TRANSPORT PROBLEMS 2011
PROBLEMY TRANSPORTU Volume 6 Issue 2

traffic flow, modelling, simulation, microscopic,
traffic measurements, TRANSIMS, SUMO

Michał MACIEJEWSKI
Institute of Machines and Motor Vehicles, Poznan University of Technology
ul. Piotrowo 3, 60-965 Poznan, Poland
Corresponding author. E-mail: michal.maciejewski@put.poznan.pl

JUNCTION TURNING RATIOS ROUTE PLANNER MODULE FOR TRANSIMS

Summary. The paper presents Junction Turning Ratios Route Planner (JTRRouter) for TRANSIMS that enables to generate fast route plans for vehicles on the basis of traffic measurements conducted at intersections. First, TRANSIMS system was introduced. Afterwards a general design, functionality, and requirements of JTRRouter were described. The authors presented a sketch of the route planning algorithm and discussed its computational complexity. In order to illustrate the most essential capabilities of the module, the authors presented an example of route planning for a fragment of a real urban network.

MODUŁ PLANOWANIA MARSZRUT DLA SYSTEMU TRANSIMS NA PODSTAWIE ROZKŁADÓW NATĘŻEŃ RUCHU NA SKRZYŻOWANIACH

Streszczenie. W artykule przedstawiono moduł planowania marszrut (JTRRouter) dla systemu TRANSIMS na podstawie rozkładów natężeń ruchu na skrzyżowaniach. Moduł ten pozwala na szybką generację planów marszrut przy wykorzystaniu wyników pomiarów. W pierwszej kolejności przedstawiono system TRANSIMS, a w dalszej części opisano ogólny projekt, funkcjonalność oraz wymagania modułu JTRRouter. Następnie autorzy zaprezentowali szkic algorytmu planowania marszrut i omówili jego złożoność obliczeniową. W celu zilustrowania najistotniejszych możliwości aplikacji autorzy przedstawili przykład planowania marszrut dla fragmentu rzeczywistej miejskiej sieci drogowej.

1. INTRODUCTION

In order to obtain simulation results that are reliable and consistent with the real traffic, besides a precise network model, a correct traffic generation method is required. Such a method may be based on data derived from traffic flow measurements performed at intersections. Unfortunately, despite great functionality and versatility of TRANSIMS [1,2], it does not contain a tool that performs route planning on the basis of intersection turning ratios. To overcome this limitation a special module, JTRRouter for TRANSIMS (Junction Turning Ratios Router for TRANSIMS), was implemented in Java. The paper presents more detailed information about assumptions, constraints, and functionality of the module. Furthermore, the input data formats are presented and light is shed on the details of the implemented route planning algorithm.

2. TRANSIMS

TRANSIMS (TRansportation ANalysis and SIMulation System) is a free integrated simulation system that enables a regional analysis of transportation systems. It supports the whole process of transportation modeling and simulation from population synthesis, through activity generation to traffic microsimulation. The process is usually run iteratively in order to obtain system equilibrium according to the first Wardrop's principle [3]. Additionally, it is possible to perform estimation of emissions on the basis of the microsimulation results.

TRANSIMS consists of several modules, each one responsible for a certain stage within the modeling and simulation process flow (Fig. 1):

- Population Synthesizer – creates a population of synthetic individuals according to detailed demographic statistics and other data sources,
- Activity Generator – generates daily activity plans for each synthetic member of the population; daily plans consist of work, shopping, school, and other kinds of activities,
- Route Planner – chooses transport modes and plans routes for the individuals on the basis of the previously generated daily plans,
- Traffic Microsimulator – simulates traffic according to the planned routes of the synthetic individuals,
- Feedback Controller – manages dataflow within the framework; controls the relaxation process by running iteratively the former three modules in order to find the system equilibrium,
- Emissions Estimator – estimates the emission of selected exhaust pollutions.

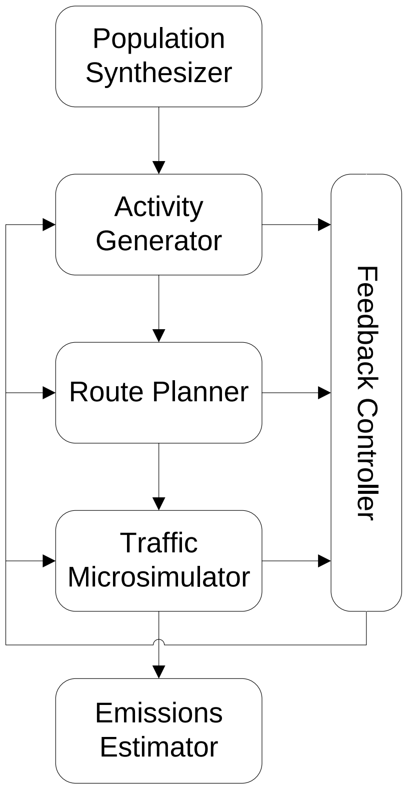

Fig. 1. Standard modeling and simulation process flow in TRANSIMS
Rys. 1. Standardowy przebieg procesu modelowania i symulacji w systemie TRANSIMS

The research was carried out with TRANSIMS version 4.0.6.01 (containing Traffic Microsimulator module version 4.0.75).

3. JTRROUTER FOR TRANSIMS

3.1. Idea

Unfortunately, not always is the original process flow desirable or even feasible. When detailed measurements of traffic flow are available, the route planning procedure can use the measured data to construct routes that emulate the real traffic. In such cases, one may skip the first two stages, the population synthesis and the activity generation, and start with the route planning. Moreover, the Feedback Controller may be skipped as well. In consequence, the process flow is non-iterative and very straightforward (Fig. 2).

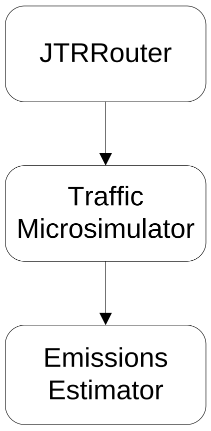

Fig. 2. Simplified modeling and simulation process flow in TRANSIMS
Rys. 2. Uproszczony przebieg procesu modelowania i symulacji w systemie TRANSIMS

In this paper, the authors consider an application of traffic volume counts performed at intersections in planning vehicle routes. A similar approach was implemented in JTRROUTER program, which is a part of SUMO microsimulation system [4,5]. Therefore, the route planner module, implemented by the authors, was called Junction Turning Ratios Router for TRANSIMS, or simply JTRRouter, after its equivalent in SUMO.

3.2. Network model

An application of JTRRouter imposes some constraints on a network model. To illustrate them, let us consider a network presented in Fig. 3. The network is built of nodes (presented with identifiers) and two-way links. Nodes 1,…,6 are intersections, and 10,…,18 are boundary nodes, where each route starts and ends. All left turns at intersection 6, and two left turns at intersection 4 (6→4→3 and 2→4→14) are prohibited. The network model represents a fragment of a real road network in Poznan, Poland, and was used in simulation research [6,7].

However, as the general assumption in TRANSIMS is that each route starts and ends at parking lots, at each boundary node two parking lots must be located, one parking lot for an incoming flow and the other for an outgoing flow (denoted by odd and even numbers, respectively). The parking lots should be placed on the opposite roadsides (as shown in Fig. 4). Usually, these parking lots are virtual, having no counterpart in the real network, and therefore, should be specified as of “boundary” type.

3.3. Input and output files

JTRRouter requires two input XML files (**flows.xml** and **turns.xml**) containing data derived from the traffic measurements, and produces two output text files (**Plan.txt** and **Vehicle.txt**) necessary for the microsimulator module.

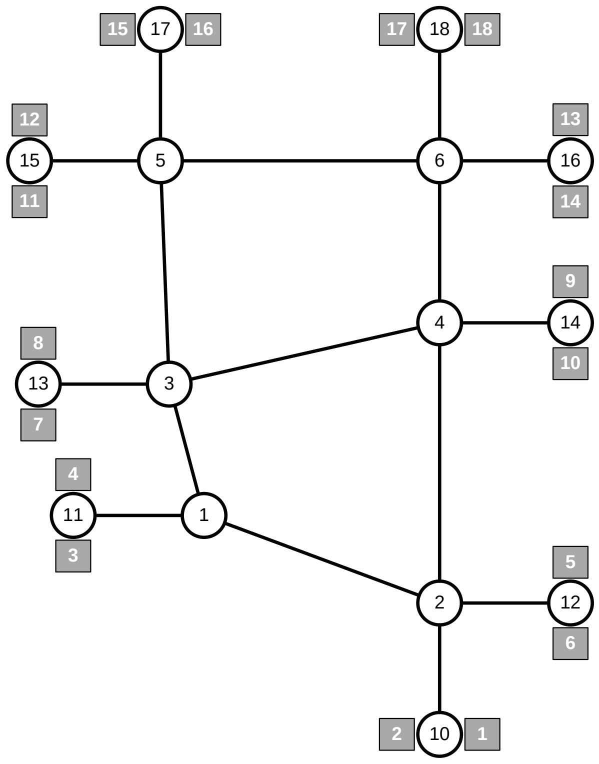

Fig. 3. Model of the road network with the parking lots
Rys. 3. Model sieci drogowej z parkingami

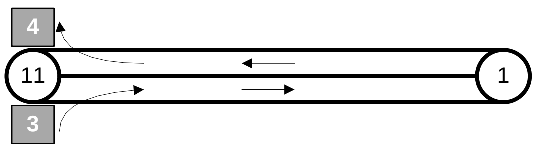

Fig. 4. Location of the parking lots for the incoming (3) and the outgoing (4) flows at boundary node 11
Rys. 4. Lokalizacja parkingów dla przepływów wchodzących (3) i wychodzących (4) przy węźle granicznym 11

The first input file, **flows.xml**, contains flow definitions specified for each of the boundary nodes (nodes 10-18 in the example). Each incoming flow consists of vehicles of different types and subtypes, therefore, its definition includes flow rates for all possible type-subtype pairs.

Fig. 5 shows a fragment of **flows.xml** file. The first line defines simulation period in seconds (*startTime* and *stopTime*) and a flow coefficient (*flowCoeff*) used for a proportional increase/decrease of all flow rates. Lines 2-7 specify flow rates for boundary node 11 (Fig. 4). According to line 2, at node 11 two parking lots are located:

- boundary parking 3 for the incoming flow of vehicles (*inParking*),
- boundary parking 4 for the outgoing flow of vehicles (*outParking*).

Each vehicle starting at node 11 (parking 3) moves toward node 1 (*next*). The next four lines (lines 3-6) define flow rates (*no*; in vehicles per hour) for each vehicle type-subtype pair.

Traversing through the network, at each intersection inlet, a vehicle/driver selects one of all allowed maneuvers. The selection may be stochastic or deterministic, and depends on the measured turning ratios. The turning ratios for each intersection inlet are specified in the second input file, **turns.xml**, where, for each inlet, a set of possible outlets (i.e. next nodes) with their selection probability is defined.

```
<flows startTime="0" stopTime="7200" flowCoeff="1.2">
  <flow node="11" inParking="3" outParking="4" next="1">
    <vehicle type="1" subtype="0" no="700"/>
    <vehicle type="2" subtype="0" no="30"/>
    <vehicle type="2" subtype="1" no="20"/>
    <vehicle type="5" subtype="0" no="10"/>
  </flow>

  <!-- other “flow” entities -->

</flows>
```

Fig. 5. Fragment of flows.xml file containing flow definitions
Rys. 5. Fragment pliku **flows.xml** zawierającego definicje przepływów

Fig. 6 illustrates possible turning maneuvers for vehicles approaching intersection 1 from boundary node 11. Each driver selects the next node (nodes 2 or 3) according to the defined ratios. In a snippet of **turns.xml** file presented in Fig. 7, the ratios were defined as 0.671 and 0.329 for choosing node 2 or 3, respectively.

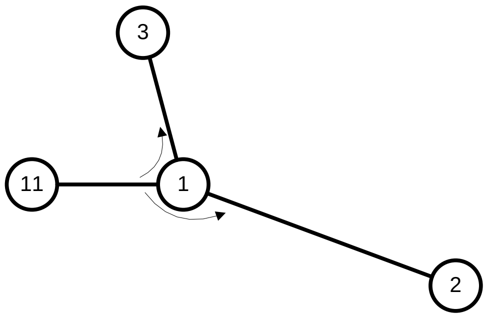

Fig. 6. Possible turning maneuvers at intersection 1 for vehicles approaching from node 11
Rys. 6. Możliwe manewry skrętu na skrzyżowaniu 1 dla pojazdów przyjeżdżających z węzła 11

```
<turns>
  <turn curr="1" prev="11">
    <next id="2" probability="0.671"/>
    <next id="3" probability="0.329"/>
  </turn>

  <!-- other “turn” entities -->

</turns>
```

Fig. 7. Fragment of **turns.xml** file containing turning ratios definitions for each intersection inlet
Rys. 7. Fragment pliku **turns.xml** zawierającego definicje rozkładu manewrów dla każdego wlotu skrzyżowania

One should note that the current version of JTRRouter for TRANSIMS, as well as its equivalent program in SUMO, does not provide the distinction of a vehicle type in turning ratio definitions.

The following files are the output of JTRRouter:

- **Plan.txt** – contains detailed information about all planned routes (departure parking lots, departure times, lists of visited nodes, etc.),
- **Vehicle.txt** – contains all vehicle specifications.

The two above-listed files, together with network definition files and a vehicle type/subtype definition file, are necessary and sufficient for running Traffic Microsimulator module.

4. ROUTE PLANNER ALGORITHM

The general schema of the route planner algorithm is presented in Fig. 8. The main part of the algorithm (lines 6-18) consists in finding all acceptable routes, and assigning them (with the previously calculated selection probabilities) to each generated vehicle.

```
1.    READ startTime, stopTime, flowCoeff
2.    READ flows
3.    READ turns

4.    INIT plans                                       {list of plans}
5.    INIT vehicles                                 {list of vehicles}

6.    FOR EACH f IN flows
7.        routes = findRoutes(f)             {all acceptable routes;
                                                      without loops}

8.        FOR EACH r IN routes
```

Fig. 8. Sketch of JTRRouter algorithm
Rys. 8. Schemat algorytmu modułu JTRRouter

The procedure `findRoutes` finds all acceptable routes starting at a given boundary node (i.e. for a given incoming flow *f*). A route is acceptable if no link is traversed in the same direction twice. On the other hand, visiting any node twice is allowed. The procedure was implemented as a depth-first search that is run recursively until a boundary node is reached, where a vehicle exits the network, or a link is traversed in the same direction for the second time, which is forbidden.

A roughly approximated upper bound of computational complexity for `findRoutes` procedure is $O(bn^m)$, where:

- b – number of boundary nodes (and boundary links; one two-way link per one node),
- n – number of intersection nodes,
- m – number of one-way links between intersections (two-way links are counted as two one-way links).

It is a very high complexity, but in the case of typical road networks, which are sparse graphs, the upper bound is significantly lower. For a typical road network, with three allowed maneuvers (right turn, straight ahead, left turn) at each intersection, a very rough upper bound is $O(b3^m)$. The actual computational complexity, however, is much lower, because recursion search rarely goes deep, since most search branches are cut on shallow recursion levels. The cuts are caused by reaching one of the boundary nodes, or by traversing a link in the same direction for the second time. Moreover, in reality a lot of turning maneuvers are prohibited (e.g. some left turns, turning into outlets of one-way links), which also shortens computation time.

Since `findRoutes` procedure is called iteratively for each boundary node (i.e. incoming flow *f*), the approximated upper bound for the whole algorithm is roughly $O(b^2n^m)$.

A route selection probability *`r.probability`* (line 9) is equal to the product of probabilities of all turning maneuvers within the route. To improve computational efficiency, route selection probabilities can be computed within `findRoutes` procedure.

All vehicles, for a given incoming flow `f`, are generated according to the measured traffic volume, with distinction of vehicle types/subtypes (line 11). Then, a route and a departure time are assigned to each generated vehicle. However, the assignment procedure (both the route selection and the time determination) may be deterministic or stochastic (lines 12-17).

Fig. 9 shows an example of four routes generated by the route planner algorithm for the network presented in Fig. 3. Tab.1 contains number of routes found by JTRRouter for each boundary node. Despite the high upper bound of computational complexity, the procedure runs fast for typical road networks. In the case of the presented network, it took only a few milliseconds to find all 308 acceptable routes and generate plans for over 6,300 vehicles (excluding I/O operations).

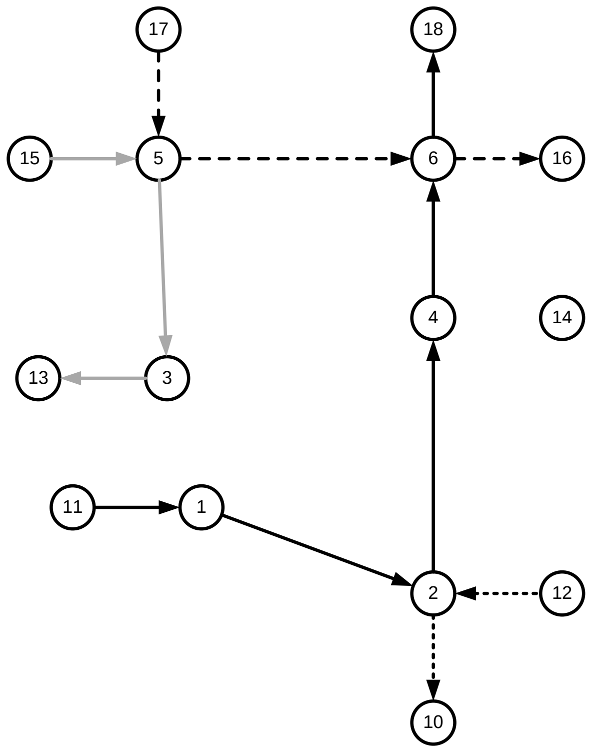

Fig. 9. Example of routes generated by JTRRouter
Rys. 9. Przykład marszrut wygenerowanych przez moduł JTRRouter

Table 1

Number of the routes generated by JTRRouter for each boundary node

Boundary node (incoming flow)	**Number of acceptable routes**
10	23
11	25
12	23
13	40
14	36
15	44
16	27
17	44
18	46
Average	**34,2**
Sum	**308**

5. CONCLUSIONS

The goal of JTRRouter is to enable the application of traffic measurements to the vehicle route planning process. JTRRouter was successfully used by the authors in several research projects [6,7]. As was shown, the implemented algorithm provides fast route plans generation. In the future, it is planned to enable definition of default turning ratios (for intersections without turning ratios explicitly specified), and to facilitate minimization of discrepancies in the traffic measurement data.

References

1. Barrett C.L., et. al.: *TRANSIMS (TRansportation ANalysis SIMulation)* Tech. Rep., LA-UR-99-1658, 1999.
2. *TRANSIMS Open-Source*. http://code.google.com/p/transims/
3. Wardrop J.G.: *Some theoretical aspects of Road traffic research*. In: Proceedings of the Institution of Civil Engineers, vol. 1(3), 1952, pp 325-362.
4. Krajzewicz D., Hertkorn G., Rössel Ch., Wagner P.: *SUMO (Simulation of Urban MObility). An open-source traffic simulation*. In: Al-Akaidi, A. (Ed.): Proceedings of the 4th Middle East Symposium on Simulation and Modelling (MESM2002), September, 2002, Sharjah, United Arab Emirates, pp. 183-187.
5. *SUMO Simulation of Urban MObility*. http://sumo.sourceforge.net.
6. Maciejewski M., Maciejewski M.: *Zastosowanie automatów komórkowych do symulacji ruchu drogowego w mieście*. Logistyka 2/2010, pp. 1159-1168.
7. Maciejewski M.: *A comparison of microscopic traffic flow simulation systems for an urban area*. Transport Problems, Vol. 5(4), 2010, pp. 27-38.

Received 28.03.2010; accepted in revised form 05.05.2011