Overview

As the development of human civilization, public infrastructure (power facilities, telecommunications and cable TV networks, road transportation, GPS monitoring, water network, etc.) are constantly improving. Road facility maintenance, management and planning cannot be completely merely relying on manpower. Computers are required to assist the job. In GIS, these fundamental facilities can be abstracted as network system composed by many interconnected lines. And the network model is the abstraction of the network system of the real world. Take the urban transportation network for example, the road and other linear features are abstracted as line segments, which are also called the network arcs. While the crossroads, the bus stops and the other point-like features are abstracted as points, in the network which are also called the network nodes. In the network model, resource information can transfer from a node to another node along the arc.

Network analysis is a process to solve the practical problem by analysis in network model, such as path analysis, service area analysis, and closest facility. Currently, network analysis has been widely used for pipeline and network designing, query and analysis of different industries such as electronic navigation, transportation, tourism, urban planning and management, logistics, electric power, telecommunication, etc.

Network Model

In GIS, there are primarily two types of network models: transportation network model and facility network model.

Transportation Network Model

Transportation Network model has no directions. This means that even though we can specify directions for network arcs, the flow medium (pedestrian or transport resources) can decide the direction, speed and destination. For example, when a driver is driving in the street, he can choose to turn right or left, time to stop, the traveling direction, etc. Of course, there will be certain restrictions, for example, one-way road, not allowed to turn around and so on, which is completely different from the facility network model.

Facility Network Model

Facility network model has directions. This means that flow medium (water flow, electric flow) will flow according to the network rules. For example, the flow of water is predefined. The direction of the water flow can be changed. But such change is not determined by the water flow itself, but the control the network flow direction through turning on or off the valve by engineers, that is, change of the network flow rules.

Network data model (Network)

Network datasets (Network) are well suited to store the data models which have the network topological relationships. Network dataset models include the network line dataset and the network node dataset, as well as the topological relationship between them. In the network dataset, the line dataset is the master while the point dataset is the sub dataset.

Network Anlysis Concepts

Network model not only has the abstracted topological relationship between arcs and nodes, also has geometric location feature and geographic attribute feature of GIS spatial data (topological relationship is the cross-correlation between geographic objects in spatial location, such as the connection relationship of nodes and lines, and lines and polygons). Concetps for network analysis are introduced below.

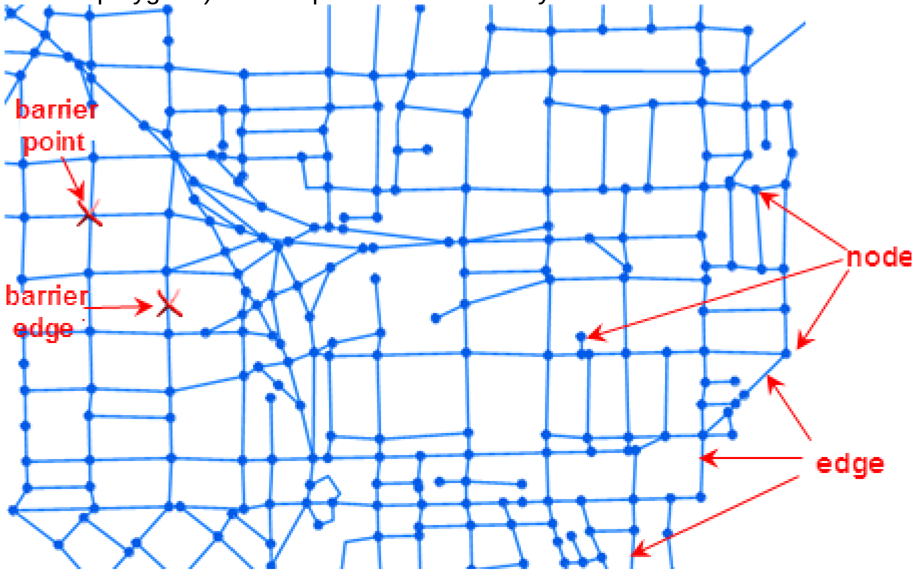


Figure3-1 Network Diagram

Node

Nodes are the places that arcs connects network, as shown in 3-1. Nodes can represent the real road intersections, rivers intersections, etc. The nodes and arcs respectively correspond to an attribute table, their adjacent relationships associate by the fields of attribute tables.

Egde

The arc segment is an edge in the network connecting with the other arcs by nodes, as shown in 3-1. The arcs can represent the real-world highways and railways of the transportation network, the transmission line in the power network, the rivers of hydrological network, etc. The interconnected relationships between arcs have topological structure.

Network

The network is a model which is composed of a group of interrelated arcs, nodes and their attributes, as shown in 3-1. The network can express the real world's roads, pipelines, etc. The monitoring of supply, demand, supply center, etc. still needs the definition of features as shown below:

Impendency

In our life, from the starting point, after a series of roads and junctions, we arrive the destination, which is bound to produce a cost measuring by distance, time and currency. In the network model, the cost of nodes and arcs is abstracted as the network resistance, and the information is stored in the attribute fields called the resistance field.

Center

The center points are the discrete devices located at the nodes of the network and have the capabilities to accept or provide resources. Facilities are the substances, resources, information, management, cultural environment required by GIS, etc. For example, the school has educational resources and the students have to go to school to learn; the retail warehousing point stored the goods required by the retail outlets, and needs to deliver the goods to the various outlets.

Barrier Nodes and Barrier Edges

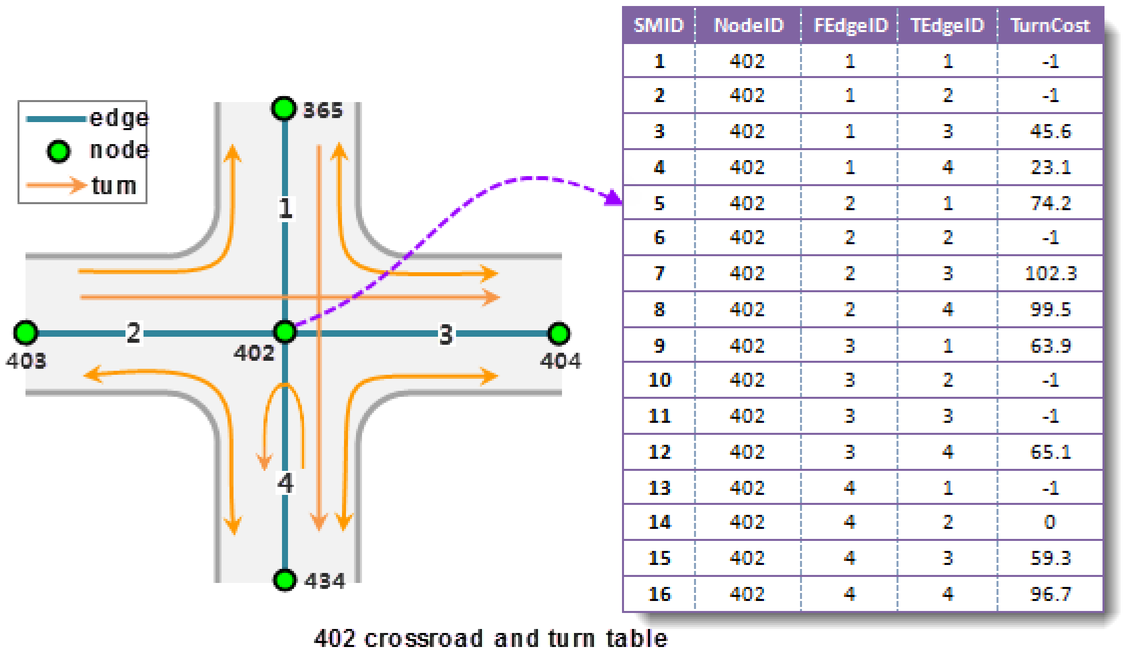
The problem of traffic jams in the city can be seen everywhere, which is a random and dynamic process, and it is no rules to follow. In order to reflect the real-time situation of transportation network, the traffic-jam arcs need to have the feature of temporarily inhibiting the passages of vehicles. When the traffic is back to normal, the attribute of the arc also can be set to normal in a real time. The concepts of obstacle edges and obstacle points can solve the above problems very well. The benefit to introduce the concepts of obstacle edges and obstacle points is that the obstacle settings are relatively independent, i.e., whether to set the obstacles has nothing to do with the current network environmental parameters.

Turn Table

For the transportation analysis, you can use a turn table to store the cost of turning. Turning is the process that the medium flows from an edge, through the middle nodes, to the other adjacent edge. Turning cost is the cost to complete a turning.

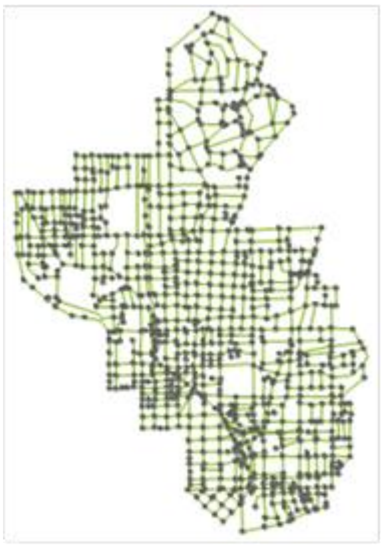
A turn table is a tabular dataset. It generally has four fields including FromEdgeID, ToEdgeID, NodeID and TurnCost. These fields correlate with the fields in edges and nodes. Each record in the table dictates an edge cost passing the road node. Turn cost is usually directional, the negative cost values of the turning generally means to prohibit the turning.

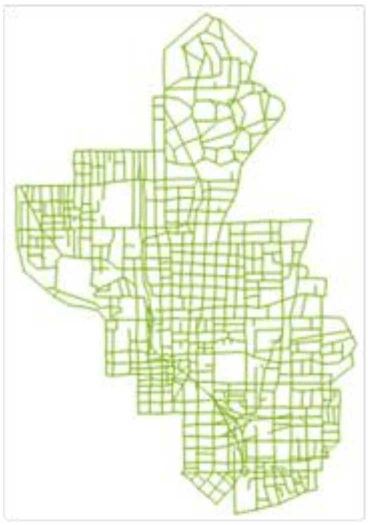
For example, for the network analysis of the roads, we often encounter the crossroads, divergence, etc., as shown in 3-2. The figure on the right is a crossroad diagram, and the table on the left is a turn table corresponding to the crossroad. In the turn table, there are the cost records that the vehicles turn at the crossroads.



SMID	NodeID	FEdgeID	TEdgeID	TurnCost
1	402	1	1	-1
2	402	1	2	-1
3	402	1	3	45.6
4	402	1	4	23.1
5	402	2	1	74.2
6	402	2	2	-1
7	402	2	3	102.3
8	402	2	4	99.5
9	402	3	1	63.9
10	402	3	2	-1
11	402	3	3	-1
12	402	3	4	65.1
13	402	4	1	-1
14	402	4	2	0
15	402	4	3	59.3
16	402	4	4	96.7

Figure 3-2 Turn table structure


Network dataset


Line dataset

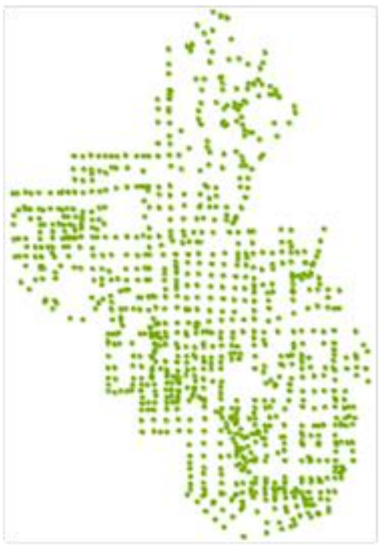

Point dataset

Figure 2-1 Line data model and physical storage

The network dataset can be used for network analysis such as path analysis, service area analysis, closest facility analysis, location-allocation analysis, adjacent node and connected node analysis.

Network Analysis Functions

SuperMap GIS provides strong network analysis functionalities to handle problems in real life. Problems that are generally can be solved includes:

Place B is lack of what. If we are going to send water from Place A to place B, then we need to check whether the water pipelines are connected between A and B. If connected, then we can turn on the valve and send water. Here connectivity analysis is applicable.

What can we do if a place is polluted and we want to trace the pollution source? We can use the tracing functionalities provided by SuperMap GIS to trace backwards from the polluted place to find out the pollution source.

A logistics company need to deliver a large number of packages to clients every day. The deliver wants to know which route he can choose to send all packages with the shortest distance. In this case, he can use traveling salesmen analysis (one type of path analysis). If we want to find the path with shortest time for a fire truck from the fire station to the site of the incident, we can use optimal path analysis.

City A needs a new post office, then planners can use service area analysis provided by SuperMap GIS to find out the service areas of neighboring post offices, therefore assisting in getting the optimal place for placing the new post office.

Police station receives a call for accident. How they can deliver a police car to the place of accident to handle this? Closest facility analysis provided by SuperMap GIS is applicable in this case.

Suppose that in a city, there are M chains for a fast food restaurant , clients from N places called for food. Then how can the restaurant schedule the delivering order to optimize the delivering resources? Logistics anslysis is applicable here to get the delivery path for each delivery staff.

SuperMap provides network analysis functionalities such as optimal path analysis, service area analysis, traveling salesmen analysis, closest facility analysis, location-allocation analysis, multiple traveling salesmen analysis (logistics analysis), tracing upstream, tracing downstream, connectivity analysis, etc.

Building Network Dataset

Refer to introduction to network analysis in help document for SuperMap iDesktop.

5.1 Methods for Building Network Dataset

Method 1: Build network dataset based on a single line dataset. It can saves non-system string.

Refer to introduction to network analysis in help document for SuperMap iDesktop.

Method 2: Build network dataset based on multiple line datasets and point datasets. It can save non-system string and the split mode.

Refer to introduction to network analysis in help document for SuperMap iDesktop.

Method 3: Build network dataset based on a single line dataset, a single point dataset and known field depicting topological relationship.

Refer to introduction to network analysis in help document for SuperMap iDesktop.

5.2 Network Dataset and Topological Relation Field

SuperMap network dataset is of vector type. It is composed of network edges, network nodes and their topological relationships. A network dataset is composed of an edge (of line type) dataset and a node (of point type) dataset. The edge dataset is the master dataset and node dataset is the sub dataset. In the attribute tables of edge dataset and node dataset, fields depicting topological relationships between edges and nodes are stored. Further explanation is provided below.

While using method 1 and method 2 to build network dataset, topology processing, spiting compound line objects, splitting at intersections between lines (determined by split mode specified), etc. will be performed besides building network dataset to make sure the network dataset built is correct. Topological relationship fields are system fields while using these two methods to build network datasets.

Edge ID field: SMEDGEID in edge attribute table, records the edge ID that is used to uniquely identify the edge.

Node ID field: SMNODEID in node attribute table, records the node ID that is used to uniquely identify the node.

From node ID field: SMFNODE in edge attribute table, records the from node ID corresponding to the value node ID field.

To node ID field: SMFNODE in edge attribute table, records the to node ID corresponding to the value node ID field.

As to method 3, since the network dataset is built completely according to the topological relationships specified by fields, therefore, the edge ID, edge from node ID, edge to node ID, node ID are specified fields, not system fields SMEDGEID, SMFNODE, SMTNODE, SMNODEID, etc. any more.

Transportation Network Analysis

6.1 Transportation Network Analysis Contents Contents

SuperMap transportation network analysis provides following functions:

- Optimal path analysis: Finds least-cost path passing through nodes in a certain order.
- Closest facility analysis: Finds one or more least-cost facility points for each event point given a group of event points and facility points.
- Traveling salesmen analysis: Finds least-cost path passing through a specified number of nodes.

- Multiple traveling salesmen analysis: Finds the cost-effective delivery paths, and gives the corresponding routes given M supply centers and N destinations (M, N are integers greater than zero).
- Service area analysis: Finds the scope of the service (service area) that the specified service stations (supply center) can serve according to specified impedance (service radius).
- Location-allocation analysis: Determines the optimal locations for one or more service providers in a certain region, so that the facilities can be one of the most cost-effective ways to provide services and merchandises.

Details will be introduced in 6.7-6.12. Besides, SuperMap transportation network analysis provides assisting functionalities such as cost matrix calculation, edge weight update, turn weight update, etc., all of which will be introduced in detail in 6.13.

You are suggested understanding transportation network analysis workflow before getting to know usage of transportation network analysis functions in-depth.

6.2 Transportation Network Analysis Workflow Contents

Transportation network analysis workflow is as follows:

1. Build network dataset.
2. Set transportation network analysis environment, including network dataset and topological relationship fields, weight fields, turn table and turn weight information, barrier information, traffic rules etc.
3. Check errors for network dataset and turn table and modify according to the data check results. Repeat until there are no errors any more.
4. Load transportation network model.
5. Perform transportation network analysis. Before analysis, parameters including points for analysis, barrier information, weight information, analysis results to return, etc. need to be set first. Note that parameters for location-allocation analysis is different from those of other network analyses.

Please refer to the help document of SuperMap iDesktop 7C fore details about how to build network dataset. Other steps are introduced in detail below.

6.3 Transportation Network Analysis Environment Setting Contents

Setting transportation network analysis environment is implemented through setting the setAnalystSetting method of the TransportationAnalyst class. We need to set a TransportationAnalystSetting object for this method. This object provides parameters required for transportation network analysis. These parameters include network dataset and topological relationship fields (node ID fields, edge ID fields, from node ID fields, to node ID fields) weight field information, turn table and turn weight information, barrier information, traffic rules, etc. The setting of these parameters will directly affect the transportation network analysis results. Therefore, transportation network analysis environment setting is essentially setting the methods for the TransportationAnalystSetting class. Table 6.1 lists methods for this class.

Table 6.1 Methods for the TransportationAnalystSetting class

Type	Name	Description
DatasetVector	get/setNetworkDataset	Gets or sets the network dataset for analysis. Required.
String	get/setNodeIDField	Gets or sets the field that identifies the node ID in the network dataset. The ID field identifying the network node must be set, and it supports 16-bit integer and 32-bit integer.
String	get/setEdgeIDField	Gets or sets the field that identifies the edge ID in the network dataset. The ID field identify the network edge must be set, and it supports 16-bit integer and 32-bit integer.
String	get/setFNodeIDField	Gets or sets the field that identifies the start node ID in the network dataset. The ID field identifying the start node must be correctly set, and it supports 16-bit integer and 32-bit integer.
String	get/setTNodeIDField	Gets or sets the field that identifies the end node ID in the network dataset. The ID field identifying the end node must be correctly set, and it supports 16-bit integer and 32-bit integer.
WeightFieldInfos	get/setWeightFieldInfos	Gets or sets the WeightFieldInfos object.
double	get/setTolerance	Gets and sets the distance tolerance from a node to an edge. The unit is the same as network dataset specified by the setNetworkDataset method.
int[]	get/setBarrierNodes	Gets or sets the ID list of the barrier nodes, and it is optional.
int[]	get/setBarrierEdges	Gets or sets the ID list of the barrier nodes, and it is optional.
DatasetVector	get/setTurnDataset	Gets or sets the turn dataset, and it is optional.
String	get/setTurnNodeIDField	Gets or sets the field of the turn node ID. It supports 16-bit integer and 32-bit integer, and it is optional.
String	get/setTurnFEdgeIDField	Gets or sets the field of the start turn edge ID. It supports 16-bit integer and 32-bit integer, and it is optional.
String	get/setTurnTEdgeIDField	Gets or sets the field of the end turn edge ID. It supports 16-bit integer and 32-bit integer, and it is optional.
String[]	get/setTurnWeightFields	Gets or sets the turn weight fields collection, and it is optional.
String	get/setNodeNameField	Gets or sets the name of the field which stores the node names, optional.
String	get/setEdgeNameField	Gets or sets the field of the edge name. It is optional. Note, if not set it, even you set the setPathGuidesReturn method as true, the analysis result will not return the path guide.
String	get/setRuleField	Gets or sets the traffic rule field of the network edge in the network dataset. The field type must be text. It is optional.
String	get/setFTSingleWayRuleValues	Gets or sets the string array which represents the from-to one-way line. It is optional.
String	get/setTFSingleWayRuleValues	Gets or sets the string array which represents the from-to one-way line. It is optional.
String	get/setProhibitedWayRuleValues	Gets or sets the string array which represents the forbidden line. It is optional.
String	get/setTwoWayRuleValues	Gets or sets the string array which represents the two-way traffic line. It is optional.
String	get/setEdgeFilter	Gets or sets the edge filter expression in the transportation analysis, optional.

Below is detailed description of certain parameters for transportation network analysis environment.

Network dataset and topological relationship fields (setNetworkDataset, setNodeIDField, setEdgeIDField, setFNodeIDField, setTNodeIDField)

Topological relationship fields include node ID fields, edge ID fields, from node ID fields, to node ID fields. Network dataset and topological relationship fields must be specified for network analysis to ensure the availability of the network and success of network analysis. For introduction to network dataset and topological relationship fields, please refer to Chapter 5.

The setNetworkDataset method of the TransportationAnalystSetting object is used to specify network dataset. setNodeIDField, setEdgeIDField, setFNodeIDField, setTNodeIDField methods are used to set node ID field, edge ID field, from node ID field and to ID field.

Edge weight field information (setWeightFieldInfos)

Edge weight is the cost to pass through this edge. Depending on the direction, we call the weight from start node to end node as from-to weight or impedance and weight from end node to start node as to-from weight or impedance. Two fields for from-to impedance and to-from impedance need to be specified from the attribute table of the network dataset. Edge eight values can be of any type, from example, time cost to pass through this edge, edge length, traffic jam condition of a certain period of time, traffic accident rate, etc.

In the TransportationAnalystSetting class, we use the setWeightFieldInfos method to set edge weight and specify a weight field info array object (WeightFieldInfos) for the method. This method is a collection of weight field info objects (WeightFieldInfo), among which each WeightFieldInfo object can possess a pair of weight fields (from-to weight field and to-from weight field) and a name that can uniquely identify this WeightFieldInfo object. Table 6.2 lists methods for the WeightFieldInfo class.

Even through only one weight field can be used for execution of transportation network analysis, however, there are advantages for specifying WeightFieldInfos. If we specify multiple types of weight values, we can easily switch among weight values while performing analysis, instead of changing transportation network analysis environment settings all the time. For instance, when we perform optimal analysis from A to B, we can specify the road length as weight value, then the analysis result would be the shortest path. If we specify time cost to pass through the road, then we will get the result for path that will cost least time. Therefore, edge weight will affect the meaning of the result.

Table 6.2 Methods for the WeightFieldInfo class

Type	Name	Description
String	get/setName	Gets or sets the name of the WeightFieldInfo object.
String	get/setFTWeightField	Gets or sets the field indicates the cost from the start to the end of the edge.
String	get/setTFWeightField	Gets or sets the field indicates the cost from the start to the end of the edge.

Barrier nodes and edges (setBarrierNodes, setBarrierEdges)

Barriers are places where traveling is prohibited. Barriers can be barrier edges or nodes. In transportation network analysis environment, you can use the setBarrierEdges and setBarrierNodes methods to set certain edges and nodes as barrier edges and nodes respectively. If an edge or a node is specified as barrier edge or node, it is prohibited to be passed through.

One thing needs to mention is that barrier edges and nodes can be specified in certain parameters while performing transportation network analysis. And the specification will be valid with the settings in transportation network analysis environment settings. This will be introduced in details in later sections.

Distance tolerance for point to edge (setTolerance)

In principle, any point on the network can be specified as a stop or barrier of the path analysis and a stop should be near the network. If the point is not on the network (neither on the edge nor on the node), the point will be included into the network according to the distance tolerance. As Figure 6-1 shows, network nodes are displayed in orange, network edges are displayed in blue, and the stop is displayed in gray. The red segment indicates the distance between the stop and the edge AB. If the distance is within the distance tolerance, the stop will be included into edge AB. This distance tolerance is specified through the setTolerance method.

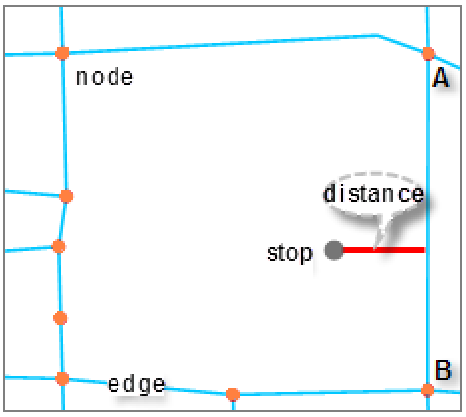


Figure 6-1 Distance tolerance from point to edge

Proper distance tolerance values is a must for correct transportation network analysis. If not properly specified, corresponding points will not be able to match to the network, therefore resulting in incorrect results or null results. Here we provide a method for your calculation of distance tolerance.

1. Gets the minimum bounding box that contains all geometric objects in this network dataset.
2. Gets the height and width of this bounding box.
3. Gets the proper tolerance value according to the formula: tolerance = Min(height, width)/40

Traffic rules (setRuleField)

Commonly used traffic rules include: from-to one way street, to-from one way street, two way street, prohibited street. From-to one way street allows traveling from start node to end node, vise versa. The setRuleField method is used for specifying fields representing traffic rules. This field should be a field of the text type and the field values are used to represent the specific traffic rules. For specifying which type of strings to represent which traffic rules, it is done using the setFTSingleWayRuleValues, setTFSingleWayRuleValues, setProhibitedWayRuleValues and setTwoWayRuleValues methods. For example, the FTSingleWayRuleValues method can be sued to set strings for from-to one way street and corresponding edges will be taken as from-to one way street.

Turn table and turn weight collection (setTurnDataset, setTurnWeightFields)

A turn table is a tabular dataset. It records cost for turning to different direction at a node. For creating turn table, please refer to the help document of SuperMap iDesktop 7C.

The setTurnDataset mehtod can be used to specify turn table, meanwhile, turn ode ID field, from edge ID field, to edge ID field in the turn table are also need to be specified through the setTurnNodeIDField, setTurnEdgeIDField and setTurnEdgeIDField methods.

Similar to edge weight values, multiple fields can be created in turn table to represent turn weights of different situations. Similarly, multiple turn weight fields can be set through the setTurnWeightFields method.

Edge name field (setEdgeNameField)

The setEdgeNameField method can be used to specify the field representing the edge name in the network dataset. Edge name can be used for display in traveling guide in transportation network analysis. If this field is not set or the specified field does not exist, the analysis result will not contain traveling guide.

Edge filter expression (setEdgeFilter)

If the edge filter expression has been set through the setEdgeFilter method, only edges satisfying the condition can participate in the transportation network analysis. Not all of the edges are needed in the practical analysis, so this method can be used to filter the edges not needed in the analysis.

6.4 Data Checking Contents

3 methods has been introduced for building network dataset in Chapter 5. As to method 3, since the network dataset is built completely according to the topological relationships specified by fields, therefore, there will be errors for the network dataset if there are errors on these fields. As to the other 2 methods, theoretically, no errors will exist. Therefore, the most possible causes for errors would be modification on network dataset.

No matters what is the reason for errors, the transportation analysis results may be influenced. Therefore, SuperMap provides the check method in the TransportationAnalystSetting class for transportation network data checking. Note that this method can be called only after transportation network analysis environment setting, that is, after the setting the method setAnalystSetting of the TransportationAnalyst class.

Syntacx:
public TransportationAnalystCheckResult check()
Return value:
The result.

This method returns a TransportationAnalystCheckResult object. You can get the edge error information and node error information with the getArcErrorInfos and getNodeErrorInfos methods. You can get the turn table error information with the TurnErrorInfos method. The above 3 methods mentioned all return Dictionary, where error information is stored, with the keys represent the SMID of the nodes or edges and the values represent the error types. The error types are represented by numbers, the meaning of which are illustrated in Table 6.3 and Table 6.4.

Table 6.3 Network dataset error types

Error type	Description
1	**Repeat node ID** Check whether there are points with repeat ID in the sub point dataset. This error type will be returned if repeat ID exists, with the error recorded in the error array.

Error type	Description
2	**Repeat edge ID** Check whether there are edges with repeat ID in the edge dataset. This error type will be returned if repeat ID exists, with the error recorded in the error array.
3	**No node for edge** In the edge attribute table of the network dataset, each record contains the start and end node ID of the edge. The edges in the edge array will be checked to see whether there are points in the sub point dataset corresponding to the nodes of the edges. This error type will be returned if any of the two nodes for an edge does not exist, with the error recorded in the error array.
4	**Spatial locations not match** Check whether edge locations match with node locations in the sub point dataset according to the from node ID and to node ID of the edge attribute table. If spatial locations do not match, this error type will be returned, with the error recorded in the error array. For example, in the attribute table of edge dataset, the from node ID and to node ID of the edge with ID of 1 are 100 and 101 respectively. However, as to spatial locations, node 100 and node 101 are not at the two ends of the edge, then this error type will be returned.
5	**Compound line object** If there are compound line objects in the network dataset, topological relationships will be caused. This error type will be returned if there are compound line objects in the network dataset, with the error recorded in the error array.

Table 6.4 Turn table error types

Error type	Description
1	**Edge for node not found** The turn table records node ID and the corresponding from edge ID and to edge ID. This error type will be returned if there are no edges in the network dataset corresponding to the node in the turn table, with the error recorded in the error array.
2	**Node not on edge** Check whether the spatial relationship of the node and its corresponding from edge and to edge in the turn table are correct (whether the nodes and edge connects at the ends of the edge). This error tpye will be returned if such errors exist, with the error recorded in the error array.

6.5 Load transportation network model Contents

When we finish setting the transportation network analysis environment, the load method of the TransportationAnalyst class needs to be called to load the transportation model. Only when the load method is called, the transportation network analysis environment will be valid for analysis.

Syntax:

public boolean load()

Return value:

True if the loading is successful, or false otherwise.

Note:

1. If there are modifications on transportaiton network analysis environment TransportationAnalystSetting or the network dataset, the load method needs to be recalled.
2. If there are no If there are modifications on transportaiton network analysis environment TransportationAnalystSetting or the network dataset, the load method does not need to be recalled. If you want to recall the method, you should dispose it first.

6.6 Transportation Network Analysis Environment Setting Contents

While performing path analysis, closest-facility analysis, traveling salesmen analysis, multiple traveling salesmen analysis, the object of the TransportationAnalystParameter type will be employed for parameter setting. Location-Allocation analysis use the LocationAnalystParameter class for parameter setting.

Parameters such as the the passed point or nodes, barrier nodes, barrier edges, weight fields name, turn weight field, whether analysis results will return passed node, edge, route collections and traveling guide, etc. Details will be introduced for the TransportationAnalystParameter class, as Table 6.5 shows:

Table 6.5 Methods for the TransportationAnalystParameter class

Type	Name	Description
int[]	get/setNodes	Gets or sets the node ID dataset passed by the analysis. It is required, but setNodes and setPoints are exclusive with each other. If you set them at the same time, only the last set before the analysis is valid. For example, firstly you specify the nodes data and then the coordinates dataset, then it will only analyze the coordinate points.
Point2Ds	get/setPoints	Gets or sets the node ID collection passed by the analysis. It is required, but setNodes and setPoints are exclusive with each other. If you set them at the same time, only the last set before the analysis is valid. For example, firstly you specify the nodes data and then the coordinates dataset, then it will only analyze the coordinate points.
String	get/setWeightName	Gets or sets the name of the Weight field information. It is the value specified by the setName method of a certain WeightFieldInfo object in the WeightFieldInfos colllection in TransportationAnalystSetting class. The default value is the Name value of the first WeightFieldInfo in the WeightFieldInfos if you don't set it in the analysis.
String	get/setTurnWeightName	Gets or sets the turn weight field, and it is optional. The turn weight field is one value in the setTurnWeightFields collection specified in TransportationAnalystSetting.
int[]	get/setBarrierEdges	Gets or sets the ID list of the barrier nodes, and it is optional.
int[]	get/setBarrierNodes	Gets or sets the ID list of the barrier nodes, and it is optional.
Point2Ds	get/setBarrierPoints	Gets or sets the ID list of the barrier nodes, and it is optional.
boolean	is/setEdgesReturn	Gets or sets whether to keep the passed edges in the analysis result. If true, after the analysis is successful, you will get an array of the passed edges from the getEdges method in the TransportationAnalystResult object; and if false, you will get a null array. The default value is false.
boolean	is/setNodesReturn	Gets or sets whether to keep the passed nodes in the analysis result. If true, after the analysis is successful, you will get an array of the passed nodes from the getNodes method in the TransportationAnalystResult object; and if false, you will get a null array. The default value is false.
boolean	is/setPathGuidesReturn	Gets or sets whether to keep the traveling guide in the analysis result. If true, after the analysis is successful, you will get an array of the traveling guide from the getPathGuides method in the TransportationAnalystResult object; and if false, you will get a null array. The default value is false.

Type	Name	Description
boolean	is/setRoutesReturn	Gets or sets whether to keep the routes (collection of GeoLineM objects) in the analysis result. If true, after the analysis is successful, you will get an array of the routes from the getRoutes method in the TransportationAnalystResult object; and if false, you will get a null array. The default value is false.
boolean	is/setStopIndexesReturn	Gets or sets whether to keep the stop indexes in the analysis result. If true, after the analysis is successful, you will get an array of the stop indexes from the getStopIndexes method in the TransportationAnalystResult object; and if false, you will get a null array. The default value is false.

Node and coordinate modes (setNodes, setPoints)

Node mode indicates that analysis points are nodes and users need to specify IDs of corresponding nodes. Coordinates point mode indicates that analysis points are coordinate points. The two modes are supported by optimal path analysis, closest facility analysis, traveling salesman analysis, multiple traveling salesmen analysis, service area analysis. Note that not all points used in analysis are specified using setNodes and setPoints. Different analysis functions need to specify different analysis points in TransportationAnalystParameter.

The two modes are exclusive with each other. If they are set at the same time, the last setting before analysis will be applied.

Barriers (setBarrierNodes, setBarrierEdges, setBarrierPoints)

The setBarrierNodes, setBarrierEdges, setBarrierPoints methods are used to set barrier nodes, barrier edges and barrier points. While setting transportation network analysis environment, barrier nodes and barrier edges can also be set. All settings will be valid for transportation network analysis.

Here we need to clarify that different from barrier nodes and barrier edges, it is possible that barrier points are not on the network (either edges or nodes). Therefore, we need to specify distance tolerance using setTolerance to assign barrier points to the nearest network. The distance tolerance is specified in traveling salesmen network analysis environment, please refer to 6.3 section for details. Currently, barrier points are supported from optimal path analysis, closest facility analysis, traveling salesman analysis, logistics analysis.

Whether to return traveling guide (setPathGuidesReturn)

First, let us learn what is traveling guide (path guide).

The path guide records path information of TransportationAnalyst in optimal path analysis, find closest facility, TSP, MTSP, etc. A path guide object contains a path line from the start to the end. The path guide items consists of the path guide item PathGuideItem. A path guide has some features: such as stops, network nodes, network edges, etc. These key information such as such as length, direction of those features are described by path guide items. Or, we can say that path guide items are key features on the path guide line.

Path guide items can be categorized into 5 types:

Stop: The analysis point to be used. Such as the points used in optimal path analysis.

Line Segment: When the stop is a common coordinate point, you should add the stop to the network. And then you can analyze based on the network. Please refer to the setTolerance method of the TransportationAnalystSetting calss.

As shown below, the red dotted line is the shortest distance from the stop to network. Note: when the stop is near the edge of the network segment. as shown below. This distance refers to the distance between stops and endpoints.

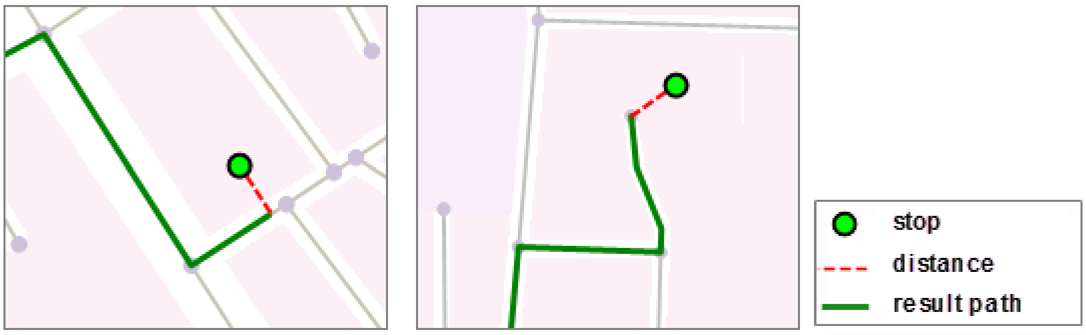


Figure Distance from stop to network

The point corresponding to the stop: The point is the corresponding point when adding the stop to the network.

Road Section: A section road when driving. In transportation analysis, we use arcs to simulate the road, so the road sections are in the arcs. Note: Multiple arcs may merger into a path guide item when the names of arcs are same, and the corner between adjacent arcs is less than 30 degrees.

As shown below, use different colors to mark sections between two stops. The first road section after stop 1 meet the conditions, but it is the first road section after the stop, so them don't merge. The blue road sections meet the conditions, so them merged into a path guide item. The pink road sections have the different arc names, so it is another path guide item. The green road section is the last section before the stop, so it is path guide item.

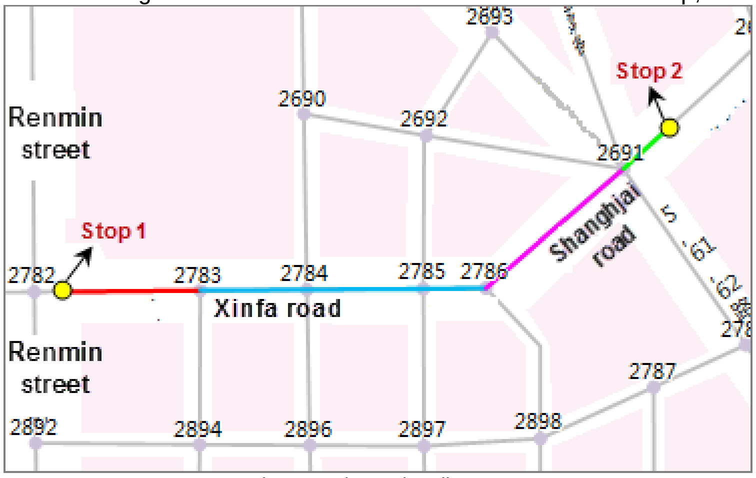


Figure Roda section diagram

Turning Point: The crossing between two adjacent sections . The crossing means the actual road crossing (crossroads or T-junction) that is likely to change the direction. As shown above, the 2783, 2786 and 2691 are all turning point. The turning point must be a network node.

The path guide item object can be used to get the ID, name, index, weight, length of the key features of the path. It can also help determine whether the item is an edge or a stop, the traveling direction, the turn type, cost, etc. The way to get to the destination from the start point can be provided through extracting and organizing the information of the key features stored in path guide items labeled with sequence numbers. Table 6.6 lists methods for the PathGuideItem class.

Table 6.6 Methods for the PathGuideItme class

Type	Name	Description
int	getID	Gets the ID of the path guide item. Except the following 3 situations, this method will always return -1. When the path guide item is the stop of node mode, and the stop is node, the ID is the node will be returned. When the path guide item is the turning point, and the turning point is node, the ID is the node will be returned. When the path guide item is the road segment, this value is the arc ID corresponding to the segment. If the road segment is combined by multiple arcs, the ID of fist segment will be returned.
boolean	isEdge	Returns whether the path guide item is of line or point type. If it is true, representing the line, such as the road segment, line and so on. If it is false, representing the point.
boolean	isStop	Gets whether the path guide item is a stop, or the corresponding point added to the network. When IsStop is true, the corresponding path guide item may be stop. Or when the stop is the coordinate point, it is the corresponding point in the network.
String	getName	Gets the name of the path guide item. Except the following 2 situations, this method will always return null string. When the path guide item is stop or turning point, this value is specified according to the node name field setNodeNameField in transportation analyst. Or it is the null string. When the path guide item is road or the line from stop to network, this value is specified according to the node name field setEdgeNameField in transportation analyst. Or it is the null string.
double	getLength	Gets the corresponding line length when the path guide item is line (namely the isEdge is true). The unit is the same as the unit used in network dataset. Length is valid only if isEdge is true. Or it is 0.0.
GeoLine	getGuideLine	Gets the corresponding path guide line segment when the item is line (isEdge is true). The ID of the node when the isEdge method returns false.
Rectangle2D	getBounds	Gets the range of this path guide item. If this item is line (isEdge is true), it is the enclosing rectangle. If this item is point (isEdge is false), it is the point itself.
DirectionType	getDirectionType	Gets the direction of the path guide item. It is valid only when the path guide item is line (isEdge is true). They can be east, south, west, north. While isEdge returns false, this method returns NONE, indicating no direction.
SideType	getSideType	Get whether the stop is left side, right side or on the road when this path guide item is stop. When the path guide item is other types except the stop, this method returns NONE.
double	getTurnType	When the path guide item is the point (i.e. isEdge returns false), get the next turning direction of the point.
TurnType	getTurnAngle	When the path guide item is the point (i.e. isEdge returns false), get the next turning direction of the point. The unit is degrees, with accuracy to 0.1 degrees. While isEdge returns true, this method returns -1.

Type	Name	Description
double	getDistance	Gets the distance from a stop to the network, and it is supported only when the path table item is the stop. The unit is the same as the network dataset used to analyze. A stop can not be on a network (neither on a edge nor on a node). You should add it to the network to analyze. This distance is the distance between the stop and the nearest edge. Please refer to the introduction to the setTolerance method of the TransportationAnalystSetting class for details.
int	getIndex	Gets the number of the path guide item. Except the following 2 situations, this method will always return -1. When the path guide item is a stop, this value is the sequence numbers in all stops, and it starts with 1. For example, if a stop is the second stop passed by the path, the Index of this stop is 2: When the path guide item is the tunning point, this value is the crossing number from this point to last turning point.
double	getWeight	Gets the weight of a path guide tem, i.e., the cost of the path guide subitem. It has the same unit with the weight field of the WeightFieldInfo object specified by the setWeightName method of the TransportationAnalystParameter class. When the path guide item is stop of road, turning point node, the cost is meaningful. Or it is 0.0. When path guide item is the road section, get the corresponding cost according to the arc and turn weights. If the turn table isn't set, the weight is 0; When the path guide item is the turning point or the stop, it is the corresponding turn weight. If the turn table isn't set, it is 0.0.

An example is provided below for you to understand path guide item and path guide vividly.

In Figure 6-2, the blue line is a path of find closest facility. In the result, you can get the path guide of this road. There are 7 items in this path guide. 2 stops (start and end nodes), 3 edges (road sections) and 2 network nodes (crosses).

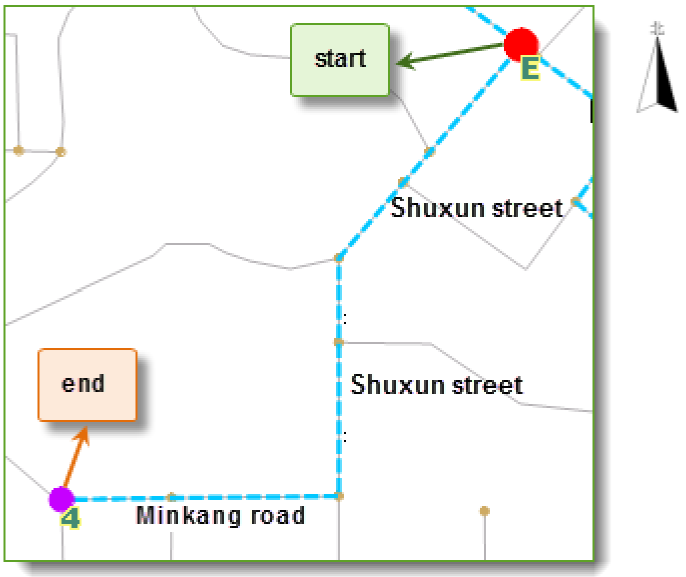
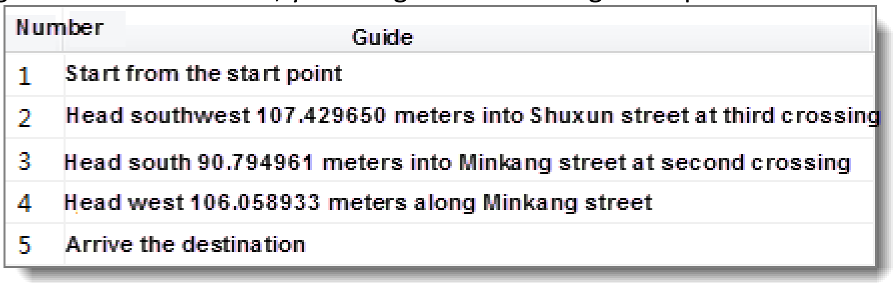


Figure Optimal path analysis

The following table lists the information for the 7 path guide items in the above figure, including whether it is stop, whether it is edge, Index, length, name, direction type and turn type, edge name etc. Note that not all information such as turn angle, cost, distance from stop to edge, etc. is listed here. Users can decide which fields are needed according to demands.

getID	isStop	isEdge	getIndex	getLength	getName	getDirectionType	getTurnType
0	true	false	1	0.0	——	NONE	AHEAD
1	false	true	-1	107.42965	Shuxun Street	SOUTH	NONE
2	false	false	3	0.0	——	NONE	LEFT
3	false	true	-1	90.79496	Shuxun Street	SOUTH	NONE
4	false	false	2	89.222265	——	NONE	RIGHT
5	false	true	-1	106.05893	Mingkang Road	WEST	NONE
6	true	false	2	0.0	——	NONE	END

Organizing the item information, you can get the following description.



Number	Guide
1	Start from the start point
2	Head southwest 107.429650 meters into Shuxun street at third crossing
3	Head south 90.794961 meters into Minkang street at second crossing
4	Head west 106.058933 meters along Minkang street
5	Arrive the destination

Figure Path guide

Note that if edge name field is not specified in TransportationAnalystSetting or the field does not exist, the path guide will not exist in the results even if setPathGuidesReturn is set to true. That is because if it is the case, the road (edge) name can not be provided for guide.

Above is operation contents before transportation network analysis. The transportation network analysis environment setting is especially important. Below is the introduction to implementation process and related concepts for all transportation network analyses.

6.7 Optimal Path Analysis Contents

The optimal path analysis is to find the path which has the least impedance to get from one point to another, and the result path will visit the points in the specified order if there are more than two points to visit. "The least impedance" can be defined as the least time, the minimum cost, the best scenery, the best road condition, the least overpasses, the least toll stations or the maximum villages, etc.

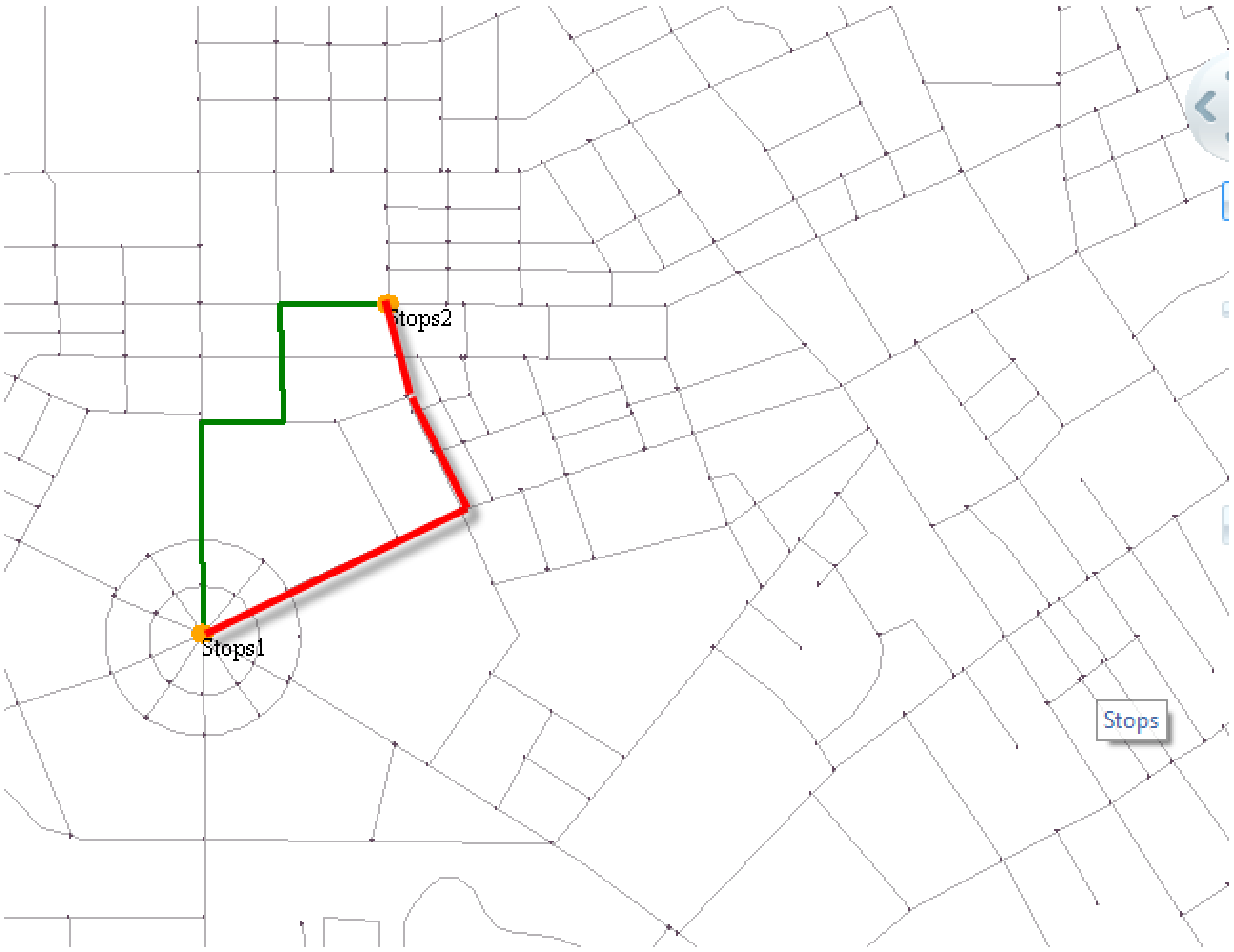


Figure 6-3 Optimal path analysis

The traveling order must be the same as the order of the specified stops is a key characteristic of optimal path analysis. We can understand optimal path analysis as the finding optimal path between two stops sequentially according to a certain order. Example: To visit 1, 2, 3, 4 orderly and find out the optimal path, we need to find out the optimal path R1_2 between 1 and 2, R2_3 between 2 and 3, R3_4 between 3, and 4 firstly. Then the optimal path to visit the 1, 2, 3, and 4 is R= R1_2 + R2_3 + R3_4.

6.7.1 Implementation of Optimal Path Analysis

The findPath method of the TransportationAnalyst class is used to implement optimal path analysis. According to the steps for optimal path analysis, after loading network model, we can call this method to perform optimal path analysis.

Syntax:

public TransportationAnalystResult findPath(TransportationAnalystParameter parameter,boolean hasLeastEdgeCount)

Parameters:

parameter: The specified transportation network analysis parameter.

hasLeastEdgeCount: Whether to carry out the optimal path analysis according to the least number of arcs. True means analyzing according to the least number of arcs.

Return value:

The analysis result object.

The N stops for the optimal path analysis are specified in parameter of the TransportationAnalystParameter type. Node mode can be selected for specifying stops using the setNodes method of the TransportationAnalystParameter object. Coordinate point mode can be selected for specifying stops using the setPoints method. The order of stops during optimal path analysis is the order for specifying points. Note that the two modes are exclusive with each other. Fore more details, please refer to 6.6.

If hasLeastEdgeCount has been set to true, the result path has the least number of edges. In this case, the result path may not be path with least cost as the number of the edges is not equal to the weight of the edge. Takes figure 6-4 as an example, there are two paths connected AB and the number of edges in the green path is less than the yellow one. If this parameter is set to true, the result path is the green one; and the result path is the yellow one if this parameter is set to false.

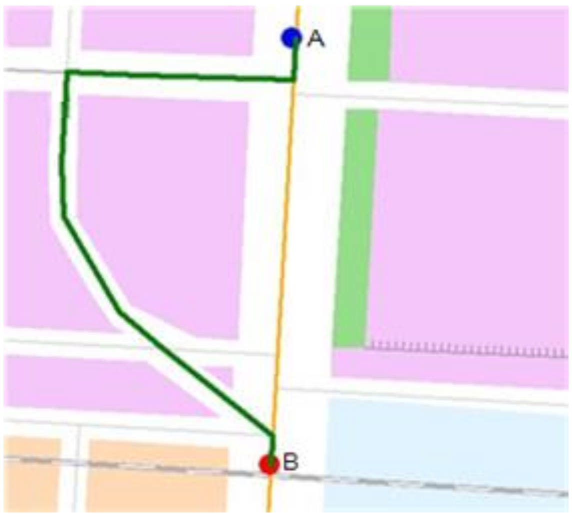


Figure 6-4 Whether to find path with least number of edges

6.7.2 Results of Optimal Path Analysis

As to optimal path analysis, closest-facility analysis, traveling salesman analysis, multiple traveling salesmen analysis, the results are returned through a TransportationAnalystResult object. This object can help return passed nodes, edges, route collections, traveling guide, weight array, etc. Table 6.7 lists methods for the TransportationAnalystResult class.

Table 6.7 Methods for the TransportationAnalystResult class

Type	Name	Description
int[][]	getNodes	Gets the node collection of the analysis result. Note: The setNodesReturn method of the TransportationAnalystParameter object must be set to true so that the analysis results can contain the node collection; otherwise, it is a null array.
int[][]	getEdges	Gets the edge collection of the analysis result. Note: The setNodesReturn method of the TransportationAnalystParameter object must be set to true so that the analysis results can contain the edge collection; otherwise, it is a null array.

Type	Name	Description
GeoLineM[]	getRoutes	Gets the routes dataset of the analysis result that is the dataset of the GeoLineM object. Note: The setRoutesReturn method of the TransportationAnalystParameter objects must be set to true so that the analysis results can contain the routes dataset; otherwise, it is a null array.
PathGuide[]	getPathGuides	Gets the path guides. Note: The setPathGuidesReturn method of the TransportationAnalystParameter object must be set as true so that the analysis results can contain the path guides; otherwise, it is a null array.
int[][]	getStopIndexes	Gets the two-dimensional array of the stop indexes which reflects the order of the stops after analysis. Note: The setStopIndexesReturn method of the TransportationAnalystParameter object must be set as true so that the analysis results can contain the stop-index dataset; otherwise, it is a null array.
double[]	getWeights	Gets the weight array represented the cost. It has the same unit with the weight field of the WeightFieldInfo object specified the setWeightName method of the TransportationAnalystParameter object.
double[][]	getStopsWeights	Gets the cost (weight) between the stops after sorting the stops according to the index. It has the same unit with the weight field of the WeightFieldInfo object specified the setWeightName method of the TransportationAnalystParameter object.

Above is the introduction to methods for TransportationAnalystResult. Below is the introduction to the meaning of optimal path analysis results.

Passed edge collection (getEdges): As the analysis result only have one path, the one-dimensional length of the array is 1, and the 2D elements are the ID of the edges passed by the path.

Passed node collection (getNodes): As the analysis result only have one path, the one-dimensional length of the array is 1, and the 2D elements are the ID of the nodes passed by the path.

Route object collection (getRoutes): Only one result path.

Weight array (getWeights): Since the analysis only has one result route, there is only one weight value which is the total cost of the path.

Stop index (getStopIndexes): The analysis result only have one path, so the one-dimensional length of the array is 1, and the 2D elements denote the orders of the stops passed by the result path:

1. Node mode: If the node ID of the array we set is 1, 3, 5, as the result order of passed nodes ID should be the same to the node order we set, so the value of the 2D elements will be 0, 1, 2, i.e., the node index of analysis order in the initial node array.
2. Coordinate points mode: If the specified coordinate points we set are Pnt1, Pnt2, Pnt3, as the result order of passed points is Pnt1, Pnt2, Pnt3, so the value of the 2D elements will be 0, 1, 2, i.e., the node index of analysis order in the initial node array.

Stop weight (getStopWeights): As the analysis result only have one path, the one-dimensional length of the array is 1. If it is specified to pass the point 1, 2, 3, then the values of the array elements are the cost from 1 to 2, and the cost from 2 to 3.

6.8 Closest Facility Analysis Contents

The closest facility analysis is to specify an event or a set of facilities and find out one or more facilities that are able to be reached with the minimum cost from the event. The result is the optimal path from the event to the facility (or from the facility to the event).

Facilities and events are the basic features for closest facility analysis. Facilities are schools, supermarkets, gas station, etc. that provide services. Event points are the event locations that need the service of the facilities.

For instance, an accident happened at certain location. it is needed to query the 3 hospitals that can be reached within 10 minutes, the hospitals that can be reached with more than 10 minutes will not be considered. In this example, the event location is the event point and the nearby hospitals are the facilities. Analysis result is as figure 6-5 shows that paths to 3 hospitals with shortest time will be provided. Since closest facility analysis is actually path analysis, barrier edges and the barrier nodes, which cannot be transversed, can be set for analysis.

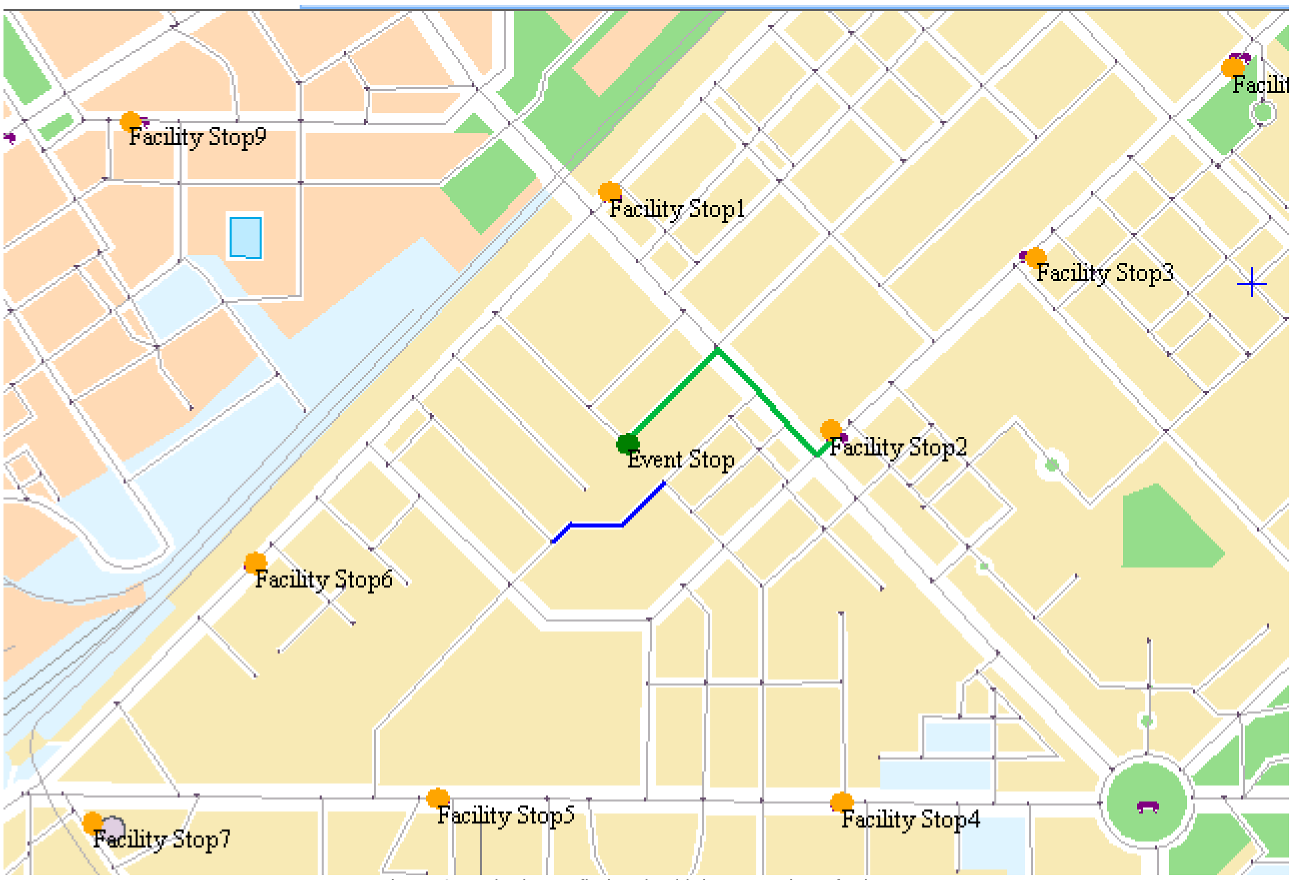


Figure 6-5 Whether to find path with least number of edges

6.8.1 Implementation of Closest Facility Analysis

According to the steps for closest facility analysis, after loading network model, we can call this method to perform analysis. The TransportationAnalyst class provides two overwritten findClosestFacility methods for implementing closest facility analysis.

Method 1: Conducts the closest facility analysis according to the given parameters, and the event point is the node ID.

Syntax:

public TransportationAnalystResult findClosestFacility(TransportationAnalystParameter parameter,int eventID, int facilityCount, boolean isFromEvent, double maxWeight)

Parameters:

parameter: The specified transportation network analysis parameter.

eventID: ID of node specified as event point.

facilityCount: Number of facilities to search.

isFromEvent: Whether to find from the event site to the facilities. If the direction is from the event site to the facilities, the field value is true; the default is false, meaning from the facilities to the event site.

maxWeight: The unit is the same as the weight filed set in the NetworkAnalystSetting class. To search the entire network, this value can be set to 0.

Return value:

The TransportationAnalystResult object.

Method 2: Conducts the closest facility analysis according to the given parameters, and the event point is the coordinate point.

Syntax:

public TransportationAnalystResult findClosestFacility(TransportationAnalystParameter parameter,Point2D eventPoint, int facilityCount, boolean isFromEvent, double maxWeight)

Parameters:

parameter: The specified transportation network analysis parameter.

eventPoint: Position of the specified event point.

facilityCount: Number of facilities to search.

isFromEvent: Whether to find from the event site to the facilities. If the direction is from the event site to the facilities, the field value is true; the default is false, meaning from the facilities to the event site.

maxWeight: The unit is the same as the weight filed set in the NetworkAnalystSetting class. To search the entire network, this value can be set to 0.

Return value:

The analysis result object.

The event point could be coordinate point or node ID, specified by the eventID and eventPoint parameters of the above two methods respectively. Facility points are specified in parameter of TransportationAnalystParameter. Similarly, there are two methods for specifying facility points: node mode and coordinate point mode. As to node mode, facility node collection can be specified through the setNodes method of the TransportationAnalystParameter object. As to coordinate point mode, coordinate point collection can be specified through the setPoints method. Please refer to section 6.6 for details.

Except for facility points, the TransportationAnalystParameter object can also be used for specifying other parameters for analysis. Please refer to section 6.6 for details.

6.8.2 Closest Facility Analysis Results

Results for closest facility analysis need to be returned by methods of the TransportationAnalystResult class. For description of these methods, please refer to section 6.6. Here the meaning of the analysis results are introduced below:

Passed edge collection (getEdges): As the number of the paths in the result is identical with the number of closest facilities, the one-dimensional length of the array is the number of the facility points in the result, and the 2D elements are the ID of the edges passed by the path.

Passed node collection (getNodes): As the number of the paths in the result is identical with the number of closest facilities, the one-dimensional length of the array is the number of the facility points in the result, and the 2D elements are the ID of the nodes passed by each path.

Route object collection (getRoutes): The count of result routes is identical to the facility node's number.

Weight array (getWeights): The number of weight value is identical to the count of the facility nodes. Each element denotes the total cost from the event point to the facility point.

Stop index (getStopIndexes): Invalid method.

Stop weight (getStopWeights): Invalid method.

6.9 Multiple Traveling Salesman Analysis Contents

Traveling salesmen problem analysis (TSP) is to find the path passing a series of specified nodes. The TSP is an unordered path analysis. The Traveling salesmen can determine the order of accessing the nodes for themselves, in order to get the minimum impedance (or close to the minimum) of the traveling path.

The traveling salesman analysis, by default, automatically uses the first point (node or coordinate point) of the points set passed by the traveling salesman analysis as the start point. Moreover, users also can specify the end point. If you select to specify the end point, then the last point of the specified points set is the end point when the salesman leaves from the first specified point and ultimately arrives the specified end point. If the end point is not specified, the salesman starts from the start point, then order to pass through other points is determined by the salesman according to the principle of least cost. Figure 6-6 are examples for traveling salesman analysis with end point specified and not specified.

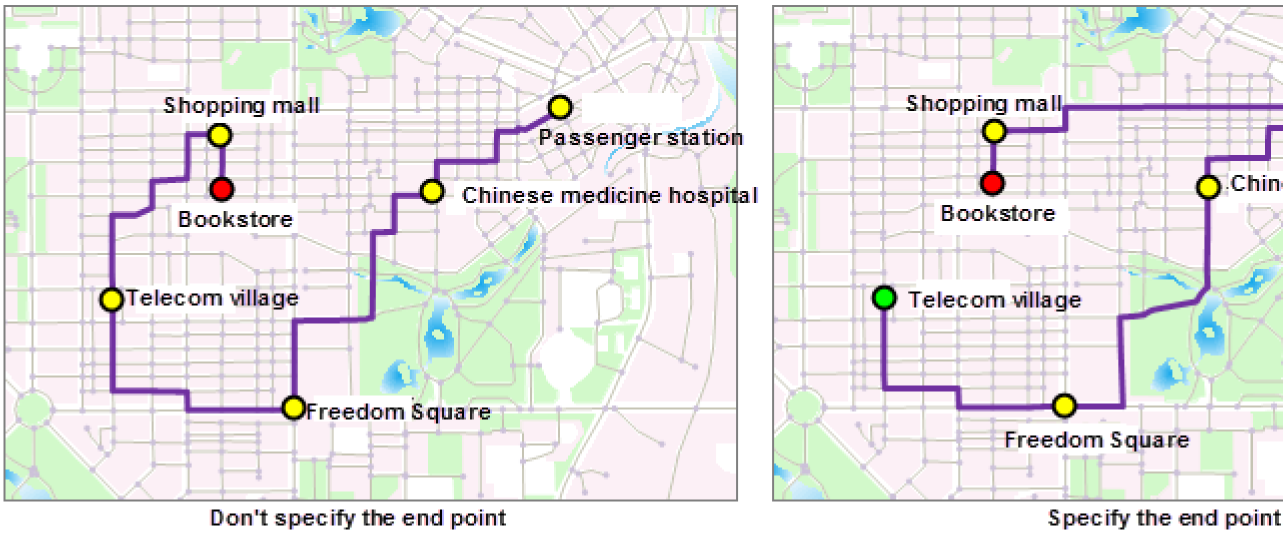


Figure 6-6 Traveling salesman analysis with end point specified or not

In addition, if you select to specify the end point and the end point is identical with the start point, i.e., the last passed point is identical with the first point, then the result of the traveling salesmen analysis is a closed path which is that the salesman leaves from the first specified point and ultimately arrives this point, as shown in Figure 6-7.

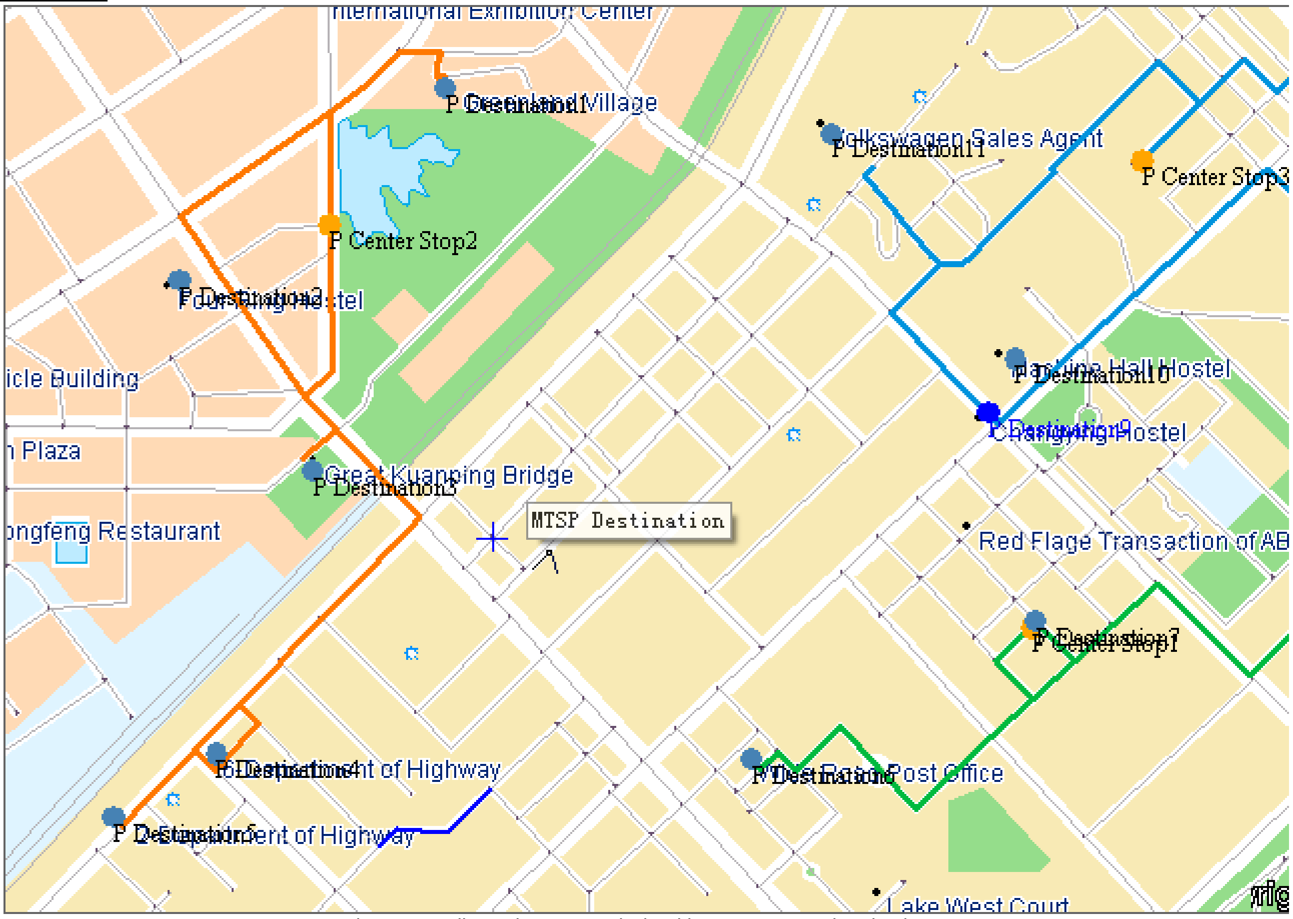


Figure Traveling salesman analysis with same start and end points

Note that optimal path analysis is similar to traveling salesman analysis that they both need to traverse all points to get path with least cost. There are differences in the order when traversing all nodes. Optimal path analysis must access all nodes in the specified order, but traveling salesman analysis accesses all nodes in the optimal order, not in the specified order.

6.9.1 Traveling Salesman Analysis Implementation

According to the steps for traveling salesman analysis, after loading network model, we can call this method to perform analysis.

Syntax:

public TransportationAnalystResult findTSPPath(TransportationAnalystParameter parameter, boolean isEndNodeAssigned)

Parameters:

parameter: The specified transportation network analysis parameter.

isEndNodeAssigned: Whether to specify the end node. Yes if true when the last node in the nodes set is the end node; no if false.

Return value:

The TransportationAnalystResult object.

Parameters for traveling salesman analysis can be specified through the parameter of the TransportationAnalystParameter type. Stops for traveling salesman analysis can be of node mode or coordinate point mode. As to node mode, node ID collection can be specified through the setNodes method of the TransportationAnalystParameter object. As to coordinate point mode, coordinate point collection can be specified through the setPoints method. Please refer to section 6.6 for details.

The TransportationAnalystParameter object can also be used for specifying other parameters for analysis. Please refer to section 6.6 for details.

6.9.2 Traveling Salesman Analysis Results

Results for traveling salesman analysis need to be returned by methods of the TransportationAnalystResult class. For description of these methods, please refer to section 6.6. Here the meaning of the analysis results are introduced below:

Passed edge collection (getEdges): As the analysis result only have one path, the one-dimensional length of the array is 1, and the 2D elements are the ID of the edges passed by the path.

Passed node collection (getNodes): As the analysis result only have one path, the one-dimensional length of the array is 1, and the 2D elements are the ID of the nodes passed by the path.

Route object collection (getRoutes): Only one result path.

Weight array (getWeights): Since the analysis only has one result route, there is only one weight value which is the total cost of the path.

Stop index (getStopIndexes): The analysis result only have one path, so the one-dimensional length of the array is 1, and the 2D elements denote the orders of the stops passed by the result path:

1. Node mode: If the node ID of the array we set is 1, 3, 5, but the result order is 3, 5, 1, then the value of the 2D elements will be 1, 2, 0, i.e., the nodes index of the analysis order in the initial node array.
2. Coordinate points mode: If the specified coordinate points are Pnt1, Pnt2, Pnt3, but the result order is Pnt2, Pnt3, Pnt1, as the result order of passed points is Pnt1, Pnt2, Pnt3, then the value of the 2D elements will be 1, 2, 0, i.e., the coordinate points index of the analysis order in the initial coordinate points array.

Stop weight (getStopWeights): As the analysis result only have one path, the one-dimensional length of the array is 1. If it is specified to pass the point 1, 2, 3, and the stops' index is 1, 0, 2, then the values of the 2D elements are the cost from 2 to 1, and the cost from 1 to 3.

6.10 Multiple Traveling Salesmen Analysis Contents

The multiple traveling salesman analysis is to find the shortest delivery route from M points of origins to N points of destinations (M and N are integers greater than 0). The MTSP of SuperMap Desktop .NET is used for finding the least cost order and route of the distribution.

The results of MTSP include the destinations of each center, and the order to pass the destinations and the corresponding routes so that the delivery cost of the center is minimum or the total cost of all the centers is minimum. Moreover, the center will ultimately return to the center after finishing the task for its responsible destinations.

For example, there are 50 newspaper retailers (delivery destinations) and 4 newspaper suppliers (delivery centers) now. You need to search for the optimal paths of these 4 suppliers to deliver newspapers to the newspaper retailers. This is a MTSP problem.

Figure 6-8 shows the result to deliver the newspapers, where the bigger red dots represent 4 newspaper suppliers (delivery centers), and the other smaller dots represent newspaper retailers (delivery destinations). Each color represents the delivery scheme of a delivery center including the delivery destinations, delivery order and delivery path.

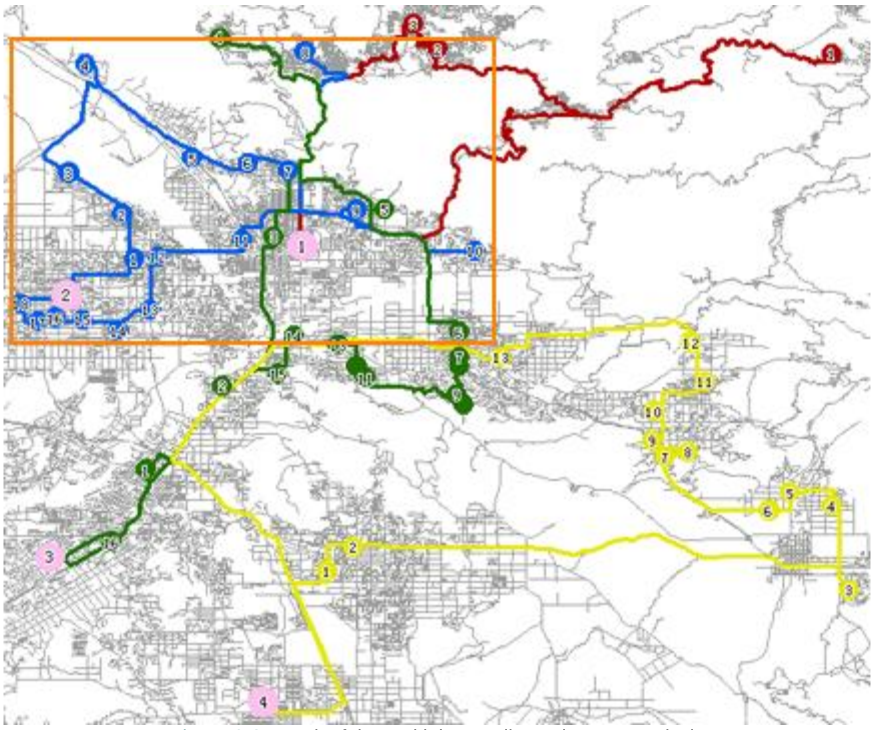


Figure 6-8 Result of the multiple traveling salesmen analysis

Figure 6-9 shows the delivery scheme of the No. 2 center within the rectangle box of the figure above. The blue and numbered small dots are delivery destinations that the No. 2 delivery center is responsible for (there are 18 in total). The No. 2 delivery center will deliver newspapers one by one according to the numbered order on the delivery destinations. It delivers to the No. 1 newspaper retailer first, and then delivers to the No. 2 newspaper retailer, and so on. After completing the delivery along the blue path derived from the analysis, it goes back to the delivery center.

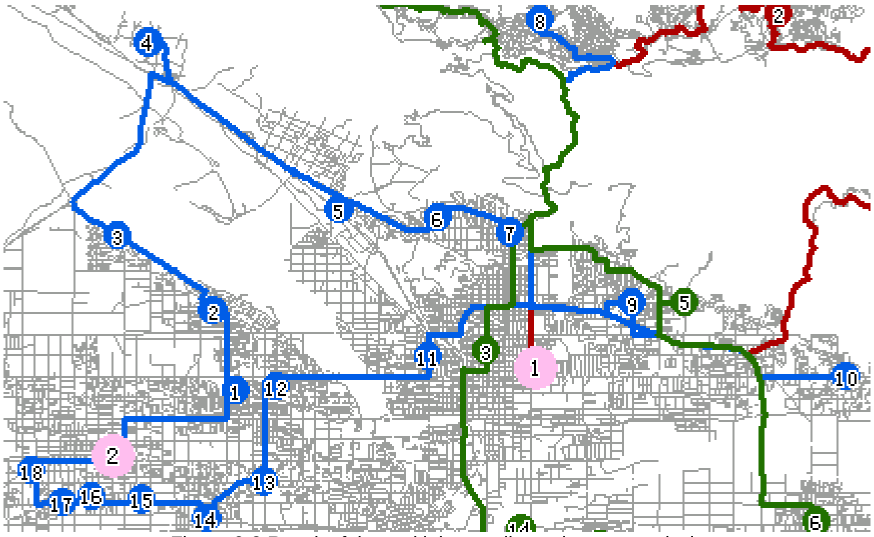


Figure 6-9 Result of the multiple traveling salesmen analysis

Note: The objective of MTSP is to find the least total cost scheme or the scheme that the cost of each center is minimum. Hence, some centers may not be involved in the delivery task.

6.10.1 Call Multiple Traveling Salesmen Analysis

According to the steps for multiple traveling salesmen analysis, after loading network model, we can call this method to perform analysis. The TransportationAnalyst class provides two overwritten findMTSPPath methods for implementing multiple travelling salesmen analysis.

Method 1: The method is used for the MTSP (logistics) which the delivery centers are the node ID array.

Syntax:

public TransportationAnalystResult findMTSPPath(TransportationAnalystParameter parameter,int[] centerNodes, boolean hasLeastTotalCost)

Parameters:

parameter: The specified transportation network analysis parameter.

centerNodes: The array of IDs for the specified supply centers.

hasLeastTotalCost: Specify whether to use the minimal total cost schema. There are two schemas can be selected in performing Logistics. One is the minimal total cost, in this case, it is possible that some delivery centers may cost more and others cost less. Another is the optimal in each delivery center, but the total cost may not be the minimum.

Return value:

The TransportationAnalystResult object.

Method 2: The method is used to analyze Multiple TSP which the delivery center points are a series of coordinates.

Syntax:

public TransportationAnalystResult findMTSPPath(TransportationAnalystParameter parameter,int[] centerNodes, boolean hasLeastTotalCost)

Parameters:

parameter: The specified transportation network analysis parameter.

centerPoints: The collection of coordinate points specified as supply centers.

hasLeastTotalCost: Specify whether to use the minimal total cost schema. There are two schemas can be selected in performing Logistics. One is the minimal total cost, in this case, it is possible that some delivery centers may cost more and others cost less. Another is the optimal in each delivery center, but the total cost may not be the minimum.

Return value:

The analysis result object.

Supply centers of multiple traveling salesmen analysis can be nodes or coordinate points. For method 1, users can use the centerNodes parameter to specify IDs of nodes for supply centers. For method 2, the centerPoints parameter can be used for setting coordinates of supply centers.

Destinations are specified using the parameter of the TransportationAnalystParameter type. Destinations can also be of node mode or coordinate point mode. As to node mode, node ID collection can be specified through the setNodes method of the TransportationAnalystParameter object. As to coordinate point mode, coordinate point collection can be specified through the setPoints method. Please refer to section 6.6 for details.

The TransportationAnalystParameter object can also be used for specifying other parameters for analysis. Please refer to section 6.6 for details.

6.10.2 Multiple Traveling Salesmen Analysis Results

Results for multiple traveling salesmen analysis need to be returned by methods of the TransportationAnalystResult class. For description of these methods, please refer to section 6.6. Here the meaning of the analysis results are introduced below:

Passed edge collection (getEdges): The one-dimensional length of the array is the number of centers involved in the delivery, in which each center is corresponding to a path, and the 2D elements are the ID of the edges passed by each path. Note: If the delivery mode is the optimal mode in each delivery center, all the centers will be involved in the delivery; if it is the least cost mode, the number of the centers involved in the delivery may be less than the number of the specified centers.

Passed node collection (getNodes): The one-dimensional length of the array is the number of centers involved in the delivery, in which each center is corresponding to a path, and the 2D elements are the ID of the nodes passed by each path. Note: If the delivery mode is the optimal mode in each delivery center, all the centers will be involved in the delivery; if it is the least cost mode, the number of the centers involved in the delivery may be less than the number of the specified centers.

Route object collection (getRoutes): If the delivery mode is optimized each delivery center, the count of the result routes is identical to the number of supply centers. If it is the least total cost mode, the count of the result routes may be less than the number of supply centers.

Weight array (getWeights): The number of weight value is identical to the count of the delivery centers in the analysis result. Each element denotes the total cost of the corresponding center. Note: If the delivery mode is optimizing each delivery center, all the centers will be involved in the delivery; if it is the least total cost mode, the number of the centers involved in the delivery may be less than the number of the specified centers.

Stop index (getStopIndexes): It is similar with the traveling salesmen analysis. The one-dimensional length of the array is the number of centers involved in the delivery. The meanings of the 2D elements are identical to the traveling salesmen analysis, which denote the order of the centers passed by each path. Note: If the delivery mode is optimizing each delivery center, all the centers will be involved in the delivery; if it is the least total cost mode, the number of the centers involved in the delivery may be less than the number of the specified centers.

Stop weight (getStopWeights): The analysis result may include multiple paths, so the one-dimensional length of the array is the number of the paths, and the 2D elements are the cost between the stops passed by the path. But note that the stops of MTSP include the center point, and the start/end point is the center point. For example, a result path begins from the stop 1, passes the stops 2, 3, 4, and the corresponding index is 1, 2, 0, then the weights of the stops are the cost from 1 to 3, from 3 to 4, from 4 to 2, and from 2 to 1.

6.11 Service Area Analysis Contents

The service area is the area that within a certain impedance taking specified point as the center and containing all the accessible edges. The service area analysis is to find the service range for a specified location (center point) according to a specified impedance value (service radius). The impedance can be time, distance or any other cost. For example, when you want to find the 30-minute service area for a specified point, within the result service area, it takes less than 30 minutes from any point to the specified point.

Service Area Analysis can be understood as not to consider the supply and the demand of the resource center, but only the resource allocation which only considers the edge resistance between the provider and the demander. This type of analysis is generally used to evaluate and analyze the service scopes of the public facilities, such as post office, hospital, supermarket, etc., so as to provide the references for selecting the optimal locations of the public facilities.

The results of the service area analysis cover all the routes and areas that the service center points can reach. The route is the path along the network edges from the service center point according to the rule that the impedance value is less than or equal to a specified service radius. The service area is a polygon area that covers all the routes of the center point according to a specified algorithm.

Figure 6-10 shows the problems to be solved by the service area analysis and what information will be provided. The red dots in the diagram represent service centers, and the region areas with different colors are service areas within the specified impedance. And the routes are denoted by the corresponding color of the service area.

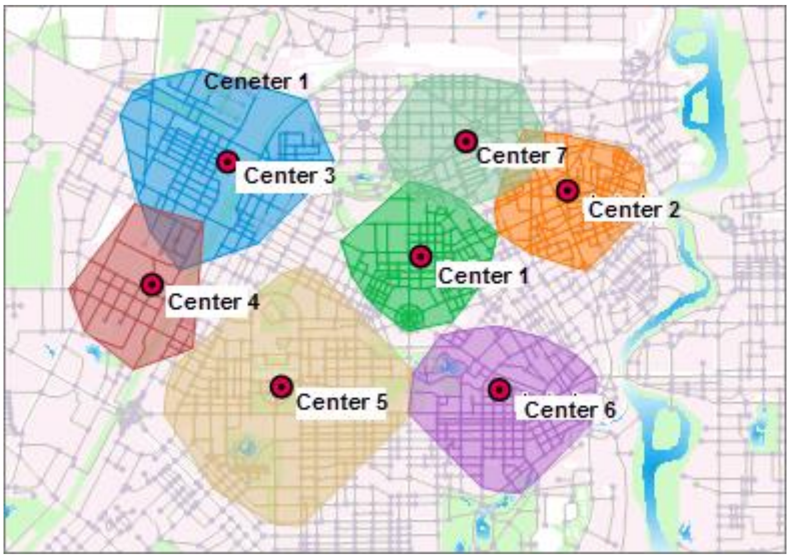


Figure 6-10 Service area analysis

If two or more service areas intersect, you can do the mutually exclusive process for them. After the mutually exclusive process, these service areas will not intersect. In Figure 6-11, the left/right figure is before/after the mutually exclusive process respectively.

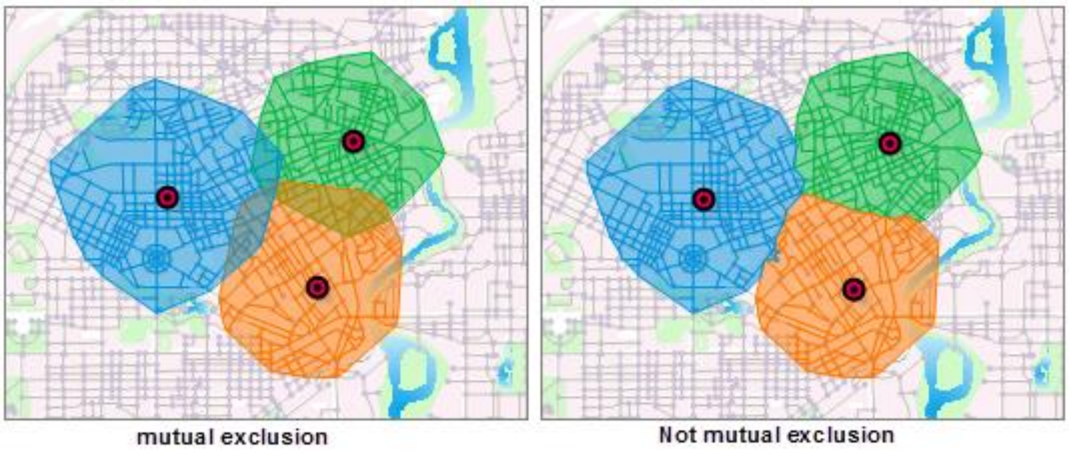


Figure 6-11 Exclusivity processing

Whether to start analyzing from the center. The default value is false. Starting analyzing from the center and not from the center, reflecting the relation pattern of the service center and the demand area in need of this service.

Starting analyzing from the center, reflecting a service center providing service to the service demanding area. For example, a milk station delivers milk to various settlements. To perform the service area analysis on this milk station for getting the service range must choose the mode of analyzing from the center point.

Whereas analyzing from a demand station is that a demand station initially gets services from the service center. For example, to perform the service area analysis on a school must choose the mode of analyzing from the demand stations, because student going to school is an initiative behavior.

The edge in network data has both from-to impedance and to-from impedance, which may not be the same. Therefore, allocating resources from supply center to demand node and allocating resources from demand node to supply center may have different analysis result.

6.11.1 Service Area Analysis

The findServiceArea method of the TransportationAnalyst class is used to implement service area analysis. Details are introduced below:

Syntax:

public ServiceAreaResult findServiceArea(TransportationAnalystParameter parameter,double[] weights,boolean isFromCenter,boolean isCenterMutuallyExclusive)

Parameters:

parameter: The specified transportation network analysis parameter.

weights: The radius array of the service area. The length of the array should be consistent with the number of the specified service center points. The radius has the same unit with the from-to weight field and the to-from weight field of the specified weight information.

isFromCenter: Whether to start analyzing from the center.

isCenterMutuallyExclusive:

Return value:

The TransportationAnalystResult object.

Supply centers are specified using the parameter of the TransportationAnalystParameter type. Supply centers can also be of node mode or coordinate point mode. As to node mode, node ID collection for supply centers can be specified through the setNodes method of the TransportationAnalystParameter object. As to coordinate point mode, coordinate point collection for supply centers can be specified through the setPoints method. The two modes are exclusive with each other. Please refer to section 6.6 for details.

Other information for service area analysis can also be specified in the parameter of the TransportationAnalystParameter type. Please refer to section 6.6 for details.

6.11.2 Service Area Results

The ServiceAreaResult class is inherited from the TransportationAnalystResult class. It inherits the getNodes, getEdges, getRoutes and getWeights methods from the TransportationAnalystResult class. Methods getNodes, getEdges, getRoutes and getWeightsare invalid for service area analysis. Moreover, two methods have been provided by ServiceAreResult to return supply center collection in the analysis results, as shown in Table 6.8.

Table 6-8 Methods for the SupplyCenter class

Type	Name	Description
GeoRegion[]	getServiceRegions	Returns the service area collection. The order of the array elements is identical to the specifying order of the center points.
int[]	getServiceRouteCounts	Gets the route count array of each service region in the ServiceAreaResult class. The order of the array element is identical to the specified order of the center points.

Please refer to section 6.6 for basic information on getNodes, getEdges and getWeights methods. Below is the introduction to the meaning of service area analysis results.

Passed edge collection (getEdges): The one-dimensional length of the array is the number of service centers, and the 2D elements are the ID of the edges (partially) covered by the service area.

Passed node collection (getNodes): The one-dimensional length of the array is the number of service centers, and the 2D elements are the ID of the nodes covered by the service area.

Route object collection (getRoutes): This array stores the specified order according to the centers and the routes (partially) covered by each service area. The elements' orders of the array which is obtained from the getServiceRouteCountsmethod of the service area analysis result is corresponding to the specified orders of the centers. The element's value is the number of the centers (partially) covered by the service area. You can know the corresponding routes of each service area with the array.

Weight array (getWeights): This array stores the total cost of each service area, that is, the weight of all routes covered by the service area, according to the specifying order of the supply centers.

6.12 Location-Allocation Analysis Contents

The location-allocation analysis is to determine the optimal location of one facility or multiple facilities, so that the facilities can provide the demand-side with services or goods with a most economical and effective method. The location-allocation not only is a location process, but also needs to allocate the demands of the demand-side to the corresponding service areas. Therefore, it is called location-allocation.

Concepts for location-allocation analysis:

Supply center: The center point that are the facilities which supply the resources and services. Corresponding to the network nodes, the related information of the resources supply centers includes the maximum impedance value, the resources supply centers' types, the nodes ID of resources supply centers in the network, etc.

Demand-side: It is generally the location that the supply center provides the services and resources for, which is also corresponding to the network node.

Maximum Impedance: It is used to restrict the cost from the demand point to the center. If the cost from the demand point, including edge and node, to the center is higher than the maximum impedance, the demand point is filtered out. The maximum impedance can be edited.

Supply center type: The types that the resources supply for the center points include non-center, fixed center and optional center.

1. These points refer to the service facilities (act as the resources supply roles) that already exist and have been built in the network.
2. The optional center points refer to the resources supply center that can build service facilities. It means the service facilities to be built will select the location among these optional center points.
3. These points will not be considered during the analyzing process. They may be not allowed to build these facilities or other facilities have been built on these points.

All the demand points used during the analysis process are the network nodes, which mean that all the network nodes as the resources demand points except the network nodes according to each types of the center points are involved in the analysis of the location-allocation. If eliminate part of the nodes , they can be set as barrier points.

Allocate from supply center: Allocating from centers indicates resources are transmitted from centers to demand points; not allocating from centers indicates demand sides go to get resources from centers.

1.The example of allocating from centers (from the supply to the demand) :

The electric energy is generated from the power station, and transmitted to clients by the electric network. Here, the power station is the center in the network model, because it can provide electricity supply. The electricity clients distribute along electric network lines (edges in the network model), and they produce "demands". In this situation, the resource is transmitted from the supplier to the demand-side by the network so as to implement the resource allocation.

2.The example of not allocating from centers (from the demand to the supply):

The relation between the school and the students also constitutes a kind of supply-and-demand allocation relation in the network. The school is the resource supplier, and it is responsible for providing school-age children with places to go to school. School-age children are demand-sides of the resource. They demand to enter the school. School-age children as demand-sides distribute along the street network, and they generate demands for the resource of the school as the supplier--student places.

Demand points are network nodes. Besides, parameters such as resource supply center and its type, maximum impedance, whether to allocate from supply center, etc. need to be specified by users. Location-

Allocation analysis methods and how to set parameters for location-allocation analysis will be introduced in the next section.

Now, we use an example to facilitate your understanding of location-allocation analysis. Currently, there are 3 primary schools in a region. It will establish 3 new primary schools in this region according to the demand. Selecting 9 locations as the optional locations, we will select 3 optimal locations for building the schools. As shown in the figure 6-12 the existed 3 schools are the fixed centers; 7 optional locations are the optional centers. The conditions to be met for building a new school: It takes the residents within the residential area no more than 30 minutes to walk to the school. The location-allocation analysis will give the optimal locations, and display the service area of each school including the existed 3 schools. As shown in figure 6-13, the No. 5, No. 6 and No. 8 optional centers are ultimately selected as the optimal locations to build the schools.

Note:

All the network nodes of the network datasets in the following two maps can be seen as the settlements and all of them can be involved in the location-allocation analysis, and the number of the residents in the settlements is the needed services number of these settlements.

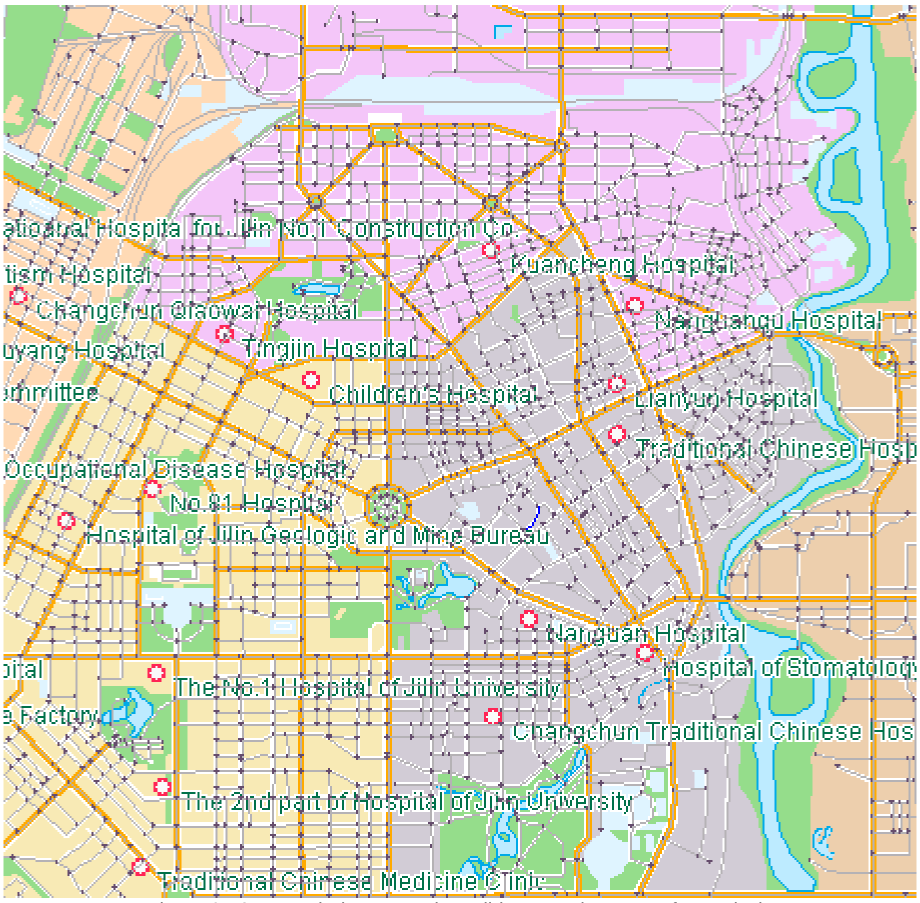

Kuancheng Hospital
Changchun Qiaowai Hospital
Tingjin Hospital
Children's Hospital
Nanguangu Hospital
Lianyun Hospital
Traditional Chinese Hosp
Occupational Disease Hospital
No.81 Hospital
Hospital of Jilin Geologic and Mine Bureau
Nanguan Hospital
Hospital of Stomatology
The No.1 Hospital of Jilin University
Changchun Traditional Chinese Hos
The 2nd part of Hospital of Jilin University
Traditional Chinese Medicine Clinic

Figure 6-12 Network dataset and candidate supply centers for analysis

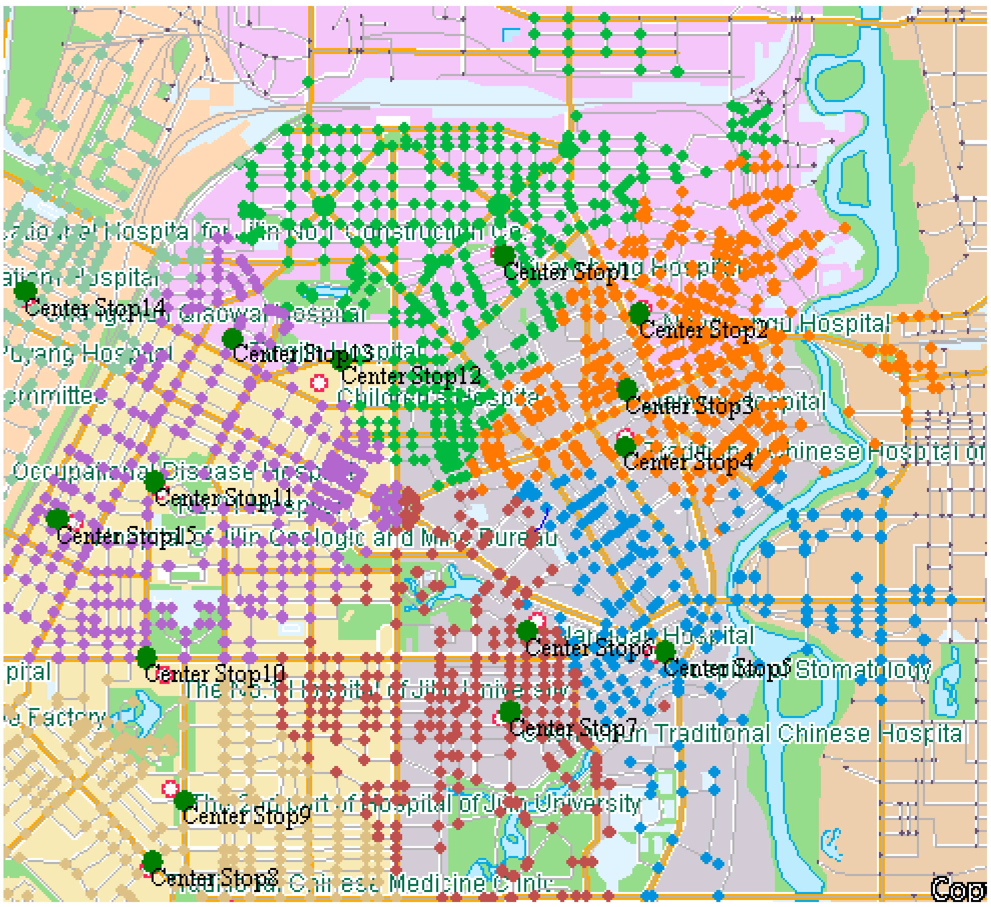


Figure 6-13 Location-allocation results

6.12.1 Location-Allocation Implementation

The findLocation method of the TransportationAnalyst class is used for location-allocation analysis.

Syntax:

public LocationAnalystResult findLocation(LocationAnalystParameter parameter)

Parameters:

parameter: The specified location-allocation parameter object.

Return value:

The LocationResult object.

This method needs an object of the LocationAnalystParameter type to be passed in as parameter. Through this object, the parameters for location-allocation analysis can be specified, including supply center collection, expected supply center count, whether to allocate from supply center, weight information, etc. Below is detailed introduced to methods of the LocationAnalystParameter calss, as shown in Table 6.9.

Table 6.9 Methods for the LocationAnalystParameter class

Type	Name	Description
SupplyCenters	get/setSupplyCenters	Returns or sets supply center collection. Please refer to Table 6.10 for details about SupplyCenter.

Type	Name	Description
int	get/setExpectedSupplyCenterCount	Returns or sets the number of the supply centers which will be used to build some establishments. If the parameter is set to zero, the minimal number of required supply centers will be computed to cover the whole analysis area.
boolean	is/setFromCenter	Returns or sets whether to analyze from the supply centers or not.
String	get/setWeightName	Returns or sets the name of the weight field information, i.e., the value of the setName method of a certain WeightFieldInfo object of the WeightFieldInfos object specified by the TransportationAnalystSetting class.
String	get/setTurnWeightField	Returns or sets the turn cost field and this field must be in the collection of turn cost fields (setTurnWeightFields) which is defined in the transportation network analysis environment settings.

Table 6.10 Methods for the SupplyCenter class

Type	Name	Description
int	get/setID	Returns or sets the ID of SupplyCenter.
double	get/setMaxWeight	Gets or sets the maximum cost (impedance) of the supply center. It has the same unit with the weight field of the WeightFieldInfo object that is specified by the setWeightName method of the LocationAnalystParameter object. The greater the impedance is, the bigger area the supply center can serve.
SupplyCenterType	get/setType	Gets or sets the type of the supply center.

6.12.2 Location-Allocation Results

Results for location-allocation analysis are stored in the LocationAnalystResult object, from where resource supply center array, demand center array can be acquired, as shown in Table 6.11.

Table 6.11 Methods for the LocationAnalystResult class

Type	Name	Description
SupplyResult[]	getSupplyResults	Returns the objects arrays of the resources supply results.
DemandResult[]	getDemandResults	Returns the objects arrays of the demand results.

Resource supply center array (getSupplyResults)

Resource supply result array (getSupplyResults) is a collection of the resource supply result object (getSupplyResult) and it contains all resource supply results of all resource supply centers. Each SupplyResult object corresponds to the result of one resource supply center, including ID of supply center, type, maximum impedance, number of demand centers, total cost and average cost, etc. Please refer to Table 6.12 for details.

Table 6.12 Methods for the SupplyResult class

Type	Name	Description
int	getID	Gets the ID of the supply center.
SupplyCenterType	getType	Gets the type of the supply center.
double	getMaxWeight	Gets the maximum cost (impedance) of the supply center. It has the same unit with the weight field of the WeightFieldInfo object that is specified by the WeightName property of the LocationAnalystParameter object.
double	getTotalWeights	Gets the total cost. It has the same unit with the weight field of the WeightFieldInfo object that is specified by the setWeightName method of the LocationAnalystParameter object. When the location-allocation analysis selects to allocate the resources from the supply center, the total cost is the sum of the cost from the supply center to all the demand nodes; conversely, when the resources are not be allocated from the supply center, the total cost is the sum of the cost from all the demand centers to the supply center.
double	getAverageWeight	Gets the average cost, i.e., the total cost is divided by the number of the demand nodes. It has the same unit with the weight field of the WeightFieldInfo object that is specified by the setWeightName method of the LocationAnalystParameter object.
int	getDemandCount	Gets the count of demand points that the supply center serves for.

Demand result array (getDemandResults)

The demand result array (getDemandResults) is a collection of demand result object (getDemandResult) and includes demand result information for all demand centers. Each DemandResult object corresponds to the demand result of one DemandResult object, including ID of the demand node, supply center that serves the demand node. Please refer to Table 6.13 for details.

Table 6.13 Methods for the DemandResult class

Type	Name	Description
int	getID	Gets the ID of the demand node.
int	getSupplyCenterID	Gets the ID of the supply center.

6.13 Other Functions Contents

Besides the transportation network analysis functions introduced above, the TransportationAnalyst also provides 3 auxiliary methods for network analysis, including cost matrix calculation, edge weight update and turn node weight update. The 3 functions are introduced below. Note that the 3 methods can only be called after loading network model.

6.13.1 Cost Matrix Calculation

Cost is the minimum cost from a node (coordinate point) to another node (coordinate point) in the network. Cost matrix is a matrix of cost for N nodes. The computeWeightMatrix method of the TransportationAnalyst class is used for calculating cost matrix.

Syntax:

public double[,] computeWeightMatrix(TransportationAnalystParameter parameter)

Parameters:

parameter: The specified transportation network analysis parameter object.

Return value:

The cost matrix.

Points for calculating cost matrix, which can be specified through the parameter of the TransportationAnalystParameter type. Users can specify node collection or coordinate point collection through the setNode or setPoints method of the TransportationAnalystParameter object. But note that the two methods are exclusive with each other. The last setting before analysis will be applied.

Except specifying point collection for calculating cost matrix, users can still other methods of the TransportationAnalystParameter object to specify other required parameters, network dataset and the topological relationship field, weight information, etc. More than that, setting barrier nodes and barrier edges will also be valid.

This method will return a cost matrix is a 2D array with the same rows and columns. The index of the elements are identical with the index of the specified points to be calculated. For example, The value of the element [1,2] is the least cost from No. 1 point in the points' index to the No. 2 point in the index. Note that the result includes the cost form a point to itself, and the value is 0.

6.13.2 Edge Weight Update

The updateEdgeWeight method from the TransportationAnalyst class is used for modifying edge weights in the network model.

Syntax:

public double updateEdgeWeight(int edgeID, int fromNodeID, int toNodeID, String weightName, double weight)

Parameters:

toNodeID: The ID of the refreshed edge.

fromNodeID: The start node ID of the refreshed edge.

toNodeID: The end node ID of the refreshed edge.

weightName: The name of the updated weight field. It is the value of the setName method of the WeightFieldInfo object of the WeightFieldInfos object in the TransportationAnalystSetting class.

weight: The weight is used to update the old value. It has the same unit with the weight field specified by the weightName property.

Return value:

Returns the weight value before updating if successful; otherwise returns Double.MIN_VALUE.

This method is used to modify the edge weight of the network model loaded in the memory, which doesn't modify the network model. Therefore, this method provides us with a method for modifying the edges temporarily, allowing us to save trouble in reloading network model after modifying network dataset or source edge weights.

This method can update the forward/reverse weight of the edge. The forward weight is the cost from the start node to the end node, whereas the reverse weight is the cost from the end node to the start node. Hence, updates the forward weight if specifying the fromNodeID is the start node ID and the toNodeID is the end node ID of the updated edge in the network dataset; whereas updates the reverse weight if specifying the fromNodeID is the end node ID and the toNodeID is the start node ID.

Note: The negative weight value denotes the edge is closed to traffic.

6.13.3 Turning Node Weight Update

The updateTurnNodeWeight method from the TransportationAnalyst class is used for modifying turn weights in the turn table of the network model.

Syntax:

public double updateTurnNodeWeight(int nodeID, int fromEdgeID, int toEdgeID, String turnWeightField, double weight)

Parameters:

fromEdgeID: The ID of the turning node to update.

fromEdgeID: The start edge ID of the turning node to update.

toEdgeID: The end edge ID of the turning node to update.

turnWeightField: The name of the turn weight field, i.e., a value for TurnWeightFields in TransportationAnalystSetting.

weight: The weight is used to update the old value. It has the same unit with the turn weight field specified by the turnWeightField property.

Return value:

Returns the weight value before updating if successful; otherwise returns Double.MIN_VALUE.

This method is used to modify the turn weight of the network model loaded in the memory, which doesn't modify the turn table. Therefore, this method provides us with a method for modifying the turn weight temporarily, allowing us to save trouble in reloading network model after modifying turn tables.

It may produce a variety of turns at a turning point. The direction of the turn is determined by the From Edge ID and To Edge ID of the turn node. This method can be used to specify the corresponding from edge ID and to edge ID, therefore, modifying the turn weight of the turning node. Note: The negative weight value denotes the edge is closed to traffic.

Please refer to Chapter 3 for details about turn table.

Facility Network Analysis

One type of network analysis. Facility network analysis is primarily used for connectivity analysis, tracing analysis, etc.