transport Documentation

Release v2.0.0-alpha-3

Milan Lovric

Jan 04, 2022

Contents

NISMOD v2 Transport Model is a national-scale (*Great Britain*) transport model developed to support policy making regarding future infrastructure.

CHAPTER 1

Contents

1.1 Documentation

NISMOD v2 Transport Model is a national-scale (*Great Britain*) transport model developed to support policy making regarding future infrastructure. It forecasts the impact of various endogenous and exogenous factors on transport demand and capacity utilisation, following an elasticity-based simulation methodology. The model consists of three submodels covering the following modes of transport: **road** (passenger and freight vehicle flows), **rail** (total station usage), and **air** (domestic and international passenger movements).

1.1.1 1. Key Features

1.1 Road Model

NISMOD v2 Transport Model predicts vehicle demand (inter-zonal flows) for passenger and freight vehicles, and stochastically simulates road traffic on all major UK roads including A-roads and motorways.

It is currently the only national-scale road traffic model capable of routing-based network assignment and provisioning a national-scale origin-destination matrix (on TEMPRo & LAD spatial zoning levels), while achieving a respectable match with AADF traffic counts, total vehicle kilometres, expected number of car trips, and the observed trip length distribution from the National Travel Survey.

The freight model has been modelled after the DfT's 2006 Base-Year Freight Matrices model, which includes traffic flows for freight vehicles (vans, rigid HGVs, and articulated HGVs) between local authority districts (LADs), sea ports, selected airports, and major distribution centres. The accuracy of the freight model is mostly limited by the spatial zoning system (LAD).

The demand prediction model is elasticity-based and it can predict future vehicle flows from exogenous (scenario-based) changes in population and GVA, and endogenously calculated changes in inter-zonal travel time and travel cost (but also dependent on exogenous interventions such as new road development and congestion charging policies).

Congested travel times on individual road links have been modelled separately for each hour of the day, using the speed-flow curves estimated on English roads (DfT's 2005 FORGE model), the overcapacity formula from WebTAG, and the passenger car unit (PCU) concept to capture different vehicle sizes.

The number of lanes on each road segment has been estimated by map-matching AADF count point locations to the OpenRoads major road network. This has allowed a distinction between single and dual carriageway A-roads, which are then assumed to have 1 and 2 lanes per direction, respectively.

The network assignment exists in two version and it has been implemented using state-of-the-art routing algorithms. The routing version uses a heuristic search algorithm A* to find the fastest path between two locations using congested link travel times, while the route-choice version uses an advanced discrete-choice model (path-size logit) to choose the optimal path based on distance, travel time, travel cost (fuel and road tolls), and the number of intersections.

The route-choice version of the network assignment uses a route set pre-generated on the IRIDIS cluster of the *University of Southampton*. This pre-generated route set consists of more than 90 million different route options which enables the national-scale assignment to run within minutes, despite each individual vehicle trip being simulated separately (including time of day choice, engine type choice, route choice).

The model can also incorporate scenarios for changes in vehicle fuel efficiency and changes in market shares of different engine types, including internal combustion engines on petrol, diesel, LPG (liquefied petroleum gas), hydrogen or CNG (compressed natural gas); hybrid EVs (electric vehicles) on petrol or diesel; plug-in hybrid EVs on petrol or diesel; fuel cell EVs on hydrogen, and battery EV. This can be used to test policies such as the fossil fuel phase-out.

Electricity and fuel consumptions are calculated using the four-parameter formula from WebTAG. Behavioural assumptions are made for plug-in hybrid EVs (electricity on urban, fuel on rural road links).

Interventions such as new road development, road expansion with new lanes, and congestion charging zones can be dynamically implemented in each simulated year.

The model can output various metrics on the road link level (e.g. road capacity utilisation, peak hour travel times), zonal level (e.g. vehicle kilometres, EV electricity consumption), inter-zonal level (e.g. predicted vehicle flows, average travel times, average travel costs) and national level (e.g. total CO2 emissions, total energy consumptions). The outputs are in csv and shapefile format, allowing them to be visualised with a software of choice.

The units for energy consumptions are:

- PETROL litres (l)
- DIESEL litres (l)
- DIESEL litres (l)
- LPG kilograms (kg)
- ELECTRICITY kilowatt-hours (kWh)
- HYDROGEN kilograms (kg)
- CNG kilograms (kg)

The units for other outputs are:

- cost skim matrices: pounds GBP (£)
- time skim matrices and link travel times: minutes (min)
- CO2 emissions: kilgrams (kg)
- vehicle-kilometres: vehicle-kilometres (vkm)
- OD matrices: vehicles per day (v/d)

Note that outputs are produced for only one (average) simulated day, so yearly values can be obtained by multiplying those values by 365 - in which case it may become more appropriate to use kilotonnes or megatonnes instead of kilograms, MWh or GWh instead of kWh etc.

1.2 Rail Model

NISMOD v2 Transport Model includes a national-scale rail model for predicting future station usage demand.

It currently uses station usage data for 3054 stations covering National Rail, London Underground, Docklands Light Railway, London Trams (previously Croydon Tramlink), Manchester Metrolink, and Tyne & Wear (Newcastle) Metro.

Elasticity-based demand model predicts station usage (entry + exit) from changes in exogenous inputs including: population, GVA, rail fare index, generalised journey time (GJT) index and car trip costs.

Car trip costs can be provided as an input or calculated from the outputs of the NISMOD road model.

Elasticities of rail fares and GJT vary per elasticity zone (London Travelcard, South-East, PTE, other).

The model implements a policy intervention for building new rail stations in future years.

1.2 Air Model

NISMOD v2 Transport Model also includes an air model that predicts domestic and international passenger movements.

Air demand data is inter-nodal, i.e. between individual airports (domestic - between two UK airports, and international - between a UK airport and an international airport).

Base-year (2015) demand data is obtain from the Civil Aviation Authority (CAA), while information about airports is obtained from CAA, NaPTAN, and ourairports.com.

Demand files use IATA codes (or ICAO where IATA is unavailable) to identify airports and ISO 3166 Alpha-2 codes to identify countries.

Elasticity-based demand model predicts passenger movements from changes in exogenous inputs including: population, GVA, domestic and international fare incides, and trip rates.

1.1.2 2. How to run the model

2.1 Using Eclipse IDE

Install *Java Development Kit* version 8 from: http://www.oracle.com.

Install *Eclipse IDE for Java Developers*: https://eclipse.org/downloads/.

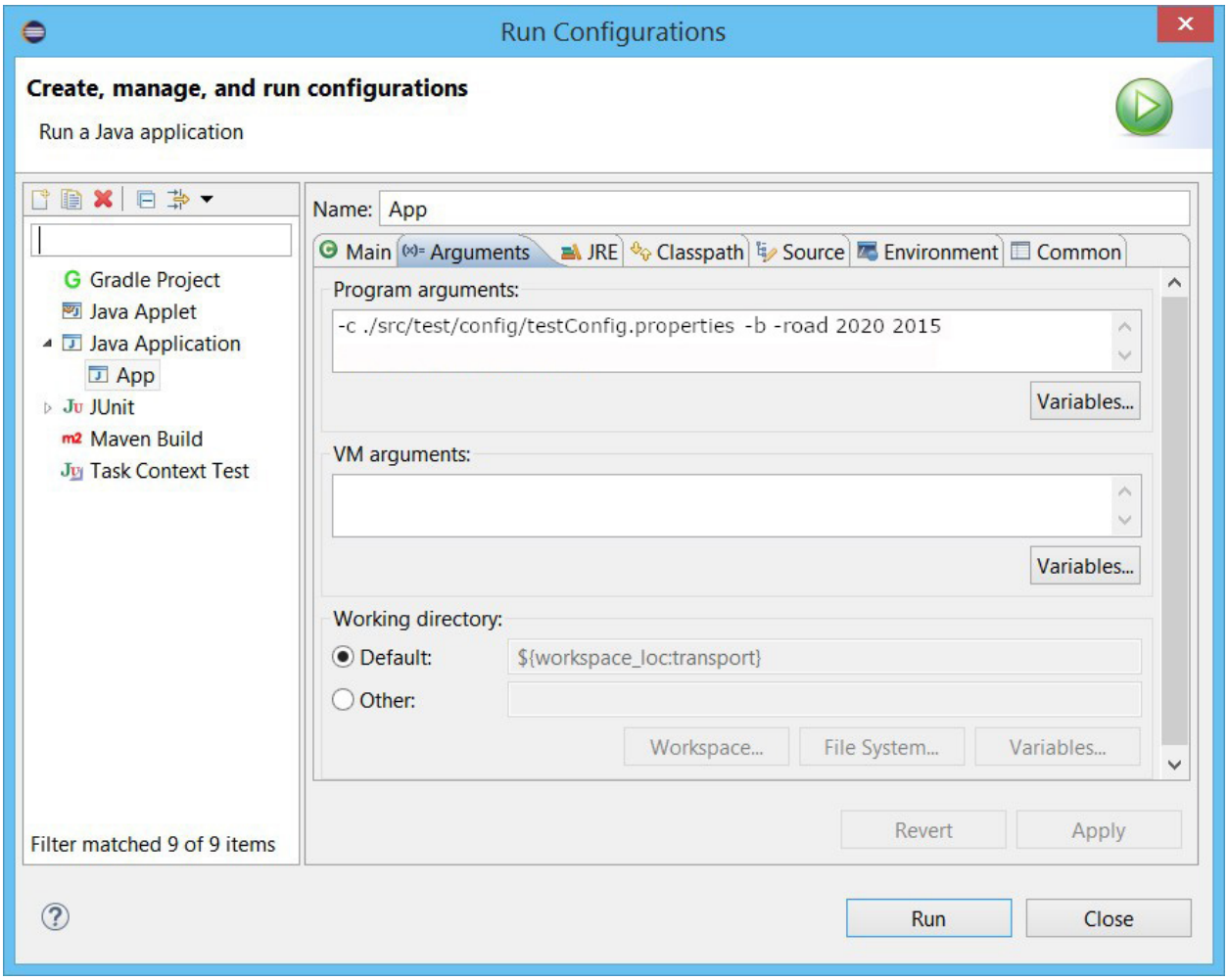
Run Eclipse and choose the workspace folder.

Import the existing Maven project from the local git folder where the code has been cloned. In Eclipse: *File -> Import -> Maven -> Existing Maven Projects*.

Wait until all Maven dependencies (specified in the *pom.xml* file) are downloaded. If the *pom.xml* file has been changed, the Maven project should be first updated (*Alt+F5*).

The classes containing the *main* method can be run as a Java application. The classes containing the methods annotated with *@Test* can be run as *JUnit* tests.

To run the main model in Eclipse, open the *Run Configuration* for *nismod.transport.App.java* and pass the path to the config file as an argument:



2.2 Using Command Prompt (Console)

Install *Java Development Kit* version 8 from: http://www.oracle.com.

Make sure the Java home environment variable is set for the operating system and pointing to the directory where *Java Development Kit* has been installed.

Download maven, install it and set the environment variables: http://maven.apache.org/.

To build the project type:

```
mvn clean install
```

To run the base-year **road** model (2015) type:

```
java -cp target/transport-0.0.1-SNAPSHOT.jar nismod.transport.App -c ./path/to/config.
↪properties -b
```

To predict and run a future year (e.g. 2020) using the results of a previously run year (e.g. 2015), for the **road** model type:

```
java -cp target/transport-0.0.1-SNAPSHOT.jar nismod.transport.App -c ./path/to/config.
↪properties -road 2020 2015
```

To predict and run a future year (e.g. 2020) using the results of a previously run year (e.g. 2015), for the **rail** model type:

```
java -cp target/transport-0.0.1-SNAPSHOT.jar nismod.transport.App -c ./path/to/config.
↪properties -rail 2020 2015
```

To predict and run a future year (e.g. 2020) using the results of a previously run year (e.g. 2015), for the **air** model type:

```
java -cp target/transport-0.0.1-SNAPSHOT.jar nismod.transport.App -c ./path/to/config.
↪properties -air 2020 2015
```

Note: Rail and air model need not be run for the base-year as 2015 demand data is given as an input.

Options:

- To increase the max heap size, run with *java -XX:MaxHeapSize=120g ...*
- To enable debug messages, run with *java -Dlog4j2.debug ...*

2.3 Showcase Demo

The model provides an interactive showcase demo with three policy interventions in the case study area of South-East England. The interventions are:

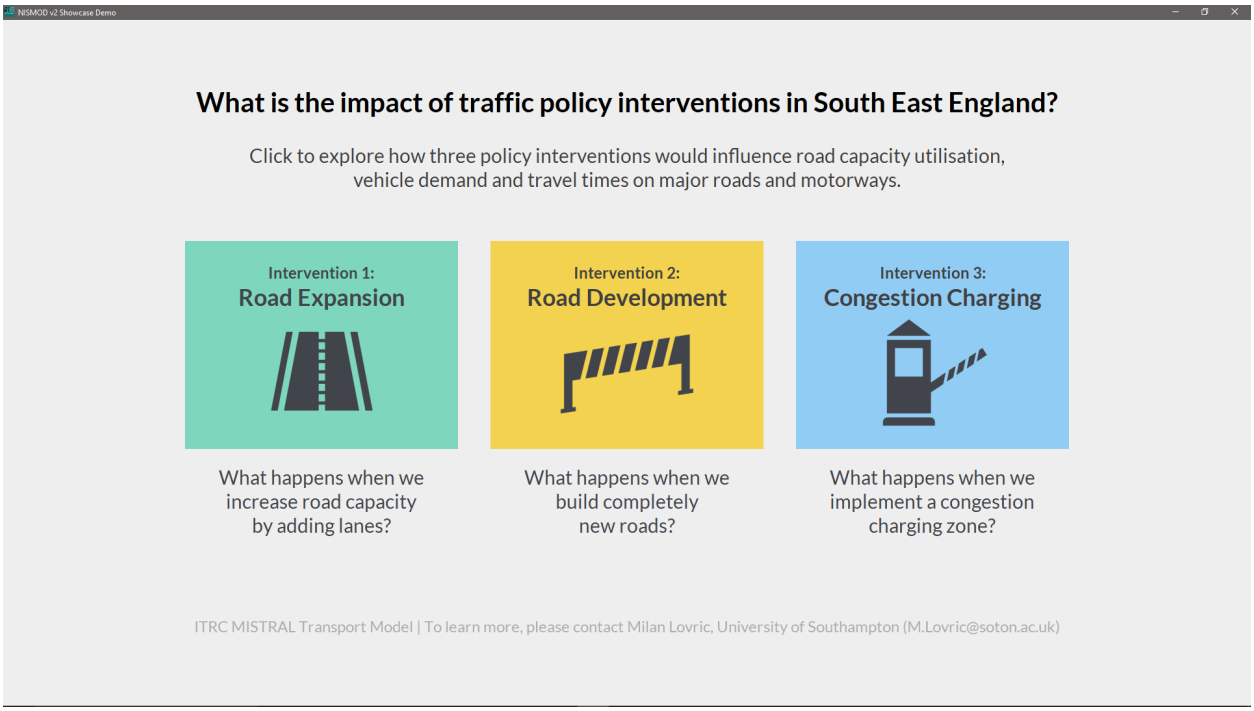
Road expansion - expanding existing road links with additional lanes.

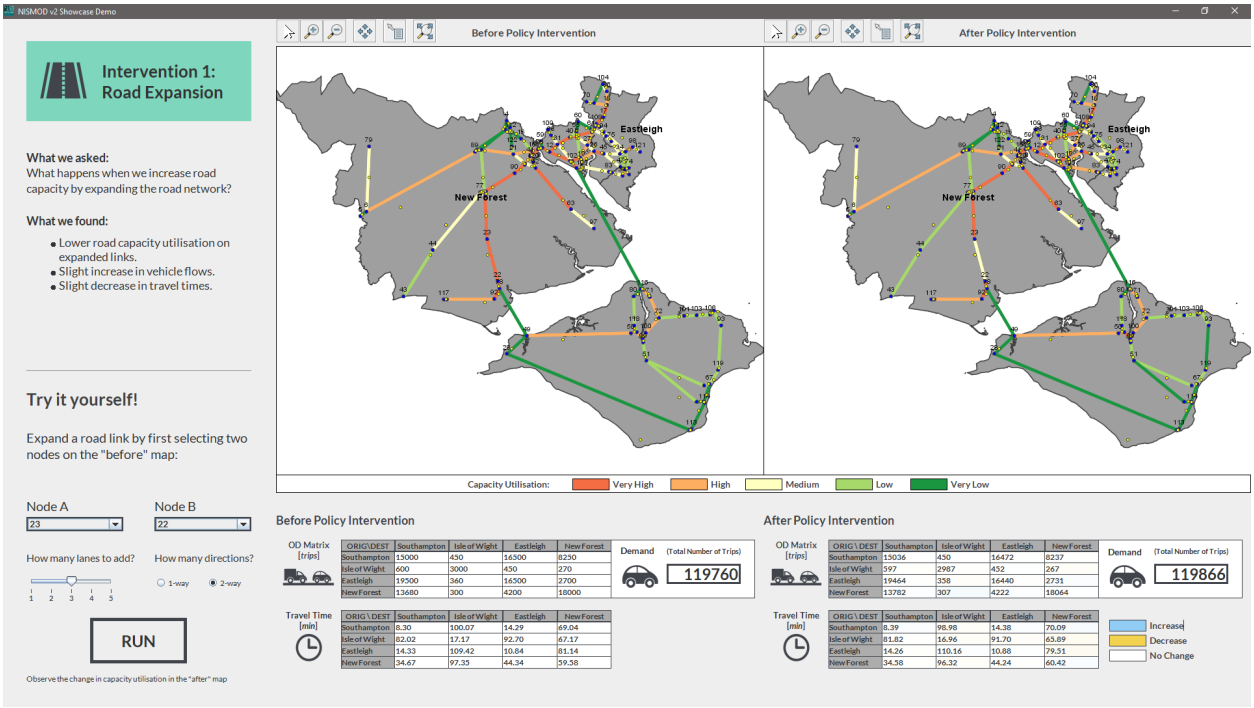
Road development - building new road links between two existing intersections.

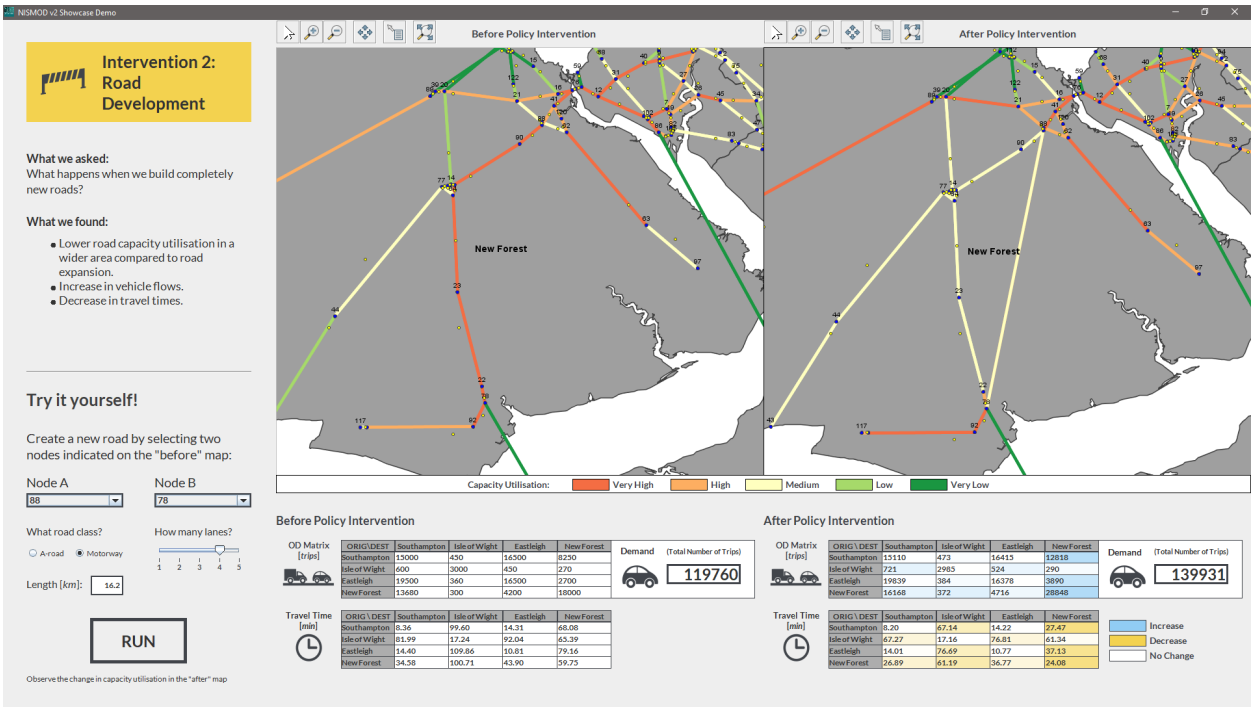
Congestion charging - time-based (peak and off-peak) congestion charging in the policy area.

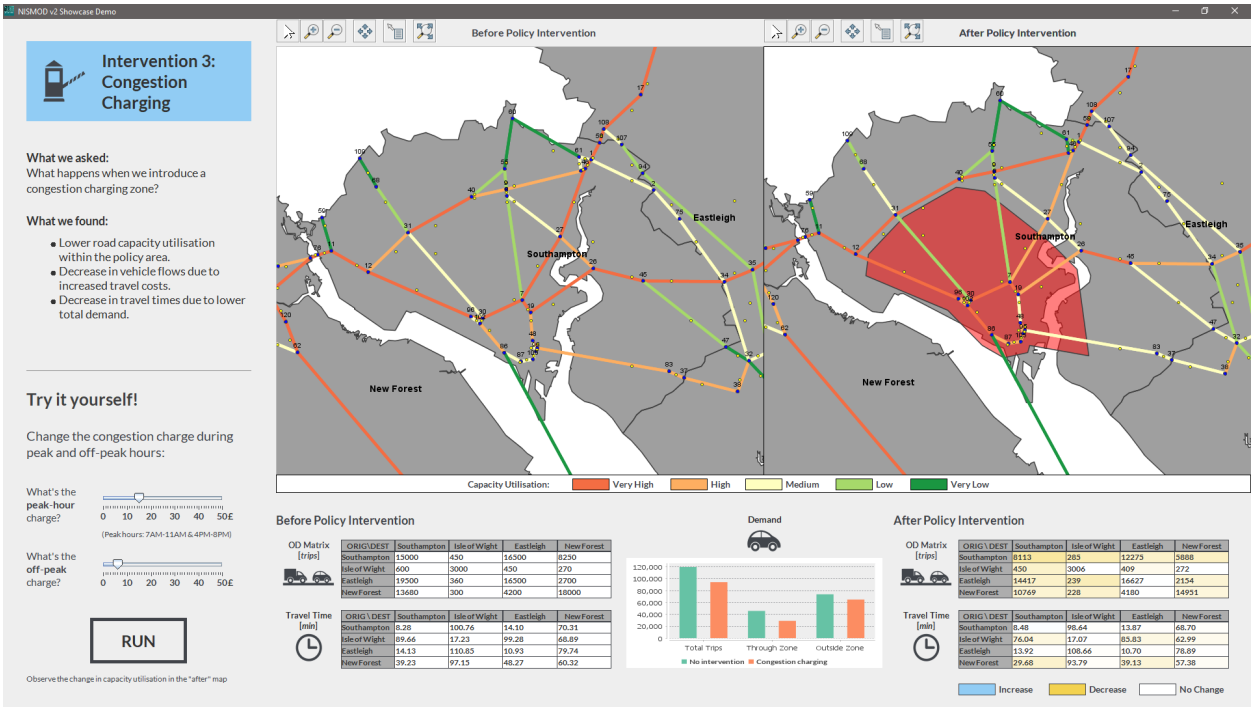
To run the showcase demo type:

```
java -cp target/transport-0.0.1-SNAPSHOT.jar nismod.transport.App -c ./path/to/config.
↪properties -d
```


NISMOD v2 Showcase Demo
What is the impact of traffic policy interventions in South East England?
Click to explore how three policy interventions would influence road capacity utilisation, vehicle demand and travel times on major roads and motorways.
Intervention 1:
Road Expansion
Intervention 2:
Road Development
Intervention 3:
Congestion Charging
What happens when we increase road capacity by adding lanes?
What happens when we build completely new roads?
What happens when we implement a congestion charging zone?
ITRC MISTRAL Transport Model | To learn more, please contact Milan Lovric, University of Southampton (M.Lovric@soton.ac.uk)


NISMOD v2 Showcase Demo
Intervention 1:
Road Expansion
What we asked:
What happens when we increase road capacity by expanding the road network?
What we found:
Lower road capacity utilisation on expanded links.
Slight increase in vehicle flows.
Slight decrease in travel times.
Try it yourself!
Expand a road link by first selecting two nodes on the "before" map:
Node A
23
Node B
22
How many lanes to add?
How many directions?
1-way
2-way
RUN
Observe the change in capacity utilisation in the "after" map
Before Policy Intervention
After Policy Intervention
Eastleigh
New Forest
Capacity Utilisation:
Very High
High
Medium
Low
Very Low
OD Matrix [trips]
Demand (Total Number of Trips)
119760
119866
Travel Time [min]
Increase
Decrease
No Change





Note: Showcase demo requires a display with a 1920 x 1080 resolution.

1.1.3 3. Cross-sectoral Dependencies

The UK transport sector has various links with other infrastructure sectors:

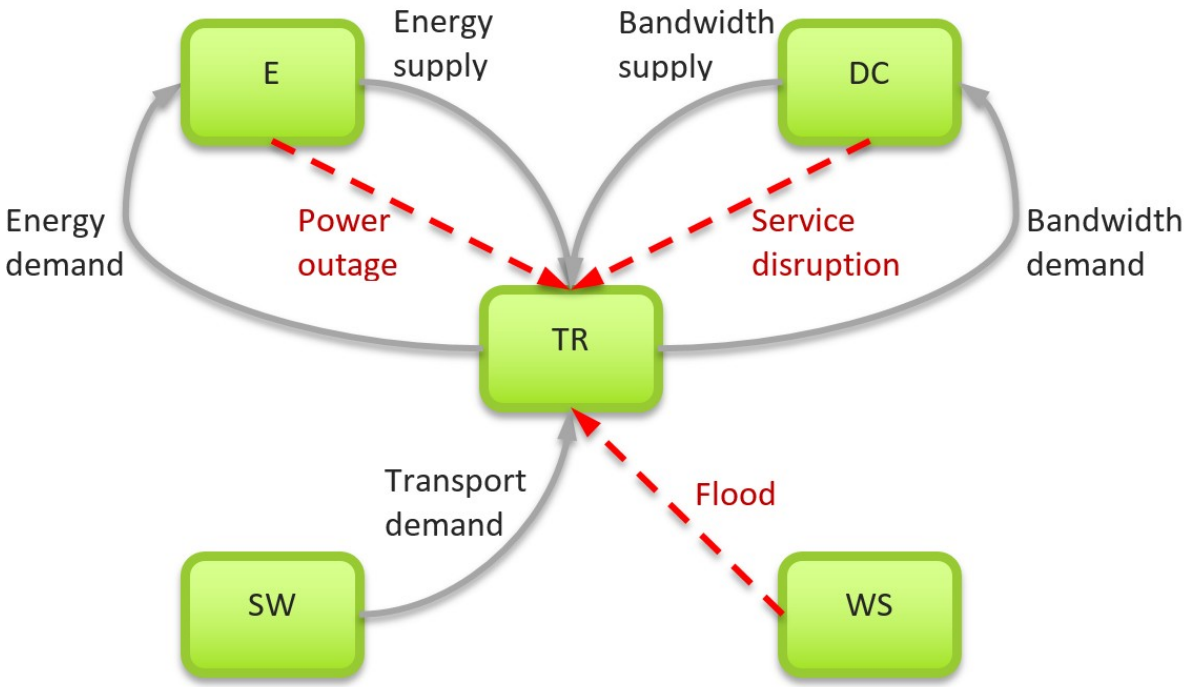
Energy: energy consumption, fuel price, electrification of vehicles, fuel transport, power outage (rail and air disruption).

Digital Communications: supporting smart mobility (e.g. mobility as a service, autonomous mobility on demand), coverage and service disruptions.

Water: floods causing road and rail disruptions.

Solid Waste: waste transport (e.g. waste exports through seaports).

To enable studies of some of those cross-sectoral interdependencies, the transport model has been integrated into a wider *Simulation Modelling Integration Framework (smif)*: https://github.com/nismod/smif



1.1.4 4. Acknowledgments

This work has been undertaken at the *University of Southampton*, as part of the ITRC consortium, under grant EP/N017064/1 (MISTRAL: Multi-scale InfraSTRucture systems AnaLytics) of the UK *Engineering and Physical Science Research Council* (EPSRC). https://www.itrc.org.uk/

The test resources contain a sample of data and shapefiles that come with the following licencing and copyright statemens:

- *Open Government Licence* http://www.nationalarchives.gov.uk/doc/open-government-licence/version/3/
- Contains *National Statistics* data © *Crown* copyright and database right 2012.
- Contains *Ordnance Survey* data © *Crown* copyright and database right 2012.

The authors acknowledge the use of the IRIDIS *High Performance Computing Facility*, and associated support services at the *University of Southampton*, in the completion of this work.

The implementation uses an open source library *GeoTools* for geospatial processing. http://www.geotools.org/about.html

1.2 'transport'

1.2.1 nismod.transport

App

public class **App**
NISMOD V2.0.0 Transport Model.

Author Milan Lovric

Methods

main

public static void **main** (String[] *args*)

1.2.2 nismod.transport.air

AirDemandModel

public class **AirDemandModel**
Main air demand prediction model (elasticity-based).

Author Milan Lovric

Fields

baseYear

public static int **baseYear**

domesticAirports

public static Map<String, *Airport*> **domesticAirports**

internationalAirports

public static Map<String, *Airport*> **internationalAirports**

Constructors

AirDemandModel

public **AirDemandModel**(String *domesticAirportsFile*, String *internationalAirportsFile*, String *baseYearDomesticPassengerFile*, String *baseYearInternationalPassengerFile*, String *populationFile*, String *GVAFile*, String *elasticitiesFile*, String *domesticAirportFareIndexFile*, String *internationalAirportFareIndexFile*, String *domesticTripRatesFile*, String *internationalTripRatesFile*, List<*Intervention*> *interventions*, Properties *props*)

Constructor for the air demand model.

Parameters

- **domesticAirportsFile** – List of domestic airports.
- **internationalAirportsFile** – List of international airports.
- **baseYearDomesticPassengerFile** – Base year domestic air passenger file (demand).
- **baseYearInternationalPassengerFile** – Base year international air passenger file (demand).
- **populationFile** – Population file.
- **GVAFile** – GVA file.
- **elasticitiesFile** – Elasticities file.
- **domesticAirportFareIndexFile** – Domestic airport fare index.
- **internationalAirportFareIndexFile** – International airport fare index.
- **domesticTripRatesFile** – Domestic trip rates file.
- **internationalTripRatesFile** – International trip rates file.
- **interventions** – List of interventions.
- **props** – Properties.

Throws

- **IOException** –
- **FileNotFoundException** –

Methods

getDomesticAirPassengerDemand

public *InternodalPassengerDemand* **getDomesticAirPassengerDemand**(int *year*)

Getter method for the air passenger demand in a given year.

Parameters

- **year** – Year for which the demand is requested.

Returns Air passenger demand.

getInternationalAirPassengerDemand

public *InternodalPassengerDemand* **getInternationalAirPassengerDemand**(int *year*)

Getter method for the air passenger demand in a given year.

Parameters

- **year** – Year for which the demand is requested.

Returns Air passenger demand.

predictAndSaveAirDemands

public void **predictAndSaveAirDemands**(int *toYear*, int *fromYear*)

Predicts air passenger demands (domestic and international) up to toYear (if flag is true, also intermediate years) and saves results.

Parameters

- **toYear** – The final year for which the demand is predicted.
- **fromYear** – The year from which the predictions are made.

predictDomesticAirDemandUsingResultsOfFromYear

public void **predictDomesticAirDemandUsingResultsOfFromYear**(int *predictedYear*, int *fromYear*)

Predicts domestic air passenger internodal demand. Uses already existing results of the fromYear, from the output folder.

Parameters

- **predictedYear** – The year for which the demand is predicted.
- **fromYear** – The year from which demand the prediction is made.

predictInternationalAirDemandUsingResultsOfFromYear

public void **predictInternationalAirDemandUsingResultsOfFromYear**(int *predictedYear*, int *fromYear*)

Predicts international air passenger internodal demand. Uses already existing results of the fromYear, from the output folder.

Parameters

- **predictedYear** – The year for which the demand is predicted.
- **fromYear** – The year from which demand the prediction is made.

saveAllResults

public void **saveAllResults**(int *year*)

Saves all results into the output folder.

Parameters

- **year** – Year of the data.

saveDomesticAirDemand

public void **saveDomesticAirDemand**(int *year*, String *outputFile*)
Saves domestic air demand to an output file.

Parameters

- **year** – Year of the data.
- **outputFile** – Output file name (with path).

saveInternationalAirDemand

public void **saveInternationalAirDemand**(int *year*, String *outputFile*)
Saves international air demand to an output file.

Parameters

- **year** – Year of the data.
- **outputFile** – Output file name (with path).

AirDemandModel.ElasticityTypes

public static enum **ElasticityTypes**

Enum Constants

COST_DOMESTIC

public static final *AirDemandModel.ElasticityTypes* **COST_DOMESTIC**

COST_INTERNATIONAL

public static final *AirDemandModel.ElasticityTypes* **COST_INTERNATIONAL**

GVA

public static final *AirDemandModel.ElasticityTypes* **GVA**

POPULATION

public static final *AirDemandModel.ElasticityTypes* **POPULATION**

Airport

public abstract class **Airport**
This class stores information about an airport.

Author Milan Lovric

Constructors

Airport

public **Airport** (String *iataCode*, String *caaName*, String *ourAirportsName*, double *longitude*, double *latitude*, String *countryCode*, String *continentCode*, long *terminalCapacity*, long *runwayCapacity*)

Constructor for the airport.

Parameters

- **iataCode** – Airport IATA code.
- **caaName** – Airport name in CAA datasets.
- **ourAirportsName** – Airport name in ourAirports dataset.
- **longitude** – Longitude coordinate.
- **latitude** – Latitude coordinate.
- **countryCode** – Code of the country in which the airport is located.
- **continentCode** – Code of the continent in which the airport is located.
- **terminalCapacity** – Airport terminal capacity (max number of passengers that can be processed).
- **runwayCapacity** – Airport runway capacity (max number of flights that can be processed).

Airport

public **Airport** (*Airport airport*)

Constructor for an airport.

Parameters

- **airport** – Airport which data is going to be copied.

Methods

getCAAName

public String **getCAAName** ()

Getter method for the airport CAA name.

Returns CAA name.

getContinent

public *ContinentCode* **getContinent** ()

Getter method for the continent in which the airport is located.

Returns ContinentCode continent.

getCountry

public Locale **getCountry**()
Getter method for the country in which the airport is located.

Returns Country locale.

getIataCode

public String **getIataCode**()
Getter method for the IATA code of the station.

Returns IATA code.

getLatitude

public double **getLatitude**()
Getter method for Latitude.

Returns Latitude.

getLongitude

public double **getLongitude**()
Getter method for longitude.

Returns Longitude.

getOurAirportsName

public String **getOurAirportsName**()
Getter method for the ourAirports name.

Returns OurAirports name.

getRunwayCapacity

public long **getRunwayCapacity**()
Getter method for the airport runway capacity.

Returns Runway capacity.

getTerminalCapacity

public long **getTerminalCapacity**()
Getter method for the airport terminal capacity.

Returns Terminal capacity.

toString

public String **toString**()

Airport.AirportGroup

public static enum **AirportGroup**
Airports grouped by DfT.

Enum Constants

DO

public static final *Airport.AirportGroup* **DO**

LH

public static final *Airport.AirportGroup* **LH**

SH

public static final *Airport.AirportGroup* **SH**

Airport.AirportGroupCAA

public static enum **AirportGroupCAA**
Airport groups by CAA used in the flight movements dataset.

Enum Constants

EU

public static final *Airport.AirportGroupCAA* **EU**

INT

public static final *Airport.AirportGroupCAA* **INT**

UK

public static final *Airport.AirportGroupCAA* **UK**

Airport.ContinentCode

public static enum **ContinentCode**
ISO continent code

Enum Constants

AF

public static final *Airport*.*ContinentCode* **AF**

AN

public static final *Airport*.*ContinentCode* **AN**

AS

public static final *Airport*.*ContinentCode* **AS**

EU

public static final *Airport*.*ContinentCode* **EU**

NA

public static final *Airport*.*ContinentCode* **NA**

OC

public static final *Airport*.*ContinentCode* **OC**

SA

public static final *Airport*.*ContinentCode* **SA**

Methods

getName

public String **getName**()

Airport.ForeignRegionCAA

public static enum **ForeignRegionCAA**

These are the foreign region groups used by CAA for international internodal passenger demand. There is 1:1 mapping between a country and a region (this is unlike the OurAirports data where one country could map to multiple regions, e.g. some Russian airports are in Asia, while some are in Europe).

Enum Constants

ATLANTIC_OCEAN_ISLANDS

public static final *Airport.ForeignRegionCAA* **ATLANTIC_OCEAN_ISLANDS**

AUSTRALASIA

public static final *Airport.ForeignRegionCAA* **AUSTRALASIA**

CANADA

public static final *Airport.ForeignRegionCAA* **CANADA**

CARRIBEAN_AREA

public static final *Airport.ForeignRegionCAA* **CARRIBEAN_AREA**

CENTRAL_AFRICA

public static final *Airport.ForeignRegionCAA* **CENTRAL_AFRICA**

CENTRAL_AMERICA

public static final *Airport.ForeignRegionCAA* **CENTRAL_AMERICA**

EASTERN_EUROPE_EU

public static final *Airport.ForeignRegionCAA* **EASTERN_EUROPE_EU**

EASTERN_EUROPE_OTHER

public static final *Airport.ForeignRegionCAA* **EASTERN_EUROPE_OTHER**

EAST_AFRICA

public static final *Airport.ForeignRegionCAA* **EAST_AFRICA**

FAR_EAST

public static final *Airport.ForeignRegionCAA* **FAR_EAST**

INDIAN_OCEAN_ISLANDS

public static final *Airport.ForeignRegionCAA* **INDIAN_OCEAN_ISLANDS**

INDIAN_SUB_CONTINENT

public static final *Airport.ForeignRegionCAA* **INDIAN_SUB_CONTINENT**

MIDDLE_EAST

public static final *Airport.ForeignRegionCAA* **MIDDLE_EAST**

NEAR_EAST

public static final *Airport.ForeignRegionCAA* **NEAR_EAST**

NORTH_AFRICA

public static final *Airport.ForeignRegionCAA* **NORTH_AFRICA**

OIL_RIGS

public static final *Airport.ForeignRegionCAA* **OIL_RIGS**

PACIFIC_OCEAN_ISLANDS

public static final *Airport.ForeignRegionCAA* **PACIFIC_OCEAN_ISLANDS**

SOUTHERN_AFRICA

public static final *Airport.ForeignRegionCAA* **SOUTHERN_AFRICA**

SOUTH_AMERICA

public static final *Airport.ForeignRegionCAA* **SOUTH_AMERICA**

UNITED_STATES_OF_AMERICA

public static final *Airport.ForeignRegionCAA* **UNITED_STATES_OF_AMERICA**

WESTERN_EUROPE_EU

public static final *Airport.ForeignRegionCAA* **WESTERN_EUROPE_EU**

WESTERN_EUROPE_OTHER

public static final *Airport.ForeignRegionCAA* **WESTERN_EUROPE_OTHER**

WEST_AFRICA

public static final *Airport.ForeignRegionCAA* **WEST_AFRICA**

Methods

getName

public String **getName**()

DomesticAirport

public class **DomesticAirport** extends *Airport*

This class stores information about a domestic (UK) airport.

Author Milan Lovric

Constructors

DomesticAirport

public **DomesticAirport**(String *iataCode*, String *atcoCode*, String *caaName*, String *naptanName*, String *ourAirportsName*, int *easting*, int *northing*, double *longitude*, double *latitude*, String *ladCode*, String *ladName*, long *terminalCapacity*, long *runwayCapacity*)

Constructor for the airport.

Parameters

- **iataCode** – Airport IATA code.
- **atcoCode** – Airport ATCO code.
- **caaName** – Airport name in CCA datasets.
- **naptanName** – Airport name in NaPTAN (for UK airpots).
- **ourAirportsName** – Airport name in ourAirports dataset.
- **easting** – Easting coordinate.
- **northing** – Northing coordinate.
- **longitude** – Longitude coordinate.
- **latitude** – Latitude coordinate.

- **ladCode** – LAD code of the zone in which the airport is located (for UK airports).
- **ladName** – LAD name of the zone in which the airport is located (for UK airports).
- **terminalCapacity** – Airport terminal capacity (max number of passengers that can be processed).
- **runwayCapacity** – Airport runway capacity (max number of flights that can be processed).

DomesticAirport

public **DomesticAirport** (*DomesticAirport airport*)
Constructor for an airport.

Parameters

- **airport** – Airport which data is going to be copied.

Methods

getAtcoCode

public String **getAtcoCode** ()
Getter method for the airport ATCO code.

Returns NaPTAN name.

getEasting

public int **getEasting** ()
Getter method for easting.

Returns Easting.

getLADCode

public String **getLADCode** ()
Getter method for the LAD code in which station is located.

Returns LAD code.

getLADName

public String **getLADName** ()
Getter method for the LAD name in which station is located.

Returns LAD name.

getNaPTANName

public String **getNaPTANName**()
Getter method for the airport NaPTAN name.

Returns NaPTAN name.

getNorthing

public int **getNorthing**()
Getter method for Northing.

Returns Northing.

toString

public String **toString**()

DomesticInternodalPassengerDemand

public class **DomesticInternodalPassengerDemand** extends *InternodalPassengerDemand*
This class encapsulates domestic internodal (domestic airport to domestic airport) passenger data.

Author Milan Lovric

Constructors

DomesticInternodalPassengerDemand

public **DomesticInternodalPassengerDemand**()

DomesticInternodalPassengerDemand

public **DomesticInternodalPassengerDemand**(String *fileName*)
Constructor that reads OD matrix from an input csv file.

Parameters

- **fileName** – Path to the input file.

Throws

- **IOException** – if any.
- **FileNotFoundException** – if any.

Methods

saveAirPassengerDemand

public void **saveAirPassengerDemand**(int *year*, String *outputFile*)
Saves air passenger demand to an output file.

Parameters

- **year** – Year of the data.
- **outputFile** – Output file name (with path).

InternationalAirport

public class **InternationalAirport** extends *Airport*
This class stores information about an international (non-UK) airport.

Author Milan Lovric

Constructors

InternationalAirport

public **InternationalAirport**(String *iataCode*, String *caaName*, String *ourAirportsName*, double *longitude*, double *latitude*, String *countryCode*, String *continentCode*, long *terminalCapacity*, long *runwayCapacity*)
Constructor for the airport.

Parameters

- **iataCode** – Airport IATA code.
- **caaName** – Airport name in CCA datasets.
- **ourAirportsName** – Airport name in ourAirports dataset.
- **longitude** – Longitude coordinate.
- **latitude** – Latitude coordinate.
- **countryCode** – Code of the country in which the airport is located.
- **countryCode** – Code of the continent in which the airport is located.
- **terminalCapacity** – Airport terminal capacity (max number of passengers that can be processed).
- **runwayCapacity** – Airport runway capacity (max number of flights that can be processed).

InternationalAirport

public **InternationalAirport**(*InternationalAirport airport*)
Constructor for an airport using an existing airport.

Parameters

- **airport** – Airport which data is going to be copied.

Methods

toString

public String **toString**()

InternationalInternodalPassengerDemand

public class **InternationalInternodalPassengerDemand** extends *InternodalPassengerDemand*
This class encapsulates international internodal (domestic airport to international airport) passenger data.

Author Milan Lovric

Constructors

InternationalInternodalPassengerDemand

public **InternationalInternodalPassengerDemand**()

InternationalInternodalPassengerDemand

public **InternationalInternodalPassengerDemand**(String *fileName*)
Constructor that reads OD matrix from an input csv file.

Parameters

- **fileName** – Path to the input file.

Throws

- **IOException** – if any.
- **FileNotFoundException** – if any.

Methods

saveAirPassengerDemand

public void **saveAirPassengerDemand**(int *year*, String *outputFile*)
Saves air passenger demand to an output file.

Parameters

- **year** – Year of the data.
- **outputFile** – Output file name (with path).

InternodalPassengerDemand

public abstract class **InternodalPassengerDemand**
This class encapsulates internodal (airport to airport) passenger data.

Author Milan Lovric

Fields

data

protected MultiKeyMap **data**

Constructors

InternodalPassengerDemand

public **InternodalPassengerDemand**()

Methods

getDemand

public Map<*Passengers*, Long> **getDemand**(String *firstIATA*, String *secondIATA*)

printDemand

public void **printDemand**()

saveAirPassengerDemand

public abstract void **saveAirPassengerDemand**(int *year*, String *file*)

setDemand

public void **setDemand**(String *firstIATA*, String *secondIATA*, long *totalPax*, long *scheduledPax*, long *charterPax*)

InternodalPassengerDemand.Passengers

public static enum **Passengers**

Enum Constants

CHARTER

public static final *InternodalPassengerDemand*.*Passengers* **CHARTER**

SCHEDULED

public static final *InternodalPassengerDemand*.*Passengers* **SCHEDULED**

TOTAL

public static final *InternodalPassengerDemand.Passengers* **TOTAL**

1.2.3 nismod.transport.decision

CongestionCharging

public class **CongestionCharging** extends *Intervention*

Intervention that implements link-based congestion charge which depends on the vehicle type and time of day when trip is made.

Author Milan Lovric

Constructors

CongestionCharging

public **CongestionCharging** (Properties *props*)

Constructor.

Parameters

- **props** – Properties object.

CongestionCharging

public **CongestionCharging** (String *fileName*)

Constructor.

Parameters

- **fileName** – Path to the input properties file.

Methods

install

public void **install** (Object *o*)

uninstall

public void **uninstall** (Object *o*)

Intervention

public abstract class **Intervention**

Abstract class for policy interventions.

Author Milan Lovric

Fields

installed

protected boolean **installed**

props

protected Properties **props**

Constructors

Intervention

protected **Intervention** (Properties *props*)

Intervention

protected **Intervention** (String *fileName*)

Methods

getEndYear

public int **getEndYear** ()

Returns The last year in which intervention still remains installed.

getProperty

public String **getProperty** (String *key*)

Parameters

- **key** – Name of the property

Returns Property

getStartYear

public int **getStartYear** ()

Returns The year in which intervention is installed.

getState

public boolean **getState** ()

install

public abstract void **install** (Object *o*)

toString

public String **toString** ()

uninstall

public abstract void **uninstall** (Object *o*)

Intervention.InterventionType

public static enum **InterventionType**

Enum Constants

CongestionCharging

public static final *Intervention.InterventionType* **CongestionCharging**

NewRailStation

public static final *Intervention.InterventionType* **NewRailStation**

RoadDevelopment

public static final *Intervention.InterventionType* **RoadDevelopment**

RoadExpansion

public static final *Intervention.InterventionType* **RoadExpansion**

NewRailStation

public class **NewRailStation** extends *Intervention*

Intervention that builds a new rail station.

Author Milan Lovric

Constructors

NewRailStation

public **NewRailStation** (Properties *props*)
Constructor.

Parameters

- **props** – Properties of the intervention.

NewRailStation

public **NewRailStation** (String *fileName*)
Constructor.

Parameters

- **fileName** – File with the properties.

Methods

getNLC

public Integer **getNLC** ()

Returns NLC code of the new rail station.

install

public void **install** (Object *o*)

uninstall

public void **uninstall** (Object *o*)

PricingPolicy

public class **PricingPolicy**
A class that encapsulates the pricing policy for congestion charging intervention.

Author Milan Lovric

Constructors

PricingPolicy

public **PricingPolicy** (String *policyName*, String *fileName*, int *maxEdgeID*, List<Integer> *edgeIDs*)
Reads congestion charge file which contains charges that depend on vehicle type and time of day (hour).

Parameters

- **policyName** – Name of the policy.
- **fileName** – File name.
- **maxEdgeID** – Maximum edge ID.
- **edgeIDs** – List of edge IDs affected by the policy.

Throws

- **IOException** – if any.
- **FileNotFoundException** – if any.

Return Map with congestion charges.

Methods

getLinkCharges

public double[] **getLinkCharges** (*VehicleType vht*, *TimeOfDay time*)
Get link charges for a particular combination of vehicle type and time of day.

Parameters

- **vht** – Vehicle type.
- **time** – Time of day.

Returns Array with link charges.

getPolicy

public EnumMap<*VehicleType*, EnumMap<*TimeOfDay*, double[]>> **getPolicy** ()
Get the entire pricing policy (for combinations of vehicle type and time of day).

Returns Pricing policy.

getPolicyEdges

public List<Integer> **getPolicyEdges** ()
Return policy edges.

Returns Policy edges.

getPolicyName

public String **getPolicyName** ()
Return policy name.

Returns Policy name.

RoadDevelopment

public class **RoadDevelopment** extends *Intervention*
Intervention that creates a new road link between two existing nodes.

Author Milan Lovric

Constructors

RoadDevelopment

public **RoadDevelopment** (Properties *props*)
Constructor.

Parameters

- **props** – Properties of the road development intervention.

RoadDevelopment

public **RoadDevelopment** (String *fileName*)
Constructor.

Parameters

- **fileName** – File with the properties.

Methods

getDevelopedEdgeID

public Integer **getDevelopedEdgeID** ()

Returns Edge ID of the developed road link.

getDevelopedEdgeID2

public Integer **getDevelopedEdgeID2** ()

Returns Edge ID of the developed road link (in other direction)

install

public void **install** (Object *o*)

uninstall

public void **uninstall** (Object *o*)

RoadExpansion

public class **RoadExpansion** extends *Intervention*
Intervention that expands a road link with a number of lanes.

Author Milan Lovric

Constructors

RoadExpansion

public **RoadExpansion** (Properties *props*)
Constructor.

Parameters

- **props** – Properties of the road expansion intervention.

RoadExpansion

public **RoadExpansion** (String *fileName*)
Constructor.

Parameters

- **fileName** – File with the properties.

Methods

getExpandedEdgeID

public Integer **getExpandedEdgeID** (*RoadNetwork roadNetwork*)

Parameters

- **roadNetwork** – Road network

Returns Edge ID which should be expanded.

install

public void **install** (Object *o*)

uninstall

public void **uninstall** (Object *o*)

1.2.4 nismod.transport.demand

AssignableODMatrix

public interface **AssignableODMatrix**
Origin-destination matrix for passenger vehicles.

Author Milan Lovric

Methods

getIntFlow

public int **getIntFlow**(String *originZone*, String *destinationZone*)
Gets the flow for a given origin-destination pair as a whole number.

Parameters

- **originZone** – Origin zone.
- **destinationZone** – Destination zone.

Returns Origin-destination flow.

getSortedDestinations

public List<String> **getSortedDestinations**()
Gets the sroted list of destinations.

Returns List of destination zones.

getSortedOrigins

public List<String> **getSortedOrigins**()
Gets the sorted list of origins.

Returns List of origin zones.

getTotalIntFlow

public int **getTotalIntFlow**()
Gets sum of all the flows in the matrix.

Returns Sum of all the flows in the matrix (i.e. number of trips).

getUnsortedDestinations

public List<String> **getUnsortedDestinations**()
Gets the unsorted list of destinations.

Returns List of destination zones.

getUnsortedOrigins

public List<String> **getUnsortedOrigins** ( )
Gets the unsorted list of origins.

Returns List of origin zones.

DemandModel

public class **DemandModel**
Main demand prediction model (elasticity-based).

Author Milan Lovric

Fields

assignmentIterations

public final int **assignmentIterations**

baseYear

public final int **baseYear**

baseYearFreight

public final int **baseYearFreight**

freightScalingFactor

public final double **freightScalingFactor**

linkTravelTimeAveragingWeight

public final double **linkTravelTimeAveragingWeight**

predictionIterations

public final int **predictionIterations**

Constructors

DemandModel

public **DemandModel** (*RoadNetwork roadNetwork*, String *baseYearODMatrixFile*, String *baseYearFreightMatrixFile*, String *populationFile*, String *GVAFile*, String *elasticitiesFile*, String *elasticitiesFreightFile*, String *energyUnitCostsFile*, String *unitCO2EmissionsFile*, String *engineTypeFractionsFile*, String *autonomousVehiclesFractionsFile*, List<*Intervention*> *interventions*, *RouteSetGenerator rsg*, *Zoning zoning*, Properties *props*)

The constructor for the demand prediction model.

Parameters

- **roadNetwork** – Road network for assignment.
- **baseYearODMatrixFile** – Base-year origin-destination matrix file name.
- **baseYearFreightMatrixFile** – Base-year freight matrix file name.
- **populationFile** – Population file name.
- **GVAFile** – GVA file name.
- **elasticitiesFile** – Elasticities file name.
- **elasticitiesFreightFile** – Elasticities freight file name.
- **energyUnitCostsFile** – Energy unit costs file name.
- **unitCO2EmissionsFile** – Unit CO2 emissions file name.
- **engineTypeFractionsFile** – Engine type fractions file.
- **autonomousVehiclesFractionsFile** – Autonomous vehicles fractions file.
- **interventions** – List of interventions.
- **rsg** – Route Set Generator with routes for both cars and freight.
- **zoning** – Zoning system (used for 'tempro' and 'combined' assignment type).
- **props** – Properties file.

Throws

- **IOException** – if any.
- **FileNotFoundException** – if any.

Methods

addCongestionCharges

public void **addCongestionCharges** (int *year*, *PricingPolicy congestionCharges*)

Adds congestion charges to the list of the existing ones.

Parameters

- **year** – Year of the policy.
- **congestionCharges** – Congestion charges.

assignBaseYear

public void **assignBaseYear**()
Assigned base year demand.

getCongestionCharges

public List<*PricingPolicy*> **getCongestionCharges**(int *year*)
Getter method for congestion charges.

Parameters

- **year** – Year of the congestion charges.

Returns Congestion charges.

getCostSkimMatrix

public *SkimMatrix* **getCostSkimMatrix**(int *year*)
Getter method for cost skim matrix in a given year.

Parameters

- **year** – Year for which the the skim matrix is requested.

Returns Cost skim matrix.

getCostSkimMatrixFreight

public *SkimMatrixFreight* **getCostSkimMatrixFreight**(int *year*)
Getter method for freight cost skim matrix in a given year.

Parameters

- **year** – Year for which the the skim matrix is requested.

Returns Cost skim matrix.

getEngineTypeFractions

public Map<*VehicleType*, Map<*EngineType*, Double>> **getEngineTypeFractions**(int *year*)
Getter method for engine type fractions in a given year.

Parameters

- **year** – Year of the data.

Returns Map with engine type fractions.

getFreightDemand

public *FreightMatrix* **getFreightDemand**(int *year*)
Getter method for the freight demand in a given year.

Parameters

- **year** – Year for which the demand is requested.

Returns Origin-destination matrix with freight vehicle flows.

getListsOfLADsForNewRouteGeneration

public HashMap<Integer, List<List<String>>> **getListsOfLADsForNewRouteGeneration**()
Getter method for the list of LADs.

Returns Lists of LADs for new route generation.

getPassengerDemand

public *ODMatrixMultiKey* **getPassengerDemand**(int *year*)
Getter method for the passenger demand in a given year.

Parameters

- **year** – Year for which the demand is requested.

Returns Origin-destination matrix with passenger vehicle flows.

getRoadNetwork

public *RoadNetwork* **getRoadNetwork**()
Getter method for the road network.

Returns Road network.

getRoadNetworkAssignment

public *RoadNetworkAssignment* **getRoadNetworkAssignment**(int *year*)
Getter method for the road network assignment in a given year.

Parameters

- **year** – Year for which the road network assignment is requested.

Returns Road network assignment.

getTimeSkimMatrix

public *SkimMatrix* **getTimeSkimMatrix**(int *year*)
Getter method for time skim matrix in a given year.

Parameters

- **year** – Year for which the the skim matrix is requested.

Returns Time skim matrix.

getTimeSkimMatrixFreight

public *SkimMatrixFreight* **getTimeSkimMatrixFreight** (int *year*)
Getter method for freight time skim matrix in a given year.

Parameters

- **year** – Year for which the the skim matrix is requested.

Returns Time skim matrix.

predictHighwayDemand

public void **predictHighwayDemand** (int *predictedYear*, int *fromYear*)
Predicts (passenger and freight) highway demand (origin-destination vehicle flows).

Parameters

- **predictedYear** – The year for which the demand is predicted.
- **fromYear** – The year from which demand the prediction is made.

predictHighwayDemandUsingResultsOfFromYear

public void **predictHighwayDemandUsingResultsOfFromYear** (int *predictedYear*, int *fromYear*)
Predicts (passenger and freight) highway demand (origin-destination vehicle flows). Uses already existing results of the fromYear.

Parameters

- **predictedYear** – The year for which the demand is predicted.
- **fromYear** – The year from which demand the prediction is made.

predictHighwayDemands

public void **predictHighwayDemands** (int *toYear*, int *baseYear*)
Predicts (passenger and freight) highway demand (origin-destination vehicle flows) for all years from baseYear to toYear

Parameters

- **toYear** – The final year for which the demand is predicted.
- **baseYear** – The base year from which the predictions are made.

removeCongestionCharges

public void **removeCongestionCharges** (int *year*, *PricingPolicy* *congestionCharges*)
Removes congestion charges from the list of the existing ones.

Parameters

- **year** – Year of the congestion charges.
- **congestionCharges** – Congestion charges.

removeCongestionCharges

public void **removeCongestionCharges** (int *year*, String *policyName*)
Removes congestion charges from the list of the existing ones using the policy name.

Parameters

- **year** – Year of the congestion charges.
- **policyName** – Name of the policy.

saveAllResults

public void **saveAllResults** (int *toYear*, int *baseYear*)
Saves all results from baseYear to toYear (including intermediate if flat is set)

Parameters

- **toYear** – The final year for which the demand is predicted.
- **baseYear** – The base year from which the predictions are made.

saveAllResults

public void **saveAllResults** (int *year*)
Saves all results into the output folder.

Parameters

- **year** – Year of the data.

saveAssignmentResults

public void **saveAssignmentResults** (int *year*, String *outputFile*)
Saves road network assignment results into a csv file.

Parameters

- **year** – Year of the data.
- **outputFile** – Output file name.

saveEnergyConsumptions

public void **saveEnergyConsumptions** (int *year*, String *outputFile*)
Saves energy consumptions into a csv file.

Parameters

- **year** – Year of the data.
- **outputFile** – Output file name.

setCongestionCharges

public void **setCongestionCharges**(int *year*, List<*PricingPolicy*> *congestionCharges*)
Setter method for congestion charges (overrides them completely).

Parameters

- **year** – Year of the congestion charges.
- **congestionCharges** – Congestion charges.

setEngineTypeFractions

public void **setEngineTypeFractions**(int *year*, Map<*VehicleType*, Map<*EngineType*, Double>> *engineTypeFractions*)
Setter method for engine type fractions in a given year.

Parameters

- **year** – Year of the data.
- **engineTypeFractions** – Map with engine type fractions.

setEngineTypeFractions

public void **setEngineTypeFractions**(int *year*, *VehicleType vht*, Map<*EngineType*, Double> *engineTypeFractions*)
Setter method for engine type fractions in a given year for a specific vehicle type.

Parameters

- **year** – Year of the data.
- **vht** – Vehicle type.
- **engineTypeFractions** – Map with engine type fractions.

DemandModel.ElasticityTypes

public static enum **ElasticityTypes**

Enum Constants

COST

public static final *DemandModel.ElasticityTypes* **COST**

GVA

public static final *DemandModel.ElasticityTypes* **GVA**

POPULATION

public static final *DemandModel.ElasticityTypes* **POPULATION**

TIME

public static final *DemandModel.ElasticityTypes* **TIME**

EstimatedODMatrix

public class **EstimatedODMatrix** extends *RealODMatrix*

Origin-destination matrix created from productions, attractions and observed trip length distribution.

Author Milan Lovric

Fields

BIN_LIMITS_KM

public static final double[] **BIN_LIMITS_KM**

BIN_LIMITS_MILES

public static final int[] **BIN_LIMITS_MILES**

OTLD

public static final double[] **OTLD**

Constructors

EstimatedODMatrix

public **EstimatedODMatrix**(HashMap<String, Integer> *productions*, HashMap<String, Integer> *attractions*, *SkimMatrix* *distanceSkimMatrix*, double[] *binLimitsKm*, double[] *observedTripLengthDistribution*)

Constructor for estimated OD matrix.

Parameters

- **productions** – Productions
- **attractions** – Attractions
- **distanceSkimMatrix** – Distance skim matrix
- **binLimitsKm** – Bin limits in km
- **observedTripLengthDistribution** – Observed trip length distribution (normalised).

EstimatedODMatrix

public **EstimatedODMatrix**(String *fileName*, *SkimMatrix* *distanceSkimMatrix*, double[] *binLimitsKm*, double[] *observedTripLengthDistribution*)

Constructor for estimated OD matrix that reads productions and attractions from an input csv file.

Parameters

- **fileName** – Path to the input file with productions and attractions
- **distanceSkimMatrix** – Distance skim matrix
- **binLimitsKm** – Bin limits in km
- **observedTripLengthDistribution** – Observed trip length distribution (normalised).

Throws

- **IOException** – if any.
- **FileNotFoundException** – if any.

Methods

createUnitMatrix

public void **createUnitMatrix**()

Creates a unit OD matrix (all ones).

deleteInterzonalFlows

public void **deleteInterzonalFlows**(String *zone*)

Deletes all inter-zonal flows to/from a particular zone (leaving only intra-zonal flows)

Parameters

- **zone** – Zone for which inter-zonal flows need to be deleted from the origin-destination matrix.

getAttractions

public HashMap<String, Integer> **getAttractions**()

Getter method for the attractions.

Returns Attractions

getBinIndexMatrix

public *ODMatrixMultiKey* **getBinIndexMatrix**()

Getter method for the bin index matrix.

Returns Bin index matrix

getObservedTripLengthDistribution

public double[] **getObservedTripLengthDistribution**()
Getter method for the observed trip length distribution.

Returns Observed trip length distribution

getProductions

public HashMap<String, Integer> **getProductions**()
Getter method for the productions.

Returns Productions

getTripLengthDistribution

public double[] **getTripLengthDistribution**()
Getter method for the trip length distribution.

Returns Trip length distribution

iterate

public void **iterate**()
Iterates scaling to productions, scaling to attractions, rounding and scaling to observed trip length distribution.

printMatrixFormatted

public void **printMatrixFormatted**(int *precision*)
Prints the matrix as a formatted table.

printMatrixFormatted

public void **printMatrixFormatted**(String *message*, int *precision*)
Prints the message and the matrix as a formatted table.

scaleToAttractions

public void **scaleToAttractions**()
Scales OD matrix to attractions.

scaleToObservedTripLenghtDistribution

public void **scaleToObservedTripLenghtDistribution**()
Scales OD matrix to observed trip length distribution.

scaleToProductions

public void **scaleToProductions**()
Scales OD matrix to productions.

updateTripLengthDistribution

public void **updateTripLengthDistribution**()
Updates trip length distribution (using the current values of the OD matrix).

FreightMatrix

public class **FreightMatrix**
Origin-destination matrix for freight vehicles (following the format of DfT's BYFM 2006 study).

Author Milan Lovric

Fields

MAX_FREIGHT_ZONE_ID

public static final int **MAX_FREIGHT_ZONE_ID**

MAX_VEHICLE_ID

public static final int **MAX_VEHICLE_ID**

Constructors

FreightMatrix

public **FreightMatrix**()

FreightMatrix

public **FreightMatrix**(String *fileName*)
Constructor that reads OD matrix from an input csv file.

Parameters

- **fileName** – Path to the input file.

Throws

- **IOException** – if any.
- **FileNotFoundException** – if any.

Methods

createUnitBYFMMatrix

public static *FreightMatrix* **createUnitBYFMMatrix**()
Creates a unit freight matrix for the specific DfT BYFM 2006 zoning system.

Returns Unit BYFM freight matrix.

createUnitMatrix

public static *FreightMatrix* **createUnitMatrix**(List<Integer> *origins*, List<Integer> *destinations*)
Creates a unit freight matrix for given lists of origin and destination zones.

Parameters

- **origins** – List of origin zones.
- **destinations** – List of destination zones.

Returns Unit freight matrix.

deleteInterzonalFlows

public void **deleteInterzonalFlows**(int *zone*)
Deletes all inter-zonal flows to/from a particular zone (leaving only intra-zonal flows)

Parameters

- **zone** – Zone for which inter-zonal flows need to be deleted from the freight matrix.

getAbsoluteDifference

public double **getAbsoluteDifference**(*FreightMatrix other*)
Gets sum of absolute differences between elements of two matrices.

Parameters

- **other** – The other matrix.

Returns Sum of absolute differences.

getFlow

public int **getFlow**(int *origin*, int *destination*, int *vehicleType*)
Gets the flow for a given origin-destination pair.

Parameters

- **origin** – Freight origin.
- **destination** – Freight destination.
- **vehicleType** – Vehicle type.

Returns Origin-destination flow.

getKeySet

public Set<MultiKey> **getKeySet**()
Gets the keyset of the multimap.

Returns Key set.

getScaledMatrix

public *FreightMatrix* **getScaledMatrix**(double *scale*)
Multiplies each value of the matrix with a scaling factor.

Parameters

- **scale** – Scaling factor.

Returns Scaled freight matrix.

getSortedDestinations

public List<Integer> **getSortedDestinations**()
Gets the sorted list of destinations.

Returns List of destinations.

getSortedOrigins

public List<Integer> **getSortedOrigins**()
Gets the sorted list of origins.

Returns List of origins.

getTotalIntFlow

public int **getTotalIntFlow**()
Gets sum of all the flows in the matrix.

Returns Sum of all the flows in the matrix (i.e. number of trips).

getUnsortedDestinations

public List<Integer> **getUnsortedDestinations**()
Gets the unsorted list of destinations.

Returns List of destinations.

getUnsortedOrigins

public List<Integer> **getUnsortedOrigins**()
Gets the unsorted list of origins.

Returns List of origins.

getVehicleTypes

public List<Integer> **getVehicleTypes**()
Gets the sorted list of vehicle types.

Returns List of vehicle types.

printMatrix

public void **printMatrix**()
Prints the full matrix.

printMatrixFormatted

public void **printMatrixFormatted**()
Prints the matrix as a formatted table.

printMatrixFormatted

public void **printMatrixFormatted**(String *s*)
Prints the matrix as a formatted table, with a print message.

Parameters

- **s** – Print message

saveMatrixFormatted

public void **saveMatrixFormatted**(String *outputFile*)
Saves the matrix into a csv file.

Parameters

- **outputFile** – Path to the output file.

scaleMatrix

public void **scaleMatrix**(*SkimMatrixFreight scale*)
Scales matrix flows using a real-valued scaling matrix.

Parameters

- **scale** – Scaling factors.

setFlow

public void **setFlow**(int *origin*, int *destination*, int *vehicleType*, int *flow*)
Sets the flow for a given origin-destination pair.

Parameters

- **origin** – Freight origin.

- **destination** – Freight destination.
- **vehicleType** – Vehicle type.
- **flow** – Origin-destination flow.

ODMatrixArray

public class **ODMatrixArray** implements *AssignableODMatrix*
Origin-destination matrix for passenger vehicles.

Author Milan Lovric

Constructors

ODMatrixArray

public **ODMatrixArray**(*Zoning zoning*)
Constructor.

Parameters

- **zoning** – Zoning system.

ODMatrixArray

public **ODMatrixArray**(*RealODMatrix realMatrix*, *Zoning zoning*)
Constructor that rounds the flows of a real-valued OD matrix.

Parameters

- **realMatrix** – Origin-destination matrix with real-valued flows.
- **zoning** – Zoning system.

ODMatrixArray

public **ODMatrixArray**(String *fileName*, *Zoning zoning*)
Constructor that reads OD matrix from an input csv file.

Parameters

- **fileName** – Path to the input file.
- **zoning** – Zoning system.

Throws

- **IOException** – if any.
- **FileNotFoundException** – if any.

Methods

calculateTripEnds

public HashMap<String, Integer> **calculateTripEnds**()
Calculates the number of trips ending in each destination zone

Returns Number of trip ends.

calculateTripStarts

public HashMap<String, Integer> **calculateTripStarts**()
Calculates the number of trips starting in each origin zone

Returns Number of trip starts.

createLadMatrixFromTEMProMatrix

public static *ODMatrixArray* **createLadMatrixFromTEMProMatrix**(*ODMatrixArrayTempro temproMatrix*, *Zoning zoning*)
Creates LAD OD matrix from TEMPro OD matrix.

Parameters

- **temproMatrix** – TEMPro ODMatrix used as weights to disaggregate LAD matrix.
- **zoning** – Zoning system with mapping between TEMPro and LAD zones.

Returns LAD based OD matrix.

createTEMProFromLadMatrix

public static *ODMatrixArrayTempro* **createTEMProFromLadMatrix**(*ODMatrixArray ladODMatrix*, *ODMatrixArrayTempro baseTempro*, *Zoning zoning*)
Creates tempro OD matrix from LAD OD matrix.

Parameters

- **ladODMatrix** – LAD to LAD OD matrix.
- **baseTempro** – TEMPro ODMatrix used as weights to disaggregate LAD matrix.
- **zoning** – Zoning system with mapping between TEMPro and LAD zones.

Returns TEMPro based OD matrix.

createUnitMatrix

public static *ODMatrixArray* **createUnitMatrix**(List<String> *origins*, List<String> *destinations*, *Zoning zoning*)
Creates a unit OD matrix for given lists of origin and destination zones.

Parameters

- **origins** – List of origin zones.

- **destinations** – List of destination zones.
- **zoning** – Zoning system.

Returns Unit OD matrix.

createUnitMatrix

public static *ODMatrixArray* **createUnitMatrix** (List<String> *zones*, *Zoning zoning*)
Creates a quadratic unit OD matrix for a given lists of zones.

Parameters

- **zones** – List of origin zones.
- **zoning** – Zoning system.

Returns Unit OD matrix.

createUnitMatrix

public static *ODMatrixArray* **createUnitMatrix** (Set<String> *zones*, *Zoning zoning*)
Creates a quadratic unit OD matrix for a given lists of zones.

Parameters

- **zones** – Set of origin zones.
- **zoning** – Zoning system.

Returns Unit OD matrix.

deleteInterzonalFlows

public void **deleteInterzonalFlows** (String *zone*)
Deletes all inter-zonal flows to/from a particular zone (leaving only intra-zonal flows)

Parameters

- **zone** – Zone for which inter-zonal flows need to be deleted from the origin-destination matrix.

getAbsoluteDifference

public int **getAbsoluteDifference** (*ODMatrixArray other*)
Gets sum of absolute differences between elements of two matrices.

Parameters

- **other** – The other matrix.

Returns Sum of absolute differences.

getFlow

public int **getFlow**(String *originZone*, String *destinationZone*)
Gets the flow for a given origin-destination pair.

Parameters

- **originZone** – Origin zone.
- **destinationZone** – Destination zone.

Returns Origin-destination flow.

getFlow

public int **getFlow**(int *originZoneID*, int *destinationZoneID*)
Gets the flow for a given origin-destination pair.

Parameters

- **originZoneID** – Origin zone ID.
- **destinationZoneID** – Destination zone ID.

Returns Origin-destination flow.

getIntFlow

public int **getIntFlow**(String *originZone*, String *destinationZone*)
Gets the flow for a given origin-destination pair as a whole number.

Parameters

- **originZone** – Origin zone.
- **destinationZone** – Destination zone.

Returns Origin-destination flow.

getIntFlow

public int **getIntFlow**(int *originZoneID*, int *destinationZoneID*)
Gets the flow for a given origin-destination pair as a whole number.

Parameters

- **originZoneID** – Origin zone ID.
- **destinationZoneID** – Destination zone ID.

Returns Origin-destination flow.

getSortedDestinations

public List<String> **getSortedDestinations**()
Gets the sorted list of destinations.

Returns List of destinations.

getSortedOrigins

public List<String> **getSortedOrigins**()
Gets the sorted list of origins.

Returns List of origins.

getTotalFlow

public int **getTotalFlow**()
Gets sum of all the flows in the matrix.

Returns Sum of all the flows in the matrix (i.e. number of trips).

getTotalIntFlow

public int **getTotalIntFlow**()
Gets sum of all the flows in the matrix.

Returns Sum of all the flows in the matrix (i.e. number of trips).

getUnsortedDestinations

public List<String> **getUnsortedDestinations**()
Gets the unsorted list of destinations.

Returns List of destinations.

getUnsortedOrigins

public List<String> **getUnsortedOrigins**()
Gets the unsorted list of origins.

Returns List of origins.

printMatrix

public void **printMatrix**()
Prints the full matrix.

printMatrixFormatted

public void **printMatrixFormatted**()
Prints the matrix as a formatted table.

printMatrixFormatted

public void **printMatrixFormatted**(String *s*)
Prints the matrix as a formatted table, with a print message.

Parameters

- **s** – Print message

saveMatrixFormatted

public void **saveMatrixFormatted**(String *outputFile*)
Saves the matrix into a csv file.

Parameters

- **outputFile** – Path to the output file.

saveMatrixFormattedList

public void **saveMatrixFormattedList**(String *outputFile*)
Saves the matrix into a csv file. Uses a list format (origin, destination, flow).

Parameters

- **outputFile** – Path to the output file.

scaleMatrixValue

public void **scaleMatrixValue**(double *factor*)
Scales (and rounds) matrix values with a scaling factor.

Parameters

- **factor** – Scaling factor.

setFlow

public void **setFlow**(int *originZoneID*, int *destinationZoneID*, int *flow*)
Sets the flow for a given origin-destination pair.

Parameters

- **originZoneID** – Origin zone ID.
- **destinationZoneID** – Destination zone ID.
- **flow** – Origin-destination flow.

setFlow

public void **setFlow**(String *originZone*, String *destinationZone*, int *flow*)
Sets the flow for a given origin-destination pair.

Parameters

- **originZone** – Origin zone.
- **destinationZone** – Destination zone.
- **flow** – Origin-destination flow.

ODMatrixArrayTempro

public class **ODMatrixArrayTempro** implements *AssignableODMatrix*

Origin-destination matrix with real values, memory use optimised for Tempro.

Author Milan Lovric

Constructors

ODMatrixArrayTempro

public **ODMatrixArrayTempro** (*Zoning zoning*)

Constructor for an empty OD matrix. Uses the maximum Tempro zone ID which will create a rather large matrix.

Parameters

- **zoning** – Zoning system.

ODMatrixArrayTempro

public **ODMatrixArrayTempro** (String *fileName*, *Zoning zoning*)

Constructor that reads OD matrix from an input csv file. Can use both matrix and list format.

Parameters

- **fileName** – Path to the input file.
- **zoning** – Zoning system.

Throws

- **IOException** – if any.
- **FileNotFoundException** – if any.

Methods

calculateTripEnds

public HashMap<String, Integer> **calculateTripEnds** ()

Calculates the number of trips ending in each destination zone

Returns Number of trip ends.

calculateTripStarts

public HashMap<String, Integer> **calculateTripStarts**()
Calculates the number of trips starting in each origin zone

Returns Number of trip starts.

clone

public *ODMatrixArrayTempro* **clone**()

createLadMatrixFromTEMProMatrix

public static *RealODMatrix* **createLadMatrixFromTEMProMatrix**(*ODMatrixArrayTempro temproMatrix*, *Zoning zoning*)
Creates real-valued LAD OD matrix from real-valued TEMPro OD matrix.

Parameters

- **temproMatrix** – TEMPro ODMatrix which should be aggregated to LAD matrix.
- **zoning** – Zoning system with mapping between TEMPro and LAD zones.

Returns LAD based real-valued OD matrix.

createTEMProFromLadMatrix

public static *ODMatrixArrayTempro* **createTEMProFromLadMatrix**(*ODMatrixArray ladODMatrix*, *ODMatrixArrayTempro baseTempro*, *Zoning zoning*)
Creates tempro OD matrix from LAD OD matrix.

Parameters

- **ladODMatrix** – LAD to LAD OD matrix.
- **baseTempro** – TEMPro ODMatrix used as weights to disaggregate LAD matrix.
- **zoning** – Zoning system with mapping between TEMPro and LAD zones.

Returns TEMPro based OD matrix.

createTEMProFromLadMatrix

public static *ODMatrixArrayTempro* **createTEMProFromLadMatrix**(*ODMatrixMultiKey ladODMatrix*, *ODMatrixArrayTempro baseTempro*, *Zoning zoning*)
Creates tempro OD matrix from LAD OD matrix.

Parameters

- **ladODMatrix** – LAD to LAD OD matrix.
- **baseTempro** – TEMPro ODMatrix used as weights to disaggregate LAD matrix.
- **zoning** – Zoning system with mapping between TEMPro and LAD zones.

Returns TEMPro based OD matrix.

createUnitMatrix

public static *ODMatrixArrayTempro* **createUnitMatrix**(List<String> *origins*, List<String> *destinations*, *Zoning zoning*)

Creates a unit OD matrix for given lists of origin and destination zones.

Parameters

- **origins** – List of origin zones.
- **destinations** – List of destination zones.
- **zoning** – Zoning system.

Returns Unit OD matrix.

createUnitMatrix

public static *ODMatrixArrayTempro* **createUnitMatrix**(List<String> *zones*, *Zoning zoning*)

Creates a quadratic unit OD matrix for a given lists of zones.

Parameters

- **zones** – List of zones.

Returns Unit OD matrix.

createUnitMatrix

public static *ODMatrixArrayTempro* **createUnitMatrix**(Set<String> *zones*, *Zoning zoning*)

Creates a quadratic unit OD matrix for a given lists of zones.

Parameters

- **zones** – Set of zones.
- **zoning** – Zoning system.

Returns Unit OD matrix.

createUnitMatrix

public static *ODMatrixArrayTempro* **createUnitMatrix**(*Zoning zoning*)

Creates a quadratic unit OD matrix.

Parameters

- **zoning** – Zoning system.

Returns Unit OD matrix.

deleteInterzonalFlows

public void **deleteInterzonalFlows**(String *zone*)

Deletes all inter-zonal flows to/from a particular zone (leaving only intra-zonal flows)

Parameters

- **zone** – Zone for which inter-zonal flows need to be deleted from the origin-destination matrix.

getAbsoluteDifference

public int **getAbsoluteDifference**(*ODMatrixArrayTempro other*)
Gets sum of absolute differences between elements of two matrices.

Parameters

- **other** – The other matrix.

Returns Sum of absolute differences.

getFlow

public int **getFlow**(String *originZone*, String *destinationZone*)
Gets the flow for a given origin-destination pair.

Parameters

- **originZone** – Origin zone.
- **destinationZone** – Destination zone.

Returns Origin-destination flow.

getIntFlow

public int **getIntFlow**(String *originZone*, String *destinationZone*)
Gets the flow for a given origin-destination pair, rounded to a whole number.

Parameters

- **originZone** – Origin zone.
- **destinationZone** – Destination zone.

Returns Origin-destination flow.

getSortedDestinations

public List<String> **getSortedDestinations**()
Gets the sorted list of destinations.

Returns List of destinations.

getSortedOrigins

public List<String> **getSortedOrigins**()
Gets the sorted list of origins.

Returns List of origins.

getTotalIntFlow

public int **getTotalIntFlow**()
Gets sum of all the (rounded) flows in the matrix.

Returns Sum of all the (rounded) flows in the matrix (i.e. number of trips).

getUnsortedDestinations

public List<String> **getUnsortedDestinations**()
Gets the unsorted list of destinations.

Returns List of destinations.

getUnsortedOrigins

public List<String> **getUnsortedOrigins**()
Gets the unsorted list of origins.

Returns List of origins.

printMatrix

public void **printMatrix**()
Prints the full matrix.

printMatrixFormatted

public void **printMatrixFormatted**()
Prints the matrix as a formatted table.

printMatrixFormatted

public void **printMatrixFormatted**(String *s*)
Prints message followed by the formatted matrix.

Parameters

- **s** – Message.

saveMatrixFormatted

public void **saveMatrixFormatted**(String *outputFile*)
Saves the matrix into a csv file. Uses a rectangular/matrix format.

Parameters

- **outputFile** – Path to the output file.

saveMatrixFormatted2

public void **saveMatrixFormatted2** (String *outputFile*)
Saves the matrix into a csv file. Uses a list format (origin, destination, flow).

Parameters

- **outputFile** – Path to the output file.

saveMatrixFormatted3

public void **saveMatrixFormatted3** (String *outputFile*)
Saves the matrix into a csv file. Uses a list format (origin, destination, flow) and number codes for zones.

Parameters

- **outputFile** – Path to the output file.

scaleMatrixValue

public void **scaleMatrixValue** (double *factor*)
Scales matrix values with a scaling factor.

Parameters

- **factor** – Scaling factor.

scaleMatrixValue

public void **scaleMatrixValue** (*ODMatrixArrayTempro* *scalingMatrix*)
Scales matrix values with another matrix (element-wise multiplication).

Parameters

- **scalingMatrix** – Scaling matrix.

setFlow

public void **setFlow** (String *originZone*, String *destinationZone*, int *flow*)
Sets the flow for a given origin-destination pair.

Parameters

- **originZone** – Origin zone.
- **destinationZone** – Destination zone.
- **flow** – Origin-destination flow.

setFlow

public void **setFlow** (int *originCode*, int *destinationCode*, int *flow*)
Sets the flow for a given origin-destination pair.

Parameters

- **originCode** – Origin zone integer code.
- **destinationCode** – Destination zone integer code.
- **flow** – Origin-destination flow.

sumMatrixSubset

public int **sumMatrixSubset** (List<String> *origins*, List<String> *destinations*)
Sums the elements of a matrix subset (provided as two lists of origins and destinations).

Parameters

- **origins** – List of origin zones (a subset).
- **destinations** – List of destination zones (a subset).

Returns Sum of the subset.

ODMatrixMultiKey

public class **ODMatrixMultiKey** implements *AssignableODMatrix*
Origin-destination matrix for passenger vehicles.

Author Milan Lovric

Constructors

ODMatrixMultiKey

public **ODMatrixMultiKey** ()

ODMatrixMultiKey

public **ODMatrixMultiKey** (String *fileName*)
Constructor that reads OD matrix from an input csv file.

Parameters

- **fileName** – Path to the input file.

Throws

- **IOException** – if any.
- **FileNotFoundException** – if any.

ODMatrixMultiKey

public **ODMatrixMultiKey** (*RealODMatrix realMatrix*)
Constructor that rounds the flows of a real-valued OD matrix.

Parameters

- **realMatrix** – Origin-destination matrix with real-valued flows.

Methods

calculateTripEnds

public HashMap<String, Integer> **calculateTripEnds**()
Calculates the number of trips ending in each destination zone

Returns Number of trip ends.

calculateTripStarts

public HashMap<String, Integer> **calculateTripStarts**()
Calculates the number of trips starting in each origin zone

Returns Number of trip starts.

clone

public *ODMatrixMultiKey* **clone**()

createLadMatrixFromTEMProMatrix

public static *ODMatrixMultiKey* **createLadMatrixFromTEMProMatrix**(*ODMatrixMultiKey tempro-Matrix*, *Zoning zoning*)
Creates LAD OD matrix from TEMPro OD matrix.

Parameters

- **temproMatrix** – TEMPro ODMatrix used as weights to disaggregate LAD matrix.
- **zoning** – Zoning system with mapping between TEMPro and LAD zones.

Returns LAD based OD matrix.

createSparseUnitMatrix

public static *ODMatrixMultiKey* **createSparseUnitMatrix**(Set<String> *zones*, HashMap<String, Point> *centroids*, double *threshold*)
Creates a unit OD matrix for a given lists of zones with a distance threshold. If straight line distance between origin and destination zone centroids is larger than threshold that flow is zero.

Parameters

- **zones** – Set of origin zones.
- **centroids** – List of zone centroids.
- **threshold** – Distance threshold in [m].

Returns Unit OD matrix.

createTEMProFromLadMatrix

public static *ODMatrixMultiKey* **createTEMProFromLadMatrix**(*ODMatrixMultiKey ladODMatrix*, *ODMatrixMultiKey baseTempro*, *Zoning zoning*)

Creates tempro OD matrix from LAD OD matrix.

Parameters

- **ladODMatrix** – LAD to LAD OD matrix.
- **baseTempro** – TEMPro ODMatrix used as weights to disaggregate LAD matrix.
- **zoning** – Zoning system with mapping between TEMPro and LAD zones.

Returns TEMPro based OD matrix.

createUnitMatrix

public static *ODMatrixMultiKey* **createUnitMatrix**(List<String> *origins*, List<String> *destinations*)

Creates a unit OD matrix for given lists of origin and destination zones.

Parameters

- **origins** – List of origin zones.
- **destinations** – List of destination zones.

Returns Unit OD matrix.

createUnitMatrix

public static *ODMatrixMultiKey* **createUnitMatrix**(List<String> *zones*)

Creates a quadratic unit OD matrix for a given lists of zones.

Parameters

- **zones** – List of origin zones.

Returns Unit OD matrix.

createUnitMatrix

public static *ODMatrixMultiKey* **createUnitMatrix**(Set<String> *zones*)

Creates a quadratic unit OD matrix for a given lists of zones.

Parameters

- **zones** – Set of origin zones.

Returns Unit OD matrix.

deleteInterzonalFlows

public void **deleteInterzonalFlows**(String *zone*)

Deletes all inter-zonal flows to/from a particular zone (leaving only intra-zonal flows)

Parameters

- **zone** – Zone for which inter-zonal flows need to be deleted from the origin-destination matrix.

getAbsoluteDifference

public double **getAbsoluteDifference** (*ODMatrixMultiKey other*)
Gets sum of absolute differences between elements of two matrices.

Parameters

- **other** – The other matrix.

Returns Sum of absolute differences.

getFlow

public int **getFlow** (String *originZone*, String *destinationZone*)
Gets the flow for a given origin-destination pair.

Parameters

- **originZone** – Origin zone.
- **destinationZone** – Destination zone.

Returns Origin-destination flow.

getIntFlow

public int **getIntFlow** (String *originZone*, String *destinationZone*)
Gets the flow for a given origin-destination pair as a whole number.

Parameters

- **originZone** – Origin zone.
- **destinationZone** – Destination zone.

Returns Origin-destination flow.

getKeySet

public Set<MultiKey> **getKeySet** ()
Gets the keyset of the multimap.

Returns Key set.

getMatrixSubset

public *ODMatrixMultiKey* **getMatrixSubset** (List<String> *origins*, List<String> *destinations*)
Creates a new OD matrix (a matrix subset) for given lists of origin and destination zones.

Parameters

- **origins** – List of origin zones.

- **destinations** – List of destination zones.

Returns Matrix subset.

getSortedDestinations

public List<String> **getSortedDestinations**()
Gets the sorted list of destinations.

Returns List of destinations.

getSortedOrigins

public List<String> **getSortedOrigins**()
Gets the sorted list of origins.

Returns List of origins.

getSumOfFlows

public int **getSumOfFlows**()
Gets sum of all the flows.

Returns Sum of flows.

getTotalFlow

public int **getTotalFlow**()
Gets sum of all the flows in the matrix.

Returns Sum of all the flows in the matrix (i.e. number of trips).

getTotalIntFlow

public int **getTotalIntFlow**()
Gets sum of all the flows in the matrix.

Returns Sum of all the flows in the matrix (i.e. number of trips).

getUnsortedDestinations

public List<String> **getUnsortedDestinations**()
Gets the unsorted list of destinations.

Returns List of destinations.

getUnsortedOrigins

public List<String> **getUnsortedOrigins**()
Gets the unsorted list of origins.

Returns List of origins.

printMatrix

public void **printMatrix**()
Prints the full matrix.

printMatrixFormatted

public void **printMatrixFormatted**()
Prints the matrix as a formatted table.

printMatrixFormatted

public void **printMatrixFormatted**(String *s*)
Prints the matrix as a formatted table, with a print message.

Parameters

- **s** – Print message

saveMatrixFormatted

public void **saveMatrixFormatted**(String *outputFile*)
Saves the matrix into a csv file.

Parameters

- **outputFile** – Path to the output file.

saveMatrixFormatted2

public void **saveMatrixFormatted2**(String *outputFile*)
Saves the matrix into a csv file. Uses a list format (origin, destination, flow).

Parameters

- **outputFile** – Path to the output file.

scaleMatrixValue

public void **scaleMatrixValue**(double *factor*)
Scales (and rounds) matrix values with a scaling factor.

Parameters

- **factor** – Scaling factor.

setFlow

public void **setFlow**(String *originZone*, String *destinationZone*, int *flow*)
Sets the flow for a given origin-destination pair.

Parameters

- **originZone** – Origin zone.
- **destinationZone** – Destination zone.
- **flow** – Origin-destination flow.

sumMatrixSubset

public int **sumMatrixSubset**(List<String> *origins*, List<String> *destinations*)
Sums the elements of a matrix subset (provided as two lists of origins and destinations).

Parameters

- **origins** – List of origin zones (a subset).
- **destinations** – List of destination zones (a subset).

Returns Sum of the subset.

RealODMatrix

public class **RealODMatrix** implements *AssignableODMatrix*
Origin-destination matrix with real values.

Author Milan Lovric

Constructors

RealODMatrix

public **RealODMatrix**()

RealODMatrix

public **RealODMatrix**(String *fileName*)
Constructor that reads OD matrix from an input csv file.

Parameters

- **fileName** – Path to the input file.

Throws

- **IOException** – if any.
- **FileNotFoundException** – if any.

Methods

calculateTripEnds

public HashMap<String, Integer> **calculateTripEnds**()
Calculates the number of trips ending in each destination zone

Returns Number of trip ends.

calculateTripStarts

public HashMap<String, Integer> **calculateTripStarts**()
Calculates the number of trips starting in each origin zone

Returns Number of trip starts.

clone

public *RealODMatrix* **clone**()

createLadMatrixFromTEMProMatrix

public static *RealODMatrix* **createLadMatrixFromTEMProMatrix**(*RealODMatrix temproMatrix*, *Zoning zoning*)
Creates LAD OD matrix from Tempro OD matrix.

Parameters

- **temproMatrix** – TEMPro ODMatrix used as weights to disaggregate LAD matrix.
- **zoning** – Zoning system with mapping between TEMPro and LAD zones.

Returns LAD based OD matrix.

createTEMProFromLadMatrix

public static *RealODMatrix* **createTEMProFromLadMatrix**(*RealODMatrix ladODMatrix*, *RealODMatrix baseTempro*, *Zoning zoning*)
Creates tempro OD matrix from LAD OD matrix.

Parameters

- **ladODMatrix** – LAD to LAD OD matrix.
- **baseTempro** – TEMPro ODMatrix used as weights to disaggregate LAD matrix.
- **zoning** – Zoning system with mapping between TEMPro and LAD zones.

Returns TEMPro based OD matrix.

createUnitMatrix

public static *RealODMatrix* **createUnitMatrix** (List<String> *origins*, List<String> *destinations*)
Creates a unit OD matrix for given lists of origin and destination zones.

Parameters

- **origins** – List of origin zones.
- **destinations** – List of destination zones.

Returns Unit OD matrix.

createUnitMatrix

public static *RealODMatrix* **createUnitMatrix** (List<String> *zones*)
Creates a quadratic unit OD matrix for a given lists of zones.

Parameters

- **zones** – List of zones.

Returns Unit OD matrix.

createUnitMatrix

public static *RealODMatrix* **createUnitMatrix** (Set<String> *zones*)
Creates a quadratic unit OD matrix for a given lists of zones.

Parameters

- **zones** – Set of zones.

Returns Unit OD matrix.

deleteInterzonalFlows

public void **deleteInterzonalFlows** (String *zone*)
Deletes all inter-zonal flows to/from a particular zone (leaving only intra-zonal flows)

Parameters

- **zone** – Zone for which inter-zonal flows need to be deleted from the origin-destination matrix.

getAbsoluteDifference

public double **getAbsoluteDifference** (*RealODMatrix* *other*)
Gets sum of absolute differences between elements of two matrices.

Parameters

- **other** – The other matrix.

Returns Sum of absolute differences.

getFlow

public double **getFlow**(String *originZone*, String *destinationZone*)
Gets the flow for a given origin-destination pair.

Parameters

- **originZone** – Origin zone.
- **destinationZone** – Destination zone.

Returns Origin-destination flow.

getIntFlow

public int **getIntFlow**(String *originZone*, String *destinationZone*)
Gets the flow for a given origin-destination pair, rounded to a whole number.

Parameters

- **originZone** – Origin zone.
- **destinationZone** – Destination zone.

Returns Origin-destination flow.

getKeySet

public Set<MultiKey> **getKeySet**()
Gets the keyset of the multimap.

Returns Key set.

getSortedDestinations

public List<String> **getSortedDestinations**()
Gets the sorted list of destinations.

Returns List of destinations.

getSortedOrigins

public List<String> **getSortedOrigins**()
Gets the sorted list of origins.

Returns List of origins.

getSumOfFlows

public double **getSumOfFlows**()
Gets sum of all the flows.

Returns Sum of flows.

getTotalIntFlow

public int **getTotalIntFlow**()
Gets sum of all the (rounded) flows in the matrix.

Returns Sum of all the (rounded) flows in the matrix (i.e. number of trips).

getUnsortedDestinations

public List<String> **getUnsortedDestinations**()
Gets the unsorted list of destinations.

Returns List of destinations.

getUnsortedOrigins

public List<String> **getUnsortedOrigins**()
Gets the unsorted list of origins.

Returns List of origins.

printMatrix

public void **printMatrix**()
Prints the full matrix.

printMatrixFormatted

public void **printMatrixFormatted**(int *precision*)
Prints the matrix as a formatted table.

Parameters

- **precision** – Number of decimal places for the matrix value.

printMatrixFormatted

public void **printMatrixFormatted**(String *s*, int *precision*)
Prints message followed by the formatted matrix.

Parameters

- **s** – Message.
- **precision** – Number of decimal places.

roundMatrixValues

public void **roundMatrixValues**()
Rounds OD matrix values.

saveMatrixFormatted

public void **saveMatrixFormatted**(String *outputFile*)
Saves the matrix into a csv file.

Parameters

- **outputFile** – Path to the output file.

saveMatrixFormatted2

public void **saveMatrixFormatted2**(String *outputFile*)
Saves the matrix into a csv file. Uses a list format (origin, destination, flow).

Parameters

- **outputFile** – Path to the output file.

scaleMatrixValue

public void **scaleMatrixValue**(double *factor*)
Scales matrix values with a scaling factor.

Parameters

- **factor** – Scaling factor.

scaleMatrixValue

public void **scaleMatrixValue**(*RealODMatrix scalingMatrix*)
Scales matrix values with another matrix (element-wise multiplication).

Parameters

- **scalingMatrix** – Scaling matrix.

setFlow

public void **setFlow**(String *originZone*, String *destinationZone*, double *flow*)
Sets the flow for a given origin-destination pair.

Parameters

- **originZone** – Origin zone.
- **destinationZone** – Destination zone.
- **flow** – Origin-destination flow.

sumMatrixSubset

public double **sumMatrixSubset**(List<String> *origins*, List<String> *destinations*)
Sums the elements of a matrix subset (provided as two lists of origins and destinations).

Parameters

- **origins** – List of origin zones (a subset).
- **destinations** – List of destination zones (a subset).

Returns Sum of the subset.

RealODMatrixTempro

public class **RealODMatrixTempro** implements *AssignableODMatrix*
Origin-destination matrix with real values, memory use optimised for Tempro.

Author Milan Lovric

Constructors

RealODMatrixTempro

public **RealODMatrixTempro** (*Zoning zoning*)
Constructor for an empty OD matrix. Uses the maximum Tempro zone ID which will create a rather large matrix.

Parameters

- **zoning** – Zoning system.

RealODMatrixTempro

public **RealODMatrixTempro** (String *fileName*, *Zoning zoning*)
Constructor that reads OD matrix from an input csv file. Can use both matrix and list format.

Parameters

- **fileName** – Path to the input file.
- **zoning** – Zoning system.

Throws

- **IOException** – if any.
- **FileNotFoundException** – if any.

Methods

calculateTripEnds

public HashMap<String, Integer> **calculateTripEnds** ()
Calculates the number of trips ending in each destination zone

Returns Number of trip ends.

calculateTripStarts

public HashMap<String, Integer> **calculateTripStarts**()
Calculates the number of trips starting in each origin zone

Returns Number of trip starts.

ceilMatrixValues

public void **ceilMatrixValues**()
Ceil OD matrix values.

clone

public *RealODMatrixTempro* **clone**()

createLadMatrixFromTEMProMatrix

public static *RealODMatrix* **createLadMatrixFromTEMProMatrix**(*RealODMatrixTempro temproMatrix*, *Zoning zoning*)
Creates real-valued LAD OD matrix from real-valued TEMPro OD matrix.

Parameters

- **temproMatrix** – TEMPro ODMatrix which should be aggregated to LAD matrix.
- **zoning** – Zoning system with mapping between TEMPro and LAD zones.

Returns LAD based real-valued OD matrix.

createTEMProFromLadMatrix

public static *RealODMatrixTempro* **createTEMProFromLadMatrix**(*RealODMatrix ladODMatrix*, *RealODMatrixTempro baseTempro*, *Zoning zoning*)
Creates tempro OD matrix from LAD OD matrix.

Parameters

- **ladODMatrix** – LAD to LAD OD matrix.
- **baseTempro** – TEMPro ODMatrix used as weights to disaggregate LAD matrix.
- **zoning** – Zoning system with mapping between TEMPro and LAD zones.

Returns TEMPro based OD matrix.

createTEMProFromLadMatrix

public static *RealODMatrixTempro* **createTEMProFromLadMatrix**(*ODMatrixMultiKey ladODMatrix*, *RealODMatrixTempro baseTempro*, *Zoning zoning*)
Creates tempro OD matrix from LAD OD matrix.

Parameters

- **ladODMatrix** – LAD to LAD OD matrix.
- **baseTempro** – TEMPro ODMatrix used as weights to disaggregate LAD matrix.
- **zoning** – Zoning system with mapping between TEMPro and LAD zones.

Returns TEMPro based OD matrix.

createUnitMatrix

public static *RealODMatrixTempro* **createUnitMatrix**(List<String> *origins*, List<String> *destinations*, *Zoning zoning*)

Creates a unit OD matrix for given lists of origin and destination zones.

Parameters

- **origins** – List of origin zones.
- **destinations** – List of destination zones.
- **zoning** – Zoning system.

Returns Unit OD matrix.

createUnitMatrix

public static *RealODMatrixTempro* **createUnitMatrix**(List<String> *zones*, *Zoning zoning*)

Creates a quadratic unit OD matrix for a given lists of zones.

Parameters

- **zones** – List of zones.

Returns Unit OD matrix.

createUnitMatrix

public static *RealODMatrixTempro* **createUnitMatrix**(Set<String> *zones*, *Zoning zoning*)

Creates a quadratic unit OD matrix for a given lists of zones.

Parameters

- **zones** – Set of zones.
- **zoning** – Zoning system.

Returns Unit OD matrix.

createUnitMatrix

public static *RealODMatrixTempro* **createUnitMatrix**(*Zoning zoning*)

Creates a quadratic unit OD matrix.

Parameters

- **zoning** – Zoning system.

Returns Unit OD matrix.

deleteInterzonalFlows

public void **deleteInterzonalFlows**(String *zone*)
Deletes all inter-zonal flows to/from a particular zone (leaving only intra-zonal flows)

Parameters

- **zone** – Zone for which inter-zonal flows need to be deleted from the origin-destination matrix.

floorMatrixValues

public void **floorMatrixValues**()
Floor OD matrix values.

getAbsoluteDifference

public double **getAbsoluteDifference**(*RealODMatrixTempro other*)
Gets sum of absolute differences between elements of two matrices.

Parameters

- **other** – The other matrix.

Returns Sum of absolute differences.

getFlow

public double **getFlow**(String *originZone*, String *destinationZone*)
Gets the flow for a given origin-destination pair.

Parameters

- **originZone** – Origin zone.
- **destinationZone** – Destination zone.

Returns Origin-destination flow.

getIntFlow

public int **getIntFlow**(String *originZone*, String *destinationZone*)
Gets the flow for a given origin-destination pair, rounded to a whole number.

Parameters

- **originZone** – Origin zone.
- **destinationZone** – Destination zone.

Returns Origin-destination flow.

getSortedDestinations

public List<String> **getSortedDestinations**()
Gets the sorted list of destinations.

Returns List of destinations.

getSortedOrigins

public List<String> **getSortedOrigins**()
Gets the sorted list of origins.

Returns List of origins.

getSumOfFlows

public double **getSumOfFlows**()
Gets sum of all the flows.

Returns Sum of flows.

getTotalIntFlow

public int **getTotalIntFlow**()
Gets sum of all the (rounded) flows in the matrix.

Returns Sum of all the (rounded) flows in the matrix (i.e. number of trips).

getUnsortedDestinations

public List<String> **getUnsortedDestinations**()
Gets the unsorted list of destinations.

Returns List of destinations.

getUnsortedOrigins

public List<String> **getUnsortedOrigins**()
Gets the unsorted list of origins.

Returns List of origins.

printMatrix

public void **printMatrix**()
Prints the full matrix.

printMatrixFormatted

public void **printMatrixFormatted**(int *precision*)
Prints the matrix as a formatted table.

Parameters

- **precision** – Number of decimal places for the matrix value.

printMatrixFormatted

public void **printMatrixFormatted**(String *s*, int *precision*)
Prints message followed by the formatted matrix.

Parameters

- **s** – Message.
- **precision** – Number of decimal places.

roundMatrixValues

public void **roundMatrixValues**()
Rounds OD matrix values.

saveMatrixFormatted

public void **saveMatrixFormatted**(String *outputFile*)
Saves the matrix into a csv file. Uses a rectangular/matrix format.

Parameters

- **outputFile** – Path to the output file.

saveMatrixFormatted2

public void **saveMatrixFormatted2**(String *outputFile*)
Saves the matrix into a csv file. Uses a list format (origin, destination, flow).

Parameters

- **outputFile** – Path to the output file.

saveMatrixFormatted3

public void **saveMatrixFormatted3**(String *outputFile*)
Saves the matrix into a csv file. Uses a list format (origin, destination, flow) and number codes for zones.

Parameters

- **outputFile** – Path to the output file.

scaleMatrixValue

public void **scaleMatrixValue**(double *factor*)
Scales matrix values with a scaling factor.

Parameters

- **factor** – Scaling factor.

scaleMatrixValue

public void **scaleMatrixValue**(*RealODMatrixTempro scalingMatrix*)
Scales matrix values with another matrix (element-wise multiplication).

Parameters

- **scalingMatrix** – Scaling matrix.

setFlow

public void **setFlow**(String *originZone*, String *destinationZone*, double *flow*)
Sets the flow for a given origin-destination pair.

Parameters

- **originZone** – Origin zone.
- **destinationZone** – Destination zone.
- **flow** – Origin-destination flow.

setFlow

public void **setFlow**(int *originCode*, int *destinationCode*, double *flow*)
Sets the flow for a given origin-destination pair.

Parameters

- **originCode** – Origin zone integer code.
- **destinationCode** – Destination zone integer code.
- **flow** – Origin-destination flow.

sumMatrixSubset

public double **sumMatrixSubset**(List<String> *origins*, List<String> *destinations*)
Sums the elements of a matrix subset (provided as two lists of origins and destinations).

Parameters

- **origins** – List of origin zones (a subset).
- **destinations** – List of destination zones (a subset).

Returns Sum of the subset.

RebalancedFreightMatrix

public class **RebalancedFreightMatrix** extends *FreightMatrix*

Freight matrix created by directly scaling flows using traffic counts. Base on DfT's BYFM 2006 zoning system (LAD + distribution centres + seaports + airports).

Author Milan Lovric

Constructors

RebalancedFreightMatrix

public **RebalancedFreightMatrix**(*RoadNetworkAssignment* *rna*, *RouteSetGenerator* *rsg*, Properties *params*)

Constructor for a rebalanced freight matrix that uses network assignment and traffic counts for matrix rebalancing.

Parameters

- **origins** – List of origin zones.
- **destinations** – List of destination zones.
- **rna** – Road network assignment.
- **rsg** – Route set generator.
- **params** – Properties.

RebalancedFreightMatrix

public **RebalancedFreightMatrix**(String *fileName*, *RoadNetworkAssignment* *rna*, *RouteSetGenerator* *rsg*, Properties *params*)

Constructor for a rebalanced freight matrix that uses network assignment and traffic counts for matrix rebalancing.

Parameters

- **fileName** – Path to the file with the initial OD matrix.
- **origins** – List of origin zones.
- **destinations** – List of destination zones.
- **rna** – Road network assignment.
- **rsg** – Route set generator.
- **params** – Properties.

Throws

- **IOException** – if any.
- **FileNotFoundException** – if any.

Methods

assignAndCalculateRMSN

public void **assignAndCalculateRMSN**()
Assigns OD matrix and calculates RMSN with traffic counts.

createUnitMatrix

public void **createUnitMatrix**()
Creates a unit OD matrix (all ones).

getRMSNvalues

public Map<*VehicleType*, List<Double>> **getRMSNvalues**()
Gets the list of RMSN values over all performed rebalancing iterations.

Returns List of RMSN values.

getScalingFactors

public *SkimMatrixFreightArray* **getScalingFactors**()
Calculates scaling factors for OD pairs.

Returns Scaling factors.

iterate

public void **iterate**(int *number*)
Iterates scaling to traffic counts.

Parameters

- **number** – Number of iterations.

scaleToTrafficCounts

public void **scaleToTrafficCounts**()
Scales OD matrix to traffic counts.

RebalancedODMatrix

public class **RebalancedODMatrix** extends *RealODMatrix*
Origin-destination matrix (LAD based) created by directly scaling flows using traffic counts.

Author Milan Lovric

Constructors

RebalancedODMatrix

public **RebalancedODMatrix**(List<String> *origins*, List<String> *destinations*, *RoadNetworkAssignment rna*, *RouteSetGenerator rsg*, Properties *params*)
Constructor for a rebalanced OD matrix that uses network assignment and traffic counts for matrix rebalancing.

Parameters

- **origins** – List of origin zones.
- **destinations** – List of destination zones.
- **rna** – Road network assignment.
- **rsg** – Route set generator.
- **params** – Properties.

Methods

assignAndCalculateRMSN

public void **assignAndCalculateRMSN**()
Assigns OD matrix and calculates RMSN with traffic counts.

createUnitMatrix

public void **createUnitMatrix**()
Creates a unit OD matrix (all ones).

getRMSNvalues

public List<Double> **getRMSNvalues**()
Gets the list of RMSN values over all performed rebalancing iterations.

Returns List of RMSN values.

getScalingFactors

public *RealODMatrix* **getScalingFactors**()
Calculates scaling factors for OD pairs.

Returns Scaling factors.

iterate

public void **iterate**(int *number*)
Iterates scaling to traffic counts.

Parameters

- **number** – Number of iterations.

scaleToTrafficCounts

public void **scaleToTrafficCounts** ()
Scales OD matrix to traffic counts.

RebalancedTemproODMatrix

public class **RebalancedTemproODMatrix** extends *RealODMatrixTempro*
Origin-destination matrix (Tempro based) created by directly scaling flows using traffic counts.

Author Milan Lovric

Constructors

RebalancedTemproODMatrix

public **RebalancedTemproODMatrix** (List<String> *origins*, List<String> *destinations*, *RoadNetworkAssignment* *rna*, *RouteSetGenerator* *rsg*, *Zoning* *zoning*, Properties *params*)
Constructor for a rebalanced OD matrix that uses network assignment and traffic counts for matrix rebalancing.

Parameters

- **origins** – List of origin zones.
- **destinations** – List of destination zones.
- **rna** – Road network assignment.
- **rsg** – Route set generator.
- **zoning** – Zoning system.
- **params** – Properties.

RebalancedTemproODMatrix

public **RebalancedTemproODMatrix** (String *fileName*, *RoadNetworkAssignment* *rna*, *RouteSetGenerator* *rsg*, *Zoning* *zoning*, Properties *params*)
Constructor for a rebalanced OD matrix that uses network assignment and traffic counts for matrix rebalancing.

Parameters

- **fileName** – Path to the file with the initial OD matrix.
- **rna** – Road network assignment.
- **rsg** – Route set generator.
- **zoning** – Zoning system.

Throws

- **IOException** – if any.
- **FileNotFoundException** – if any.

Methods

assignAndCalculateRMSN

public void **assignAndCalculateRMSN**()
Assigns OD matrix and calculates RMSN with traffic counts.

createUnitMatrix

public void **createUnitMatrix**()
Creates a unit OD matrix (all ones).

getRMSNvalues

public List<Double> **getRMSNvalues**()
Gets the list of RMSN values over all performed rebalancing iterations.

Returns List of RMSN values.

getScalingFactors

public *RealODMatrixTempro* **getScalingFactors**()
Calculates scaling factors for OD pairs.

Returns Scaling factors.

getSortedDestinations

public List<String> **getSortedDestinations**()
Gets the list of destinations.

Returns List of destinations.

getSortedOrigins

public List<String> **getSortedOrigins**()
Gets the list of origins.

Returns List of origins.

iterate

public void **iterate**(int *number*)
Iterates scaling to traffic counts.

Parameters

- **number** – Number of iterations.

scaleToTrafficCounts

public void **scaleToTrafficCounts**()
Scales OD matrix to traffic counts.

SkimMatrix

public interface **SkimMatrix**
Skim matrix for storing inter-zonal travel times or costs (for passenger vehicles).

Author Milan Lovric

Methods

getAbsoluteDifference

public double **getAbsoluteDifference**(*SkimMatrix other*)
Gets sum of absolute differences between elements of two matrices.

Parameters

- **other** – The other matrix.

Returns Sum of absolute differences.

getCost

public double **getCost**(String *originZone*, String *destinationZone*)
Gets cost for a given origin-destination pair using ONS codes.

Parameters

- **originZone** – Origin zone ONS code.
- **destinationZone** – Destination zone ONS code.

Returns Origin-destination cost.

getCost

public double **getCost**(int *originZoneID*, int *destinationZoneID*)
Gets cost for a given origin-destination pair using int zone IDs.

Parameters

- **originZone** – Origin zone ID.
- **destinationZone** – Destination zone ID.

Returns Origin-destination cost.

getSortedDestinations

public List<String> **getSortedDestinations**()
Gets the sroted list of destinations.

Returns List of destination zones.

getSortedOrigins

public List<String> **getSortedOrigins**()
Gets the sorted list of origins.

Returns List of origin zones.

getUnsortedDestinations

public List<String> **getUnsortedDestinations**()
Gets the unsorted list of destinations.

Returns List of destination zones.

getUnsortedOrigins

public List<String> **getUnsortedOrigins**()
Gets the unsorted list of origins.

Returns List of origin zones.

printMatrix

public void **printMatrix**()
Prints the matrix.

printMatrixFormatted

public void **printMatrixFormatted**()
Prints the matrix as a formatted table.

printMatrixFormatted

public void **printMatrixFormatted**(String *s*)
Prints the matrix as a formatted table, with a print message.

Parameters

- **s** – Print message

saveMatrixFormatted

public void **saveMatrixFormatted**(String *outputFile*)
Saves the matrix into a csv file (matrix format).

Parameters

- **outputFile** – Path to the output file.

saveMatrixFormattedList

public void **saveMatrixFormattedList**(String *outputFile*)
Saves the matrix into a csv file (list format).

Parameters

- **outputFile** – Path to the output file.

setCost

public void **setCost**(String *originZone*, String *destinationZone*, double *cost*)
Sets cost for a given origin-destination pair using ONS codes.

Parameters

- **originZone** – Origin zone.
- **destinationZone** – Destination zone.
- **cost** – Origin-destination cost.

setCost

public void **setCost**(int *originZoneID*, int *destinationZoneID*, double *cost*)
Sets cost for a given origin-destination pair using int zone IDs.

Parameters

- **originZoneID** – Origin zone ID.
- **destinationZoneID** – Destination zone ID.
- **cost** – Origin-destination cost.

SkimMatrixArray

public class **SkimMatrixArray** implements *SkimMatrix*
Skim matrix for storing inter-zonal travel times or costs (for passenger vehicles).

Author Milan Lovric

Constructors

SkimMatrixArray

public **SkimMatrixArray** (*Zoning zoning*)
Constructor for an empty skim matrix. Uses the maximum LAD ID.

Parameters

- **zoning** – Zoning system.

SkimMatrixArray

public **SkimMatrixArray** (String *fileName*, *Zoning zoning*)
Constructor that reads skim matrix from an input csv file. Can use both matrix and list format.

Parameters

- **fileName** – Path to the input file.
- **zoning** – Zoning system.

Throws

- **IOException** – if any.
- **FileNotFoundException** – if any.

Methods

getAbsoluteDifference

public double **getAbsoluteDifference** (*SkimMatrix other*)
Gets sum of absolute differences between elements of two matrices.

Parameters

- **other** – The other matrix.

Returns Sum of absolute differences.

getAverageCost

public double **getAverageCost** ()
Gets average OD cost.

Returns Average cost.

getAverageCost

public double **getAverageCost** (*ODMatrixMultiKey flows*)
Gets average OD cost weighted by demand.

Parameters

- **flows** – The demand as an origin-destination matrix.

Returns Average cost.

getCost

public double **getCost** (String *originZone*, String *destinationZone*)
Gets cost for a given origin-destination pair.

Parameters

- **originZone** – Origin zone.
- **destinationZone** – Destination zone.

Returns Origin-destination cost.

getCost

public double **getCost** (int *originZoneID*, int *destinationZoneID*)
Gets cost for a given origin-destination pair.

Parameters

- **originZoneID** – Origin zone ID.
- **destinationZoneID** – Destination zone ID.

Returns Origin-destination cost.

getSortedDestinations

public List<String> **getSortedDestinations** ()
Gets the sorted list of destinations.

Returns List of destinations.

getSortedOrigins

public List<String> **getSortedOrigins** ()
Gets the sorted list of origins.

Returns List of origins.

getSumOfCosts

public double **getSumOfCosts** ()
Gets sum of OD costs.

Returns Sum of costs.

getSumOfCosts

public double **getSumOfCosts**(*ODMatrixMultiKey flows*)
Gets sum of costs multiplied by demand flows.

Parameters

- **flows** – The demand as an origin-destination matrix.

Returns Sum of costs.

getUnsortedDestinations

public List<String> **getUnsortedDestinations**()
Gets the unsorted list of destinations.

Returns List of destinations.

getUnsortedOrigins

public List<String> **getUnsortedOrigins**()
Gets the unsorted list of origins.

Returns List of origins.

printMatrix

public void **printMatrix**()
Prints the matrix.

printMatrixFormatted

public void **printMatrixFormatted**()
Prints the matrix as a formatted table.

printMatrixFormatted

public void **printMatrixFormatted**(String *s*)
Prints the matrix as a formatted table, with a print message.

Parameters

- **s** – Print message

saveMatrixFormatted

public void **saveMatrixFormatted**(String *outputFile*)
Saves the matrix into a csv file.

Parameters

- **outputFile** – Path to the output file.

saveMatrixFormattedList

public void **saveMatrixFormattedList** (String *outputFile*)
Saves the matrix into a csv file. Uses a list format (origin, destination, cost).

Parameters

- **outputFile** – Path to the output file.

setCost

public void **setCost** (String *originZone*, String *destinationZone*, double *cost*)
Sets cost for a given origin-destination pair.

Parameters

- **originZone** – Origin zone.
- **destinationZone** – Destination zone.
- **cost** – Origin-destination cost.

setCost

public void **setCost** (int *originZoneID*, int *destinationZoneID*, double *cost*)
Sets cost for a given origin-destination pair.

Parameters

- **originZoneID** – Origin zone ID.
- **destinationZoneID** – Destination zone ID.
- **cost** – Origin-destination cost.

SkimMatrixArrayTempro

public class **SkimMatrixArrayTempro** implements *SkimMatrix*
Skim matrix for storing inter-zonal travel times or costs (for passenger vehicles).

Author Milan Lovric

Constructors

SkimMatrixArrayTempro

public **SkimMatrixArrayTempro** (*Zoning zoning*)
Constructor for an empty skim matrix. Uses the maximum Tempro ID.

Parameters

- **zoning** – Zoning system.

SkimMatrixArrayTempro

public **SkimMatrixArrayTempro**(String *fileName*, *Zoning zoning*)
Constructor that reads skim matrix from an input csv file. Can use both matrix and list format.

Parameters

- **fileName** – Path to the input file.
- **zoning** – Zoning system.

Throws

- **IOException** – if any.
- **FileNotFoundException** – if any.

Methods

getAbsoluteDifference

public double **getAbsoluteDifference**(*SkimMatrix other*)
Gets sum of absolute differences between elements of two matrices.

Parameters

- **other** – The other matrix.

Returns Sum of absolute differences.

getAverageCost

public double **getAverageCost**()
Gets average OD cost.

Returns Average cost.

getAverageCost

public double **getAverageCost**(*ODMatrixMultiKey flows*)
Gets average OD cost weighted by demand.

Parameters

- **flows** – The demand as an origin-destination matrix.

Returns Average cost.

getCost

public double **getCost**(String *originZone*, String *destinationZone*)
Gets cost for a given origin-destination pair.

Parameters

- **originZone** – Origin zone.

- **destinationZone** – Destination zone.

Returns Origin-destination cost.

getCost

public double **getCost**(int *originZoneID*, int *destinationZoneID*)
Gets cost for a given origin-destination pair.

Parameters

- **originZoneID** – Origin zone ID.
- **destinationZoneID** – Destination zone ID.

Returns Origin-destination cost.

getSortedDestinations

public List<String> **getSortedDestinations**()
Gets the sorted list of destinations.

Returns List of destinations.

getSortedOrigins

public List<String> **getSortedOrigins**()
Gets the sorted list of origins.

Returns List of origins.

getSumOfCosts

public double **getSumOfCosts**()
Gets sum of OD costs.

Returns Sum of costs.

getSumOfCosts

public double **getSumOfCosts**(*ODMatrixMultiKey flows*)
Gets sum of costs multiplied by demand flows.

Parameters

- **flows** – The demand as an origin-destination matrix.

Returns Sum of costs.

getUnsortedDestinations

public List<String> **getUnsortedDestinations**()
Gets the unsorted list of destinations.

Returns List of destinations.

getUnsortedOrigins

public List<String> **getUnsortedOrigins**()
Gets the unsorted list of origins.

Returns List of origins.

printMatrix

public void **printMatrix**()
Prints the matrix.

printMatrixFormatted

public void **printMatrixFormatted**()
Prints the matrix as a formatted table.

printMatrixFormatted

public void **printMatrixFormatted**(String *s*)
Prints the matrix as a formatted table, with a print message.

Parameters

- **s** – Print message

saveMatrixFormatted

public void **saveMatrixFormatted**(String *outputFile*)
Saves the matrix into a csv file.

Parameters

- **outputFile** – Path to the output file.

saveMatrixFormattedList

public void **saveMatrixFormattedList**(String *outputFile*)
Saves the matrix into a csv file. Uses a list format (origin, destination, cost).

Parameters

- **outputFile** – Path to the output file.

setCost

public void **setCost**(String *originZone*, String *destinationZone*, double *cost*)
Sets cost for a given origin-destination pair.

Parameters

- **originZone** – Origin zone.
- **destinationZone** – Destination zone.
- **cost** – Origin-destination cost.

setCost

public void **setCost**(int *originZoneID*, int *destinationZoneID*, double *cost*)
Sets cost for a given origin-destination pair.

Parameters

- **originZoneID** – Origin zone ID.
- **destinationZoneID** – Destination zone ID.
- **cost** – Origin-destination cost.

SkimMatrixFreight

public interface **SkimMatrixFreight**
Skim matrix for storing inter-zonal travel times or costs (for freight vehicles).

Author Milan Lovric

Fields

MAX_FREIGHT_ZONE_ID

public static final int **MAX_FREIGHT_ZONE_ID**

MAX_VEHICLE_ID

public static final int **MAX_VEHICLE_ID**

Methods

getAbsoluteDifference

public double **getAbsoluteDifference**(*SkimMatrixFreight other*)
Gets sum of absolute differences between elements of two matrices.

Parameters

- **other** – The other matrix.

Returns Sum of absolute differences.

getAverageCost

public double **getAverageCost**()
Gets average cost.

Returns Average cost.

getAverageCost

public double **getAverageCost**(*FreightMatrix demand*)
Gets average cost weighted by the freight demand.

Parameters

- **demand** – Freight OD matrix.

Returns Average cost weighted by freight matrix.

getCost

public double **getCost**(int *originZone*, int *destinationZone*, int *vehicleType*)
Gets cost for a given origin-destination pair and a vehicle type.

Parameters

- **originZone** – Origin zone.
- **destinationZone** – Destination zone.
- **vehicleType** – Vehicle type.

Returns Origin-destination cost.

printMatrix

public void **printMatrix**()
Prints the matrix.

printMatrixFormatted

public void **printMatrixFormatted**()
Prints the matrix as a formatted table.

printMatrixFormatted

public void **printMatrixFormatted**(String *s*)
Prints the matrix as a formatted table, with a print message.

Parameters

- **s** – Print message

saveMatrixFormatted

public void **saveMatrixFormatted**(String *outputFile*)
Saves the matrix into a csv file (list format for freight).

Parameters

- **outputFile** – Path to the output file.

setCost

public void **setCost**(int *originZone*, int *destinationZone*, int *vehicleType*, double *cost*)
Sets cost for a given origin-destination pair and a vehicle type.

Parameters

- **originZone** – Origin zone.
- **destinationZone** – Destination zone.
- **vehicleType** – Vehicle type.
- **cost** – Origin-destination cost.

SkimMatrixFreightArray

public class **SkimMatrixFreightArray** implements *SkimMatrixFreight*
Skim matrix for storing inter-zonal travel times or costs (for freight vehicles).

Author Milan Lovric

Constructors

SkimMatrixFreightArray

public **SkimMatrixFreightArray**()

SkimMatrixFreightArray

public **SkimMatrixFreightArray**(String *fileName*)
Constructor that reads freight skim matrix from an input csv file.

Parameters

- **fileName** – Path to the input file.

Throws

- **IOException** – if any.
- **FileNotFoundException** – if any.

Methods

getAbsoluteDifference

public double **getAbsoluteDifference** (*SkimMatrixFreight other*)
Gets sum of absolute differences between elements of two matrices.

Parameters

- **other** – The other matrix.

Returns Sum of absolute differences.

getAverageCost

public double **getAverageCost** ()
Gets average OD cost (ignores empty matrix cells).

Returns Average cost.

getAverageCost

public double **getAverageCost** (*FreightMatrix flows*)
Gets average OD cost weighted by demand.

Parameters

- **flows** – The demand as an origin-destination matrix.

Returns Average cost.

getCost

public double **getCost** (int *originZone*, int *destinationZone*, int *vehicleType*)
Gets cost for a given origin-destination pair and a vehicle type.

Parameters

- **originZone** – Origin zone.
- **destinationZone** – Destination zone.
- **vehicleType** – Vehicle type.

Returns Origin-destination cost.

printMatrix

public void **printMatrix** ()
Prints the matrix.

printMatrixFormatted

public void **printMatrixFormatted** ()
Prints the matrix as a formatted table.

printMatrixFormatted

public void **printMatrixFormatted**(String *s*)
Prints the matrix as a formatted table, with a print message.

Parameters

- **s** – Print message

saveMatrixFormatted

public void **saveMatrixFormatted**(String *outputFile*)
Saves the matrix into a csv file.

Parameters

- **outputFile** – Path to the output file.

setCost

public void **setCost**(int *originZone*, int *destinationZone*, int *vehicleType*, double *cost*)
Sets cost for a given origin-destination pair and a vehicle type.

Parameters

- **originZone** – Origin zone.
- **destinationZone** – Destination zone.
- **vehicleType** – Vehicle type.
- **cost** – Origin-destination cost.

SkimMatrixFreightMultiKey

public class **SkimMatrixFreightMultiKey** implements *SkimMatrixFreight*
Skim matrix for storing inter-zonal travel times or costs (for freight vehicles).

Author Milan Lovric

Constructors

SkimMatrixFreightMultiKey

public **SkimMatrixFreightMultiKey**()

SkimMatrixFreightMultiKey

public **SkimMatrixFreightMultiKey**(String *fileName*)
Constructor that reads freight skim matrix from an input csv file.

Parameters

- **fileName** – Path to the input file.

Throws

- **`IOException`** – if any.
- **`FileNotFoundException`** – if any.

Methods

getAbsoluteDifference

public double **`getAbsoluteDifference`**(*SkimMatrixFreight other*)
Gets sum of absolute differences between elements of two matrices.

Parameters

- **`other`** – The other matrix.

Returns Sum of absolute differences.

getAverageCost

public double **`getAverageCost`**()
Gets average OD cost (ignores empty matrix cells).

Returns Average cost.

getAverageCost

public double **`getAverageCost`**(*FreightMatrix flows*)
Gets average OD cost weighted by demand.

Parameters

- **`flows`** – The demand as an origin-destination matrix.

Returns Average cost.

getCost

public double **`getCost`**(int *originZone*, int *destinationZone*, int *vehicleType*)
Gets cost for a given origin-destination pair and a vehicle type.

Parameters

- **`originZone`** – Origin zone.
- **`destinationZone`** – Destination zone.
- **`vehicleType`** – Vehicle type.

Returns Origin-destination cost.

getKeySet

public Set<MultiKey> **getKeySet**()
Gets the keyset of the multimap.

Returns Keyset.

printMatrix

public void **printMatrix**()
Prints the matrix.

printMatrixFormatted

public void **printMatrixFormatted**()
Prints the matrix as a formatted table.

printMatrixFormatted

public void **printMatrixFormatted**(String *s*)
Prints the matrix as a formatted table, with a print message.

Parameters

- **s** – Print message

saveMatrixFormatted

public void **saveMatrixFormatted**(String *outputFile*)
Saves the matrix into a csv file.

Parameters

- **outputFile** – Path to the output file.

setCost

public void **setCost**(int *originZone*, int *destinationZone*, int *vehicleType*, double *cost*)
Sets cost for a given origin-destination pair and a vehicle type.

Parameters

- **originZone** – Origin zone.
- **destinationZone** – Destination zone.
- **vehicleType** – Vehicle type.
- **cost** – Origin-destination cost.

SkimMatrixMultiKey

public class **SkimMatrixMultiKey** implements *SkimMatrix*
Skim matrix for storing inter-zonal travel times or costs (for passenger vehicles).

Author Milan Lovric

Constructors

SkimMatrixMultiKey

public **SkimMatrixMultiKey** (*Zoning zoning*)
Skim matrix constructors.

Parameters

- **zoning** – Zoning system.

SkimMatrixMultiKey

public **SkimMatrixMultiKey** (String *fileName*, *Zoning zoning*)
Constructor that reads skim matrix from an input csv file. Can use both matrix and list format.

Parameters

- **fileName** – Path to the input file.
- **zoning** – Zoning system.

Throws

- **IOException** – if any.
- **FileNotFoundException** – if any.

Methods

getAbsoluteDifference

public double **getAbsoluteDifference** (*SkimMatrix other*)
Gets sum of absolute differences between elements of two matrices.

Parameters

- **other** – The other matrix.

Returns Sum of absolute differences.

getAverageCost

public double **getAverageCost** ()
Gets average OD cost.

Returns Average cost.

getAverageCost

public double **getAverageCost** (*ODMatrixMultiKey flows*)
Gets average OD cost weighted by demand.

Parameters

- **flows** – The demand as an origin-destination matrix.

Returns Average cost.

getAverageZonalCosts

public HashMap<String, Double> **getAverageZonalCosts** (List<String> *zones*)
Gets average zonal cost (used for the rail model).

Parameters

- **zones** – Zones for which zonal costs are required.

Returns Map of average zonal costs.

getAverageZonalCosts

public HashMap<String, Double> **getAverageZonalCosts** (List<String> *zones*, *ODMatrixMultiKey flows*)
Gets average zonal cost weighted by demand (used for the rail model).

Parameters

- **zones** – Zones for which zonal costs are required.
- **flows** – The demand as an origin-destination matrix.

Returns Map of average zonal costs.

getCost

public double **getCost** (String *originZone*, String *destinationZone*)
Gets cost for a given origin-destination pair.

Parameters

- **originZone** – Origin zone.
- **destinationZone** – Destination zone.

Returns Origin-destination cost.

getCost

public double **getCost** (int *originZoneID*, int *destinationZoneID*)
Gets cost for a given origin-destination pair.

Parameters

- **originZone** – Origin zone ID.

- **destinationZone** – Destination zone ID.

Returns Origin-destination cost.

getKeySet

public Set<MultiKey> **getKeySet**()
Gets the keyset of the multimap.

Returns Keyset.

getSortedDestinations

public List<String> **getSortedDestinations**()
Gets the sorted list of destinations.

Returns List of destinations.

getSortedOrigins

public List<String> **getSortedOrigins**()
Gets the sorted list of origins.

Returns List of origins.

getSumOfCosts

public double **getSumOfCosts**()
Gets sum of OD costs.

Returns Sum of costs.

getSumOfCosts

public double **getSumOfCosts**(*ODMatrixMultiKey flows*)
Gets sum of costs multiplied by demand flows.

Parameters

- **flows** – The demand as an origin-destination matrix.

Returns Sum of costs.

getUnsortedDestinations

public List<String> **getUnsortedDestinations**()
Gets the unsorted list of destinations.

Returns List of destinations.

getUnsortedOrigins

public List<String> **getUnsortedOrigins**()
Gets the unsorted list of origins.

Returns List of origins.

printMatrix

public void **printMatrix**()
Prints the matrix.

printMatrixFormatted

public void **printMatrixFormatted**()
Prints the matrix as a formatted table.

printMatrixFormatted

public void **printMatrixFormatted**(String *s*)
Prints the matrix as a formatted table, with a print message.

Parameters

- **s** – Print message

saveMatrixFormatted

public void **saveMatrixFormatted**(String *outputFile*)
Saves the matrix into a csv file.

Parameters

- **outputFile** – Path to the output file.

saveMatrixFormattedList

public void **saveMatrixFormattedList**(String *outputFile*)
Saves the matrix into a csv file. Uses a list format (origin, destination, cost).

Parameters

- **outputFile** – Path to the output file.

setCost

public void **setCost**(String *originZone*, String *destinationZone*, double *cost*)
Sets cost for a given origin-destination pair.

Parameters

- **originZone** – Origin zone.

- **destinationZone** – Destination zone.
- **cost** – Origin-destination cost.

setCost

public void **setCost**(int *originZoneID*, int *destinationZoneID*, double *cost*)
Sets cost for a given origin-destination pair.

Parameters

- **originZone** – Origin zone ID.
- **destinationZone** – Destination zone ID.
- **cost** – Origin-destination cost.

1.2.5 nismod.transport.disruption

Disruption

public abstract class **Disruption**
Abstract class for a disruption.

Author Milan Lovric

Fields

installed

protected boolean **installed**

props

protected Properties **props**

Constructors

Disruption

protected **Disruption**(Properties *props*)

Disruption

protected **Disruption**(String *fileName*)

Methods

getEndYear

public int **getEndYear**()

Returns The last year in which intervention still remains installed.

getProperty

public String **getProperty**(String *key*)

Parameters

- **key** – Name of the property

Returns Property

getStartYear

public int **getStartYear**()

Returns The year in which intervention is installed.

getState

public boolean **getState**()

Returns The state of the disruption (installed or not).

install

public abstract void **install**(Object *o*)

toString

public String **toString**()

uninstall

public abstract void **uninstall**(Object *o*)

RoadDisruption

public class **RoadDisruption** extends *Disruption*

Disruption on road links.

Author Milan Lovric

Constructors

RoadDisruption

public **RoadDisruption** (Properties *props*)
Constructor.

Parameters

- **props** – Properties of the road development intervention.

RoadDisruption

public **RoadDisruption** (String *fileName*)
Constructor.

Parameters

- **fileName** – File with the properties.

Methods

getListOfDisruptedEdgesIDs

public List<Edge> **getListOfDisruptedEdgesIDs** ()

Returns List of disrupted edge IDs.

getListOfRemovedRoutes

public List<*Route*> **getListOfRemovedRoutes** ()

Returns List of removed routes

install

public void **install** (Object *o*)

uninstall

public void **uninstall** (Object *o*)

1.2.6 nismod.transport.network.road

RoadNetwork

public class **RoadNetwork**
Routable road network built from the shapefiles.

Author Milan Lovric

Fields

averageSpeedFerry

public double **averageSpeedFerry**

freeFlowSpeedARoad

public double **freeFlowSpeedARoad**

freeFlowSpeedMRoad

public double **freeFlowSpeedMRoad**

maximumEdgeID

public int **maximumEdgeID**

maximumNodeID

public int **maximumNodeID**

numberOfLanesARoadCollapsedDualCarriageway

public int **numberOfLanesARoadCollapsedDualCarriageway**

numberOfLanesARoadDualCarriageway

public int **numberOfLanesARoadDualCarriageway**

numberOfLanesARoadRoundabout

public int **numberOfLanesARoadRoundabout**

numberOfLanesARoadSingleCarriageway

public int **numberOfLanesARoadSingleCarriageway**

numberOfLanesARoadSlipRoad

public int **numberOfLanesARoadSlipRoad**

numberOfLanesMRoadCollapsedDualCarriageway

public int **numberOfLanesMRoadCollapsedDualCarriageway**

numberOfLanesMRoadDualCarriageway

public int **numberOfLanesMRoadDualCarriageway**

numberOfLanesMRoadSlipRoad

public int **numberOfLanesMRoadSlipRoad**

Constructors

RoadNetwork

public **RoadNetwork**(URL *zonesUrl*, URL *networkUrl*, URL *nodesUrl*, URL *AADFurl*, String *areaCodeFileName*, String *areaCodeNearestNodeFile*, String *workplaceZoneFileName*, String *workplaceZoneNearestNodeFile*, String *freightZoneToLADfile*, String *freightZoneNearestNodeFile*, Properties *params*)

Parameters

- **zonesUrl** – Url for the shapefile with zone polygons.
- **networkUrl** – Url for the shapefile with road network.
- **nodesUrl** – Url for the shapefile with nodes.
- **AADFurl** – Url for the shapefile with AADF counts.
- **areaCodeFileName** – Path to the file with census output areas.
- **areaCodeNearestNodeFile** – Path to the file with nearest nodes to output area centroids.
- **workplaceZoneFileName** – Path to the file with workplace zones.
- **workplaceZoneNearestNodeFile** – Path to the file with nearest nodes to workplace zone centroids.
- **freightZoneToLADfile** – Path to the file with freight zone to LAD mapping.
- **freightZoneNearestNodeFile** – Path to the file with nearest nodes to freight zones that are points.
- **params** – Properties with parameters from the config file.

Throws

- **IOException** – if any.

Methods

addRoadLink

public void **addRoadLink**(Edge *edge*)

This adds edge (including its object) to the network - useful for restoring from a list of removed edges (e.g. during disruption).

Parameters

- **edge** – Edge to be added to the network.

createCustomFeatureType

public static SimpleFeatureType **createCustomFeatureType**(String *linkDataLabel*)

Creates a custom schema for the network.

Parameters

- **linkDataLabel** – The label for the link data (e.g. "DayVolume").

Returns SimpleFeature type.

createNetworkFeatureCollection

public SimpleFeatureCollection **createNetworkFeatureCollection**(Map<Integer, Double> *linkData*, String *linkDataLabel*, String *shapefilePath*)

Creates a custom feature collection for the network.

Parameters

- **linkData** – Data assigned to network links.
- **linkDataLabel** – The label of the link data.
- **shapefilePath** – The path to the shapefile into which data will be stored.

Throws

- **IOException** – if any.

Returns Feature collection.

createNewRoadLink

public Edge **createNewRoadLink**(Node *fromNode*, Node *toNode*, int *numberOfLanes*, char *roadCategory*, double *length*, int *edgeID*)

Creates a new (unidirectional) road link (edge) between existing intersections (nodes).

Parameters

- **fromNode** – Start node of the new road link.
- **toNode** – End node of the new road link.
- **numberOfLanes** – Number of lanes in the road link.
- **roadCategory** – Road category.

- **length** – Length of the road link.

Returns Newly created edge.

exportToShapefile

public void **exportToShapefile**(String *fileName*)
Exports a directed multigraph representation of the network as a shapefile.

Parameters

- **fileName** – The name of the output shapefile.

Throws

- **IOException** – if any.

getAADFCarTrafficCounts

public Integer[] **getAADFCarTrafficCounts**()
Get car traffic counts data for each link (for combined counts return 1/2 of the count per direction).

Returns AADF traffic counts per link.

getAADFFreightTrafficCounts

public Map<*VehicleType*, Integer[]> **getAADFFreightTrafficCounts**()
Get car traffic counts data for each link (for combined counts return 1/2 of the count per direction).

Returns AADF traffic counts per freight vehicle type and per link.

getAADFShapefile

public ShapefileDataStore **getAADFShapefile**()

getAreaCodeToNearestNodeID

public HashMap<String, Integer> **getAreaCodeToNearestNodeID**()
Getter method for the area code to the nearest node mapping.

Returns Area code to the nearest node mapping.

getAreaCodeToPopulation

public HashMap<String, Integer> **getAreaCodeToPopulation**()
Getter method for the area code to population mapping.

Returns Area code to population mapping.

getAstarFunctions

public AStarIterator.AStarFunctions **getAstarFunctions**(Node *destinationNode*)
Getter method for the AStar functions (edge cost and heuristic function) based on distance.

Parameters

- **destinationNode** – Destination node.

Returns AStar functions.

getAstarFunctionsTime

public AStarIterator.AStarFunctions **getAstarFunctionsTime**(Node *destinationNode*, double[] *linkTravelTime*)
Getter method for the AStar functions (edge cost and heuristic function) based on travel time.

Parameters

- **destinationNode** – Destination node.
- **linkTravelTime** – Link travel times to use for edge weighting.

Returns AStar functions.

getAverageAcessEgressDistance

public double **getAverageAcessEgressDistance**(int *node*)
Average access/egress distance to access a node that has gravitating population.

Parameters

- **node** – Node to which

Returns Gravitating population.

getAverageAcessEgressDistanceFreight

public double **getAverageAcessEgressDistanceFreight**(int *node*)
Average access/egress distance to access a node that has gravitating population.

Parameters

- **node** – Node to which

Returns Gravitating population.

getAverageSpeedFerry

public double **getAverageSpeedFerry**()

getDijkstraTimeWeighter

public DijkstraIterator.EdgeWeighter **getDijkstraTimeWeighter**(double[] *linkTravelTime*)
Getter method for the Dijkstra edge weighter with time.

Parameters

- **linkTravelTime** – Link travel times to use for edge weighting.

Returns Dijkstra edge weighter with time.

getDijkstraWeighter

public DijkstraIterator.EdgeWeighter **getDijkstraWeighter**()
Getter method for the Dijkstra edge weighter.

Returns Dijkstra edge weighter.

getEdgeIDtoEdge

public Edge[] **getEdgeIDtoEdge**()
Getter method for edgeID to edge mapping.

Returns Edge ID to edge mapping.

getEdgeIDtoOtherDirectionEdgeID

public Integer[] **getEdgeIDtoOtherDirectionEdgeID**()
Getter method for edgeID to other direction edgeID mapping.

Returns Edge ID to other direction edge ID mapping.

getEdgeLength

public double **getEdgeLength**(int *edgeID*)
Gets edge length for a given edge ID.

Parameters

- **edgeID** – Edge ID.

Returns Edge length.

getEdgeToZone

public HashMap<Integer, String> **getEdgeToZone**()
Getter method for the edge to zone mapping.

Returns Node to zone mapping.

getEdgesType

public *EdgeType*[] **getEdgesType**()

Getter method for the array saying if the edge is A-road, motorway, or ferry. Array index is edge ID (without -1 shift).

Returns Map between the edge ID and whether the edge is ferry.

getEndNodeBlacklist

public boolean[] **getEndNodeBlacklist**()

getFastestPath

public *RoadPath* **getFastestPath**(DirectedNode *from*, DirectedNode *to*, double[] *linkTravelTime*)

Gets the fastest path between two nodes using astar algorithm and provided link travel times. Links which have no travel time provided will use free flow travel times.

Parameters

- **from** – Origin node.
- **to** – Destination node.
- **linkTravelTime** – The map with link travel times.

Returns Fastest path.

getFastestPathDijkstra

public *RoadPath* **getFastestPathDijkstra**(DirectedNode *from*, DirectedNode *to*, double[] *linkTravelTime*)

Gets the fastest path between two nodes using Dijkstra's algorithm and provided link travel times. Links which have no travel time provided will use free flow travel times.

Parameters

- **from** – Origin node.
- **to** – Destination node.
- **linkTravelTime** – The map with link travel times.

Returns Fastest path.

getFreeFlowSpeedARoad

public double **getFreeFlowSpeedARoad**()

getFreeFlowSpeedMRoad

public double **getFreeFlowSpeedMRoad**()

getFreeFlowTravelTime

public double[] **getFreeFlowTravelTime**()
Getter method for free flow travel time.

Returns Free flow travel time.

getFreightZoneToLAD

public HashMap<Integer, String> **getFreightZoneToLAD**()
Getter method for the freight zone to LAD mapping.

Returns Area code to the nearest node mapping.

getFreightZoneToNearestNode

public HashMap<Integer, Integer> **getFreightZoneToNearestNode**()
Getter method for the freight zone to the nearest node mapping.

Returns Area code to the nearest node mapping.

getGravitatingPopulation

public int **getGravitatingPopulation**(int *node*)
Population gravitating to a node.

Parameters

- **node** – Node to which the population gravitates.

Returns Gravitating population.

getGravitatingWorkplacePopulation

public int **getGravitatingWorkplacePopulation**(int *node*)
Workplace population gravitating to a node.

Parameters

- **node** – Node to which the workplace population gravitates.

Returns Gravitating workplace population.

getIsEdgeUrban

public Boolean[] **getIsEdgeUrban**()
Getter method for the array saying if the edge is urban (true), rural (false), or unkown (null). Array index is edge ID (without -1 shift).

Returns Array saying if the edge is urban/rural/unkown.

getMaximumEdgeID

public int **getMaximumEdgeID**()
Getter method for maximum edge ID.

Returns Maximum edge ID.

getMaximumNodeID

public int **getMaximumNodeID**()
Getter method for maximum node ID.

Returns Maximum node ID.

getNetwork

public DirectedGraph **getNetwork**()
Getter method for the road network.

Returns Directed graph representation of the road network.

getNetworkShapefile

public ShapefileDataStore **getNetworkShapefile**()

getNewNetworkShapefile

public ShapefileDataStore **getNewNetworkShapefile**()

getNodeIDtoNode

public Node[] **getNodeIDtoNode**()
Getter method for nodeID to node mapping.

Returns Node ID to node mapping.

getNodeToAverageAccessEgressDistance

public double[] **getNodeToAverageAccessEgressDistance**()
Getter method for the node to average access/egress distance mapping [in metres].

Returns Node ID to average access/egress distance mapping.

getNodeToAverageAccessEgressDistanceFreight

public double[] **getNodeToAverageAccessEgressDistanceFreight**()
Getter method for the node to average access/egress distance mapping for freight [in metres].

Returns Node ID to average access/egress distance mapping for freight.

getNodeToGravitatingPopulation

public int[] **getNodeToGravitatingPopulation**()
Getter method for the node to gravitating population mapping.

Returns Node to gravitating population mapping.

getNodeToZone

public HashMap<Integer, String> **getNodeToZone**()
Getter method for the node to zone mapping.

Returns Node to zone mapping.

getNodesShapefile

public ShapefileDataStore **getNodesShapefile**()

getNumberOfLanes

public int[] **getNumberOfLanes**()
Getter method for the number of lanes for each link.

Returns Link id to number of lanes mapping.

getNumberOfLanesARoad

public int **getNumberOfLanesARoad**(String *wayType*)

getNumberOfLanesMRoad

public int **getNumberOfLanesMRoad**(String *wayType*)

getStartNodeBlacklist

public boolean[] **getStartNodeBlacklist**()

getWorkplaceCodeToPopulation

public HashMap<String, Integer> **getWorkplaceCodeToPopulation**()
Getter method for the workplace zone to population mapping.

Returns Workplace zone to population mapping.

getWorkplaceZoneToNearestNode

public HashMap<String, Integer> **getWorkplaceZoneToNearestNode**()
Getter method for the workplace code to the nearest node mapping.

Returns Workplace code to the nearest node mapping.

getZoneToAreaCodes

public HashMap<String, List<String>> **getZoneToAreaCodes**()
Getter method for the zone to area codes mapping.

Returns Zone to area codes mapping.

getZoneToNodes

public HashMap<String, List<Integer>> **getZoneToNodes**()
Getter method for the zone to nodes mapping.

Returns Zone to nodes mapping.

getZoneToWorkplaceCodes

public HashMap<String, List<String>> **getZoneToWorkplaceCodes**()
Getter method for the LAD zone to workplace zones mapping.

Returns Zone to workplace code mapping.

getZonesShapefile

public ShapefileDataStore **getZonesShapefile**()

isBlacklistedAsEndNode

public boolean **isBlacklistedAsEndNode**(int *nodeId*)
Finds out if the node is blacklisted as a path end node.

Parameters

- **nodeId** – Node ID.

Returns Whether nodes is blacklisted or not.

isBlacklistedAsStartNode

public boolean **isBlacklistedAsStartNode**(int *nodeId*)
Finds out if the node is blacklisted as a path start node.

Parameters

- **nodeId** – Node ID.

Returns Whether nodes is blacklisted or not.

makeEdgesAdmissible

public void **makeEdgesAdmissible**()
Overrides actual edge lengths with straight line distances, when they are smaller than straight line distances.

removeRoadLink

public void **removeRoadLink**(Edge *edge*)
Removes an edge from the road network.

Parameters

- **edge** – Edge to remove from the road network.

replaceNetworkEdgeIDs

public void **replaceNetworkEdgeIDs**(URL *networkShapeFile*)
Replaces edge IDs in the road network object with fixed edge IDs provided in a shapefile.

Parameters

- **networkShapeFile** – Path to the shapefile with the network with edge IDs.

Throws

- **IOException** – if any.

sortGravityNodes

public void **sortGravityNodes**()
For each zone (LAD) sorts the list of contained nodes based on the gravitating population.

sortGravityNodesFreight

public void **sortGravityNodesFreight**()
For each zone (LAD) sorts the list of contained nodes based on the gravitating workplace population.

toString

public String **toString**()

RoadNetwork.EdgeType

public static enum **EdgeType**

Enum Constants

AROAD

public static final *RoadNetwork.EdgeType* **AROAD**

FERRY

public static final *RoadNetwork.EdgeType* **FERRY**

MOTORWAY

public static final *RoadNetwork.EdgeType* **MOTORWAY**

RoadNetworkAssignment

public class **RoadNetworkAssignment**
Network assignment of origin-destination flows.

Author Milan Lovric

Fields

alpha

public final double **alpha**

assignmentFraction

public final double **assignmentFraction**

averageAccessEgressSpeedCar

public final double **averageAccessEgressSpeedCar**

averageAccessEgressSpeedFreight

public final double **averageAccessEgressSpeedFreight**

averageIntersectionDelay

public final double **averageIntersectionDelay**

baseYear

public final int **baseYear**

betaARoad

public final double **betaARoad**

betaMRoad

public final double **betaMRoad**

flagAStarIfEmptyRouteSet

public final boolean **flagAStarIfEmptyRouteSet**

flagIncludeAccessEgress

public final boolean **flagIncludeAccessEgress**

flagIntrazonalAssignmentReplacement

public final boolean **flagIntrazonalAssignmentReplacement**

flagUseRouteChoiceModel

public final boolean **flagUseRouteChoiceModel**

interzonalTopNodes

public final int **interzonalTopNodes**

maximumCapacityARoad

public final int **maximumCapacityARoad**

maximumCapacityMRoad

public final int **maximumCapacityMRoad**

nodesProbabilityWeighting

public final double **nodesProbabilityWeighting**

nodesProbabilityWeightingFreight

public final double **nodesProbabilityWeightingFreight**

peakHourPercentage

public final double **peakHourPercentage**

topTemproNodes

public final int **topTemproNodes**

volumeToFlowFactor

public final double **volumeToFlowFactor**

Constructors

RoadNetworkAssignment

public **RoadNetworkAssignment**(*RoadNetwork roadNetwork*, *Zoning zoning*, Map<*EnergyType*, Double> *energyUnitCosts*, Map<*EnergyType*, Double> *unitCO2Emissions*, Map<*VehicleType*, Map<*EngineType*, Double>> *engineTypeFractions*, Map<*VehicleType*, Double> *fractionsAV*, Map<*VehicleType*, Double> *vehicleTypeToPCU*, Map<*VehicleType*, Map<*EngineType*, Map<*WebTAG*, Double>>> *energyConsumptionParams*, Map<*VehicleType*, Map<*EngineType*, Double>> *relativeFuelEfficiencies*, Map<*TimeOfDay*, Double> *timeOfDayDistribution*, Map<*VehicleType*, Map<*TimeOfDay*, Double>> *timeOfDayDistributionFreight*, Map<*TimeOfDay*, Map<Integer, Double>> *defaultLinkTravelTime*, HashMap<String, Double> *areaCodeProbabilities*, HashMap<String, Double> *workplaceZoneProbabilities*, List<*PricingPolicy*> *congestionCharges*, Properties *params*)

Parameters

- **roadNetwork** – Road network.
- **zoning** – Zoning system.
- **energyUnitCosts** – Energy unit costs.
- **unitCO2Emissions** – Unit CO2 emissions.
- **engineTypeFractions** – Market shares of different engine/fuel types.
- **fractionsAV** – Fraction of autonomous vehicles for different vehicle types.
- **vehicleTypeToPCU** – Vehicle to PCU conversion.
- **energyConsumptionParams** – Base fuel consumption rates.
- **relativeFuelEfficiencies** – Relative fuel efficiencies (compared to base year).
- **timeOfDayDistribution** – Time of day distribution.
- **timeOfDayDistributionFreight** – Time of day distribution for freight.
- **defaultLinkTravelTime** – Default link travel times.
- **areaCodeProbabilities** – Probabilities of trips starting/ending in each census output area.
- **workplaceZoneProbabilities** – Probabilities of freight trips starting/ending in each census output area.
- **congestionCharges** – Congestion charges.

- **params** – Assignment parameters.

Methods

assignFlowsAndUpdateLinkTravelTimes

public void **assignFlowsAndUpdateLinkTravelTimes**(*AssignableODMatrix passengerODM*, *FreightMatrix freightODM*, *RouteSetGenerator rsg*, Properties *props*, double *weight*)

Assigns passenger and freight origin-destination matrix to the road network using the fastest path based on the current values in the linkTravelTime field. Finally, updates link travel times using weighted averaging.

Parameters

- **passengerODM** – Passenger origin-destination matrix.
- **freightODM** – Freight origin-destination matrix.
- **rsg** – Route set generator to store fastest routes generated during the assignment (but could be pregenerated too).
- **props** – Parameters from the config file.
- **weight** – Weighting parameter.

assignFlowsAndUpdateLinkTravelTimes

public void **assignFlowsAndUpdateLinkTravelTimes**(*AssignableODMatrix passengerODM*, *FreightMatrix freightODM*, *RouteSetGenerator rsg*, *Zoning zoning*, Properties *params*, double *weight*)

Assigns passenger and freight origin-destination matrix to the road network using specification in the config file. Finally, updates link travel times using weighted averaging.

Parameters

- **passengerODM** – Passenger origin-destination matrix.
- **freightODM** – Freight origin-destination matrix.
- **rsg** – Route set generator object with routes to be used for the assignment (if route choice used) or an object in which to store routes (if routing used).
- **zoning** – Zoning system (necessary for 'tempro' and 'combined' assignment types).
- **params** – Parameters from the config file.
- **weight** – Weighting parameter.

assignFlowsAndUpdateLinkTravelTimesIterated

public void **assignFlowsAndUpdateLinkTravelTimesIterated**(*AssignableODMatrix passengerODM*, *FreightMatrix freightODM*, *RouteSetGenerator rsg*, Properties *props*, double *weight*, int *iterations*)

Iterates assignment and travel time update a fixed number of times.

Parameters

- **passengerODM** – Passenger origin-destination matrix.
- **freightODM** – Freight origin-destination matrix.
- **rsg** – Route set generator object with routes to be used for the assignment (if route choice used) or an object in which to store routes (if routing used).
- **props** – Properties.
- **weight** – Weighting parameter.
- **iterations** – Number of iterations.

assignFlowsAndUpdateLinkTravelTimesIterated

public void **assignFlowsAndUpdateLinkTravelTimesIterated**(*AssignableODMatrix passengerODM*, *FreightMatrix freightODM*, *RouteSetGenerator rsg*, *Zoning zoning*, Properties *params*, double *weight*, int *iterations*)

Iterates assignment and travel time update a fixed number of times.

Parameters

- **passengerODM** – Passenger origin-destination matrix.
- **freightODM** – Freight origin-destination matrix.
- **rsg** – Route set generator object with routes to be used for the assignment (if route choice used) or an object in which to store routes (if routing used).
- **zoning** – Zoning system (necessary for 'tempro' and 'combined' assignment types).
- **params** – Parameters from the config file.
- **weight** – Weighting parameter.
- **iterations** – Number of iterations.

assignFreightFlowsHourlyRouting

public void **assignFreightFlowsHourlyRouting**(*FreightMatrix freightMatrix*, Map<*TimeOfDay*, *RouteSetGenerator*> *routeStorage*, Properties *props*)

Assigns freight origin-destination matrix to the road network using A-star routing. Zone ID ranges from the BYFM DfT model:

- England: 1 - 867
- Wales: 901 - 922
- Scotland: 1001 - 1032
- Freight airports: 1111 - 1115
- Major distribution centres: 1201 - 1256
- Freight ports: 1301 - 1388

Parameters

- **freightMatrix** – Freight origin-destination matrix.
- **routeStorage** – Route storage (stores fastest routes separately for each hour of the day).
- **props** – Properties.

assignFreightFlowsRouteChoice

public void **assignFreightFlowsRouteChoice**(*FreightMatrix freightMatrix*, *RouteSetGenerator rsg*, Properties *routeChoiceParameters*)

Assigns freight origin-destination matrix to the road network using a route choice model and pre-generated routes. Zone ID ranges from the BYFM DfT model:

- England: 1 - 867
- Wales: 901 - 922
- Scotland: 1001 - 1032
- Freight airports: 1111 - 1115
- Major distribution centres: 1201 - 1256
- Freight ports: 1301 - 1388

Parameters

- **freightMatrix** – Freight origin-destination matrix.
- **rsg** – Route set generator containing the routes.
- **routeChoiceParameters** – Route choice parameters.

assignFreightFlowsRouting

public void **assignFreightFlowsRouting**(*FreightMatrix freightMatrix*, *RouteSetGenerator rsg*, Properties *props*)

Assigns freight origin-destination matrix to the road network using A-star routing. Zone ID ranges from the BYFM DfT model:

- England: 1 - 867
- Wales: 901 - 922
- Scotland: 1001 - 1032
- Freight airports: 1111 - 1115
- Major distribution centres: 1201 - 1256
- Freight ports: 1301 - 1388

Parameters

- **freightMatrix** – Freight origin-destination matrix.
- **rsg** – Route storage (reduces the number of routing calls).
- **props** – Properties.

assignPassengerFlowsHourlyRouting

public void **assignPassengerFlowsHourlyRouting**(*AssignableODMatrix passengerODM*, Map<*TimeOfDay*, *RouteSetGenerator*> *routeStorage*, Properties *props*)

Assigns passenger origin-destination matrix to the road network using A-star routing algorithm. Calculates the fastest path based on the current values in the linkTravlinkTravelTimePerTimeOfDayelTime instance field, which means different fastest routes may be used in different hours of the day.

Parameters

- **passengerODM** – Passenger origin-destination matrix with flows to be assigned.
- **routeStorage** – Stores routes for each hour of the day separately.
- **props** – Properties.

assignPassengerFlowsRouteChoice

public void **assignPassengerFlowsRouteChoice**(*AssignableODMatrix passengerODM*, *RouteSetGenerator rsg*, Properties *routeChoiceParameters*)

Assigns passenger origin-destination matrix to the road network. Uses the route choice and pre-generated paths.

Parameters

- **passengerODM** – Passenger origin-destination matrix.
- **rsg** – Route set generator containing the routes.
- **routeChoiceParameters** – Route choice parameters.

assignPassengerFlowsRouteChoiceTempro

public void **assignPassengerFlowsRouteChoiceTempro**(*AssignableODMatrix passengerODM*, *Zoning zoning*, *RouteSetGenerator rsg*, Properties *routeChoiceParameters*)

Assigns passenger origin-destination matrix to the road network using the Tempro zoning system. Uses the route choice and pre-generated paths.

Parameters

- **passengerODM** – Passenger origin-destination matrix with flows to be assigned.
- **zoning** – Contains Tempro zone information.
- **rsg** – Route set generator containing the routes.
- **routeChoiceParameters** – Route choice parameters.

assignPassengerFlowsRouteChoiceTemproDistanceBased

public void **assignPassengerFlowsRouteChoiceTemproDistanceBased**(*AssignableODMatrix passengerODM*, *Zoning zoning*, *RouteSetGenerator rsg*, Properties *routeChoiceParameters*)

Assigns passenger origin-destination matrix to the road network using the combined Tempro/LAD zoning sys-

tem. When Tempro zones a farther than a distance threshold, it seeks for the nodes within LAD zones that have a route set. Uses the route choice and pre-generated paths (after a distance threshold, there will be less inter-zonal routes).

Parameters

- **passengerODM** – Passenger origin-destination matrix with flows to be assigned.
- **zoning** – Contains Tempro zone information.
- **rsg** – Route set generator containing the routes.
- **routeChoiceParameters** – Route choice parameters.

assignPassengerFlowsRouting

public void **assignPassengerFlowsRouting**(*AssignableODMatrix passengerODM*, *RouteSetGenerator rsg*, Properties *props*)

Assigns passenger origin-destination matrix to the road network using A-star routing algorithm. Calculates the fastest path based on the current values in the linkTravlinkTravelTimePerTimeOfDayelTime instance field, however only one route will be used for the same OD pair (the route that was calculated first).

Parameters

- **passengerODM** – Passenger origin-destination matrix with flows to be assigned.
- **rsg** – To store routes during the assignment (reduces the number of routing calls).
- **props** – Routing parameters.

assignPassengerFlowsTempro

public void **assignPassengerFlowsTempro**(*AssignableODMatrix passengerODM*, *Zoning zoning*, *RouteSetGenerator rsg*, Properties *props*)

Assigns passenger origin-destination matrix to the road network using the Tempro zoning system. Calculates the fastest path based on the current values in the linkTravelTime instance field.

Parameters

- **passengerODM** – Passenger origin-destination matrix with flows to be assigned.
- **zoning** – Contains Tempro zone information.
- **rsg** – Route set (here new routes will be stored).
- **props** – Properties.

calculateAbsDifferenceCarCounts

public HashMap<Integer, Integer> **calculateAbsDifferenceCarCounts**()

Calculates absolute differences between car volumes and traffic counts. For combined counts, takes the average of two absolute differences.

Returns Absolute differences between car volumes and traffic counts.

calculateAssignedFreightMatrix

public *FreightMatrix* **calculateAssignedFreightMatrix**()
Calculate freight OD matrix from trip list.

Returns Freight matrix.

calculateAssignedODMatrix

public *ODMatrixMultiKey* **calculateAssignedODMatrix**()
Calculate assigned OD matrix from trip list.

Returns ODMatrixMultiKey OD matrix.

calculateCO2Emissions

public HashMap<String, Double> **calculateCO2Emissions**()
Calculates total CO2 emissions (in kg) for each type of passenger and freight vehicle.

Returns Total consumption for each engine type.

calculateCarEnergyConsumptions

public Map<*EnergyType*, Double> **calculateCarEnergyConsumptions**()
Calculates total energy consumption for each car/AV energy type (in litres for fuels and in kWh for electricity).

Returns Total consumption for each energy type.

calculateCostSkimMatrix

public *SkimMatrix* **calculateCostSkimMatrix**()
Calculates cost skim matrix (zone-to-zone financial costs).

Returns Inter-zonal skim matrix (cost).

calculateCostSkimMatrixFreight

public *SkimMatrixFreight* **calculateCostSkimMatrixFreight**()
Calculates cost skim matrix (zone-to-zone financial costs) for freight.

Returns Inter-zonal skim matrix (cost).

calculateDifferenceCarCounts

public HashMap<Integer, Integer> **calculateDifferenceCarCounts**()
Calculates differences between car volumes and traffic counts. For combined counts, takes the average of the two differences.

Returns Differences between car volumes and traffic counts.

calculateDirectionAveragedAbsoluteDifferenceCarCounts

public HashMap<Integer, Double> **calculateDirectionAveragedAbsoluteDifferenceCarCounts**()
Calculates absolute differences between car volumes and traffic counts averaged for both directions. For combined counts, takes the average of two absolute differences.

Returns Direction averaged absolute differences between car volumes and traffic counts.

calculateDirectionAveragedPeakLinkCapacityUtilisation

public double[] **calculateDirectionAveragedPeakLinkCapacityUtilisation**()
Calculate peak-hour link capacity utilisation (%) averaged by two directions.

Returns Peak-hour link capacity utilisation.

calculateDistanceSkimMatrix

public *SkimMatrix* **calculateDistanceSkimMatrix**()
Updates cost skim matrix (zone-to-zone distances).

Returns Inter-zonal skim matrix (distance).

calculateDistanceSkimMatrixFreight

public *SkimMatrixFreight* **calculateDistanceSkimMatrixFreight**()
Updates cost skim matrix (zone-to-zone distances) for freight.

Returns Inter-zonal skim matrix (distance).

calculateDistanceSkimMatrixTempro

public *SkimMatrix* **calculateDistanceSkimMatrixTempro**()
Updates cost skim matrix (zone-to-zone distances).

Parameters

- **zoning** – Zoning system.

Returns Inter-zonal skim matrix (distance).

calculateEnergyConsumptions

public Map<*EnergyType*, Double> **calculateEnergyConsumptions**()
Calculates total energy consumption for each energy type of passenger cars and freight vehicles (in litres for fuels and in kWh for electricity).

Returns Total consumption for each engine type.

calculateEnergyConsumptionsPerVehicleType

public Map<*VehicleType*, Map<*EnergyType*, Double>> **calculateEnergyConsumptionsPerVehicleType**()
Calculates total energy consumptions per vehicle type.

Returns Total consumption for each energy type.

calculateFreightEnergyConsumptions

public Map<*EnergyType*, Double> **calculateFreightEnergyConsumptions**()
Calculates total energy consumption for each freight vehicle engine type (in litres for fuels and in kWh for electricity).

Returns Total consumption for each energy type.

calculateFreightLADTripEnds

public HashMap<String, Integer> **calculateFreightLADTripEnds**()
Calculates the number of freight trips ending in a LAD.

Returns Number of trips.

calculateFreightLADTripStarts

public HashMap<String, Integer> **calculateFreightLADTripStarts**()
Calculates the number of freight trips starting in a LAD.

Returns Number of trips.

calculateGEHStatisticForCarCounts

public Double[] **calculateGEHStatisticForCarCounts**(double *volumeToFlowFactor*)
Calculates GEH statistic for simulated and observed hourly car flows. For combined counts, combines the volumes on two road directions. Two obtain hourly flows, multiplies daily link volumes (and traffic counts) with volumeToFlowFactor. The formula is taken from WebTAG Unit M3.1.

Parameters

- **volumeToFlowFactor** – Converts daily vehicle volume to hourly flow (e.g. 0.1 for peak flow; 1/24.0 for daily average)

Returns GEH statistic for simulated and observed hourly car flows.

calculateGEHStatisticForFreightCounts

public Map<*VehicleType*, Double[]> **calculateGEHStatisticForFreightCounts**(double *volumeToFlowFactor*)
Calculates GEH statistic for simulated and observed hourly freight vehicle flows. For combined counts, combines the volumes on two road directions. Two obtain hourly flows, multiplies daily link volumes (and traffic counts) with volumeToFlowFactor. The formula is taken from WebTAG Unit M3.1.

Parameters

- **volumeToFlowFactor** – Converts daily vehicle volume to hourly flow (e.g. 0.1 for peak flow; 1/24.0 for daily average)

Returns GEH statistic for simulated and observed hourly freight vehicle flows, per vehicle type.

calculateGEHStatisticPerTimeOfDay

public Double[] **calculateGEHStatisticPerTimeOfDay** (*TimeOfDay hour*)

Calculates GEH statistic for simulated and observed hourly flow. It uses linkVolumesInPCUPerTimeOfDay, so make sure only car flows have been assigned. For combined counts, takes the average of the two differences. The formula is taken from WebTAG Unit M3.1.

Parameters

- **hour** – Hour for which to calculate GEH statistics.

Returns GEH statistic for simulated and observed hourly car flows.

calculateLADTripEnds

public HashMap<String, Integer> **calculateLADTripEnds** ()

Calculates the number of passenger (car/AV) trips ending in a LAD.

Returns Number of trips.

calculateLADTripStarts

public HashMap<String, Integer> **calculateLADTripStarts** ()

Calculates the number of passenger (car/AV) trips starting in a LAD.

Returns Number of trips.

calculateLinkVolumeInPCU

public double[] **calculateLinkVolumeInPCU** (List<*Trip*> *tripList*)

Calculates daily link volumes in PCU.

Parameters

- **tripList** – Trip list.

Returns Map of link volumes in PCU.

calculateLinkVolumeInPCUPerTimeOfDay

public Map<*TimeOfDay*, double[]> **calculateLinkVolumeInPCUPerTimeOfDay** (List<*Trip*> *tripList*)

Calculates link volumes in PCU per time of day.

Parameters

- **tripList** – Trip list.

Returns Link volumes in PCU per time of day

calculateLinkVolumePerVehicleType

public Map<*VehicleType*, int[]> **calculateLinkVolumePerVehicleType** (List<*Trip*> *tripList*)
Calculates daily link volumes per vehicle type.

Parameters

- **tripList** – Trip list.

Returns Map of link volumes per vehicle type.

calculateMADforExpandedSimulatedVolumes

public double **calculateMADforExpandedSimulatedVolumes** (double *expansionFactor*)
Calculate prediction error (mean absolute deviation for expanded simulated volumes and observed traffic counts).

Parameters

- **expansionFactor** – Expansion factor expands simulated volumes.

Returns Mean absolute deviation.

calculateODCarEnergyConsumptions

public Map<*EnergyType*, *SkimMatrix*> **calculateODCarEnergyConsumptions** ()
Calculates origin-destination energy consumption for car vehicles for each energy type (in litres/kg for fuels and in kWh for electricity).

Returns Zonal consumption for each energy type.

calculatePeakLinkCapacityUtilisation

public double[] **calculatePeakLinkCapacityUtilisation** ()
Calculate peak-hour link capacity utilisation (capacity / max. capacity).

Returns Peak-hour link capacity utilisation [%].

calculatePeakLinkDensities

public double[] **calculatePeakLinkDensities** ()
Calculate peak-hour link densities (PCU/lane/km/hr).

Returns Peak-hour link densities.

calculatePeakLinkPointCapacities

public double[] **calculatePeakLinkPointCapacities** ()
Calculate peak-hour link point capacities (PCU/lane/hr).

Returns Peak-hour link point capacities.

calculateRMSNforExpandedSimulatedVolumes

public double **calculateRMSNforExpandedSimulatedVolumes**(double *expansionFactor*)
Calculate prediction error (RMSN for expanded simulated volumes and observed traffic counts).

Parameters

- **expansionFactor** – Expansion factor expands simulated volumes.

Returns Normalised root mean square error.

calculateRMSNforFreightCounts

public Map<*VehicleType*, Double> **calculateRMSNforFreightCounts**()
Calculate prediction error (RMSN for for simulated freight volumes and observed traffic counts).

Returns Normalised root mean square errors for each freight vehicle separately.

calculateRMSNforSimulatedVolumes

public double **calculateRMSNforSimulatedVolumes**()
Calculate prediction error (RMSN for simulated volumes and observed traffic counts).

Returns Normalised root mean square error.

calculateTimeSkimMatrix

public *SkimMatrix* **calculateTimeSkimMatrix**()
Calculated travel time skim matrix (zone-to-zone travel times).

Returns Inter-zonal skim matrix (time).

calculateTimeSkimMatrixFreight

public *SkimMatrixFreight* **calculateTimeSkimMatrixFreight**()
Calculated travel time skim matrix (zone-to-zone travel times) for freight.

Returns Inter-zonal skim matrix (time).

calculateZonalCarEnergyConsumptions

public Map<*EnergyType*, HashMap<String, Double>> **calculateZonalCarEnergyConsumptions**(double *originZoneEnergyWeight*)
Calculates spatial energy consumption for car vehicles for each energy type (in litres/kg for fuels and in kWh for electricity).

Parameters

- **originZoneEnergyWeight** – Percentage of energy consumption assigned to origin zone (the rest assigned to destination zone).

Returns Zonal consumption for each energy type.

calculateZonalTemporalTripStartsForElectricVehicles

public HashMap<String, Map<*TimeOfDay*, Integer>> **calculateZonalTemporalTripStartsForElectricVehicles** (*Vehicle vht*

Calculates the number of electric vehicles (BEV, PHEV) of a given type (CAR, VAN, RIGID, ARTIC), starting in each LAD in each hour.

Parameters

- **vht** – Vehicle type (calculation will include the autonomous version of the same vehicle type too).

Returns Number of trips.

calculateZonalTemporalTripStartsForHydrogenVehicles

public HashMap<String, Map<*TimeOfDay*, Integer>> **calculateZonalTemporalTripStartsForHydrogenVehicles** (*Vehicle vht*

Calculates the number of hydrogen vehicles (ICE_H2, FCEV_H2) of a given type (CAR, VAN, RIGID, ARTIC), starting in each LAD in each hour.

Parameters

- **vht** – Vehicle type (calculation will include the autonomous version of the same vehicle type too).

Returns Number of trips.

calculateZonalTemporalVehicleElectricityConsumptions

public HashMap<String, Map<*TimeOfDay*, Double>> **calculateZonalTemporalVehicleElectricityConsumptions** (*V v d b o i Z e g W*

Calculates zonal (per LAD) and temporal (per hour) electricity consumption for a given vehicle type (in kWh).

Parameters

- **vht** – Vehicle type (e.g., if CAR provided, CAR_AV consumption will be added too).
- **originZoneEnergyWeight** – Percentage of energy consumption assigned to origin zone (the rest assigned to destination zone).

Returns Electricity consumption per zone and time of day.

calculateZonalTemporalVehicleHydrogenConsumptions

public HashMap<String, Map<*TimeOfDay*, Double>> **calculateZonalTemporalVehicleHydrogenConsumptions**(*Vehic*
vht,
dou-
ble
orig-
in-
Zonel
er-
gy-
Weigl

Calculates zonal (per LAD) and temporal (per hour) hydrogen consumption for a given vehicle type (in kg).

Parameters

- **vht** – Vehicle type (e.g., if CAR provided, CAR_AV consumption will be added too).
- **originZoneEnergyWeight** – Percentage of energy consumption assigned to origin zone (the rest assigned to destination zone).

Returns Electricity consumption per zone and time of day.

calculateZonalVehicleCO2Emissions

public Map<*VehicleType*, HashMap<String, Double>> **calculateZonalVehicleCO2Emissions**(double
orig-
in-
ZoneEn-
er-
gy-
Weight)

Calculates zonal CO2 emissions (in kg) for each vehicle type (sum across all engine types).

Parameters

- **originZoneEnergyWeight** – Percentage of CO2 emission assigned to origin zone (the rest assigned to destination zone).

Returns Zonal CO2 emissions for each vehicle type.

calculateZonalVehicleKilometresPerVehicleType

public Map<String, Map<*VehicleType*, Double>> **calculateZonalVehicleKilometresPerVehicleType**()

Calculates vehicle kilometres in each LAD and for each vehicle type. Ignores access and egress to major roads. Ignores minor roads.

Returns Vehicle kilometres.

calculateZonalVehicleKilometresPerVehicleTypeFromTemproTripList

public Map<String, Map<*VehicleType*, Double>> **calculateZonalVehicleKilometresPerVehicleTypeFromTemproTr**

Calculates vehicle kilometres in each LAD using Tempro-based trips. Optionally includes access and egress (for Tempro-based model). Optionally includes minor trips (Tempro intra-zonal).

Parameters

- **includeAccessEgress** – True if access and egress should be included in the calculation.
- **includeMinorTrips** – True if minor trips should be included in the calculation.

Returns Vehicle kilometres.

calculateZonalVehicleKilometresPerVehicleTypeFromTripList

public Map<String, Map<*VehicleType*, Double>> **calculateZonalVehicleKilometresPerVehicleTypeFromTripList**

Calculates vehicle kilometres in each LAD and per vehicle type. Optionally includes access and egress (for LAD-based model).

Parameters

- **includeAccessEgress** – True if access and egress should be included in the calculation.

Returns Vehicle kilometres.

getAADFCarTrafficCounts

public Integer[] **getAADFCarTrafficCounts**()
Getter method for AADF car traffic counts.

Returns Car traffic counts.

getAADFFreightTrafficCounts

public Map<*VehicleType*, Integer[]> **getAADFFreightTrafficCounts**()
Getter method for AADF freight traffic counts.

Returns Freight traffic counts.

getAreaCodeProbabilities

public HashMap<String, Double> **getAreaCodeProbabilities**()
Getter method for output area probabilities.

Returns Output area probabilities.

getCopyOfLinkTravelTimes

public Map<*TimeOfDay*, double[]> **getCopyOfLinkTravelTimes**()

Returns The copy of all link travel times.

getCopyOfLinkTravelTimesAsMap

public Map<*TimeOfDay*, Map<Integer, Double>> **getCopyOfLinkTravelTimesAsMap**()

Returns The copy of all link travel times as map.

getEndNodeProbabilities

public HashMap<Integer, Double> **getEndNodeProbabilities**()
Getter method for node probabilities.

Returns Node probabilities.

getEnergyConsumptionParameters

public Map<*VehicleType*, Map<*EngineType*, Map<*WebTAG*, Double>>> **getEnergyConsumptionParameters**()
Getter method for energy consumption WebTAG parameters.

Returns Energy consumption parameters.

getEnergyUnitCosts

public Map<*EnergyType*, Double> **getEnergyUnitCosts**()
Getter method for energy unit costs.

Returns Energy unit costs.

getEngineTypeFractions

public Map<*VehicleType*, Map<*EngineType*, Double>> **getEngineTypeFractions**()
Getter method for engine type fractions.

Returns Engine type fractions.

getFlagUseRouteChoiceModel

public boolean **getFlagUseRouteChoiceModel**()
Getter method for the use route choice model flag.

Returns Flag.

getLinkFreeFlowTravelTimes

public double[] **getLinkFreeFlowTravelTimes**()
Getter method for the link free-flow travel times.

Returns Link volumes

getLinkTravelTimes

public Map<*TimeOfDay*, double[]> **getLinkTravelTimes**()
Getter method for the link travel times per time of day.

Returns Link travel times per time of day.

getLinkVolumeInPCU

public double[] **getLinkVolumeInPCU**()
Getter method for daily link volumes in PCU.

Returns Link volumes in PCU.

getLinkVolumeInPCUPerTimeOfDay

public Map<*TimeOfDay*, double[]> **getLinkVolumeInPCUPerTimeOfDay**()
Getter method for link volumes in PCU per time of day.

Returns Link volumes in PCU per time of day.

getLinkVolumePerVehicleType

public Map<*VehicleType*, int[]> **getLinkVolumePerVehicleType**()
Getter method for daily link volumes per vehicle type.

Returns Link volumes in PCU per time of day.

getObservedTripLengthDistribution

public double[] **getObservedTripLengthDistribution**(double[] *binLimitsInKm*, boolean *flagIncludeAccessEgress*, boolean *flagIncludeMinorTrips*)
Calculates observed trip length distribution.

Parameters

- **binLimitsInKm** – Bin limits in kilometres.

- **flagIncludeAccessEgress** – If true include access and eggress into trip distance calculation. * @param flagIncludeMinorTrips If true include minor road trips into trip distance calculation.

Returns Observed trip length distribution.

getObservedTripLengthFrequencies

public double[] **getObservedTripLengthFrequencies**(double[] *binLimitsInKm*, boolean *flagIncludeAccessEgress*, boolean *flagIncludeMinorTrips*)

Calculates observed trip length frequencies.

Parameters

- **binLimitsInKm** – Bin limits in kilometres.
- **flagIncludeAccessEgress** – If true include access and egress into trip distance calculation.
- **flagIncludeMinorTrips** – If true include minor road trips into trip distance calculation.

Returns Observed trip length distribution.

getRoadNetwork

public *RoadNetwork* **getRoadNetwork**()

Getter method for the road network.

Returns Road network.

getStartNodeProbabilities

public HashMap<Integer, Double> **getStartNodeProbabilities**()

Getter method for node probabilities.

Returns Node probabilities.

getTripList

public ArrayList<*Trip*> **getTripList**()

Getter method for the trip list.

Returns Trip list.

getVolumeToFlowFactor

public double **getVolumeToFlowFactor**()

Getter method for the volume to flow factor.

Returns Volume to flow factor.

getWorkplaceZoneProbabilities

public HashMap<String, Double> **getWorkplaceZoneProbabilities**()
Getter method for workplace zones probabilities.

Returns Workplace zones probabilities.

initialiseTripList

public void **initialiseTripList**(int *initialCapacity*)
Initialise trip list for passengers and freight (e.g. expected total sum or passenger and freight flows).

Parameters

- **initialCapacity** – Initial capacity of the trip list.

loadLinkTravelTimes

public void **loadLinkTravelTimes**(int *year*, String *fileName*)
Loads link travel times from a file.

Parameters

- **year** – Year of the assignment.
- **fileName** – Input file name (with path).

printGEHstatistic

public void **printGEHstatistic**()
Prints GEH statistics for comparison between simulated and observed hourly car flows.

printGEHstatistic

public void **printGEHstatistic**(double *volumeToFlowFactor*)
Prints GEH statistics for comparison between simulated and observed hourly car flows.

Parameters

- **volumeToFlowFactor** – Converts daily vehicle volume to hourly flow (e.g. 0.1 for peak flow; 1/24.0 for daily average)

printGEHstatisticFreight

public void **printGEHstatisticFreight**()
Prints GEH statistics for comparison between simulated and observed hourly freight vehicle flows.

printGEHstatisticFreight

public void **printGEHstatisticFreight**(double *volumeToFlowFactor*)
Prints GEH statistics for comparison between simulated and observed hourly freight vehicle flows.

Parameters

- **volumeToFlowFactor** – Converts daily vehicle volume to hourly flow (e.g. 0.1 for peak flow; 1/24.0 for daily average)

printHourlyGEHstatistic

public void **printHourlyGEHstatistic**()
Prints GEH statistics for comparison between simulated and observed hourly car flows.

printRMSNstatistic

public void **printRMSNstatistic**()
Prints RMSN statistic for comparison between simulated daily car volumes and observed daily traffic counts.

printRMSNstatisticFreight

public void **printRMSNstatisticFreight**()
Prints RMSN statistic for comparison between simulated daily freight volumes and observed daily freight traffic counts.

resetLinkVolumes

public void **resetLinkVolumes**()
Resets link volumes to zero.

resetTripList

public void **resetTripList**()
Reset trip list for passengers and freight.

saveAssignmentResults

public void **saveAssignmentResults**(int *year*, String *outputFile*)
Saves assignment results to output file.

Parameters

- **year** – Year of the assignment.
- **outputFile** – Output file name (with path).

saveEnergyConsumptionsPerVehicleType

public void **saveEnergyConsumptionsPerVehicleType**(int *year*, String *outputFile*)
Saves energy consumptions per vehicle type to an output file.

Parameters

- **year** – Assignment year.

- **outputFile** – Output file name (with path).

saveHourlyCarVolumes

public void **saveHourlyCarVolumes** (int *year*, String *outputFile*)
Saves hourly car volumes to output file.

Parameters

- **year** – Year of the assignment.
- **outputFile** – Output file name (with path).

saveLinkTravelTimes

public void **saveLinkTravelTimes** (int *year*, String *outputFile*)
Saves travel times into a file.

Parameters

- **year** – Year of the assignment.
- **outputFile** – Output file name (with path).

saveOriginDestinationCarElectricityConsumption

public void **saveOriginDestinationCarElectricityConsumption** (String *outputFile*)
Saves origin-destination matrix of car electricity consumption.

Parameters

- **outputFile** – Output file name (with path).

savePeakLinkPointCapacities

public void **savePeakLinkPointCapacities** (int *year*, String *outputFile*)
Saves peak link point capacities into a file.

Parameters

- **year** – Year of the assignment.
- **outputFile** – Output file name (with path).

saveTotalCO2Emissions

public void **saveTotalCO2Emissions** (int *year*, String *outputFile*)
Saves total CO2 emissions to an output file.

Parameters

- **year** – Year of the assignment.
- **outputFile** – Output file name (with path).

saveTotalEnergyConsumptions

public void **saveTotalEnergyConsumptions** (int *year*, String *outputFile*)
Saves total electricity consumption to an output file.

Parameters

- **year** – Year of the assignment.
- **outputFile** – Output file name (with path).

saveZonalCarEnergyConsumptions

public void **saveZonalCarEnergyConsumptions** (int *year*, double *originZoneEnergyWeight*, String *outputFile*)
Saves zonal car energy consumptions to an output file.

Parameters

- **year** – Assignment year.
- **originZoneEnergyWeight** – Percentage of energy consumption assigned to origin zone (the rest assigned to destination zone).
- **outputFile** – Output file name (with path).

saveZonalTemporalTripStartsForEVs

public void **saveZonalTemporalTripStartsForEVs** (int *year*, *VehicleType* *vht*, String *outputFile*)
Saves zonal (LAD) and temporal (hourly) number of EV trips to an output file.

Parameters

- **year** – Assignment year.
- **vht** – Vehicle Type.
- **outputFile** – Output file name (with path).

saveZonalTemporalTripStartsForH2

public void **saveZonalTemporalTripStartsForH2** (int *year*, *VehicleType* *vht*, String *outputFile*)
Saves zonal (LAD) and temporal (hourly) number of H2 fuelled trips to an output file.

Parameters

- **year** – Assignment year.
- **vht** – Vehicle Type.
- **outputFile** – Output file name (with path).

saveZonalTemporalVehicleElectricity

public void **saveZonalTemporalVehicleElectricity** (int *year*, *VehicleType* *vht*, double *originZoneEnergyWeight*, String *outputFile*)
Saves zonal (LAD) and temporal (hourly) vehicle electricity consumptions to an output file.

Parameters

- **year** – Assignment year.
- **vht** – Vehicle Type (it will include consumption of autonomous vehicles too).
- **originZoneEnergyWeight** – Percentage of energy consumption assigned to origin zone (the rest assigned to destination zone).
- **outputFile** – Output file name (with path).

saveZonalTemporalVehicleHydrogen

public void **saveZonalTemporalVehicleHydrogen**(int *year*, *VehicleType vht*, double *originZoneEnergyWeight*, String *outputFile*)

Saves zonal (LAD) and temporal (hourly) vehicle hydrogen consumptions to an output file.

Parameters

- **year** – Assignment year.
- **vht** – Vehicle Type (it will include consumption of autonomous vehicles too).
- **originZoneEnergyWeight** – Percentage of energy consumption assigned to origin zone (the rest assigned to destination zone).
- **outputFile** – Output file name (with path).

saveZonalVehicleCO2Emissions

public void **saveZonalVehicleCO2Emissions**(int *year*, double *originZoneEnergyWeight*, String *outputFile*)

Saves zonal vehicle CO2 emissions to an output file.

Parameters

- **year** – Assignment year.
- **originZoneEnergyWeight** – Percentage of CO2 emission assigned to origin zone (the rest assigned to destination zone).
- **outputFile** – Output file name (with path).

saveZonalVehicleKilometres

public void **saveZonalVehicleKilometres**(int *year*, String *outputFile*)

Saves zonal vehicle-kilometres.

Parameters

- **year** – Assignment year.
- **outputFile** – Output file name (with path).

saveZonalVehicleKilometresWithAccessEgress

public void **saveZonalVehicleKilometresWithAccessEgress**(int *year*, String *outputFile*)

Saves zonal vehicle-kilometres that include access/egress and minor trips

Parameters

- **year** – Assignment year.
- **outputFile** – Output file name (with path).

setElectricityUnitCost

public void **setElectricityUnitCost**(double *electricityUnitCost*)
Setter method for the electricity unit cost.

Parameters

- **electricityUnitCost** – The cost of 1 kWh in £.

setEndNodeProbabilities

public void **setEndNodeProbabilities**(HashMap<Integer, Double> *endNodeProbabilities*)
Setter method for node probabilities.

Parameters

- **endNodeProbabilities** – Node probabilities.

setEnergyConsumptionParameters

public void **setEnergyConsumptionParameters**(*VehicleType vehicleType*, *EngineType engineType*, Map<*WebTAG*, Double> *parameters*)
Setter method for the energy consumption parameters.

Parameters

- **vehicleType** – Vehicle type
- **engineType** – Engine type
- **parameters** – Energy consumptions parameters (A, B, C, D)

setEnergyUnitCost

public void **setEnergyUnitCost**(*EnergyType energyType*, double *energyUnitCost*)
Setter method for the energy unit cost.

Parameters

- **energyType** – The type of a car engine.
- **energyUnitCost** – The cost of 1 L (of fuel) or 1 kWh (of electricity) in £.

setEngineTypeFractions

public void **setEngineTypeFractions**(*VehicleType vht*, Map<*EngineType*, Double> *engineTypeFractions*)
Setter method for energy type fractions.

Parameters

- **vht** – Vehicle type
- **engineTypeFractions** – Map with engine type fractions.

setStartNodeProbabilities

public void **setStartNodeProbabilities** (HashMap<Integer, Double> *startNodeProbabilities*)
Setter method for node probabilities.

Parameters

- **startNodeProbabilities** – Node probabilities.

updateCostSkimMatrix

public void **updateCostSkimMatrix** (*SkimMatrix costSkimMatrix*)
Updates cost skim matrix (zone-to-zone financial costs).

Parameters

- **costSkimMatrix** – Inter-zonal skim matrix (cost).

updateCostSkimMatrixFreight

public void **updateCostSkimMatrixFreight** (*SkimMatrixFreight costSkimMatrixFreight*)
Updates cost skim matrix (zone-to-zone financial costs) for freight.

Parameters

- **costSkimMatrixFreight** – Inter-zonal skim matrix (cost) for freight.

updateLinkTravelTimes

public void **updateLinkTravelTimes** ()
Updates link travel times per time of day.

updateLinkTravelTimes

public void **updateLinkTravelTimes** (double *weight*)
Updates link travel times using weighted averaging between new values (calculated from link volumes) and older values (stored in the instance field).

Parameters

- **weight** – Parameter for weighted averaging.

updateLinkVolumeInPCU

public void **updateLinkVolumeInPCU** ()
Updates daily link volumes in PCU from the trip list and stores it into instance variable.

updateLinkVolumeInPCUPerTimeOfDay

public void **updateLinkVolumeInPCUPerTimeOfDay**()
Updates link volumes in PCU per time of day from object's trip list and stores into instance variable.

updateLinkVolumePerVehicleType

public void **updateLinkVolumePerVehicleType**()
Updates daily link volumes per vehicle type from trip list and stores into instance variable.

updateTimeSkimMatrix

public void **updateTimeSkimMatrix**(*SkimMatrix timeSkimMatrix*)
Updates travel time skim matrix (zone-to-zone travel times).

Parameters

- **timeSkimMatrix** – Inter-zonal skim matrix (time).

updateTimeSkimMatrixFreight

public void **updateTimeSkimMatrixFreight**(*SkimMatrixFreight timeSkimMatrixFreight*)
Updates travel time skim matrix (zone-to-zone travel times) for freight.

Parameters

- **timeSkimMatrixFreight** – Inter-zonal skim matrix (time).

RoadNetworkAssignment.EnergyType

public static enum **EnergyType**

Enum Constants

CNG

public static final *RoadNetworkAssignment*.*EnergyType* **CNG**

DIESEL

public static final *RoadNetworkAssignment*.*EnergyType* **DIESEL**

ELECTRICITY

public static final *RoadNetworkAssignment*.*EnergyType* **ELECTRICITY**

HYDROGEN

public static final *RoadNetworkAssignment*.*EnergyType* **HYDROGEN**

LPG

public static final *RoadNetworkAssignment*.*EnergyType* **LPG**

PETROL

public static final *RoadNetworkAssignment*.*EnergyType* **PETROL**

RoadNetworkAssignment.EngineType

public static enum **EngineType**

Enum Constants

BEV

public static final *RoadNetworkAssignment*.*EngineType* **BEV**

FCEV_H2

public static final *RoadNetworkAssignment*.*EngineType* **FCEV_H2**

HEV_DIESEL

public static final *RoadNetworkAssignment*.*EngineType* **HEV_DIESEL**

HEV_PETROL

public static final *RoadNetworkAssignment*.*EngineType* **HEV_PETROL**

ICE_CNG

public static final *RoadNetworkAssignment*.*EngineType* **ICE_CNG**

ICE_DIESEL

public static final *RoadNetworkAssignment*.*EngineType* **ICE_DIESEL**

ICE_H2

public static final *RoadNetworkAssignment*.*EngineType* **ICE_H2**

ICE_LPG

public static final *RoadNetworkAssignment*.*EngineType* **ICE_LPG**

ICE_PETROL

public static final *RoadNetworkAssignment*.*EngineType* **ICE_PETROL**

PHEV_DIESEL

public static final *RoadNetworkAssignment*.*EngineType* **PHEV_DIESEL**

PHEV_PETROL

public static final *RoadNetworkAssignment*.*EngineType* **PHEV_PETROL**

RoadNetworkAssignment.TimeOfDay

public static enum **TimeOfDay**

Enum Constants

EIGHTAM

public static final *RoadNetworkAssignment*.*TimeOfDay* **EIGHTAM**

EIGHTPM

public static final *RoadNetworkAssignment*.*TimeOfDay* **EIGHTPM**

ELEVENAM

public static final *RoadNetworkAssignment*.*TimeOfDay* **ELEVENAM**

ELEVENPM

public static final *RoadNetworkAssignment*.*TimeOfDay* **ELEVENPM**

FIVEAM

public static final *RoadNetworkAssignment*.*TimeOfDay* **FIVEAM**

FIVEPM

public static final *RoadNetworkAssignment*.*TimeOfDay* **FIVEPM**

FOURAM

public static final *RoadNetworkAssignment*.*TimeOfDay* **FOURAM**

FOURPM

public static final *RoadNetworkAssignment*.*TimeOfDay* **FOURPM**

MIDNIGHT

public static final *RoadNetworkAssignment*.*TimeOfDay* **MIDNIGHT**

NINEAM

public static final *RoadNetworkAssignment*.*TimeOfDay* **NINEAM**

NINEPM

public static final *RoadNetworkAssignment*.*TimeOfDay* **NINEPM**

NOON

public static final *RoadNetworkAssignment*.*TimeOfDay* **NOON**

ONEAM

public static final *RoadNetworkAssignment*.*TimeOfDay* **ONEAM**

ONEPM

public static final *RoadNetworkAssignment*.*TimeOfDay* **ONEPM**

SEVENAM

public static final *RoadNetworkAssignment*.*TimeOfDay* **SEVENAM**

SEVENPM

public static final *RoadNetworkAssignment*.*TimeOfDay* **SEVENPM**

SIXAM

public static final *RoadNetworkAssignment*.*TimeOfDay* **SIXAM**

SIXPM

public static final *RoadNetworkAssignment*.*TimeOfDay* **SIXPM**

TENAM

public static final *RoadNetworkAssignment*.*TimeOfDay* **TENAM**

TENPM

public static final *RoadNetworkAssignment*.*TimeOfDay* **TENPM**

THREEAM

public static final *RoadNetworkAssignment*.*TimeOfDay* **THREEAM**

THREEPM

public static final *RoadNetworkAssignment*.*TimeOfDay* **THREEPM**

TWOAM

public static final *RoadNetworkAssignment*.*TimeOfDay* **TWOAM**

TWOPM

public static final *RoadNetworkAssignment*.*TimeOfDay* **TWOPM**

RoadNetworkAssignment.VehicleType

public static enum **VehicleType**

Enum Constants

ARTIC

public static final *RoadNetworkAssignment*.*VehicleType* **ARTIC**

ARTIC_AV

public static final *RoadNetworkAssignment*.*VehicleType* **ARTIC_AV**

CAR

public static final *RoadNetworkAssignment*.*VehicleType* **CAR**

CAR_AV

public static final *RoadNetworkAssignment*.*VehicleType* **CAR_AV**

RIGID

public static final *RoadNetworkAssignment*.*VehicleType* **RIGID**

RIGID_AV

public static final *RoadNetworkAssignment*.*VehicleType* **RIGID_AV**

VAN

public static final *RoadNetworkAssignment*.*VehicleType* **VAN**

VAN_AV

public static final *RoadNetworkAssignment*.*VehicleType* **VAN_AV**

Fields

value

int **value**

Methods

getValue

public int **getValue**()

RoadPath

public class **RoadPath** extends Path
Directed path (a list of directed nodes).

Author Milan Lovric

Constructors

RoadPath

public **RoadPath**()

RoadPath

public **RoadPath**(Collection *nodes*)

Methods

buildEdges

protected List **buildEdges**()
Internal method for building the edge set of the walk. This method calculated the edges upon every call.

Returns The list of edges for the walk, or null if the edge set could not be calculated due to an invalid walk.

isValid

public boolean **isValid**()

Route

public class **Route**
Route is a sequence of directed edges with a choice utility.

Author Milan Lovric

Constructors

Route

public **Route** (*RoadNetwork roadNetwork*)

Route

public **Route** (*RoadPath path*, *RoadNetwork roadNetwork*)
Construtor from a given path.

Parameters

- **path** – A path from which to construct a route.
- **roadNetwork** – Road network.

Methods

addEdge

public boolean **addEdge** (DirectedEdge *edge*)
Adds a directed edge to the end of the current route.

Parameters

- **edge** – Directed edge to be added.

Returns true if edge addition was successful, false otherwise.

addEdgeWithoutValidityCheck

public void **addEdgeWithoutValidityCheck** (DirectedEdge *edge*)
Adds a directed edge to the end of the current route.

Parameters

- **edge** – Directed edge to be added.

addEdgeWithoutValidityCheck

public void **addEdgeWithoutValidityCheck** (int *edgeID*)
Adds a directed edge to the end of the current route.

Parameters

- **edgeID** – Directed edge to be added.

calculateConsumption

public Map<*EnergyType*, Double> **calculateConsumption**(*VehicleType vht*, *EngineType et*, double[] *linkTravelTime*, Map<*VehicleType*, Map<*EngineType*, Map<*WebTAG*, Double>>> *energyConsumptionParameters*, Map<*VehicleType*, Map<*EngineType*, Double>> *relativeFuelEfficiency*)

Calculates energy consumption of the route.

Parameters

- **vht** – Vehicle type.
- **et** – Energy type.
- **linkTravelTime** – Link travel time.
- **energyConsumptionParameters** – Base year energy consumption parameters.
- **relativeFuelEfficiency** – Relative fuel efficiency compared to base year.

Returns Consumption for each type.

calculateCost

public void **calculateCost**(*VehicleType vht*, *EngineType et*, *TimeOfDay tod*, double[] *linkTravelTime*, Map<*VehicleType*, Map<*EngineType*, Map<*WebTAG*, Double>>> *energyConsumptionParameters*, Map<*VehicleType*, Map<*EngineType*, Double>> *relativeFuelEfficiency*, Map<*EnergyType*, Double> *energyUnitCosts*, List<*PricingPolicy*> *congestionCharges*)

Calculates the cost of the route.

Parameters

- **vht** – Vehicle type.
- **et** – Engine type.
- **tod** – Time of day.
- **linkTravelTime** – Link travel times.
- **energyConsumptionParameters** – Base year energy consumption parameters.
- **relativeFuelEfficiency** – Relative fuel efficiency (compared to base year).
- **energyUnitCosts** – Energy unit costs.
- **congestionCharges** – Congestion charges.

calculateLength

public void **calculateLength**()

Calculates the length of the route.

calculateTravelTime

public void **calculateTravelTime**(double[] *linkTravelTime*, double *avgIntersectionDelay*)

Calculates the route travel time based on link travel times.

Parameters

- **linkTravelTime** – Link travel times.
- **avgIntersectionDelay** – Average intersection delay (in minutes).

calculateUtility

public void **calculateUtility**(*VehicleType* *vht*, *EngineType* *et*, *TimeOfDay* *tod*, double[] *linkTravelTime*, Map<*VehicleType*, Map<*EngineType*, Map<*WebTAG*, Double>>> *energyConsumptionParameters*, Map<*VehicleType*, Map<*EngineType*, Double>> *relativeFuelEfficiency*, Map<*EnergyType*, Double> *energyUnitCosts*, List<*PricingPolicy*> *congestionCharges*, Map<*RouteChoiceParams*, Double> *params*)

Calculates the utility of the route.

Parameters

- **vht** – Vehicle type.
- **et** – Engine type.
- **tod** – Time of day.
- **linkTravelTime** – Link travel times.
- **energyConsumptionParameters** – Energy consumption parameters (A, B, C, D) for a combination of vehicle type and engine type.
- **relativeFuelEfficiency** – Relative fuel efficiency compared to the base year.
- **energyUnitCosts** – Energy unit costs.
- **congestionCharges** – Congestion charges.
- **params** – Route choice parameters.

contains

public boolean **contains**(Edge *edge*)

Checks if route contains the edge.

Parameters

- **edge** – Edge object.

Returns True if route contains the edge.

contains

public boolean **contains**(int *edgeID*)

Checks if route contains the edge.

Parameters

- **edgeID** – Edge id.

Returns True if route contains the edge.

equals

public boolean **equals** (Object *obj*)

getCost

public double **getCost** ()
Getter method for route cost.

Returns Route cost.

getDestinationNode

public DirectedNode **getDestinationNode** ()
Getter method for destination node.

Returns Destination node.

getEdges

public TIntArrayList **getEdges** ()
Getter method for the list of edges.

Returns List of edge IDs.

getFormattedString

public String **getFormattedString** ()
Gets formatted string representation of the route.

Returns Route as a string.

getFormattedStringEdgeIDsOnly

public String **getFormattedStringEdgeIDsOnly** ()
Gets formatted string representation of the route using edge IDs only.

Returns Route as a string.

getLength

public double **getLength** ()
Getter method for route length.

Returns Route length.

getNumberOfIntersections

public int **getNumberOfIntersections**()
Getter method for number of intersections.

Returns Number of intersections.

getOriginNode

public DirectedNode **getOriginNode**()
Getter method for route origin node.

Returns Origin node.

getRoadNetwork

public *RoadNetwork* **getRoadNetwork**()
Getter method for the road network.

Returns Road network.

getTime

public double **getTime**()
Getter method for route time.

Returns Route time.

getUtility

public double **getUtility**()
Getter method for route utility.

Returns Route utility.

hashCode

public int **hashCode**()

isEmpty

public boolean **isEmpty**()
Checks if route is empty or not.

Returns True if route is empty.

isValid

public boolean **isValid**()
Checks if route is valid (successive edges in the route are connected in a directional way).

Returns True if route is valid.

setUtility

public void **setUtility**(double *utility*)
Setter method for route utility.

Parameters

- **utility** – Route utility.

toString

public String **toString**()

trimToSize

public void **trimToSize**()
Trims edges list to size and calculate length (onetime operation).

Route.WebTAG

public static enum **WebTAG**

Enum Constants

A

public static final *Route*.*WebTAG* **A**

B

public static final *Route*.*WebTAG* **B**

C

public static final *Route*.*WebTAG* **C**

D

public static final *Route*.*WebTAG* **D**

RouteSet

public class **RouteSet**
RouteSet is a choice set of possible routes between an origin and a destination node.

Author Milan Lovric

Constructors

RouteSet

public **RouteSet** (*RoadNetwork roadNetwork*)
Constructor.

Parameters

- **roadNetwork** – Road network.

Methods

addRoute

public void **addRoute** (*Route route*)
Adds a route to the choice set.

Parameters

- **route** – Route to be added.

addRouteWithoutAnyChecks

public void **addRouteWithoutAnyChecks** (*Route route*)
Adds a route to the choice set.

Parameters

- **route** – Route to be added.

addRouteWithoutValidityAndEndNodesCheck

public void **addRouteWithoutValidityAndEndNodesCheck** (*Route route*)
Adds a route to the choice set.

Parameters

- **route** – Route to be added.

addRouteWithoutValidityCheck

public void **addRouteWithoutValidityCheck** (*Route route*)
Adds a route to the choice set.

Parameters

- **route** – Route to be added.

calculatePathsizes

public void **calculatePathsizes**()
Calculate path sizes (also calculates route lengths if they had not been calculated before).

calculateProbabilities

public void **calculateProbabilities**()
Calculates choice probabilities using logit formula.

calculateUtilities

public void **calculateUtilities**(*VehicleType* *vht*, *EngineType* *et*, *TimeOfDay* *tod*, double[] *linkTravelTime*, Map<*VehicleType*, Map<*EngineType*, Map<*WebTAG*, Double>>> *energyConsumptionParameters*, Map<*VehicleType*, Map<*EngineType*, Double>> *relativeFuelEfficiency*, Map<*EnergyType*, Double> *energyUnitCosts*, List<*PricingPolicy*> *congestionCharges*, Map<*RouteChoiceParams*, Double> *params*)
Re-calculates utilities for all the routes.

Parameters

- **vht** – Vehicle type.
- **et** – Engine type.
- **tod** – Time of day.
- **linkTravelTime** – Link travel times.
- **energyConsumptionParameters** – Base year energy consumption parameters.
- **relativeFuelEfficiency** – Relative fuel efficiency compared to the base year.
- **energyUnitCosts** – Energy unit costs.
- **congestionCharges** – Congestion charges.
- **params** – Route choice parameters.

choose

public *Route* **choose**()
Chooses a route based on the probabilities.

Returns Chosen route.

correctUtilityWithPathSize

public void **correctUtilityWithPathSize**(int *routeIndex*)
Corrects utility with path size for a particular route within the choice set.

Parameters

- **routeIndex** – index of the route (list element) within the choice set

getChoiceSet

public List<*Route*> **getChoiceSet** ( )
Getter method for the choice set.

Returns Choice set (list of routes).

getDestinationNode

public DirectedNode **getDestinationNode** ( )

Returns Destination node of the choice set.

getIndexOfRoute

public int **getIndexOfRoute** (*Route route*)
Gets the index of a route in the choice set.

Parameters

- **route** – The route which index is sought for.

Returns Route index.

getOriginNode

public DirectedNode **getOriginNode** ( )

Returns Origin node of the choice set.

getPathsizes

public double[] **getPathsizes** ( )
Getter method for pathsizes.

Returns Choice pathsizes.

getProbabilities

public double[] **getProbabilities** ( )
Getter method for choice probabilities.

Returns Choice probabilities.

getProbabilitiesAsList

public ArrayList<Double> **getProbabilitiesAsList** ()
Getter method for choice probabilities.

Returns Choice probabilities.

getSize

public int **getSize** ()

Returns Size of the choice set (number of routes).

getUtilities

public ArrayList<Double> **getUtilities** ()
Getter method for choice utilities.

Returns Choice utilities.

printChoiceSet

public void **printChoiceSet** ()
Prints the entire choice set.

printPathsizes

public void **printPathsizes** ()
Prints pathsizes for the route set.

printProbabilities

public void **printProbabilities** ()
Prints probabilities for the route set.

printStatistics

public void **printStatistics** ()
Prints statistic for the route set (choice set size for each node pair).

printUtilities

public void **printUtilities** ()
Prints utilities for the route set.

RouteSet.RouteChoiceParams

public static enum **RouteChoiceParams**

Enum Constants

COST

public static final *RouteSet*.*RouteChoiceParams* **COST**

DELAY

public static final *RouteSet*.*RouteChoiceParams* **DELAY**

INTERSEC

public static final *RouteSet*.*RouteChoiceParams* **INTERSEC**

LENGTH

public static final *RouteSet*.*RouteChoiceParams* **LENGTH**

TIME

public static final *RouteSet*.*RouteChoiceParams* **TIME**

RouteSetGenerator

public class **RouteSetGenerator**
RouteSetGenerator can generate, save and read route sets for the route choice.

Author Milan Lovric

Fields

INITIAL_ROUTE_CAPACITY

public static final int **INITIAL_ROUTE_CAPACITY**

INITIAL_ROUTE_SET_CAPACITY

public static final int **INITIAL_ROUTE_SET_CAPACITY**

Constructors

RouteSetGenerator

public **RouteSetGenerator**(*RoadNetwork roadNetwork*, Properties *props*)
Constructor for the route set generator.

Parameters

- **roadNetwork** – Road network.
- **props** – Parameters from the config file.

Methods

addRoute

public void **addRoute**(*Route route*)
Adds a route to the route set.

Parameters

- **route** – Route to be added.

addRouteWithoutValidityCheck

public void **addRouteWithoutValidityCheck**(*Route route*)
Adds a route to the route set.

Parameters

- **route** – Route to be added.

calculateAllPathsizes

public void **calculateAllPathsizes**()
Calculates all pathsizes for all the route sets (expensive operation).

clearRoutes

public void **clearRoutes**()
Clears all stored routes.

generateRouteSetBetweenFreightZones

public void **generateRouteSetBetweenFreightZones**(int *originFreightZone*, int *destinationFreightZone*)
Generates routes between two freight zones. A freight zone can be either an LAD (<= 1032) or a point. Zone ID ranges from the BYFM DfT model:

- England: 1 - 867
- Wales: 901 - 922
- Scotland: 1001 - 1032
- Freight airports: 1111 - 1115
- Major distribution centres: 1201 - 1256
- Freight ports: 1301 - 1388

Parameters

- **originFreightZone** – Origin freight zone.
- **destinationFreightZone** – Destination freight zone.

generateRouteSetBetweenFreightZones

public void **generateRouteSetBetweenFreightZones**(int *originFreightZone*, int *destinationFreightZone*, int *topNodes*)

Generates routes between two freight zones. A freight zone can be either an LAD (<= 1032) or a point. Zone ID ranges from the BYFM DfT model:

- England: 1 - 867
- Wales: 901 - 922
- Scotland: 1001 - 1032
- Freight airports: 1111 - 1115
- Major distribution centres: 1201 - 1256
- Freight ports: 1301 - 1388

Parameters

- **originFreightZone** – Origin freight zone.
- **destinationFreightZone** – Destination freight zone.
- **topNodes** – Number of topNodes to consider for inter-zonal routes.

generateRouteSetForFreightMatrix

public void **generateRouteSetForFreightMatrix**(*FreightMatrix freightMatrix*, int *topNodes*)

Generates routes for all non-zero OD flows in the freight OD matrix. Zone ID ranges from the BYFM DfT model:

Parameters

- **freightMatrix** – Freight matrix.
- **topNodes** – Number of topNodes to consider for inter-zonal routes.

generateRouteSetForFreightMatrix

public void **generateRouteSetForFreightMatrix**(*FreightMatrix freightMatrix*, int *sliceIndex*, int *sliceNumber*)

Generates routes for a slice of the OD matrix (useful for cluster computing), for topNodes only. There might still be some overlap between the slices as some nodes (to which point freight zones are assigned appear again in LAD freight zones).

Parameters

- **freightMatrix** – Freight matrix.
- **sliceIndex** – Index of the OD matrix slice for which to generate routes [1..N].
- **sliceNumber** – Number of slices to divide matrix into (N).

generateRouteSetForFreightMatrix

public void **generateRouteSetForFreightMatrix**(*FreightMatrix freightMatrix*, int *sliceIndex*, int *sliceNumber*, int *topNodes*)

Generates routes for a slice of the OD matrix (useful for cluster computing), for topNodes only. There might still be some overlap between the slices as some nodes (to which point freight zones are assigned appear again in LAD freight zones).

Parameters

- **freightMatrix** – Freight matrix.
- **sliceIndex** – Index of the OD matrix slice for which to generate routes [1..N].
- **sliceNumber** – Number of slices to divide matrix into (N).
- **topNodes** – Number of topNodes to consider for inter-zonal routes.

generateRouteSetForODMatrix

public void **generateRouteSetForODMatrix**(*ODMatrixMultiKey matrix*, int *topNodes*)

Generates routes for all non-zero OD flows in the OD matrix. For inter-zonal flows generates routes only between top N nodes.

Parameters

- **matrix** – Origin-destination matrix.
- **topNodes** – Number of topNodes to consider for inter-zonal routes.

generateRouteSetForODMatrix

public void **generateRouteSetForODMatrix**(*ODMatrixMultiKey matrix*)

Generates routes for all non-zero OD flows in the OD matrix.

Parameters

- **matrix** – Origin-destination matrix.

generateRouteSetForODMatrix

public void **generateRouteSetForODMatrix**(*ODMatrixMultiKey matrix*, int *sliceIndex*, int *sliceNumber*, int *topNodes*)

Generates routes for a slice of the OD matrix (useful for cluster computing), for topNodes only

Parameters

- **matrix** – Origin-destination matrix.
- **sliceIndex** – Index of the OD matrix slice for which to generate routes [1..N].
- **sliceNumber** – Number of slices to divide matrix into (N).
- **topNodes** – Number of topNodes to consider for inter-zonal routes.

generateRouteSetForODMatrix

public void **generateRouteSetForODMatrix**(*ODMatrixMultiKey matrix*, int *sliceIndex*, int *sliceNumber*)

Generates routes for a slice of the Tempro OD matrix (useful for cluster computing).

Parameters

- **matrix** – Origin-destination matrix.
- **sliceIndex** – Index of the OD matrix slice for which to generate routes [1..N].
- **sliceNumber** – Number of slices to divide matrix into (N).

generateRouteSetForODMatrixTempro

public void **generateRouteSetForODMatrixTempro**(*ODMatrixMultiKey matrix*, *Zoning zoning*)

Generates routes for all non-zero OD flows in the OD matrix.

Parameters

- **matrix** – Origin-destination matrix.
- **zoning** – Tempro zoning system.

generateRouteSetForODMatrixTempro

public void **generateRouteSetForODMatrixTempro**(*RealODMatrixTempro matrix*, *Zoning zoning*, int *sliceIndex*, int *sliceNumber*)

Generates routes for a slice of the OD matrix (useful for cluster computing).

Parameters

- **matrix** – Origin-destination matrix.
- **zoning** – Tempro zoning system.
- **sliceIndex** – Index of the OD matrix slice for which to generate routes [1..N].
- **sliceNumber** – Number of slices to divide matrix into (N).

generateRouteSetForODMatrixTemproDistanceBased

public void **generateRouteSetForODMatrixTemproDistanceBased**(*RealODMatrixTempro matrix*, *Zoning zoning*, int *sliceIndex*, int *sliceNumber*)

Generates routes for a slice of the OD matrix (useful for cluster computing). The number of routes increases the smaller the distance between two Tempro zones.

Parameters

- **matrix** – Origin-destination matrix.
- **zoning** – Tempro zoning system.
- **sliceIndex** – Index of the OD matrix slice for which to generate routes [1..N].
- **sliceNumber** – Number of slices to divide matrix into (N).

generateRouteSetNodeToNode

public void **generateRouteSetNodeToNode**(int *origin*, int *destination*)

Generates a route set between two nodes (if it does not already exist in the route set).

Parameters

- **origin** – Origin node ID.
- **destination** – Destination node ID.

generateRouteSetWithLinkElimination

public void **generateRouteSetWithLinkElimination**(int *origin*, int *destination*)

Generates a route set between two nodes using the link elimination method - It first finds the fastest path and then blocks each of its links and tries to find an alternative path.

Parameters

- **origin** – Origin node ID.
- **destination** – Destination node ID.

generateRouteSetWithRandomLinkEliminationRestricted

public void **generateRouteSetWithRandomLinkEliminationRestricted**(int *origin*, int *destination*)

Generates a route set between two nodes using the random link elimination method - It first finds the fastest path and then blocks random links within the fastest path and tries to find an alternative path. The search is limited by the total number of path finding calls and the required number of generated paths.

Parameters

- **origin** – Origin node ID.
- **destination** – Destination node ID.

generateRouteSetWithRandomLinkEliminationRestricted

public void **generateRouteSetWithRandomLinkEliminationRestricted**(int *origin*, int *destination*, int *routeLimit*, int *generationLimit*)

Generates a route set between two nodes using the random link elimination method - It first finds the fastest path and then blocks random links within the fastest path and tries to find an alternative path. The search is limited by the total number of path finding calls and the required number of generated paths.

Parameters

- **origin** – Origin node ID.
- **destination** – Destination node ID.
- **routeLimit** – Maximum allowed number of generated routes.
- **generationLimit** – Number of generation trials to get a potentially new route.

generateRouteSetZoneToZone

public void **generateRouteSetZoneToZone** (String *originLAD*, String *destinationLAD*)
Generates routes between all combinations of nodes from two LAD zones

Parameters

- **originLAD** – Origin LAD.
- **destinationLAD** – Destination LAD.

generateRouteSetZoneToZone

public void **generateRouteSetZoneToZone** (String *originLAD*, String *destinationLAD*, int *topNodes*)
Generates routes between top N nodes (sorted by gravitating population) from two LAD zones. If origin and destination LAD are the same (i.e., intra-zonal), then use all the nodes

Parameters

- **originLAD** – Origin LAD.
- **destinationLAD** – Destination LAD.
- **topNodes** – Number of top nodes within LAD to consider.

generateRouteSetZoneToZoneTempro

public void **generateRouteSetZoneToZoneTempro** (String *originZone*, String *destinationZone*, *Zoning* *zoning*)
Generates routes between the nearest nodes of two Tempro zones.

Parameters

- **originZone** – Origin Tempro zone.
- **destinationZone** – Destination Tempro zone.
- **zoning** – Tempro zoning system.

generateRouteSetZoneToZoneTemproDistanceBased

public void **generateRouteSetZoneToZoneTemproDistanceBased** (String *originZone*, String *destinationZone*, *Zoning* *zoning*)
Generates routes between the nearest nodes of two Tempro zones.

Parameters

- **originZone** – Origin Tempro zone.
- **destinationZone** – Destination Tempro zone.
- **zoning** – Tempro zoning system.

generateSingleNodeRoutes

public void **generateSingleNodeRoutes** ()
Generates single nodes routes.

getNumberOfRouteSets

public int **getNumberOfRouteSets**()
Gets the numbers of route sets (OD pairs).

Returns Number of route sets.

getNumberOfRoutes

public int **getNumberOfRoutes**()
Gets the total number of routes.

Returns Number of routes.

getRoadNetwork

public *RoadNetwork* **getRoadNetwork**()
Getter method for the road network.

Returns Road network.

getRouteSet

public *RouteSet* **getRouteSet**(int *origin*, int *destination*)
Getter method for a route set between a specific origin and a destination.

Parameters

- **origin** – Origin node ID.
- **destination** – Destination node ID.

Returns Route set.

getStatistics

public String **getStatistics**()
Gets route set statistics in a string.

Returns Route set statistics.

printChoiceSets

public void **printChoiceSets**()
Prints all route sets.

printStatistics

public void **printStatistics**()
Prints all route set statistics.

readRoutes

public void **readRoutes** (String *fileName*)
Reads route sets from a text file.

Parameters

- **fileName** – File name.

readRoutesBinary

public void **readRoutesBinary** (String *fileName*)
Reads route sets from a text file.

Parameters

- **fileName** – File name.

readRoutesBinaryGZIPpedWithoutValidityCheck

public void **readRoutesBinaryGZIPpedWithoutValidityCheck** (String *fileName*)
Reads route sets from a text file.

Parameters

- **fileName** – File name.

readRoutesBinaryShortWithoutValidityCheck

public void **readRoutesBinaryShortWithoutValidityCheck** (String *fileName*)
Reads route sets from a text file.

Parameters

- **fileName** – File name.

readRoutesBinaryWithoutValidityCheck

public void **readRoutesBinaryWithoutValidityCheck** (String *fileName*)
Reads route sets from a text file.

Parameters

- **fileName** – File name.

readRoutesWithoutValidityCheck

public void **readRoutesWithoutValidityCheck** (String *fileName*)
Reads route sets from a text file without checking whether the routes are valid.

Parameters

- **fileName** – File name.

removeRoutesWithEdge

public void **removeRoutesWithEdge** (int *edgeID*)
Removes all the routes that contain a given edge (used for disruption).

Parameters

- **edgeID** – Edge ID.

removeRoutesWithEdge

public void **removeRoutesWithEdge** (int *edgeID*, List<*Route*> *removedRoutes*)
Removes all the routes that contain a given edge and store in the list.

Parameters

- **edgeID** – Edge ID.
- **removedRoutes** – List of removed routes.

saveRoutes

public void **saveRoutes** (String *fileName*, boolean *append*)
Saves all route sets into a text file.

Parameters

- **fileName** – File name.
- **append** – Whether to append to an existing file.

saveRoutesBinary

public void **saveRoutesBinary** (String *fileName*, boolean *append*)
Saves all route sets into a binary file.

Parameters

- **fileName** – File name.
- **append** – Whether to append to an existing file.

saveRoutesBinaryGZIPped

public void **saveRoutesBinaryGZIPped** (String *fileName*, boolean *append*)
Saves all route sets into a binary file.

Parameters

- **fileName** – File name.
- **append** – Whether to append to an existing file.

saveRoutesBinaryShort

public void **saveRoutesBinaryShort** (String *fileName*, boolean *append*)
Saves all route sets into a binary file. It also uses unsigned short (2 Bytes, which has a max. value of 65535).

Parameters

- **fileName** – File name.
- **append** – Whether to append to an existing file.

Trip

public class **Trip**
This class stores information about a performed trip.

Author Milan Lovric

Fields

destination

protected int **destination**

engine

protected *EngineType* **engine**

hour

protected *TimeOfDay* **hour**

multiplier

protected int **multiplier**

origin

protected int **origin**

route

protected *Route* **route**

vehicle

protected *VehicleType* **vehicle**

Constructors

Trip

public **Trip**(*VehicleType vehicle*, *EngineType engine*, *Route route*, *TimeOfDay hour*, Integer *origin*, Integer *destination*)

Constructor for an LAD-based trip. Origin and destination fields are used for freight trips (according to DfT's BYFM zonal coding). Origin and destination for passenger car/AV trips are 0 as their correct origin and destination LAD zone can be obtained using the first and the last node of the route.

Parameters

- **vehicle** – Vehicle type.
- **engine** – Engine type.
- **route** – Route.
- **hour** – Time of day.
- **origin** – Origin zone for freight trips (null for passenger trips).
- **destination** – Destination zone for freight trips (null for passenger trips).

Trip

public **Trip**(*VehicleType vehicle*, *EngineType engine*, *Route route*, *TimeOfDay hour*, Integer *origin*, Integer *destination*, int *multiplier*)

Constructor for a trip. Origin and destination are used for freight trips (according to DfT's BYFM zonal coding). Origin and destination for passenger car/AV trips are 0 as their correct origin and destination LAD zone can be obtained using the first and the last node of the route. Multiplier is used to store multiple instances of the same trip (vs creating multiple objects), thus reducing the memory footprint.

Parameters

- **vehicle** – Vehicle type.
- **engine** – Engine type.
- **route** – Route.
- **hour** – Time of day.
- **origin** – Origin zone for freight trips (null for passenger trips).
- **destination** – Destination zone for freight trips (null for passenger trips).
- **multiplier** – Multiplies the same trip.

Methods

getAccessEgressConsumption

protected Map<*EnergyType*, Double> **getAccessEgressConsumption**(double[] *linkTravelTime*, double[] *averageAccessEgressMap*, double *averageAccessEgressSpeed*, Map<*VehicleType*, Map<*EngineType*, Map<*WebTAG*, Double>>> *energyConsumptionParameters*, Map<*VehicleType*, Map<*EngineType*, Double>> *relativeFuelEfficiency*)

Calculate trip consumption only on access and egress.

Parameters

- **linkTravelTime** –
- **averageAccessEgressMap** –
- **averageAccessEgressSpeed** –
- **energyConsumptions** –
- **relativeFuelEfficiency** –

Returns Trip consumptions.

getCO2emission

public Double **getCO2emission**(double[] *linkTravelTime*, double[] *averageAccessEgressMap*, double *averageAccessEgressSpeed*, Map<*VehicleType*, Map<*EngineType*, Map<*WebTAG*, Double>>> *energyConsumptionParameters*, Map<*VehicleType*, Map<*EngineType*, Double>> *relativeFuelEfficiency*, Map<*EnergyType*, Double> *unitCO2Emissions*, boolean *flagIncludeAccessEgress*)

Calculates total CO2 emission for the trip.

Parameters

- **linkTravelTime** – Link travel time.
- **averageAccessEgressMap** – Average access/egress distance to a node for LAD-based trips.
- **averageAccessEgressSpeed** – Average accces/egress speed.
- **energyConsumptionParameters** – Energy consumption parameters.
- **relativeFuelEfficiency** – Relative fuel efficiency.
- **unitCO2Emissions** – Unit CO2 emissions.
- **boolean** – flagIncludeAccessEgress Whether to include access/egress.

Returns CO2 emissions per energy type.

getConsumption

public Map<*EnergyType*, Double> **getConsumption**(double[] *linkTravelTime*, double[] *averageAccessEgressMap*, double *averageAccessEgressSpeed*, Map<*VehicleType*, Map<*EngineType*, Map<*WebTAG*, Double>>> *energyConsumptionParameters*, Map<*VehicleType*, Map<*EngineType*, Double>> *relativeFuelEfficiency*, boolean *flagIncludeAccessEgress*)

Calculate trip consumption including access and egress.

Parameters

- **linkTravelTime** – Link travel time.
- **averageAccessEgressMap** – Average access/egress distance to a node for LAD-based trips.
- **averageAccessEgressSpeed** – Average acess/egress speed.
- **energyConsumptionParameters** – Energy consumption parameters.
- **relativeFuelEfficiency** – Relative fuel efficiency.
- **boolean** – flagIncludeAccessEgress Whether to include access/egress.

Returns Trip consumptions.

getCost

public double **getCost**(double[] *linkTravelTime*, double[] *averageAccessEgressMap*, double *averageAccessEgressSpeed*, Map<*EnergyType*, Double> *energyUnitCosts*, Map<*VehicleType*, Map<*EngineType*, Map<*WebTAG*, Double>>> *energyConsumptionParameters*, Map<*VehicleType*, Map<*EngineType*, Double>> *relativeFuelEfficiency*, List<*PricingPolicy*> *congestionCharges*, boolean *flagIncludeAccessEgress*)

Calculate cost of the trip (fuel cost + congestion charge, if any).

Parameters

- **linkTravelTime** – Link travel time.
- **averageAccessEgressMap** – Average access/egress distance to a node for LAD-based trips.
- **averageAccessEgressSpeed** – Average access/egress speed.
- **energyUnitCosts** – Energy unit costs.
- **energyConsumptionParameters** – Energy consumption parameters.
- **relativeFuelEfficiency** – Relative fuel efficiency.
- **congestionCharges** – Congestion charges.
- **boolean** – flagIncludeAccessEgress Whether to include access/egress.

Returns Total trip cost.

getDestination

public int **getDestination**()
Gets freight trip destination zone (using DfT BYFM zone coding).

Returns Freight trip destination zone.

getDestinationLAD

public String **getDestinationLAD**(Map<Integer, String> *nodeToZoneMap*)
Gets trip destination zone (LAD).

Parameters

- **nodeToZoneMap** – Mapping from nodes to zones.

Returns Trip destination zone.

getDestinationLadID

public int **getDestinationLadID**()
Gets trip destination LAD zone ID.

Returns Trip destination zone LAD ID.

getDestinationNode

public DirectedNode **getDestinationNode**()
Gets the trip destination node.

Returns Destination node.

getEngine

public *EngineType* **getEngine**()
Getter method for engine type.

Returns Vehicle engine type.

getLength

public double **getLength**(double[] *averageAccessEgressMap*)
Get trip length including access/egress.

Parameters

- **averageAccessEgressMap** – Mapping between nodeID and average access/egress for that node.

Returns Trip length including access/egress [in km]

getMultiplier

public int **getMultiplier**()
Getter method for the multiplier.

Returns Multiplier.

getOrigin

public int **getOrigin**()
Gets freight trip origin zone (using DfT BYFM zone coding).

Returns Freight trip origin zone.

getOriginLAD

public String **getOriginLAD**(Map<Integer, String> *nodeToZoneMap*)
Gets trip origin zone (LAD).

Parameters

- **nodeToZoneMap** – Mapping from nodes to zones.

Returns Trip origin zone.

getOriginLadID

public int **getOriginLadID**()
Gets trip origin LAD zone ID.

Returns Origin zone LAD ID.

getOriginNode

public DirectedNode **getOriginNode**()
Gets the trip origin node.

Returns Origin node.

getRoute

public *Route* **getRoute**()
Getter method for the route.

Returns Route.

getTimeOfDay

public *TimeOfDay* **getTimeOfDay**()
Getter method for the time of day.

Returns Time of day.

getTravelTime

public double **getTravelTime**(double[] *linkTravelTime*, double *avgIntersectionDelay*, double[] *averageAccessEgressMap*, double *averageAccessEgressSpeed*, boolean *flagIncludeAccessEgress*)

Calculates travel time including access/egress.

Parameters

- **linkTravelTime** – Link-based travel time (should be for the same hour as the trip's time of day).
- **avgIntersectionDelay** – Average intersection delay.
- **averageAccessEgressMap** – Mapping between nodeID and average access/egress for that node.
- **averageAccessEgressSpeed** – Average access/egress speed.
- **flagIncludeAccessEgress** – Whether to include access/egress travel time.

Returns Trip travel time including access/egress [in min].

getVehicle

public *VehicleType* **getVehicle**()

Getter method for vehicle type.

Returns Vehicle type.

isTripGoingThroughCongestionChargingZone

public boolean **isTripGoingThroughCongestionChargingZone**(String *policyName*, List<*PricingPolicy*> *congestionCharges*)

Check whether trip is going through a congestion charging zone for a particular policy.

Parameters

- **policyName** – Policy name.
- **congestionCharges** – Congestion charges.

Returns True if it is going through the congestion charging zone.

toString

public String **toString**()

TripMinor

public class **TripMinor** extends *Trip*

This class stores information about a performed trip on minor roads (for which the network is not modelled).

Author Milan Lovric

Fields

zoning

public static *Zoning* **zoning**

Constructors

TripMinor

public **TripMinor**(*VehicleType vehicle*, *EngineType engine*, *TimeOfDay hour*, Integer *originTemproZoneID*, Integer *destinationTemproZoneID*, double *length*, *Zoning zoning*)

Constructor for a trip. Origin and destination are used for freight trips (according to DfT's BYFM zonal coding). Origin and destination for passenger car/AV trips are 0 as their correct origin and destination zone can be obtained using the first and the last node of the route.

Parameters

- **vehicle** – Vehicle type.
- **engine** – Engine type.
- **route** – Route.
- **hour** – Time of day.
- **originTemproZoneID** – Origin tempro zone ID.
- **destinationTemproZoneID** – Destination tempro zone ID.
- **length** – Trip length;
- **zoning** – Zoning system.

TripMinor

public **TripMinor**(*VehicleType vehicle*, *EngineType engine*, *TimeOfDay hour*, Integer *originTemproZoneID*, Integer *destinationTemproZoneID*, double *length*, *Zoning zoning*, int *multiplier*)

Constructor for a trip. Origin and destination are used for freight trips (according to DfT's BYFM zonal coding). Origin and destination for passenger car/AV trips are 0 as their correct origin and destination zone can be obtained using the first and the last node of the route.

Parameters

- **vehicle** – Vehicle type.
- **engine** – Engine type.
- **route** – Route.
- **hour** – Time of day.
- **originTemproZoneID** – Origin tempro zone ID.
- **destinationTemproZoneID** – Destination tempro zone ID.
- **length** – Trip length;
- **zoning** – Zoning system.
- **multiplier** – Multiplies the same trip.

Methods

getAccessEgressConsumption

protected Map<*EnergyType*, Double> **getAccessEgressConsumption**(double[] *linkTravelTime*, double[] *distanceFromTemproZoneToNearestNode*, double *averageAccessEgressSpeed*, Map<*VehicleType*, Map<*EngineType*, Map<*WebTAG*, Double>>> *energyConsumptionParameters*, Map<*VehicleType*, Map<*EngineType*, Double>> *relativeFuelEfficiency*)

getCO2emission

public double **getCO2emission**(double *averageSpeed*, Map<*VehicleType*, Map<*EngineType*, Map<*WebTAG*, Double>>> *energyConsumptionParameters*, Map<*VehicleType*, Map<*EngineType*, Double>> *relativeFuelEfficiency*, Map<*EnergyType*, Double> *unitCO2Emissions*)

Gets CO2 emission for the minor trip.

Parameters

- **averageSpeed** – Average speed for a minor trip.
- **energyConsumptionParameters** – Energy consumption parameters.
- **relativeFuelEfficiency** – Relative fuel efficiency.

Returns CO2 emission for the trip.

getCO2emission

public Double **getCO2emission**(double[] *linkTravelTime*, double[] *distanceFromTemproZoneToNearestNode*, double *averageAccessEgressSpeed*, Map<*VehicleType*, Map<*EngineType*, Map<*WebTAG*, Double>>> *energyConsumptionParameters*, Map<*VehicleType*, Map<*EngineType*, Double>> *relativeFuelEfficiency*, Map<*EnergyType*, Double> *unitCO2Emissions*, boolean *flagIncludeAccessEgress*)

getConsumption

public Map<*EnergyType*, Double> **getConsumption**(double *averageSpeed*, Map<*VehicleType*, Map<*EngineType*, Map<*WebTAG*, Double>>> *energyConsumptionParameters*, Map<*VehicleType*, Map<*EngineType*, Double>> *relativeFuelEfficiency*)

Gets energy consumptions for the minor trip.

Parameters

- **averageSpeed** – Average speed for a minor trip.

- **energyConsumptionParameters** – Energy consumption parameters.
- **relativeFuelEfficiency** – Relative fuel efficiency.

Returns Energy consumptions for the trip.

getConsumption

public Map<*EnergyType*, Double> **getConsumption**(double[] *linkTravelTime*, double[] *distanceFromTemproZoneToNearestNode*, double *averageAccessEgressSpeed*, Map<*VehicleType*, Map<*EngineType*, Map<*WebTAG*, Double>>> *energyConsumptionParameters*, Map<*VehicleType*, Map<*EngineType*, Double>> *relativeFuelEfficiency*, boolean *flagIncludeAccessEgress*)

getCost

public double **getCost**(double *averageSpeed*, Map<*EnergyType*, Double> *energyUnitCosts*, Map<*VehicleType*, Map<*EngineType*, Map<*WebTAG*, Double>>> *energyConsumptionParameters*, Map<*VehicleType*, Map<*EngineType*, Double>> *relativeFuelEfficiency*)

Gets fuel cost for the minor trip.

Parameters

- **averageSpeed** – Average speed for a minor trip.
- **energyUnitCosts** – Energy unit costs.
- **energyConsumptionParameters** – Energy consumption parameters.
- **relativeFuelEfficiency** – Relative fuel efficiency.

Returns Energy consumptions for the trip.

getCost

public double **getCost**(double[] *linkTravelTime*, double[] *distanceFromTemproZoneToNearestNode*, double *averageAccessEgressSpeed*, Map<*EnergyType*, Double> *energyUnitCosts*, Map<*VehicleType*, Map<*EngineType*, Map<*WebTAG*, Double>>> *energyConsumptionParameters*, Map<*VehicleType*, Map<*EngineType*, Double>> *relativeFuelEfficiency*, List<*PricingPolicy*> *congestionCharges*, boolean *flagIncludeAccessEgress*)

getDestinationLAD

public String **getDestinationLAD**()

Gets trip destination zone (LAD), from tempro to LAD mapping (not from route nodes).

Returns Trip destination zone.

getDestinationLAD

public String **getDestinationLAD** (Map<Integer, String> *nodeToZoneMap*)
Gets trip destination zone (LAD), from tempro to LAD mapping (not from route nodes).

Parameters

- **nodeToZoneMap** – Mapping from nodes to zones.

Returns Trip destination zone.

getDestinationLadID

public int **getDestinationLadID** ()
Gets trip destination LAD zone ID.

Returns Trip destination zone LAD ID.

getDestinationTemproZone

public String **getDestinationTemproZone** ()
Gets trip destination tempro zone.

Returns Trip destination tempro zone.

getLength

public double **getLength** ()
Gets trip length (no separate access/egress for minor road trips).

Returns Trip length [in km].

getOriginLAD

public String **getOriginLAD** ()
Gets trip origin zone (LAD), from tempro to LAD mapping (not from route nodes).

Returns Trip origin zone.

getOriginLAD

public String **getOriginLAD** (Map<Integer, String> *nodeToZoneMap*)
Gets trip origin zone (LAD), from tempro to LAD mapping (not from route nodes).

Parameters

- **nodeToZoneMap** – Mapping from nodes to zones.

Returns Trip origin zone.

getOriginLadID

public int **getOriginLadID**()
Gets trip origin LAD zone ID.

Returns Origin zone LAD ID.

getOriginTemproZone

public String **getOriginTemproZone**()
Gets trip origin tempro zone code.

Returns Trip origin tempro zone code.

getTravelTime

public double **getTravelTime**(double *averageSpeed*)
Gets travel time for the minor trip.

Parameters

- **averageSpeed** – Average speed for a minor trip.

Returns Travel time.

getTravelTime

public double **getTravelTime**(double[] *linkTravelTime*, double *avgIntersectionDelay*, double[] *distanceFromTemproZoneToNearestNode*, double *averageAccessEgressSpeed*, boolean *flagIncludeAccessEgress*)

getZoning

public *Zoning* **getZoning**()
Getter for the zoning system.

Returns Zoning.

toString

public String **toString**()

TripTempro

public class **TripTempro** extends *Trip*
This class stores information about a performed trip (when using the Tempro zoning system).

Author Milan Lovric

Fields

zoning

public static *Zoning* **zoning**

Constructors

TripTempro

public **TripTempro**(*VehicleType vehicle*, *EngineType engine*, *Route route*, *TimeOfDay hour*, Integer *originTemproZoneID*, Integer *destinationTemproZoneID*, *Zoning zoning*)

Constructor for a passenger car trip using the Tempro zoning system. Origin and destination are integer IDs of Tempro zones.

Parameters

- **vehicle** – Vehicle type.
- **engine** – Engine type.
- **route** – Route.
- **hour** – Time of day.
- **originTemproZoneID** – Origin tempro zone ID.
- **destinationTemproZoneID** – Destination tempro zone ID.
- **zoning** – Zoning system.

TripTempro

public **TripTempro**(*VehicleType vehicle*, *EngineType engine*, *Route route*, *TimeOfDay hour*, Integer *originTemproZoneID*, Integer *destinationTemproZoneID*, *Zoning zoning*, int *multiplier*)

Constructor for a passenger car trip using the Tempro zoning system. Origin and destination are integer IDs of Tempro zones. Multiplier is used to store multiple instances of the same trip (vs creating multiple objects), thus reducing the memory footprint.

Parameters

- **vehicle** – Vehicle type.
- **engine** – Engine type.
- **route** – Route.
- **hour** – Time of day.
- **originTemproZoneID** – Origin tempro zone ID.
- **destinationTemproZoneID** – Destination tempro zone ID.
- **zoning** – Zoning system.
- **multiplier** – Multiplies the same trip.

Methods

getAccessEgressConsumption

protected Map<*EnergyType*, Double> **getAccessEgressConsumption**(double[] *linkTravelTime*, double[] *distanceFromTemproZoneToNearestNode*, double *averageAccessEgressSpeed*, Map<*VehicleType*, Map<*EngineType*, Map<*WebTAG*, Double>>> *energyConsumptionParameters*, Map<*VehicleType*, Map<*EngineType*, Double>> *relativeFuelEfficiency*)

getCO2emission

public Double **getCO2emission**(double[] *linkTravelTime*, double[] *distanceFromTemproZoneToNearestNode*, double *averageAccessEgressSpeed*, Map<*VehicleType*, Map<*EngineType*, Map<*WebTAG*, Double>>> *energyConsumptionParameters*, Map<*VehicleType*, Map<*EngineType*, Double>> *relativeFuelEfficiency*, Map<*EnergyType*, Double> *unitCO2Emissions*, boolean *flagIncludeAccessEgress*)

getConsumption

public Map<*EnergyType*, Double> **getConsumption**(double[] *linkTravelTime*, double[] *distanceFromTemproZoneToNearestNode*, double *averageAccessEgressSpeed*, Map<*VehicleType*, Map<*EngineType*, Map<*WebTAG*, Double>>> *energyConsumptionParameters*, Map<*VehicleType*, Map<*EngineType*, Double>> *relativeFuelEfficiency*, boolean *flagIncludeAccessEgress*)

getCost

public double **getCost**(double[] *linkTravelTime*, double[] *distanceFromTemproZoneToNearestNode*, double *averageAccessEgressSpeed*, Map<*EnergyType*, Double> *energyUnitCosts*, Map<*VehicleType*, Map<*EngineType*, Map<*WebTAG*, Double>>> *energyConsumptionParameters*, Map<*VehicleType*, Map<*EngineType*, Double>> *relativeFuelEfficiency*, List<*PricingPolicy*> *congestionCharges*, boolean *flagIncludeAccessEgress*)

getDestinationLAD

public String **getDestinationLAD**()

Gets trip destination zone (LAD), from Tempro to LAD mapping (not from route nodes).

Returns Trip destination zone.

getDestinationLAD

public String **getDestinationLAD** (Map<Integer, String> *nodeToZoneMap*)
Gets trip destination zone (LAD), from tempro to LAD mapping (not from route nodes).

Parameters

- **nodeToZoneMap** – Mapping from nodes to zones.

Returns Trip destination zone.

getDestinationLadID

public int **getDestinationLadID** ()
Gets trip destination LAD zone ID.

Returns Trip destination zone LAD ID.

getDestinationTemproZone

public String **getDestinationTemproZone** ()
Gets trip destination tempro zone.

Returns Trip destination tempro zone.

getLength

public double **getLength** ()
Gets trip length including access/egress (from Tempro centroid to node).

Returns Trip length [in km].

getOriginLAD

public String **getOriginLAD** ()
Gets trip origin zone (LAD), from Tempro to LAD mapping (not from route nodes).

Returns Trip origin zone.

getOriginLAD

public String **getOriginLAD** (Map<Integer, String> *nodeToZoneMap*)
Gets trip origin zone (LAD), from tempro to LAD mapping (not from route nodes).

Parameters

- **nodeToZoneMap** – Mapping from nodes to zones.

Returns Trip origin zone.

getOriginLadID

public int **getOriginLadID**()
Gets trip origin LAD zone ID.

Returns Origin zone LAD ID.

getOriginTemproZone

public String **getOriginTemproZone**()
Gets trip origin tempro zone code.

Returns Trip origin tempro zone code.

getTravelTime

public double **getTravelTime**(double[] *linkTravelTime*, double *avgIntersectionDelay*, double[] *distanceFromTemproZoneToNearestNode*, double *averageAccessEgressSpeed*, boolean *flagIncludeAccessEgress*)

getZoning

public *Zoning* **getZoning**()
Getter for the zoning system.

Returns Zoning.

1.2.7 nismod.transport.optimisation

SPSA

public class **SPSA**
Implements SPSA optimisation algorithm (Simultaneous Perturbation Stochastic Approximation). This version optimises the cells of the OD matrix. http://www.jhuapl.edu/SPSA/

Author Milan Lovric

Fields

THETA_MAX

public static final double **THETA_MAX**

THETA_MIN

public static final double **THETA_MIN**

Constructors

SPSA

public **SPSA**()

Methods

getLossFunctionEvaluations

public List<Double> **getLossFunctionEvaluations**()

Returns Loss function evaluations for all iterations.

getThetaEstimate

public *RealODMatrix* **getThetaEstimate**()
Getter function for the optimisation result (OD matrix).

Returns Estimated OD matrix.

initialise

public void **initialise**(*RoadNetworkAssignment* *rna*, Properties *props*, *RealODMatrix* *initialTheta*, double *a*, double *A*, double *c*, double *alpha*, double *gamma*)
Initialise the SPSA algorithm with starting values.

Parameters

- **rna** – Road network assignment.
- **props** – Parameters from the config file.
- **initialTheta** – Initial OD matrix.
- **a** – SPSA parameter.
- **A** – SPSA parameter.
- **c** – SPSA parameter.
- **alpha** – SPSA parameter.
- **gamma** – SPSA parameter.

lossFunction

public double **lossFunction**()
Calculate the loss function of the latest theta estimate (OD matrix).

Returns RMSN for the difference between volumes and traffic counts.

runSPSA

public void **runSPSA**(int *maxIterations*)

Run the algorithm.

Parameters

- **maxIterations** – Maximum number of iterations.

SPSA2

public class **SPSA2**

Implements SPSA optimisation algorithm (Simultaneous Perturbation Stochastic Approximation). This version of the algorithm keeps OD matrix constant, but optimises start and end node probabilities (the probability of a trip starting/ending at a particular node within LAD). http://www.jhuapl.edu/SPSA/

Author Milan Lovric

Fields

THETA_MAX

public static final double **THETA_MAX**

THETA_MIN

public static final double **THETA_MIN**

Constructors

SPSA2

public **SPSA2**()

Methods

getLossFunctionEvaluations

public List<Double> **getLossFunctionEvaluations**()

Returns Loss function evaluations for all iterations.

initialise

public void **initialise**(*RoadNetworkAssignment rna*, Properties *props*, *ODMatrixMultiKey odm*, HashMap<Integer, Double> *initialThetaStart*, HashMap<Integer, Double> *initialThetaEnd*, double *a*, double *A*, double *c*, double *alpha*, double *gamma*)

Initialise the SPSA algorithm with starting values.

Parameters

- **rna** – Road network assignment
- **props** – Parameters from the config file.
- **odm** – Origin-destination matrix.
- **initialThetaStart** – Initial start node probabilities.
- **initialThetaEnd** – Initial end node probabilities.
- **a** – SPSA parameter.
- **A** – SPSA parameter.
- **c** – SPSA parameter.
- **alpha** – SPSA parameter.
- **gamma** – SPSA parameter.

lossFunction

public double **lossFunction**()
Calculate the loss function of the latest theta estimate (OD matrix).

Returns RMSN for the difference between volumes and traffic counts.

runSPSA

public void **runSPSA**(int *maxIterations*)
Run the algorithm.

Parameters

- **maxIterations** – Maximum number of iterations.

SPSA3

public class **SPSA3**
Implements SPSA optimisation algorithm (Simultaneous Perturbation Stochastic Approximation). This version optimises OD matrix and start/end node probabilities simultaneously. http://www.jhuapl.edu/SPSA/

Author Milan Lovric

Fields

THETA_MAX_FLOW

public static final double **THETA_MAX_FLOW**

THETA_MAX_PROBABILITY

public static final double **THETA_MAX_PROBABILITY**

THETA_MIN_FLOW

public static final double **THETA_MIN_FLOW**

THETA_MIN_PROBABILITY

public static final double **THETA_MIN_PROBABILITY**

Constructors

SPSA3

public **SPSA3** ( )

Methods

getLossFunctionEvaluations

public List<Double> **getLossFunctionEvaluations** ( )
Getter function for loss function evaluations for all iterations.

Returns Loss function evaluations for all iterations.

getThetaEstimate

public *RealODMatrix* **getThetaEstimate** ( )
Getter function for the optimisation result (OD matrix).

Returns Estimated OD matrix.

getThetaEstimateEnd

public HashMap<Integer, Double> **getThetaEstimateEnd** ( )
Getter function for the optimisation result (end nodes probabilities).

Returns Estimated end nodes probabilities.

getThetaEstimateStart

public HashMap<Integer, Double> **getThetaEstimateStart** ( )
Getter function for the optimisation result (start nodes probabilities).

Returns Estimated start nodes probabilities.

initialise

public void **initialise**(*RoadNetworkAssignment* *rna*, *RouteSetGenerator* *rsg*, Properties *routeChoiceParams*, *RealODMatrix* *initialTheta*, HashMap<Integer, Double> *initialThetaStart*, HashMap<Integer, Double> *initialThetaEnd*, double *a1*, double *A1*, double *c1*, double *a2*, double *A2*, double *c2*, double *alpha*, double *gamma*)

Initialise the SPSA algorithm with starting values.

Parameters

- **rna** – Road network assignment.
- **rsg** – Route set generator.
- **routeChoiceParams** – Route choice parameters.
- **initialTheta** – Initial OD matrix.
- **initialThetaStart** – Initial start node probabilities.
- **initialThetaEnd** – Initial end node probabilities.
- **a1** – SPSA parameter for OD estimation.
- **A1** – SPSA parameter for OD estimation.
- **c1** – SPSA parameter for OD estimation.
- **a2** – SPSA parameter for nodes probability estimation.
- **A2** – SPSA parameter for nodes probability estimation.
- **c2** – SPSA parameter for nodes probability estimation.
- **alpha** – SPSA parameter.
- **gamma** – SPSA parameter.

lossFunction

public double **lossFunction**()

Calculate the loss function of the latest theta estimate (OD matrix).

Returns RMSN for the difference between volumes and traffic counts.

runSPSA

public void **runSPSA**(int *maxIterations*)

Run the algorithm.

Parameters

- **maxIterations** – Maximum number of iterations.

saveNodeProbabilities

public void **saveNodeProbabilities**(String *outputFile*)

Saves node probabilities to an output file.

Parameters

- **outputFile** – Output file name (with path).

SPSA4

public class **SPSA4**

Implements SPSA optimisation algorithm (Simultaneous Perturbation Stochastic Approximation). This version optimises Tempro level OD matrix. http://www.jhuapl.edu/SPSA/

Author Milan Lovric

Fields

THETA_MAX

public static final double **THETA_MAX**

THETA_MIN

public static final double **THETA_MIN**

Constructors

SPSA4

public **SPSA4** (Properties *props*)

Methods

getLossFunctionEvaluations

public List<Double> **getLossFunctionEvaluations** ()

Returns Loss function evaluations for all iterations.

getThetaEstimate

public *RealODMatrixTempro* **getThetaEstimate** ()

Getter function for the optimisation result (OD matrix).

Returns Estimated OD matrix.

initialise

public void **initialise** (*RoadNetworkAssignment* *rna*, *Zoning* *zoning*, *RouteSetGenerator* *rsg*, *RealODMatrixTempro* *initialTheta*, double *a*, double *A*, double *c*, double *alpha*, double *gamma*)

Initialise the SPSA algorithm with starting values.

Parameters

- **rna** – Road network assignment.
- **zoning** – Zoning system for tempro zones.
- **rsg** – Route set generator with routes to be used in assignment.
- **initialTheta** – Initial OD matrix.
- **a** – SPSA parameter.
- **A** – SPSA parameter.
- **c** – SPSA parameter.
- **alpha** – SPSA parameter.
- **gamma** – SPSA parameter.

lossFunction

public double **lossFunction**()
Calculate the loss function of the latest theta estimate (OD matrix).

Returns Loss function.

runSPSA

public void **runSPSA**(int *maxIterations*)
Run the algorithm.

Parameters

- **maxIterations** – Maximum number of iterations.

SPSA5

public class **SPSA5**
Implements SPSA optimisation algorithm (Simultaneous Perturbation Stochastic Approximation). This version optimises Tempro level OD matrix. http://www.jhuapl.edu/SPSA/

Author Milan Lovric

Fields

THETA_MAX

public static final double **THETA_MAX**

THETA_MIN

public static final double **THETA_MIN**

Constructors

SPSA5

public **SPSA5** (Properties *props*)

Methods

getLossFunctionEvaluations

public List<Double> **getLossFunctionEvaluations** ()

Returns Loss function evaluations for all iterations.

getThetaEstimate

public *RealODMatrixTempro* **getThetaEstimate** ()
Getter function for the optimisation result (OD matrix).

Returns Estimated OD matrix.

initialise

public void **initialise** (*RoadNetworkAssignment* *rna*, *Zoning* *zoning*, *RouteSetGenerator* *rsg*, *RealODMatrixTempro* *initialTheta*, double *a*, double *A*, double *c*, double *alpha*, double *gamma*)
Initialise the SPSA algorithm with starting values.

Parameters

- **rna** – Road network assignment.
- **zoning** – Zoning system for tempro zones.
- **rsg** – Route set generator with routes to be used in assignment.
- **initialTheta** – Initial OD matrix.
- **a** – SPSA parameter.
- **A** – SPSA parameter.
- **c** – SPSA parameter.
- **alpha** – SPSA parameter.
- **gamma** – SPSA parameter.

lossFunction

public double **lossFunction** ()
Calculate the loss function of the latest theta estimate (OD matrix).

Returns Loss function.

runSPSA

public void **runSPSA**(int *maxIterations*)
Run the algorithm.

Parameters

- **maxIterations** – Maximum number of iterations.

1.2.8 nismod.transport.rail

RailDemandModel

public class **RailDemandModel**

Fields

baseYear

public static int **baseYear**

Constructors

RailDemandModel

public **RailDemandModel**(String *baseYearRailStationUsageFile*, String *populationFile*, String *GVAFile*, String *elasticitiesFile*, String *railStationJourneyFaresFile*, String *railStationGeneralisedJourneyTimesFile*, String *carZonalJourneyCostsFile*, String *railTripRatesFile*, List<*Intervention*> *interventions*, Properties *props*)
Constructor for the rail demand model.

Parameters

- **baseYearRailStationUsageFile** – Base year rail station usage file (demand).
- **populationFile** – Population file.
- **GVAFile** – GVA file.
- **elasticitiesFile** – Elasticites file.
- **railStationJourneyFaresFile** – Rail fares file.
- **railStationGeneralisedJourneyTimesFile** – GJT file.
- **carZonalJourneyCostsFile** – Zonal car journey costs file.
- **railTripRatesFile** – Rail trip rates file.
- **interventions** – List of interventions.
- **props** – Properties.

Throws

- **IOException** –
- **FileNotFoundException** –

Methods

addNLCofDevelopedStation

public void **addNLCofDevelopedStation**(int *NLC*)
Adds NLC of a newly built rail station.

Parameters

- **NLC** – Id of a newly built rail station.

addYearOfDevelopment

public void **addYearOfDevelopment**(int *year*)
Adds a year in which a new rail station is built.

Parameters

- **year** – Year in which a new rail station is built.

getRailStationDemand

public *RailStationDemand* **getRailStationDemand**(int *year*)
Getter method for the passenger rail station demand in a given year.

Parameters

- **year** – Year for which the demand is requested.

Returns Rail station demand with total passenger counts (entry + exit).

predictAndSaveRailwayDemands

public void **predictAndSaveRailwayDemands**(int *toYear*, int *fromYear*)
Predicts rail station demand (total passenger counts at each station) up to toYear (if flag is true, also intermediate years) and saves results.

Parameters

- **toYear** – The final year for which the demand is predicted.
- **fromYear** – The year from which the predictions are made.

predictRailwayDemand

public void **predictRailwayDemand**(int *predictedYear*, int *fromYear*)
Predicts passenger railway demand (passenger counts at each station). Rail station demand for fromYear needs to be contained in the memory.

Parameters

- **predictedYear** – The year for which the demand is predicted.
- **fromYear** – The year from which demand the prediction is made.

predictRailwayDemandUsingResultsOfFromYear

public void **predictRailwayDemandUsingResultsOfFromYear** (int *predictedYear*, int *fromYear*)
Predicts passenger railway demand (passenger counts at each station). Uses already existing results of the fromYear, from the output folder.

Parameters

- **predictedYear** – The year for which the demand is predicted.
- **fromYear** – The year from which demand the prediction is made.

predictRailwayDemands

public void **predictRailwayDemands** (int *toYear*, int *baseYear*)
Predicts rail station demand (total passenger counts at each station) for all years from baseYear to toYear.

Parameters

- **toYear** – The final year for which the demand is predicted.
- **baseYear** – The base year from which the predictions are made.

printNLCsOfNewStations

public void **printNLCsOfNewStations** ()
Prints NLCs of new rail stations.

printYearsOfNewStations

public void **printYearsOfNewStations** ()
Prints years in which development of new rail stations takes place.

saveAllResults

public void **saveAllResults** (int *year*)
Saves all results into the output folder.

Parameters

- **year** – Year of the data.

saveRailStationDemand

public void **saveRailStationDemand** (int *year*, String *outputFile*)
Saves rail station demand to an output file.

Parameters

- **year** – Year of the data.
- **outputFile** – Output file name (with path).

saveZonalRailStationDemand

public void **saveZonalRailStationDemand**(int *year*, String *outputFile*)
Saves zonal rail station demand to an output file.

Parameters

- **year** – Year of the data.
- **outputFile** – Output file name (with path).

setRailStationDemand

public void **setRailStationDemand**(int *year*, *RailStationDemand* *demand*)
Setter method for the passenger rail station demand in a given year.

Parameters

- **year** – Year for which the demand is set.
- **demand** – Rail station demand.

RailDemandModel.ElasticityArea

public static enum **ElasticityArea**

Enum Constants

LT

public static final *RailDemandModel*.*ElasticityArea* **LT**

OTHER

public static final *RailDemandModel*.*ElasticityArea* **OTHER**

PTE

public static final *RailDemandModel*.*ElasticityArea* **PTE**

SE

public static final *RailDemandModel*.*ElasticityArea* **SE**

RailDemandModel.ElasticityTypes

public static enum **ElasticityTypes**

Enum Constants

COST_CAR

public static final *RailDemandModel.ElasticityTypes* **COST_CAR**

COST_RAIL

public static final *RailDemandModel.ElasticityTypes* **COST_RAIL**

GVA

public static final *RailDemandModel.ElasticityTypes* **GVA**

POPULATION

public static final *RailDemandModel.ElasticityTypes* **POPULATION**

TIME

public static final *RailDemandModel.ElasticityTypes* **TIME**

RailStation

public class **RailStation**
This class stores information about a rail station.

Author Milan Lovric

Constructors

RailStation

public **RailStation**(int *nlc*, *RailModeType mode*, String *stationName*, String *naptanName*, int *easting*, int *northing*, int *yearUsage*, double *dayUsage*, int *runDays*, String *ladCode*, String *ladName*, *ElasticityArea area*)
Constructor for a station.

Parameters

- **nlc** – National Location Code.
- **mode** – Which mode is served by this station.
- **stationName** – Station name.
- **naptanName** – Longer name from NaPTAN.
- **easting** – Easting coordinate.
- **northing** – Northing coordinate.

- **yearUsage** – Yearly station usage (entries and exits combined).
- **dayUsage** – Daily station usage (yearly usage divided by the number of operational days in a year).
- **runDays** – The number of operational days in a year.
- **ladCode** – LAD code of the zone in which the station is located.
- **ladName** – LAD name of the zone in which the station is located.
- **area** – Elasticity area in which the station is located.

RailStation

public **RailStation**(*RailStation station*)
Constructor for a station.

Parameters

- **station** – Rail station which data is going to be copied.

Methods

getArea

public *ElasticityArea* **getArea**()
Getter method for the elasticity area in which station is located.

Returns LAD name.

getDayUsage

public double **getDayUsage**()
Getter method for daily usage.

Returns Daily usage.

getEasting

public int **getEasting**()
Getter method for easting.

Returns Easting.

getLADCode

public String **getLADCode**()
Getter method for the LAD code in which station is located.

Returns LAD code.

getLADName

public String **getLADName**()
Getter method for the LAD name in which station is located.

Returns LAD name.

getMode

public *RailModeType* **getMode**()
Getter method for the rail mode type.

Returns Rail mode type.

getNLC

public int **getNLC**()
Getter method for the NLC (National Location Code) of the station.

Returns NLC.

getNaPTANName

public String **getNaPTANName**()
Getter method for the station NaPTAN name.

Returns NaPTAN name.

getName

public String **getName**()
Getter method for the station name.

Returns Name.

getNorthing

public int **getNorthing**()
Getter method for Northing.

Returns Northing.

getRunDays

public int **getRunDays**()
Getter method for number of operational days.

Returns Number of operaiontal days.

getYearlyUsage

public int **getYearlyUsage**()
Getter method for yearly usage.

Returns Yearly usage.

setDailyUsage

public void **setDailyUsage**(double *usage*)
Setter method for daily usage.

Parameters

- **usage** – Daily usage.

setYearlyUsage

public void **setYearlyUsage**(int *usage*)
Setter method for yearly usage.

Parameters

- **usage** – Yearly usage.

toString

public String **toString**()

RailStation.RailModeType

public static enum **RailModeType**

Enum Constants

DLR

public static final *RailStation.RailModeType* **DLR**

LRAIL

public static final *RailStation.RailModeType* **LRAIL**

NRAIL

public static final *RailStation.RailModeType* **NRAIL**

TUBE

public static final *RailStation.RailModeType* **TUBE**

RailStationDemand

public class **RailStationDemand**
This class stores passenger rail demand = station usage data (entries + exists).

Author Milan Lovric

Constructors

RailStationDemand

public **RailStationDemand** (String *fileName*)

RailStationDemand

public **RailStationDemand** (List<String> *header*)
Constructor for empty rail station demand.

Parameters

- **header** –

Methods

addStation

public void **addStation** (*RailStation station*)
Add a rail station data to the rail demand (overwrites existing one).

Parameters

- **station** –

calculateDailyZonalUsageAverage

public HashMap<String, Double> **calculateDailyZonalUsageAverage** ()
Calculates daily zonal usage (the average for all stations within LAD).

Returns Daily zonal usage per station.

calculateDailyZonalUsageTotal

public HashMap<String, Double> **calculateDailyZonalUsageTotal** ()
Calculates daily zonal usage (the sum for all stations within LAD).

Returns Daily zonal usage.

calculateYearlyZonalUsageAverage

public HashMap<String, Integer> **calculateYearlyZonalUsageAverage**()
Calculates yearly zonal usage (the average for all stations within LAD).

Returns Yearly zonal usage per station.

calculateYearlyZonalUsageTotal

public HashMap<String, Integer> **calculateYearlyZonalUsageTotal**()
Calculates yearly zonal usage (the sum for all stations within LAD).

Returns Yearly zonal usage.

createListOfStationsWithinEachLAD

public HashMap<String, List<*RailStation*>> **createListOfStationsWithinEachLAD**()
Creates a list of stations within each LAD.

Returns List of stations within each LAD.

getHeader

public List<String> **getHeader**()
Getter method for the header.

Returns header

getRailDemandList

public List<*RailStation*> **getRailDemandList**()
Getter method for the rail demand list.

Returns Rail demand list

getRailDemandMap

public Map<Integer, *RailStation*> **getRailDemandMap**()
Getter method for the rail demand map.

Returns Rail demand map.

printRailDemand

public void **printRailDemand**(String *message*)
Print rail demand.

Parameters

- **message** – Message to print before the demand.

printRailDemandNLCSorted

public void **printRailDemandNLCSorted**(String *message*)
Print rail demand sorted on NLC.

Parameters

- **message** – Message to print before the demand.

printRailDemandNameSorted

public void **printRailDemandNameSorted**(String *message*)
Print rail demand sorted on station name.

Parameters

- **message** – Message to print before the demand.

printRailDemandUsageSorted

public void **printRailDemandUsageSorted**(String *message*)
Print rail demand sorted on station usage.

Parameters

- **message** – Message to print before the demand.

removeStation

public boolean **removeStation**(int *NLC*)
Remove station with a given NLC code.

Parameters

- **NLC** – Station code.

Returns true if station existed in demand and was successfully removed.

saveRailStationDemand

public void **saveRailStationDemand**(int *year*, String *outputFile*)
Saves rail station demand to an output file.

Parameters

- **year** – Year of the data.
- **outputFile** – Output file name (with path).

saveZonalRailStationDemand

public void **saveZonalRailStationDemand**(int *year*, String *outputFile*)
Saves zonal rail station demand to an output file.

Parameters

- **year** – Year of the data.
- **outputFile** – Output file name (with path).

sortStationsOnNLC

public void **sortStationsOnNLC**()
Sorts stations on NLC in an ascending order.

sortStationsOnName

public void **sortStationsOnName**()
Sorts stations on station name in an ascending order.

sortStationsOnUsage

public void **sortStationsOnUsage**()
Sorts stations on usage in a descending order.

1.2.9 nismod.transport.scripts

ArcAnalysis

public class **ArcAnalysis**
This scripts parses the outputs for the CaMkOx case study (adjust paths as needed).

Author Milan Lovric

Fields

TABLE_FONT

public static final Font **TABLE_FONT**

arcZones

static final List<String> **arcZones**

Methods

main

public static void **main**(String[] *args*)

RunArcRail

public class **RunArcRail**

Methods

main

public static void **main**(String[] *args*)

1.2.10 nismod.transport.showcase

CapacityUtilisationLegend

public class **CapacityUtilisationLegend** extends JPanel
Capacity utilisation legend to include in the dashboards.

Author Milan Lovric

Fields

LEGEND_FONT

public static final Font **LEGEND_FONT**

Constructors

CapacityUtilisationLegend

public **CapacityUtilisationLegend**()
Create the panel.

CongestionChargingDashboard

public class **CongestionChargingDashboard** extends JFrame
Dashboard for the road expansion policy intervention.

Author Milan Lovric

Fields

AFTER_MAP_X

public static final int **AFTER_MAP_X**

AFTER_MAP_Y

public static final int **AFTER_MAP_Y**

AFTER_TABLE_SHIFT

public static final int **AFTER_TABLE_SHIFT**

BEFORE_MAP_X

public static final int **BEFORE_MAP_X**

BEFORE_MAP_Y

public static final int **BEFORE_MAP_Y**

BETWEEN_MAP_SPACE

public static final int **BETWEEN_MAP_SPACE**

COMBOBOX_BORDER

public static final Border **COMBOBOX_BORDER**

EMPTY_BORDER

public static final Border **EMPTY_BORDER**

LEFT_MARGIN

public static final int **LEFT_MARGIN**

MAP_HEIGHT

public static final int **MAP_HEIGHT**

MAP_WIDTH

public static final int **MAP_WIDTH**

MATRIX_SCALING_FACTOR

public static final double **MATRIX_SCALING_FACTOR**

OPACITY_FACTOR

public static final double **OPACITY_FACTOR**

RUN_BUTTON_BORDER

public static final Border **RUN_BUTTON_BORDER**

SECOND_MARGIN

public static final int **SECOND_MARGIN**

TABLE_BORDER

public static final Border **TABLE_BORDER**

TABLE_FONT

public static final Font **TABLE_FONT**

TABLE_LABEL_WIDTH

public static final int **TABLE_LABEL_WIDTH**

TABLE_ROW_HEIGHT

public static final int **TABLE_ROW_HEIGHT**

TOTAL_DEMAND_BORDER

public static final Border **TOTAL_DEMAND_BORDER**

Constructors

CongestionChargingDashboard

public **CongestionChargingDashboard**()
Create the frame.

Throws

- **IOException** – if any.
- **AWTException** – if any.

Methods

main

public static void **main**(String[] *args*)
Launch the application.

Parameters

- **args** – Arguments.

LandingGUI

public class **LandingGUI**
Main GUI for the Show-case Demo.

Author Milan Lovric

Fields

BUTTON_HEIGHT

public static final int **BUTTON_HEIGHT**

BUTTON_SPACE

public static final int **BUTTON_SPACE**

BUTTON_WIDTH

public static final int **BUTTON_WIDTH**

BUTTON_X

public static final int **BUTTON_X**

BUTTON_Y

public static final int **BUTTON_Y**

CREDITS_FONT_SIZE

public static final int **CREDITS_FONT_SIZE**

DARK_GRAY

public static final Color **DARK_GRAY**

DASHBOARD

public static final Color **DASHBOARD**

ICON1_HEIGHT

public static final int **ICON1_HEIGHT**

ICON1_WIDTH

public static final int **ICON1_WIDTH**

ICON2_HEIGHT

public static final int **ICON2_HEIGHT**

ICON2_WIDTH

public static final int **ICON2_WIDTH**

ICON3_HEIGHT

public static final int **ICON3_HEIGHT**

ICON3_WIDTH

public static final int **ICON3_WIDTH**

LABEL1_Y

public static final int **LABEL1_Y**

LABEL2_Y

public static final int **LABEL2_Y**

LABEL3_Y

public static final int **LABEL3_Y**

LABEL_HEIGHT

public static final int **LABEL_HEIGHT**

LABEL_WIDTH

public static final int **LABEL_WIDTH**

LABEL_X

public static final int **LABEL_X**

LIGHT_GRAY

public static final Color **LIGHT_GRAY**

MAIN_TITLE_FONT_SIZE

public static final int **MAIN_TITLE_FONT_SIZE**

MID_GRAY

public static final Color **MID_GRAY**

PASTEL_BLUE

public static final Color **PASTEL_BLUE**

PASTEL_GREEN

public static final Color **PASTEL_GREEN**

PASTEL_YELLOW

public static final Color **PASTEL_YELLOW**

SCREEN_HEIGHT

public static final int **SCREEN_HEIGHT**

SCREEN_WIDTH

public static final int **SCREEN_WIDTH**

SUBTITLE_FONT_SIZE

public static final int **SUBTITLE_FONT_SIZE**

TOOLBAR

public static final Color **TOOLBAR**

counter

public static int **counter**

Constructors

LandingGUI

public **LandingGUI**()
Create the application.

Methods

main

public static void **main**(String[] *args*)
Launch the application.

Parameters

- **args** – Arguments.

NetworkVisualiserDemo

public class **NetworkVisualiserDemo**
For visualising the road network.

Author Milan Lovric

Fields

ROAD_LINK_WIDTH

public static final float **ROAD_LINK_WIDTH**

Constructors

NetworkVisualiserDemo

protected **NetworkVisualiserDemo**()

Methods

getMap

public static JFrame **getMap**(URL *zonesUrl*, URL *networkUrl*, URL *nodesUrl*, URL *AADFurl*, String *mapTitle*, String *linkDataLabel*)
Visualises the road network with link data.

Parameters

- **zonesUrl** – Url for the zones shapefile.
- **networkUrl** – Url for the road network shapefile.
- **nodesUrl** – Url for the nodes shapefile.
- **AADFurl** – Url for the traffic counts shapefile.
- **mapTitle** – Map title for the window.
- **linkDataLabel** – Label describing the link data used.

Throws

- **IOException** – if any.

Returns JFrame with the map.

main

public static void **main** (String[] *args*)

Parameters

- **args** – Arguments.

Throws

- **IOException** – if any.

visualise

public static void **visualise** (*RoadNetwork roadNetwork*, String *mapTitle*)

Visualises the road network as loaded from the shapefiles.

Parameters

- **roadNetwork** – Road network.
- **mapTitle** – Map title for the window.

Throws

- **IOException** – if any.

visualise

public static JFrame **visualise** (URL *zonesUrl*, URL *networkUrl*, URL *nodesUrl*, URL *AADFurl*, String *mapTitle*, String *linkDataLabel*)

Visualises the road network with link data.

Parameters

- **zonesUrl** – Url for the zones shapefile.
- **networkUrl** – Url for the road network shapefile.
- **nodesUrl** – Url for the nodes shapefile.
- **AADFurl** – Url for the traffic counts shapefile.

- **mapTitle** – Map title for the window.
- **linkDataLabel** – Label describing the link data used.

Throws

- **IOException** – if any.

Returns JFrame with the map.

visualise

public static JFrame **visualise**(*RoadNetwork roadNetwork*, String *mapTitle*, Map<Integer, Double> *linkData*, String *linkDataLabel*, String *shapefilePath*)

Visualises the road network with dailyVolume.

Parameters

- **roadNetwork** – Road network.
- **mapTitle** – Map title for the window.
- **linkData** – Data used to classify and colour road links.
- **linkDataLabel** – Label describing the link data used.
- **shapefilePath** – The path to the shapefile into which data will be stored.

Throws

- **IOException** – if any.

Returns JFrame with the map.

visualise

public static JFrame **visualise**(*RoadNetwork roadNetwork*, String *mapTitle*, Map<Integer, Double> *linkData*, String *linkDataLabel*, String *shapefilePath*, URL *congestionChargeZoneUrl*)

Visualises the road network with link data and congestion charging zone.

Parameters

- **roadNetwork** – Road network.
- **mapTitle** – Map title for the window.
- **linkData** – Data used to classify and colour road links.
- **linkDataLabel** – Label describing the link data used.
- **shapefilePath** – The path to the shapefile into which data will be stored.
- **congestionChargeZoneUrl** – The path to the shapefile with the congestion charge zone boundary.

Throws

- **IOException** – if any.

Returns JFrame with the map.

RoadDevelopmentDashboard

public class **`RoadDevelopmentDashboard`** extends JFrame
Dashboard for the road development policy intervention.

Author Milan Lovric

Fields

AFTER_MAP_X

public static final int **`AFTER_MAP_X`**

AFTER_MAP_Y

public static final int **`AFTER_MAP_Y`**

BEFORE_MAP_X

public static final int **`BEFORE_MAP_X`**

BEFORE_MAP_Y

public static final int **`BEFORE_MAP_Y`**

BETWEEN_MAP_SPACE

public static final int **`BETWEEN_MAP_SPACE`**

COMBOBOX_BORDER

public static final Border **`COMBOBOX_BORDER`**

EMPTY_BORDER

public static final Border **`EMPTY_BORDER`**

LEFT_MARGIN

public static final int **`LEFT_MARGIN`**

MAP_HEIGHT

public static final int **`MAP_HEIGHT`**

MAP_WIDTH

public static final int **MAP_WIDTH**

MATRIX_SCALING_FACTOR

public static final double **MATRIX_SCALING_FACTOR**

OPACITY_FACTOR

public static final double **OPACITY_FACTOR**

RUN_BUTTON_BORDER

public static final Border **RUN_BUTTON_BORDER**

SECOND_MARGIN

public static final int **SECOND_MARGIN**

TABLE_BORDER

public static final Border **TABLE_BORDER**

TABLE_FONT

public static final Font **TABLE_FONT**

TABLE_LABEL_WIDTH

public static final int **TABLE_LABEL_WIDTH**

TABLE_ROW_HEIGHT

public static final int **TABLE_ROW_HEIGHT**

TOTAL_DEMAND_BORDER

public static final Border **TOTAL_DEMAND_BORDER**

Constructors

RoadDevelopmentDashboard

public **RoadDevelopmentDashboard**()
Create the frame.

Throws

- **IOException** – if any.
- **AWTException** – if any.

Methods

main

public static void **main**(String[] *args*)
Launch the application.

Parameters

- **args** – Arguments.

RoadExpansionDashboard

public class **RoadExpansionDashboard** extends JFrame
Dashboard for the road expansion policy intervention.

Author Milan Lovric

Fields

AFTER_MAP_X

public static final int **AFTER_MAP_X**

AFTER_MAP_Y

public static final int **AFTER_MAP_Y**

BEFORE_MAP_X

public static final int **BEFORE_MAP_X**

BEFORE_MAP_Y

public static final int **BEFORE_MAP_Y**

BETWEEN_MAP_SPACE

public static final int **BETWEEN_MAP_SPACE**

COMBOBOX_BORDER

public static final Border **COMBOBOX_BORDER**

EMPTY_BORDER

public static final Border **EMPTY_BORDER**

FLAG_EXPAND_MORE_LINKS

public static boolean **FLAG_EXPAND_MORE_LINKS**

LEFT_MARGIN

public static final int **LEFT_MARGIN**

MAP_HEIGHT

public static final int **MAP_HEIGHT**

MAP_WIDTH

public static final int **MAP_WIDTH**

MATRIX_SCALING_FACTOR

public static final double **MATRIX_SCALING_FACTOR**

OPACITY_FACTOR

public static final double **OPACITY_FACTOR**

RUN_BUTTON_BORDER

public static final Border **RUN_BUTTON_BORDER**

SECOND_MARGIN

public static final int **SECOND_MARGIN**

TABLE_BORDER

public static final Border **TABLE_BORDER**

TABLE_FONT

public static final Font **TABLE_FONT**

TABLE_LABEL_WIDTH

public static final int **TABLE_LABEL_WIDTH**

TABLE_ROW_HEIGHT

public static final int **TABLE_ROW_HEIGHT**

TOTAL_DEMAND_BORDER

public static final Border **TOTAL_DEMAND_BORDER**

Constructors

RoadExpansionDashboard

public **RoadExpansionDashboard**()
Create the frame.

Throws

- **IOException** – if any.
- **AWTException** – if any.

Methods

main

public static void **main**(String[] *args*)
Launch the application.

Parameters

- **args** – Arguments.

TableChangeLegend

public class **TableChangeLegend** extends JPanel
Table change legend (vertical) to include in the dashboards.

Author Milan Lovric

Fields

LEGEND_FONT

public static final Font **LEGEND_FONT**

Constructors

TableChangeLegend

public **TableChangeLegend**()
Create the panel.

TableChangeLegendHorizontal

public class **TableChangeLegendHorizontal** extends JPanel
Table change legend (horizontal) to include in the dashboards.

Author Milan Lovric

Fields

LEGEND_FONT

public static final Font **LEGEND_FONT**

Constructors

TableChangeLegendHorizontal

public **TableChangeLegendHorizontal**()
Create the panel.

1.2.11 nismod.transport.utility

ConfigReader

public class **ConfigReader**
Configuration file reader.

Author Milan Lovric

Constructors

ConfigReader

public **ConfigReader**()

Methods

getProperties

public static Properties **getProperties**(String *configFile*)
Reads properties from the configuration file.

Parameters

- **configFile** – Path to the configuration file.

Returns Loaded properties.

InputFileReader

public class **InputFileReader**
InputFileReader reads input files and provides them as various data structures required by other classes.

Author Milan Lovric

Constructors

InputFileReader

public **InputFileReader**()

Methods

readAVFractionsFile

public static HashMap<Integer, Map<*VehicleType*, Double>> **readAVFractionsFile**(String *fileName*)
Reads autonomous vehicles fractions file.

Parameters

- **fileName** – File name.

Returns Map with predictions of autonomous vehicles fractions.

readAirElasticitiesFile

public static Map<*AirDemandModel.ElasticityTypes*, Double> **readAirElasticitiesFile**(String *fileName*)
Reads air elasticities file.

Parameters

- **fileName** – File name.

Returns Map with elasticity parameters.

readAirportFareIndexFile

public static HashMap<Integer, HashMap<String, Double>> **readAirportFareIndexFile**(String *fileName*)

Reads airport fare index file.

Parameters

- **fileName** – File name.

Returns Map with airport fare indices.

readDomesticAirportsFile

public static Map<String, *Airport*> **readDomesticAirportsFile**(String *fileName*)

Reads domestic airports file.

Parameters

- **fileName** – File name.

Returns Mapping between IATA code and airport information.

readElasticitiesFile

public static Map<*ElasticityTypes*, Double> **readElasticitiesFile**(String *fileName*)

Reads elasticities file.

Parameters

- **fileName** – File name.

Returns Map with elasticity parameters.

readEnergyConsumptionParamsFile

public static Map<*VehicleType*, Map<*EngineType*, Map<*WebTAG*, Double>>> **readEnergyConsumptionParamsFile**(String *fileName*)

Reads engine type fractions file.

Parameters

- **fileName** – File name.

Returns Map with engine type fractions.

readEnergyUnitCostsFile

public static HashMap<Integer, Map<*EnergyType*, Double>> **readEnergyUnitCostsFile**(String *fileName*)

Reads energy unit costs file.

Parameters

- **fileName** – File name.

Returns Map with energy unit costs.

readEngineTypeFractionsFile

public static HashMap<Integer, Map<*VehicleType*, Map<*EngineType*, Double>>> **readEngineTypeFractionsFile**(String *fileName*)

Reads engine type fractions file.

Parameters

- **fileName** – File name.

Returns Map with engine type fractions.

readFreightTripRatesFile

public static HashMap<Integer, Map<*VehicleType*, HashMap<Integer, Double>>> **readFreightTripRatesFile**(String *fileName*)

Reads freight trip rates file.

Parameters

- **fileName** – File name.

Returns Map with yearly zonal trip rates for freight vehicles.

readGVAFile

public static HashMap<Integer, HashMap<String, Double>> **readGVAFile**(String *fileName*)

Reads GVA file.

Parameters

- **fileName** – File name.

Returns Map with GVA data.

readInternationalAirportsFile

public static Map<String, *Airport*> **readInternationalAirportsFile**(String *fileName*)

Reads international airports file.

Parameters

- **fileName** – File name.

Returns Mapping between IATA code and airport information.

readLinkTravelTimeFile

public static Map<*TimeOfDay*, Map<Integer, Double>> **readLinkTravelTimeFile**(int *year*, String *fileName*)

Reads link travel time file.

Parameters

- **year** – Year of the assignment.

- **fileName** – File name.

Returns Link travel time per time of day.

readPassengerTripRatesFile

public static HashMap<Integer, HashMap<String, Double>> **readPassengerTripRatesFile**(String *fileName*)

Reads passenger trip rates file (zonal).

Parameters

- **fileName** – File name.

Returns Map with yearly zonal trip rates.

readPopulationFile

public static HashMap<Integer, HashMap<String, Integer>> **readPopulationFile**(String *fileName*)

Reads population file.

Parameters

- **fileName** – File name.

Returns Map with population data.

readRailElasticitiesFile

public static Map<*RailDemandModel*.*ElasticityTypes*, Map<*RailDemandModel*.*ElasticityArea*, Double>> **readRailElasticitie**

Reads rail elasticities file.

Parameters

- **fileName** – File name.

Returns Map with elasticity parameters.

readRailStationCostsFile

public static HashMap<Integer, HashMap<Integer, Double>> **readRailStationCostsFile**(String *fileName*)

Reads rail station costs file.

Parameters

- **fileName** – File name.

Returns Map with rail journey costs.

readRelativeFuelEfficiencyFile

public static HashMap<Integer, Map<*VehicleType*, Map<*EngineType*, Double>>> **readRelativeFuelEfficiencyFile**(String *fileName*)

Reads relative fuel efficiency file.

Parameters

- **fileName** – File name.

Returns Map with relative fuel efficiency.

readTimeOfDayDistributionFile

public static Map<Integer, Map<*TimeOfDay*, Double>> **readTimeOfDayDistributionFile**(String *fileName*)

Reads time of day distribution file for passenger car vehicles.

Parameters

- **fileName** – File name.

Returns Time of day distribution.

readTimeOfDayDistributionFreightFile

public static Map<Integer, Map<*VehicleType*, Map<*TimeOfDay*, Double>>> **readTimeOfDayDistributionFreightFile**(Str *file Na*

Reads time of day distribution file for freight vehicles.

Parameters

- **fileName** – File name.

Returns Time of day distribution.

readTripRatesFile

public static HashMap<Integer, Double> **readTripRatesFile**(String *fileName*)

Reads trip rates file.

Parameters

- **fileName** – File name.

Returns Map with yearly trip rates.

readUnitCO2EmissionFile

public static HashMap<Integer, Map<*EnergyType*, Double>> **readUnitCO2EmissionFile**(String *fileName*)

Reads unit CO2 emissions file.

Parameters

- **fileName** – File name.

Returns Map with unit CO2 emissions.

readVehicleTypeToPCUFile

public static Map<*VehicleType*, Double> **readVehicleTypeToPCUFile** (String *fileName*)
Reads vehicle type to PCU conversion file.

Parameters

- **fileName** – File name.

Returns Map with PCU equivalents.

readZonalCarCostsFile

public static HashMap<Integer, HashMap<String, Double>> **readZonalCarCostsFile** (String *fileName*)
Reads zonal car journey costs file.

Parameters

- **fileName** – File name.

Returns Map with cost data.

readZonalCarEnergyConsumptionsFile

public static HashMap<Integer, HashMap<*EnergyType*, HashMap<String, Double>>> **readZonalCarEnergyConsumptionsFi**

Reads zonal car energy consumptions file.

Parameters

- **fileName** – File name.

Returns Map with zonal energy consumptions data.

readZonalVehicleCO2EmissionsFile

public static HashMap<Integer, HashMap<*VehicleType*, HashMap<String, Double>>> **readZonalVehicleCO2EmissionsFile**

Reads zonal vehicle CO2 emissions file.

Parameters

- **fileName** – File name.

Returns Map with CO2 emissions data.

PropertiesReader

public class **PropertiesReader**
Properties file reader.

Author Milan Lovric

Constructors

PropertiesReader

public **PropertiesReader**()

Methods

getProperties

public static Properties **getProperties**(String *configFile*)
Reads properties from the configuration file.

Parameters

- **configFile** – Path to the configuration file.

Returns Loaded properties.

RandomSingleton

public class **RandomSingleton**
Creates only one instance of the random number generator that can be used throughout the whole model. Simulation results can then be reproduced by using the same seed.

Author Milan Lovric

Methods

getInstance

public static *RandomSingleton* **getInstance**()
Getter for the singleton instance of the random number generator.

Returns Random number generator.

nextDouble

public double **nextDouble**()
Generates a pseudorandom real number between 0 and 1.

Returns Pseudorandom real double.

nextInt

public int **nextInt**(int *bound*)
Generates a pseudorandom whole number smallet than the bound.

Parameters

- **bound** – Upper bound.

Returns Pseudorandom whole number.

setSeed

public void **setSeed**(long *seed*)
Setter method for the seed of the random number generator.

Parameters

- **seed** – Seed of the random number generator.

1.2.12 nismod.transport.visualisation

BarVisualiser

public class **BarVisualiser** extends JFrame
For visualising bar charts using JFreeChart.

Author Milan Lovric

Constructors

BarVisualiser

public **BarVisualiser**(DefaultCategoryDataset *dataset*, String *title*, String *paletteName*, boolean *invertColours*)

Methods

main

public static void **main**(String[] *args*)

saveToPNG

public void **saveToPNG**(String *fileName*)

LineVisualiser

public class **LineVisualiser** extends JFrame
For visualising pie charts using JFreeChart.

Author Milan Lovric

Constructors

LineVisualiser

public **LineVisualiser** (DefaultCategoryDataset *dataset*, String *title*)

Methods

main

public static void **main** (String[] *args*)

saveToPNG

public void **saveToPNG** (String *fileName*)

NetworkVisualiser

public class **NetworkVisualiser**
For visualising the road network.

Author Milan Lovric

Fields

ROAD_LINK_WIDTH

public static final float **ROAD_LINK_WIDTH**

Constructors

NetworkVisualiser

protected **NetworkVisualiser** ( )

Methods

visualise

public static void **visualise** (*RoadNetwork roadNetwork*, String *mapTitle*)
Visualises the road network as loaded from the shapefiles.

Parameters

- **roadNetwork** – Road network.
- **mapTitle** – Map title for the window.

Throws

- **IOException** – if any.

visualise

public static void **visualise** (*RoadNetwork roadNetwork*, String *mapTitle*, Map<Integer, Double> *linkData*, String *linkDataLabel*, String *shapefilePath*)
Visualises the road network with dailyVolume.

Parameters

- **roadNetwork** – Road network.
- **mapTitle** – Map title for the window.
- **linkData** – Data used to classify and colour road links.
- **linkDataLabel** – Label describing the link data used.
- **shapefilePath** – The path to the shapefile into which data will be stored.

Throws

- **IOException** – if any.

visualise

public static void **visualise** (*RoadNetwork roadNetwork*, String *mapTitle*, Map<Integer, Double> *linkData*, String *linkDataLabel*, String *shapefilePath*, URL *congestionChargeZoneUrl*)
Visualises the road network with dailyVolume.

Parameters

- **roadNetwork** – Road network.
- **mapTitle** – Map title for the window.
- **linkData** – Data used to classify and colour road links.
- **linkDataLabel** – Label describing the link data used.
- **shapefilePath** – The path to the shapefile into which data will be stored.
- **congestionChargeZoneUrl** – The path to the shapefile with the congestion charge zone boundary.

Throws

- **IOException** – if any.

PieChartVisualiser

public class **`PieChartVisualiser`** extends JFrame
For visualising pie charts using JFreeChart.

Author Milan Lovric

Constructors

PieChartVisualiser

public **`PieChartVisualiser`**(DefaultPieDataset *dataset*, String *title*, String *paletteName*, boolean *threeD*)

Methods

main

public static void **`main`**(String[] *args*)

saveToPNG

public void **`saveToPNG`**(String *fileName*)

1.2.13 nismod.transport.zone

Zoning

public class **`Zoning`**
For mapping Tempro zones to the nodes of the road network.

Author Milan Lovric

Fields

MAX_NEAREST_NODES

public static int **`MAX_NEAREST_NODES`**

TOP_LAD_NODES

public static int **`TOP_LAD_NODES`**

Constructors

Zoning

public **Zoning** (URL *zonesUrl*, URL *nodesUrl*, *RoadNetwork* *rn*, Properties *params*)
Constructor for the zoning system.

Parameters

- **zonesUrl** – Url for the zones shapefile.
- **nodesUrl** – Url for the nodes shapefile.
- **rn** – Road network.
- **paramas** – Properties file with parameters.

Throws

- **IOException** – if any.

Methods

getAccessEgressFactor

public double **getAccessEgressFactor** ()
Getter for access/egress scaling factor.

Returns Access/egress scaling factor.

getLADToListOfContainedZones

public HashMap<String, List<String>> **getLADToListOfContainedZones** ()
Getter for LAD to list of contained Tempro zones mapping.

Returns LAD to list of contained zones.

getLADToName

public HashMap<String, String> **getLADToName** ()
Getter for LAD code to LAD name mapping.

Returns LAD code to LAD name mapping.

getLadCodeToIDMap

public HashMap<String, Integer> **getLadCodeToIDMap** ()
Getter for LAD zone ONS code to ID.

Returns LAD zone ONS code to LAD zone ID map.

getLadIDToCodeMap

public String[] **getLadIDToCodeMap** ()
Getter for LAD zone ID to ONS code.

Returns LAD zone ID to LAD zone ONS code.

getNodeToZoneMap

public HashMap<Integer, String> **getNodeToZoneMap** ()
Getter for node to zone mapping (for each node gives the zone in which it is located).

Returns Node to zone map.

getTemproCodeToIDMap

public HashMap<String, Integer> **getTemproCodeToIDMap** ()
Getter for Tempro zone ONS code to ID.

Returns Tempro zone code to Tempro zone ID map.

getTemproIDToCodeMap

public String[] **getTemproIDToCodeMap** ()
Getter for LAD zone ID to ONS code.

Returns Tempro zone ID to Tempro zone ONS code.

getZoneIDToLadID

public int[] **getZoneIDToLadID** ()
Getter for Tempro zone ID to LAD zone ID mapping.

Returns Tempro zone ID to LAD zone ID array.

getZoneIDToNearestNodeDistanceMap

public double[] **getZoneIDToNearestNodeDistanceMap** ()
Getter for zone ID to nearest node distance mapping (in meters).

Returns Zone to distance map.

getZoneIDToNearestNodeIDFromLADTopNodesMap

public int[] **getZoneIDToNearestNodeIDFromLADTopNodesMap** ()
Getter for zone ID to nearest node ID among top LAD nodes mapping.

Returns Zone to node map.

getZoneIDToNearestNodeIDMap

public int[] **getZoneIDToNearestNodeIDMap**()
Getter for Tempro zone ID to nearest node ID mapping.

Returns Zone to node map.

getZoneToCentroid

public HashMap<String, Point> **getZoneToCentroid**()
Getter for Tempro zone to its centroid mapping.

Returns Tempro zone to centroid mapping.

getZoneToLADMap

public HashMap<String, String> **getZoneToLADMap**()
Getter for Tempro zone to LAD zone mapping.

Returns Tempro zone to LAD zone map.

getZoneToListOfContainedNodes

public HashMap<String, List<Integer>> **getZoneToListOfContainedNodes**()
Getter for Tempro zone to list of contained nodes mapping.

Returns Zone to list of contained nodes.

getZoneToMinMaxDimension

public double[][] **getZoneToMinMaxDimension**()
Getter for Tempro zone ID to min [0] and max [1] dimension of the zone bounding box (envelope) [in km].

Returns Zone min and max dimension (width or height).

getZoneToNearestNodeDistanceMap

public HashMap<String, Double> **getZoneToNearestNodeDistanceMap**()
Getter for zone centroid to nearest node distance mapping (in meters).

Returns Zone to distance map.

getZoneToNearestNodeIDFromLADTopNodesMap

public HashMap<String, Integer> **getZoneToNearestNodeIDFromLADTopNodesMap**()
Getter for zone centroid to nearest node ID among top LAD nodes mapping.

Returns Zone to node map.

getZoneToNearestNodeIDMap

public HashMap<String, Integer> **getZoneToNearestNodeIDMap**()
Getter for Tempro zone centroid to nearest node ID mapping.

Returns Zone to node map.

getZoneToNodeDistanceMatrix

public double[][] **getZoneToNodeDistanceMatrix**()
Getter for Tempro zone to all nodes distance matrix [in metres].

Returns Zone to node distance matrix.

getZoneToSortedListOfNodeAndDistancePairs

public HashMap<String, List<Pair<Integer, Double>>> **getZoneToSortedListOfNodeAndDistancePairs**()
Getter for Tempro zone to sorted node distances mapping (distances to ALL nodes in the network).

Returns Zone to sorted list of nodes and distances.

getZoneToZoneDistanceMatrix

public double[][] **getZoneToZoneDistanceMatrix**()
Getter for Tempro zone (centroid) to Tempro zone (centroid) distance matrix [in metres].

Returns Zone to node distance matrix.

1.3 Authors

1.3.1 Development

- Milan Lovric <M.Lovric@soton.ac.uk>

1.3.2 Integration

- Will Usher <william.usher@ouce.ox.ac.uk>
- Tom Russell <tom.russell@ouce.ox.ac.uk>
- Roald Schoenmakers <roald.schoenmakers@ouce.ox.ac.uk>
- Thibault Lestang <thibault.lestang@cs.ox.ac.uk>

1.3.3 Management

- Simon Blainey <S.P.Blainey@soton.ac.uk>
- John Preston <J.M.Preston@soton.ac.uk>

1.4 License

```
MIT License

Copyright (c) 2018 National Infrastructure Systems Model

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Index

A

B

C

D

E

F

G

H

I

L

M

N

O

P

R

S

T

U

V

W

Z