CHAPTER 19

New Dynamic Events-Based Public Transport Router

Sergio Arturo Ordóñez

19.1 Basic Information

Entry point to documentation:

`http://matsim.org/extensions` → eventsBasedPTRouter

Invoking the module:

`http://matsim.org/javadoc` → eventsBasedPTRouter → `RunControlerWS`, `RunControlerWSV`, `RunControlerWW` classes

Selected publications:

Ordóñez Medina and Erath (2013b)

In public transport route choice, decisions and actions of a particular user depend not only on his/her own preferences, like value of time, crowd avoidance or willingness to pay. They also depend on the decisions and actions of many other public transport users, operators and authorities. Even private transport users' decisions are also involved, as everybody shares the same infrastructure.

This implementation of MATSim used a SBPTR, as mentioned above, meaning that when an agent needed a route for a given start time, origin and destination, the SBPTR found the shortest path in a schedule-based network (assuming public transport vehicles are always on time and always have space). Within the mobility simulation, a vehicle could arrive early or late and/or it could be full, thus not allowing additional passengers to board. With a negative result, the agent obtained a bad score and this plan would have probably been replaced with a more favorable one during the iterative learning process. This scenario's problem occurred when the agent tried to

How to cite this book chapter:

Ordóñez, S A. 2016. New Dynamic Events-Based Public Transport Router. In: Horni, A, Nagel, K and Axhausen, K W. (eds.) *The Multi-Agent Transport Simulation MATSim*, Pp. 123–132. London: Ubiquity Press. DOI: http://dx.doi.org/10.5334/baw.19.

find a new route for the same start time, origin and destination, the public transport scheduled network shortest path remained the same; agents could not improve their experiences by changing the route.

To address this shortcoming, a new EBPTR (Events-Based Public Transport Router) was proposed (Ordóñez Medina and Erath, 2013b), modeled, implemented and tested. It took the given schedule as a base for the first iteration, but updated information on travel times, occupancy of the public transport vehicles, and waiting times was propagated between subsequent iterations. Thus, when same day executions were performed, new routes could be generated for the same start time, origin and destination, because the system is remembered delayed bus services (longer travel times), or train services where the vehicle arrived full (longer waiting times). However, the network used to route agents required a new topology to account for such variables. This approach allowed then to account for emergent phenomena; in situations where overcrowded vehicles prohibited boarding, it made sense for some agents to travel a few stops in the outbound direction. They could then transfer to an inbound vehicle with sufficient capacity and board. Although more memory was needed, similar or even better computation times were achieved when shortest path calculations awe performed, due to the simpler network topology. Furthermore, to achieve user equilibrium required a significantly smaller number of iterations.

19.2 Events-Based Public Transport Router

A new EBPTR was developed for MATSim to more realistically model public transport route choice, where agents learn, over time, that transit vehicles are not always on time, do not always have sufficient space to allow boarding and trips with more comfort are often preferable.

Network Topology Figure 19.1(b) shows the structure of the proposed public transport network, compared with the original structure (Figure 19.1(a)). Inspired by the network designed by Spiess and Florian (1989) this implementation had two types of nodes. The first type represented a stop facility (green-black squares) as point in space, while the second type (yellow-red dots) represented a stop-route relation which could be seen as a physical or virtual platform for each line passing a particular stop facility. For example, different platforms in a metro system needed to be modeled as different stop facilities, because different services arrived at each platform and walking paths were needed to change from one platform to another. For bus stop facilities, they represented virtual platforms; in reality, buses from different lines serving the same bus stop would normally use the same physical infrastructure e.g., a bus bay. To connect those nodes, there were four types of links. The in-vehicle links joined two consecutive stop-route nodes in the direction of the correspondent route. The boarding links connected a stop node with each corresponding stop-route node. The alighting links were opposite, connecting stop-route nodes with their corresponding stop node. Finally, walking links connected a stop node with all other stop nodes located within walking distance.

Link Costs Each link in this network had a related time-dependent disutility function. Different costs were saved for different times in the day for a given time bin (at this time, 15 minutes). In-vehicle link disutilities depend on vehicle travel time, travel distance, level of occupancy and a fare rate, if this system is distance-based. Boarding link disutilities depended on waiting times, a transfer cost, and a fixed fare if this system was entry-based; thus it was possible to relate specific stop-route waiting times to these links. As the first waiting link was not a transfer, this cost had to be subtracted from the whole path cost, but this detail did not affect the shortest path calculation.

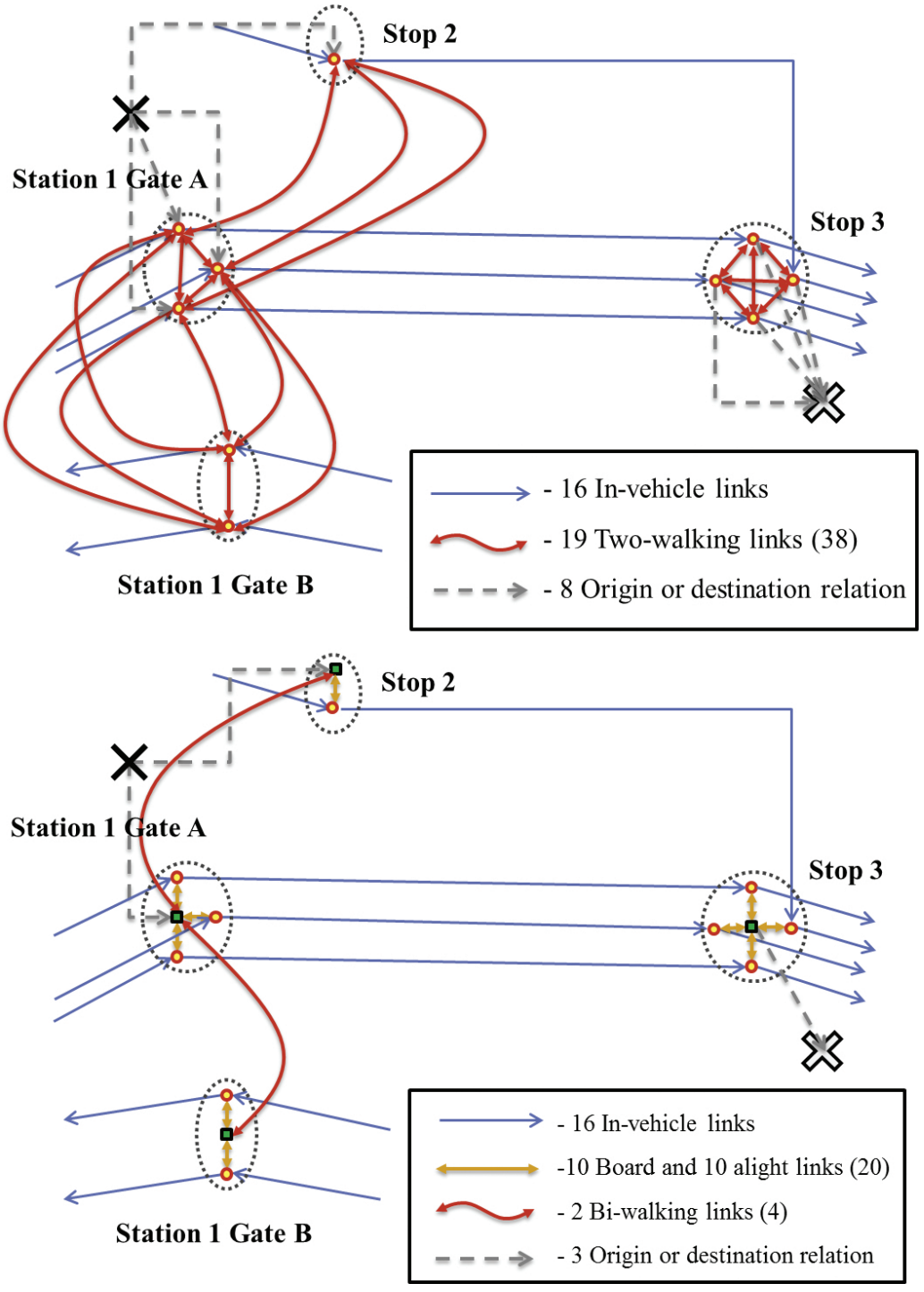

Figure 19.1: Comparison of the network topologies of the schedule-based transit router (a) and the new events-based transit router (b).

Alighting links had no associated cost, but a fare could be related to them. Finally, walking links depended on the walking travel time and distance. Equation (19.1) shows linear versions of these functions used in this model assuming a distance-based fare system.

$$\begin{aligned} C_{iv}(t) &= (\beta_{iv} * t_{iv}(t))(1 + g(p_{oc}(t))) + \beta_{vd} * l_{iv} + f_{iv} * l_{iv} \\ C_{bo}(t) &= \beta_{wt} * t_{wt}(t)) + c_{tr} \\ C_{al}(t) &= 0 \\ C_{wk}(t) &= \beta_{wk} * t_{wk} + \beta_{wd} * l_{wk} \\ C_{path}(t) &= \sum C_{iv}(t') + \sum C_{bo}(t') + \sum C_{al}(t') + \sum C_{tr}(t') - c_{tr} \end{aligned} \tag{19.1}$$

C_{path}: Total cost of the path.
C_{iv}: Cost of one in-vehicle link.
C_{bo}: Cost of one boarding link.
C_{al}: Cost of one alighting link.
C_{wk}: Cost of one walking link.
β_{iv}: Personalized cost per unit of time traveling in a vehicle.
β_{vd}: Personalized cost per unit of distance traveling in a vehicle.
β_{wt}: Personalized cost per unit of time waiting in a stop.
β_{wk}: Personalized cost per unit of time walking.
β_{wd}: Personalized cost per unit of distance walking.
c_{tr}: Personalized cost for making a transfer.
f_{iv}: Vehicle dependent fare rate by distance traveled.
$t_{iv}(t)$: In-vehicle travel time (from Stop-stop travel times structure).
$t_{wt}(t)$: Waiting time (from Stop-route waiting times structure).
t_{wk}: Walking time.
l_{iv}: In-vehicle distance.
l_{wk}: Walking distance.
$p_{oc}(t)$: Occupancy level in the in-vehicle link (from Route-stop occupancy structure).
$g(p)$: Simplified function of how occupancy level increases the cost (Equation (19.2)).

$$g(p) = \begin{cases} 0 & if \quad p \leq p_{sit} \\ r_{sta} * p + b_{sta} & if \quad p_{sit} < p < 1 \\ b_{full} & if \quad p = 1 \end{cases} \tag{19.2}$$

p_{sit}: Occupancy level when no more seats are available.
r_{sta}, b_{sta}: Parameters of percentage increase in discomfort from standing in the vehicle.
b_{full}: Maximum percentage increase when the vehicle is full.

Shortest Path Algorithm To find a public transport route between an origin and a destination, for a given time of day, the applied method was the same as currently implemented in MATSim; first, the algorithm looked for the stop-nodes within walking distance from both origin and destination. An initial cost was associated with each of these stop-nodes, according to access and egress walking times. Then, starting from all the origin-stop-nodes with a given access cost, a multi-node time dependent Dijkstra algorithm found the shortest path, to the destination-stop-nodes with related egress costs. Thus, the path determined the best O-D (Origin-Destination) combination as well. The algorithm was time-dependent because it recognized that while it proceeded through the path, time advanced; thus, different costs are obtained from the links while time advanced. The total disutility of this path was compared with the cost of a full walking trip. If the cost is less, the path is converted to a sequence of stages: in-vehicle stages for each in-vehicle link in the path and walking stages for each walking link. Boarding and alighting links were ignored for this conversion.

Structures to Save Travel Times, Waiting Times and Vehicle Occupancy As mentioned earlier, the mobsim of MATSim generated atomic units of information called events, which described changes for each person, e.g., boarding or alighting: each vehicle, e.g., entering and leaving a link during the simulation. The goal was to save information on public transport experience in one simulation and find better public transport routes for agents in the next iteration. This feedback mechanism was already implemented in MATSim for private transport; the car router used each saved link's time-dependent travel times from a previous iteration to calculate better routes in the road network, by changing the costs of the links. To allow the EBPTR to learn from the previous iteration, information about (a) stop-stop travel times, (b) stop-route waiting times and (c) route-stop-stop vehicle occupancy, was required.

- Stop-stop travel times: To account for public transport vehicle delays, travel time between consecutive stops had be saved. Two stops are consecutive if they were consecutive for at least one public transport route. A first option was using the previously discussed travel times structure that saved time-dependent travel times for each road network link. Because a vehicle had to follow known road links between two consecutive stops, these travel times could be summed. One problem: this structure accounted for all the vehicles in the network, but travel times of cars and buses were very different, particularly in links with public transport stops. Thus, a special structure was implemented to save these stop-stop travel times. The structure averaged all the public transport vehicle times from one stop to the next during a certain time bin. More specifically, each value comprised the time from when the vehicle arrived at a certain stop until it arrived at the next stop, denoted in the simulation by consecutive `VehicleArrivesAtFacility` events. This meant that the first stop waiting time and all queue times (if the vehicle had to queue before the bay or platform was available) were included. In other words, when an agent routed the first in-vehicle link of each trip, the full dwell time would be included. Hence, this agent assumed it was the first passenger entering the vehicle. For all the other in-vehicle links the in-vehicle waiting was included. These stop-stop times were the main component of the in-vehicle link disutilities.
- Stop-route waiting times: Waiting times are a fundamental aspect of public transport route choice and can be long due to vehicle delays (i.e., due to the stop location), or full public transport vehicles of one or several consecutive services (i.e., due to the route demand and stop position within the route). For that reason, waiting times were saved for each stop-route relation. Similarly, the structure averaged all agent waiting times in a certain stop, for a certain route, during a certain time bin in the day. More specifically, each value comprised the time from when the agent arrived at the public transport stop until it entered the vehicle, denoted in the simulation by consecutive `AgentArrivesToFacility` and `PersonEnterVehicle` events. These waiting times were the principal component of boarding link disutilities. If no observations were found for a certain stop-route-time, the model returned half the corresponding headway, specified by the transit schedule.
- Route-stop occupancy: By accounting for occupancy level, one can model routing decisions where people take longer/slower routes to feel more comfortable in emptier vehicles, i.e., valuing a higher chance to travel while seated. Occupancy depends on specific route demand and the stop position within the route. Here, occupancy was assumed to be constant between two consecutive stops. When a vehicle departed from a certain stop (denoted in the simulation as `VehicleDepartsFromFacility` event) this structure averaged the occupancy level with the other vehicles on the same route departing from the same stop during the same time bin. As there were only a few vehicles recorded for each time bin, it was unlikely to find observations for a specific bin. In this case, the structure returned the value of the next time bin, where at least one observation was found for the corresponding stop and route.

19.3 Functional Results

Relaxation Process The number of iterations needed by MATSim's co-evolutionary algorithm to reach a stable state was a critical variable; efforts were made to reduce it (Meister et al., 2006; Fourie et al., 2013).

The EBPTR effectively reduced the iterations public transport users needed to reach equilibrium. Using a 25 % sample of the Singapore scenario, Figure 19.2 shows average score plan evolution for 355 207 agents over 100 iterations. These 100 iterations were executed four times to use both routers for two different replanning strategies. Agents saved five plans in memory. At iteration 0, both EBPTR and SBPTR started with routes described in the schedule; however, the EBPTR returned routes that performed better in this first simulation. This occurred because, for each pair of consecutive stops, the EBPTR used the average of all scheduled route times that contained this pair as the first estimate. On the other hand, the SBPTR used the specific scheduled time of the corresponding route. Results indicated the average stop-stop time seemed to be a more reliable estimate for this first iteration.

For the rest of the iterations, the Figure 19.2 shows how the scores evolved. The first replanning strategy stipulated that 30 % of the agents were re-routed at each iteration. This evolution is shown in the first graph of the figure. Using SBPTR, agents received the same route over and over again as the start time, origin and destination did not change between iterations. Small variations in scores occurred because of the stochastic simulation nature explained above. Although scores started in the same range, using EBPTR allowed better-performing routes to be found within a very small number of iterations.

For a more realistic comparison, a second replanning strategy was tested, where just 20 % of the agents were re-routed and the activity start times were modified randomly within a half an hour for 10 % of the agents. The second graph of the figure shows how both routers managed to improve agents' plan scores. But with the EBPTR, number of iterations needed to achieve the average executed score, achieved after 100 iterations for the SBPTR (120), was only 5. The target marginal score, as a measure of change in score over iterations, was taken arbitrarily as 0.1 utilities per iteration, or the rate produced after 200 iterations with the SBPTR. In contrast, this target rate was achieved after 77 iterations with the EBPTR, a 2.6 improvement factor .

Modeling Advantages Because of the links disutility function in the proposed network account for aspects like waiting times or occupancy levels and because MATSim allows for modeling heterogeneity among agents, the router could be a very powerful tool to model observed emergent behavior in public transport route choice. In Singapore, like many other crowded cities in the world, some commuters decide to travel backwards for a few stops and then transfer to a train in the opposite direction to find a seat or space in a public transport vehicle Chakirov and Erath (2011). With the SBPTR this kind of least cost path could not be found, but with the newer proposal, this was possible. Although proportions did not match actual observations as the Singapore scenario lacked appropriate and calibrated utility parameters for traveling and waiting time under crowded conditions, Figure 19.3 shows totals of people traveling backwards from different stops in the island after 100 iterations (see Figure 19.2 (a)).

19.3.1 Comparing Quality Attributes With the Current Implementation

Computation Time The tests described next were executed using 12 computational nodes, accessing 70 GB of shared memory, using the Singapore scenario described in Chapter 57. Before the first iteration, if plans were not routed, MATSim prepared every agent with an initial route. As mentioned before, the stop-stop travel times and stop-route waiting times were initially taken from the schedule. Because of its simpler network structure the EBPTR took 01:17:35 to initially

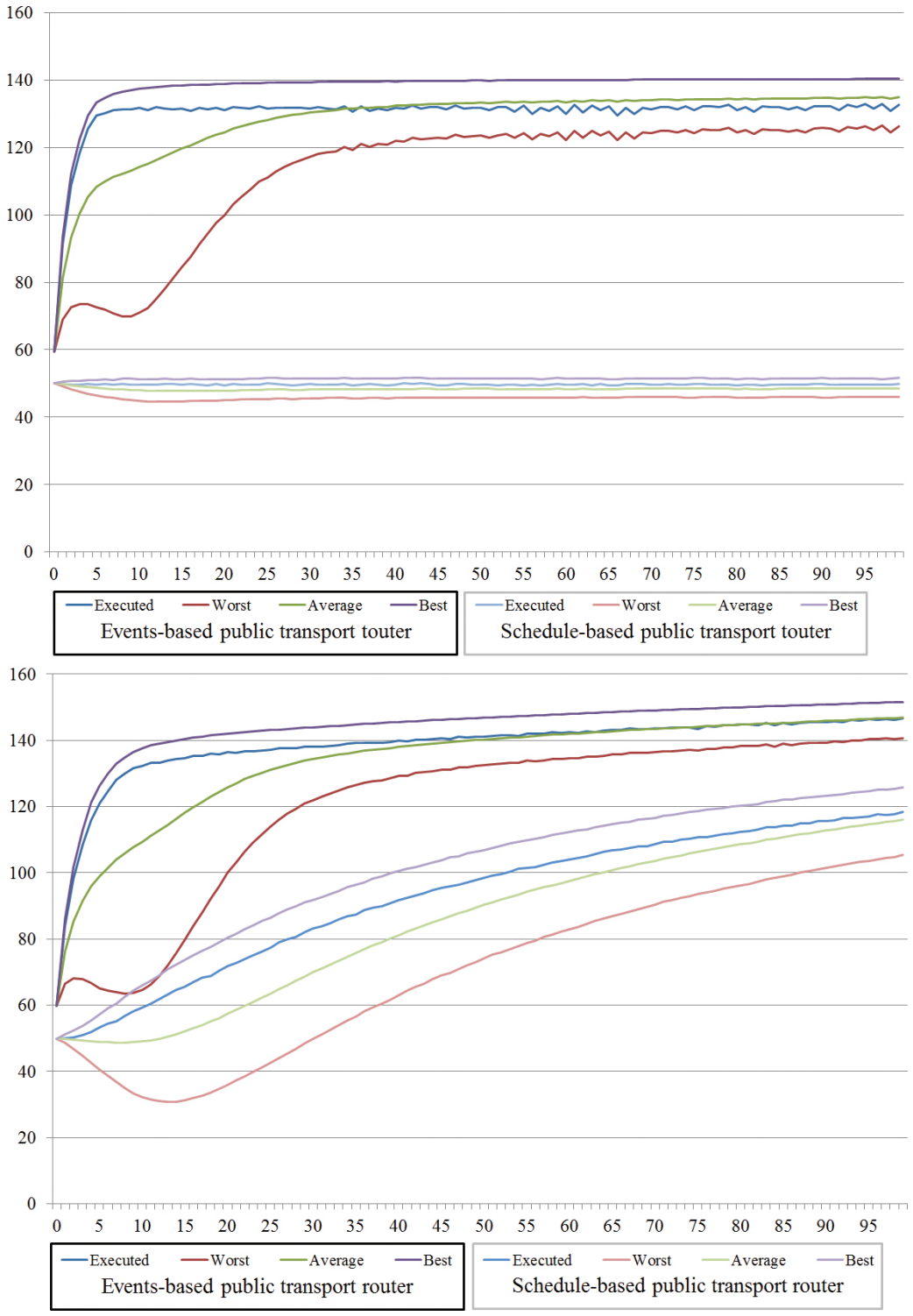

Figure 19.2: Comparison of score evolution: a) 30 % re-route, b) 20 % re-route and 10 % time allocation.

Figure 19.3: Number of agents traveling backwards at each MRT (Mass Rapid Transit, Singapore) station of the Singaporean rail system.

route the 355 207 users, compared with 01:28:55 needed by the SBPTR, producing a performance gain of about 12.7 % for this scenario. When running MATSim iterations with the EBPTR, computation times principally changed in two processes: mobility simulation (mobsim) and replanning. Figure 19.4 shows computation times measured for the first 20 iterations of the process. Although the EBPTR needed more time in mobsim, it continued to require considerably less time for rerouting during the replanning, due to a simpler network topology. The longer mobsim time was due to information saving in the new structures during the simulation. However, on average, the EBPTR outperformed SBPTR, per iteration, by about 3 minutes or 11 %. As mentioned above, 2.6 times more iterations were needed for the SBPTR to achieve a specific point in the relaxation process. For 77 iterations with the EBPTR, computation amounts 35:25:43, and for 200 iterations with the SBPTR, computation amounts 99:10:51; a 2.8 improvement factor in our experimental setting.

Memory Consumption The EBPTR needed more memory than the SBPTR, because the EBPTR managed more information. The necessary extra memory was allocated to the three structures described before. Given the Singapore scenario conditions described, the extra memory was calculated as follows. One numeric value needed eight Bytes, and with a time bin of 15 minutes, 120 bins were needed for 30 hours. The Stop-stop travel times structure saved two values (average and number of observations) for each time bin and each pair of consecutive stops. The number of pairs for the Singaporean public transport system was 6 602. Thus, this structure needed approximately 12.7 MB. Similarly, the stop-route waiting time structure saved two values (average and number of observations) for each time bin and each pair of stop/route combinations. The number of stop/route relations for the Singaporean public transport system was 27 156. Thus, this structure needed approximately 52.1 MB. Finally, the vehicle occupancy structure saved the average and number of vehicle occupancy observations for 26 353 route-stop relations for each of the 120 time bins, requiring approximately 50.7 MB. In total, less than 120 MB were needed for the three structures.

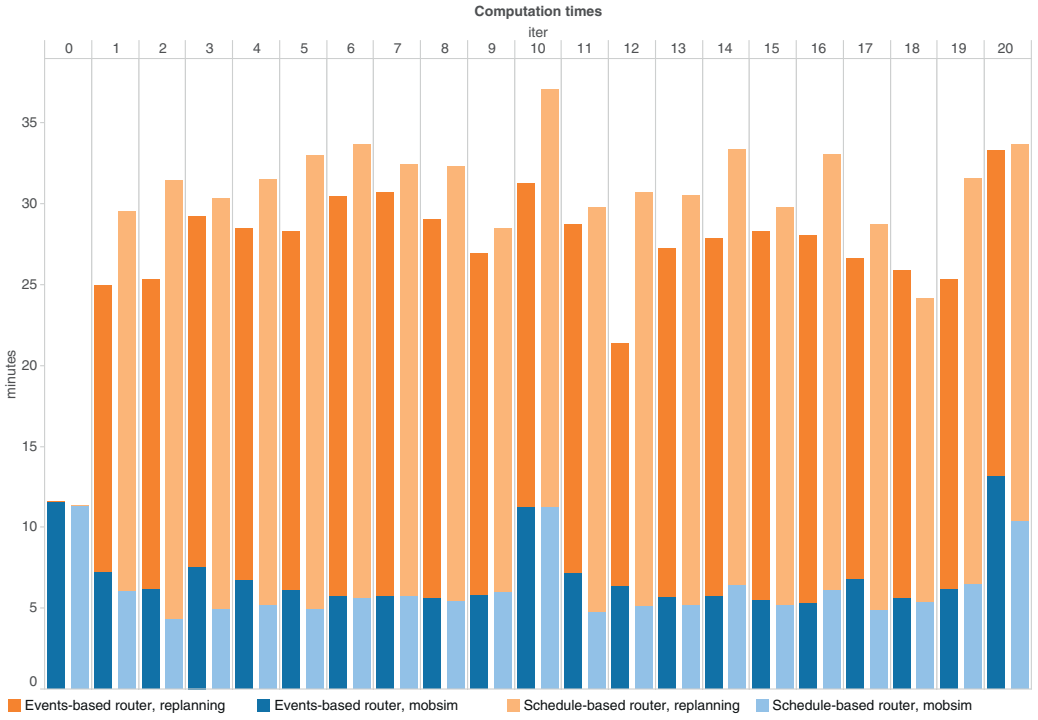

Figure 19.4: Comparison of computation times for 20 iterations.

On the other hand, the size of the network where public transport routes were calculated was smaller for the EBPTR. Although, in the case of Singapore, it created 31 939 nodes compared with 27 156 of the SBPTR (4 783 new stop-nodes), the number of links is dramatically smaller. The SBPTR created 424 070 walking links and 26 353 travel links (450 423 in total). The EBPTR created the same 26 353 travel links, plus 27 156 boarding links, plus 27 156 alighting links and just 4 390 walking links (85 055 links in total); less than a fifth in total. As a node needed 48 bytes and a link 128 bytes, the SBPTR needed roughly 46.8 MB more memory for links and just 229.6 KB less for nodes. The EBPTR saved 46.5 MB for the network, concluding that in total the SBPTR needed 70 MB less memory. This quantity was negligible compared with the total memory needed for the whole simulation (more than 40 GB).

19.4 Conclusion and Future Work

In this work, a new public transport router for MATSim was designed, implemented and tested. It produced more diverse routes in large scale scenarios, taking into account many complexities of urban public transport systems. On the supply side, the system simulated congestion, public transport vehicles occupancy levels, queues in public transport stops, bay sizes, and bus or train bunching. On the demand side, in addition to commonly used factors like in-vehicle time, number of transfers and walking time, the new router took disutility of additional waiting time due to congestion or overcrowded vehicles, comfort level inside public transport vehicles and preference heterogeneity among agents for all mentioned factors into account.

The utility of the new approach was tested in a large scale Singapore scenario. Using a simplified public transport only simulation, 100 iterations of a 25 % scenario (355 207 agents) with 30 % of the agents re-routing each iteration took just 45 hours approximately, or about 27 minutes per

iteration, using 12 cores and 70 GB of memory. The computation decreased by 11 %, compared to the standard MATSim. If just 20 % of the plans were re-routed, using 35 cores accessing 85 GB of memory, the time per iteration would be reduced to less than 13 minutes, achieving 100 iterations in less than one day. But, most importantly, for computation time gains, we showed that the proposed events-based router was able to reach a steady state in a much smaller number of iterations.

If the proposed router works better than the original one, should it be changed? The current scheduled-based router of MATSim would still be relevant if the topology of its network were changed for the proposed one. It should also be applied to scenarios where the public transport system operates very reliably and punctually, with few cases of overcrowding. In that case, routing calculations would be as fast as the events-based router (with the new network structure), and the mobility simulation would be faster, as no information (in-vehicle time, wait time and occupancy) would be needed. In other words, it could be applied to city models where public transport users can reliably plan their trips using only a timetable.

Scrutinizing the resulting network loading, the biggest potential advantage of the proposed events-based router was its capacity to generate emergent behavior in congested public transport systems, in line with actual observations. Future research should aim at estimating the various route choice behavior parameters corresponding to the functionalities of the proposed system and calibrating the simulation. Although the values used came from a stated preference survey commissioned by the Land Transport Authority for the case of Singapore, advanced studies could, for example, be tailored to quantify preference heterogeneity. Furthermore, results from work in progress about the value of a seat in Singapore and discomfort disutility can improve prediction confidence. Finally, information from the Singapore smart card data could be used for revealed preference estimation of further behavioral parameters, like quality of a transfer described, e.g., by the number of escalators, to further refine the system.