GTARC
GERMAN TURKISH
ADVANCED RESEARCH CENTRE FOR ICT

IMA

Intermodal Mobility Assistance

IMA
INTERMODAL
MOBILITY
ASSISTANCE

Deliverable 8.1 (b)

Integration, Testing and Validation – Simulation Studies

DLR Projektträger

GEFÖRDERT VOM
Bundesministerium
für Bildung
und Forschung

Concept of evaluation for IMA: simulator environment

The evaluation of the IMA system can be divided into multiple parts: a real user test, a qualitative comparison with other platforms, and a performance analysis using the IMA simulator. The third part will be further elaborated in this document.

Considering the nature of the simulator, it is above all suited for quantitative analyses, crunching big numbers of route queries simultaneously and analyzing big datasets about the results. Therefore, we decided that there are two main objectives for using the simulation engine:

1. Analyzing the performance of IMA route suggestions for unimodal vs. intermodal trips
2. Comparing IMA route suggestions to the Google Maps API

Side objective: analyzing the processing time according to number of queries per minute, kind of trips, etc. (maybe analyzing the potential of reducing the time by providing more agents with load-balancing)

Simulator performance evaluation

Before performing the large simulator tests of the next parts, some test runs should be performed that indicate the performance of the simulator. Specifically, it is not yet clear how long the engine runs for a defined set of iterations, users, locations, etc. Therefore, some sample tests will be described in the following part.

These are the variables that can be changed for the the test samples:

1. Number of users (profiles that can have different preferences)
2. Number of iterations (# of trips for each user profile)
3. Location radius (size of area where origin/destination is randomly picked) *(this should not affect performance very significantly, I think)*
4. Route distance (between origin and destination areas. Presumably the longer the distance, the longer the search time, but the effect might be small)
5. Number of modes
6. Uni-/intermodal trips

3 probably does not have a very high effect on performance as the size of the area does not matter, but the number of origin/destination locations inside the area, so this variable can be ignored (an extra test for the sake of interest is always possible). Because 4 cannot be adjusted for the tests we want to make, this variable can be ignored too.

Combining these variables, different ‘sample sets’ can be designed for testing:

A0 - Base version (like when you would use IMA as a client).
1 user, 1 iteration, all modes, intermodal trips
A1 - Base – 2 users, 1 iteration
A2 - Base – 4 users, 1 iteration
A3 - Base – 8 users, 1 iteration
A4 - Base – 1 user, 10 iterations
A5 - Base – 1 user, 25 iterations

A6 - Base – 1 user, 50 iterations
A7 - Base – 1 user, 100 iterations
A8 - Base – 1 user, 200 iterations

B0 – Mass test – 2 users, 25 iterations (50 in total)
B1 – Mass test – 2 users, 50 iterations (100 in total)
B2 – Mass test – 8 users, 25 iterations (200 in total)
B3 – Mass test – 8 users, 50 iterations (400 in total)
To check the effect of unimodal/only one mode queries, the test sample with the longest search time can be adjusted (this can be another test sample if B3 turns out to take very long)
B4 – Mass test – 8 users, 50 iterations, only unimodal trips
B5 – Mass test – 8 users, 50 iterations, only intermodal trips
B6 – Mass test – 8 users, 50 iterations, only car (unimodal)
B7 – Mass test – 8 users, 50 iterations, only PT (unimodal)
B8 – Mass test – 8 users, 50 iterations, only bike (unimodal)

For each sample, the runtime will be registered and the effects of variables will be analyzed. With these results, the optimal number of users and iterations can be defined. If there is a linear relationship between users*iterations and runtime, then there is no use in splitting up queries in separate sets, but if the relationship is exponential, there can be a huge advantage. In the meantime, this document uses a suggested number of 50 iterations per set.

Apart from this, there might be other ways to reduce runtimes for the simulator (providing more agents with load-balancing?). This might be tested by the guys responsible for the simulator.

1. Uni- vs. intermodal trips performance

The goal of 1) is to derive the quality of the intermodal route suggestions provided by IMA, compared with only unimodal alternatives. To do this, we should first define what unimodal and intermodal trips mean, and how we will assess their performance.

- Unimodal trips: trips that use only one mode, for example public transport / car / bike / sharing bike/car. Keep in mind that walking towards a public transport station or towards a sharing bike/car location does not count as an extra mode. Changing between public transport vehicles (bus, tram, U-Bahn, S-Bahn, train) also counts as using one mode.
- Intermodal trips: trips that use at least more than one mode. This can be for example biking to a station and taking the tram, or taking the train and driving the last kilometers by sharing cars. Keep in mind that walking + PT/sharing service does not count as an intermodal trip.

There are different variables that can be changed to analyze the effects on IMA's performance:

- Location: in the simulator, a radius can be drawn around origin and destination locations. The system then searches for random routes to and from the areas inside these circles. It would be interesting to compare differences in performance between long and short distance trips, for example inner-center trips vs. outskirt-city trips. It is hypothesized that intermodal trips are more suitable for longer distance trips.
- Preferences: in IMA, different weights can be given to travel time, travel costs and CO2 exhaust. Considering this, we can analyze whether the system gives better uni-/intermodal trip advice for users with specific preferences (i.e. high weight for CO2 exhaust). This can be done by attributing different profiles to virtual 'users' of the system.

A current issue with the IMA engine is that, when a lot of queries need to be processed at the same time, the search time can amount exponentially. Therefore, strategies need to be devised to have faster results. A possible way to do this, is to split up the simulation queries into different parts, so that there are already results to analyze when other queries still need to be processed. For example, the queries can be split up according to the variables mentioned earlier: uni-/intermodal, origin-destination combination and preferences.
In that way, one query 'package' could include 'unimodal trips from A to B in the city center for users with high weights for travel time'. An extra way to limit the search time is to restrict the number of iterations per query (for example 100 or 500 iterations), depending on how fast the engine can handle this. A test run before analyzing 'real' results could give more information on runtime.

Origin-destination definition

For this test, a distinction can be made between inner-city trips and outskirts-to-center trips.

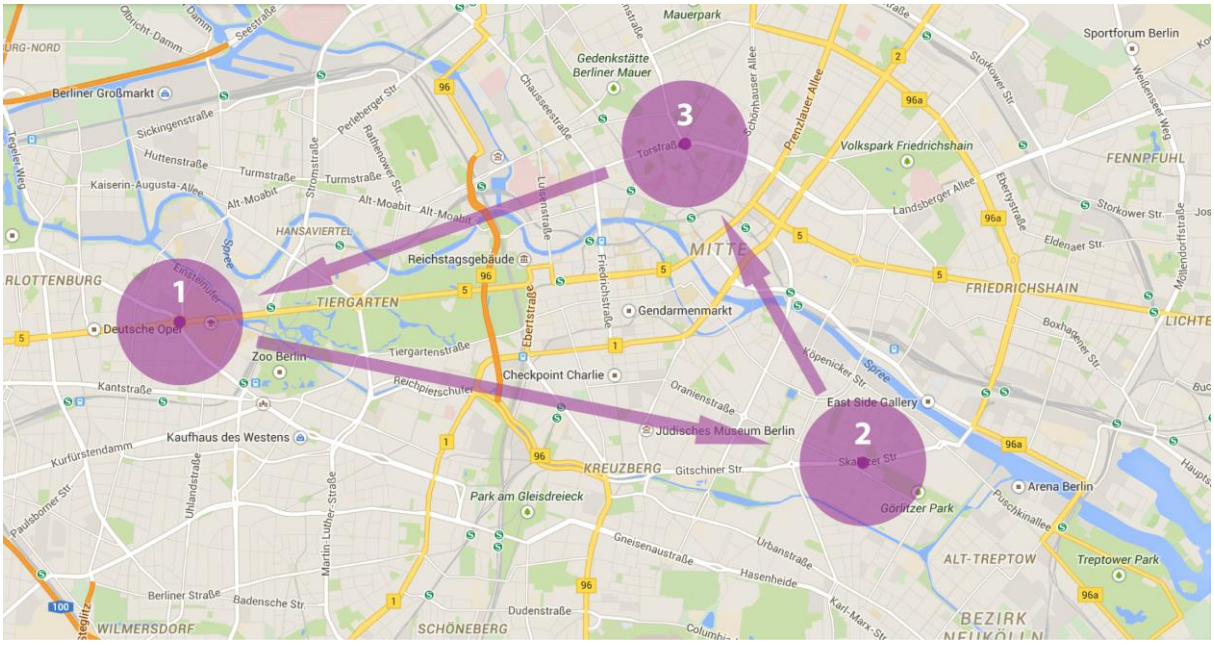
Inner city trips

To prevent bias originating from one specific origin-destination combination, but to limit the number of possible combinations and thus computations, three locations in the city center are chosen and combined to give a variety of inner-city trips.

1. Ernst-Reuter-Platz (TU-Berlin)
2. Görlitzer Bahnhof (Skalitzer Str – Wiener Str)
3. Rosenthaler Platz (Brunnenstr – Torstr)

Around these locations, a radius of 500m (thus 1 km diameter circle) is drawn wherein the simulator can randomly pick an origin/destination.
This will then be the three origin-destination combinations:

a) Ernst-Reuter-Platz ➔ Görlitzer Bhf
b) Görlitzer Bhf ➔ Rosenthaler Platz
c) Rosenthaler Platz ➔ Ernst-Reuter-Platz

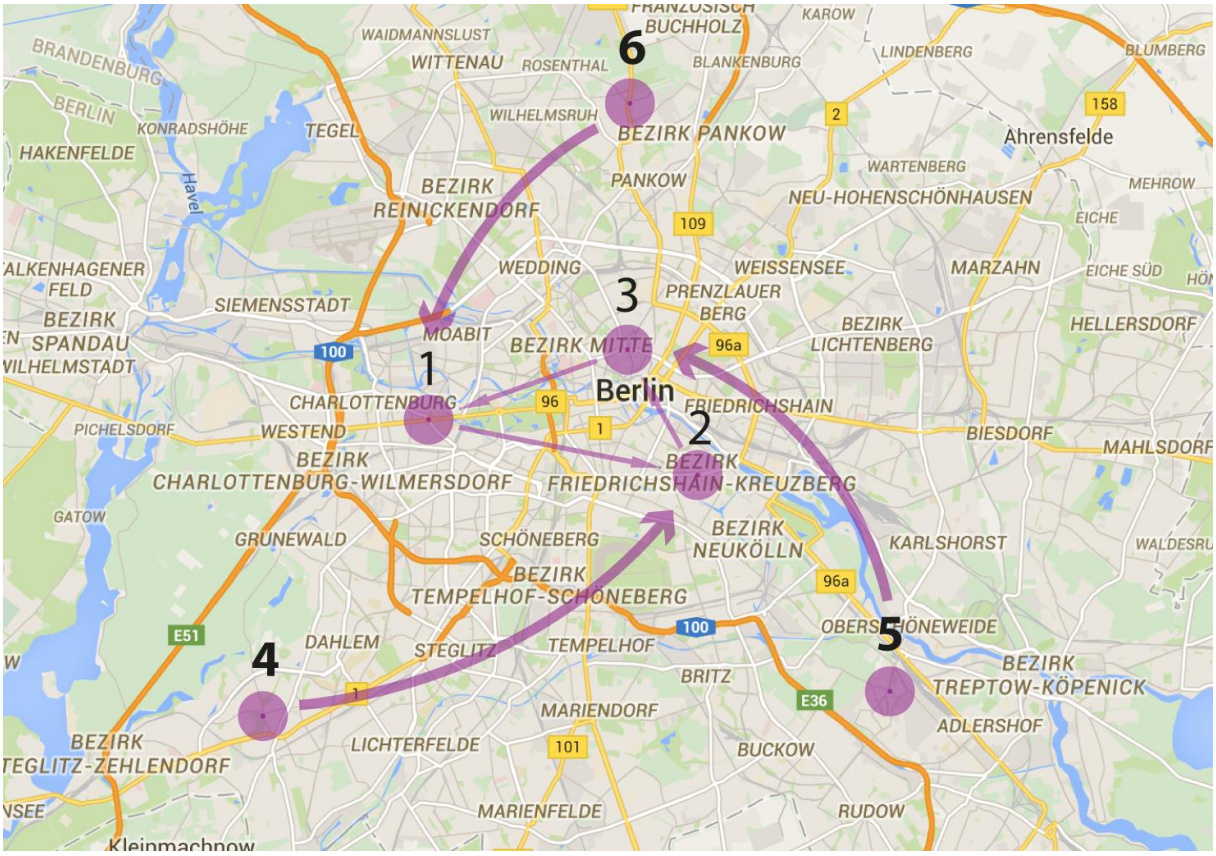


Outskirts-to-center trips

4. Zehlendorf (Onkel-Tom-Str – Schützallee)
5. Johannisthal (Sterndamm – Königsheideweg)
6. Pankow Nordend (Dietzgenstr – Schillerstr)

These locations were chosen randomly in three of Berlin's outskirts towns. In total, 9 single-direction combinations with the inner-city locations can be made, but for the sake of simplicity and to prevent too many computations, three are chosen that are not the shortest, most direct links:

d) Zehlendorf ➔ Görlitzer Bahnhof
e) Johannisthal ➔ Rosenthaler Platz
f) Pankow Nordend ➔ Ernst-Reuter-Platz



These 6 origin-destination combinations can then be combined with different user preferences, that will be defined in the next part.

Preference definition

In IMA, three preferences can be set with weights. Adjusting the weight of one parameter will automatically adjust the weights for the other two. In that way, four 'distinct' user profiles can be set:

1. Neutral (33%, 33%, 33%)
2. High value for time (100% time)
3. High value for costs (100% costs)
4. High value for CO2 emissions (100% CO2)

These preference sets can be entered in the simulator as different 'users'. But first, the users need to be 'doubled' for the real purpose of the test: checking performance differences between intermodal and unimodal trips. Therefore, each user profile exists as one that only makes unimodal trips (u), and one that only makes intermodal trips (i).

User set: 1u 1i 2u 2i 3u 3i 4u 4i

Tests

Combining these 8 user preference profiles with the 6 OD-combinations, there are 48 unique trip-user combination sets that need to be run in the simulator. Depending on the performance speed of the simulator, a specific number of iterations can be used. For example, each set can have 50 iterations (= 50 random trips from and to the defined

radiuses). This would give a total of 2400 trips to be computed. This number can be adjusted if test runs indicate this is too much.

A possible distribution of computation runs can be divided over the different user preferences and OD-combinations, for example:
a1u and a1i
a2u and a2i
…
b1u and b1i
b2u and b2i
…
until f4u and f4i
This would result in 24 separate computation runs with 2x50 iterations each.

Evaluation

With the first sets of results becoming available, the evaluation can already start on some aspects. For example, the performance differences between uni- and intermodal trips for one particular OD-combination can be analyzed (i.e. difference between a1u and a1i, 100 iterations). Possible evaluation criteria are: travel time, travel costs, exhaust, search time, … . (This depends on the data that can be obtained from the simulator, not completely certain yet).

When all computations are done and all data is obtained, comparisons can be made between different groups of iterations:

i. All unimodal trips vs all intermodal trips
ii. Performance of unimodal inner city trips vs intermodal inner city trips
iii. Performance of unimodal outskirt-to-city trips vs intermodal ones
iv. Comparison of inner city vs outskirt-to-city performances
v. Differences between the 4 user preference profiles in unimodal and intermodal trips

2. IMA vs. Google Maps performance

The goal of this part is to assess the performance of IMA's route suggestions, by comparing them to the same queries in Google Maps (using their API so this can be done automatically in the simulator). As Google Maps does not provide intermodal trips, it is not possible to compare these, and neither are sharing car/bike services. Nevertheless, unimodal car, bike and PT trips from both the IMA and Google engines can be compared.

To do this, separate queries for each mode need to be inserted in the simulator (for example 'public transport trips from A to B'. Unfortunately, Google does not provide preferences for travel time, travel costs and CO2 exhaust like IMA does, so the IMA preferences should reflect Google's routing algorithm preferences. Google primarily focuses on finding the quickest routes, instead of looking at transport costs or exhaust. Thus, the user preferences for IMA should reflect this high travel time weight. The objective of this analysis thus is primarily oriented towards the reliability of IMA's suggestions, compared to Google's system.

Again, the query packages should be split up in order to limit search times. For this part, queries can be split per mode and per origin-destination combination.

Another API that might be introduced to the simulator is Allryder, a Berlin-based route planner company that provides multimodal (not intermodal) route suggestions. They provide enterprise solutions for their platform at http://enterprise.allryder.com/. Note: since April 22nd, 2015, the app Allryder has been renamed to Ally, but the enterprise page is still accessible under the name 'Allryder'.

Other platforms, like TripGo and Qixxit (see my Comparison with other platforms document), will be difficult to compare with in the simulator because they do not have an API that can be used. Thus, comparisons with IMA would need to be done manually (example cases in aforementioned document). These comparisons would then be more qualitatively oriented towards usability and overall experience.

Profile definition

For this test, IMA's route suggestions for car, bike and public transport can be compared with Google's suggestions. As Google does not have the ability to give weights to different preferences and takes travel time as most important value instead, IMA's user preferences should be set to 100% for travel time. Neither is it possible in Google to obtain intermodal routing advice, so IMA's user profiles should be set to unimodal trips.

This results in six different user profiles:

1. IMA car (time value 100%)
2. IMA PT ""
3. IMA bike ""
4. Google car
5. Google PT
6. Google bike

For simplicity, the same six origin-destinations as in the previous test can be used (and in that case, even comparisons with the intermodal IMA trips vs Google can be made), the only difference being that the area radius needs to be much smaller so that the same origin and destination locations as in Google can be tested. That way, not that many iterations are needed. This results in 6x6 route sets. Depending on the number of iterations that will be used (this still needs to be decided), these sets can be run together or apart from each other.