Using MATSim as a travel model plugin to UrbanSim

Kai Nagel

December 15, 2008

1 Getting started

1. Note: There is the opus source tree, and the opus data directory. The first is where all the python code it; the second is where `OPUS_HOME` points to.

2. Download `opus_matsim-yyyy-mm-dd-hhmm.tgz` from

   `https://fgvsp01.vsp.tu-berlin.de/repos/shared-svn/public/scratch/opus_matsim/`

   and move it into `OPUS_HOME`.

3. Go into `OPUS_HOME` and type

   ```
   tar zxvf opus_matsim-yyyy-mm-dd-hhmm.tgz
   ```

   After that, you should have an `opus_matsim` directory, with subdirectories

   ```
   data          [[scenario input data]]
   matsim_config [[matsim config files]]
   bin           [[binaries]]
   classes       [[java classes that go beyond standard matsim]]
   jar           [[empty]]
   ```

4. Download `MATSim_libs_rXXXX.zip` and `MATSIM_rYYYY.jar` from www.matsim.org, on the "nightly builds" www page (currently: `http://matsim.org/files/builds`; some background info on `http://matsim.org/downloads/nightly`), and move it into `OPUS_HOME/opus_matsim/jar/` .

5. Go into `OPUS_HOME/opus_matsim/jar` and type

   ```
   unzip MATSim_libs_rXXXX.zip
   ln -s MATSim_rYYYY.jar MATSim.jar
   ```

   Afterwards, you should have

   ```
   .../opus_matsim/jar/MATSim.jar  [[as symbolic link]]
   .../opus_matsim/jar/MATSim_rYYYY.jar
   .../opus_matsim/jar/libs/...    [[lots of *.jar files]]
   ```

6. Go into the opus source tree, into `opus_matsim/tests`, and python-run `tests.py`.

   That runs a meaningless test scenario, and should run without errors. (It actually does not use the java stuff; it just tests the opus side.)

7. Go into the opus source tree, into `opus_matsim/configs`. You should be able to run `seattle_parcel.xml` by any of the normal means (from the GUI, from the command line, or by using `start_run.py` in the same directory).

   If this runs through, you are ready to go!

2 Visualizer

This section works only unter macosx and linux-i586. linux-amd64 should be straightforward (let me know if you can't figure it out). win also works in many cases, but I don't know where to put the dynamic library.

There should now be output in `OPUS_HOME/opus_matsim/output`. Go into `ITERS/it.0`. There should be a file `0.otfvis.mvi`. Type

```
  OPUS_HOME/opus_matsim/bin/matsim-vis-XXXX 0.otfvis.mvi
```

where `XXXX` refers to the version that is correct for your OS. This should bring up the visualizer with the usual play and stop buttons.

3 Configuration on the OPUS side

Look at `seattle_parcel.xml`. The relevant section is in `travel_model_configuration`.
The first part is

```
<models type="selectable_list">
 <opus_matsim.models.get_cache_data_into_matsim choices="Run|Skip" type="model_choice">Skip</opus_matsim.models.get_cache_data_into_matsim>
 <opus_matsim.models.run_travel_model choices="Run|Skip" type="model_choice">Run</opus_matsim.models.run_travel_model>
 <opus_matsim.models.get_matsim_data_into_cache choices="Run|Skip" type="model_choice">Run</opus_matsim.models.get_matsim_data_into_cache>
</models>
```

This should be pretty clear.

Next is

```
      <sampling_rate type="float">0.01</sampling_rate>
```

This denotes the sampling rate on which MATSim runs. 0.01 means only one percent of travellers are considered. This causes some peculiarities in terms of realism; but I would still recommend to leave this as is until it is really clear how things work (in particular the "warm start" capability).
Next comes the general matsim config file:

```
      <matsim_config_filename type="file">matsim_config/seattle_matsim_0.xml</matsim_config_filename>
```

Note: The root to this is `OPUS_HOME`.
Next are the years in which the travel model should be run:

```
    <years_to_run key_name="year" type="category_with_special_keys">
      <run_description type="dictionary">
        <year type="integer">2001</year>
      </run_description>
      <run_description type="dictionary">
        <year type="integer">2002</year>
        <matsim_config_filename type="file">matsim_config/seattle_matsim_2002.xml</matsim_config_filename>
      </run_description>
      <run_description type="dictionary">
        <year type="integer">2003</year>
      </run_description>
    </years_to_run>
  </travel_model_configuration>
```

Note that you can over-ride the general matsim config file in any given year. This will read the special matsim config file *instead* of the general one.

It seems to me that the above configuration commands can be *either* into `model_manager` *or* (somewhere) into `scenario_manager`. I only tested the first set-up.

4 Configuring MATSim

MATSim is configured by modifying the MATSim config files, which are referenced as described above. This is regular MATSim set-up and described elsewhere (seach in `matsim.org`).

Noteworthy are the following items:

- The root of the MATSim configuration is `OPUS_HOME/opus_matsim`. I do not know if this is good or not, but it makes life a bit simpler with a couple of aspects.

  (That is, the `../opus_matsim/...` is the same as `./...`. There was a reason why that was needed; need to check ...)

- `inputNetworkFile` gives the road network, in MATSim format. This should have come with the `*.tgz`-file. Note that any `seattle` scenario still uses the full PSRC network.

- `inputPlansFile` gives the "relaxed" plans file from which MATSim starts. If this is commented out, MATSim will construct its initial plans file purely from OPUS input, and take much longer to relax.

- `flowCapacityFactor` and `storageCapacityFactor` should correspond to the `sampling_rate` above.

  (For the `seattle` scenario, I have set the flow capacity to only half of this because there is otherwise not congestion. This is most probably due to the fact that the cut-out is was debugging purposes only, and did not correct the resulting boundary effects (e.g. trips to jobs outside the scenario, or from homes outside the scenario, are simply dropped).)

- `lastIteration` gives the last iteration, and in consequence the number of iterations.

5 Warm start

The above uses a pre-existing plans file from which to start the iterations.

5.1 Generating a warm start plans file

The generation of a warm start plans file was in principle already described above:

1. Remove the `inputPlansFile` entry ("cold start"), and set the number of iterations to a plausible number (say "100").
2. Then run the UrbanSim-MATSim set-up until MATSim has finished.
3. Then search for the resulting plans file in

   `opus_matsim/output/ITERS/it.<lastIt>`,

   and move it into a convenient place (e.g. `opus_matsim/data/...`).
4. Finally configure the `inputPlansFile` so that it uses this newly generated "relaxed" plans file.

5.2 Warm start runs

Note that in this situation, MATSim will always start from that same initial plans file. I.e. if you keep calling MATSim during an UrbanSim run, with a changing population, the MATSim initial plans file will be less and less correct, and you will need more and more iterations to get MATSim back into a relaxed state.

Technically, what it will do is keep all persons that have not changed (i.e. are still there, live at the same home, have the same employment status, and, if applicable, work at the same place). All other persons are reset to a "zero" state with respect to MATSim.

See "hot start" below for more info.

6 Hot start

It would be desirable to have the output of one MATSim run for one UrbanSim year available as input for the next MATSim run in another UrbanSim year.

This is meant by "hot" start (as opposed to warm start).

If you really know what you are doing, this can be achieved by the following:

1. Make a separate MATSim config file for every UrbanSim run in which you want to run MATSim.
2. Enter these config file names into the OPUS config file for every year of the run.
3. In the MATSim config file, make sure that for every call of MATSim *except the first one*, the following is used:

```
<param name="inputPlansFile" value="../opus_matsim/output/output_plans.xml.gz" />
```

4. Use consecutive iteration numbers, e.g.

   First MATSim call:

   ```
   <param name="firstIteration" value="0" />
   <param name="lastIteration" value="99" />
   ```

   Second MATSim call:

   ```
   <param name="firstIteration" value="100" />
   <param name="lastIteration" value="199" />
   ```

   etc.

It will, however, be relatively difficult to see if this really worked. So, for the time being, I would rather consider this a theoretical possibility than a practical option.

7 Caveats

Many.

- As usual everything was ripped up again in the last week of the project, because after my project presentation we found really much better ways of doing things. But it was ripped up, and not much used/tested afterwards. Hard to say if that was a good thing, but it is considerably faster now.
- *Important:* Liming and I put in a way to tag the agents on the OPUS side. *However,* I removed that again. So please do not assume that this is still there.

  (Reason: It became really difficult to have the same tagged persons sample from one UrbanSim run to the next, and this made the warm start capability close to impossible. What it does now instead: Go through the *complete* UrbanSim population and pick those persons that are in the pre-existing MATSim plans file.)
- There are peculiarities associated with running MATSim with a 1% sample. I would say that the advantages (much easier computing) weigh more heavily than the disadvantages. But the results are useful for sketch planning only, not for quantitative analysis. Feel free to try 10% or even 100% if you know what you are doing, and have the patience, RAM, and disk space.
- The feedback from MATSim to UrbanSim is really not that well established. Right now, it only overwrites the `single_vehicle_to_work_travel_cost` value, and it over-writes this in a unit that is different from what was used previously. (Now it is in "seconds"; previously it was ??, and in spite of the name as of now it does not look at toll.)

  This is not so much due to the fact that there is no better solution, but due to the fact that the UrbanSim standard seems to be pretty weak: The documentation talks about `logsum1`, `logsim2`, and `logsum3`, but I have not seen them anywhere in the psrc/seattle examples.

  In addition, there seems no default way to use these values: psrc/seattle has specific python functions that eventually use some of the travel model output.