BIOROUTE C++ documentation

Introduction

This documentation explains the usage of bioroute v1.0, a implementation of Metropolis-Hastings path sampling (MHPS) algorithm for route choice modeling using GPS data.

Features:

1. MHPS algorithm.
2. Random walk algorithm.
3. Deals with probabilistic map matching results.
4. Generation of biogeme inputs (model file and data file).

Installation

The software is programed in C++. The building environment is C++. The software has been tested on Mac OSX, and Linux (Ubuntu 12.04). Dependencies

- Boost > 1.49
- biogeme (TODO put biogeme in path)
- Postgresql > 9.0
- libpqxx
- Nnu GSL
- libshp
- libkml
- libconfig
- xmlwrapp
-

It is likely that cmake fails to find some dependencies. In this case, please update the corresponding package look-up file in src/cmake/Modules/, see documentation http://www.cmake.org/Wiki/CMake:How_To_Find_Libraries.

Mac OSX

The software has been tested on Mac OSX 10.7, and 10.8. In Mac OSX, the XCode and command line tool have to be installed. The link to the header files of XCode should be given in src/CMakeLists.txt, for example:

```
include_directories("/Applications/Xcode.app/Contents/Developer/Platforms/MacOSX.platform/Developer/SDKs/MacOSX10.7.sdk/usr/include/")
```

Linux

g++ 4.4 is tested. It should be noted that C++11 implementation is required for the software.

Srv4

The software is installed on srv4 under /data/jingmin/bin/

Usage

The basic usage is following:

```
bioroute -c COMMAND -f PARAM_FILE -r -n NBR_OBSERVATIONS -s
```

`-c COMMAND`	the function to be executed
`-f PARAM_FILE`	is the path to the parameter file which is explained in Section Parameters.
`-r`	specified if the observations synthetic , not real.
`-s`	specified if the network is synthetic, not real.
`-n NBR_OBSERVATIONS`	number of observations (only for *SimulateObservation*)

COMMAND

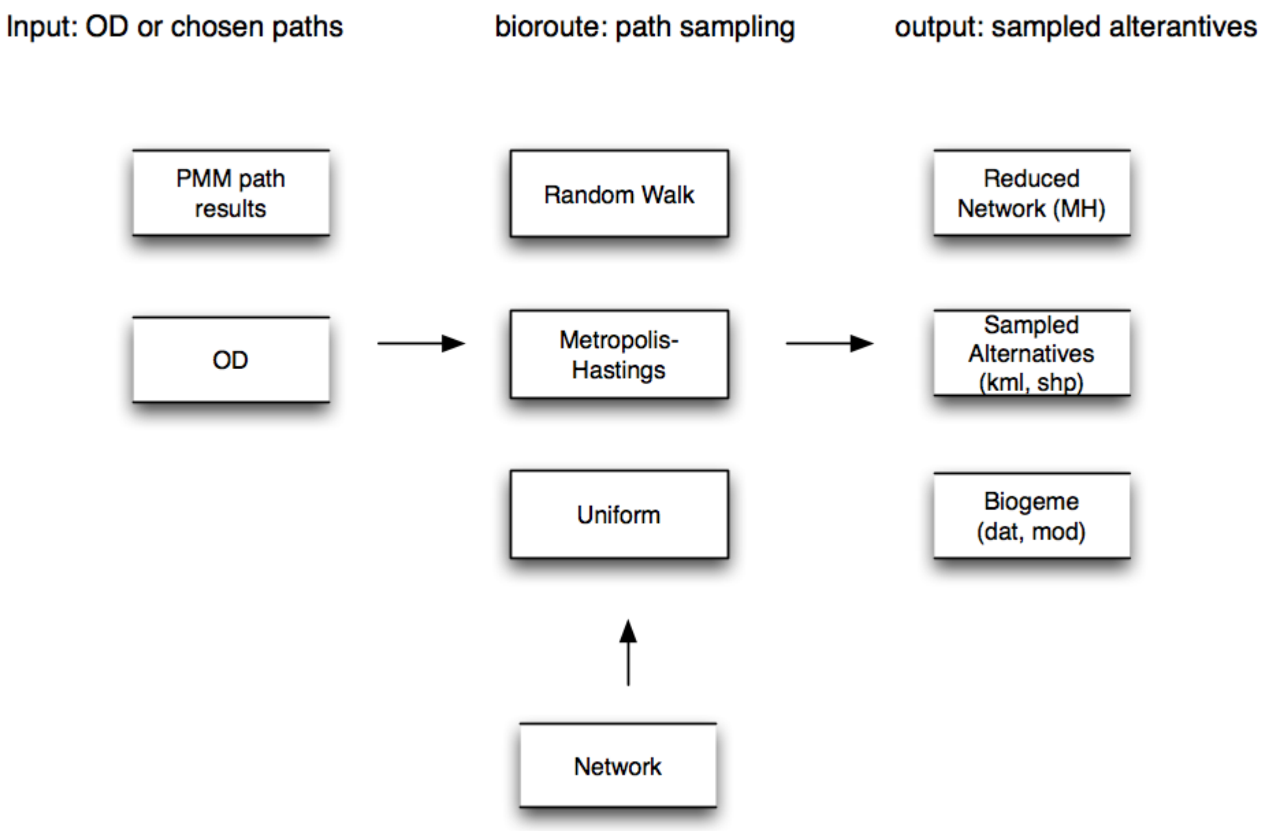
- *Sample*: sampling alternatives for route choice observations.
- *SampleWithOd*: sampling alternatives with the given OD.
- *SampleEqualProbability*: sampling alternatives with equal probability
- *MHEnumeratePaths*: enumerate paths
- *WriteBiogeme*: write biogeme model and data files
- *SimulateObservation*: simulate route choice observations with specified route choice model.
- *ExportNetwork*:
- *TestNetwork*:
- *Verify*: verify sampling results.
- *SimulateObservationsError*: simulation errors in observations.
- *WriteNetworkToDB*: write network to postgresql database.
- *AnalyzeChoiceSet*:
- *WriteChoiceSetSHP*:
- *KML2SHP*:
- *OBS2SHP*:
- *AnalyzeOBS*:

There are many functionalities (commands) implemented. Each function uses a set of parameters that have to be specified in the xml file. The following introduces how to use these commands.

You first need to set up the global experiment environment, including the experiment folder, the output.

Sampling

The main usage of bioroute is to perform sampling of path alternatives for route choice models. The flow chart or this software is depicted in the following graph. There are two types of inputs. One is the transportation network (see network parameters for parameter specifications), another one is the the chosen alternatives or OD.



Path sampling works for a pair of OD. In route choice context, there are path observations generated from probabilistic map matching. The OD can be extracted from those path observations. So there are two types of OD input, pmm results (see path input) or OD (see OD input) can be used.

- If path input is used, the command for sampling is `*Sample*`.
- If OD input is used, the command for sampling is `*SampleWithOD*`

There are tree sampling algorithms implemented: uniform, random walk and metropolis-hastings. Simply set *pathSampleAlgorithm* to 'EQ', 'RW', or 'MH'.

- Uniform is the simplest algorithm. The implementation uses MH algorithm where the sampling function has the constant value 1. See Uniform.
- The random walk algorithm is developed by Frejinger [Frejinger2008]. There are two parameters (kuma A and kuma B) to be specified. See Random Walk.
- MetroPolis-Hastings path sampling algorithm is proposed by Flötteröd [Flötteröd2013], and adapted for route choice modeling by Chen [CHEN2013]. See MH sampling

Sampling alternatives for route choice observations

The most important feature of *bioroute* is to sample alternatives for GPS map matching results generated by *pmm*

```
bioroute -c Sample -f PARAM_FILE
```

Sampling alternatives with the given OD

Sampling alternatives with equal probability

Simulate route choice observations

A case study

This case study implements the real data experiment described in Jingmin Chen's thesis. The map matching results of 19 trips are given. Decompress the content to a folder, and we note the path to this directory as **EXPERIMENT_HOME** The configuration file *config_gps_route_choice.xml* is in the param folder.

```
<param name="experimentDirectory" value="EXPERIMENT_HOME/" type=string"/>
```

Note: always put a / after a directory!

Input

The unimodal map matching is applied to generate path observations from GPS data. The gps data is in *gps* folder. * Map matching results with *kml* format are in *path_observations* folder.

```
<param name="observationDirectory" value="path_observations/"string"/>
```

We would like to use MH path sampling to generate biogeme inputs. Each set of sampled alternatives is composed of 100 draws. In order to test the effect of the sampling size, different sets of alterantives are produced: 30, 50, 100. There will be three inputs generated. One contains 30 draws in the sampled alterantives; one contains 50, and one contains 100.

```
<param name="SAMPLE_COUNT" value="100" type="int"/>
<param name="choiceSetInBiogemeData" value="30,50,100," type="string"/>
```

Note `,` after *100*. It is a required syntax.

- Set the *burn-in* and *sampling interval* for MH path sampling

```
<param name="WARMUP_ITERATIONS" value="500000" type="int"/>
<param name="SAMPLEINTERVAL_ELEMENT" value="100000" type="int"/>
```

- Set ζ and ω_2 parameters for the sampling

```
<param name="mh_link_scale"  value="1.026" type="float"/>
<param name="mh_obs_scale"  value="1.0" type="float"/>
```

- Set the sampling results folder. The choice set for each observation, along with the biogeme dat file and mod file will be produced in this folder.

```
<param name="choiceSetFolder" value="1.026_1.0/" type="string"/>
```

- Set the network file

```
<param name="OsmNetworkFileName" value="EXEPRIMENT_HOME/lausanne.osm" type="string"/>
```

Run MH sampling

```
EXPERIMENT_HOME$ /data/jingmin/bin/bioroute -c Sample -f params/config_gps_route_choice.xml
```

Output

- For each candidate path, a set of alterantives are generated. For an observation with file name *3 -6623*, there are 7 candidate paths, noted from *3 -6623_1* to *3 -6623_7*. For path 1 *3 -6623_1_sample.kmlreduced.kml.kml* and *3 -6623_1_sample.kmlreduced.kml.shp* are the reduced network (described in MH paper) in *kml* and *shp* format. *3 -6623_1_sample.kml* is the sampling result.
- Three biogeme data files are produced, with 30, 50, 100 draws respectively. The files are *observations_30.dat*, *observations_50.dat* and *observations_100.dat*
-

Parameters

Experiment Environment

The experiment folder

First of all, all files including GPS, path observations, configurations, results have to be at the same location. The directory is denoted as *EXPERIMENT_HOME* in this tutorial. It has to be set in *experimentDirectory* parameter.

The generated choice set

You should specify where the results should be produced. It is a folder under the *experimentDirectory*, it can be multiple level. The value is relative to *experimentDirectory*, for example:

```
<param name="choiceSetFolder" value="1018/r1.026_1.0/" type="string"/>
```

It is better that you create this folder before running the experiment.

The number of concurrent threads can be set:

```
<param name="nbrOfThreads" value="25" type="int"/>
```

But if there is an unexpected / unknown error, reduce this value.

Input & Output

network parameters

The network input can have two formats. One is from database (*databaseHost* parameter), another is the original osm source file (*OsmNetworkFileName* parameter). The software always tries to connect to the database if *databaseHost* is not empty. If the connection failed, the software stops functioning. So, it is preferable to use OSM file.

- If you want to **use OSM file**, the connection to the database should be disabled by setting *databaseHost* parameter as empty.

```
<param name="databaseHost" value="" type="string"/>
```

Then the (absolute) path to the osm file should be given for *OsmNetworkFileName*, for example:

```xml
<param name="OsmNetworkFileName" value="/users/jchen/newbioroute/playground/network_data/lausanne.osm" type="string"/>
```

- If you want to **use database**, the connection should be instructed for *databaseHost* parameter with host, port, user name, password, and the name of the database:

```xml
<param name="databaseHost" value="host=transporsrv4.epfl.ch port=5432 user=jchen password=xxx  dbname=osm_jchen" type="string"/>
```

Since we are only dealing with car network. All the other networks should be disabled by setting following parameters

```xml
<param name="enableCarNetwork" value="1" type="int"/>
<param name="enableBusNetwork" value="0" type="int"/>
<param name="enableMetroNetwork" value="0" type="int"/>
<param name="enableTrainNetwork" value="0" type="int"/>
<param name="enableWalkNetwork" value="0" type="int"/>
<param name="enableBikeNetwork" value="0" type="int"/>
```

path input

The path observations are generated from pmm as *kml* files in a folder, under *EXPERIMENT_HOME*. The folder should be set at *observationDirectory*, which is relative to *EXPERIMENT_HOME*, e.g.

```xml
<param name="observationDirectory" value="path_observations/" type="string"/>
```

Then the software will be able to recognize that the observations are in *EXPERIMENT_HOME/path_observations/*. Noted that / is required at the end of the value.

The absolute path to the corresponding gps files should be set in *dataDirectory* .. code:: xml

<param name="observationDirectory" value="EXPERIMENT_HOME/gps/" type="string"/>

OD input

The origin and destination have to be set with the OD of the corresponding nodes in OSM network. For example

```xml
<param name="OriginId" value="296253984" type="int"/>
<param name="DestinationId" value="973818127" type="int"/>
```

Sampling algorithm

The number of samples

```xml
<!-- total number of samples to be generated-->
        <param name="SAMPLE_COUNT" value="100" type="int"/>

        <!-- For biogeme data, different number of alternatives can be specified -->
        <!-- It is used to test the effect of different number of alterantives -->
        <!-- Syntax is `NUMBER,`, for example, 30,50,100, -->
<param name="choiceSetInBiogemeData" value="30,50,100," type="string"/>
```

Random Walk

Random walk mainly has to par

```
<param name="kumaA" value="30.0" type="float"/>
<param name="kumaB" value="1.0" type="float"/>
```

MH sampling

The sampling weight for the MH path sampling is defined as. .. math:

```
b(i)=\frac{\ln2}{mh\_length\_coef (\zeta-1) L_{j_{sp}}}(mh\_length\_coef * L + mh\_sb\_coef * NbrTrafficSingals) + mh\_ps\_coef * PS + mh\_obs\_coef * \lambda * P
```

The variables:

- $L_{j_{sp}}$ The length of the shortest path.
- LThe length of the path.
- $NbrTrafficSingals$Number of traffic signals in the path
- PSPath size of the path
- PThe observation score ($\lambda$is automatically calibrated)

The parameters

- $mh_length_coef$and mh_sb_coefcoefficients for link-additive path attributes.
- ζparameter for link cost scale (1.026 in the thesis)
- $mh_ps_coef$the coefficient for path size.
- $mh_obs_coef$is in fact ω_2 in the thesis

In MH sampling, there are a lot of cases where **shortest path** is performed. Shortest path only takes link-additive attributes. The function for the shortest path calculation is

mh_router_cost_link_scale * (mh_length_coef * Length + mh_sb_coef * NbrOffrafficSginals)

The default values are

- *mh_router_cost_link_scale*: -1.0
- *mh_length_coef*: -0.01
- *mh_sb_coef*: 0.0

The **insertion probability of a node** is defined according to the sampling weight of the path comprising of the shortest path from the origin to the node, plus the shortest path from the node to the destination

Uniform

Uniform is a special case of MH sampling. Simply set *pathSampleAlgorithm* to *"EQ"*

```
<param name="pathSampleAlgorithm" value="EQ" type="string"/>
```

Output

Overwrite old sampled files

There are cases that the sampling is stopped for some reason. A part of the results have been produced. But you have to run the software again in order to get all results. In this case, if a result already exists (as kml result in *choiceSetFolder*), you can ask the software not to perform the sampling again by setting *overwriteSampleFile* to 0.

- If *overwriteSampleFile* is "1", the software redo the sampling and overwrite the result.
- If *overwriteSampleFile* is "0", the software skip the corresponding task.

Frejinger2008	Frejinger, E., Bierlaire, M., and Ben-Akiva, M. (2009). Sampling of Alternatives for Route Choice Modeling, Transportation Research Part B: Methodological 43(10
Flötteröd2013	Flötteröd, G., and Bierlaire, M. (2013). Metropolis-Hastings sampling of paths, Transportation Research Part B: Methodological 48
CHEN2013	Chen, J. (2013). Modeling route choice behavior using smartphone data, Ecole Polytechnique Féedéerale de Lausanne, Switzerland