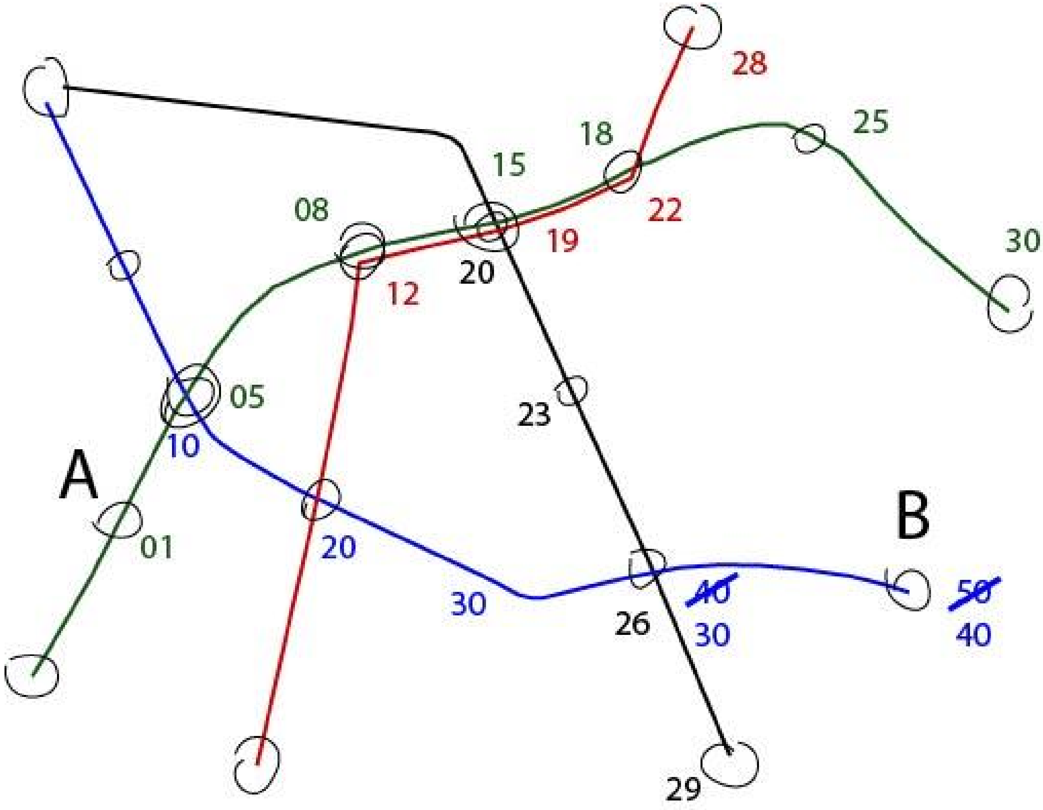


Adding Realism and Efficiency to Public Transportation in MATSim

Marcel Rieser, SBB
Denis Métrailler, SBB
Johannes Lieberherr, ttools gmbh

Conference paper STRC 2018

STRC 18th Swiss Transport Research Conference
Monte Verità / Ascona, May 16-18, 2018

Adding Realism and Efficiency to Public Transportation in MATSim

Marcel Rieser
marcel.rieser@sbb.ch
(until 24.4.2018, afterwards:
rieser@simunto.com)

Denis Métrailler
denis.metrailler@sbb.ch

Johannes Lieberherr
Johannes.lieberherr@ttools.ch

SBB
Passenger Division, Service Planning
Wylerstrasse 123,
3000 Bern, Switzerland

ttools gmbh
Simplonweg 21
3008 Bern, Switzerland

May 2018

Abstract

The current simulation of public transportation in MATSim has several shortcomings regarding the simulation of railway transportation. This is mostly due to the fact that for the simulation of public transportation, the same data structures are used as for the simulation of private car traffic, while the desired level of detail of the modelling varies strongly. For example, while the road network is modelled in detail with intersections and sometimes even lanes, railway transportation is usually modelled on a more conceptual level with stops and connections, and not with rails and switches.

The paper describes current shortcomings for modelling railway transportation in MATSim and takes a look at current work-arounds. Then, a new module is presented allowing the deterministic simulation of public transportation, solving many of the mentioned shortcomings.

A second part will present a related work: A new alternative public transport router has been implemented for MATSim, which not only provides a more realistic public transport route choice behaviour by MATSim's agents, but also increases efficiency in terms of computation time and memory usage.

In a third part, results will be presented highlighting how the aforementioned changes (positively) affect the simulation of public transportation in MATSim.

Keywords

public transport, modelling, routing, railway transportation, agent-based simulation, MATSim

1. Introduction

Many transport modelling tools first focused on road traffic only, adding alternative modes like public transport (pt) or bike at a later stage. The multi-agent transport simulation MATSim (Horni et al., 2016; MATSim, 2018) is no exception here. Early functionality supported the simulation of car-traffic only (e.g. Raney et al., 2003; Meister et al., 2009), while support for multimodal models was added a few years later (Rieser et al., 2009), followed by full support for the detailed simulation of public transport shortly thereafter (Rieser, 2010).

When the simulation of public transport was added to MATSim, the implementation was heavily influenced by the existing car-only simulation. For example, all movements of public transport vehicles were simulated on network links which were originally designed to simulate private car traffic. In the long term, these implementation decisions lead to several shortcomings when modelling large, complex public transport systems, especially such with a major share of railway transportation.

The Swiss Federal Railways (SBB) is currently building a MATSim model covering all of Switzerland. Although the model is multimodal, special attention is put on modeling public transportation. Due to the nature of SBB's business, schedules and passenger interaction need to be accurately modelled. During this work, several shortcomings in MATSim's public transport modelling were identified. SBB decided to invest in implementing alternative software modules for MATSim to solve these shortcomings. This paper reports on the found problems, the alternatives implemented and the effects of the changes for building large multimodal MATSim models.

The paper describes present shortcomings and the currently used work-arounds and their drawbacks in Chapter 2, before presenting a new public transport simulation for MATSim in Chapter 3 that overcomes the currently known shortcomings and drawbacks of work-arounds. Chapter 4 presents a new public transport router for MATSim, which—besides being very fast—allows agents to select more realistic routes than with the currently available pt router in MATSim. Both the improved simulation and routing of public transport are then applied to an agent-based model of all of Switzerland currently being built at SBB. The effects of this application are described in Chapter 5, and an outlook on future potential improvements is given in Chapter 6. Final conclusions are drawn in Chapter 7.

2. Current Shortcomings and Work-Arounds

2.1 Network-Based Simulation

In the current default public transport (pt) simulation of MATSim, all pt vehicles are simulated on links of the network. The mobility simulation in MATSim, QSim, models such links traditionally with first-in-first-out (FIFO) queues where vehicles can proceed with the allowed free speed. Vehicles may get delayed only if they, or a vehicle in front of them on the same link, cannot cross a node due to the downstream link being full. This behavior works well in the case of road traffic where cars typically drive close to the speed limit if the situation allows for. But for railway transportation, the assumption of vehicles generally driving with the allowed speed limit does not hold. Rather, the actual speed is dependent on the type of train, if it is on time or not, and other factors.

The level of detail on how the respective networks are modelled differs as well. Road networks are typically fairly detailed with links and nodes representing roads and intersections. Rail networks, on the other hand, are often modelled with a higher level of abstraction for transport planning purposes, often just connecting stops or service points represented by nodes with links, and not with rails and switches. There exist more detailed models, but those are usually used to simulate and verify the practicability of schedules given the infrastructure (e.g. OpenTrack, 2018), and not for general transport planning.

When the default, queue-based network is used to simulate public transport vehicles, the following two major problems can be observed:

- PT vehicles do often not arrive on time, but too early or too late depending on the specified speeds on the links.
- PT vehicles might block each other on links, leading to additional delays, where in reality they could pass each other.

These problems lead to the situation that case studies modelling changes in the public transport schedule are difficult to assess, as the experienced schedule by the agents may vary broadly from the planned schedule. The fact that the experienced schedule deviates from the planned schedule mostly due to modelling artefacts, and not due to real-world effects like late departure due to a high number of passengers boarding or alighting a train, makes the results of such case studies even more questionable.

A common work-around is to set a rather high free speed on links used exclusively by public transport vehicles, combined with the setting in the transit schedule that vehicles must await the scheduled departure time at each stop. This results in pt vehicles not accumulating gains by arriving too early at a stop, and thus trying to keep the effects on agents due to early arrival to a minimum. But for long distance services, connecting cities with non-stop connections, the early arrival could still be substantial.

An alternative approach is thus to assign each transit route its own sequence of links in the network. The free speed on these links can then be calculated to result in the exact planned travel time between two stops. This also solves the problem of overtaking, since each transit route is served on its own links, and vehicles of the same route should by design not overtake each other. But in large, complex schedules this could result in dozens of parallel links between nodes. This introduces quite some overhead due to the high number of additional links. In addition, the analysis of such scenarios becomes more complex as well, as often the values of multiple links need to be aggregated. Also, it is not possible for transit vehicles to make up for lost time, as the speed was calculated according to the schedule.

As a final work-around, a feature of MATSim could be used that allows to calculate the speed of a vehicle on a link at the time the vehicle enters the links. In theory, this would allow to calculate the optimal speed, even taking current delays into account. In practice, this works well if stops served after each other in a transit route are connected with a single link. As soon as two stops are connected with a series of links, the calculation becomes very complex, requiring downstream links to be considered as well in the calculation to determine the remaining distance and take possible speed limits into account. In addition, this approach does not solve the problem of vehicles needing to overtake each other according to the schedule.

None of the presented approaches to work around the shortcomings in MATSim's current public transport simulation provides a satisfactory solution. This is mostly due to the fact that the simulation of public transport, combined with the more abstract network modelling approach for railway transportation, does not work well with the queue-based network used for the simulation.

2.2 Public Transport Router

The public transport router currently available in MATSim is a slightly modified version of Dijkstra's shortest path algorithm (Dijkstra, 1959), an algorithm usually applied for finding paths in graphs or road networks. To be used for public transport routing, a graph or network has to be created from the transit schedule, typically containing vertices representing stops and edges representing connections between stops (either by train/bus/… or short-distance walking for transfers). For large transit schedules, e.g. for an application of the full schedule of Switzerland, the generated graph becomes huge, requiring large amounts of memory and resulting in slow query times. This in turn leads to long times for MATSim's replanning phase as soon as public transport is used. A limited work-around is to increase the parallelization of the replanning in MATSim by assigning more threads to it. But given the finite number of computational cores todays CPUs have, its effect is limited and is not a real solution for large-scale applications.

In addition to the performance problems, the public transport routing has some additional shortcomings for complex scenarios. Namely, the default pt router assumes the same minimal transfer times for all possible transfers, neglecting the fact that minimally required transfer times may differ between rural stops (e.g. connecting a regional train line with a local bus) and urban hubs (connecting a large number of different train lines with a large number of trams and buses). Also, passengers have different preferences and price sensitivities, possibly resulting in choosing different variants for the same trips. But as the default pt router applies the same generalized cost function with the same parameters for each routing request, all passengers would choose the same connection for comparable trips, resulting in unrealistic route choices in some cases.

3. Deterministic Public Transport Simulation

The so-called deterministic public transport simulation (detPTSim) aims to provide a public transport simulation for MATSim where the transit vehicles serve the stops strictly according to schedule. Such a simulation is mostly useful to simulate trains or other modes that operate on dedicated tracks, and which are not likely to experience delays due to missing interaction with other transport systems.

Instead of using a network where the pt vehicles are moved along the links and handle stops, detPTSim uses one central queue containing all pt vehicles to simulate public transportation. Each pt vehicle has a series of events, describing departures and arrivals at stop facilities as well as time for passenger interaction (boarding or alighting), based on the public transport schedule. Sorting the queue by the time of the next event of each vehicle makes it essentially a priority queue. In each simulated time step, all vehicles whose next event should have happened by the current time step are handled. This handling includes the creation of the appropriate MATSim events like VehicleArrivesAtFacilityEvent, PersonLeavesVehicleEvent, PersonEntersVehicleEvent and VehicleDepartsAtFacilityEvent as well as the interaction with passengers (e.g. sending alighting agents to the ActivityEngine to start their next activity).

The implementation uses a PriorityQueue to manage the pt vehicles, where the vehicles are removed from the head of the queue for handling, and afterwards re-inserted with their next event defining their insertion place in the queue. In the case of passenger interaction, vehicles might re-insert themselves every second as long as passengers are boarding or alighting, or while the vehicles await their departure time. This allows to let passengers alight or board not all in the same second when a vehicle arrives at a stop, but in a steady flow according to the vehicles' door configuration.

Depending on the configuration, it is possible that pt vehicles cannot depart in time at a stop due to ongoing passenger interaction. In such a case, the vehicle will depart with a delay as soon as all passengers alighting or boarding at that stop have done so. In such a case, the public transport vehicle will still arrive on time at the next stop, given it does not have to travel backwards in time. If the next stop's arrival time has already passed, the vehicle will essentially immediately teleport to the next stop and arrive there in the same second as it departed the first stop. This allows to keep the schedule as stable as possible for the agents using public transportation, while it still allows to perform analyses regarding the locations and time of day where delays start to build.

As the public transport vehicles are not simulated on a network but directly teleported from stop to stop, no link-events (e.g. LinkEnterEvent, LinkLeaveEvent) are created for the pt vehicles. For reasons of visualization and analysis—especially to maintain backwards compatibility with the default pt simulation in MATSim—, such events could still be of interest. The detPTSim thus allows to optionally create such events. It does so by linearly interpolating the position of a pt vehicle along the network route between two stops at the time of departure at a stop. By precalculating at what time the transit vehicle would leave or enter a link in order to reach its next stop at the specified time, matching simulation events can be generated at the right time and fed into MATSim's event handling routines.

Altogether, the deterministic public transport simulation allows to simulate public transport in MATSim in a simpler fashion, ensuring timely departures and arrivals according to the schedule without the need to tweak the network, and being backwards-compatible regarding the generated MATSim events. While it does not directly support the interaction with private car traffic, the deterministic public transport simulation can be used alongside MATSim's default pt simulation which offers this functionality, to selectively simulate some pt vehicles strictly according the schedule with the detPTSim, while other pt vehicles can interact with road traffic and thus experience delays due to traffic jams.

4. Public Transport Routing: SwissRailRaptor

4.1 Modern Algorithms for Public Transport Routing

As highlighted in Section 2, a major hindrance in applying MATSim's public transport simulation for large models is the bad performance, along with missing behavioral details. Since the implementation of MATSim's current default public transport router (Rieser, 2010), research on public transport routing has produced a number of newer and faster algorithms.

One of the fastest algorithms is the Transfer Patterns algorithm (Bast et al., 2010) used by Google for its transit maps. It achieves its high speed by pre-calculating all possible connections between two stops, inserting current departure times at request time to return the actual connections. While this approach yields very quick answer times, it requires an enormous initialization time (several hours up to days) to pre-calculate all possible connections for a nation-wide schedule, and also requires large amounts of memory to cache all this information, making it unfitting for MATSim.

Witt (2015) proposes a graph-based approach, using a special time-expanded graph of the schedule with nodes representing vehicle trips (part of transit routes that can be travelled between two stops without changing the vehicle), and links representing transfers. An optimized version of it (Witt, 2016) achieves even better performance, but again requires substantial and complex pre-processing and initialization time (up to several hours), making it unsuitable for MATSim as well.

The connection scan algorithm (CSA) proposed by Dibbelt et al. (2013) claims to require only minor pre-processing and tries to make use of memory- and CPU-cache-locality to achieve good performance. It sorts all departures at all stops into one huge array that can then linearly be scanned. One problem with this approach is its missing spatial awareness: When searching for a public transport connection within a larger city (a typical application in MATSim simulations), it also has to scan all departures in other regions and cities as well. And even when searching a connection between two cities (e.g. between Berne and Zurich), it still has to scan all departures of all buses, trams and trains in Geneva, Lausanne, Basel, Olten, and so on, giving away large parts of the CPU-cache locality gains. More advanced variants of the algorithm exists (e.g. Strasser and Wagner, 2014), using concepts similar to domain decomposition to introduce spatial awareness, but making the algorithm more complex. Due to the strict sorting by departure time, it also becomes more difficult to not only search for the earliest-arrival connection, but for example for the connection with the least transfers or with the least costs, where travel time and transfers add to the cost (as it is currently the case with MATSim's default pt router).

The Raptor algorithm (Delling et al., 2012) includes spatial awareness and suggests CPU-cache-friendly data structures for optimal performance, requires only limited pre-processing while still offering very good routing performance. The algorithm works by first looking at all transit routes directly reachable from the start location, marking all transit stops reachable by those routes. In the next round, it does the same for all transit routes reachable by transfers at the stops from the previous round, again marking all stops newly reachable. Whenever – within a round – a trip arrives at a stop it is checked for every corresponding journey if there is already a journey with smaller journey time to the actual stop (local pruning) or even to the destination stop (target pruning). If one or the other is true, the journey is discarded. Otherwise, the trip is saved and processed further in the next round. This process (which is illustrated on the cover picture) is repeated until the destination stop is reached and no earlier arrival can be found. The algorithm supports variants for multi-criterial routing request (e.g. finding the fastest route or the one with fewest transfers) as well as for range queries (also called profile queries, finding the optimal connection given a departure time window) and is easy to adapt to use a generalized cost function to determine the optimal connection.

VISUM (PTV Group, 2018) uses a branch-and-bound algorithm described in Friedrich et al. (2001). The algorithm creates a search tree, starting with the departure stop as the root, adding branches for each stop directly reachable from this root. Each level of the tree thus corresponds to a transfer. A breadth-first search strategy is applied along with heuristics whether a leaf node (representing a reached stop) should be expanded and another level should be added (i.e. a transfer should be performed). The algorithm may return multiple paths in that search tree, corresponding to different journeys to travel from the origin to the destination stop in order to split the demand to multiple alternatives.

Traditionally, many routing algorithms build routing graphs that closely resemble the spatial aspects of the schedule, e.g. with nodes representing stop locations and links representing pt services between stops or transfers. With such a structure, they usually are limited to some variant of graph-based shortest-path calculations, which typically expand along isochrones in the graph. The Raptor algorithm, Witt's trip-based graph and VISUM's branch-and-bound algorithm are different from that, as these three algorithms all apply some kind of breadth-first search where they first check all stops directly reachable, then all stops reachable with 1 transfer, then all stops with 2 transfers, and so on. Likely, this gives them some computational advantage especially for long-distance trips, as they can more quickly advance over long distances if there are direct services available. Also, they do not depend on Priority Queues often used for graph-based routing, which impose a performance limitation by their own.

4.2 Implementation

4.2.1 Base Implementation and Optimizations

Based on the analysis of several public transport routing algorithms, the Raptor algorithm was assessed to offer the most benefits when applied in MATSim's context. Thus, the algorithm was implemented in Java, using MATSim's data structures to initialize it and implementing MATSim's API so it can be easily integrated, considering the following points/extensions:

- The Raptor routes (sets of non-overtaking trips) are taken directly (1:1) from the MATSim routes. As we will see in 5.1.2, it is crucial for the performance that the MATSim departures (trips) are summarized compactly within the MATSim routes.

- All stops reachable by foot from the start or destination coordinates are initialized by the corresponding walking time. The algorithm is extended naturally such that it searches from the initialized start stops to the initialized destination stops.
- Instead of minimizing the arrival time, the score is minimized.

The implementation—named SwissRailRaptor—can act as a drop-in replacement for MATSim's default public transport router. Great care was given to the implementation to be computationally highly efficient, as bad performance is one of the major problems of the default pt router.

An important aspect influencing the routing performance is the definition of transfers. The more transfers that are possible during the routing, the longer the runtime of the algorithm is, as the breadth of the search space increases with each possible transfer. The original paper (Delling et al., 2012) does not touch this point—like many other papers about public transport routing—but just assumes that possible transfers between stops are well defined and just need to be included in the algorithm somehow. In MATSim, transfers are by default not enumerated, but calculated based on the spatial proximity of stops. Depending on the stops' coordinates and a configurable transfer radius, close-by stops are identified and connected as transfers.

In an effort to further improve the performance of the routing algorithm, the list of possible transfers is heavily pruned using several heuristics. Inspiration for this pruning of transfers came from Witt (2015), who applied similar technics to reduce the transfers in a time-expanded graph. A few of the heuristics applied in SwissRailRaptor are inspired by the aforementioned paper, while many others were specifically developed to be of use given the application along the Raptor algorithm.

In a first step, transfers were defined to be between stops of specific transit lines and routes, and not only between transit stop facilities. While this may sound counterintuitive at first (many stops are served by more than one line or route, so having transfers between routes' stops instead of stop facilities adds a large number of transfers), it helps to keep the set of reachable transit lines smaller which affects the performance of the Raptor algorithm the most.

Most notably, transfers are not included in SwissRailRaptor in the following cases:

- The transfer leads to the last stop of a route. As that line will not continue anywhere else, it does not make sense to transfer to it.
- The transfer starts at the first stop of a route. We must have arrived here by a transfer, so do not immediately transfer further.
- The transfer leads to a route that serves the same stops in the same sequence as the current route. Assuming that vehicles serving the same stops in the same sequence do

not overtake each other, one can simply remain in the current vehicle instead of transferring.

- One could have transferred at the previous stop to the same route going in the opposite direction. It usually does not make sense to travel to a stop, just to travel back again to a previous stop with another service.
- There is no possible connecting service on the destination route. This can happen with special routes running only during the morning or evening hours. There is no use in transferring from a route running only in the evening to a route only running during the morning hours.
- The transfer leads to a stop served by the same route, either before or after the originating stop. This can happen if a large transfer radius is configured and two stops of a route are nearby each other. A transfer usually does not make sense as one could just leave the vehicle earlier or later, respectively.

Applying these heuristics helped to reduce the number of possible transfers substantially, resulting in an improved performance of the SwissRailRaptor.

4.2.2 Extensions for More Realism

Once the basic implementation of SwissRailRaptor was available, a number of extensions and additional features were added with the goal to improve the realism of the calculated routes.

Stop-specific minimal transfer times. Support was added to MATSim and SwissRailRaptor to define custom minimal transfer times between stop facilities. Originally, just a single value was applied for all transfers, no matter if it was a transfer in a rural area or at a major station. This resulted in agents sometimes finding very fast connections that would (often) not have been possible in reality at major stations. In order to support different transfer times between different stop-relations, MATSim's transit schedule data structure was extended to incorporate a list of minimal transfer times between stop facilities. SwissRailRaptor makes use of this list to adapt the pre-calculated, distance-based transfer times. This also enabled SwissRailRaptor to provide specific transfers connecting stops that are further apart than the default maximum transfer distance. As these times are taken into account during the initialization of the router, this feature has no impact on the performance of the router.

Person-specific routing parameters. Not every person might value travel time or transfers the same. Some might prefer fewer transfer even if they have to travel longer, while others might prefer to change more often in order to arrive a few minutes earlier. SwissRailRaptor introduces an API similar to MATSim's scoring function, which allows to specify specific parameters for each person. By default, the same parameters are applied for each routing request. But by providing a custom implementation of the interface "RaptorParametersForPerson", it is possible to define custom routing parameters for each routing request, based on the agent requesting a route. The default implementation does not imply any performance penalty, while the performance impact of a custom parameter provider depends on its actual implementation, but should be negligible in typical use cases.

Route-specific cost parameters. Not only have different persons different tastes and require thus different routing parameters, the valuation of different transport systems may even be different for a single person. Some people might opt to pay a higher price in order to travel faster and arrive earlier, while other passengers might value cost more and prefer slower but cheaper means of travelling. In order to choose between different public transport alternatives, e.g. high-speed trains vs. regular trains vs. long-distance coaches, the cost function for routing must be able to differentiate the different transit routes somehow. To stay consistent, not only the router should be able to distinguish different means of public transportation, but also MATSim's scoring. For this reason, SwissRailRaptor provides a so called "passenger mode mapping". Instead of just creating legs with mode "pt", it can map the modes assigned to transit routes in the schedule to arbitrary other values. As these other values are used as modes for the passengers' legs, corresponding scoring parameters need to be defined which SwissRailRaptor can use as well for its routing purpose. Although the mode mappings are the same for all transit routes and all agents, the person specific parameter provider mentioned before can provide different values for the mapped passenger modes, enabling the routing of complex scenarios efficiently.

Departure time window. MATSim's approach to routing currently provides a fixed departure time, typically expecting a route whose arrival time or cost is as early or as small as possible. This does not work very well for public transport schedules with long headways. If, for example, a service runs only once an hour, an agent might have to wait for more than 50 minutes before boarding and travelling, depending on its departure time. In reality, persons would likely adapt their departure time to the schedule, foregoing extensive waiting times or complex routes with many transfers when more direct ones are available at other times. In MATSim, such an adaption would need to happen separately by the use of the time allocation mutator module, hoping that this (random) mutation brings the agent closer to the optimal departure time. As this requires a large number of iterations without providing any guarantee for better departure times, it is not optimal. SwissRailRaptor supports the calculation of optimal routes within a time range, such that agents could choose the best connection instead of just the next one. But as MATSim does not yet easily allow to adapt the departure time based on a routing result, this variant is not yet integrated into MATSim's replanning process, but only available for stand-alone routing requests. There are plans to adapt MATSim in such a way to more easily incorporate such feedback from the router to the replanning process in order to adapt the departure time, at which point SwissRailRaptor will likely support it.

Intermodal Access and Egress. In the current implementation of the SwissRailRaptor, there is already an intermodal extension, that can be used to use various modes to access from and/or to egress to public transport.

5. Application

Both the deterministic pt simulation (detPTSim) and SwissRailRaptor are now in use as part of SBB's multimodal MATSim model of Switzerland (Scherr et al., 2018). In this chapter, we report on some of the findings of this application, first considering computational performance gains and afterwards assessing the quality of calculated pt routes.

5.1 Performance

As a result of the implementation of the SwissRailRaptor within the MATSim framework, the simulation time of a MATSim scenario has drastically decreased. In the following section, performance of the SwissRailRaptor is analyzed with regard to initialization and query time, memory usage and reduction of numbers of transfers. Also, the effect of the SwissRailRaptor on the MATSim model of SBB (SIMBA MOBi.Modell.CH) is discussed.

The standard MATSim PT Router (from MATSim 0.9.0) and the new SwissRailRaptor are compared using the schedule of the SBB MATSim model (MOBi.OeV.2016). As the structure of the schedule plays an important role on the performance, an artificial schedule (MOBi.OeV.2016_ext) is generated with more routes by splitting the departures on one route to two routes. This keeps the timetable exactly identical and does not force additional transfers upon agents and thus allows to better assess the influence of the schedule on the routing performance. The global statistics of both schedules are shown in Table 1. To benchmark the two routers, a random sample of the SBB synthetic population with a total of 28'000 legs is used, and for each leg a pt route is calculated.

Table 1 Transit Schedules for the performance tests

Name	# Lines	#Routes	#Departures	#Stops
MOBi.OeV.2016	1'914	19'770	209'408	25'064
MOBi.OeV.2016_ext	1'914	36'537	209'408	25'064

5.1.1 Initialization

Keeping the initialization time low and a small memory footprint is an important feature of the SwissRailRaptor. Table 2 summarizes the values measured for the initialization. For both schedules, the initialization time has been reduced by at least a factor of 20 and the number of transfer links by a factor of 2.4. The memory footprint of the SwissRailRaptor keeps its promises and is 10 times smaller. The smaller memory usage makes it possible to set up MATSim with pt scenarios on desktop computers for tests and debugging purposes.

Table 2 Initialization time, memory and number of transfer links

Schedule	Routes	Router	Transfer Links	Time [s]	Memory [GB]
MOBi.OeV.2016	19’770	MATSim PT Router	22’777’634	584	6.4
MOBi.OeV.2016	19’770	SwissRailRaptor	9'713'158	26	0.6
MOBi.OeV.2016_ext	36’537	MATSim PT Router	83'305'258	2627	16.2
MOBi.OeV.2016_ext	36’537	SwissRailRaptor	27’130’326	67	1.61

As explained in 4.2.1, the SwissRailRaptor has seven criteria to filter out transfers between transit stops. These rules reduce the number of transfer links from around 35M to under 10M. Table 3 shows the influence of each of the criteria (note that a transfer could be filtered by more than one criterion). We note that the criterion “destination route has only earlier departures” is filtering out the most transfers. This has certainly to do with the way SBB is modelling its schedule in MATSim. With other schedules, the effects might be different. Still, applying all these rules reduces the mean query time of the SwissRailRaptor from 5.6 ms to 3.6 ms without altering the quality of the router.

Table 3 Analysis of transfer filter criteria

Criterion	Number of transfers	Percentage of all transfers
Cannot reach additional stops by transferring	7’266’779	21%
Transfer leads to a stop served by the same route	3’532’970	10%
Transfer starts at first stop in a route	2’341’250	7%
Transfer leads to last stop in a route	2’370’790	7%
Destination route has only earlier departures	14’534’856	41%
From and to stop of transfer are the same	327’590	1%
Could have transferred earlier in opposite direction	5’754’015	16%
Total filtered transfers	25’091’394	
Total kept transfers	9’713’158	

5.1.2 Query performance

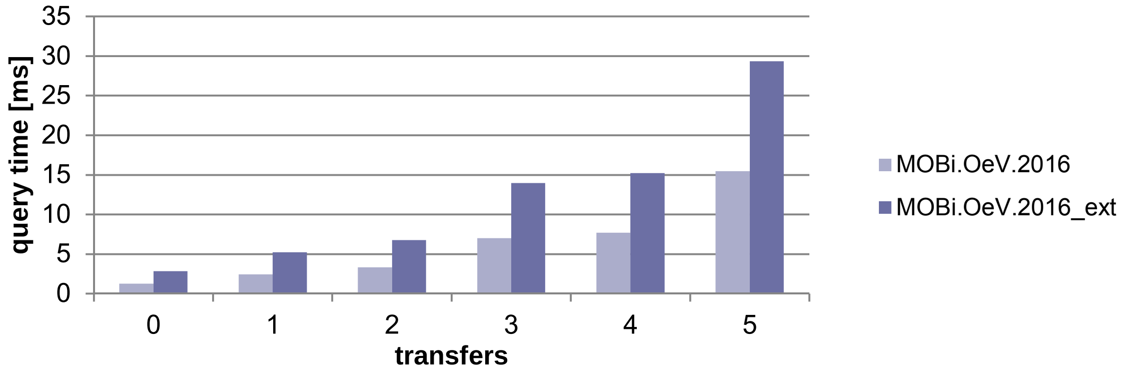
Beyond the initialization, the SwissRailRaptor also performs well in query performance. When calculating 28’000 routes for the randomly selected agents from our population, SwissRailRaptor is 90 times faster than the standard MATSim PT Router (see Table 4). Doubling the number of routes leads to a double of the mean query time, scaling almost linearly. This is mostly due to the way we created the alternative schedule, but clearly shows the impact of the schedule’s structure on the routing performance.

Table 4 Query times statistics for the SwissRailRaptor and the MATSim PT Router.

Schedule	Router	mean query time [ms]	standard deviation [ms]
MOBi.OeV.2016	SwissRailRaptor	3	7.4
MOBi.OeV.2016	MATSim PT Router	273.5	515.8
MOBi.OeV.2016_ext	SwissRailRaptor	7.3	12.1

The standard deviation being larger than the mean query time is due to the query times depending strongly on the number of transfers required to reach the destination stop. Figure 1 breaks down the query time as function of number of transfers. The query time of a journey is exponentially increasing with the number of transfers. We also see that the difference in query time between the two schedules increases as the number of transfers increases. This observation should be put into perspective as a large majority of the pt journeys will have a low number of transfers in typical MATSim scenarios.

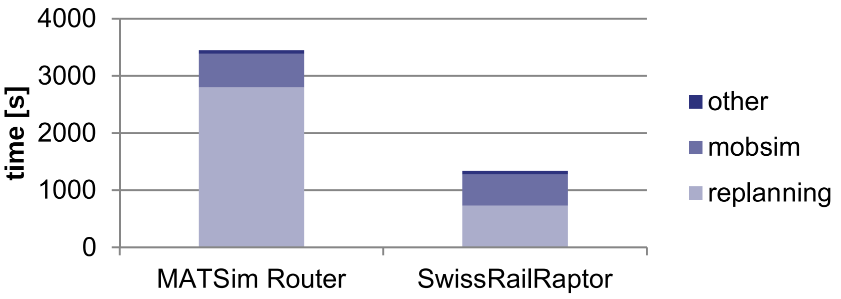
Figure 1 Mean query time of the SwissRailRaptor as function of the number of transfers for two schedules with different numbers of routes



5.1.3 MOBi.Modell.CH

SBB’s MATSim model of the existing state 2015 contains approximately 1 million agents (10% sample of the Swiss population, complemented with exogenous demand from tourists and foreign travelers). Using SwissRailRaptor, the replanning phase of an iteration is 3.8 times faster as when the default pt router is used. Considering all phases of an iteration, this leads to a total speedup of 2.6 per iteration as shown in Figure 2. Using an On-Demand IT infrastructure like AWS (Amazon Web Services), this gain allows for faster results and also cheaper simulations.

Figure 2 Iteration duration averaged over 100 iterations.

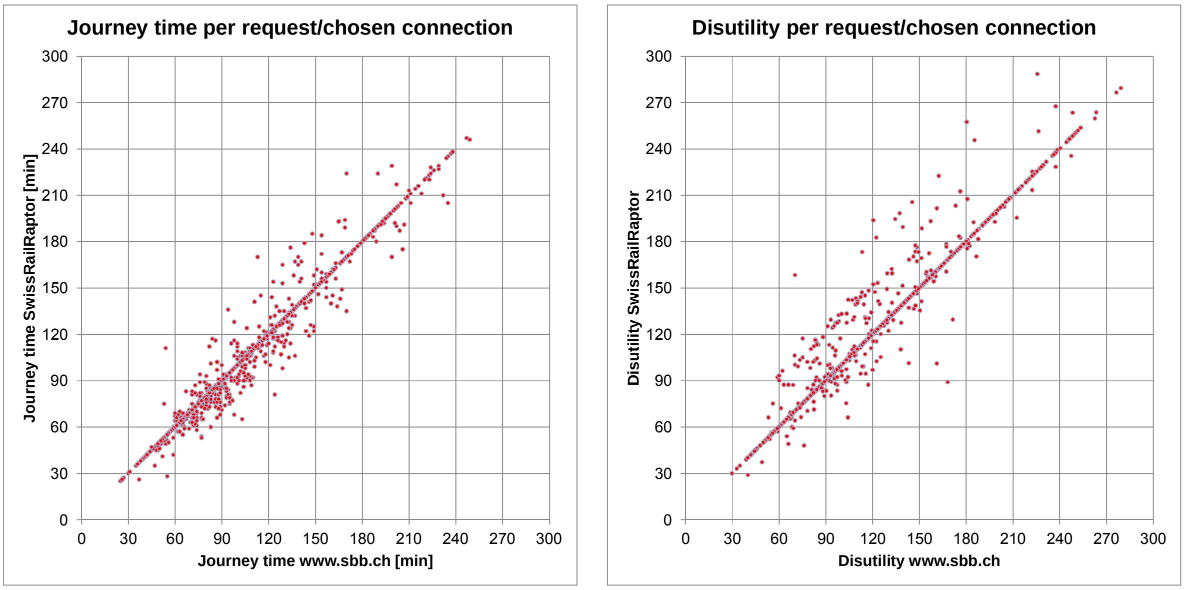


5.2 Quality

We examine the quality of the SwissRailRaptor router by comparing it with the results of the timetable of the entire swiss public transportation system. This timetable is drawn from the public passenger information system (www.sbb.ch). We perform this comparison by making about 1200 typical routing requests on the same fixed date with the SwissRailRaptor and with www.sbb.ch. The schedule for SwissRailRaptor is transformed from the HRDF-timetable-format (HAFAS, 2018) to the MATSim-timetable format using the pt2MATSim (2018) library based on the publicly available HRDF-timetable-data for Switzerland from https://opentransportdata.swiss/de/. We do this by taking into account the stop-specific minimal transfer times (which are defined in the HRDF-timetable-data and can be used now in MATSim as well, see 4.2.2). This guarantees that almost the same timetable basis is used for both routers. Nevertheless, there are still properties in the timetable data which are important for operational timetable systems that cannot yet be modelled in MATSim (see Chapter 6).

One routing request consists of the desired departure time, the origin and destination station of the journey. As www.sbb.ch calculates several connections per request and it is not always clear which connection is the best, we use for the comparison the "departure time window" extension of SwissRailRaptor (see Section 4.2.2). So we get a set of connections for every request to SwissRailRaptor or www.sbb.ch. Each of these connections are assessed by the disutility $f_{AT} \cdot |desiredDepartureTime - departureTime| + f_{JT} \cdot journeyTime + f_{nT} \cdot numberOfTransfers$ and the connection with the lowest disutility is selected from SwissRailRaptor and www.sbb.ch respectively. The following figure compares the journey time and the disutility for these chosen connections:

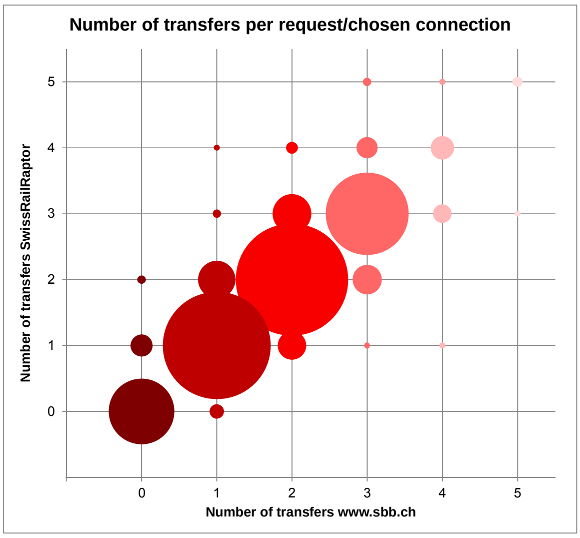
Figure 3 Comparison of journey time (left) and disutility (right) per request/chosen connection



Both scatter plots show one point per request/chosen connection, the x-axis corresponds to the value of www.sbb.ch, the y-axis to the value of SwissRailRaptor. The correlation is clearly evident, both in terms of journey time and disutility. For the travel time the correlation is 98.2%, for the disutility the correlation is 97.4%.

Figure 4 Comparison of number of transfers per request

Number of transfers per request/chosen connection							
SwissRailRaptor	5				1		3
	4		1		10	*16*	1
	3		1	25	*198*	13	2
	2		24	*367*	44	4	
	1	6	*337*	41	2	1	
	0	*125*	14	2			
		0	1	2	3	4	5
		www.sbb.ch					



The two evaluations in Figure 4 show the relationship between the number of transfers in connections found by www.sbb.ch and SwissRailRaptor. The number in cell (j, i) (row j, column i) in the matrix to the left corresponds to the number of requests with i transfers in www.sbb.ch and j transfers in SwissRailRaptor. In 84.5% of the requests the number of transfers in www.sbb.ch and SwissRailRaptor is equal. The figure on the right shows the same graphically (the radius of the circles corresponds to the number of requests).

6. Outlook

Both newly developed MATSim modules, the deterministic public transport simulation and the SwissRailRaptor, had a very positive impact on the quality and performance of SBB's nation-wide MATSim model. While the results are highly satisfying, there are already several ideas how public transport in MATSim can be improved even further. Among those things are:

- Adapting MATSim so the departure time (actually the previous activity's end time) can adapt to the "best" journey returned by the router.
- Support for run-through services, which are common in railway schedule. A run through service is a train run, where the same vehicle first serves one line and afterwards another line, and passenger can remain inside the vehicle to connect without having to transfer at a stop. This would also eliminate artificial transfers on circular lines, which are currently modelled to run from one to the same stop in a loop.
- Support for split and joined trains, where parts of one train run splits into two different destinations.
- Specific transfer times between two routes (or between two vehicle runs) on a particular stop, that are different from the general transfer time on this stop. (e.g. the connecting train is departing just on the opposite platform).
- Transfer priorities, as not all stops might provide the same ease of transfer between two lines. This could lead to further pruning of transfers, and thus even higher performance of SwissRailRaptor.
- Stops that are served only for boarding or alighting, as it is often the case with night trains or in rural areas in off-peak hours (e.g. bus lines only departing at a station if passengers exit from a train).
- Modeling of public transport fares in the router but also in plan scoring. The current implementation does not directly support fares, but only applies costs based on travel times and transfers.

7. Conclusions

The newly developed MATSim modules helped enormously to improve the quality of SBB's MATSim model. First, the much faster SwissRailRaptor allowed us to perform more runs in the same time as MATSim's standard pt router would have. Besides being able to use the hardware more efficiently, the shorter feedback times were especially helpful to make faster progress while calibrating the model. Second, the deterministic PT simulation produces more realistic passenger flows, reduces simulation artifacts, makes the effect of parameter changes more predictable and the analysis of simulations and thus the calibration process more stable. The additional features of SwissRailRaptor like stop-dependent transfer times create more realistic passenger-routes, leading to a better fit between the model and empirical data (e.g. number of boardings/alightings per stop). In addition, SwissRailRaptor provides additional features (e.g. person- and route-specific parameters) that will help us advance our multimodal MATSim model even further in the future in order to better predict changes in the demand to the offered services—and ultimately provide better services to SBB's passengers.

The developed software modules for MATSim are available as open-source software from https://github.com/SchweizerischeBundesbahnen/matsim-sbb-extensions and thus also mark the first open-source contributions from the Swiss Federal Railways to MATSim.

We are confident that other MATSim models in regions with significant public transportation demand and schedule-based services will benefit from the applications of these new modules.

Acknowledgements

We would like to thank the Swiss Federal Railways (SBB) for providing the resources to work on MATSim as part of building the multimodal MATSim model, Jörg Ruppe (SBB) to provide us access to a high-performance server which helped us to quickly get the first versions of the model up and running, the SBB Cloud team to help us figure out and set up the required computing resources, and Han van der Kleij (SBB) for coaching us in Agile Development practices.

The project would not have been possible without being able to base the work on the foundation of MATSim, for which we'd like to thank the MATSim community. A special thank goes to Prof. Dr. Kai Nagel for helping us formulate and prioritize the requirements to improve public transportation in MATSim.

Bibliography

Bast H., E. Carlsson, A. Eigenwillig, R. Geisberger, C. Harrelson, V. Raychev and F. Viger (2010) *Fast Routing in Very Large Public Transportation Networks using Transfer Patterns.* In: de Berg M., Meyer U. (eds) Algorithms – ESA 2010. ESA 2010. Lecture Notes in Computer Science, vol 6346. Springer, Berlin, Heidelberg.

Delling D., T. Pajor and R. Werneck (2012) *Round-Based Public Transit Routing*. In: Proceedings of the 14th Meeting on Algorithm Engineering and Experiments (ALENEX'12).

Dibbelt J., T. Pajor, B. Strasser and D. Wagner (2013) *Intriguingly Simple and Fast Transit Routing*. In: Bonifaci V., Demetrescu C., Marchetti-Spaccamela A. (eds) Experimental Algorithms. SEA 2013. Lecture Notes in Computer Science, vol 7933. Springer, Berlin, Heidelberg.

Dijkstra, E. W. (1959) *A note on two problems in connexion with graphs*, Numerische Mathematik, 1, 269–271.

Friedrich, M., I. Hofsäß, S. Wekeck (2001) *Timetable/based Transit Assignment Using Branch & Bound Techniques*, Transportation Research Records, No. 1752, p.100-107.

HAFAS (2018) HaCon Fahrplan-Auskunfts-System, webpage, http://hacon.de/hafas

Horni, A., K. Nagel and K.W. Axhausen (eds.) (2016) *The Multi-Agent Transport Simulation MATSim*. Ubiquity Press, London. http://dx.doi.org/10.5334/baw

MATSim (2018) Multi-Agent Transport Simulation, webpage, https://www.matsim.org

Meister, K., M. Rieser, F. Ciari, A. Horni, M. Balmer and K.W. Axhausen (2009) *Anwendung eines agentenbasierten Modells der Verkehrsnachfrage auf die Schweiz*. Strassenverkehrstechnik, 53 (5) 269-280.

OpenTrack (2018) OpenTrack – Simulation of Railway Networks, webpage, http://www.opentrack.ch

Pt2MATSim (2018) Package to create a multi-modal MATSim network and schedule from public transit data (GTFS or HAFAS) and an OSM map of the area, webpage, https://github.com/matsim-org/pt2matsim

PTV Group (2018), webpage, https://www.ptvgroup.com

Raney, B., N. Cetin, A. Völlmy, M. Vrtic, K.W. Axhausen and K. Nagel (2003) *An agent-based microsimulation model of Swiss travel: First results*. Networks and Spatial Economics, 3 (1) 23-41.

Rieser, M. (2010) *Adding Transit to an Agent-Based Transportation Simulation: Concepts and Implementation*, Ph.D. Thesis, TU Berlin, Berlin, http://dx.doi.org/10.14279/depositonce-2581.

Rieser, M., D. Grether and K. Nagel (2009) *Adding mode choice to a multi-agent transport simulation*, Transportation Research Record, 2132, 50–58, http://dx.doi.org/10.3141/2132-06.

Scherr, W., P. Bützberger, N. Frischknecht (2018) *Micro Meets Macro: A Transport Model Architecture Aiming at Forecasting a Passenger Railway's Future*. Conference paper. Swiss Transportation Research Conference. Ascona.

Strasser, B. and D. Wagner (2014) *Connection Scan Accelerated*. In: Proceedings of the Sixteenth Workshop on Algorithm Engineering and Experiments (ALENEX'14).

Witt, S. (2015) *Trip-Based Public Transit Routing*. In: Bansal N., Finocchi I. (eds) Algorithms – European Symposium on Algorithms 2015. Lecture Notes in Computer Science, vol 9294. Springer, Berlin, Heidelberg.

Witt, S. (2016) *Trip-based public transit routing using condensed search trees*. In: Proceedings of the 16th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems (ATMOS'16), volume 54 of OpenAccess Series in Informatics (OASIcs).