1 Deliverable D2.2: Siemens C-ITS Use Case (v6)

The lead responsible for this deliverable is Siemens.

2 Document History

Authors.

- TE Thomas Eiter (KBS)
- PS Patrik Schneider (Siemens)
- SP Peter Schüller (KBS)

Versions.

- 20.12.2018: Initial version of this document (PS)
- 14.01.2019: Added traffic flow calculation (PS)
- 23.05.2019: Improvement traffic flow calculation (PS)
- 24.10.2019: Added experiments 1 (PS)
- 31.01.2020: Structure changed, and content of ECAI paper added (PS)
- 15.03.2020: Added experiments (PS)

3 High-Level Description

The first Siemens use case is in the field of (cooperative) intelligent transportation systems (ITS). In particular, the use case includes the analyzation of V2X communication messages with the goal *G1* of detecting undesired traffic events (accidents, traffic jams) and the goal *G2* of finding a reconfiguration of traffic light controllers that ameliorates the effects of the undesired event at least partially.

We distinguish a traffic flow between the states of free flow and traffic jam, which is extended to a third phase "synchronized flow" in the three-phase traffic theory [B. Kerner 2002].

Traffic flow can be described by local and global flow variables, where local variables are measured on a specific location and are for instance:

- Traffic flow count: $q = \frac{n}{T}$, where n is the number of vehicles that pass a cross-section and T a unit of time (time interval);
- Traffic density: $k = \frac{n}{X}$, where n is the number of vehicles that are on segment of length X at a specific time point;
- Nr of vehicles: $r = \frac{n}{X}$, where n is the count of vehicles on segment X at a specific time point;

- Average speed: $s = \frac{o}{X}$, where o is the average speed of all vehicles on on segment X at a specific time point.

Global variables describe for the whole network and are extracted from the full data, ie., if a full simulation run is possible, such as produced by traffic simulation tool such as SUMO. They make a good choice for overal performance measurement. Global variables are for instance:

- Average speed: average speed of all vehicles in a run;
- Average delay: average delay in ms of all vehicles;
- Nr of stops: number of stops of all vehicles.

Global and local variables only give a snapshot of the traffic flow, hence, more elaborate models are need to capture the flow over a certain time period. A common classification of models is based on the dimension of representation (vehicle or flow based) and the granularity of behaviour rules (microscopic vs. macroscopic), which leads to three classes of models:

- Macroscopic flow models that use mathematical theories for describing the traffic flow. For instance, bottleneck nodes and queuing theory are used to calculate the capacity change on specific nodes over time.
- Microscopic flow models are based on the behaviour of individual vehicles, which may emerge from an internal agent model that reacts on external stimuli such as other vehicles, e.g. a car-following model. These flow models also build the foundation of traffic simulations such as SUMO or Vissim.
- Mesoscopic flow models were developed to fill the gap between macroscopic and microscopic models, where different levels of aggregations are introduced, but also the behaviour of individual vehicles or traffic control systems are represented.

4 Problem Description

Now, we outline the main problems that need to be addressed in the ITS use case. Goal *G1* includes the detection of specific events such a traffic jams, the collections of process informations such as traffic statistics. Goal *G1* os the main focus of this work, with the aim of the finding of re-configuration strategies for signal plans, which increase the traffic flow if applied to specific situation. We face the following challenges:

- Traffic can be modelled in different ways, many options of representation are available, however, we desire a compromise between level of details and efficient computation of changes;
- Signal plans and changes in the plans have to be modelled in detail, where hard and soft contraints can be occur;

- Changes in signal planes should not be random, but have to be based on one or more “intelligent” optimization strategies.

After evaluating the possibilites to directly encode a microscopic flow model, we came to the conclusion that the granularity of these model is to fine, which will lead to large ASP models. Therefore, we will base our work on mesoscopic flow models, which still give us enough modelling power to express our problem, but will have smaller models.

The figure below shows a road network in SUMO with two intersections that connect three roads (one horizontal and two vertical) with two incoming and two outgoing lanes for each road.

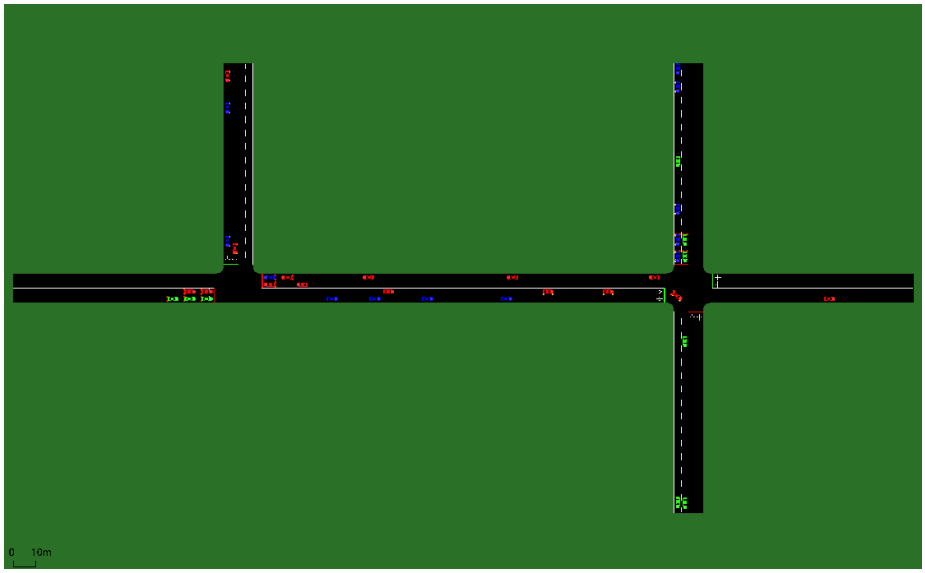

Figure 1: Example of SUMO Model

5 Mesoscopic Traffic Flow Model

The mesoscopic traffic flow model covers the elements of a microscopic and a macroscopic flow model such as (1) the static traffic network and intersection topologies, (2) dynamic states such as vehicle counts and signal phase states, as well as (3) the transformation in the dynamic states. It is accordingly composed of a static component defining the road network and local topologies, and a dynamic component defining traffic flow, which is based on (a) a time model using a timeline, data items of aggregations, and streams, (b) traffic flow generation for different node types, to capture the number of vehicles present at different time

points; and (c) signal phase plans, to capture the generation of signal phases at different time points.

The figure below shows an extracted mesoscopic flow model of Figure 1, which includes nodes for intersections, links, sources, and sinks.

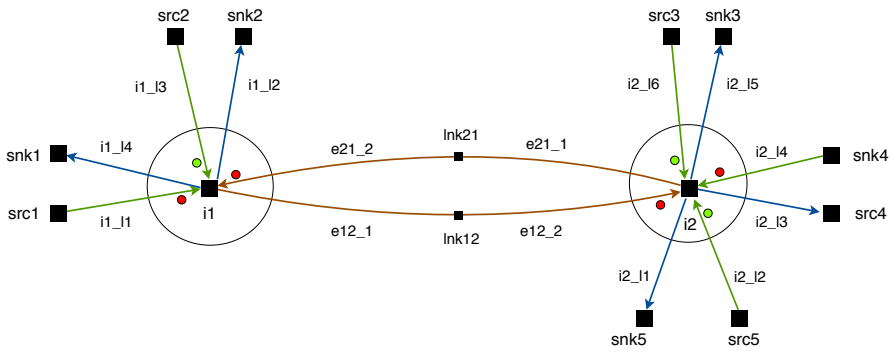

Figure 2: Mesoscopic Traffic Flow Model

5.1 Static component

The static traffic model describes the structure of the road network including the roads, lanes, intersections, intersection topologies, and traffic control installations such as traffic lights (TLs), which have one or more TL signal groups attached. TL controller assigns red (for stop) and green (for go) states to signal phases. We assume for our setting that we have one global TL controller that manages all intersections. The signal phases for each signal group are encoded in a "default" signal plan, where the green/red split of a full phase length is defined. The static component of road network (with several intersections) is represented by a directed, labeled graph as follows:

- The nodes represent real entities (e.g., intersections) or virtual (e.g., boundary to external environment) "crossing points". Every node has a fixed type assigned, where the type can be a source node, sink node, intersection node, and link node;
- The edges represent entire lanes or segments of lanes, and define the (possible) traffic flow in the network by connecting the nodes;
- Node and edge labelings and provide contextual information including the bottleneck capacity to each node (highest possible throughput of vehicles) and the maximum load capacity (maximum capacity of vehicles that can simultaneously on an edge);
- The traffic flow distribution on intersection nodes, defining the ratio on how incoming traffic flow (by vehicles) splits into outgoing traffic flow;

- Implicit denial constraints on intersections to ensure that orthogonal crossing lanes are not in a green phase at the same time point.

The static component is extracted from a given traffic simulation model (such as one generated by SUMO). A suitable segmentation algorithm has to provide the following steps:

- segmentate the road network and reduction it into equi-distant edges,
- add new source/sink nodes for the boundary of the network,
- add link nodes between edges, and
- reduce each intersections to a single nodes.

5.2 Dynamic component

The dynamic component extends the static component with temporal information related to single simulation steps also called time points. The temporal information consists of two kinds of streamed data items which might change in each simulation step: (a) the number of vehicles (NrV), which is the count of vehicles on a specific edge, and (b) the signal phase states (SPS), which is the TL status of specific incoming lanes on (intersection) nodes at a specific time point. Note that the data includes observed data extracted from the simulation model, but also values derived from the extracted data.

The dynamic component includes traffic flow (TF) calculation steps that bring dynamicity into our model. It generates NrV data items based on the classes of nodes, and assigns them at the next time point to edges in the network. Sink nodes play a special role and accumulate the incoming data items, but do not generate data items themselves. A TF calculation is performed at each time point and adds for each edge in our network the new data items to the data streams. It is based either on the incoming edges for intersection and link nodes, or on a predefined generation function for source nodes. The result of each node's calculation are new data items that are propagated to the outgoing edges, updating their state. The state of an edge is captured by the set of data items attached to it on a specific timepoint. Simplest is the calculation of link nodes, as one incoming NrV data item generates a single outgoing data item. Intersection nodes are more difficult to handle, as incoming data items must be split into more outgoing data items according to the TF distribution to the outgoing edges. In Figure 3, we illustrate three calculation steps for four time points represented by the dotted arrows.

Signal phase states (SPS) can be defined similar to TF streams, but only consider SPS data items. Signal phase states, shortly signal states, for traffic lights are defined in signal phase plans, where we assume that each intersection has one plan assigned that covers all the signal groups and lanes of the intersection.

In Figure 4, we give an example of a signal plan as used in SUMO. The length of a full cycle is the sum of durations (60 in our example). Each signal group state

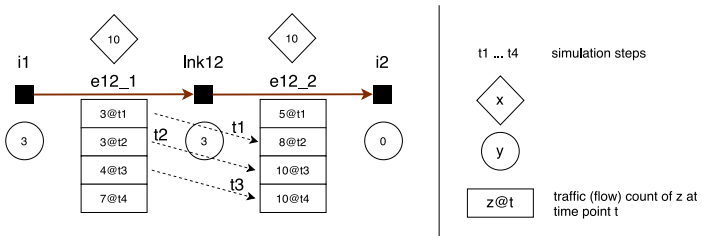

Figure 3: Flow Calculation Step

is shown in state, where "r" , resp., "G", represents red, resp., green phases for the duration. In this encoding, each position is an index and assigns the state to a signal group, i.e., the first position defines the state of the signal group for first lane. The plan also shows implicit constraints, e.g., that the groups at position 1-4 and 5-12 should not be green at the same time.

```
<tlLogic id="I1" type="static" programID="0" offset="0">
    <phase duration="30" state="rrrrGGGGGGGG"/>
    <phase duration="30" state="GGGGGrrrrrrr"/>
</tlLogic>
<tlLogic id="I2" type="static" programID="0" offset="0">
    <phase duration="30" state="GGGGGGrrrrrrGGGGGGrrrrrr"/>
    <phase duration="30" state="rrrrrrGGGGGGrrrrrrGGGGGG"/>
</tlLogic>
```

Figure 4: Example of SUMO signal plan

5.3 Encoding in Datalog

We illustrate the ASP encoding for the signal plan optimization. The encoding is divided into two programs P_I , which contains the data encoding, and P_G, which contains the guess-and-check problem encoding.

We start with the ASP encoding of the traffic network graph, where nodes are represented by predicates *node_src* for a source node; sink, link, and intersection nodes are encoded similarly. The predicate *link* is used to define each edges of the graph and is used to constructs the structure of the traffic network. The label functions bottleneck l_V, resp., load capacities l_E are used to assign the maximum flow in a node, resp., the maximum capacity of an edge. In our examples these functions are uniform, hence the constants *const max_capacity* and *const threshold_red* are used. Traffic distribution in intersection nodes is modelled with the predicate $ratio(n1, n2, r)$, where $n1$ is the incoming and $n2$ the outgoing edge and r is the percentiles of the total nr. of vehicles moving from $n1$ to $n2$. Safety conflicts for traffic light phases are modelled using $conflict(J, X, Y)$, stating that signal phase for the lanes X and Y are not allowed to be green simultaneously.

6 Progress Report on Integration

We recall the elaborated integration of fog side, where the stream reasoner is located, and the cloud side, where the configurator is located: First, every re-configuration of signal plans is triggered by events detected by the stream reasoner that monitors the traffic flow. Second, the mentioned streamed data items (by nr. of vehicles) are created also be the stream reasoner by aggregating the raw stream data, whereby the data items are sent using the process information channel. Third, the result of the re-configuration is fed directly to the signal controllers on the fog side.

Since a full integration of the above three steps is a lengthy task and needs the full development of all involved components, i.e., a stream reasoner for monitoring, a communication platform that support all channels, a controller that manages the re-configuration, and an extension of the signal controller to update new signal plans. Hence, we report on an initial implementation, where we materialize all the streamed data items based on the output of SUMO simulation. This is performed as a preprocessing step and postprocessing step, where the results are fed back the to SUMO simulation, which currently triggers a restart of the simulation. Note that we plan to provide a full integration, where the pre- and postprocessing is happening while the simulation is running.

The initial implementation allows us to perform a first set of experiments, which includes the evaluation of the mesoscopic traffic flow model and should prove that reconfigurations based on this model lead to an increase in traffic flow.

7 Test Environment and Experiments

The test environment is based on the traffic simulation SUMO, which builds the input the data for the static model and parts of the dynamic model. We also use SUMO for conducting the initial experiments, which should show that the mesoscopic traffic flow model approximates a SUMO simulation sufficiently.

7.1 Test Approach

Every use case instance 'I' comprises information about:

- Use case maps/road topologies: lanes, their lengths, crossings, positions of traffic lights, signs, etc.
- Initial traffic situation, which is described by settings of car spawn points, e.g. number of cars generated per time interval, random seeds for generators, routes of cars, dynamic conditions (parking), etc.
- Strategy regulating traffic conditions, e.g. timers of traffic lights, temporary road signs, etc., during the simulation run.

There are two approaches to evaluation of use cases: static and dynamic. The workflow of in the static case is as follows:

1. Given a use case instance 'I' SUMO executes the simulation and outputs an event log
2. The log together with the instance 'I' is provided to the reconfigurator which outputs a strategy 'S' and a start time point 'T'
3. SUMO takes the instance, starts the simulation using the strategy defined in the instance and changes it to 'S' starting from 'T'

Detection of events in this case is performed by a stream reasoner on the fog side, which allows us to query, e.g., an average number of cars or their average speed in some lane within a given time interval. The aggregated data is then provided to a decision component (cloud side), such as Ticker or similar, which makes a decision if some complex event, e.g. an traffic congestion or an accident, is observed.

Opposite to the static case the dynamic one acts during the simulation. That is, SUMO should periodically output logs to the database and query for a new strategy. The workflow suggests that the event recognition and reconfiguration run in parallel to the simulation and can periodically influence the decision strategy of SUMOs.

7.2 Test Cases

The idea of the test cases is to understand the nature of SUMO simulations and the actions which can influence the traffic situation during the simulation.

7.2.1 [TC1] Reproducibility of Evaluation Method: Determinism and Seed of Simulation

The SUMO simulation tool is essential for evaluation of all use cases, and in this test case it is the only part of the representation (model) of the domain that we want to test.

Preparation: identify a map and initial configuration that produces a traffic jam.

Test procedure:

1. Run this map and this traffic configuration 100 times in SUMO and record the log, starting with
   (i) the same initial random seed, and with
   (ii) different initial random seeds.
2. Compare the correspondence between the logs to quantify how much randomness influences the result.

3. The test is successful if with same random seed we achieve the same resulting log over all runs, and if with non-synchronized random seed we achieve

    - a standard deviation of less than X cars/sec of car throughput over all runs,
    - a standard deviation of less than X km/h of average car speed over all runs,
    - a standard deviation of less than X cars/sec in standing state over all simulation runs.

7.2.2 [TC2] Reproducibility of Evaluation Method: Influencing Traffic Lights: Predefined

Influencing traffic light controllers in SUMO is essential for evaluation of all use cases. This test case verifies that we can do this in a reproducible way.

Preparation: identify a map and initial configuration with a bit of traffic jams and build a traffic light controller module M that can modify the switching plan of one or all traffic lights.

Test procedure:

1. Run this simulation starting from the same random seed and record the log

    (i) without traffic light influencing module,
    (ii) with M setting all traffic lights to red,
    (iii) with M doubling the length of the green phase of one traffic light of one lane L,
    (iv) with M halving the length of the green phase of one traffic light of one lane L.

2. Compare parameters of the simulations to see which effect the module had.

3. The test is successful if

    - Car throughput is lowest in (ii).
    - Car throughput over the lane L is higher in (iii) than in (i).
    - Car throughput over the lane L is lower in (iv) than in (i).

7.2.3 [TC3] Reproducibility of Evaluation Method: Influencing Traffic Lights: Online

Influencing traffic light controllers while the simulation is running would be the most realistic way of evaluating the use case. This test case verifies that we can do this in a reproducible way.

Preparation: as in TC2 but prepare a traffic light controller module M that has some API from which an external tool can modify the switching plan without predefining it before the simulation starts. Also some API call should exist in SUMO to run simulation until time step X and then wait (to interact with M) and then continue the simulation after M has done the required changes.

Test procedure:

1. Run this simulation 10 times starting from the same random seed and record the log
   (i) without interacting with M,
   (ii) without interacting with M but freezing time after step T, waiting 1 second, and continuing,
   (iii) with freezing time after step T, injecting the same change as in TC2 (iii) and continuing simulation,
   (iv) with freezing time after step T, injecting the same change as in TC2 (iv) and continuing simulation.
2. Compare parameters of the simulations to see which effect the module had.
3. The test is successful if
   - Standard deviation of car throughput is below X in the logs of (i), the logs of (ii), and the logs of (i)+(ii),
   - standard deviation of car throughput over the lane L is below X in (iii) and (iv), and
   - car throughput over the lane L is higher in (iii) than in (i) and lower in (iv) than in (i).

7.3 Experiments

The experiments evolve with the progress of our implementation. We update this section and add results of new experiments on-going. Furthermore the encodings an logs are available at the webpage http://www.kr.tuwien.ac.at/research/projects/loctrafflog/pais2020.

7.3.1 [TC1] Reproducibility of Evaluation Method

In the first experiment, we aim to show that SUMO simulations run are deterministic, if the same initialization seed is used. For this, we conducted a set of experiments, where we compare the simulation output of (a) restarting the simulation 100 times with the same seed, and (b) restarting the simulation 100 times with random seeds. From this experiment, we can conclude that the same seed leads to the same simulation output, and if a change in seeding occurs, also the result of the simulation output changes.

7.3.2 [E1] Mesoscopic Flow Model compared to SUMO Simulations

This experiment has as the aim of evaluating the quality of our mesoscopic flow model, which we encoded in ASP. For this, we use an experimental setting with a predefined traffic network (show in Figure 1) and the following traffic patterns:

- light (few vehicles),
- medium (more vehicles with free flow), and
- heavy (traffic jam) traffic.

This setting was used to conduct 100 runs with SUMO and 100 runs with the ASP-based model, where the traffic throughput in terms of nr. of vehicles over a fixed period was compared. The results indicate that ASP-based model approximates the SUMO simulation with a deviation of about 10%. Interestingly, the deviation seems higher with the low traffic patterns, since a single vehicle could lead to a larger deviation.