A Tool for Efficient Derivation of Optimal Signal Schedules for Multimodal Intersections

Nicola Bertocci **Laura Carnevali** **Leonardo Scommegna** **Enrico Vicario**

Dept. of Information Engineering, University of Florence

Software Technologies Lab - https://stlab.dinfo.unifi.it

{*nicola.bertocci, laura.carnevali, leonardo.scommegna, enrico.vicario*}*@unifi.it*

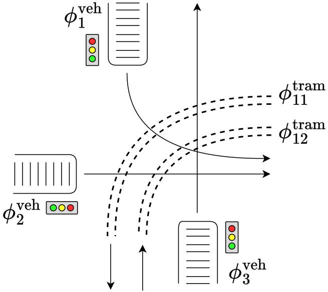
EPEW'24

Venice, June 2024

- this is about:
  - Evaluation of the impact of right-of-way tram traffic over car traffic
  - Multimodal intersection of multiple car flows and tram tracks
  - Omnibus Java Library for compositional intersection analysis
  - . . . enabling efficient identification of optimal signal scheduling

Tramways and Multimodal Car Intersections

- **Pros**: Improve sustainability and urban transportation
- **Cons**: impact availability of car intersection due to right-of-way policy
- **Aim**: evaluate and mitigate the impact of trams in car traffic
- **Challenge**: impact depends on various factors e.g., semaphore scheduling, arrival rate, and periodicity of tram traffic

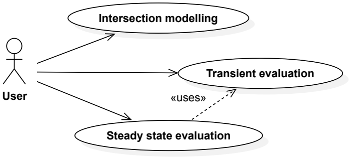


Analysis of Urban Intersection

- Modelling urban transportation systems:
  - Support early assessment and runtime adaptation of design choices
  - **measures of interest**: expected queue lengths, waiting times, expected fuel consumption
- **Microscopic models**:
  - Represent behavior of individual vehicles
  - Capture interactions among vehicles and fine details about driver actions e.g., impatience
  - **Expensive to analyse** and do not scale with queue size
- **Macroscopic models**:
  - Encode aggregated features e.g., density, flow, and average speed
  - Computational efficient
  - **Structurally missing** the representation of synchronous phenomena like tram arrivals

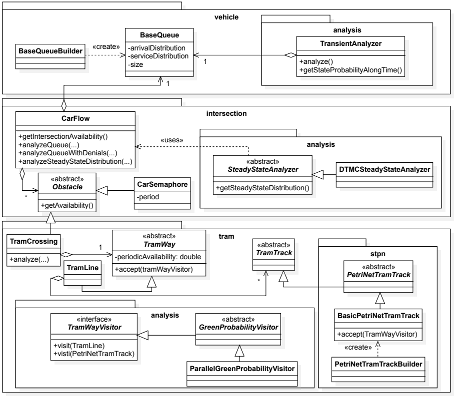
Omnibus Java Library 1/2

- Evaluation of multimodal intersection with right-of-way tram traffic [1]
- **Compositional approach** for efficient intersection analysis
  - **Microscopic Model** for Tram Traffic evaluation with Stochastic Time Petri Nets (STPNs)
  - **Macroscopic Model** for Car Traffic evaluation with finite-capacity vacation queues
- **Representation** of complex intersections with the metamodel
- Calculation of the **intersection availability** over time
- Steady-state distribution of the **number of queued cars** at multiples of the hyper-period (i.e. l.c.m. of tram and car semaphore periods)
- Evaluate the **expected number of queued cars over time**



[1] *available at https://github.com/oris-tool/omnibus under the AGPL v3 licence*

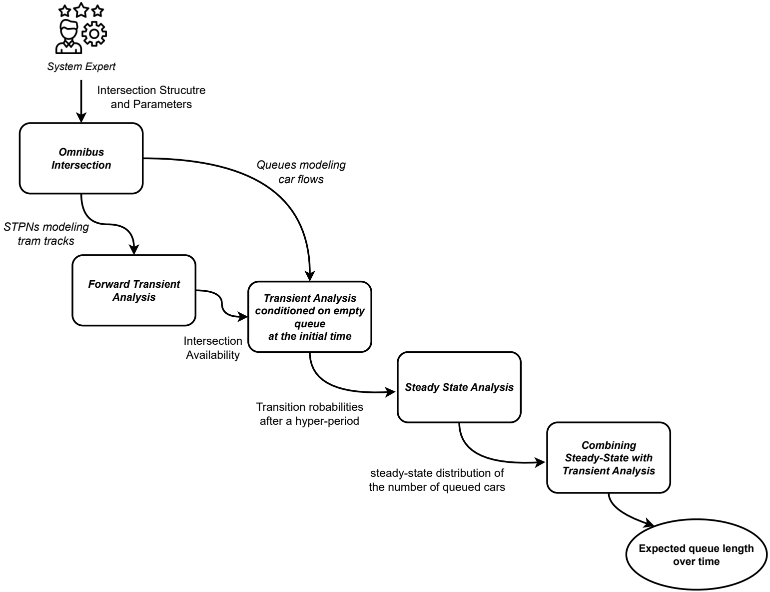
Omnibus Java Library 2/2



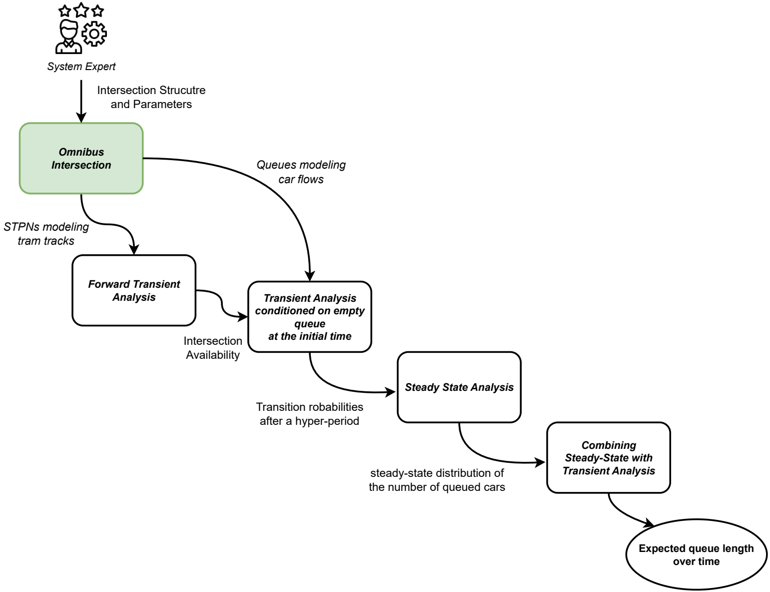
- Designed to facilitate maintainability and extensibility
- Intuitive headless interface
- Easily integrable in complex analysis workflows
  - Analysis of relevant use cases for operation and management of the intersection
  - Already tested to derive optimal signal scheduling[2]

[2] *Bertocci, Carnevali, Scommegna, Vicario. Efficient derivation of optimal signal schedules for multimodal intersections. SIMPAT 2024*

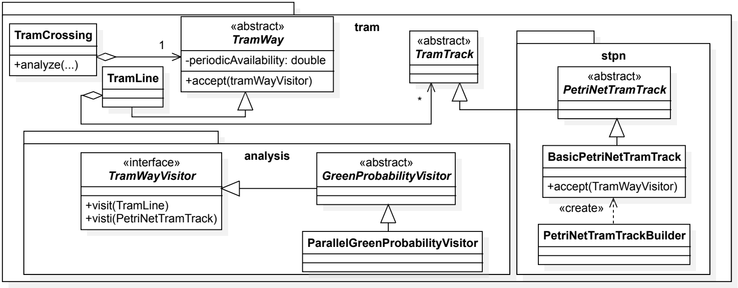
Omnibus Workflow



Omnibus Workflow



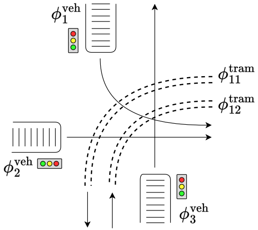
Intersection Modelling in Omnibus - Tramway



- Independent multiple-track **tram lines** with **right of way** over car flows
- DET inter-arrival times with DET offsets (phases)
- Each arrival is characterized by GEN delays possibly with bounded support
- GEN intersection crossing times

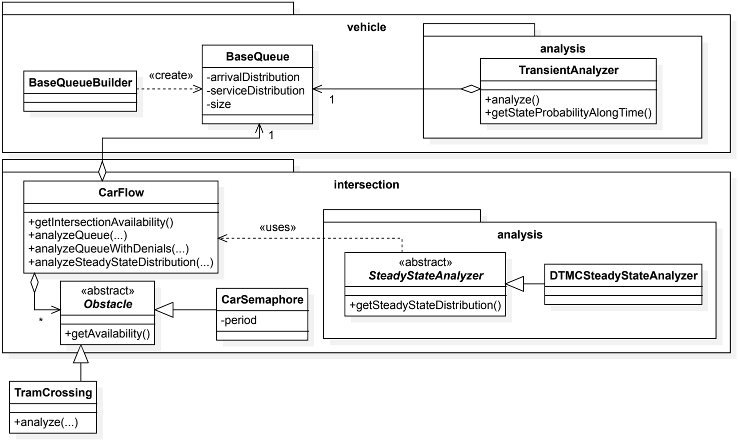
Intersection Modelling in Omnibus - Tramway

parameter	value
tram line period	220 s
Φ_{11}^{tram} offset	0 s
Φ_{12}^{tram} offset	40 s
Φ_{11}^{tram} delay distribution	UNIF(0 s,120 s)
Φ_{12}^{tram} delay distribution	UNIF(0 s,40 s)
Φ_{11}^{tram}, Φ_{2}^{tram} crossing time distribution	UNIF(6 s,14 s)




```java
// track 1 parameter definition
BigInteger t1_periodTime = BigInteger.valueOf(220);
BigInteger t1_phaseTime = BigInteger.ZERO;
BigInteger t1_delayEFTime = BigInteger.ZERO;
BigInteger t1_delayLFTime = BigInteger.valueOf(120);
BigInteger t1_crosslightAntTime = BigInteger.valueOf(5);
BigInteger t1_leavingEFTime = BigInteger.valueOf(6);
BigInteger t1_leavingLFTime = BigInteger.valueOf(14);
// track 1 instantiation
PetriNetTramTrack bin1 = PetriNetTramTrackBuilder.getInstance(t1_name, t1_periodTime,
     t1_phaseTime, t1_delayEFTime, t1_delayLFTime, t1_crosslightAntTime, t1_leavingEFTime,
      t1_leavingLFTime);
// track 2 instantiation
...
// tram line definition
TramLine tramLine = new TramLine("line1");
tramLine.addTramTrack(bin1, bin2);
// tram cross definition
TramCrossing tramCross = new TramCrossing(tramLine);
```

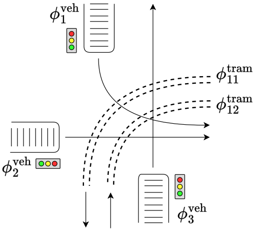
Intersection Modelling in Omnibus - Road Intersection



- Independent single-lane **car flows** associated with a queue
- EXP inter-arrival times, EXP intersection leaving times
- A semaphore with period P associated with every car flow
- semaphore and tram crossing as **obstacles** for the car flow

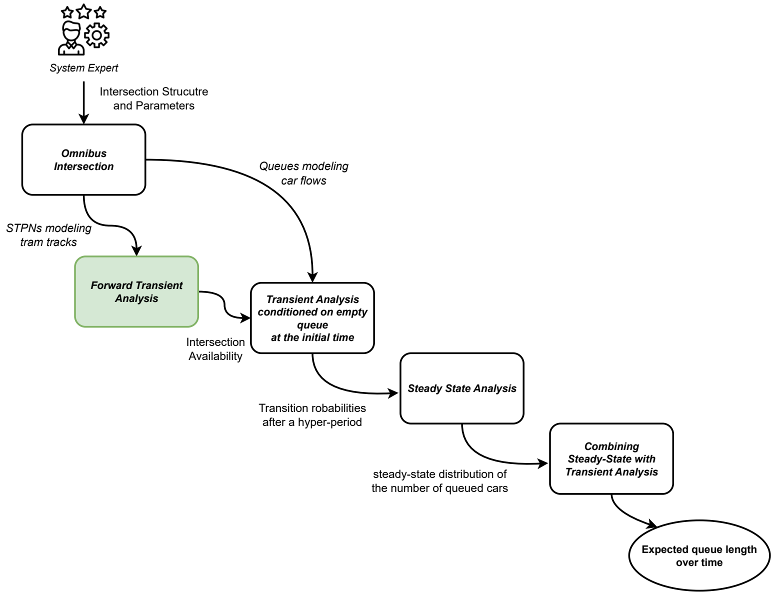
Intersection Modelling in Omnibus - Road Intersection

parameter	value
traffic light period	110 s
Φ_1^{veh} arrival rate	$0.05\,\mathrm{s}^{-1}$
Φ_2^{veh} arrival rate	$0.1\,\mathrm{s}^{-1}$
Φ_3^{veh} arrival rate	$0.15\,\mathrm{s}^{-1}$
Φ_1^{veh}, Φ_2^{veh}, Φ_3^{veh} leaving rate	$0.092\,\mathrm{s}^{-1}$



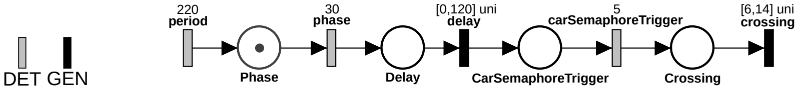

```java
// vehicle flow 1 parameter definition
BigDecimal arrivalRate1 = BigDecimal.valueOf(0.05);
BigDecimal mu = BigDecimal.valueOf(0.092);
BigInteger maxQueueSize = BigInteger.valueOf(31);
BigInteger initialCars = BigInteger.valueOf(0);
// vehicle flow 1 instantiation
CarFlow carFlow1 = new CarFlow("carFlow1");
carFlow1.setQueue(BaseQueueBuilder.getInstance(arrivalRate1,mu,maxQueueSize,initialCars));
// vehicle flow 2 and 3 instantiation
...
// semaphore instatiation with period 110s
CarSemaphore carSem1 = new CarSemaphore(new BigInteger("110"), TIMESTEP);
CarSemaphore carSem2 = new CarSemaphore(new BigInteger("110"), TIMESTEP);
CarSemaphore carSem3 = new CarSemaphore(new BigInteger("110"), TIMESTEP);
// adding obstacles to vehicle flow 1
carFlow1.addObstacle(tramCross);
carFlow1.addObstacle(carSem1);
// adding obstacle to vehicle flows 2 and 3
...
```

Omnibus Workflow



Microscopic model of tram traffic

- Tram track model in terms of **Stochastic Time Petri Net (STPN)**
- Fitting observed data by means of GEN distributions [3]
  - Delay (wrt the nominal arrival time) never observed to be larger than 120 s
    $\Rightarrow$ modeled by a uniform distribution over $[0, 120]$ s
  - Crossing time observed to have mean $\omega = 10$ s and standard deviation $\sigma = 4$ s
    $\Rightarrow$ modeled by a uniform distribution over $[\mu - \sigma, \mu + \sigma] = [6, 14]$ s
  - Any other distribution in the class of **expolynomial** functions could be used
- Deterministic temporal parameters
  - Periodic inter-arrival times with period $T = 220$ s and phase $O = 30$ s
  - Travel time from the wayside system to the intersection 5 s
    (time advance with which car semaphores are triggered red wrt tram arrival)

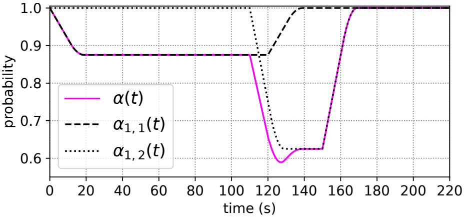


[3] *Carnevali et al, "Stochastic modeling and analysis of road-tramway intersections", Innovations in Systems and Software Engineering, 2020*

Evaluation of the intersection availability for car traffic

- Forward transient analysis of the STPN of each track (separate analyses)[4]
  - Derives the transient probabilities of the reachable markings, which yield the transient probability that no tram of the track is crossing the intersection
- A tram line consisting of two tram tracks
  - $\alpha_i(t)$:=Prob{no tram of track i is occupying the intersection}
  - $\alpha(t)$:=Prob{no tram is occupying the intersection}$=\prod_i \alpha_i(t)$

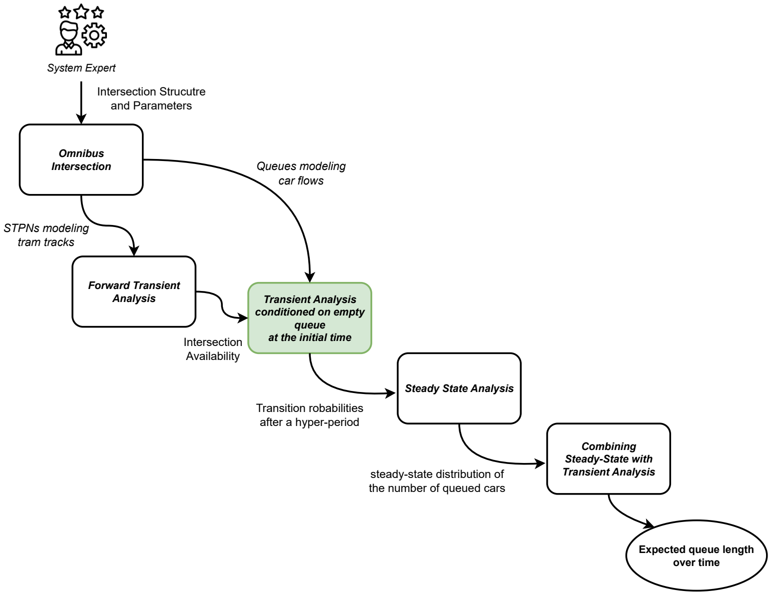
```
// tram crossings analysis
tramCross.analyze(new ParallelGreenProbabilityVisitor(), timeStep);
```



parameter	value
tram line period	220 s
Φ_{11}^{tram} offset	0 s
Φ_{12}^{tram} offset	40 s
Φ_{11}^{tram} delay distribution	UNIF(0 s,120 s)
Φ_{12}^{tram} delay distribution	UNIF(0 s,40 s)
Φ_{11}^{tram}, Φ_{2}^{tram} crossing time distribution	UNIF(6 s,14 s)

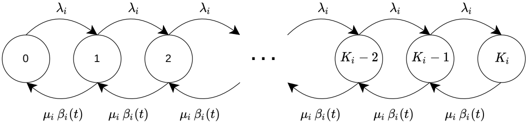
[4] *Stochastic state classes method implemented in the SIRIO library:* *https://www.oris-tool.org/sirio*

Omnibus Workflow



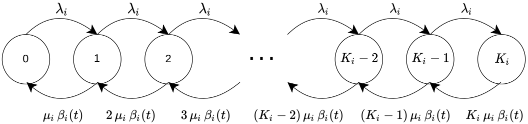
Macroscopic models of car traffic (alternative models)

- M/M **finite-capacity queue** with GEN vacation time
- **Vacation time**: intersection not available due to tram passage or red signal
- $R_{k,j}(t)$:=Prob{j queued cars at t | k queued cars at the initial time}
- $\beta(t)$:=Prob{intersection available at t and car semaphore green at t}
- M/M/1/K queue: the intersection is the server
  - Arrival rate λ, service rate $\mu\,\beta(t)$



$$\begin{cases} R'_{k,0}(t) = -\lambda R_{k,0}(t) + \mu\beta(t)R_{k,1}(t) \\ R'_{k,j}(t) = -\lambda R_{k,j}(t) - \mu\beta(t)R_{k,j}(t) \\ \qquad + \lambda R_{k,j-1}(t) + \mu\beta(t)R_{k,j+1}(t) \\ R'_{k,K}(t) = \lambda R_{k,K-1}(t) - \mu\beta(t)R_{k,K}(t) \end{cases}$$

- M/M/K/K queue: the street preceding the intersection is the server
  - Arrival rate λ, service rate $k\,\mu\,\beta(t)$ in state k

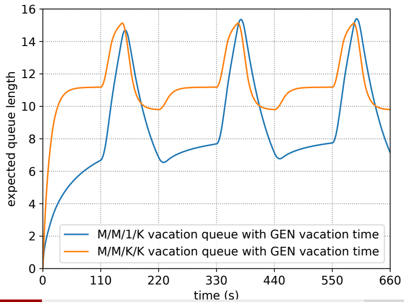


$$\begin{cases} R'_{k,0}(t) = \mu\beta(t)R_{k,1}(t) - \lambda R_{k,0}(t) \\ R'_{k,j}(t) = -j\mu\beta(t)R_{k,j}(t) - \lambda R_{k,j}(t) \\ \qquad + j\mu\beta(t)R_{k,j+1}(t) + \lambda R_{k,j-1}(t) \\ R'_{k,K}(t) = -K\mu\beta(t)R_{k,K}(t) + \lambda R_{k,K-1}(t) \end{cases}$$

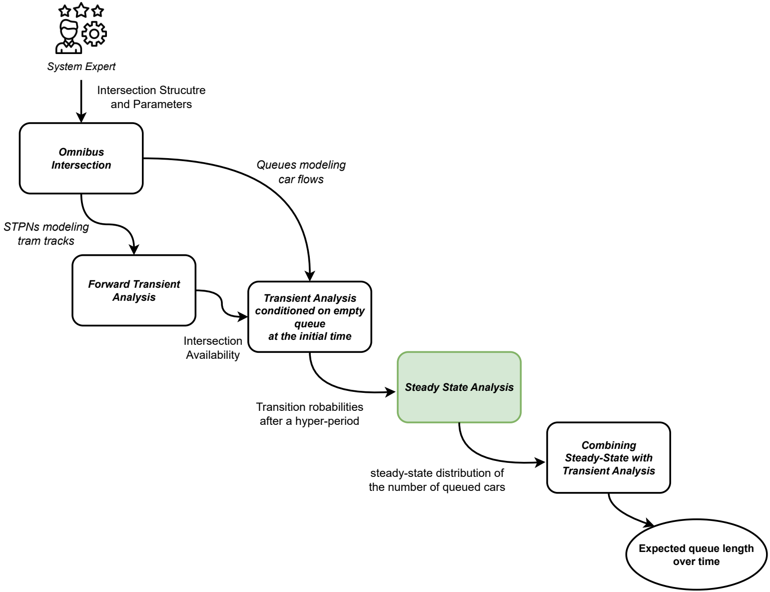
Transient evaluation

- Numerical solution of the system of differential equation by discretization
- Expected number of queued cars: $\bar{Q}(t) = \sum_{k=0}^{K} \text{Prob}\{\text{k initial queued cars}\} \sum_{j=0}^{K} j\, R_{k,j}(t)$
- An intersection with a car flow and a tram line made of two tracks
  - Queue size $K = 31$ (street length 150 m, car length 4.5 m, safe distance 0.3 m)
  - Arrival rate $\lambda = 0.9\,\text{s}^{-1}$ (both M/M/1/K and M/M/K/K)
  - Leaving rate $\mu = 1.138\,\text{s}^{-1}$ (M/M/1/K), $\mu = 0.092\,\text{s}^{-1}$ (M/M/K/K) (intersection length 12.2 m, car speed 50 km h^{-1})

```
// Car Flow transient analysis
double[] expectedState = carFlow.analyzeQueue(new TransientAnalyzer(), BigInteger.valueOf(
    carFlow.getObstaclesHyperPeriod()), timeStep).getExpectedStateAlongTime());
```



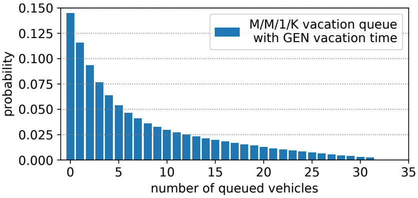
Omnibus Workflow

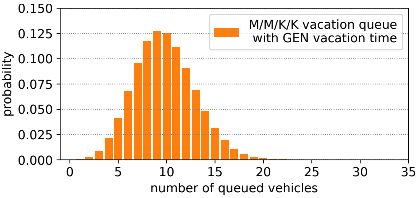


Steady-state evaluation

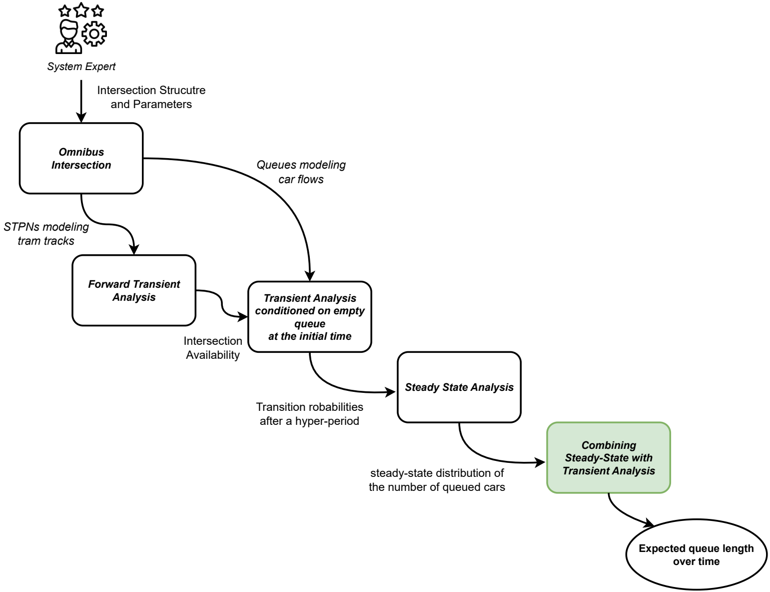
- Steady-state distribution of the number of queued cars at multiples of the hyper-period (i.e. l.c.m. of tram and car semaphore periods)
- Derivation of the embedded DTMC by transient analysis of queue behavior within an hyperperiod
- The embedded DTMC is irreducible, aperiodic, and positive recurrent

```
// Car Flow steady-state analysis
double[] steadyStateDistribution = carFlow1.analyzeSteadyStateDistribution(
                    new DTMCSteadyStateAnalyzer(), new TransientAnalyzer(), timeStep)
                    .getSteadyStateDistribution();
```



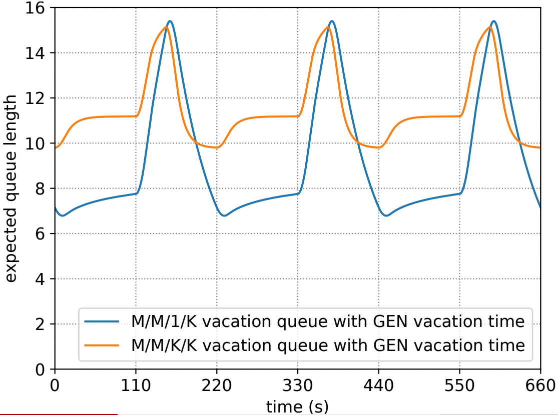


Omnibus Workflow



Combining Steady-state and Transient Analysis

- **Steady state analysis**: steady-state distribution of the number of queued cars at multiples of the hyper-period
- Transient analysis conditioned on the steady-state distribution of the number of queued cars at the initial time
- Evaluation of the expected queue size over **arbitrary-duration intervals**



Validation wrt a microscopic traffic simulator

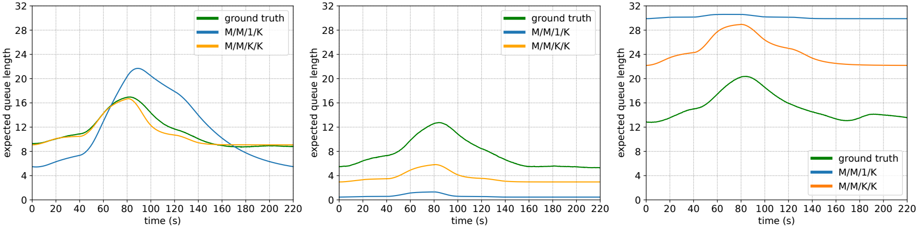
- An intersection with a bidirectional tram line and a single-lane car flow
  - Street length $S \in \{50, 150, 450\}$ m (determines the maximum queue size K)
  - Car arrival rate $\lambda \in \{0.5, 0.9, 1.3\}\,\mathrm{s}^{-1}$
  - Maximum car speed $V \in \{30, 50, 70\}\,\mathrm{km\,h}^{-1}$ (determines the leaving rate μ)
  - Tram period $T = 220$ s, phase 0 s for track1 and 40 s for track2
  - UNIF delay over $[0, 120]$ s for track1 and $[0, 40]$ s for track2
- Ground truth obtained by **SUMO (Simulation of Urban MObility)**
  - 1500 runs for each of the 27 scenarios
  - Total execution time $\simeq$ 11 days
- Expected queue size over time computed by analysis ($\bar{Q}_{\mathrm{A}}(t)$) and simulation ($\bar{Q}_{\mathrm{S}}(t)$): **Normalized Root Mean Square Deviation**

$$\mathrm{NRMSD}(A, S) := \sqrt{\frac{\sum_{t=0}^{N-1} (\bar{Q}_{\mathrm{A}}(t) - \bar{Q}_{\mathrm{S}}(t))^2}{N}} \cdot \frac{1}{K}$$

- **Results**:
  - In the worst case, NRMSD < 0.37 achieved in < 2.7 s
  - $> 80\%$ of the cases, NRMSD < 0.2; $\simeq 50\%$ of the cases, NRMSD < 0.1
  - **Simulation takes tens of seconds** to achieve comparable NRMSD

Validation results

- Expected number of queued cars over time
  - $S = 150\,\mathrm{m}$, $\lambda = 1.3\,\mathrm{s}^{-1}$, $V = 70\,\mathrm{km\,h}^{-1}$ (best case for the M/M/K/K model)
  - $S = 150\,\mathrm{m}$, $\lambda = 0.5\,\mathrm{s}^{-1}$, $V = 70\,\mathrm{km\,h}^{-1}$ (lower arrival rate wrt the best case)
  - $S = 150\,\mathrm{m}$, $\lambda = 1.3\,\mathrm{s}^{-1}$, $V = 30\,\mathrm{km\,h}^{-1}$ (lower car speed wrt the best case)



- Analysis with the M/M/K/K queue reproduces the **ground truth pattern** capturing peaks and troughs of the expected number of queued vehicles
- The analysis is able to **capture relative variations** as the values of parameters change

Omnibus Beyond Intersection Analysis

- Omnibus provides a significantly lower computational load wrt microscopic simulators
- Omnibus analysis accurately captures the car flow patterns
- **Idea**: Exploit Omnibus to find parameters that optimize quantitative measures of interest
- **Example**: Derivation of optimal signal schedules[5]
    - Identification of a state space of 390 possible signal schedule
    - Evaluation of the impact of each signal schedule with both SUMO and Omnibus
    - SUMO requires over 94 hours to explore the overall state space
    - Omnibus requires $\approx 38\,\mathrm{s}$ to explore the overall state space
    - The overall SUMO schedule ranking is comparable to the omnibus schedule ranking
    - The two best schedules derived by our analysis method are the 1-st and the 4-th in the SUMO ranking
    - 7 of the best 10 schedules according to Omnibus are in the best 10 positions in the SUMO ranking

[5] *Bertocci, Carnevali, Scommegna, Vicario. Efficient derivation of optimal signal schedules for multimodal intersections. SIMPAT 2024*

Discussion and Future Directions

- An approach to performance evaluation of multimodal urban intersections
- The approach performs both **transient and steady-state analyses**
  - Enables evaluation with time-varying params over arbitrary-duration intervals
- Validation wrt the results obtained through the SUMO traffic simulator
  - In the worst case, NRMSD < 0.37 achieved in $< 2.7\,\text{s}$
  - In more than 80% of the cases, NRMSD < 0.2
  - In nearly 50% of the cases, NRMSD < 0.1
  - Simulation takes tens of seconds to achieve comparable NRMSD
- The approach is open to **many extensions**
  - Joint evaluation of performance of multiple intersections
  - Bursts and platoons arrival through non-EXP inter-arrival times and service times
  - Evaluation of other quantitative measures of interest