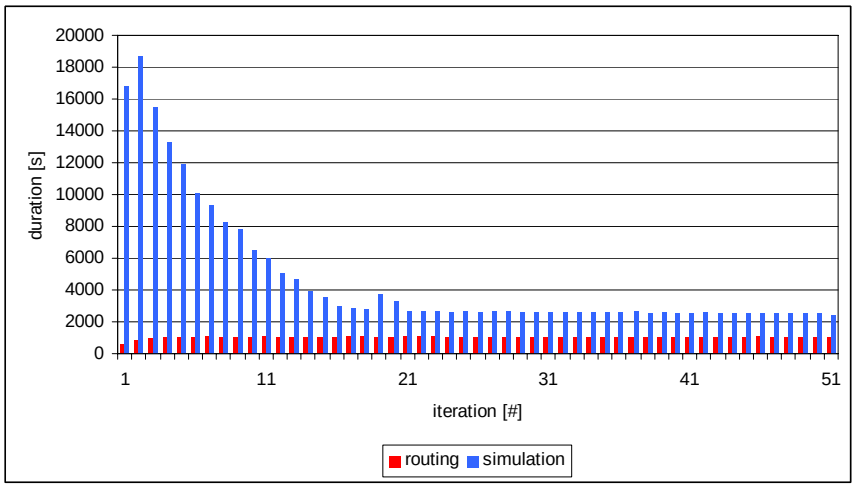

Figure 1: *Duration of the route computation (red) and the simulation (blue) over iterations for the test scenario (described in chapter 5)*

Obviously, most of the computation time is used by the simulation during the first runs. Along the DUA iterations, this time (almost monotonously) decreases during the first steps. The reason why the simulation needs much time in the first steps can be found by observing the number of vehicles within the simulation over the simulation time and iteration step (see Figure 2a).

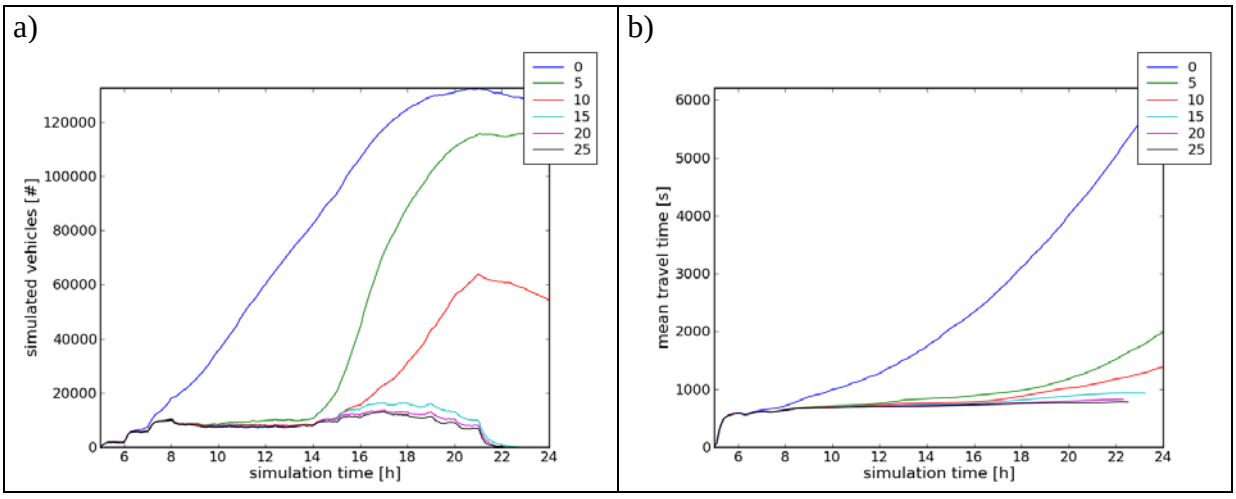

Figure 2: *Development of the maximum number of vehicles (a) and the vehicles' mean travel time (b) over simulation time and iterations (shown iterations: 0, 5, 10, 15, 20, 25)*

According to Figure 2b, one can see that, in the first steps of the assignment, many of the vehicles are staying within the simulation for a long time. Because they do not leave the simulation, the absolute number of vehicle movements the simulation has to compute is increased which results in a large duration. Also, because each vehicle has – besides other parameters – its own route stored within the simulation, the large number of vehicles that are within the network causes memory consumption problems.

Figures 2a also shows that the maximum number of vehicles that are within the network simultaneously is shrinking over iteration steps. As a conclusion, the major problem is the large number of vehicles kept in the simulation during the first DUA steps.

4 Proposed Modifications

Both proposed modifications do not change the route assignment itself, but only try to reduce the number of simulated vehicles, especially within the first iteration steps. The following methods that achieve this were evaluated:

a) Increasing vehicle amounts in the simulation

This approach resembles the macroscopic incremental assignment [7]. While route computation is still done for all vehicles in each iteration step with reference to the edge travel times computed by the simulation, the simulation only uses a fraction of the vehicles in the analyzed scenario. In each iteration step, this number of vehicles emitted into the simulation is increased. The motivation is

- to keep the simulation fast during the first steps by reducing the number of vehicles; and
- to reduce the number of jams that occur by reducing the number of vehicles.

The edge travel times generated by the simulation are used for finding new routes and computing the duration of known routes as in the original algorithm.

This approach will be named “inc_sim” in the evaluation section, meaning that the number of vehicles is incremented in the simulation.

b) Increasing simulation end time

Instead of increasing the amount of vehicles within the whole simulation, this approach uses the complete number of vehicles, but the time the simulation ends is increased in regular intervals.

The reason for trying this method is that, from our observations, simulated networks are jammed by “falsely” routed vehicles when the demand is high enough. Also, in most cases, we are interested in having simulations of whole days. This means that during the first simulated hours, with only minor traffic, no large jams occur even if no proper assignment was yet computed. The idea of the method is to benefit from the travel times collected during the first simulation runs, where only minor traffic is within the network and no jams occur, and incrementally increase the demand by following the daily demand time-series. Another benefit of this approach is that the number of vehicles inserted into the network is bounded by the simulation end time.

The edge travel times generated by the simulation are used as described earlier. In the next sections, it will be named “inc_end” for “increasing the simulation end time”.

However, both methods loose information. As travel times over edges are not proportional to the number of vehicles that use the edge, both modifications lead the router to operate on edge travel times that do not correspond to the real demand. Nonetheless, because the original algorithm also changes the assignment in a probabilistic way and, instead of operating on global measures, only takes into account a driver’s knowledge about the network. One can thus be optimistic that the information loss can be compensated incrementally, while the number of vehicle (inc_sim) or the simulation end time (inc_time) is increased.

The lack of possibility to evaluate the effects of this information loss makes it necessary to evaluate both methods using a simulation.

5 Evaluation

The scenario we used for evaluating the different methods was developed for the INVENT project and contains a road network and the respective demand for a normal weekday for the city of Magdeburg using 1h-matrices for the hours between 5am and 9pm. Both the network and the demand were given in VISUM format, so that several modifications were needed for making it useable in a microscopic simulation. The road network was extended by highway on and off ramps, lane-to-lane connections, and traffic lights. The demand was converted from the original O/D-matrices that use about 250 districts into single vehicle trips. The resultant scenario consists of a network made of about 4800 edges (roads) and 2500 nodes (junctions) and a traffic demand of about 600,000 vehicles.
The evaluation was done by executing 50 iterations of the original algorithm, which was known to compute a valid route set for the scenario. Then, the following modification settings were run for also 50 iterations, each:

- "inc_sim10": inc_sim increasing the number of simulated vehicles in steps of $1/10^{th}$ of the demand
- "inc_sim20": inc_sim increasing the number of simulated vehicles in steps of $1/20^{th}$ of the demand
- "inc_sim30": inc_sim increasing the number of simulated vehicles in steps of $1/30^{th}$ of the demand
- "inc_sim40": inc_sim increasing the number of simulated vehicles in steps of $1/40^{th}$ of the demand
- "inc_sim50": inc_sim increasing the number of simulated vehicles in steps of $1/50^{th}$ of the demand
- "inc_time1800": inc_time using steps of 1800s
- "inc_time3600": inc_time using steps of 3600s
- "inc_time5400": inc_time using steps of 5400s
- "inc_time7200": inc_time using steps of 7200s

We chose three main indicators for evaluating the quality of the assignment process among the different methods. The first one is the execution time which we try to reduce; the second one is the memory consumption, which shall be reduced, too. While we measure the duration directly, we decided to use the maximum number of vehicles, which are simultaneously within the simulated area, as a measurement for memory consumption. There are three reasons for using this value: a) the number of currently simulated vehicles is directly available within the simulation and can be saved and evaluated, b) because the network and other infrastructure information stay the same over the described simulation runs, the maximum number of simultaneously running vehicles is the only reason for different memory footprints of the application, and c) this value does not depend on the used architecture or a single vehicle's memory size. The third indicator is the mean travel time of all vehicles to simulate after the simulation's end. In the case that all vehicles have left the network, this gives a good assessment of the assignment's quality, because it prefers faster routes and penalizes jams, retrieving better scores for assignments that simulate drivers who want to reach their destination fast.
Figure 3 shows the maximum number of vehicles that were within the simulation simultaneously. All iteration steps of each of the named methods were taken into account. As expected, the maximum number of simulated vehicles is reduced within the proposed methods, when

compared to the original ("plain") method. Still, in almost all cases, except inc_sim50, this value is higher than that for the simulation in the equilibrated state, as obtained from the 50th iteration of the plain method, which is shown as a red line in Figure 3. This means that, during the process, jams occur which may fill the simulation under circumstances.

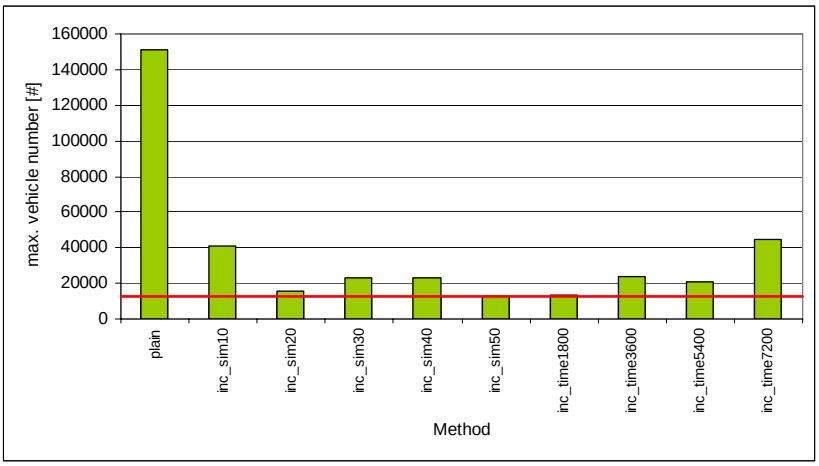

Figure 3: *Maximum number of vehicles within the simulation over all iteration steps of each of the evaluated methods; red line: maximum number of vehicles in the final step of the original algorithm*

The reduced number of the simulated vehicle movements results in a lower computation times, as indicated in Figure 4. Each of the bars shows the time in hours that was needed to perform 50 iterations for each method. In most cases, except inc_sim50, where the number of really simulated vehicles was reached in the last iteration step, the respective method needed less time to compute an assignment for all vehicles which resulting mean vehicle travel times were lower than the one of the original algorithm. Each bar shown in Figure 4 pictures the complete time needed by the respective method to perform 50 iteration steps. The dark component of each of these bars shows the time, the according method needed to compute an assignment which results in a mean travel time that lies below the one obtained using after performing the original method for 50 iterations. The light component is the additional time of the 50 iterations, that the respective method used to further improve the assignment beyond the quality, reached by the original method.

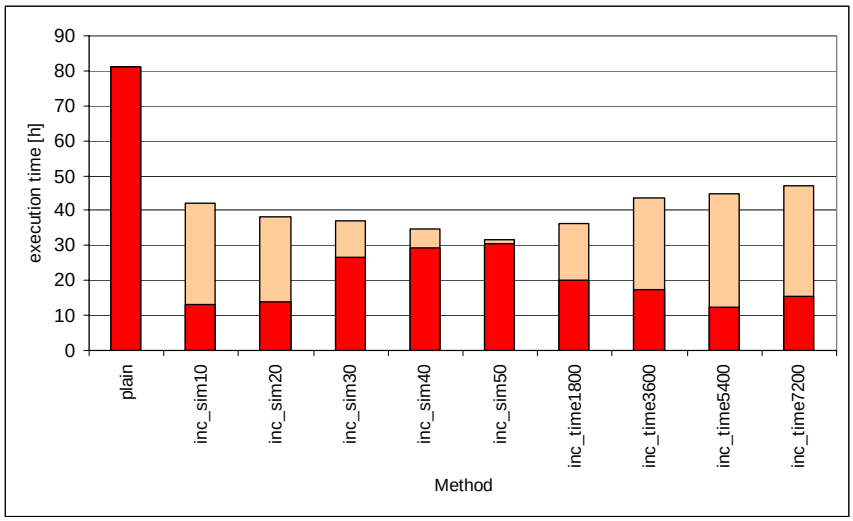

Figure 4: *Execution time of 50 iterations of each of the methods (colors: see text)*

Because most of the modifications were able to reach the original algorithm's quality in less than 50 iterations, the following iterations could achieve further improvement of the mean travel times, shown in Figure 5.

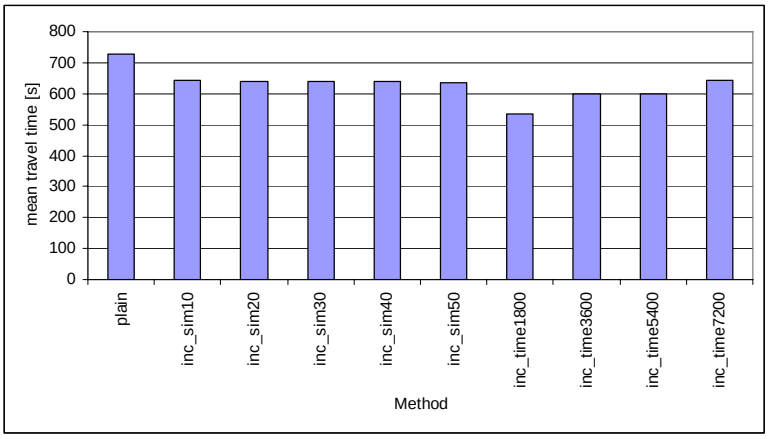

Figure 5: *Mean vehicle travel time after 50 iterations of the methods*

The figures show that using any of the modifications is of benefit. Now, one should ask the question which method and parameter set is the best. Though some of the presented methods need a lower execution time to achieve the same quality (see Figure 4), inc_time1800 seems to combine both of the described qualities – execution speed and reduction of used memory – at best.

6 Conclusions and Future Work

We have shown that both approaches, increasing the number of simulated vehicles after staring with only a fraction of the complete demand, and to slowly increase the simulation end

time, result in a faster computation of the DUA without loosing the original algorithm's quality.
However, both methods have the drawback that their parameter, the percentage with which the number of the simulated vehicles is increased, and the amount of time by which the simulation's end is extended, respectively, have to be given explicitly. The next development steps are to establish a system which recognizes starting jams within the simulation and uses this information as a feedback to the DUA system. Such a system should then adapt the parameter to the current DUA progress.
Also, for the "inc_sim" approach, we have only considered changing the fraction of the processed vehicles within the simulation. Furthermore, it should be also evaluated whether all routes must be computed by the router module and whether only computing the used ones can reduce the computational effort without affecting the results' quality.

7 References

[1] Gawron, C. (1998). Simulation-based traffic assignment – computing user equilibria in large street networks. Ph.D. Dissertation, University of Köln, Germany.

[2] Krajzewicz, D., Hertkorn, G., Rössel, C., Wagner, P. (2002). SUMO (Simulation of Urban MObility): an open-Source Traffic Simulation. Proceedings of the 4th Middle East Symposium on Simulation and Modeling (MESM2002), 183 - 187, SCS European Publishing House.

[3] Krajzewicz, D., Hartinger, M., Hertkorn, G., Nicolay, E., Rössel, C., Ringel, J., Wagner, P. (2004). Recent Extensions to the open source Traffic Simulation SUMO. Proceedings of the 10th World Conference on Transport Research (on CD), WCTR04 - 10th World Conference on Transport Research, Istanbul (Turkey), 4-8 Jul. 2004.

[4] Krajzewicz, D., Bonert, M. and Wagner, P. (2006). The Open Source Traffic Simulation Package SUMO. In: RoboCup 2006, RoboCup 2006, Bremen, Germany, 14-20 Jun. 2006.

[5] ITS/DLR. SUMO Website: http://sumo.sourceforge.net. 2001-2008

[6] Behrisch, M., Bonert M., Brockfeld, E., Krajzewicz, D. and Wagner, P. (2008). Event traffic forecast for metropolitan areas based on microscopic simulation. International Symposium on Transport Simulation 2008.

[7] Thomas, R. (1991). Traffic assignment techniques, Vermont, USA, Avebury Technical.