Multi-modal traffic in TRANSIMS

Kai Nagel
Swiss Federal Institute of Technology (ETH) Zürich, Switzerland

April 10, 2001

Abstract

Traffic simulations need to include other modes of transportation besides car. This paper explains how this can be implemented in a large scale micro-simulation package. The most important element is to recognize that trips start and end on the walk network, and to explicitely model this mode.

1 Introduction

Transportation planning considers questions of the long-term impact of infrastructure or policy changes – such as an additional subway line or an increase of gasoline costs. Typical questions are: Where does traffic/congestion shift to? Will trips be dropped or added? Might people change where they live or work as a longterm response to changes in the transportation system?

Traditional models for transportation planning typically put most of their emphasis on car traffic. Other modes are often considered via modal split calculations only – that is, once transportation demand has been calculated, some of the demand may decide to take other modes instead of the car, usually via a discrete choice model based on travel times and different values of time on the different modes.

The reason for this is that static assignment models do not have a realistic representation of the dynamics. For example, time-dependent effects of car traffic, such as queue build-up and spill-back during the rush-hour, cannot be represented. Similarly, more complicated effects in other modes, such as congestion of a pedestrian facility, or bus-car interaction, cannot be modeled.

The approach is nevertheless justified as long as the other modes quantitatively do not play a large role, and it may be justified under certain circumstances if car mode is the only congested mode. In many metropolitan regions, this is not true: other modes besides cars are contributing more than 25% of all trips, and these other modes are heavily congested.

As a result of these and other thoughts, there is currently a push towards microscopic (or agent-based) transportation planning models. In these models, all entities including

the travelers themselves are individually resolved. This makes is in principle straightforward to include arbitrary elements of reality. The downside is that it may be hard to implement, it may use considerable computer resources, and data for calibration of these effects may be hard to obtain. Nevertheless, microscopic transportation simulation models are feasible, and several groups are working on them.

This paper will first give an introduction into how such microscopic transportation simulation packages are designed. This is followed by a section about a possible implementation of multiple modes into such a package. The paper is concluded by a discussion and a summary.

2 Large scale transportation simulations

One aspect of any transportation simulation package needs to be the actual traffic dynamics. Ideally, one wants to have travelers walking to the vehicles, entering the bus or the car, then those vehicles drive through the traffic, stop at lights, obey speed limits, and make turns at intersections after getting into the necessary lane. This is the task of the traffic micro-simulation module.

One aspect of such a traffic simulation is that travelers need to know how they navigate through the system. When do they enter a bus, when do they make a turn, etc.? This task is typically solved by a routing module.

Any routing module needs as input the origin and the destination of the trip. Traditionally, one uses origin-destination matrices here – that is, tables which contain information about how many people travel between any pair of destinations in the region. A disadvantage of this method is that all information about the travelers themselves is lost at this level – and in consequence, decisions cannot be coupled to such information any more. For example, a person who arrives late at work because of congestion may skip lunch in order to catch up. An alternative method of demand generation for travel is called “activity-based”. This method is closer to how individual humans think and plan – that is, for each individual in the simulation the module generates individual plans for activities.

It is not sufficient to run these three modules in sequence since, for example, plans depend on congestion but congestion depends on plans. In order to find a solution which is consistent between the modules, it is common to run feedback iterations, where the agents slowly adapt to the situations they encounter. – The next sections will describe the modules including feedback in more detail.

2.1 Demand generation

In order to generate the demand for transportation, many of today’s projects are based on activities. That is, instead of using an abstract origin-destination matrix, they look into people’s motivation to travel. This is achieved by having the simulation generate sequenced activities for the day, for example: Sleep until 7am. – Be at work at 9am. – Go to lunch at 12:30pm for one hour. – Leave work after 8 hours of work. – Go to the

kindergarten to pick up a child. – Go shopping. – Go home and stay there for the rest of the day. These activities come with locations, that is, they generate demand for travel. Two successive activities which take place at two different locations will generate a trip request.

It is clear that one needs to have information about the network impedance (i.e. the travel times between different locations) when making activity plans. We will come back to this in the section about feedback and in the outlook.

Once the simulation "knows" where and when people do their activities, transportation is generated via connecting activities that take place at different locations. This includes modal choice (walking, bicycle, train, car, etc.) and the precise routing.

2.2 Traffic simulation

So far, we have generated "plans" of the individuals. The travelers in the simulation compute their plans for the following time period, which may for example be a day. Now these plans need to be executed, which is called the traffic micro-simulation. These simulations come at many different levels of resolution and fidelity, reaching from the traditional steady-state flow-based cost function to very detailed micro-simulations.

If one is interested in time-dependent results, as for example the queue build-up during the onset of rush periods, the simulation needs to be sufficiently realistic to contain such dynamics. Traditional flow-based cost functions are *not* able to realistically deal with such dynamical effects, at least not in a straightforward way. Thus, the right traffic simulation has to be chosen according to what aspects of the dynamics one wants to have represented for a given question. There are currently more than hundred traffic microsimulations [1]. However, in most of these travelers do not follow individual plans as was explained above. Examples of plan-following traffic simulations are TRANSIMS [2], DYNAMIT and MITSIM [3], DYNASMART [4], and LEGO [5]. In some of these, travelers do not "know" their full routes but only their destinations (and are routed via the simulation which knows the paths); the practical impact of this difference is not known.

2.3 Feedback

The traffic simulation needs input from the demand generation, since it executes the plans from the demand generation. However, the demand generation depends on the traffic simulation because for example congestion only shows up in the traffic simulation, and demand adjusts to such shortages. In order to deal with this situation, one iterates between demand generation and traffic simulation. For example, the demand generation module is run assuming no congestion, the resulting traffic simulation is run, then the demand simulation is run again now including the congestion from the last traffic simulation run, etc., until a steady state is reached. That is, the system is systematically relaxed towards a consistent state.

An important issue in this context is the question of computational efficiency versus behavioral realism. Traditional static equilibrium assignment has, once the origin-

destination matrix and the network including link-cost function are specified, a unique solution (in terms of the paths flows). This allows one to concentrate on the fastest computational algorithm to find this solution. In iterated transportation simulations, this issue is considerably more complicated. Although such iterated simulations relax to some kind of reproducible steady state, this state is stochastic. Also, it is not clear if this state is unique or if it depends on the path of the computation – in dynamic traffic assignment (DTA) computational evidence indicates that it is unique [6, 7] but once one includes other aspects such as public transit or activities rescheduling it is easy to construct scenarios where this is no longer true.[1] Another important aspect is that real systems may not be at this relaxed state at all. In that case, transients matter much more, which means that the rules of the synthetic travelers would have to be much more realistic than for just reaching the relaxed state.

2.4 Computing

A metropolitan region easily consists of several millions of travelers. In order to get an idea of the necessary computing let us assume that we want to simulate a scenario of 24 hours (= 86400 secs) with 1 million travelers. Simulations typically do second-by second updates. In each update, several variables such as traveler speed or traveler location are computed. Let us assume that there are ten such variables, and the update of each variable needs 100 basic computer cycles. This is not a lot: For example, a simple algorithm for lane change will consider the following: Do I need to change lanes? Where are my neighbors? Is there enough space for me to change lanes? Also, fetching data from RAM typically takes about 10 cycles. A result of these assumptions is that, with a 1 GHz-CPU, the scenario takes

$$\frac{24 \text{ hours} \times 10^6 \times 10 \times 100}{1 \text{ GHz}} \approx 24 \text{ hours}$$

of computing time. Running 50 iterations thus will take 50 days of computing time on a single CPU. Using parallel computers will reduce this number accordingly; for example, using a so-called Beowulf of 50 Pentium PCs will lead to a computing time of between one and two days. See Refs. [9, 10] for more information.

3 Multi-modal approach

3.1 Conceptual approach

The arguably best starting point for getting a clean concept of a multi-modal transportation simulation package is to recognize that all activities take place on the walking network. Even if the car is parked in the garage, people have to walk to it. As a result,

[1] Proofs of ergodicity [8] for such systems do not always apply for actual implementations, and they are also often not practical since they imply thousands of iterations instead of the usual number of fifty.

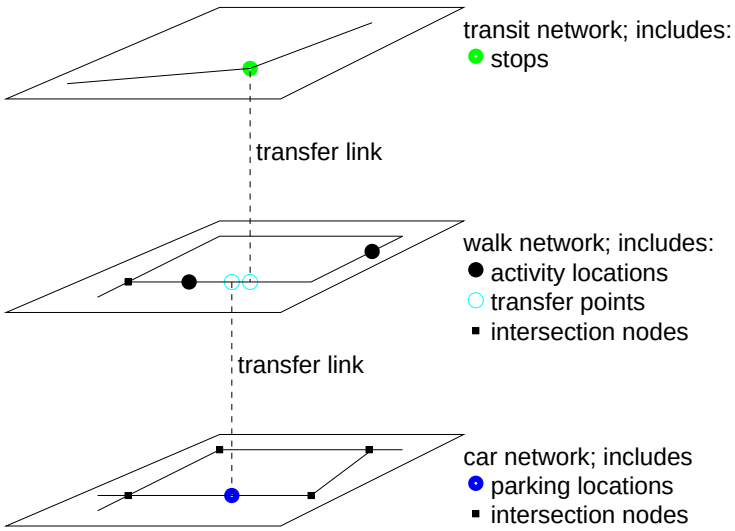

Figure 1: Conceptual representation of network layers

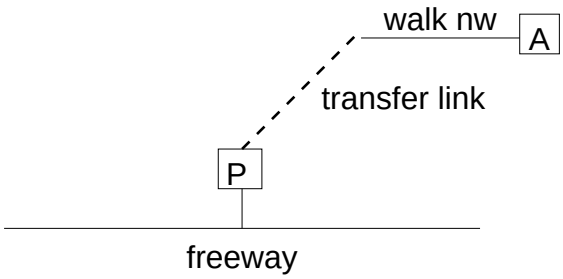

Figure 2: Conceptual representation of an activity location which can only be reached via a freeway.

the simulation needs to have a walking network, on which all activities are located, and which has transfer points to the other modes of transportation (Fig. 1).

In the actual implementation, the walking network may look similar to the car network, and it may use the same nodes and links, but conceptually it is a separate thing. The walking network does not have to be connected; Fig. 2 shows an example of a activity location which can only be reached via a freeway.[2]

[2]Also, in this construction, different lines of public transit are not connected – if one wants to change transit lines, one first has to go, via a transfer link, back to the walking layer. In consequence, the transit layer consists of a number of unconnected lines.

3.2 Multi-modal activities

TRANSIMS activities files come with a specification of the desired mode how an activity should be reached. This specification is a mode string – a sequence of letters denoting the different modes. Examples for mode strings are:

- wcw: walk–car–walk
- wbw: walk–bus–walk
- wbwbw: walk–bus–walk–bus
- wtw: walk–transit–walk (transit includes walking)

This mode specification can be changed via the feedback process. That is, a traveler can try out different modes, and eventually settle on one according to arbitrary performance criteria. This will be described in more detail on a section on feedback.

3.3 Multi-modal router

It is possible to construct a search graph such that a generalized Dijkstra algorithm can be used to find a "best" multi-modal path through the network [11]. The construction assumes that the mode string is given, for example "wbwbw", where again "w" stands for walk and "b" stands for bus. This example would involve a route which uses two different bus lines.

The search graph for the algorithm is obtained as follows (see Fig. 3): For each additional entry in the mode string, a separate network layer is assumed. That is, for "wbwbw" we have, from bottom to top, first a walking layer, then a bus layer, then again a walking layer, etc. The layers are connected via the transfer links as was already discussed above. The main difference is that for the algorithm, network levels are replicated if they appear at more than one position in the mode string.

When consulting Fig. 3, it becomes clear that a path through the search graph can be found in an organized way via a Dijkstra algorithm: Assume that the starting location is on the left, and the traveler wants to reach a destination on the right. Since we have "wbwbw" as mode sequence given, the algorithm will construct five levels in exactly that sequence, as can be seen in the figure.

The Dijkstra algorithm will then start in the bottom left, and expand systematically in its usual way through the search graph. Transfer links are logically treated the same way as links for other modes, that is, they have a link travel time and possibly other attributes. In this way, the algorithm will deterministically find the fastest path through the expanded network, which will correspond to a path of the desired mode sequence. Note that it will be possible to stop the algorithm if the destination can be reached faster without changing bus lines. For further information, see Ref. [11].

The router also keeps track of car availability. It will plan car trips only with available cars (which are usually parked close to home), and it will make sure that the car is returned at the end of the day. In complicated situations (such as a different family

member returning the car), it is however the microsimulation which will find errors and notify the selector to correct them. For example, the microsimulation will, during a 48h simulation, notice if a car was not returned the previous evening since it will not be available for the trip.

3.4 Router implementation

The above description is conceptual. For efficiency, the implementation does several simplifications and modifications:

- The walk network and the street network share the same nodes. This makes sense since most of them are the same anyway. By setting the links accordingly, it will still be possible to have nodes which can be reached by car only or by walking only.
- Activity locations and transfer points are made the same on the walk network.
- There is only at most one activity location and one parking location on each link. This means, for example, that all parking along a link is aggregated into a single location.
- In fact, the walking and street levels are not replicated at all in the actual implementation. Instead, a counter along the "fringe" nodes of the Dijkstra algorithm keeps track of where in the mode string sequence each fringe node is during the algorithm.

3.5 Multi-modal traffic micro-simulation

As said above, the micro-simulation executes all plans simultaneously in a realistic representation of the traffic system, and it is here that interactions between travelers, for example causing jams or missed busses, are computed. As also said above, TRANSIMS, in its current implementation, assumes that all plans (including routes) are pre-calculated before the simulation starts.

In principle, the micro-simulation is just a good representation of reality. With respect to multi-modal travel, this means that again all travel of persons begins at an activities location, which is located on the walking network. If the travelers use additional modes besides walking, the simulation moves the travelers to, say, the bus stop or the parking lot, where they enter a car.

That means, for example, that traffic lights follow schedules or adaptive procedures as they do in reality, and drivers change lanes according to traffic conditions or in order to be in correct lanes for intended turns. It also means that the simulation logically moves the travelers from their starting locations (usually home) to the bus stop or the parking lot, then lets them enter their vehicles, etc.

A convenient feature of the design as it is described here is that the simulation can actually deal with multi-modal plans even if the corresponding mode of travel is not

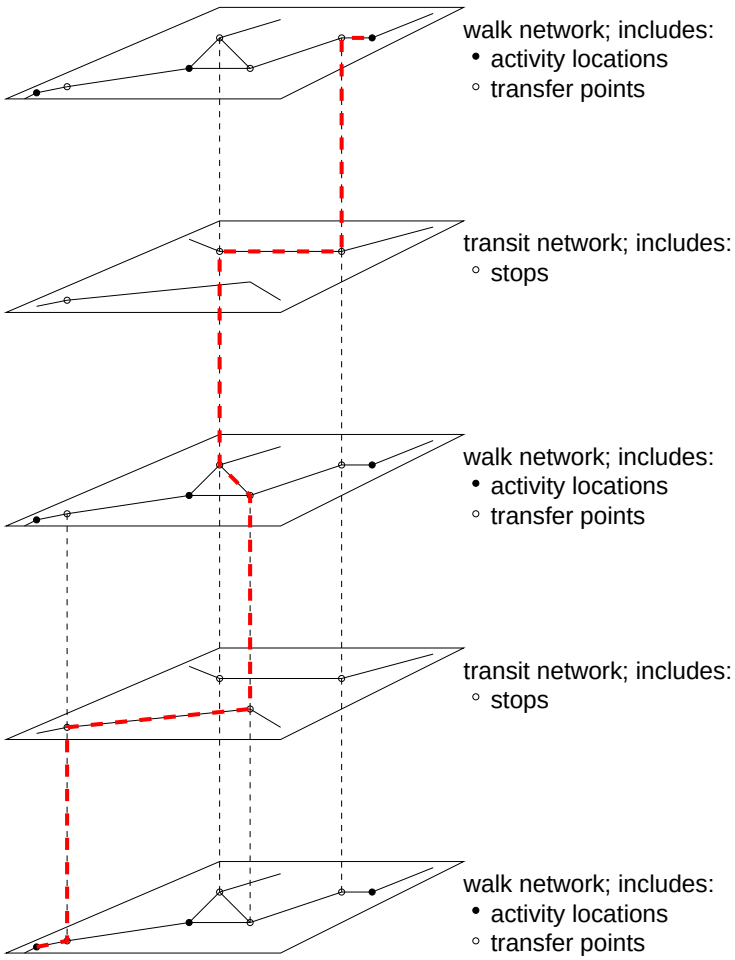

Figure 3: Conceptual representation of product construction, including one path.

implemented in the micro-simulation. For example, walking may not explicitely be implemented. In such a case, the micro-simulation will just assume that the plan contains completely correct information, and execute that. For example, the plan may say that the traveler leaves home at time t_1 and arrives at the bus stop at time t_2. If walking is not explicitely implemented, the micro-simulation will just assume that the travelers leaves home at t_1 and arrives at the bus stop at time t_2. Conveniently, this works for arbitrary modes which are not implemented, so that the router does not have to take into account possible limitations of the micro-simulation, and also micro-simulations with different capabilities can run on the same set of plans. It should be clear that for such modes no

interactions can be modeled. For example, when walking is not modeled, missing a train because of crowding at the subway station will not happen.

A different aspect of the multi-modal implementation is the actual representation of public transit schedules. In the TRANSIMS traffic micro-simulation, public transit service is generated the same way car traffic is generated, i.e. via drivers picking up their transit vehicle at the depot and following a prescribed route with prescribed stops, a prescribed schedule, and possibly prescribed procedures for the case of schedule deviations. This is similar to the modeling of a shared ride, where a driver in one car has to pick up one or several passengers and drop them off later. In that way, effects like the bunching of busses or schedule delays because of congestion will come out naturally. Also, buses have a capacity limit, possibly leaving passengers behind at the bus stops, and it is possible to make the loading/unloading times a function of the crowdedness. If the actual routes of drivers and transit vehicles are not available, then synthetic routes can be generated from the schedule. Such synthetic routes will however obviously not include certain effects of reality, as for example the effect that a vehicle which switches lines may not be available because of a delay.

3.6 Explicit pedestrian traffic in TRANSIMS?

Currently, TRANSIMS does not explicitely implement pedestrians; they are treated as any other non-implemented mode as described above. It should be clear that TRANSIMS could benefit from the addition of a realistic pedestrian mode, in particular in cities where pedestrian facilities suffer from congestion. It should also be clear that including pedestrian traffic into TRANSIMS would be relatively straightforward, and could be done via arbitrary models as long as they follow the same plans format. This, however, puts constraints on the pedestrian model design, namely that is has to follow the graph-oriented structure of TRANSIMS. In particular, it must be possible to define paths from one activity location to another, plus from activity locations to transfer points and vice versa. Agents need to be able to follow such paths, and the information must be structured in a way so that the router can work with it. This excludes, for example, random wandering across a wide plaza or through an old city where a person only approximately knows where he or she is. Future versions of TRANSIMS may eventually be extended in this regard. Also note that aspects of this can be included via putting delays on the transfer links.

3.7 Feedback

No computational module will be able to anticipate everything which can happen in other modules. For example, strongly fluctuating congestion could lead to missing the train although according to mean travel times everything looks fine. Similarly, the shared ride may depart without me if I am too late, or the parking lot at the park-and-ride station may be full. In many cases, researchers may want to define non-optimal behavioral decision models for humans, which would exclude from the beginning that travelers take, say, the fastest path.

For all these elements, TRANSIMS provides its feedback mechanism (also called "selector"). Instead of (computationally) dwelling forever on a decision between two different modes, or a range of possible starting times, the simulation can just decide "to try it out". For example, a transit and a car option can be tried by an individual, and the performance of both be evaluated. In that performance evaluation, aspects like the number of stop signs, or the predictability of the respective options can be taken into account. This allows both the modeling of non-optimizing behavioral models, and the inclusion of non-linear costs which cannot be picked up by Dijkstra-type routers (also see below). In the second case, an "optimal" solution probably cannot be guaranteed, but an individual improvement procedure can be implemented.

In many cases, this also solves the question of "where to put" certain decisions. For example, a mode choice decision could be made on the level of the router, via putting values of time and other things on different modes and then finding the optimal mode for the trip. It however also makes sense to look at this during the activities planning, since mode choice and activities chaining are intimately related. Via the feedback mechanism, neither of these modules has to make the decision alone – instead, in critical cases where several solutions are plausible, the selector will be able to make the individuals try out different solutions and then pick a good one according to heuristic (or optimizing) rules. Clearly, there is a tradeoff between modeling effort for the trip planning and the number of iterations one may have to run.

4 Discussion and Outlook

Any design has shortcomings. Although the TRANSIMS design seems robust in the sense that design decisions from many years ago have not gotten in the way of newer issues, there are always elements which are easier to model with a certain design and others which are harder to model. Here are some issues:

- The router will find a good or the best transit line for a trip. In many cases, however, several alternative lines could be taken, for example at typical transit malls in American Central Business Districts. In such cases, the router will pick one line which seems convenient from the schedule; but it may happen that, when the traveler arrives at the stop, another line might be better.

  Including these effects into the router is possible, but only by defining additional "virtual" transit lines for each segment of the transit network which is served by several lines. For example, if a transit mall is served along its entire length by bus lines 1, 3, and 9, then a new line with a not yet existing number could be defined along the length of the path where the three lines overlap. Unfortunately, this yields a possibly high number of such "virtual" lines, since all combinatorial combinations of lines and segments could become new virtual lines. Also, the micro-simulation would have to pick it up in some way, that is, the route plan would have to state which bus lines are equivalent. – What this still would not capture is if two different paths via different transit lines could bring the traveler

to the next activity.

- Dijkstra-type routers cannot deal with non-additive costs, which are used by most public transit fare systems or some road pricing schemes. The TRANSIMS router cannot deal with this directly, but at least for simple cases the feedback mechanism can pick it up. For example, it can force the simulation to try out both the transit and the car option, and add the monetary costs only at the point when the relative performance between both is evaluated.

5 Summary

This paper explains the concept of the implementation of multi-modal traffic in TRANSIMS. For this, the paper has two parts: A first, explaining the TRANSIMS design using examples from car-only runs, and a second, which explains the provisions for multi-modal trips. The emphasis of the paper is on the second aspect. TRANSIMS starts by assuming that all activities are located on the walk network, and in consequence any trip connecting two activities at different locations will involve walking at the beginning and at the end of the trip. Transitions to other modes are made via transfer points and transfer links. The activities generator of TRANSIMS provides the preferred mode to reach an activity, and the multi-modal router routes these trips. The traffic microsimulation reads these plans, executes them in detail if the corresponding mode is implemented, and "believes" them if the mode is not implemented. The latter means that for a non-implemented mode the traveler is just moved to her/his destination according to the timing provided in the plan; no interaction between travelers is computed in this case. This also gives a clear implementation path to the addition of realistic pedestrian modules. The TRANSIMS feedback mechanism provides an answer to some of the usual conceptual questions, for example of where to place the mode decision (into the activities planning or into the routing) or what to do about non-additive link costs.

Acknowledgments

The multi-modal implementation of TRANSIMS is documented in [2]. The work leading to this was started when the author was still at Los Alamos, as a collaboration of many including Chris Barrett, Dick Beckman, Madhav Marathe, Riko Jacob, Goran Konjevod, and Stephen Eubank. Many others have contributed to the current implementation.

References

[1] SMARTEST (Simulation Modelling Applied to Road Transportation European Scheme Test. See www.its.leeds.ac.uk/smartest/.

[2] TRANSIMS, TRansportation ANalysis and SIMulation System, since 1992. See transims.tsasa.lanl.gov.

[3] DYNAMIT/MITSIM, 1999. Massachusetts Institute of Technology, Cambridge, Massachusetts. See its.mit.edu.

[4] H.S. Mahmassani, T. Hu, and R. Jayakrishnan. Dynamic traffic assignment and simulation for advanced network informatics (DYNASMART). In N.H. Gartner and G. Improta, editors, *Urban traffic networks: Dynamic flow modeling and control*. Springer, Berlin/New York, 1995.

[5] In *Verkehr und Mobilität*, number 66 in Stadt Region Land. Institut für Stadtbauwesen, Technical University, Aachen, Germany, 1998.

[6] M. Rickert and K. Nagel. Issues of simulation-based route assignment. Presented at the International Symposium on Traffic and Transportation Theory (ISTTT) in Jerusalem, 1999. See www.inf.ethz.ch/~nagel/papers.

[7] J. A. Bottom. *Consistent anticipatory route guidance*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 2000.

[8] C. Cantarella and E. Cascetta. Dynamic process and equilibrium in transportation network: Towards a unifying theory. *Transportation Science A*, 25(4):305–329, 1995.

[9] M. Rickert and K. Nagel. Dynamic traffic assignment on parallel computers in TRANSIMS. *Future generation computer systems*, 17(5):637–648, 2001.

[10] K. Nagel and M. Rickert. Parallel implementation of the TRANSIMS microsimulation, submitted. See www.inf.ethz.ch/~nagel/papers.

[11] C. L. Barrett, R. Jacob, and M. V. Marathe. Formal language constrained path problems. Los Alamos Unclassified Report (LA-UR) 98-1739, see transims.tsasa.lanl.gov, 1997.