Katholieke Universiteit Leuven

Departement Elektrotechniek ESAT-SCD (SISTA) / TR 03-100

Development of an Improved Traffic Cellular Automaton Model for Traffic Flows on a Highway Road Network

Sven Maerivoet[†] and Bart De Moor[†]

July 2003
Paper accepted for publication in
Proceedings of the 10th World Congress and Exhibition on Intelligent Transport Systems and Services
16-20/11/2003, Madrid (Spain)

This report is available by anonymous ftp from *ftp.esat.kuleuven.ac.be* in the directory *pub/sista/smaerivo/reports/paper-itss03.pdf*

[†]Katholieke Universiteit Leuven
Department of Electrical Engineering ESAT-SCD (SISTA)
Kasteelpark Arenberg 10, 3001 Leuven, Belgium
Phone: (+32) (0)16/32.17.09 Fax: (+32) (0)16/32.19.70
E-mail: {*sven.maerivoet,bart.demoor*}*@esat.kuleuven.ac.be*
WWW: *http://www.esat.kuleuven.ac.be/scd*

Research supported by
Research Council KUL: GOA-Mefisto 666, several PhD/postdoc & fellow grants;
Flemish Government: FWO: PhD/postdoc grants, projects, G.0240.99 (multilinear algebra), 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), research communities (ICCoS, ANMMM); **AWI**: Bil. Int. Collaboration Hungary/Poland; **IWT**: PhD Grants, Soft4s (softsensors),
Belgian Federal Government: DWTC (IUAP IV-02 (1996-2001) and IUAP V-22 (2002-2006), PODO-II (CP/40: TMS and Sustainability);
EU: CAGE; ERNSI; Eureka 2063-IMPACT; Eureka 2419-FliTE;
Contract Research/agreements: Data4s, Electrabel, Elia, LMS, IPCOS, VIB.

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

```
@inproceedings{MAERIVOET:03,

   author         =   ”Sven Maerivoet and Bart De Moor”,
   title          =   ”Development of an Improved Traffic Cellular Automaton
                       Model for Traffic Flows on a Highway Road Network”,
   booktitle      =   ”Proceedings of the 10th World Congress and Exhibition
                       on Intelligent Transport Systems and Services”,
   organization   =   ”ERTICO, ITS Europe”,
   year           =   ”2003”,
   month          =   nov,
   address        =   ”Madrid, Spain”

}
```

DEVELOPMENT OF AN IMPROVED TRAFFIC CELLULAR AUTOMATON MODEL FOR TRAFFIC FLOWS ON A HIGHWAY ROAD NETWORK

Sven Maerivoet and Bart De Moor

Katholieke Universiteit Leuven
Department of Electrical Engineering ESAT-SCD (SISTA)
Kasteelpark Arenberg 10, 3001 Leuven (Belgium)
Phone: +32 16 32 17 09 / Fax: +32 16 32 19 70
E-mail: {sven.maerivoet,bart.demoor}@esat.kuleuven.ac.be
http://www.esat.kuleuven.ac.be/scd

SUMMARY

As traffic demand on road networks steadily increases every year, the need for alleviation of global congestion arises. Our contribution lies in the assisting of traffic engineers and policy makers who are faced with the problem of controlling the traffic flows. We develop a flexible microscopic simulation environment and discuss the qualitative behaviour of the different traffic cellular automata models that form its basis. We conclude by giving a functional description of the traffic simulator, as well as a possible platform independent parallel implementation that uses Java in a heterogeneous distributed computing environment.

INTRODUCTION

One of the main challenges today in traffic flow modelling, lies in the construction of macroscopic and microscopic models that lend themselves to a faithful representation of road traffic. This proves to be of tremendous importance in alleviating road congestion, as these models are used in several key aspects in the control of traffic flows by means of advanced traffic management and information systems (ATMS/ATIS).

With the increasing traffic demand on the Belgian road network, the need for a better usage of the existing road infrastructure (i.e., the supply) arises in order to counter the forthcoming congestion. Within this context, our research is aimed at improving the current and future traffic conditions. To this end, we develop a flexible testbed environment that is capable of providing us with a simulation model of a real-world road network. In a first setup, this will consist of the Flemish primary highway network, connecting several densely populated areas. At a later stage, we include the secondary national road network, which can be used as a fall-back option for traffic control measures. Eventually, we have an integrated simulation environment, capable of dealing with highway traffic flows, as well as local urban traffic.

Nowadays, the focus of traffic simulation has shifted towards large scale applications, i.e., on a country wide level. Policy makers need adequate models to base their decisions on, detailed models are deployed in various different real-life scenarios when controlling the traffic flows,… All this leads us to the conclusion that, in order to create reliable simulation models, we need on the one hand to perform very detailed computations, while on the other hand keeping the performance at a high enough level. The latter is a necessity when using control measures that require *faster than real-time* simulations (e.g., a ramp metering setup using model predictive control that is coupled with a high fidelity microscopic traffic flow model).

Our paper addresses the construction of such a high-performance simulation system. The difference with several other available systems, is that ours is built entirely using the Java programming language. This allows us to use a very *heterogeneous* computing environment, composed of different kinds of computers, running different operating systems and cooperating in a distributed fashion. The challenge lies in getting these computing units to work together, in a reliable and faster than real-time way.

In the following section, we give an introduction to both the macroscopic and microscopic classes of traffic flow models, putting more emphasis on the latter class. The next two sections contain a description of the cellular automata models and some qualitative behavioural observations. We then move on to a subsequent section, considering some notes on distributed implementations of traffic simulators. The next two sections give a description of a road network simulator, from a functional and an implementation point of view. The final section recapitulates the main conclusions drawn from this paper.

TRAFFIC FLOW MODELS

In this section, we present a brief overview of the different main classes of traffic flow models, focussing on the class of microscopic models.

Macroscopic and microscopic flavours

Traffic flow models can traditionally be classified as being either macroscopic or microscopic in nature. The former considers a traffic flow as a compressible fluid and uses fluid dynamical partial differential equations to propagate flow and density between large discretized time/space sections. The latter however, considers each vehicle in a traffic flow individually, resulting in a more detailed description of the dynamical process behind a traffic flow. Note that a third model class exists, namely the mesoscopic models. These models are, from an implementation point of view, closely related to the macroscopic ones. The difference is that the underlying theory of the former models is based on many-particle systems which can collectively be described using a gas kinetic analogy (1).

During the mid-nineties, many ready-to-use traffic flow simulators were built, with the goal of exporting the current state-of-the-art in traffic research to a broader public. Both classes of macroscopic and microscopic models flourished in their specific application domains. The former models quickly found their way to control oriented traffic modelling, with measures such as ramp metering, velocity harmonisation, dynamic rerouting, … Due to their intrinsic heavy computational overhead, the latter models were not so frequently used in real-time applications, but instead provided detailed descriptions of the traffic flows under study, lending themselves for off-line studies on exhaust gas pollutions, noise emissions, …

A cumbersome modelling approach

Whereas macroscopic (and mesoscopic) models are based on aggregate quantities such as flow, density and mean speed, microscopic models operate directly on the individual entities (i.e., vehicles) in a traffic flow. This means that they explicitly model the dynamics of and interactions between the different vehicles, thus in a sense resembling more closely reality. Two submodels which are of major importance, lie at the heart of all this:

- the car following submodel,
- and the lane changing submodel.

Both models are computationally very expensive, especially when employed with a fine discretisation scheme (e.g., time is discretised in tenths of a second, space is continuous). This led many researchers to dismissal of using microscopic simulators for real-time scenarios. Another major disadvantage, is the high number of parameters contained in both submodels (reaction times, speed distributions, acceleration characteristics, mean time and space headways, …). Most of these parameters aren't always that significant, and others are so sensitive that they influence the eventual outcome of a simulation too much. Because of their highly stochastic nature, calibrating and validating these models is rather difficult, even requiring some wizardry at times (2,3).

Efficient microscopic traffic flow models

However, a decade ago, a new kind of microscopic model was introduced. Instead of adopting a 'continuous' time/ space as in classical microscopic models, the cellular automata programming paradigm was proposed as a discrete dynamical system for the modelling of traffic flows (4). Although the level of detail is a bit lower than their fully continuous counterparts, they still provide the transportation engineer with sufficient detail in order to capture the emergent microscopic effects of the dynamical processes behind traffic flows. In the next section, we go into more detail regarding these traffic cellular automata (TCA) models, as they are called.

TRAFFIC CELLULAR AUTOMATA MODELS

In this section, we discuss the background of traffic cellular automata models, their associated physical setup (i.e., their implementation) and some thermodynamical considerations.

Background

TCA models represent traffic flows using very simplified models that have, on a microscopic scale, a rather low accuracy. This in turn has an important advantage: they provide an efficient and fast performance when used in computer simulations. Because of the fact that the focus in TCA models lies heavily on numerical simulations, it should be noted that exact analytical forms of the models are possible, although they are not easy to derive (5). The models explicitly describe local interactions between individual vehicles in a traffic flow, by means of a ruleset that reflects the rule based behaviour of a cellular automaton evolving in time and space. These interactions are based on the two car following and lane changing submodels. Note that another layer can rest on top of these submodels, namely the routing of the vehicles through static or dynamic traffic assignment.

Physical setup

There are two main ingredients in TCA models: the physical environment of the system and the ruleset that describes the local interactions.

The physical environment

The system itself consists of the individual cells that are used in the cellular automaton. In traffic cellular automata for single-lane roads, the layout consists of a one-dimensional lattice of cells. Each of these cells is occupied by at most one vehicle, which moves from one cell to another, leading to the description of 'particle-hopping models' (6).

The ruleset

The ruleset reflects the previously mentioned car following and lane changing submodels, describing the behaviour of the individual vehicles. These rules are consecutively applied to all vehicles in parallel. It is important to stress that the interactions between cells, as described by the cellular automaton's ruleset, are all *local* interactions. Complex global emergent behaviour arises from these simple local activities. Car following dynamics are typically expressed as a stimulus-response relationship. In the case of TCA, a vehicle's stimulus is mainly composed of its speed and distance to its leader; its response is given by a new speed, which is directly used in the update procedure that moves all vehicles to their new locations.

Thermodynamical considerations

Looking at TCA models from a physicist's point of view, they can exhibit several phases (i.e., co-existing states in equilibrium). Two of such phases are the free flowing and congested regimes. They correspond for example to the transitions between a gaseous state and a liquid state. It is important to make a distinction between first-order and second-order phase transitions: the former occur 'instantaneous', the latter proceed smoothly. Many traffic phenomena show a *first-order transition* (e.g., the transition from going from the free flowing to the congested regime).

From the theory of such two-phase models, it follows that *hysteresis effects* can occur when going from one phase to another (7). This property is closely related to the existence of *metastable states* in cellular automata: these states are characterized by a very high flow which can suddenly break down (called a 'capacity drop'). This naturally leads to a hysteresis phenomenon when traffic is recovering from congestion.

QUALITATIVE BEHAVIOUR OF TCA MODELS

Let us now give an overview of several classical TCA models. For each model, we discuss its qualitative behavioural characteristics, using illustrative time-space diagrams that are able to represent all traffic operations as the system evolves. In these diagrams, the paths of all the vehicles' movements are traced, resulting in a set of trajectories (the space direction is vertical, the time direction is horizontal); each vehicle is represented by a pixel. Congestion waves are clearly visible as they move upstream with time: vehicles entering such a wave reduce their speed (nearly horizontal trajectories) for a while, until they can accelerate again (steep ascending trajectories). Because the time axis points to the right and the space axis points upward, vehicles drive diagonally upwards to the right and congestion waves propagate diagonally downwards to the right.

We refer the reader to our technical report (5) that contains an extensive account of the different qualitative aspects related to TCA models, including results from probe vehicles, observations on the density-flow diagrams and insights into the distributions of mean speeds, mean space and time headways.

In all the experiments, we assume that a homogeneous set of vehicles is driving on a unidirectional, single-lane, closed ring road (Indianapolis scenario). This has the nice property that the number of vehicles (i.e, the global density) is conserved during a simulation run. The size of one cell in the system's lattice corresponds to 7.5 metres, each time step is equivalent to 1 second and the maximum speed of a single vehicle is 5 cells/second, corresponding to 135 km/h. The total length of the ring road is 2.25 km (corresponding to 300 cells).

The considered TCA models are: the fully deterministic CA-184, the Stochastic TCA, the Velocity Dependent Randomisation TCA and the Time Oriented TCA.

Wolfram's CA-184

Around 1985, Stephen Wolfram popularised the mathematical concept of cellular automata. In his research, he classified, amongst other things, all 256 one-dimensional two-state single-neighbour cellular automata (8). One of these is rule 184, in which particles (vehicles) move to the right if their right neighbour site is empty (they do not move otherwise). All of them move simultaneously at each discrete time step, so the CA-184 constitutes a fully deterministic system. Note that we can generalise and allow higher speeds for a vehicle, provided that we take each time the necessary safe space headway into account. Although the model has been extensively studied (in particulary by the use of mean field theory for an analytical treatment) and its behaviour is well understood, it remains a theoretical case with very limited practical usage.

Looking at a typical time-space diagram of the CA-184 in Figure 1, we see that it constitutes a fully deterministic system that strictly repeats itself. A characteristic of the congestion waves is that they have an eternal lifetime in the system.

Stochastic TCA

Applying cellular automata models to traffic flow theory, started with the seminal work done by Kai Nagel and Michael Schreckenberg (4). They developed what is known as the STCA, a cellular automaton that incorporates a noise parameter in its ruleset. This parameter, the *slowdown probability*, reflects the probability that a vehicle slows down for no apparent reason. The principal difference between other models of traffic flows and traffic cellular automata, are that the latter need some kind of fluctuations for the explanation of traffic jams, hence the presence of the noise parameter.

Much criticism was expressed due to this alteration, as it has no theoretical background and is introduced quite heuristically. Furthermore, if we look at a typical time-space diagram of the STCA (such as the one in Figure 2), we see that the braking behaviour gives rise to many artificial phantom mini-jams that have no corresponding counterpart in reality. The created jams are very unstable, which contradicts real-life observations.

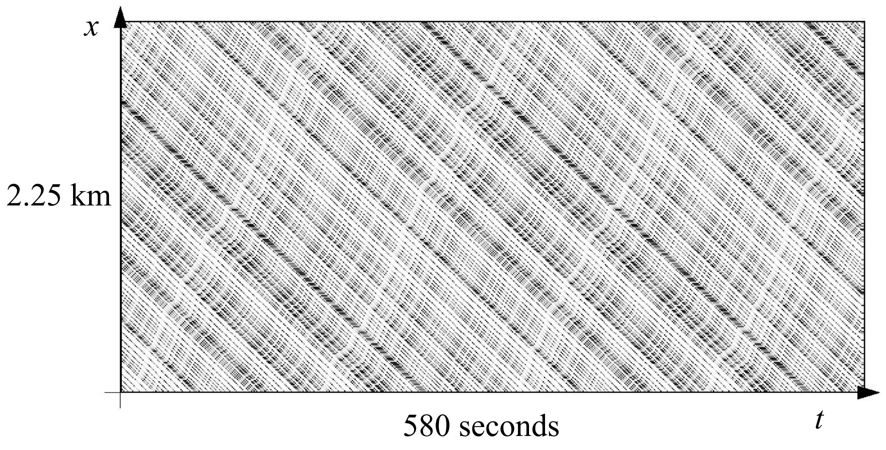

Figure 1: *A typical time-space diagram of the CA-184 TCA model. The global density of the system was set to 0.3. There are many deterministic congestion waves present; they have an eternal lifetime in the system.*

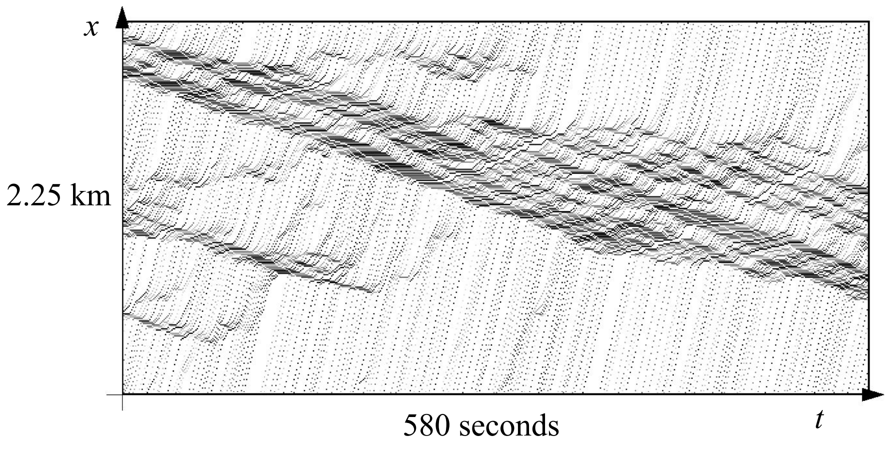

Figure 2: *A typical time-space diagram of NaSch's STCA model. The slowdown probability was set to 0.5 and the global density of the system to 0.2; the large congestion wave is very ravelled out, resulting in many mini-jams localised in its neighbourhood.*

Velocity Dependent Randomisation TCA

A possible approach to achieve stable traffic jams, is to reduce the outflow from such jams. This can, for instance, be accomplished by implementing so-called *slow-to-start* rules, which derive their name from the fact that vehicles exiting jam fronts are obliged to wait a small amount of time.

The VDR TCA is such an example (9). It has metastable states that are characterised by the fact that traffic near the critical density behaves in a metastable way. Sufficiently large disturbances of this fragile equilibrium can cause the flow to undergo a sudden drop, destroying the state of very high flow. The system settles into a phase separated state with a large megajam and a free flow zone. This phenomenon can be observed very clearly from the time-space diagram in Figure 3. The large jam persists as long as the global density is not significantly lowered (recovery of traffic shows a hysteresis phenomenon).

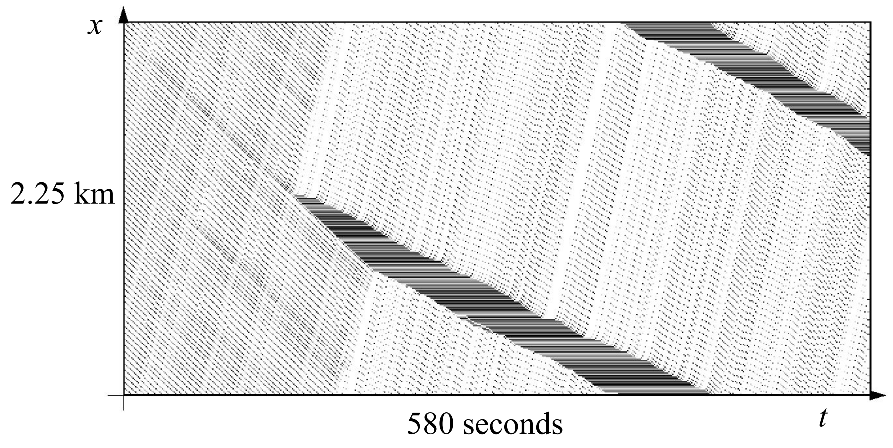

Figure 3: *A typical time-space diagram of the VDR TCA model. The slowdown probability was set to 0.01, the slow-to-start probability to 0.5 and the global density to 0.2. We can see the formation of a persistent compact jam; there's also a significant decrease of the density in front and behind the jam's formation.*

Time Oriented TCA

In order to tackle the problem of the STCA's unrealistic acceleration behaviour, the TCA's ruleset was constructed using a different approach: instead of using absolute speeds and space headways, travelled distance and time headways were used. One of the important results using this methodology, was the time oriented cellular automaton (TOCA), which introduced acceleration and deceleration probabilities and required – as opposed to the classical space clearance – a minimal time clearance for acceleration of the vehicles (10).

The TOCA's car following behaviour is very different from that of the STCA. There is some amount of *elasticity* introduced in the system, leading to more realistic accelerations of the vehicles. Considering the global qualitative behaviour of the system, we can study the time-space diagram in Figure 4. Here we see that an initially homogeneous state is destroyed, followed by the creation of several mini-jams. Some of these jams vanish, and the vehicles that leave them join into one persistent jam that is created. There is an important difference between the single jam here and the one of the VDR TCA (in the phase separated state): in the VDR TCA, vehicles are almost exclusively *all stopped* inside the jam, whereas in the TOCA model, they more or less resemble *stop-and-go* traffic.

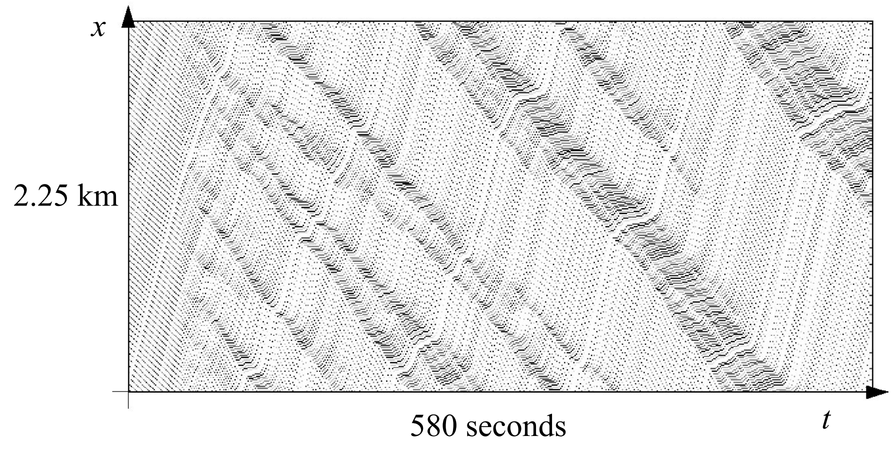

Figure 4: *A typical time-space diagram of the TOCA model. The average time headway was set to 1.15 seconds, both the acceleration and deceleration probabilities to 0.9 and the global density to 0.2. We can see an initially homogeneous state being destroyed, followed by the formation of many mini-jams that vanish and lead to the formation of one persistent jam.*

In the previous paragraphs we focussed our attention on the qualitative behaviour displayed by some of the classic traffic cellular automata models. The primary tool used for assessment, was by studying the time-space evolution of the system, exhibiting the formation of congestion waves on a closed ring road. In the next section, we discuss some considerations related to the distributed implementation of traffic simulators.

DISTRIBUTED SIMULATION MODELS

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, TRANSIMS, …), with some exceptions such as PARAMICS.

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 (11).

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 highway road network (12). 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.

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 (13). Whereas supercomputers performed intensive tasks in the past, we can nowadays observe a shift towards grid based computing (14). For us, the challenge now is to get reliable and efficient (i.e., faster than real-time) operation of this latest networked architecture.

In the next two sections, we present our approach for constructing a road network simulator: we first specify a functional description, immediately followed by a possible parallel implementation.

FUNCTIONAL DESCRIPTION OF THE NETWORK SIMULATOR

In this section, we give a minimal specification of the road network simulator based on microscopic traffic cellular automata models. Note that we do not consider other aspects such as the four step approach (trip generation and distribution, modal choice and route assignment) at this time.

Geographical scope

As outlined earlier, we restrict ourselves to Flanders' highway road network (as depicted in Figure 5). The main reason for not considering the underlying secondary national road network at this moment, is that we do not have any traffic data about it available. In contrast, we do have measurements coming from some 1600 sensors (most of them are single loop detectors). These detectors are unevenly distributed over all the Flemish highways, collecting point measurements: every minute the total flow of vehicles is locally measured, as well as the occupancy. From both these measurements, a detector's controller logic derives the vehicle type, resulting in the flow for passing cars and trucks, and an estimate of the (local) space mean speed.

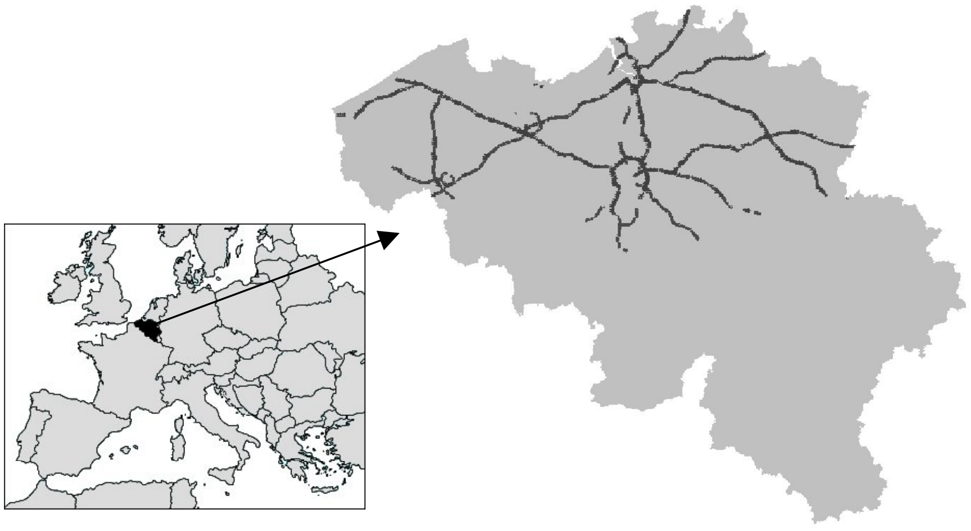

Figure 5: *Flanders' highway road network, lying in the northern part of Belgium. All roads are equipped with some 1600 sensors in total (mostly single loop detectors and some cameras), measuring local flows, occupancies and space mean speeds every minute (per lane).*

Network structure

Opting for an intuitive structure, our network consists of *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 unidirectional 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*. 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 submodel to actually use this information). With these elementary building blocks, the highway road 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. Not withstanding this freedom, they do have to agree 'functionally': e.g., lane changes should either be mandatory or discretionary (12), timesteps should be comparable, …

Origin-destination information

All routing behaviour is expressed when executing the lane change submodel. This requires suitable input data, which is estimated by approximate reconstruction of the origin-destination (OD) flows. These OD flows are based on data recorded by the previously mentioned sensors in the road network. These OD flows are then converted to turning fractions that are applied at link intersections. This means that most vehicles do no follow specific routes, but are probalistically guided through the network. We do however include the possibility that vehicles can have individual routing plans, which may even change during a simulation run.

In this respect, we note that in order to use the traffic simulator as a method to generate traffic forecasts, this has to be done in an anticipative way (15). This is a required condition because informed drivers react to information they obtained about the past. And this information is itself based on past reactions, clearly excluding static routing algorithms. Even more so, origin-destination information needs to be dynamic in nature, which is not always that obvious.

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 for example, cars and trucks, with trucks occupying extra cells. A convenient method for representating this difference, is using 'passenger-car-equivalents' (PCE). Note that the rulesets 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 private 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.

Gathering 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, so that at later stage, we can for example add a congestion based routing module that uses this information to guide some of the vehicles (i.e., the informed drivers) as they travel through the network.

A POSSIBLE PARALLEL IMPLEMENTATION

The flexible functional description set out in the previous section, allows several implementations. In this section, we present such a possible approach, in which parallellism is achieved using distributed computing. We first introduce the background of our approach, then give a brief overview of the software technologies used, after which we elaborate on several programmatorical and technical aspects, concluding with some issues related to sychronisations problems, cycles in the road network graph and sharing of data.

Philosophy behind the implementation

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 network are tightly coupled with each other. In this case, we have opted for a mixture of the 'master/worker' and 'command' application patterns (16). This means that we have one master computer that controls *N* distinct worker computers who execute the *different* tasks.

Technologies used

Currently, most distributed implementations of traffic simulators use classic communication techniques such as a Message Passing Interface (MPI) and/or Parallel Virtual Machines (PVM). 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.

<u>Direct communication</u>

It is very important that the computation/communication ratio remains as high as possible, after parallellising the simulator (16). 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 (17), implemented using Java's Remote Method Invocation (RMI) facility, and provided as a service of the Jini Network Technology (18). 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 technology – Jini – 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.

Programmatorical and technical aspects

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 (dynamic) origin-destination matrices, possible routing plans, … It then performs a domain decomposition on a geographical basis, dividing the network in exactly *N* zones (the splitting of the links is preferably done at junction nodes). Note that this encompasses static scheduling; at a later stage we can opt for a dynamic scenario, providing us with some means to perform load balancing. This can, for example, be accomplished by keeping track of the workers' computing times and redividing and reassigning zones.

Distributing the tasks

The next step consists of the master distributing the tasks 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. 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. Figure 6 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.

A simulation step

The master then initiates communication with all the workers, telling them to advance to the next timestep of the master clock. 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,
- one way or another, communication should only be performed when there are vehicles to transfer,
- 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).

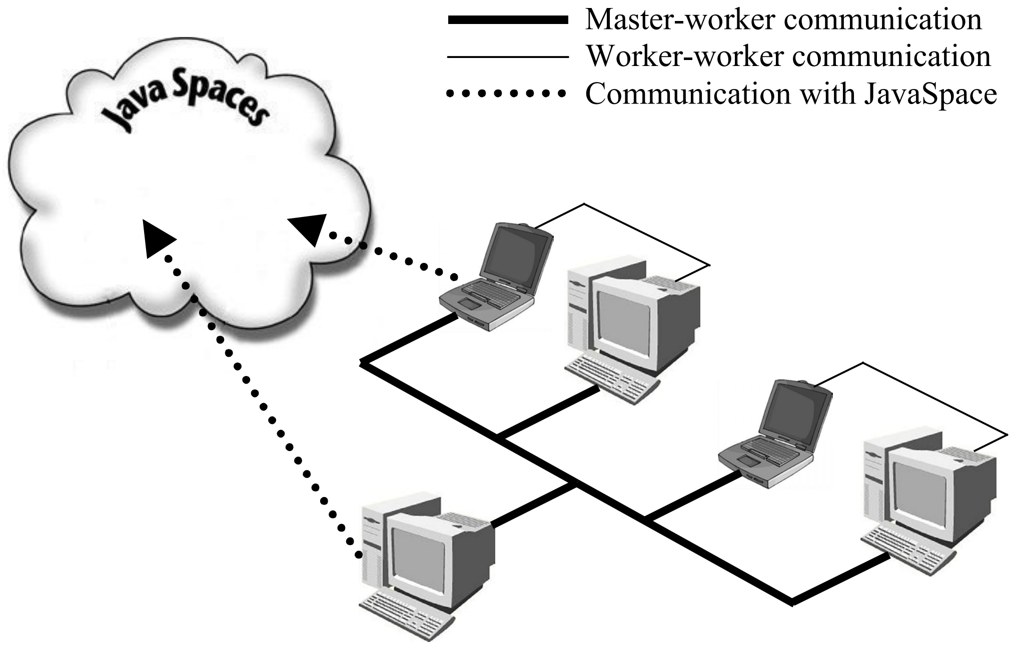
Master-worker communication
Worker-worker communication
Communication with JavaSpace
Java Spaces

Figure 6: *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).*

Some issues related to synchronisation, graph cycles and data sharing

Because the workers in the computing environment need to exchange information at their boundaries, deadlock 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 (12). 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 submodel. Each vehicle's speed is then re-evaluated, after which all vehicles are moved to their new positions, completing the car following submodel.

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, …

CONCLUSIONS

Acknowledging the fact that the alleviation of congestion requires specific control measures, we need to have an accurate and detailed real-world simulation environment that can capture the effects of policy decisions. Traffic cellular automata models provide us with sufficient detail in order to capture the microscopic nature of traffic flows. In this paper, we have described some qualitative behavioural observations regarding several of the classical TCA models. We then gave a functional description of a flexible testbed, based on a microscopic road network simulator. The use of TCA models coupled with a parallel implementation, establishes an efficient scheme that allows faster than real-time performance. We furthermore augment this scenario, with the inherent implementation of platform independence in a heterogeneous distributed computing environment.

ACKNOWLEDGEMENTS

MCS **Sven Maerivoet** is a doctoral student in the field of traffic flow modelling and simulation at the Katholieke Universiteit Leuven, Belgium. In 2001, he received his Master's degree in Computer Science at the University of Antwerp (UIA).

Prof. dr. ir. **Bart De Moor** is a full professor at the Katholieke Universiteit Leuven, Belgium.

This research was carried out with the support of the Belgian Federal Government (DWTC) PODO-II, DWTC-project CP/40 *"Sustainability Effects of Traffic Management Systems"*, and is furthermore supported by:

- **Research Council KUL**: GOA-Mefisto 666, several PhD/postdoc & fellow grants;
- **Flemish Government**: **FWO**: PhD/postdoc grants, projects, G.0240.99 (multilinear algebra), 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), research communities (ICCoS, ANMMM); **AWI**: Bil. Int. Collaboration Hungary/Poland; **IWT**: PhD Grants, Soft4s (softsensors),
- **Belgian Federal Government**: DWTC (IUAP IV-02 (1996-2001) and IUAP V-22 (2002-2006), PODO-II (CP/40: TMS and Sustainability);
- **EU**: CAGE; ERNSI; Eureka 2063-IMPACT; Eureka 2419-FliTE;
- **Contract Research/agreements**: Data4s, Electrabel, Elia, LMS, IPCOS, VIB.

REFERENCES

(1) Serge P. Hoogendoorn and Piet H.L. Bovy, "*State-of-the-art of Vehicular Traffic Flow Modelling*", Journal of Systems and Control Engineering, Special Issue on Road Traffic Modelling and Control, vol. 215, nr. 4, pages 283-304, 2001.

(2) Elmar Brockfeld, Reinhart D. Kühne, Alexander Skabardonis and Peter Wagner, "*Towards a benchmarking of microscopic traffic flow models*", Transportation Research Board, 82nd Annual Meeting, Washington D.C., 2003.

(3) R. Jayakrishan, Jun-Seok Oh and Abd-El Kader Sahraoui, "Calibration and Path Dynamics Issues In Microscopic Simulation for ATMS and ATIS", Institute of Transportation Studies, University of California, Technical Report UCI-ITS-WP-00-22, 2000.

(4) Kai Nagel and Michael Schreckenberg, *"A cellular automaton model for freeway traffic"*, Journal de Physique I France, vol. 2, pages 2221-2229, 1992.

(5) Sven Maerivoet and Bart De Moor, "A Survey on the Qualitative Behaviour of Single-Lane Traffic Cellular Automata", Technical Report 03-101 (in preparation), Katholieke Universiteit Leuven, Department of Electrical Engineering ESAT-SCD (SISTA), 2003.

(6) Kai Nagel, "*Particle hopping models and traffic flow theory*", Physical Review E, vol. 53, nr. 5, pages 4655-4672, 1996.

(7) Dominic Jost, "Breakdown and recovery in traffic flow models", Department of Computer Science, ETH Zürich, Master's Thesis, 2002.

(8) Stephen Wolfram, "A New Kind of Science", Wolfram Media, Inc., ISBN 1-579-955008-8, 2002.

(9) R. Barlović, L. Santen, A. Schadschneider and M. Schreckenberg, *"Metastable States in Cellular Automata for Traffic Flow"*, European Physics Journal, vol. B5, nr. 793, 1998.

(10) W. Brilon and N. Wu, "*Evaluation of Cellular Automata for Traffic Flow Simulation on Freeway and Urban Streets*", Traffic and Mobility: Simulation-Economics-Environment, Institut für Kraftfahrwesen, RWTH Aachen, Duisburg, pages 163-180, 1999.

(11) Kai Nagel and Michael Schreckenberg, "Traffic Jam Dynamics in Stochastic Cellular Automata", Los Alamos National Laboratory and Santa Fe Institute, Technical Report 95ATS089, 1995.

(12) Sven Maerivoet, "Het gebruik van microscopische verkeerssimulatie bij een onderzoek naar de fileproblematiek op de Antwerpse Ring", Universitaire Instelling Antwerpen, Master's Thesis, 2001.

(13) Kai Nagel and Marcus Rickert, "*Parallel implementation of the TRANSIMS micro-simulation*", Parallel Computing, vol. 27, nr. 1611, 2001.

(14) Mark Baker, Rajkumar Buyya and Domenico Laforenza, "*Grids and Grid technologies for wide-area distributed computing*", Software – Practice and Experience, 2002.

(15) Kai Nagel, "*Distributed intelligence in large scale traffic simulations on parallel computers*", Collective Cognition: Mathematical Foundations of Distributed Intelligence, Santa Fe Institute, 2002.

(16) Eric Freeman, Susanne Hupfer and Ken Arnold, "JavaSpaces: Principles, Patterns and Practice", Addison-Wesley, 3rd edition, ISBN 0-201-30955-6, 1999.

(17) Yale University, Department of Computer Science, Programming Languages and Software Systems Group (Yale Linda Group),
URL: http://www.cs.yale.edu/Linda/linda.html

(18) "JavaSpaces Service Specification v2.0", Sun Microsystems, Inc., 2003.