Katholieke Universiteit Leuven

Departement Elektrotechniek

ESAT-SCD (SISTA) / TR 06-058

Dynamic Traffic Assignment based on Cellular Automata

Sven Maerivoet[†] and Bart De Moor[†]

March 2006
Technical report

This report is available by anonymous ftp from *ftp.esat.kuleuven.be* in the directory *pub/sista/smaerivo/reports/paper-06-058.pdf*

[†]Katholieke Universiteit Leuven
Department of Electrical Engineering ESAT-SCD (SISTA)
Kasteelpark Arenberg 10, 3001 Leuven, Belgium
Phone: (+32) (0) 16 32 86 64 Fax: (+32) (0) 16 32 19 70
E-mail: {*sven.maerivoet,bart.demoor*}*@esat.kuleuven.be*
WWW: *http://www.esat.kuleuven.be/scd*
Our research is supported by: **Research Council KUL**: GOA AMBioRICS, several PhD/postdoc & fellow grants,
Flemish Government: **FWO**: PhD/postdoc grants, projects, G.0407.02 (support vector machines), G.0197.02 (power islands), G.0141.03 (identification and cryptography), G.0491.03 (control for intensive care glycemia), G.0120.03 (QIT), G.0452.04 (new quantum algorithms), G.0499.04 (statistics), G.0211.05 (Nonlinear), research communities (ICCoS, ANMMM, MLDM),
IWT: PhD Grants, GBOU (McKnow),
Belgian Federal Science Policy Office: IUAP P5/22 ('Dynamical Systems and Control: Computation, Identification and Modelling', 2002-2006), PODO-II (CP/40: TMS and Sustainability),
EU: FP5-Quprodis, ERNSI,
Contract Research/agreements: ISMC/IPCOS, Data4s,TML, Elia, LMS, Mastercard.

Please use the following BibTEX entry when referring to this document:

```
@techreport{MAERIVOET:06,

   author      =  "Sven Maerivoet and Bart De Moor",
   title       =  "Dynamic Traffic Assignment based on Cellular Automata",
   institution =  "Katholieke Universiteit Leuven"
   number      =  "06-058"
   year        =  "2006",
   month       =  mar

}
```

Dynamic Traffic Assignment based on Cellular Automata

Sven Maerivoet* and Bart De Moor
Department of Electrical Engineering ESAT-SCD (SISTA)†,
Katholieke Universiteit Leuven
Kasteelpark Arenberg 10, 3001 Leuven, Belgium

(Dated: March 31, 2006)

Within the framework of transportation demand modelling, we are left with three major approaches, being trip-based, activity-based, and equilibrium-based as put forward by Boyce. This diversification in the scientific field is a clear sign that different techniques are considered, based on distinct ideas. All techniques nevertheless borrow certain elements from one another, implying some generality between the models. As such, a travel forecasting model will most certainly be a give-and-take between requirements/desires and the current state-of-the-art [21]. Looking at the structure behind these methodologies, it is known that a core component in each of them is the concept of traffic assignment [23]. In this paper, we propose a method for performing dynamic traffic assignment, whereby we integrate departure time choice (leading to the phenomenon of peak spreading) and dynamic route choice, coupled with a dynamic network loading model. The method is built around a traffic flow model that is represented as a computationally efficient cellular automaton. The paper ends with a brief overview of some possible applications.

PACS numbers: 02.50.-r,89.40.-a

Keywords: dynamic traffic assignment, departure time choice, dynamic route choice, dynamic network loading

Contents

† Phone: +32 (0) 16 32 17 09 Fax: +32 (0) 16 32 19 70
URL: http://www.esat.kuleuven.be/scd
*Electronic address: sven.maerivoet@esat.kuleuven.be

Within the framework of transportation demand modelling, we are left with three major approaches, being trip-based, activity-based, and equilibrium-based as put forward by Boyce. This diversification in the scientific field is a clear sign that different techniques are considered, based on distinct ideas. All techniques nevertheless borrow certain elements from one another, implying some generality between the models. As such, a travel forecasting model will most certainly be a give-and-take between requirements/desires and the current state-of-the-art [21].

Looking at the structure behind these methodologies, it is known that a core component in each of them is the concept of traffic assignment [23]. In this part of the dissertation, we propose a method for performing dynamic traffic assignment, whereby we integrate departure time choice (leading to the phenomenon of peak spreading) and dynamic route choice, coupled with a dynamic network loading model. The method is built around a traffic flow model that is represented as a computationally efficient cellular automaton. The chapter ends with a brief overview of some possible applications.

I. INTEGRATED DYNAMIC TRAFFIC ASSIGNMENT

As already mentioned, the four step model of transportation demand modelling, contains a step called traffic assignment, in which all traffic demand is assigned to the network: the routes vehicles will follow are calculated, such that the load on the road network is evenly distributed over all links. This distribution is governed by Wardrop's criteria, e.g., the user equilibrium W1 which states that *"the journey times on all the routes actually used are equal, and less than those which would be experienced by a single vehicle on any unused route"*.

With respect to the assignment of vehicles to the available routes, there are mainly two different approaches that can be followed: static (STA) or dynamic traffic assignment (DTA). When discussing the DTA approach, we hitherto mainly focused on the route choice and dynamic network loading components. In this section, we introduce *integrated dynamic traffic assignment*, by which we mean that our modelling framework consists of three components:

- departure time choice (DTC),
- dynamic route choice (DRC),
- and dynamic network loading (DNL).

The first part of this section discusses two approaches towards DTA, namely analytical DTA and simulation-based DTA. In a second part, we explain our modelling framework that encapsulates the previously mentioned three components. The DNL component is expanded upon separately in section II.

A. Approaches to dynamic traffic assignment

It is important to capture the temporal character of congestion: the buildup and dissolution play an important role. As such, travel times depend on the history of the system, which should not be neglected. The 'dynamic' part in DTA refers to this dependency, implying that two fundamental components are considered, i.e., *route choice* and *dynamic network loading*. The former calculates the routes that vehicles take, by assuming that an equilibrium condition holds; the latter is responsible for loading all the vehicles onto the network, by explicitly simulating the physical propagation of the time-varying traffic flows[98]. The last two decades, a third fundamental component is considered, i.e., *departure time choice*, which is typically associated with the route choice behaviour: the choice of which route to take now become time-dependent.

In this section, we briefly describe the two main approaches towards DTA, being analytical DTA and simulation-based DTA. Note that some of the models described in this overview constitute more than a simple DTA procedure; they are embodied as complete travel forecasting models.

1. Analytical dynamic traffic assignment

The attractiveness of analytical DTA is that it can build on a long line of previous research, starting in the 1950s with the work of Beckmann et al. [12]. According to the overview of Joksimovic and Bliemer, we can distinguish between three categories of analytical DTA models[99] [42]:

- **Mathematical programming**
  The root of this technique can be traced back to the seminal work of Beckmann et al. in 1955, who reformulated the Wardrop equilibrium as a convex optimisation problem [12]. Closely related to this work, is the approach taken in static traffic assignment; more information can be found in the work of Patriksson [75]. A major drawback of mathematical programming, is that it is not able to fully capture the time-varying interactions of traffic flows and travel times.

- **Optimal control theory**
  Around 1989, a number of researchers proposed to describe the temporal evolution of traffic flows in the framework of optimal control theory (OCT), which deals with dynamic systems. Due to some of the severe limitations of this approach, the OCT research in the context of traffic flow modelling has not received much attention nowadays. For an overview of the OTC approach, we refer the reader to the work of Peeta and Ziliaskopoulos [77].

- **Variational inequalities**
  The most promising approach to tackle analytical DTA models, is based on a variational inequality (VI), which allows a natural incorporation of flow propagation constraints that explicitly contain link travel times. From the early nineties on, the VI technique has been applied to the DTA problem, fuelled by its ability to include certain measures in the models, e.g., the integration of road pricing policies. More information can be found in the works of Nagurney [74] and Patriksson [76].

  Some recent examples of this class of analytical DTA models include the work of Lo and Szeto who propose a formulation that is based on a dynamic user equilibrium and an encapsulation of the cell-transmission model [54, 55], the work of Bliemer et al. who propose a multi-class DTA model (i.e., different vehicle types) called *INteractive DYnamic traffic assignment* (INDY) [18–20, 64], and the work of Liu et al. who introduce a probabilistic framework for travel times with random perception errors [53].

Three of the more appealing aspects of this modelling class, are (i) the fact that due to their analytical nature, the solutions can be readily evaluated, (ii) the existence of well-defined algorithms that can lead to stable convergence, and (iii) their construction implies that computational complexity does not increase with a raise in traffic demand. A drawback however, is the fact that with respect to the current state-of-the-art, these analytical DTA models are not capable of handling large-scale road networks due to the computational complexity of the models themselves. Notwithstanding this critique, the aforementioned work by Bliemer et al. seems promising in this direction [18, 20].

2. Simulation-based dynamic traffic assignment

In order to address the problems associated with the application of analytical DTA techniques to large-scale road networks, simulation-based DTA can provide a solution. The iterative approach taken by this model class can largely be stated as follows:

1. The traffic demand is specified to a dynamic route choice model.
2. Based on the current travel times, the dynamic route choice model is executed, specifying the routes for all vehicles.
3. The dynamic network loading model is executed (note that the DNL model can be micro-, meso-, or macroscopic in nature).
4. The experienced travel times on all routes are extracted from the last simulation, and fed back to the route choice model in step 2. The iterations terminate when the algorithm converges.

Whereas the analytical expression of a DNL model is in general too cumbersome to include detailed traffic operations such as traffic lights, certain control measures, ..., the simulation-based DTA methodology can easily incorporate these effects (e.g., in a microscopic model). And although it can deal with large networks, there exists a subtle but major caveat, i.e., it uses a heuristic approach with respect to convergence. To state it more clearly, *convergence is not guaranteed*. An often employed method is a *relaxation procedure* that performs iterations until all travel times in the road network are stationary [8, 33–35]. In this context, because the concept of mathematical convergence is too strict, researchers then refer

to the terminology of ‘relaxation’. Note that the question as to whether or not the simulation relaxes, or when it does, if it then relaxes to a unique equilibrium or instead exhibits oscillations, still remains an open debate; in some worst cases, even a gridlock situation can occur [69, 71, 82].

Some examples of simulation-based DTA models include the following:

- Microscopic models: AIMSUN2 [7, 8], MITSIM [97], Paramics [25, 52], TRANSIMS [73], VISSIM [79].
- Mesoscopic models: CONTRAM [90], DynaMIT [15, 16, 88], DynaSMART [66], METROPOLIS [29, 65].
- Macroscopic models: METANET [68].

B. Integrated dynamic traffic assignment

As already highlighted in the introduction of section I, our transportation demand modelling framework encapsulates three components, i.e., departure time choice (DTC), dynamic route choice (DRC), and dynamic network loading (DNL). In the first part of this section, we give an overview of the framework, the second part then goes into more detail with respect to the modelling of traffic demand, after which the following two parts describe the DTC and DRC components. The section ends with some remarks on convergence criteria.

With respect to the issue of solving the combined DTC-DRC problem, we briefly mention some of the existing research approaches in literature: Yang and Meng consider optimal pricing strategies (i.e., a congestion toll) by solving a system optimisation problem that combines the DTC and DRC models, as well as the optimal tolls of bottlenecks [96], de Palma and Marchal present the METROPOLIS toolbox who directly use the DTC model to assign a static OD matrix to a commuter period [29, 65], Ettema et al. describe a microscopic traffic flow model in which drivers base their departure time and route choice decisions on a mental model of the traffic conditions (i.e., the mean and variances of the experienced travel times) [31], Lago and Daganzo describe a departure-time user equilibrium model, combining Vickrey’s DTC work and the LWR first-order macroscopic model [48, 49], Szeto and Lo propose a formulation based on the cell-transmission model through a variational inequality problem [89], and Lim and Heydecker identify a new equilibrium condition for the combined DTC-DRC problem, providing a computationally efficient solution algorithm that is based on the concept of reasonable paths [51].

1. *Overview of the framework*

Our approach towards dynamic traffic assignment is simulation-based; it combines a departure time choice (DTC) model with a dynamic route choice (DRC) model, and incorporates a dynamic network loading (DNL) model based on an efficient traffic cellular automaton. A flow chart of the framework is given in Fig. 1.

In general, we can describe our modelling technique as follows:

1. Based on the information contained in a static OD matrix that is valid for the time period under consideration (e.g., the morning rush hour), we create a population of N agents by disaggregating the entire OD matrix.
2. Based on the description of the road network, and on the information from the static OD matrix, we generate a set of routes that connect the origins and the destinations. This set actually denotes *feasible routes*, as not all paths between an origin and a destination should be considered (doing this selection beforehand avoids extra complications in the DRC model).
3. All agents’ departure times are re-calculated according to the DTC model (they are uniformly distributed over the entire time period, before the first iteration, thus corresponding to the assumption of a constant flow).
4. Using the information contained in the route set, and the current experienced travel times on the links in the road network, all agents get a route assigned, corresponding to their current shortest path in the network (shortest in the sense of minimal cost expressed as the travel time).
5. All agents are then put into queues located at the origins of the road network, after which the DNL model is executed until all of them have reached their destination. Note that advanced traveller information systems (ATIS) can influence the route choice model, as well as the DNL model; ATIS can also draw information from the last run of the DNL model. Furthermore, event-based incidents leading to, e.g., lane closures can easily be included in the DNL model.

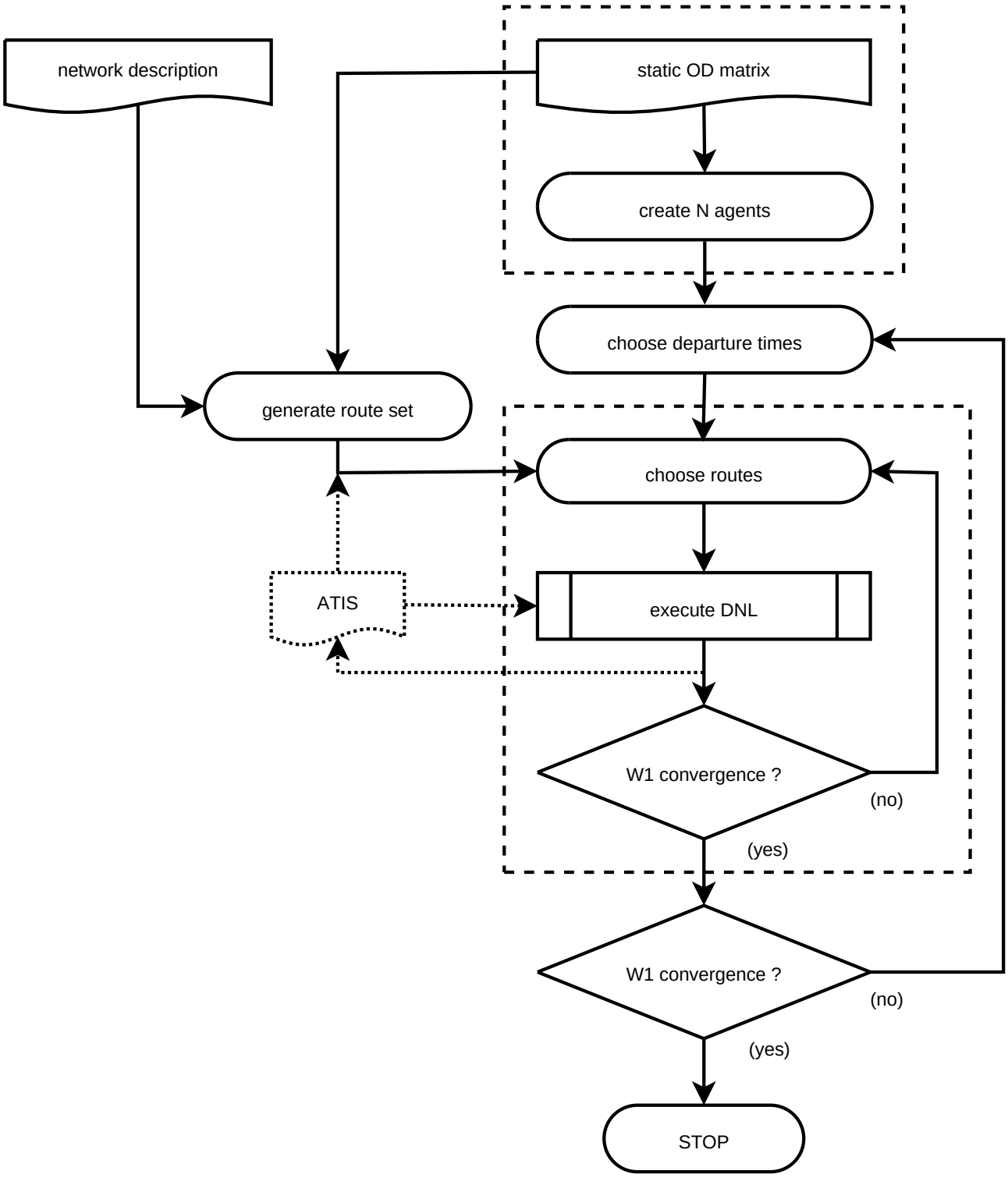

FIG. 1: A flow chart of our framework for simulation-based dynamic traffic assignment (DTA); it combines a departure time choice (DTC) model with a dynamic route choice (DRC) model, and incorporates a dynamic network loading (DNL) model based on an efficient traffic cellular automaton. The part containing the module for advanced traveller information systems (ATIS) is optional. The upper dashed block represents the disaggregation of the traffic demand into individual agents (i.e., commuters), the lower dashed block denotes the DRC model.

6. A convergence criterion is checked, corresponding to a *W1 user equilibrium*; the associated cost for each agent is its total travel time. If convergence is not reached, we return to step 4 and recalculate the new shortest routes, based on the last experienced total travel times after execution of the DNL (their departure times remain fixed during subsequent iterations).

7. If the previous step resulted in sufficient convergence with respect to the routing of the vehicles, another convergence

criterion is checked, but this time with respect to the agents' departure times. In this step, schedule delay costs are taken into account for all agents, thereby including effects of arriving too early or too late at their destinations. If the algorithm does not converge, we return to step 3.

Note that as opposed to analytical DTA models for the combined DTC and DRC problem, we adopt two different convergence criteria in this framework. In the former they are simultaneously combined, whereas we work with a *sequential procedure*[100]. The modified *W1 user equilibrium* as used in step 7 when checking the convergence, is based on a generalised travel cost. It corresponds to the following formulation: "*The generalised travel costs for all agents are equal and less than those which would be experienced by a single agent departuring at a different time.*"

In the following sections, we discuss the modelling of traffic demand, after which we describe the DTC and DRC components, concluding with some remarks on the convergence of simulation-based DTA.

2. Traffic demand generation

In most travel forecasting models, traffic demand is typically expressed as one or more origin-destination (OD) matrices. Within the classic four step model, it is assumed that steps (I) to (III) result in an OD matrix that is fed to the fourth step, i.e., the traffic assignment which calculates the routes vehicles will take.

The estimation of these OD matrices, forms a theme on its own; a typically encountered problem is the fact that due to the large amount of unknown variables (it is a considerably under determined system of equations), additional constraints need to be introduced. Besides the already existing OD matrix estimation techniques, we also mention some other methodologies, e.g., the doctoral dissertation of Bierlaire, which provides a nice overview of several different OD matrix estimation techniques [17], Abrahamsson who gives a detailed literature survey of OD estimation based on traffic counts [1], and the work of Balakrishna who provides a detailed methodology for the joint calibration of OD matrix estimation and route choice models within the DynaSMART model [3, 66].

Another approach for traffic demand specification builds upon the activity-based modelling approach. Examples in this direction include the work of Balmer et al. who develop a methodology for creating individual demand (i.e., tailored towards individual commuters) out of a general specification; they allow for the construction of agents' plans out of general OD matrix data [4, 6], the work of McNally who integrated household activities, land-use distributions, regional demographics, ... into a microscopic model for traffic demand forecasting [67]. A recent approach closely related to ours, is the one by Kemper who disaggregates one static OD matrix into individual commuters, whilst taking into account traffic flow profiles over time. However, his model currently does not support departure time choice [44, 45].

Within our framework, we start from a known *single static OD matrix* that captures an entire time period (e.g., the morning rush hour). We furthermore assume *unimodal traffic* in the network, but we allow for a distinction between cars and trucks. From this single OD matrix we create a population of N agents by disaggregation (note that by 'plans' we mean an agent's route end points, but not the actual path of the route taken). As can be seen in Fig. 2, our approach also borrows ideas from the activity-based modelling approach, such that — as opposed to the paradigm of a single static OD matrix — it is flexible enough to incorporate historical OD matrices, an explicit synthetic population of households, land-use data, ... Note that if necessary, it is possible to recreate a single OD matrix from the generated agents.

3. Departure time choice (DTC)

Within our framework we explicitly provide *time-of-day modelling* in the form of a departure time choice model. We assume the single static OD matrix specifies the traffic demand over a given time period $[t_{\text{demand,start}}, t_{\text{demand,end}}]$, e.g., between 07:00 and 09:00. As such, it represents the total number of commuters that *want to depart* during the specified time period. For this latter time period, it is important that it completely encapsulates the period of heavy congestion (otherwise, some exogeneously given boundary conditions are needed); it is also preferred that all commuters can reach their destinations within the time period.

At the first iteration of the DTC model's execution, all agents' departure times are uniformly distributed over the entire time period (we thus assume a constant flow); the departure time for the i^{th} agent is given as:

$$t_{\text{departure},i} = t_{\text{demand,start}} + (i-1)\,\frac{t_{\text{demand,end}} - t_{\text{demand,start}}}{N}. \tag{1}$$

Once the DNL model has been executed and the agents have reached their destinations (assuming an equilibrium situation was achieved according to step 6 of the framework in section I B 1), we can calculate the general costs experienced by each

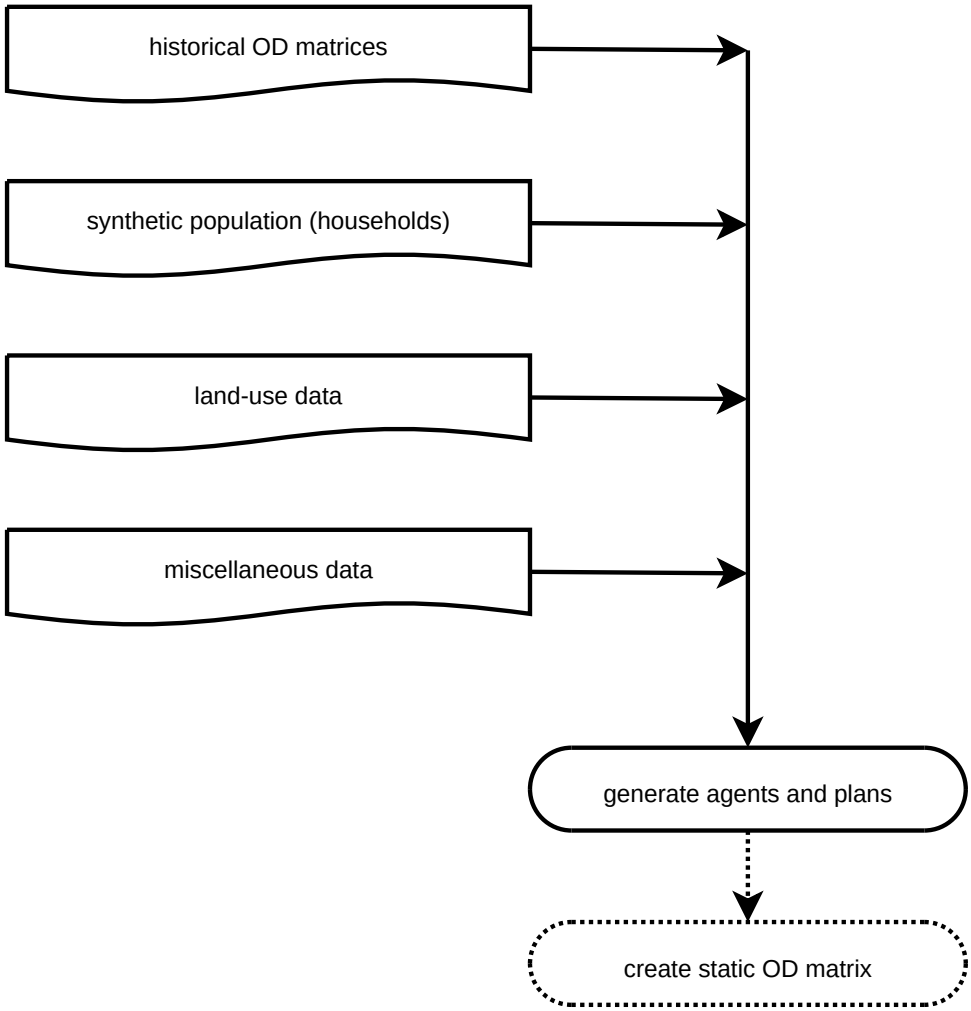

FIG. 2: An overview of our flexible approach that borrows ideas from activity-based modelling. As opposed to the paradigm of a single static OD matrix, it can incorporate historical OD matrices, an explicit synthetic population of households, land-use data, ... If necessary, it is possible to recreate a single OD matrix from the generated agents.

individual traveller. At this point, we know the departure times $t_{\text{departure},i}$ of all the agents, as well as their waiting times $\mu_i(t_{\text{departure},i})$ in the queues at the origins, and their travel times $T_i(t_{\text{departure},i})$. With the latter two of these quantities, we associate certain costs, resulting in $C_{\mu,i}$ and $C_{T,i}$. With respect to the departure time, we incorporate so-called *schedule delay costs* $C_{\text{sd},i}$, as used in Vickrey's bottleneck model. In this model[101], each agent is assumed to have a certain *preferred arrival time* (PAT) $t_{\text{PAT},i}$; extra costs are incurred if the agent arrives too early or too late [93]. Combining all these costs, the *generalised travel cost* for each agent is then defined as:

$$\begin{aligned} C_{\text{total},i}(t_{\text{departure},i}) \;=\; & C_{\mu,i}(\mu_i(t_{\text{departure},i})) + \\ & C_{T,i}(T_i(t_{\text{departure},i})) + \\ & \max\{C_{\beta,i}(t_{\text{PAT},i} - (t_{\text{departure},i} + T_i(t_{\text{departure},i}))), 0\} + \\ & \max\{C_{\gamma,i}(t_{\text{departure},i} + T_i(t_{\text{departure},i}) - t_{\text{PAT},i}), 0\}, \end{aligned} \tag{2}$$

with $C_{\beta,i} < C_{T,i} < C_{\gamma,i}$ the costs for arriving too early or too late, respectively[102]; as such, arriving too late carries a higher weight than arriving too early.

Note that it is also possible to specify a cost associated with an agent's departure time; this cost can for instance be represented as a *utility function* $U_{\text{departure}}(t)$, which assigns a 'score' to each possible time instant for departure (e.g., some people don't want to leave too early). As such, the range of the time period in which a commuter wishes to depart is demarcated, such that utility maximisation of the individual results in a chosen departure time [11].

Our DTC model then selects the 10% agents with the highest costs and shifts their departure times towards those associated with the lowest costs (we reserve an *indifference band* for small shifts on the order of five minutes). In terms of Wardrop,

this corresponds to a *W1 user equilibrium*, but now with respect to the agents' departure times: "*The generalised travel costs for all agents are equal and less than those which would be experienced by a single agent departuring at a different time.*" At this step, it is important to take into account certain constraints, e.g., placing a restriction on the maximum number of agents that can depart in a certain time period (as the absolute inflow into the network is bounded by the capacity at that point).

Let us finally mention that there are other approaches towards integrated DTC modelling, e.g., the model of Levinson and Kumar, which constitutes an implementation of the four step model. They extend it with a DTC model that is based on a binomial logit model, whereby commuters can choose between travelling in the peak hour or in the shoulder hours of the peak period (they use a static user equilibrium method for solving the DRC component) [50]. Discrete choice modelling is a popular approach for dealing with these kinds of decisions in a DTA context [13, 14]. All techniques have one thing in common: when the projected demand exceeds the capacity, this will lead to the phenomenon of *peak spreading*. Commuters change their departure times in order to accomodate the excess demand during the assignment procedure.

Note that the previously discussed algorithm is based on a *single static OD matrix* that encapsulates, e.g., the morning rush hour. Another approach would be to use multiple exogeneously given *time-dependent/dynamic OD matrices*. Each of these latter non-overlapping matrices covers, e.g., one half hour. As the DNL simulation is executed, all traffic demand matrices are to be consecutively assigned to the network. When performing this step, agents' departure times can be shifted *within* one matrix's time period, but also *between* subsequent matrices' time periods. In this case, it is important to put a constraint on the maximum number of vehicles transferred between such OD periods; e.g., it is impossible to allow a million agents to start at the same time instant at a single origin.

4. Dynamic route choice (DRC)

With respect to the routing of vehicles through the network, there are two approaches possible, i.e.:

- **Pre-route assignment**
  This method is also known as *equilibrium assignment*, because each driver now tends to minimise his travel cost; it is built on the assumption that travellers have *perfect information* of the *experienced travel times* (corresponding to *day-to-day learning* of the traffic pattern on a typical day, given the current traffic demand [26]), such that their travel time costs are fixed and known at the time of departure. Due to its equilibrium nature, iterations are required until convergence is met.
- **En-route assignment**
  As opposed to the former method, this one is based on the *instantaneous travel times*, and allows vehicles to change their routes as they proceed through the road network (this is what is meant by 'dynamic' route choice). Note that this approach requires only one simulation run of the DNL, whereby vehicles can change their route at each junction they encounter in the road network. Within this method, it is quite straightforward to incorporate route guidance; the caveat however is that each vehicle should have a default route, i.e., pre-trip assignment, in case no attention is given to route guidance.

Because in general there exists more than one route between an origin and destination pair, commuters have to select which route they will take. We use the following approach towards route choice:

1. At the first iteration of the DRC model's execution, we calculate an initial travel time for each link in the road network; this travel time is defined as the length of the link divided by its free-flow speed. In subsequent iterations, we define the travel time of a link as the arithmetic average of the travel times of all the vehicles traversing the link.
2. Based on the current travel times of all the links, shortest paths are calculated for each OD-pair, using, e.g., Dijkstra's algorithm [30], the heuristic shortest-path algorithms of Jacob et al. [41], or the efficient algorithm of Rosswog et al. which is based on tree heuristics that uses the hierarchical information of a road network [84], and taking into account the set of predefined feasible routes $\mathcal{R}_{\text{pre}}$ (see step 2 in section I B 1).
3. All vehicles now select a route from this set, according to the following multinomial logit model[103], which gives the probability of a given alternative route i at departure time $t_{\text{departure}}$ as:

$$p_i(t_{\text{departure}}) = \frac{e^{\mu(-U_i(t_{\text{departure}})+\epsilon_i)}}{\sum_{j\in\mathcal{R}} e^{\mu(-U_j(t_{\text{departure}})+\epsilon_j)}}, \tag{3}$$

in which $U_i(t_{\text{departure}})$ is the utility of route i at the specified departure time, $\mathcal{R} \subseteq \mathcal{R}_{\text{pre}}$ is the set of all available routes between a vehicle's origin and destination pair, μ is a dispersion factor and ϵ_i is a stochastic error term.

5. *Some remarks on the convergence of simulation-based DTA*

As already mentioned in section I A 2, convergence is in general not a guarantee in simulation-based dynamic traffic assignment. Issues such as existence and uniqueness of a well-defined equilibrium, as well as stability of the solution, still remain an open question. Due to the stochastic nature of the simulations, convergence can be hindered in the case of oscillations, or even a gridlock situation.

The primary technique adopted in this case, is called *heuristic relaxation*: a pre-defined performance measure is compared for subsequent iterations, until the difference is no longer significant. Two possible performance measures are the following:

- the total travel time in the system, i.e., the sum of the travel times of all the vehicles at the end of the simulation,
- or the combined variances of the travel times of all links separately, or the variance of the total travel time in the system.

Note that it is advisable to set a pre-defined upper limit on the number of feedback iterations in the framework depicted in section I B 1 (this might indicate a failure to converge). Finally note that this construction *assumes* the existence of an equilibrium; it is however *not a necessity*, as the algoritm will stop anyway after a finite number of iterations.

II. AN EFFICIENT DYNAMIC NETWORK LOADING MODEL (DNL)

Today, a main challenge is the construction of macroscopic and microscopic models that lend themselves to a faithful representation of road traffic, as these models are used in several key aspects in the control of traffic flows.

Within this context, our research is aimed at assisting traffic engineers who wish to evaluate what-if scenarios and/or perform real-time control of traffic flows. Whereas the former requires a sufficiently detailed model, the latter calls for an efficient implementation that allows fast simulations. The challenge thus consists of the development of a flexible testbed environment that is capable of providing us with a detailed simulation model of a real-world road network, not containing too many parameters that require extensive calibration: the TCA models fit this description nicely.

Although these TCA models allow for fast computations, they are nevertheless computationally very expensive because they are based on behavioural models that need to be applied to each vehicle at each timestep (i.e., the car-following and lane-changing models). We thus need to find the most optimal solution in terms of time and space complexity. A logical step in this direction, is an efficient parallellisation scheme that lowers the computational overhead involved. This can be accomplished by using distributed computing, where we partition the road network in several distinct geographical regions that are assigned to different machines which run in parallel.

We automatically gain platform independency using Java™ . The challenge now is to get reliable and efficient (i.e., faster than real-time) operation of a very heterogeneous computing environment. To this end, the simulator consists of one master, controlling several different workers that efficiently simulate local traffic flows.

In this section, we first give an introduction that describes traffic flow simulation from a historical perspective, paying attention to the role of open-source software development. We then give a functional description of our DNL model, called *Cellular Automata Traffic SIMulation* (CATSIM). This is followed by some implementation details of the code, after which we discuss our approach towards an efficiency increase through the paradigm of distributed computing [59, 62].

A. Development of traffic flow simulators

In this section, we first give a brief overview of the development of traffic flow simulators, looked at from the perspective of the programming languages involved and the computational complexity of the models. The second part of this section considers the effects of developing software under an open-source flag. It is worthwhile to take a look at this aspect, as most of the traffic flow models tend to be developed in an accessible academic setting, but once commercialisation 'kicks in', the model's internals tend to get shrouded in legalese. In our discussion, we contrast open-source development with the classic approach of non-disclosure of the software's internals, give pointers to some of the existing licences that can regulate the commercial and non-commercial use of this type of software, and finally conclude with a note on legal issues related to intellectual property rights, the patenting of ideas, inventions, and algorithms.

1. Traffic simulation from a historical perspective

Traffic flow simulators have come a long way since their inception in the early fifties (see, e.g., the *TRAffic Network Simulator* – TRANS, which shows a remarkable parallel with early traffic cellular automata models [43]). In those days, computers operated in both an *analogue* and *digital* fashion. However, as the former became more expensive when larger systems were simulated, the latter gained a strong foothold in the simulation community [38]. Nowadays, as desktop computers get smaller and more powerful, the traffic flow simulation software has undergone a drastic evolution. It is implemented in either procedural languages (e.g., C, FORTRAN, ...) or object-oriented ones (mainly C++ and Java™), with the latest trend to employ script-based languages (e.g., Ruby) *within* the environment of a simulator itself (see for example the OmniTRANS project [92]). The simulators are applied to moderately sized transportation networks, whilst still allowing a rather detailed view on traffic operations.

Another evolution that is noticeable, is the upcoming market of complete travel forecasting models that are based on dynamic traffic flow models. This class of software applications has features such as complete GUIs, fully integrated travel demand modelling, calculation of measures of effectiveness (e.g., noise and pollutant emissions), ... Examples of such full-fledged models are TransCAD [24], OmniTRANS [92], DynaSMART [66], ... In many cases, the DNL core of these models is formed by a mesoscopic or macroscopic model, but there is an evolving trend towards more realistically microscopic models. However, noting the current state-of-the-practice in the field of traffic flow engineering, we note that it is becoming more and more appealing to move from static paradigms towards the use of fully integrated DTA models on a commercial basis [60].

In our view, it is not necessary to get all the dynamics correct on a detailed microscopic level. As such, TCA models can offer a certain degree of detail, while retaining computational performance and remaining comparable to their mesoscopic/macroscopic counterparts. One of the main advantages of the TCA modelling paradigm is that it does not require many parameters, as opposed to other microscopic traffic flow models in which the plethora of parameters and features clouds a clear understanding of the models' dynamic properties.

2. The benefits of software development under an open-source flag

Despite the fact that most of the traffic flow models tend to be developed in an accessible academic setting, the traditional approach towards the creation of the majority of ready-to-use software is mainly oriented towards its commercialisation. As a consequence, it is beneficial from a marketing perspective to provide prospective customers with complete packages that integrate transportation planning models, e.g., the four-step model.

In many cases, the main stream company policy is aimed towards the non-disclosure of the models' internals, effectively reducing these commercial packages to advanced versions of black-box models. When such software starts to grow more mature and complex, it becomes increasingly difficulty to answer the question *"What is really under the hood ?"* The importance of this statement should not be underestimated, as it is vital for transportation engineers to be acquainted with a model's inner workings, features, and limitations, when interpreting results for, e.g., policy decisions.

This lack of openness, can be remedied by developing the software under an open-source flag. From this point on, the complete underlying model structure remains revealed at all times, as it is now possible for many programmers to read, redistribute, and modify the source code. When a company exhibits this sagacity, the unlocked potential of open source can be fully brought into play. One of the main benefits of this paradigm is that there are effectively 'many eyes looking at one single problem'. As a direct result, the debugging, maintenance, and support life cycles of such software become more transparent, as opposed to the monolithic approach typically encountered in propriety software [80]. If such an open-source project is properly managed (which implicitly requires skilled people), it can receive an increased gain from the feedback of its user base. Already, several successful examples of this type of software development can be found in real life, e.g., the Linux operating system, the Netscape and Mozilla web browsers, the StarOffice suite and OpenOffice.org project, ... Within the traffic community, the open-source approach is slowly starting to pick up, for instance with a prime example such as the *Simulation of Urban MObility*[104] (SUMO) [46].

When releasing open-source software, there literally exists a myriad of licences that regulate the commercial and non-commercial use of this type of software, as well as its incorporation in third-party software. Archetypical examples are the Free Software Foundation's GNU General Public Licence[105] (GPL) with the popular catch phrase *"free as in free speech, not as in free beer"*, the Open-Source Initiative[106] (OSI) which provides a marketing vessel for 'selling' free software, the Creative Commons Licences[107] (CCL) that offer a flexible copyright for creative work, ...

Finally, note that in our discussion, we did not state anything about legal issues such as the management of intellectual property rights, issues related to the patenting of ideas, inventions, and algorithms, et cetera. Indeed, most licences undoubtedly steer clear of these topics, allowing their interpretation to remain up to the developer and/or the company. However, the central core that forms the business model for open-source software, is to freely share the software, whilst

selling support. With respect to academic institutions and their management of intellectual property, dissemination of algorithms by means of publications in journals might be discouraged. In these cases, we still deem it appropriate to publish the results, as we believe that the money remains in the selling of the software. Another less-commercial track that can be followed, is to release the software as a web service, thus effectively hiding the underlying code of an algorithm's implementation.

B. Functional description of the simulator

Considering the CATSIM DNL model from a functional point of view, this section first gives a description of the topological and geographical structure of the road network, after which we explain some vehicle-related information, ending with what kinds of statistical data can be collected during a simulation run.

1. Topological and geographical structure of the road network

In CATSIM we opt for an intuitive structure, whereby the network is topologically decomposed into *nodes* and *edges*. Geographically, these graph characteristics correspond to *nodes* and *links*. Different link types may exist (such as on-ramps, off-ramps, merging areas, ...). For reasons of efficiency, we define each link to consist of one or more undivisible road segments. All links are connected by special *junction nodes*, where vehicles are transferred from one link to another (more than one link can enter or exit a junction node). The intermediate nodes connecting the different segments of a link are called *bend nodes* (note that the entire road section containing all the segments is represented with just one single CA lattice). They allow for a more realistic modelling of the road network. Note that the specification of a node requires *X*, *Y* and *Z coordinates*, thus we take road gradients (e.g., elevations, tunnels, ...) explicitly into account (although it is up to the car-following model to actually use this information). With these elementary building blocks, the motorway network can easily be constructed using data provided by satellite images and/or geographical information systems (GIS).

It should be stated that in our current specification, there is no definition of what the actual underlying low-level TCA models are. They may even vary from link to link if necessary, giving a flexible and open architecture. Notwithstanding this freedom, they do have to agree 'functionally', e.g., lane changes should either be mandatory or discretionary, time steps should be comparable, ...

2. Vehicle-related information

As heterogenity of a traffic stream is a necessity for a rich dynamical behaviour, we allow for different classes of vehicles. This includes cars and trucks, with trucks occupying extra cells. A convenient method for representating this difference, is using passenger car units. Note that the rule sets do not model a vehicle as explicitly occupying more than one cell, but instead adjust the safe space headway to account for the difference in vehicle length.

Furthermore, as opposed to most other implementations, our cells do not just contain a number indicating the presence and/or speed of a vehicle. Instead, we allow for complete objects to be contained in the cells, e.g., a vehicle with a commuter's personal routing plan. Because most interactions of the vehicles are based on local information, we add another subtle refinement: information such as link travel times for example, can be put in a *central data storage* that is available to the network simulator. This means that some vehicles can be considered as *'informed drivers'* having access to this data storage, and are thereby able to reroute their trip in order to avoid encountering network congestion.

3. Collecting statistical data

The simulator's road network can be equipped with artificial loop detectors. They accurately compute various statistics from the passing traffic flow, continuously storing all results in the central data storage. Even travel times recorded by probe vehicles can be contained, such that this information becomes available to some of the vehicles (i.e., the informed drivers) as they travel through the network.

C. Code implementation details

Beside the functional description of the previous section, we now shed some light on our proposed choice of programming language for implementing a traffic flow simulator. Afterwards, we illustrate some technical aspects related to the implementation of CA lattices, as well as some details regarding the implementation of links in a road network.

1. Choice of programming language

Whereas earlier designs of traffic flow simulators were based on procedural languages (e.g., pure C code), we nowadays observe a trend towards the adoption of object-oriented programming languages. In this spirit, we propose to use the Java™ language, as it has been designed around a *"write once, run anywhere"* (WORE) philosophy, implying cross-platform portability without needing any recompilation of the code base.

2. Some technical aspects related to the implementation of CAs

Classical implementations of CA models were typically aimed at obtaining a high computational speed. This led to the use of techniques, e.g., single-bit coding schemes, typically targeted towards specific hardware platforms. The coming of popular object-oriented programming languages such as C++ and Java™ , coupled with the steady increase of computational power in average desktop computers, makes the original line of work a bit outdated.

As with respect to the implementation of a CA's grid itself, there are two approaches possible:

- **site oriented**: this is typically based on *an array of cells*, which is more suited for links having *high densities*,
- **particle oriented**: this is typically based on *a linked list of vehicles*, which is more suited for links having *low densities*.

In practice, it is best to consider the best of both worlds, i.e., only relevant sites are updated. In this view, a site corresponds to a lateral section of a multi-lane link (i.e., all cells located at the same longitudinal position).

When exchanging vehicles between consecutive links at junction nodes, we use a *lane connectivity table* that contains the numbers of all outgoing and incoming lanes, each time in the local numbering scheme (the same holds for all intersection logic). The following table gives the connectivity for the example of a main road and on-ramp towards a merge section as depicted in Fig. 3:

$$\begin{pmatrix} 1a.1 & \rightarrow & 2.2 \\ 1a.2 & \rightarrow & 2.3 \\ 1b.1 & \rightarrow & 2.1 \end{pmatrix}$$

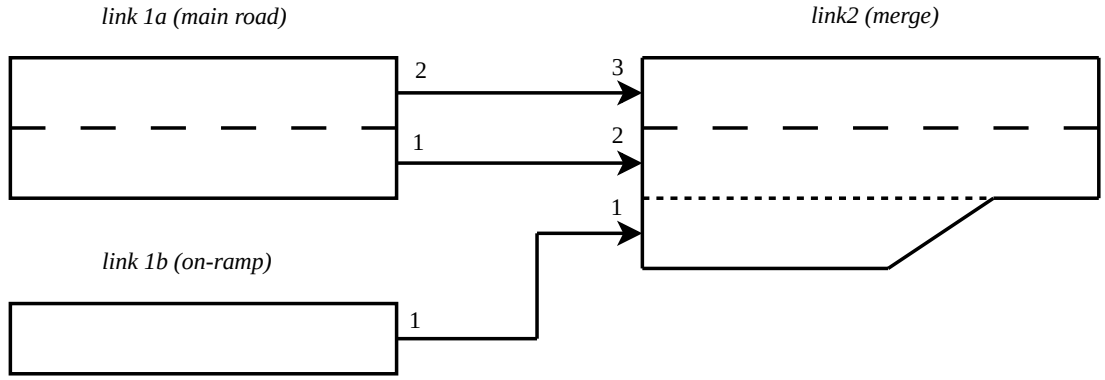

FIG. 3: A graphical sketch of the lane connectivity for a main road and on-ramp towards a merge section; vehicles are exchanged between consecutive links based on a lane connectivity table corresponding to the diagram.

Furthermore, in CATSIM, each link has both a car-following and lane-changing TCA model with corresponding parameter vectors. Separating the parameters from the models allows us to keep the latter while performing on-line adjustments to the former. Finally, the slowdown probabilities are a property of the links, not of individual vehicles.

D. Increasing efficiency through distributed computing

As already stated, using microscopic traffic simulators places a large computational burden on the employed machine architecture. Many existing simulators were initially designed to run on a single CPU. Only afterwards were they converted for parallel operations (e.g., AIMSUN2 [7, 8], TRANSIMS [70, 73], ...), with some exceptions such as PARAMICS which was designed in a parallel fashion from the ground up [25, 52].

The same train of thought holds for most of the traffic cellular automata models. In the beginning, when they were built using parallel implementations, the parallellisation scheme was strongly reflected in their codebase, relying heavily on the underlying machine architecture. Examples are models whose computations were performed on a large number of CPUs (e.g., 1024), all contained in one shared memory architecture, employing special techniques such as single-bit encoding et cetera [72].

In the recent past, we already developed a microscopic traffic simulator in Java™ , called *Mitrasim 2000*. Instead of being a true parallel implementation, it was based on a *client-server architecture* (CSA), in the sense that the simulator ran on one machine (the server); several different other machines (the clients) showed an animation of the traffic evolution on the motorway network [58]. A major problem was that, mainly due to the single CPU architecture, the simulator did not achieve real-time speed at all. However, our past experiences allow us to build a more efficient and scalable simulator, in which parallellism can be implemented through *distributed computing*.

In our framework, the concept of distributed computing implies that we no longer use a homogeneous environment of CPUs working in a shared memory architecture. Instead, a very *heterogeneous computing environment* is provided, like for example a Beowulf cluster [70]. Whereas supercomputers performed intensive tasks in the past, we can nowadays observe a shift towards grid-based computing [2]. For us, the challenge now is to get reliable and efficient (i.e., faster than real-time) operation of this latest networked architecture. In Fig. 4, we can see an example of distributing the load of the motorway network over a group of computing units.

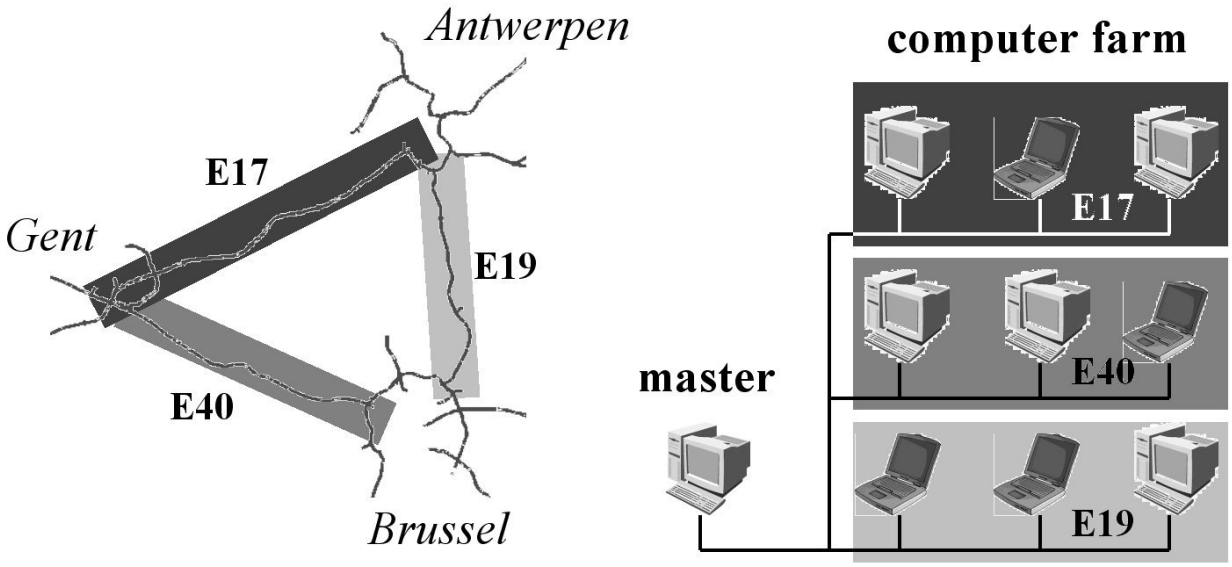

FIG. 4: The idea behind distributed computing in the CATSIM dynamic network loading model: one computer (the master) controls several workers in a heterogeneous computing environment. All these computing units work together, whereby the load of the entire road network is distributed. In the shown example, three major motorways are modelled whereby the responsibility of each motorway is assigned to several grouped workers.

The flexible functional description set out in section II B, allows several implementations. In this part, we present such a possible approach, in which parallellism is achieved using distributed computing. In the following sections, we first shed some light on the difference between high-throughput and high-performance computing. We then give a description of the technologies used with respect to direct communication between different processing units, as well as a method that provides us with a shared memory. In a subsequent section, we explain the adopted parallellisation scheme from a programmatorical and technical point of view. We end with a brief consideration of some issues related to synchronisation, graph cycles, and data sharing.

1. High-throughput versus high-performance computing

As desktop computers got increasingly more powerful during the last decade, the paradigm of distributed computing has started to gain serious importance. Within this concept, a distinction is made between two radically different methodologies:

- **High-throughput computing (HTC)**
  In this setting, software is installed in a heterogeneous computing environment, thereby distributing the processing power over different computing nodes (e.g., all desktop computers in a university's research group). Users can submit tasks, which are then optimally assigned to these nodes, taking into account priorities, waiting queues, performance, ... At each time, attention is given to the fact that a user can regain and keep control over all the processing power of his/her own machine, at which point the running task is scheduled and resumed at another free computing node. As a result, HTC offers a large degree of fault-tolerant computing power, available of long periods of time. Two examples of these kinds of environments are the *Condor* [91] and the *H2O* projects [47].
- **High-performance computing (HPC)**
  Another important aspect of distributed systems, is their ability to quickly execute certain tasks. As opposed to HTC, for which the speed of the requested computation is not per se a strict constraint but the availability of computing power on a large spatial scale is, HTC is centred around a close tie between computing power in space and time. Examples of applications in this direction are aspects such as ray tracing [57], and microscopic traffic flow simulation (see, e.g., the introduction of section II D).

2. Technologies used

As the whole simulator will be constructed for the Java™ Virtual Machine (JVM), we automatically achieve *cross-platform portability*; this is a necessity in order to efficiently address the heterogeneous computing environment.

Reliably controlling such a networked architecture requires a strict scheduling scheme: all processing nodes in the computing network are tightly coupled with each other. In this case, we have opted for a mixture of the *'master/worker'* and *'command' application patterns* [32]. This means that we have one master computer that controls N distinct worker computers who execute the different tasks.

Currently, most distributed implementations of traffic simulators use classic communication techniques such as a *Message Passing Interface* (MPI) [56] and/or *Parallel Virtual Machines* (PVM) [36]. In spite of this, we strongly believe that this is not sufficient for our envisioned architecture. There's a trade-off involved, between on the one hand efficient direct communication and on the other hand an accessible shared memory architecture:

- **Direct communication**
  It is very important that the computation/communication ratio remains as high as possible, after parallellising[108] the simulator [32]. Translating this to our Java™ implementation, we establish *dedicated communication channels* between the master and the workers and between the workers themselves; *sockets* provide a suitable and efficient method for this type of communication. Furthermore, all communication done between the workers, should be scheduled simultaneously with the computations they perform, so delay times can be minimized.
- **Shared memory**
  Because we are working in a heterogeneous computing environment, no standard shared memory architecture is directly available. There is however a nice solution to this: *JavaSpaces* provide a generic environment that can be accessed by any worker located anywhere in the network. It is based on the concept of *Linda Spaces* [37], implemented using Java™ 's *Remote Method Invocation* (RMI) facility [39], and provided as a service of the *Jini Network Technology* [32, 87]. A major advantage is that when using a JavaSpace, no explicit network addresses (e.g., IP addresses and TCP port numbers) need to be known when communicating. All communication can be done *anonymously* (loose coupling in space), and even *asynchronously* (loose coupling in time). Despite its flexibility, the JavaSpaces service is — as stated before — in essence a medium for loosely coupled communication, and thus not well suited for performant dedicated communications. Because of this, we only use it for setup purposes and infrequently accessed resources and information.

  Furthermore, it is important to keep the following two points in mind:

  – the underlying Jini technology is a network based protocol, that does not offer any real-time guarantees,
  – and JavaSpaces is semi-scalable: it runs in a single JVM, implying that the service itself might become a bottleneck for communication, or worse, it might fail. A solution to this problem can be to use multiple JavaSpaces that are clustered together.

After discussing the communication aspects and shared memory setup mentioned in the previous section, we now explain the adopted parallellisation scheme from both a programmatorical and technical point of view. We describe how the workers' tasks are set up and distributed. We then give details on how the dedicated communication channels are constructed, ending with some comments on the execution of a simulation step.

- **Setting up the workers' tasks**
  We assume that, at the beginning of the simulation, the master has all the information available about the road network infrastructure, the travel demand and routing plans, ... It then performs a *domain decomposition* on a geographical basis, dividing the network in exactly N partitions (the splitting of the links is preferably done far away enough from any junction nodes, such that we can avoid the complexities of intersection logic). Note that this encompasses static load balancing (see Fig. 4 for an example of such a decomposition); it is also possible to opt for another scheme, thereby providing us with some means to perform dynamic load balancing. This can, for example, be accomplished by keeping track of the workers' computation times and redividing and reassigning partitions.

- **Distributing the tasks**
  The next step consists of the master distributing the tasks (i.e., the different motorway stretches in each partition) into the JavaSpace. All the workers then check this JavaSpace and each worker picks one task. Note that we assume at this point that each task comprises more or less an equal amount of workload.

- **Setting up dedicated communication channels**
  Once all tasks are distributed among the workers, they proceed to create direct channels for communication with their nearest-reachable neighbours (all the workers' IP addresses are broadcasted in the JavaSpace, together with information on the neighbouring partitions). This is necessary, as all workers need to exchange information of the traffic flows at their respective boundaries. The master also advertises its location in the JavaSpace, after which each worker requests a private communication channel to the master. Fig. 5 shows the relations between the computing units in the heterogeneous network architecture. Note that, because of its reliability, the TCP/IP protocol remains the communication mechanism of the underlying network transport layer.

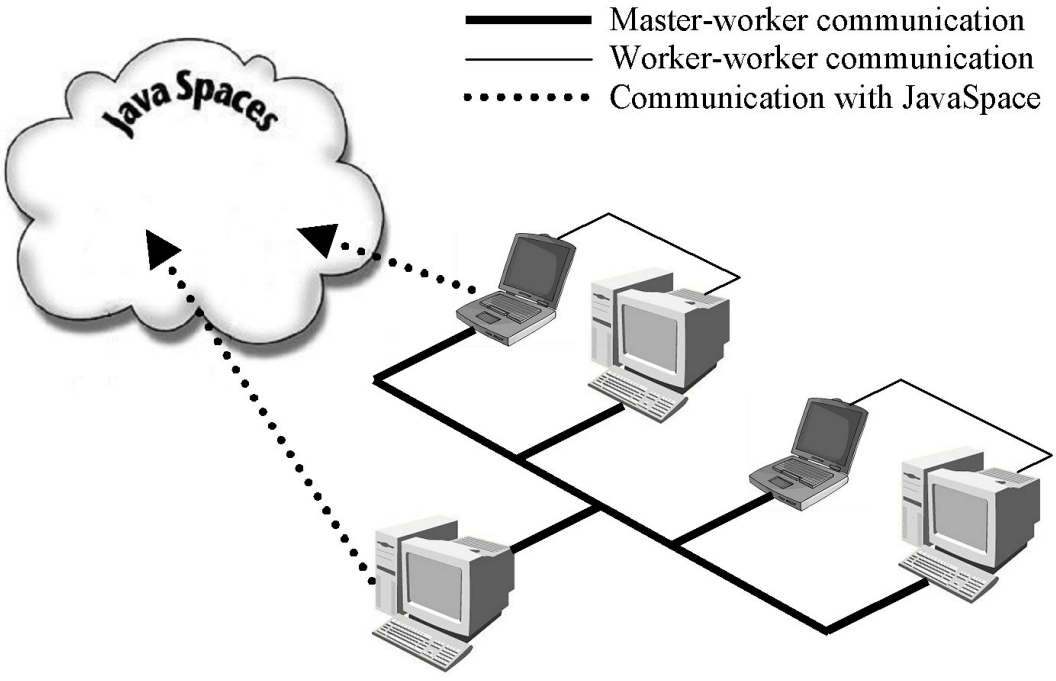

FIG. 5: A schematic overview of the three different kinds of dedicated communication channels; communication between the master and the workers (shown as thick lines), between workers' neighbours (shown as thin lines) and with the JavaSpace (shown as dotted arrows).

- **A simulation step**
  The master then initiates communication with all the workers, instructing them to advance to the next timestep of the master clock. At this stage, several intricate aspects need to be dealt with:

– each worker knows its neighbours, and communicates with them in order to transfer vehicles that are crossing zones,
– communication should only be performed when there are vehicles to transfer; consecutive links have small overlapping regions such that vehicles transfers only occur within these regions,
– for reasons of computational efficiency, we propose a hybrid cell/vehicle oriented approach: when simulating, only active cells (i.e., containing vehicles) are updated,
– because the JavaSpace itself is not efficient enough yet and because it assumes loose coupling, we only use it for information that is not frequently accessed (e.g., link travel times that are conventionally broadcasted on a radio station or displayed above a certain road section), as mentioned in section II D 2.

Note that with respect to the real-time simulation of traffic flows, several promising projects were carried out. Examples of this are the simulations of Duisburg, Germany [9] and those of the German Autobahn network of the North- Rhine-Westphalia (NRW) region [81–83, 94]. This latter example also provides the user with a prediction of the traffic state up to one hour in the future. It can be consulted on-line via a website `http://www.autobahn.nrw.de` (see Fig. 6). The prediction is based on the *On-Line SIMulator* (OLSIM), which is an implementation of the brake-light BL-TCA model [28, 78]. The tuning of the simulation to the current state of the real-world road network, is done by comparing measurements from virtual detectors in the model and real-world loop detectors from the motorways at certain checkpoints (containing sources and sinks). Whenever a mismatch is found, vehicles are either added or removed, taking into account to avoid severe disturbances of the current traffic flow [9, 94].

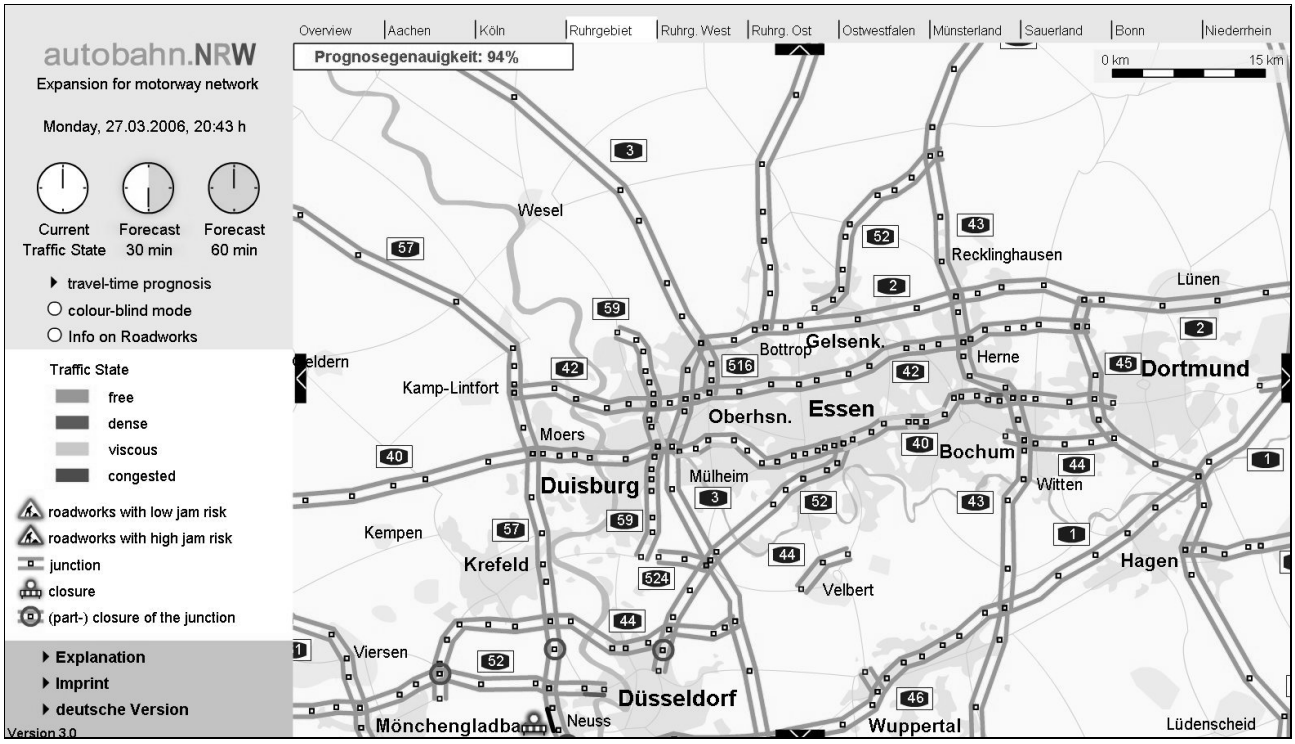

FIG. 6: A visualisation of the traffic in the Autobahn network of the North- Rhine-Westphalia (NRW) region in Germany. The figure shows the traffic state in the Ruhr area, predicted one half hour in the future, as broadcasted on an on-line website.

4. Issues related to synchronisation, graph cycles, and data sharing

Because the workers in the computing environment need to exchange information at their boundaries, *deadlocks* may occur in which some workers are mutually waiting for each other. However, in our implementation as described earlier, this can not happen because all the workers are directed by the master computer. This kind of *arbitration by an external third party*, is frequently done in systems needing robust synchronisation.

In a previous attempt at describing traffic in a road network, all links were initially topologically sorted after which they could be processed [58]. This excluded the presence of cycles in the graph describing the road network, which we now consider to be a major flaw of any simulator that exhibits this phenomenon. Using traffic cellular automata models solves

this problem seamlessly, because all vehicle updates are now being executed simultaneously. In general, vehicles are tagged for lane changes (taking care of side effects such as ping-pong traffic, then they execute their respective lane changes completing the lane-change model. Each vehicle's speed is then re-evaluated, after which all vehicles are moved to their new positions, completing the car-following model.

We conclude this section by mentioning that any data that should be shared among the workers, can be kept by a dedicated data server. In our implementation, we choose this data server to be the JavaSpace itself, providing an anonymous service of which the network address no longer explicitly needs to be known. The fact that the JavaSpace service is not yet efficient enough, is no problem because all time critical operations are done using dedicated communication channels, whereas the JavaSpace is only used for distributing tasks, sharing infrequently accessed information, ...

III. SOME EXAMPLE APPLICATIONS

Let us briefly consider some of the target applications of our framework (i.e., the integrated DTA methodology from section I and the DNL model described in section II), these encompass traffic state estimation, sustainability effects of traffic management systems, and assessing the impacts of traffic control measures.

A. Reliable state estimation of the road network

As implied at the end of section II D 3, it is possible to estimate the collective state of the traffic on the entire road network, based on information from the real-world (e.g., measurements stemming from single inductive loop detectors). As such, the framework can either simulate traffic in an off-line setting, based on historical data captured, e.g., in a single static OD matrix. It is then possible to derive information for a typical day, whereby the following aspects can be studied:

- the lengths of jams in both time and space,
- travel time losses and robustness properties,
- indicators for high-risk zones that contain recurrent congestion,
- and assessing the impact of an incident, leading to, e.g., lane closures.

In an on-line setting, the framework needs to be fed with real-time data, after which the simulation is ran to get a global updated view of the traffic state. Coupled with a prediction step, this leads to a powerful methodology that can be used to steer traffic, e.g., by advertising travel times on variable message signs (VMS).

B. Sustainability effects of traffic management systems

When thinking in a sustainable mobility framework, one approach could be to limit the traffic demand and to balance this demand over different traffic modes. As a complementary approach, one could also try to optimise the use of the existing infrastructure. With respect to the latter approach, we carried out a project, funded by the Belgian Federal Science Policy (DWTC) [61, 63].

One of the central components within the project, is a method to assess the 'quality' of a simulated traffic situation. To this end, we needed to define goals we would like to achieve; stated in control terms, this corresponds to a cost function, called the *sustainable cost function* (SCF). In the scope of this project on sustainable mobility, a definition of the cost function includes penalisations for *pollutant emissions* (environmental costs), *congestion* (socio-economic costs), *noise emissions*, *dangerous situations* (like shock waves), ... The cost function is expressed in terms of the states of the model and can be evaluated during simulation (all these costs are expressed in monetary terms). Within the project, we controlled traffic flows with respect to this cost function. If we use the SCF in steps 4 and 6 of the framework described in section I B 1, then this will lead to a social optimum with respecto to the concept of road pricing policies. The framework can then be used as a mirror of the real world.

C. Assessing the impacts of traffic control measures

As we believe all political decisions should hinge on advice from studies, these require an a posteriori interpretation with a good dose of common sense. Most of the time, such studies try to assess the impact of policy decisions that are implemented by means of local and global control measures. Typical decisions and measures include the following:

- rerouting effects, requiring a study of day-to-day and within-day replanning of commuters,
- ATMS effects (e.g., ramp metering, speed harmonisation, platoon driving, ...),
- policy decisions (e.g., overtaking prohibitions for trucks, road pricing strategies, ...),
- and changes in the road infrastructure, possibly leading to induced traffic, which we believe requires a more activity-based approach.

IV. CONCLUSIONS

In this chapter, we constructed a framework that allows us to perform dynamic traffic assignment (DTA), integrating departure time choice (DTC) and dynamic route choice (DRC), coupled with a dynamic network loading (DNL) model. The method is built around a traffic flow model that is represented as a computationally efficient cellular automaton. After explaining two of the mainstream DTA approaches, i.e., analytical and simulation-based, we gave an overview of each of the framework's components. In a separate section, we payed explicit attention to the DNL model, considering traffic flow simulation from a historical perspective, and discussing the benefits of open-source software development. After a functional description of the simulator, some code implementation details were given, ending with an overview of parallellisation through distributed computing. In a final section of the chapter, we discussed some example applications such as traffic state estimation, sustainability effects of traffic management systems, and assessing the impacts of traffic control measures.

Acknowledgements

Dr. Bart De Moor is a full professor at the Katholieke Universiteit Leuven, Belgium. Our research is supported by: **Research Council KUL**: GOA AMBioRICS, several PhD/postdoc & fellow grants, **Flemish Government**: **FWO**: PhD/postdoc grants, projects, G.0407.02 (support vector machines), G.0197.02 (power islands), G.0141.03 (identification and cryptography), G.0491.03 (control for intensive care glycemia), G.0120.03 (QIT), G.0452.04 (new quantum algorithms), G.0499.04 (statistics), G.0211.05 (Nonlinear), research communities (ICCoS, ANMMM, MLDM), **IWT**: PhD Grants, GBOU (McKnow), **Belgian Federal Science Policy Office**: IUAP P5/22 ('Dynamical Systems and Control: Computation, Identification and Modelling', 2002-2006), PODO-II (CP/40: TMS and Sustainability), **EU**: FP5-Quprodis, ERNSI, **Contract Research/agreements**: ISMC/IPCOS, Data4s,TML, Elia, LMS, Mastercard.

[1] Torgil Abrahamsson. *Estimation of Origin-Destination Matrices Using Traffic Counts: A Literature Survey*. IR-98-021, International Institute for Applied Systems Analysis, May 1998.

[2] Mark Baker, Rajkumar Buyya, and Domenico Laforenza. Grids and grid technologies for wide-area distributed computing. In *Software – Practice and Experience*, 2002.

[3] Ramachandran Balakrishna. Calibration of the demand simulator in a dynamic traffic assignment system. Master's thesis, Massachusetts Institute of Technology, June 2002.

[4] Michael Balmer, Kay W. Axhausen, and Kai Nagel. A demand generation framework for large scale micro simulations. In *Proceedings of the 85th Annual Meeting of the Transportation Research Board*, Washington, D.C., January 2006.

[5] Michael Balmer, Kai Nagel, and Bryan Raney. Large-scale multi-agent simulations for transportation applications. *Journal of Intelligent Transportation Systems*, 8(4):205–222, May 2004.

[6] Michael Balmer, Marcel Rieser, Arnd Vogel, Kay W. Axhausen, and Kai Nagel. Generating day plans based on origin-destination matrices. In *Proceedings of the 5th Swiss Transport Research Conference (STRC)*, Monte Verità, Ascona, March 2005.

[7] J. Barceló. Microscopic traffic simulation: A tool for the analysis and assessment of its systems. In *Proceedings of the Half Year Meeting of the Highway Capacity Committee*, Lake Tahoe, July 2002.

[8] Jaime Barceló and Jordi Casas. *Heuristic Dynamic Assignment based on Microscopic Traffic Simulation*. 9th Meeting of the Euro Working Group on Transportation, June 2002. Bari, Italy.

[9] R. Barlović, J. Esser, K. Froese, W. Knospe, L. Neubat, M. Schreckenberg, and J. Wahle. Online traffic simulation with cellular automata. *Traffic and Mobility: Simulation-Economics-Environment*, pages 117–134, July 1999. Institut für Kraftfahrwesen, RWTH Aachen, Duisburg.

[10] Richard Batley, Tony Fowkes, David Watling, Gerard Whelan, Andrew Daly, and Eiji Hato. *Models for analysing route choice*. Institute for Transport Studies, University of Leeds, UK.

[11] Richard P. Batley and Richard G. Clegg. Driver route and departure time choices: The evidence and the models. In *Proceedings of the 33rd Annual UTSG conference*, St. Anne's College, Cambridge, 2001.

[12] Martin Beckmann, Charles Bartlett McGuire, and Christopher B. Winsten. *Studies in the Economics of Transportation*. Number RM-1488. Yale University Press, May 1955.

[13] M. Ben-Akiva and M. Bierlaire. Discrete choice methods and their applications to short term travel decisions. In R.W. Hall, editor, *Handbook of Transportation Science*. Kluwer Academic Publishers, Massachusetts, 1999.

[14] M.E. Ben-Akiva and S.R. Lerman. *Discrete Choice Analysis: Theory and Application to Travel Demand*. MIT Press, Cambridge Massachusetts, 1985.

[15] Moshe Ben-Akiva, Michel Bierlaire, H.N. Koutsopoulos, and R. G. Mishalani. DynaMIT: Dynamic network assignment for the management of information to travelers. In *Proceedings of the 4th Meeting of the EURO Working Group on Transportation*, Newcastle, United Kingdom, 1996.

[16] Moshe Ben-Akiva, Michel Bierlaire, H.N. Koutsopoulos, and R. G. Mishalani. DynaMIT: a simulation-based system for traffic prediction. In *Proceedings of the DACCORD Short Term Forecasting Workshop*, Delft, The Netherlands, February 1998.

[17] Michel Bierlaire. *Mathematical models for transportation demand analysis*. PhD thesis, Facultés Universitaires Notre-Dame de la Paix de Namur, December 1995.

[18] M.C.J. Bliemer. *Analytical dynamic traffic assignment with interacting user-classes: theoretical advances and applications using a variational inequality approach*. PhD thesis, Faculty of Civil Engineering and Geosciences, TU Delft, Delft, The Netherlands, 2001.

[19] Michiel Bliemer, Erik Versteegt, and Ronald Castenmiller. Indy: A new analytical multiclass dynamic traffic assignment model. In *TRISTAN V: The Fifth Triennial Symposium on Transportation Analysis*, Le Gosier, Guadeloupe, June 2004.

[20] Michiel C.J. Bliemer, Ronald J. Castenmiller, and Piet H.L. Bovy. *Analytical multiclass dynamic traffic assignment using a dynamic network loading procedure*.

[21] David Boyce. A practitioners guide to urban travel forecasting models. In *Proceedings of the Metropolitan Conference on Public Transportation Research*, pages 309–319, Chigaco, Illinois, June 1998.

[22] David Boyce, Der-Horng Lee, and Bin Ran. Analytical models of the dynamic traffic assignment problem. *Networks and Spatial Economics*, 1(3/4), 2001.

[23] David E. Boyce. Forecasting travel on congested urban transportation networks: Review and prospects for network equilibrium models. In *Proceedings of TRISTAN V: The Fifth Triennial Symposium on Transportation Analysis*, Le Gosier, Guadeloupe, June 2004.

[24] Caliper Corporation. *TransCAD*, 2001.

[25] Gordon Cameron, Brian J. N. Wylie, and David McArthur. *PARAMICS – Moving Vehicles on the Connection Machine*. Technical report, Edinburgh Parallel Computing Centre (EPCC), The University of Edinburgh, 1994. IEEE, ISSN 1063-9535.

[26] E. Cascetta and C. Cantarella. A day-to-day and within-day dynamic stochastic assignment model. *Transportation Research A*, 25(5):277–291, 1991.

[27] Nurhan Cetin, Adrian Burri, and Kai Nagel. A large-scale agent-based traffic microsimulation based on queue model. In *Proceedings of the 3rd Swiss Transport Research Conference (STRC)*, Monte Verità, Ascona, Switserland, March 2003.

[28] Roland Chrobok, Sigurdur F. Hafstein, and Andreas Pottmeier. Olsim: A new generation of traffic information systems. In V. Macho and K. Kremer, editors, *Forschung und wissenschaftliches Rechnen*, number 63, pages 11–25, 2004.

[29] André de Palma and Fabrice Marchal. Real cases applications of the fully dynamic METROPOLIS tool-box: an advocacy for large-scale mesoscopic transportation systems. *Networks and Spatial Economics*, 2(4):347–369, 2002.

[30] E. W. Dijkstra. A note on two problems in connection with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[31] Dick Ettema, Guus Tamminga, and Harry Timmermans. A micro-simulation model system of departure time and route choice under travel time uncertainty. In *Proceedings of the 10th International Conference on Travel Behaviour Research*, Lucerne, August 2003.

[32] Eric Freeman, Susanne Hupfer, and Ken Arnold. *JavaSpaces: Principles, Patterns and Practice*. Addison-Wesley, 3rd edition, November 1999. ISBN 0-201-30955-6.

[33] C. Gawron, S. Krauß, and P. Wagner. Dynamic user equilibria in traffic simulation models. In *Proceedings of the Workshop on Traffic and Granular Flow '97*, number 307, Duisburg, Germany, 1997.

[34] Christian Gawron. An iterative algorithm to determine the dynamic user equilibrium in a traffic simulation model. *International Journal of Modern Physics C*, 9(3):393–407, February 1998.

[35] Christian Gawron. *Simulation-Based Traffic Assignment*. PhD thesis, Universität zu Köln, February 1998.

[36] Al Geist, Adam Beguelin, Jack Dongarra, Weicheng Jiang, Robert Manchek, and Vaidy Sunderam. *PVM: Parallel Virtual Machine – A Users' Guide and Tutorial for Networked Parallel Computing*. Massachusetts Institute of Technology Press, 1994.

[37] David Gelernter. *Linda Spaces*. Yale Linda Group, Yale University, 1996.

[38] D.L. Gerlough. Simulation of traffic flow. In D.L. Gerlough and D.G. Capelle, editors, *An Introduction to Traffic Flow Theory*, pages 97–118, Washington, D.C., 1964. Highway Research Board. Special Report 79.

[39] William Grosso. *Java RMI*. O'Reilly, 2002. ISBN 1-56592-452-5.

[40] C. Hendrickson and G. Kocur. Schedule delay and departure time decisions in a deterministic model. *Transportation Science*, 15(1):62–77, 1981.

[41] Riko Jacob, Madhav Marathe, and Kai Nagel. *A Computational Study of Routing Algorithms for Realistic Transportation Networks*. LA-UR-98-2249, Los Alamos National Laboratory, 1998.

[42] D. Joksimovic and M. Bliemer. *Dynamic Assignment Models for Assessing Road Pricing Policies*. MD-PIT Meeting, Faculty of

Civil Engineering and Geosciences, Delft University of Technology, Amsterdam, The Netherlands, September 2002.
[43] Jesse H. Katz. Simulation of a traffic network. *Source Communications of the ACM archive*, 6(8):480–486, August 1963.
[44] Carsten Kemper. Dynamic traffic assignment with static od matrices. In *Proceedings of the 2nd International Symposium*.
[45] Carsten Kemper. Dynamic traffic flow model – a new approach with static data. In *Proceedings of the 5th European Congress and Exhibition on Intelligent Transport Systems (ITS)*, Hannover, Germany, June 2005.
[46] Daniel Krajzewicz. *Simulation of Urban MObility – SUMO*. Deutsches Zentrum für Luft- und Raumfahrt (DLR), December 2004.
[47] Dawid Kurzyniec, Tomasz Wrzosek, Dominik Drzewiecki, and Vaidy Sunderam. Towards self-organizing distributed computing frameworks: The H2O approach. *Parallel Processing Letters*, 13(2):273–290, 2003.
[48] Alejandro Lago. *Spatial Models of Morning Commute Consistent with Realistic Traffic Behavior*. UCB-ITS-DS-2003-1, Institute of Transportation Studies, University of California at Berkeley, 2003. ISSN 0192 4109.
[49] Alejandro Lago and Carlos F. Daganzo. *A Network Model of Departure Time Choice with Spillovers and Merging Effects. Part I: A Building Block*. UCB-ITS-RR-2003-5, Institute of Transportation Studies, University of California at Berkeley, July 2003.
[50] David Levinson and Ajay Kumar. Integrating feedback into the transportation planning model: Structure and application. In *Proceedings of the 72nd Annual Meeting of Transportation Research Board*, Washington, D.C., January 1993.
[51] Yongtaek Lim and Benjamin Heydecker. Dynamic departure time and stochastic user equilibrium assignment. *Transportation Research B*, 39:97–118, 2005.
[52] Quadstone Limited. *Paramics-online v3 – System Overview*. Technical report, Quadstone Limited, 16 Chester Street, Edinburgh EH3 7RA, Scotland, November 2000.
(URL: `http://www.paramics-online.com`).
[53] Henry X. Liu, Xuegang Ban, Bin Ran, and Pitu Mirchandani. An analytical dynamic traffic assignment model with probabilistic travel times and perceptions. *Transportation Research Record*, 2002.
[54] Hong K. Lo. A dynamic traffic assignment formulation that encapsulates the cell-transmission model. In A. Cedar, editor, *Transportation and Traffic Traffic Theory*, pages 327–350. Elsevier Science, 1999.
[55] Hong K. Lo and W.Y. Szeto. A cell-based variational inequality formulation of the dynamic user optimal assignment problem. *Transportation Research B*, 36:421–443, 2002.
[56] Ewing Lusk. *MPI-2: Extensions to the Message-Passing Interface*. Message Passing Interface Forum, November 2003.
[57] Sven Maerivoet. *Advanced Computer Graphics using OpenGL*. Universitaire Instelling Antwerpen, Department of Mathematics and Computer Science, 2001.
[58] Sven Maerivoet. Het gebruik van microscopische verkeerssimulatie bij een onderzoek naar de fileproblematiek op de antwerpse ring. Master's thesis, Universitaire Instelling Antwerpen, June 2001. Promotor: prof. dr. Serge Demeyer.
[59] Sven Maerivoet. Efficient microscopic simulation of large scale highway traffic flows. In *23rd Benelux Meeting on Systems and Control*, Helvoirt, The Netherlands, March 2004.
[60] Sven Maerivoet. *Models in Aid of Traffic Management*. Seminar slides for 'Transportmodellen ter ondersteuning van het mobiliteits- en vervoersbeleid', May 2004. Federaal Wetenschapsbeleid, Belgium.
[61] Sven Maerivoet. Sustainability effects of traffic management systems. In *The Transport Science and Technology Congress, Athens, Greece, 2004*, September 2004. Slides presented at TransTec 2004.
[62] Sven Maerivoet and Bart De Moor. Development of an improved traffic cellular automaton model for traffic flows on a highway road network. In *Proceedings of the 10th World Congress and Exhibition on Intelligent Transport Systems and Services (CD-ROM)*, Madrid, Spain, November 2003. ERTICO, ITS Europe.
[63] Sven Maerivoet and Bart De Moor. *Sustainability Effects of Traffic Management Systems*. TR 06-033, Department of Electrical Engineering ESAT-SCD (SISTA), Katholieke Universiteit Leuven, March 2006. Final report for the DWTC PODO-II - CP/40 project.
[64] K.M. Malone, M.C.J. Bliemer, C.H. Menard, and H.H. Versteegt. *Testing INDY for use as the Dynamic Traffic Assignment model in the Dutch National Model*. 03-7N-084-73431, TNO Inro, Delft, June 2003. ISBN 90-5986-015-2.
[65] Fabrice Marchal. Integrating time of use in large scale traffic simulations. In *Proceedings of the 3rd Swiss Transport Research Conference (STRC)*, Monte Verità, Ascona, March 2003.
[66] Maryland Transportation Initiative, University of Maryland. *DYNASMART-X 0.930 User's Guide*, December 2003.
[67] Michael G. McNally. *An Activity-Based Microsimulation Model for Travel Demand Forecasting*. UCI-ITS-AS-WP-96-1, Institute of Transportation Studies, Center for Activity Systems Analysis, University of California, 1996.
[68] A. Messmer and M. Papageorgiou. METANET: A macroscopic simulation program for motorway networks. *Traffic Engineering and Control*, 31:466–470, 1990.
[69] Kai Nagel. Experiences with iterated traffic microsimulations in dallas. In D.E. Wolf and M. Schreckenberg, editors, *Proceedings of the Workshop on Traffic and Granular Flow '98*. Springer Verlag, 1998.
[70] Kai Nagel and Marcus Rickert. Parallel implementation of the TRANSIMS micro-simulation. *Parallel Computing*, 27(12):1611–1639, 2001.
[71] Kai Nagel, Marcus Rickert, Patrice M. Simon, and Martin Pieck. *The dynamics of iterated transportation simulations*. Submitted to Transportation Research C, February 2000.
[72] Kai Nagel and Michael Schreckenberg. *Traffic Jam Dynamics in Stochastic Cellular Automata*. LA-UR 95-2132, Los Alamos National Laboratory, 1995.
[73] Kai Nagel, Paula Stretz, Martin Pieck, Shannon Leckey, Rick Donnelly, and Christopher L. Barrett. *TRANSIMS traffic flow characteristics*. Technical report, Los Alamos National Laboratory, June 1998.
[74] A. Nagurney. *Network economics: a variational inequality approach*. Kluwer Academic Publisher, Boston, USA, 1993.
[75] M. Patriksson. *The Traffic Assignment Problem: Models and Methods*. VSP International Science Publishers, Utrecht, The Netherlands, 1994. ISBN 90-6764-181-2.
[76] M. Patriksson. *Nonlinear programming and variational inequality problems: a unified approach*. Kluwer Academic Publisher, Dordrecht, The Netherlands, 1999.

[77] Srinivas Peeta and Athanasios K. Ziliaskopoulos. Foundations of dynamic traffic assignment: The past, the present and the future. *Networks and Spatial Economics*, 1:233–265, 2001. Kluwer Academic Publishers, The Netherlands.
[78] Andreas Pottmeier, Sigurdur F. Hafstein, Roland Chrobok, and Michael Schreckenberg. *OLSIM: An Approved Traffic Information System*. Physics of Transport and Traffic, University of Duisburg-Essen, 2004.
[79] PTV. *PTV Vision software suite (VISUM/VISEM/VISEVA/VISSIM)*. PTV AG – Traffic Mobility Logistics, 2005.
[80] Eric Steven Raymond. *The Cathedral and the Bazaar*, 2000.
[81] M. Rickert, P. Wagner, and Ch. Gawron. Real-time traffic simulation of the German Autobahn network. In *Proceedings of the 4th Workshop on Parallel Systems and Algorithms PASA'96*. World Scientific Publishing Co. Pte. Ltd., 1996.
[82] Marcus Rickert. *Traffic Simulation on Distributed Memory Computers*. PhD thesis, Universität zu Köln, December 1997.
[83] Marcus Rickert and Peter Wagner. Parallel real-time implementation of large-scale, route-plan-driven traffic simulation. *Internation Journal of Modern Physics C*, 7(2):133–153, 1996.
[84] S. Rosswog, C. Gawron, S. Hasselberg, R. Böning, and P. Wagner. Computational aspects in traffic simulation problems. *Future Generation Computer Systems*, 17(5):659–665, March 2001.
[85] K.A. Small. The scheduling of consumer activities: work trips. *American Economic Review*, 72(3):647–679, 1982.
[86] K.A. Small. A discrete choice model for ordered alternatives. *American Economic Review*, 55(2):409–424, 1987.
[87] Sun Microsystems, Inc. *JavaSpaces Service Specification v2.0*, 2003.
[88] Srinivasan Sundaram. Development of a dynamic traffic assignment system for short-term planning operations. Master's thesis, Massachusetts Institute of Technology, June 2002.
[89] W.Y. Szeto and Hong K. Lo. A cell-based simultaneous route and departure time choice model with elastic demand. *Transportation Research B*, 38B(7):593–612, August 2004.
[90] Nicholas B. Taylor. The CONTRAM dynamic traffic assignment model. *Networks and Spatial Economics*, 3:297–322, 2003.
[91] Douglas Thain, Todd Tannenbaum, and Miron Livny. Distributed computing in practice - the condor experience. *Concurrency and Computation - Practice and Experience*, 2004.
[92] H.H. Versteegt, K.M. Malone, A.I. Martins, R.J. Castenmiller, and M.C.J. Bliemer. *Follow-up study on testing INDY for use as a Dynamic Traffic Assignment model in the Dutch National Model*. 03-7N-233-74021, TNO Inro, December 2003.
[93] William S. Vickrey. Congestion theory and transport investment. *American Economic Review*, 59:251–260, 1969.
[94] Joachim Wahle, Roland Chrobok, Andreas Pottmeier, and Michael Schreckenberg. A microscopic simulator for freeway traffic. *Networks and Spatial Economics*, 2:371–386, 2002.
[95] J.H. Wu, Y. Chen, and M. Florian. The continuous dynamic network loading problem: A mathematical formulation and solution method. *Transportation Research B*, 32(3):173–187, 1998.
[96] Hai Yang and Qiang Meng. Departure time, route choice and congestion toll in a queueing network with elastic demand. *Transportation Research B*, 32(4):247–260, 1998.
[97] Qi Yang. *A Simulation Laboratory for Evaluation of Dynamic Traffic Management Systems*. PhD thesis, Massachusetts Institute of Technology, June 1997.
[98] The DNL component in a DTA approach has also received special attention, e.g., in the work of Wu et al. who discuss it as a system of functional equations [95]. In this sense, the DNL model can be considered as a within-day model as explained by Cascetta and Cantarella [26].
[99] More information can be found in the overviews of Bliemer [18], Boyce et al. [22], and Peeta and Ziliaskopoulos [77].
[100] In our framework, a traveller wishing to undertake a journey is faced with two options: at what time does he/she depart, and which route will he/she take ? Different possibilities exist for tackling this problem; one example is the approach taken in the DynaSMART model [66], in which a commuter can select between changing his/her departure time, changing the route, changing both, or remaining with his/her current choice; the choice is based on a multinomial logit model.
[101] In the period after Vickrey, Hendrickson and Kocur were among the first to consider DTC in combination with a deterministic queueing model in a user equilibrium setting [40]. A while later, Small developed a DTC procedure that is based on a generalisation of the multinomial logit model [85, 86].
[102] The symbols β and γ stem from Vickrey's original formulation; in his work, he denoted the cost associated with the travel time as α.
[103] See the report of Batley et al. for a thorough overview of different logit models for route choice [10].
[104] `http://sumo.sourceforge.net`
[105] `http://www.gnu.org/licenses`
[106] `http://www.opensource.org`
[107] `http://creativecommons.org/licenses`
[108] As Balmer et al. noted, "*With 100 Mbit Ethernet, the best possible real-time ratio of a parallel traffic simulation with a one-second time step is approximately 170.*" [5]. In these systems, the bottleneck is not the communication speed but the *network latency*, i.e., the time needed to initiate a message; the only way to tackle this is to use different hardware with a lower communication overhead, e.g., Myrinet (`http://www.myri.com`), increasing the real-time ratio towards 800 [27].