SuperMap Objects Java 6R Technical Document ——Traffic Transfer Analyst

SuperMap Software Co., Ltd.

Beijing · China

Legal Statement

SuperMap Software Co.. Ltd.:

Add: 7/F Tower B, Technology Fortune Center, No. 8 Xueqing Road,

Haidian District, Beijing, 100192, P. R. China

Tel: +86-10-82736655-4170

Fax: +86-10-82734630

HomePage: www.supermap.com

Sales: request@supermap.com

Tech Support: globalsupport@supermap.com

SuperMap Software welcomes all advices and suggestions from you.

Content

1

Summary

In real world, the lines used in public transport transfer analysis are the lines of the bus and subway route lines. After getting these line data, we usually abstract these data to lines and points, stations for points and bus routes for lines. The public data modeling will create logic relationship between these abstract points and line to simulate public transport in real word. This section will mainly introduce how to abstract data and create public transport model. Moreover, we are also going to show how to carry out public transport analysis by using public transport model.

2

Public Transport Data Modeling

In daily life, the public transport transfer involves the route information of bus, subway, and coach, which will be considered as points and lines in a public transport transfer model. Points represent the public transport stops, and lines display the routes. In order to imitate the relationship between public transport stops and routes in reality, the correspondent logical relationship will be established between the points and lines in the public transport transfer model.

The following steps will lead you to establish a public transport transfer model.

2.1 Public Transport Data Abstraction

To be simple, public transport data abstraction means transforming the transport stations, routes into the point and line that can be processed by computers. As a result, the public transport network is abstracted as a series of point dataset and line dataset.

Public Transport Point Dataset

The station name field is the dispensable field for public transport modeling, which represents the names of the stations.

Public Transport Line Dataset

The raw data of the public transport routes will be stored as line datasets, whose route number or name field is indispensable for public transport modeling. In order to make sure the system can identify as an integrity route, each integrity route must be stored totally in one line dataset. For instance, Bus No.123 should be stored in one dataset, rather than two or more. Users are allowed to definite other fields that used to describe other properties of the public transport routes, for instance, the weight information, the route type, and the ticket price.

There are three types of public transport routes, uniline route, two-way route, and loop route. In order to produce proper result, these three routes will be abstracted in different ways as follows:

(1) The two-way route, which has the same go-way and return-way is not necessary to consider the direction. It can be abstract by a line. For instance, the Bus No.1 in the Diagram 1. The left

part is the situation in reality which consists of double lines, while the right part is the abstracted two-way line in the model which has only one line without direction consideration.

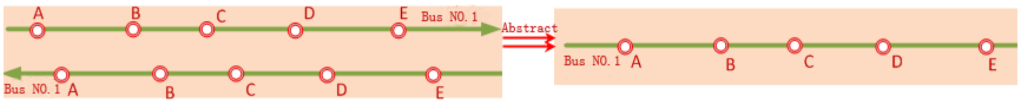


Diagram 1 A two-way route in reality and model

(2) When the go-way is not the same with the return-way, it is necessary to display this route by two lines, and to consider the route directions. In the Diagram 2, the left part displays the up route and down route of Bus No.3 in real life, and the right part illustrates the abstract result. This kind of routes has been abstracted as two lines.

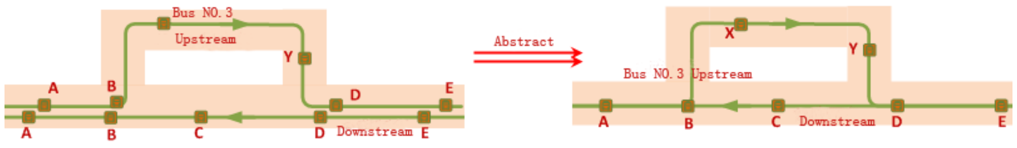


Diagram 2 A uniline route in reality and model

(3) A loop route has only one terminal, pleas look at Diagram 3, the Bus No.5 goes alone with the double loop routes in the left part, while it has only one loop route without direction in the model.

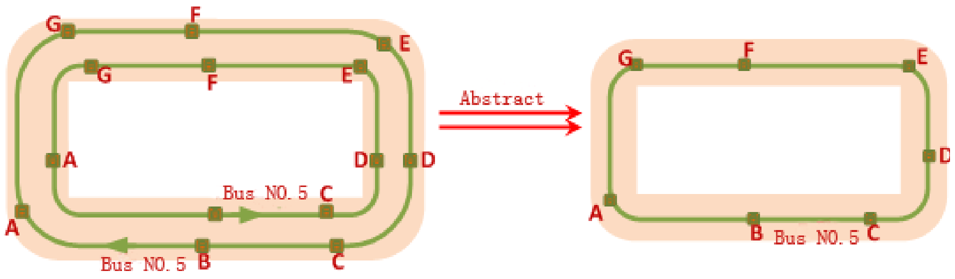


Diagram 3 The loop routes in reality and model

2.2 Building Public Transport Data Model

The meothod buildTrafficTransferModel() of the NetworkBuilder class can be used to build public transport data model. This function gives the reality logical relationship to the point data and line data that abstracted from the real public transport network so as to simulate the relations of public transport stations and routes in the real world.

Syntax:

public static DatasetVector buildTrafficTransferModel(TrafficTransferAnalystSetting trafficTransferAnalystSetting, Datasource outputDatasource,String outputModelDatasetName)

Parameters:

trafficTransferAnalystSetting: the designated environment configuration object.

outputDatasource: datasource stores the property dataset of the output public transport transfer model.

outputModelDatasetName: the names of the property dataset of the output public transport transfer model.

Return Value:

Returns the datasets of the public transport transfer. The public transport model dataset is a property table dataset, containing the logical relationship between the stops and route lines. The process of building a public transport model is the process of matching the stops and route lines automatically, and assigning values to the property table.

3

Environment Configuration of the Public Transport Analysis

Environmental configuration of the public transport transfer analysis is mainly used to build public transport transfer model. In order to do the configuration, we just need to configure the methods of a class, named TrafficTransferAnalystSetting. For its methods information, please refer to the public transport transfer analysis in Table 1.

Table 1 TrafficTransferAnalystSetting Class Methods

Methods	**Description**

getLineSettings()
or
setLineSettings(TransferLineSettings lineSettings)
	Setting or obtaining the environmental configuration set of public transport transfer lines, please refer to class TransferLineSettings for detail.

getStopSettings()
or
setStopSettings(TransferStopSettings stopSettings)
	Setting or obtaining the environmental configuration set of public transport transfer stops, please refer to class TransferStopSettings for detail.

getSnapTolerance()
or
setSnapTolerance(double value)
	Setting or obtaining the snap tolerance of the stops, which is used to decide if the stops are passed by the lines.

getMergeTolerance()
or
setMergeTolerance(double value)
	Setting or obtaining merge tolerance, which takes the points between whose distance is less than the tolerance as one point to process.

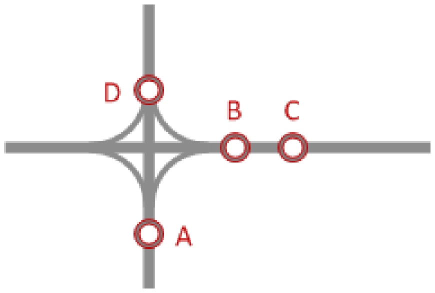
getWalkingTolerance() or setWalkingTolerance (double value)	Setting or obtaining walking threshold value, which is the longest distance a traveler can stand between transfers. When the necessary walking distance is longer than it, the transfer way will be

	cancelled.

The merge tolerance takes the points between whose distance less than the tolerance as one point to process.

The merged points, which are abstracted to be one point in the system, is given a MergerID, storing in the SmUserID. The content of SmUserID is the same with MergerStopID that is stored in the public transport transfer model dataset. Please refer to the public transport transfer analysis part for the detail description. The relationship between the merged points and the true points are one to one and one to many.

Please look at figure 1, the distance between A, B is 150m, A, D is 120m, B, C is 50m, B, D is 110m. If stopMergeToerance is 100m, then the points should be merged are B and C, assigning value to MergerID. The sequence of merging stops is at random. The property table of the merged stops in Diagram 4 is list in Table 2.



SmID	SmUserID	Name
1	1	A
2	2	B
3	2	C
4	3	D

Diagram 4 the merged stops

Table 2 the property of the merged stops in Diagram 4

The public transport transfer should be taken place at the public transport transfer stops. There are two situations to deal with, first, the stops of two public bus lines are at the same point, and in this case, the traveler does not have to walk to transfer. In another case, however, the traveler has to have a walk to transfer, and the walking threshold value is the longest distance a traveler can stand.

Stop snap tolerance is used to decide if the stop is passed by the bus line. If the distance between stop A and line 1 is smaller than the snap tolerance, then the stop A will be taken as a stop along line 1.

The following is the detail introduction of classes that referred by TrafficTransferAnalystSetting.

TransferLineSettings class object is the set of TransferLineSetting class object. Please refer to Table 3 for the method information of TransferLineSetting:

Table 3 TransferLineSetting Class Methods

Methods	**Description**

getDataset()
or
setDataset(DatasetVector datasetVector)
	Setting or obtaining the dataset that contains the public lines.

getNameField()
or
getNameField(String value)
	Setting or obtaining the name field of public lines, which is the compulsory field of the public line dataset.

getAliasField()
or
setAliasField(String value)
	Setting or obtaining the alias field of the public bus line.

getFirstTimeField() or setFirstTimeField (String value)	Setting or obtaining the first bus departing time field of the public bus line.

getLastTimeField()
or
setLastTimeField(String value)
	Setting or obtaining the last bus departing time field of the public bus line.

getIntervalField()
or
setIntervalField(String value)
	Setting or obtaining the bus interval time field for the lines.

getSpeedField() or setSpeedField (String value)	Setting or obtaining speed field.
getWeightFieldInfos() or setWeightFieldInfos (TransferWeightFieldInfos transferWeightFieldInfos)	Setting or obtaining the set of bus line weight field information.

getFareFieldInfo()
or
setFareFieldInfo(FareFieldInfo fareFieldInfo)
	Setting or obtaining the ticket price field information of the bus line.

getLineTypeField()
or
setLineTypeField(String value)
	Setting or obtaining the field of bus line type: value 0 means one-way route; value 1 represents two-way route; value 2 represents loop route.

In the establishment of public bus model, TransferLineSettings class can make use of public bus line set to build public transport transfer model.

Table 4 FareFieldInfo Class Methods

Methods	**Description**

getFareTypeField()
or
setFareTypeField(String value)
	Setting or obtaining the ticket type field of public bus, i.e. the charge mode.
	0: flat fare;
	1: metered fare;
	2: sectional fare;

getStartFareRangeField() or setStartFareRangeField (String value)	Setting or obtaining the scope field of the basic fare. The basic fare scope represents a certain distance that should be charged the basic fare. The unit of the property value is decided by the public bus fare type. In the case of metered fare, the unit is distance. For instance, the flag-fall price is 3 kilometer, the property value is 3, the unit is distance (kilometer); In the case of sectional fare, the unit of the property is stop number. For instance, the traveler that is taking within 5 stops will be charged the basic fare, then the return value is 5, and the unit of it is the number of stops.
getStartFareField() or setStartFareField (String value)	Setting or obtaining the start fare fields.
getFareStepField() or setFareStepField (String value)	Setting or obtaining the fare increasing step field, which is the increased fare generated from each additional distance unit (charge by distance).

TransferWeightFieldInfos class object is the set of TransferWeightFieldInfo class object; TransferWeightFieldInfo has four methods, TransferWeightFieldInfo.Name (the name of public transport transfer weight field information) and TransferWeightFieldInfo.WeightField (weight field name). In order to explain the exact meaning of them, we have to get to know the setWeightName(String value) method of TrafficTransferAnalystParameter class. TrafficTransferAnalystParameter class is the parameter item that shall be used in public transport transfer analysis, and the method setWeightName(String value) is the specific weight information, i.e. TrafficTransferAnalystParameter. setWeightName(String value), which is TransferWeightFieldInfo.setName()at here.

The relationship between TransferWeightFieldInfo.setName(String value) and TransferWeightFieldInfo.setWeightFieldName(String value): it is usually more than one line vector datasets are used in the establishment of the public transport transfer model. However, in analysis practice, some lines require distance weight, while others require time weight or other weight information, and so on. Furthermore, the same weight information's field name could be different in different datasets. The public transport transfer analysis parameter of TransferWeightFieldInfo.setWeightFieldName has only one name that represents the weight information. Hence, in order to solve the complicate problems caused by multiple line datasets, we apply TransferWeightFieldInfo.setName to represent different meaning or the same meaning but different TransferWeightFieldInfo.setWeightFieldName (the real field in the line dataset method table to express weight. Additionally, this approach can conduct the public transport transfer analysis with different meaning weights.

TransferStopSettings class object is the set of TransferStopSetting, the method information of TransferStopSetting class is illustrated Table 5:

Table 5 TransferStopSetting Class Methods

Methods	**Description**

getDataset()
or
getDataset(DatasetVector datasetVector)
	Setting or obtaining the dataset that contains the public transport transfer stops.

getNameField() or setNameField (String value)	Setting or obtaining the public transport transfer stops name field, which is the compulsory field of the dataset.
getAliasField() or setAliasField (String value)	Setting or obtaining the alias of the bus stops.

In the establishment of public transport transfer model, TransferStopSettings class can make use of multiple point datasets that contain the stops to build public transport transfer model.

4

Public Transport Analysis Procedures and Methods

This section introduces the procedures and methods that are used to carry out a public transport transfer analysis. The method of the TrafficTransferAnalyst class helps to achieve public transport transfer analysis. The following subjects will talk about how to conduct the analysis in detail.

4.1 Loading Public Transport Model

After finishing the configuration of the parameters of the public transport analysis, it is time to load the public transport model. The method of load () of the TrafficTransferAnalyst class can be used to load the models in the SuperMap or other established models.

Syntax:

public boolean load (DatasetVector modelDataset);

Parameters:

modelDataset: public transport model datasets, which is a property table dataset of a public transport data model created by NetworkBuilder.

Return Value:

Returns True if the loading activity is achieved, otherwise returns False.

Note:

(1) You can conduct the public transport analysis after the public transport model is loaded.

(2) Once the parameters of the public transport transfer environment are resetted, the public transport model needs to be reloaded.

4.2 Public Transport Analysis Implementation

4.2.1 Public Transport Analysis Implementation

Implementation of the public transport transfer analysis by the method findTransferPath() of the TrafficTransferAnalyst class.

Syntax:

public TrafficTransferAnalystResult findTransferPath(TrafficTransferAnalystParameter trafficTransferAnalystParameter)

Parameters:

trafficTransferAnalystParameter: parameter objects of the correspondent public transport transfer analysis.

This parameter is an object of TrafficTransferAnalystParameter class. This class sets the parameters of the public transport transfer analysis. This class can be used to set the name identify of public transport weight, transfer times limitation, weight ratio of walking and public transport, initial stop ID and terminal ID, datasets of initial and terminal stops, and coordinates of the initial and terminal stops. For more details, please refer to Table 6:

Table 6 The methods of TrafficTransferAnalystParameter Class

Methods	**Description**

getStartStopID()
or
setStartStopID(int value)
	Setting or obtaining the ID of the start stop.

getStartStopDataset() or setStartStopDataset (DatasetVector datasetVector)	Setting or obtaining the dataset that the start stop belongs to.
getEndStopID() or setEndStopID (int value)	Setting or obtaining the ID of the end stop.
getEndStopDataset() or setEndStopDataset(DatasetVector	Setting or obtaining the dataset that the end stop belongs to.

datasetVector)	

getStartStopPosition()
or
setStartStopPosition(Point2D point2D)
	Setting or obtaining the coordinate of the start stop.

getEndStopPosition()
or
setStartStopPosition(Point2D point2D)
	Setting or obtaining the coordinate of the end stop.

getWeightName() or setWeightName (String value)	Setting or obtaining the identify name of the weight information. It is the value set by the setName() method of a certain TransferWeightFieldInfo objects in the TransferWeightFieldInfos which is set in the TransferLineSetting class.

getMaxTransferGuideCount()
or
setMaxTransferGuideCount(int value)
	Setting or obtaining the maximum count of the TransferGuide objects. The TransferGuide records the transfer path guide from the start stop to the end stop. It contains the items of the TransferGuide (TransferGuideItem), with each representing a length of transfer or walking line.

getWalkingRatio() or setWalkingRatio (double value)	Setting or obtaining the ratio of walking to taking bus.
This ratio is used to evaluate the traffic transfer schemes. There are restrictions on the count of the schemes (set by the setMaxTransferGuideCount() method), so it is necessary to select the most optimal schemes. For example, there are two schemes. Scheme1: bus 10km, walking 1km； Scheme2: bus 15km, walking 0.5km； Supposes the ratio is 15: The cost in scheme1 is 10 + 1*15 = 25	

The cost in scheme2 is 15 + 0.5*15 = 22.5
In this case the scheme2 is better.
Supposes the ratio is 2:
The cost in scheme1 is 10 + 1*2 = 12
The cost in scheme2 is 15 + 0.5*2 = 17
In this case the scheme1 is better.

The Result of Public Transport Transfer Analysis

The result of public transport transfer analysis is stored in the TrafficTransferAnalystResult class, which has only one method, getTransferGuides(). It is the guide object of the public transport transfer, the object of the TransferGuide class.

The public transport transfer guide records the transfer schemes between the initial stop and the terminal stop designed in analysis, which consists of the public transport transfer items (objects of TransferGuideItem class). Each item represents a segment of transfer or walk line. The number of items in the public transport transfer guide object can be gotten by the TransferGuide class. The serial number will help to get the item object, guide distance, and guide cost of the public transport transfer guide. The item information, such as initial and terminal stops information, public transport lines information, can be found in the TransferGuideItem class. The following table is an introduction of class TransferGuideItem methods.

Table 7 The methods of TransferGuideItem Class

Methods	**Description**
getStartStopInfo()	Obtaining the information about the start stop of the segment represented by this TransferGuideItem object. The information includes the dataset which this stop belongs to, the stop ID, the name of the stop, etc. For more information, please refer to the TransferStopInfo class.
getEndStopInfo()	Obtaining the information about the end stop of the segment represented by this TransferGuideItem object. The information includes the dataset which this stop belongs to, the stop ID, the name of the stop, etc. For more information, please refer to the TransferStopInfo class.
getLineInfo()	Obtaining the information about the segment represented by the TransferGuideItem object. The information includes the dataset which

	this line belongs to, the ID of the line of this segment, the name of the line of this segment, the interval between two buses of this segment, the time of the first and last bus of this segment, the speed of this segment, the distance of this segment, the cost of this segment, the number of the stops this segment has. For more information, please refer to the TransferLineInfo class.
getStartIndex()	Obtaining the index of the start stop of the segment represented by the TransferGuideItem object in the whole traffic line which the segment belongs to.
getEndIndex()	Obtaining the index of the end stop of the segment represented by the TransferGuideItem object in the whole traffic line which the segment belongs to.
getPassStopCount()	Obtaining the number of the stops this segment represented by the TransferGuideItem object has.
getStartPosition()	Obtaining the coordinates of the start stop of the segment represented by the TransferGuideItem object.
getEndPosition()	Obtaining the coordinates of the end stop of the segment represented by the TransferGuideItem object.
getDistance()	Obtaining the distance of the segment represented by the TransferGuideItem object.
getWeight()	Obtaining the cost of the segment represented by the TransferGuideItem object.
isWalking()	Whether to pass the segment represented by the TransferGuideItem object by walk or not. If the value of the IsWalking property is true that means passing the segment by walk. At this, the property of StartStopInfo, EndStopInfo, LineInfo, StartIndex, EndIndex, PassStopCount is meaningless.
getRoute()	Obtaining the Geoline object of the segment represented by the TransferGuideItem object.

Table 8 The methods of TransferStopInfo Class

Methods	**Description**
getDataset()	Obtaining the dataset that the transfer stop belongs to.
getID()	Obtaining the ID of the stop.
getName()	Obtaining the name of the stop.
getAlias()	Obtaining the alias of the stop.

Table 9 The methods of TransferLineInfo Class

Methods	Description
getDataset()	Obtaining the dataset which the line belongs to.
getID()	Obtaining the line ID.
getName()	Obtaining the name of the line.
getAlias()	Obtaining the alias of the line.
getFirstTime()	Obtaining the time of the first bus of the line.
getLastTime()	Obtaining the time of the last bus of the line.
getInterval()	Obtaining the interval between every two buses of the line.
getSpeedField()	Obtaining the speed of the bus of the line.
getFareInfo()	Obtaining the fare information of this line For more information, please refer to the FareInfo class.
getTotalDistance()	Obtaining the total distance of the line.
getTotalWeight()	Obtaining the cost of the line.
getStopCount()	Obtaining the number of the stops the line has.
getLineType()	Obtaining the type of the traffic line, including One-way street, Two-way street, Loop street.

Table 10 The methods of FareInfo Class

Methods	Description
getType()	Obtaining the type of the bus fare. SuperMap provides three types of the bus fare; they are charging by distance, charging by stops and single fare. Please refer to FareType class.
getStartFareRange()	Obtaining the range of the start fare. The start fare will be used within the range of the start fare. The unit of the returned value of this method depends on the type of the bus fare. If the fare type is DISTANCE, the unit would be the distance unit, for example, taxi will charge the same within 3 km. In this case, this method will return 3, and the unit will be km. If the fare type is STOPS, the unit would be the number of the stops, for example, buses will charge the same within 5 stops, in this case, the method

	will return 5, and the unit will be the number of the stops.
getStartFare()	Obtaining the start fare.
getFareStep()	Obtaining the step of the fare. The step of the fare is the cost increased for each added stop (the fare type is STOPS) or distance unit (the fare type is DISTANCE).

4.2.2 Implementation of the Public Transport Query

Query public transport lines by stops.

Syntax:

public TransferLineInfo[] findLines(DatasetVector stopDataset,int stopID)

Parameters:

stopDataset: the dataset that public transport belongs to.

stopID: the ID of the public transport stops.

Return Value:

Return array of the public transport lines information, i.e. the TransferLineInfo class object array.

Query stops by public transport lines.

Syntax:

public TransferStopInfo[] findStops(DatasetVector lineDataset,int lineID)

Parameters:

lineDataset : the dataset that public transport belongs to.

lineID: the ID of public transport lines.

Return Value:

Return array of stops information, i.e. the TransferStopInfo class object array.

4.2.3 Other Methods

There are other methods provided by the TrafficTransferAnalyst class. These methods are used to add some conditions in the public transport analysis. Please refer to the related section in the Programmer Reference for detail.