# 14 APPENDIX VI

This appendix presents details of the implementation of dynamic traffic assignment in AIMSUN.

## 14.1 USER-DEFINED LINK COST FUNCTIONS

The user-defined cost functions can be formulated in terms of the most common mathematical functions and operators (+ , -, *, /, ln, log, exp, etc.). Their terms, which can be defined as parameters, constants or variables, must correspond to the numerical attributes of any object in the model (links, sections, turnings, vehicle types, etc.), whose values might be either fixed (i.e. lengths, theoretical capacities, number of lanes, etc.) or subject to change during the simulation (i.e. link flows, average link speeds, average link travel times, etc.).

Two types of user-defined cost functions can be distinguished: basic cost functions (named *cost functions*) and cost functions that consider vehicle type (named *cost functions with vehicle type*).

Basic cost functions do not distinguish between vehicle types and therefore cannot make use of variables that have any vehicle type reference. The parameter for this type of function is only the link, expressed as two parameters: S (section reference of the link) and T (turning reference of the link).

Basic Cost Function Signature: ***FunctionName*** (**S**, **T**)→ *double*

where **S** is a section reference

**T** is a turning reference

Basic Cost Function Body: Defined as a numerical expression using the following items:

| Name | Description | Result |
|---|---|---|
| ***Basic Items and Mathematical Operators*** | | |
| **Numerical Constant** | | Numerical Exp. |
| **+** | Addition | Numerical Exp. |
| **-** | Subtraction | Numerical Exp. |
| ***** | Multiplication | Numerical Exp. |
| **/** | Division | Numerical Exp. |
| **^** | Power | Numerical Exp. |
| **Mathematical functions** | | |
| **ABS(***exp***)** | Absolute value of *exp* | Numerical Exp. |
| **SIN(***exp***)** | Sine of numerical expression *exp* | Numerical Exp. |
| **COS(***exp***)** | Cosine of numerical expression *exp* | Numerical Exp. |

| | | |
|---|---|---|
| **TAN(***exp***)** | Tangent of numerical expression *exp* | Numerical Exp. |
| **ASIN(***exp)* | Inverse Sine of numerical expression *exp* | Numerical Exp. |
| **ACOS(***exp)* | Inverse Cosine of numerical expression *exp* | Numerical Exp. |
| **ATAN(***exp***)** | Inverse Tangent of numerical expression *exp* | Numerical Exp. |
| **SINH(***exp)* | Hyperbolic Sine of numerical expression *exp* | Numerical Exp. |
| **COSH(***exp)* | Hyperbolic Cosine of numerical expression *exp* | Numerical Exp. |
| **TANH(***exp***)** | Hyperbolic Tangent of numerical expression *exp* | Numerical Exp. |
| **SQRT(***exp***)**: | Square root of numerical expression *exp* | Numerical Exp. |
| **EXP(***exp***)**: | Exponential of numerical expression *exp* | Numerical Exp. |
| **LOG(***exp)* | Base 10 logarithm of numerical expression *exp* | Numerical Exp. |
| **LN(***exp)* | Natural logarithm of numerical expression *exp* | Numerical Exp. |
| **MIN(**exp1**,** exp2**)** | Minimum between two numerical expressions *exp1* and *exp2* | Numerical Exp. |
| **MAX(**exp1**,** exp2**)** | Maximum between two numerical expressions *exp1* and *exp2* | Numerical Exp. |
| ***Logical Operators and Conditional Statements*** | | |
| *exp1* **==** *expr2* | Equal to (where *exp1* and *exp2* are numerical expressions). | Logical Exp. |
| *exp1* **<** *expr2* | Less than (where *exp1* and *exp2* are numerical expressions). | Logical Exp. |
| *exp1* **<=** *expr2* | Less than or equal to (where *exp1* and *exp2* are numerical expressions). | Logical Exp. |
| *exp1* **>** *expr2* | Greater than (where *exp1* and *exp2* are numerical expressions). | Logical Exp. |
| *exp1* **>=** *expr2* | Greater than or equal to (where *exp1* and *exp2* are numerical expressions). | Logical Exp. |
| *exp1* **&** *expr2* | Logical And (where *exp1* and *exp2* are logical expressions). | Logical Exp. |
| *exp1* **\|** *expr2* | Logical Or (where *exp1* and *exp2* are logical expressions). | Logical Exp. |
| **!***expr1* | Logical negation (where *exp1* is a logical expression). | Logical Exp. |
| **IF(***expr1***;***expr2***;** *expr3***)** | Conditional statement returns the numerical expression *expr2* if the logical expression *expr1* is true; otherwise, it returns the numerical expression expr3. | Numerical Exp. |

The basic cost function expression (related to one link, composed of section *s* and turning movement *t*) may contain the model attributes, link attributes, section attributes and turning attributes of the model. (All of these expressions contain references to section *S* and turning *T*, which the user cannot change).

| Name | Description |
|---|---|

| Network attributes | |
|---|---|
| **NetMaxSectCapacity()** | Returns the Maximum Section Capacity of all sections in the network |
| **NetMaxTurnCapacity()** | Returns the Maximum Link Capacity of all links in the network |
| **Section attributes** | |
| RAND(S) | Random number between 0 and 1 generated using the seed of Section *S* |
| **SectMaxSpeed(S)** | Maximum Speed of Section *S* in Km/h or mph |
| **SectCapacity(S)** | Theoretical Capacity of Section *S* |
| **SectLength(S)** | Length of Section *S* in meters or feet |
| **SectSlope(S)** | Slope percentage of Section *S* |
| **SectUserDefinedCost(S)** | User defined cost of Section *S* |
| **Turning attributes** | |
| **TurnMaxSpeed(S, T)** | Maximum Speed of Turning *T* in Km/h or mph |
| **TurnLength(S, T)** | Length of Turning *T* in meters or feet |
| | Link attributes |
| **TurnCapacity(S, T)** | Capacity Link defined by Section *S* and turning *T* |

The basic cost function expression can also contain link statistical and section statistical data (aggregating all vehicle types). "Mean" refers to a temporal mean during a time period defined by the route choice cycle and the number of intervals that it considers (*cycle* times *number of intervals*) (AIMSUN 2002).

| Name | Description |
|---|---|
| **Section Statistical Data** | |
| **SectStaFlow(S)** | Mean Statistical Flow of Section *S* in veh/h |
| **SectStaDensity(S)** | Mean Statistical Density of Section *S* in veh/Km or veh/miles |
| **SectStaSpeed(S)** | Mean Statistical Speed of Section *S* in Km/h or mph |
| **SectStaTravelT (S)** | Mean Statistical Travel Time of Section *S* in seconds |
| **SectStaDelayT (S)** | Mean Statistical Delay Time of Section *S* in seconds |
| **SectStaStopT (S)** | Mean Statistical Stop Time of Section *S* in seconds |
| **SectStaNbStopsVeh (S)** | Mean Statistical Number of Stops in Section *S* |
| **SectStaQueueLMean (S)** | Mean Queue Length per Lane in Section *S*. |
| **SectStaQueueLMaxn (S)** | Maximum Queue Length per Lane in Section *S*. |
| **Link Statistical Data** | |
| TURNSTAFLOW(S, T) | Mean Statistical Flow of Link composed of section *S* and turning *T* in veh/h |
| **TurnStaSpeed(S, T)** | Mean Statistical Speed of Link composed of section *S* and turning *T* in Km/h or mph. |
| **TurnStaTravelT (S, T)** | Mean Statistical Travel Time of Link composed of section *S* and turning *T* in seconds. The function returns the statistical travel time as long as at least one vehicle has driven along the whole section. If the section has remained empty for the whole period, it returns the travel time in free flow conditions. If no vehicles have crossed the section because all the vehicles in the section have remained at a standstill, it returns the delay time of the first vehicles in the section. |
| **TurnStaDelayT (S, T)** | Mean Statistical Delay Time of Link composed of section *S* and turning |

| | |
|---|---|
| | *T* in seconds. |
| **TurnStaStopT (S, T)** | Mean Statistical Stop Time of Link composed of section *S* and turning *T* in seconds. |
| **TurnStaNbStopsVeh (S, T)** | Mean Statistical Number of Stops of Link composed of section *S* and turning *T.* |
| **TurnStaQueueLMean (S, T)** | Mean Queue Length per Lane in Link composed of section *S* and turning *T*. |
| **SectStaQueueLMaxn (S, T)** | Maximum Queue Length per Lane in Link composed of section *S* and turning *T*. |

The basic cost function expression may also contain the initial and past (or historical) link costs. This is only possible if the basic cost function corresponds to the expression of a dynamic cost function.

| Name | Description |
|---|---|
| **Link Cost** | |
| **TurnPastCost(S, T,** *offset*) | Past Cost in link composed of section *S* and turning *T*. Past costs are link costs stored in previous simulations. If parameter *offset* is 0, then it takes the past costs stored at the same time. Therefore, for a positive *offset* it takes the next past costs stored and for a negative value it takes the previous past costs with respect to the current time. |
| **TurnCostInitial (S, T)** | Initial Cost in link composed of section *S* and turning *T*, when vehicle types are not considered. In case the link has an initial cost function per vehicle type, then it returns a –1. When the initial cost is not available, it returns 0. |

Examples of user-defined basic cost functions:

- Example *InitialCost1 (S, T)* corresponds to the expression of the default initial cost function, when no distinction is made between different vehicle types:

**SectLength(S) /(SectMaxSpeed(S) /3.6))+(TurnLength(S, T) / (TurnMaxSpeed(S, T) /3.6)**

- Example *InitialCost2 (S, T)* corresponds to the expression of the default initial cost function, when no distinction is made between different vehicle types and the capacity weight is 2:

**TurnStaTravelT(S,T)+TurnStaTravelT(S,T)*2.0***

**(1-TurnCapacity(S,T)/NetMaxTurnCapacity()))**

- Example *DynamicCost1 (S, T)* of the default dynamic cost function, when no distinction is made between different vehicle types:

**Max(TurnStaTravelT(S,T),TurnCostInitial(S,T))**

- Example *DynamicCost2 (S, T)* of the default dynamic cost function, when no distinction is made between different vehicle types and the capacity weight is 2:

**Max(TurnStaTravelT(S,T)+TurnStaTravelT(S,T)*2.0***

**(1-(TurnCapacity(S,T)/NetMaxSectCapacity())),TurnCostInitial(S,T))**

- Example *DynamicCost3 (S, T)* of the dynamic cost function, when no distinction is made between different vehicle types. It represents a general cost $general\cos t = \alpha * TravelTime + \beta * Dis\tan ce + \gamma * MonetaryCost$ , where the weights α, β, and γ are 0.5, 0.25 and 0.25 respectively.

**0.5*TurnStaTravelT(S,T) + 0.25* TurnLength(S, T) + 0.25*SectUserDefinedCost(S)**

- Example *DynamicCost4 (S, T)* of the dynamic cost function, when no distinction is made between different vehicle types. It represents the observed link travel time smoothed by the past cost from the previous simulation (learning mechanism):

**0.50 * TurnStaTravelT(S,T) + 0.25 * TurnPastCost(S, T, 0) + 0.25 * TurnPastCost(S, T, 1)**

Cost functions that consider vehicle type can distinguish between vehicle types and consequently can make use of variables that have vehicle type references. The parameter for this type of function is the link, expressed as two parameters: *S* (section reference of the link), *T* (turning reference of the link) and the vehicle type *VT* reference.

Signature: ***FunctionName*** (**S**, **T, VT**)→ *double*

where **S** is a section reference

**T** is a turning reference

**VT** is a vehicle type reference

Body: defined as a numerical expression using the same basic items, mathematical operators, mathematical functions, logical operators and conditional statements used in the expression of a basic cost function.

The expression for the cost function that considers vehicle type (related to one link, composed of section *s* and turning movement *t*) can contain the same model attributes, link attributes, section attributes and turning attributes as the basic cost function, but it can contain vehicle type attributes. (All these expressions contain references to section *S* and turning *T* and vehicle type *VT*, which the user cannot change).

| Name | Description |
|---|---|
| **Vehicle Type attributes** | |
| **VehTypeLength(VT)** | Mean Length of Vehicle Type *VT* in meters or feet |
| **VehTypeWidth(VT)** | Mean Width of Vehicle Type *VT* in meters or feet |

| | |
|---|---|
| VEHTYPEMAXSPEED(VT) | Mean Maximum Desired Speed of Vehicle Type *VT* in Km/h or mph |
| **VehTypeDecel(VT)** | Mean Maximum Acceleration of Vehicle Type *VT* in $m/s^2$ or $ft/s^2$ |
| **VehTypeMaxDecel(VT)** | Mean Normal Deceleration of Vehicle Type *VT* in $m/s^2$ or $ft/s^2$ |
| **VehTypeSpeedAc(VT)** | Mean Speed Acceptance of Vehicle Type *VT* |
| **VehTypeGiveWayTime(VT)** | Mean Give Way Time of Vehicle Type *VT* |
| **VehTypeGuidanceAc (VT)**: | Mean Guidance Acceptance of Vehicle Type *VT* |
| **CheckVehType(VT,** *string***)** | Returns 1 if *VT* belongs to vehicle type with *string* as name; otherwise, it returns 0. Example CheckVehType (VT, car) |

In addition to statistical section and link data aggregating all vehicle types, the cost function that considers the vehicle type expression,  (see basic cost functions) may also contain link statistical and section statistical data (that distinguish vehicle types).

| Name | Description |
|---|---|
| **Section Statistical Data** | |
| **SectStaFlow(S, VT)** | Mean Statistical Flow of Section *S* of Vehicle Type *VT* in veh/h |
| **SectStaDensity(S, VT)** | Mean Statistical Density of Section *S* of Vehicle Type *VT* in veh/Km or veh/miles |
| **SectStaSpeed(S, VT)** | Mean Statistical Speed of Section *S* of Vehicle Type *VT* in Km/h or mph |
| **SectStaTravelT (S, VT)** | Mean Statistical Travel Time of Section *S* of Vehicle Type *VT* in seconds |
| **SectStaDelayT (S, VT)** | Mean Statistical Delay Time of Section *S* of Vehicle Type *VT* in seconds |
| **SectStaStopT (S, VT)** | Mean Statistical Stop Time of Section *S* of Vehicle Type *VT* in seconds |
| **SectStaNbStopsVeh (S, VT)** | Mean Statistical Number of Stops in Section *S* of Vehicle Type *VT* |
| **SectStaQueueLMean (S, VT)** | Mean Queue Length per Lane in Section *S* of Vehicle Type *VT* |
| **SectStaQueueLMaxn (S, VT)** | Maximum Queue Length per Lane in Section *S* of Vehicle Type *VT* |
| **Link Statistical Data** | |
| TURNSTAFLOW(S, T, VT) | Mean Statistical Flow of Link composed of section *S* and turning *T* of Vehicle Type *VT* in veh/h |
| **TurnStaSpeed(S, T, VT)** | Mean Statistical Speed of Link composed of section *S* and turning *T* of vehicle type *VT* in Km/h or mph |
| **TurnStaTravelT (S, T, VT)** | Mean Statistical Travel Time of Link composed of section *S* and turning *T* of Vehicle Type *VT* in seconds. The function returns the statistical travel time as long as at least one vehicle has driven along the whole section. If the section has remained empty for the whole period it returns the travel time in free flow conditions. If no vehicles have driven along the section because all vehicles on the section have remained at a standstill, it returns the delay time of the first vehicles in the section. |
| **TurnStaDelayT (S, T, VT)** | Mean Statistical Delay Time of Link composed of section *S* and turning *T* of Vehicle Type *VT* in seconds |
| **TurnStaStopT (S, T, VT)** | Mean Statistical Stop Time of Link composed of section *S* and turning *T* of Vehicle Type *VT* in seconds |
| **TurnStaNbStopsVeh (S, T, VT)** | Mean Statistical Number of Stops of Link composed of section *S* and turning *T* of Vehicle Type *VT* |
| **TurnStaQueueLMean (S, T, VT)** | Mean Queue Length per Lane in Link composed of section *S* and turning *T* of Vehicle Type *VT* |

| | |
|---|---|
| **SectStaQueueLMaxn (S, T, VT)** | Maximum Queue Length per Lane in Link composed of section *S* and turning *T* of Vehicle Type *VT* |

The expression for the cost function that considers vehicle type may also contain the initial and past (or historical) link costs. This is only possible if the basic cost function corresponds to the expression of a dynamic cost function.

| Name | Description |
|---|---|
| **Link Cost** | |
| **TurnPastCost(S, T, VT,** *offset*) | Past Cost in link composed of section *S* and turning *T* of vehicle type *VT*. Past costs are link costs stored in previous simulations. If parameter *offset* is 0, then it takes the past costs stored at the same time. Therefore, for a positive *offset* it takes the next past costs stored and for a negative value, it takes the previous past costs with respect to the current time. |
| **TurnCostInitial (S, T, VT)** | Initial Cost in link composed of section *S* and turning *T* of vehicle type *VT*. In the case the link has a initial cost function not considering vehicle types, then gives the common cost. |

Examples of user-defined basic cost functions:

- Example *InitialCost3* of the default initial cost function that distinguishes vehicle types:

**SectLength(S)/(min(SectMaxSpeed(S)*VehTypeSpeedAc(VT),VehTypeMaxSpeed(VT))/3.6 +TurnLength(S,T)/min(TurnMaxSpeed(S,T)*VehTypeSpeedAc(VT),VehTypeMaxSpeed(VT) )/3.6**

- Example *InitialCost4* of the default initial cost function that distinguishes vehicle types and takes the capacity weight as 2:

**TurnStaTravelT(S,T,VT)+TurnStaTravelT(S,T,VT)*2.0* (1-TurnCapacity(S,T)/NetMaxTurnCapacity())**

- Example *DynamicCost5* of the default dynamic cost function:

**Max(TurnStaTravelT(S,T,VT),TurnCostInitial(S,T,VT))**

- Example *DynamicCost6* of the default dynamic cost function, taking the capacity weight as 2:

**Max(TurnStaTravelT(S,T,VT)+TurnStaTravelT(S,T,VT)* 2.0*(1-(TurnCapacity(S,T)/NetMaxSectCapacity())),TurnCostInitial(S,T,VT))**

- Example *DynamicCost7* of the dynamic cost function taking the section attribute user-defined cost and using a conditional statement to distinguish a different user-defined cost for lorries:

**IF(CheckVehType(VT,truck)>0;TurnStaTravelTVT(S,T,VT)+ SectUserDefinedCost (S); TurnStaTravelTVT(S,T,VT)+ SectUserDefinedCost (S)+10)**

- Example *DynamicCost8 (S, T)* of the dynamic cost function, which represents the observed link travel time smoothed by the past cost from the previous simulation (learning mechanism):

**0.50 * TurnStaTravelT(S,T,VT) + 0.25 * TurnPastCost(S, T,VT, 0) + 0.25 * TurnPastCost(S, T, VT, 1)**

- Example *DynamicCost9 (S, T)* of the dynamic cost function, which represents the observed link travel time smoothed by the past cost from the previous simulation (learning mechanism) if the vehicle type's guidance acceptance is lower than 1; otherwise, it directly takes the observed link travel time:

**IF(VehTypeGuidanceAc (VT)>=1.0; TurnStaTravelT(S,T,VT) ;**
**0.75 * TurnStaTravelT(S,T,VT) + 0.25 * TurnPastCost(S, T,VT, 0))**

## *14.2 USER-DEFINED ROUTE CHOICE MODELS*

As an alternative to the default route choice models, the user can define his or her own route choice models using the function editor.

The user-defined route choice model has to give the probability for each one of *n* alternative paths to be chosen by vehicles, taking into account the vehicle type.

User-defined route choice model signature: ***FunctionName*** **(O, D, TR, S, N, F, I, VT)**→ *double*

where **O** is an origin centroid reference

**D** is a destination centroid reference

**TR** is a set of path references

**S** is a section reference

**N** is the number of alternatives to consider

**F** is the index of the first alternative path to consider

**I** is the index of the path that is used to evaluate the probability (from 0 to N-1).

**VT** is a vehicle type reference

Function Body: defined as a numerical expression using the same basic items, mathematical operators, mathematical functions, logical operators, conditional statements and check vehicle type used in the expression of a user-defined link cost function (described above)

The route choice model may contain expressions related to aggregation:

| Name | Description |
|---|---|
| **Aggregation** | |
| **SUM(J,** *ini, end, exp***)** | Total sum of numerical expression *exp*, changing the parameter *J* from the integer *ini* to the integer *end*. An equivalent algebraic formula is $\sum_{J=ini}^{end} expr$ . |
| **PROD(J,** *ini, end, exp***)** | Total product of numerical expression *exp*, changing the parameter *J* from the integer *ini* to the integer *end*. An equivalent algebraic formula is $\prod_{J=ini}^{end} expr$ |

The route choice model may contain expressions related to path attributes:

| Name | Description |
|---|---|
| **Path attributes** | |
| **SPCurrCost(O, TR, S , K, VT)** | Current Cost of Path *K* of Vehicle Type *VT* from Section *S*. The value of *K* has to be defined between F and F+N-1, where F and N are parameters of the route choice functions. |
| **SPCurrTravelTime(O, TR, S , K, VT):** | Current travel time of Path *K* of Vehicle Type *VT* from Section *S*. The value of *K* has to be defined between F and F+N-1, where F and N are parameters of the route choice functions. |
| **SPDist(O, TR, S , K, VT)** | Physical Length of Path *K* of Vehicle Type *VT* from Section *S*. The value of *K* has to be defined between F and F+N-1, where F and N are parameters of the route choice functions. |
| **SPExpTravelTime(O, TR, S, K, VT)** | Experimented travel time of Path *K* of Vehicle Type *VT* from Section *S*. The value of *K* has to be defined between F and F+N-1, where F and N are parameters of the route choice functions. |
| **SPCommonDist(O, TR, S , K, L, VT)** | Common Physical Length between Path *K* and Path *L,* considering Vehicle Type *VT* from Section *S*. The values of *K* and *L* have to be defined between F and F+N-1, where F and N are parameters of the route choice functions. |
| **SPCommonCost(O, TR, S , K, L, VT)** | Common Cost between Path *K* and Path *L* considering Vehicle Type *VT* from Section *S*. The values of *K* and *L* have to be defined between F and F+N-1, where F and N are parameters of the route choice functions. |
| **SPPastCost (O, TR, S , K, VT, T)** | Past Cost of Path *K* of Vehicle Type *VT* from Section *S* at interval *T* (if equal to 0, it is the past cost of the current interval; if equal to 1, it is the past cost of the next interval; if equal to – 1, it is the past cost of the previous interval; and so on). The value of *K* has to be defined between F and F+N-1, where F and N are parameters of the route choice functions. |

Example of user-defined route choice models:

- Example *RC1* of the logit model that has a scale factor of 60:

**1/SUM(J,0,N-1,EXP(-60*(SPCurrCost(TR,S,F+J,VT)/3600.0-**
**SPCurrCost(TR,S,F+I,VT)/3600.0)))**

- Example *RC2,* in which the cars follow a logit model that has a scale factor of 60 and the other vehicles follow a uniform distribution:

**IF(CheckVehType(VT,car)>0;**
**1/SUM(J,0,N-1,EXP(-60*(SPCurrCost(O,TR,S,F+J,VT)/3600.0-**
**SPCurrCost(O,TR,S,F+I,VT)/3600.0)))**
**;1/N)**

- Example *RC3* of the logit model that has a scale factor of 60 and uses the past cost of the current interval:

**1/SUM(J,0,N-1,EXP(-60*(SPPastCost(O,TR,S,F+J,VT,0)/3600.0-**
**SPPastCost(O,TR,S,F+I,VT,0)/3600.0)))**

- Example *RC4* of the logit model that has a scale factor of 60, using a combination of the current cost and the past cost of the current and next interval:

**1/SUM(J,0,N-1,EXP(-**
**60*((0.5*SPCurrCost(O,TR,S,F+J,VT)+0.25*SPPastCost(O,TR,S,F+J,VT,0)+0.25*SPPastCos**
**t(O,TR,S,F+J,VT,1))/3600.0-**
**(0.5*SPCurrCost(O,TR,S,F+I,VT)+0.25*SPPastCost(O,TR,S,F+I,VT,0)+0.25*SPPastCost(O,T**
**R,S,F+I,VT,1))/3600.0)))**

## *14.3 PATH ANALYSIS TOOL*

### 14.3.1 PATH ANALYSIS

To gain insight into what occurs in a heuristic dynamic assignment, with the aim of properly calibrating and validating the simulation model, the user should have access to the analysis of the routes used. The path information that is available is as follows:

- Shortest path information. The user can view all the shortest path information that is being used by vehicles during the simulation.
- User-defined path information. The user can view all the user-defined path information.
- Shortest path display. The user can simultaneously view different shortest paths and their links in the network.
- Initial path assignment. The user can view all the probabilities considered when a vehicle enters the system.

- Dynamic path assignment. The user can view all the probabilities considered when a vehicle carries out path reassignment during the trip.

### 14.3.1.1 SHORTEST PATH INFORMATION

The shortest path information allows the links of which the path is composed to be viewed and it provides the following information:

- The cost in time (in seconds) from each of the links in the path to the destination centroid. This can be calculated as either the sum of the initial or dynamic cost of all the links of which the path is composed.

- The travel time (in seconds) from each of the links in the path to the destination centroid. This is equal to the cost only if the capacity weight parameter is set to zero.

- The distance (in metres) from each of the links in the path to the destination centroid.

In the example shown in Figure 14.1, 53 vehicles are using the tree of paths calculated at time 08:12:00 (3 buses and 50 cars), 11 vehicles are using the one calculated at time 08:16:00, and so on. In the example, the shortest path from section 51 to centroid 1 goes through sections 51, 35-37, 38- 40, etc. (displayed in the network using a different colour). The cost of the whole path is 173.3 seconds, the travel time is 117.7 seconds and the distance is 929.5 metres.

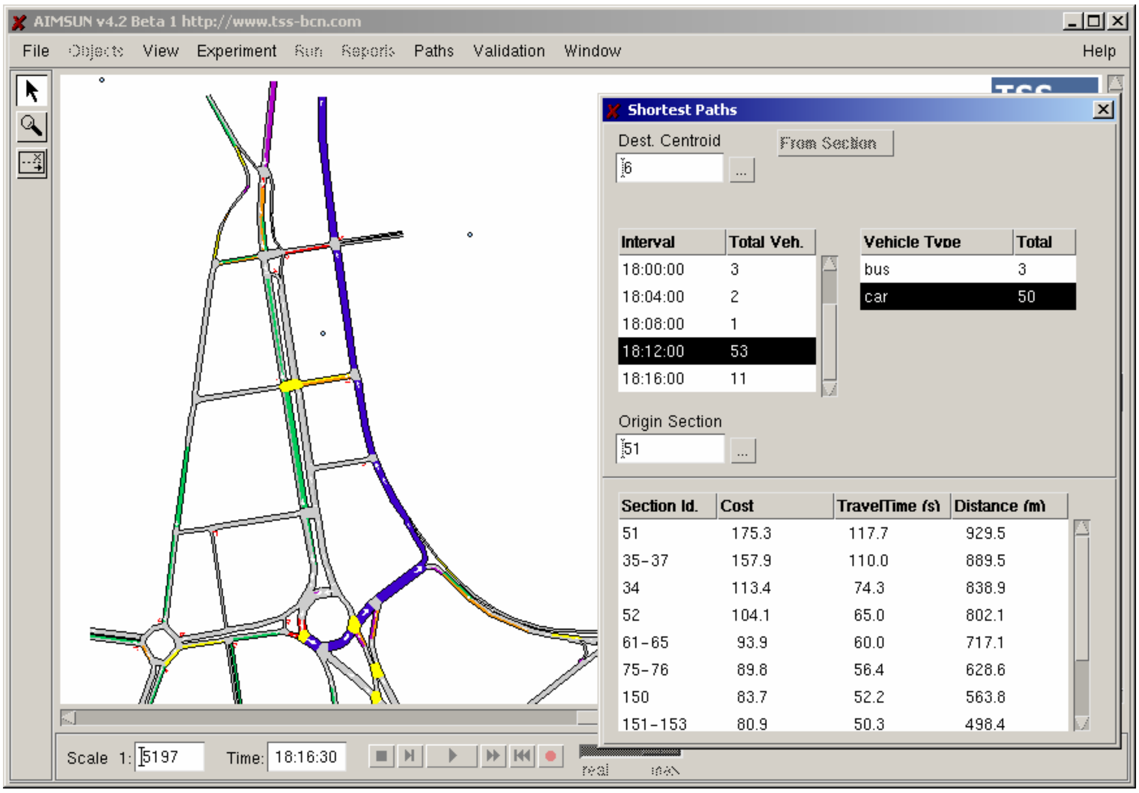


Figure 14.1. Shortest path information

### 14.3.1.2 USER-DEFINED PATH INFORMATION

The user-defined path information provides information about the user-defined paths of one OD pair (from one origin to one destination). The information displayed is the following:

- The cost in time (in seconds) from each of the links in the path to the destination centroid. This can be calculated as either the sum of the initial or dynamic cost of all the links of which the path is composed.
- The travel time (in seconds) from each of the links in the path to the destination centroid. This is equal to the cost only if the capacity weight parameter is set to zero.
- The distance (in metres) from each of the links in the path to the destination centroid.

In the example shown in Figure 14.2, from Origin 1 to Destination 6, there are two user-defined paths: *UserDefPath1* and *UserDefPath2.* The user-defined path *UserDefPath1* has a total cost of 283.8 seconds, a travel time of 170.7 seconds and a distance of 1080.2 metres.

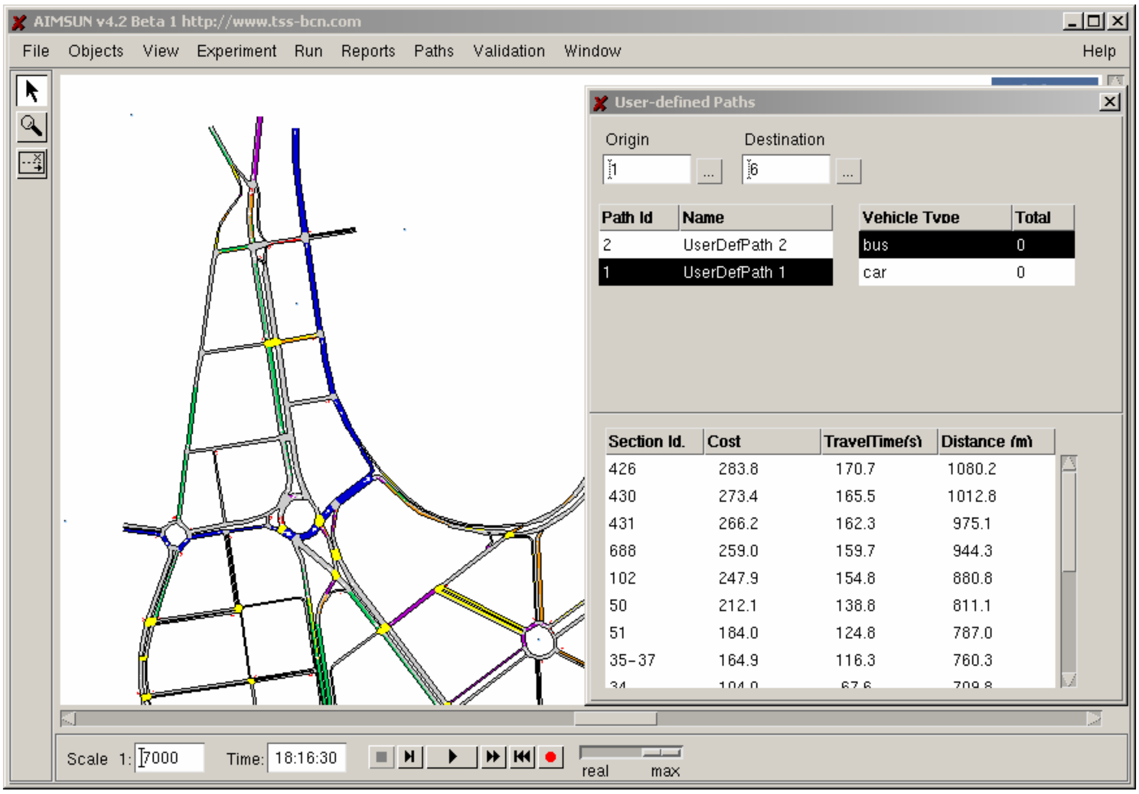


Figure 14.2. User-defined path information

### 14.3.1.3 SHORTEST PATH DISPLAY

The shortest path display enables different shortest paths calculated at different time intervals to be displayed simultaneously. This output enables the evolution of the shortest paths calculated in different time intervals to be known. The paths displayed could show any of the following:

- One destination centroid from one origin section
- One destination centroid from one origin centroid
- One destination centroid from all origin centroid

Figure 14.3 shows the shortest path calculated at time 18:16:00, from all origin centroids to destination centroid 6, where all the links are shown in red (or the colour displayed in the window dialog), except the links shared by different paths, which are shown in black. Figure 14.4 shows (all the links are red) the shortest path from origin centroid 22 to destination centroid 1, calculated 4 minutes after the beginning of the simulation during the warm-up period and the shortest path calculated at 18:16:00 (all the links are green).

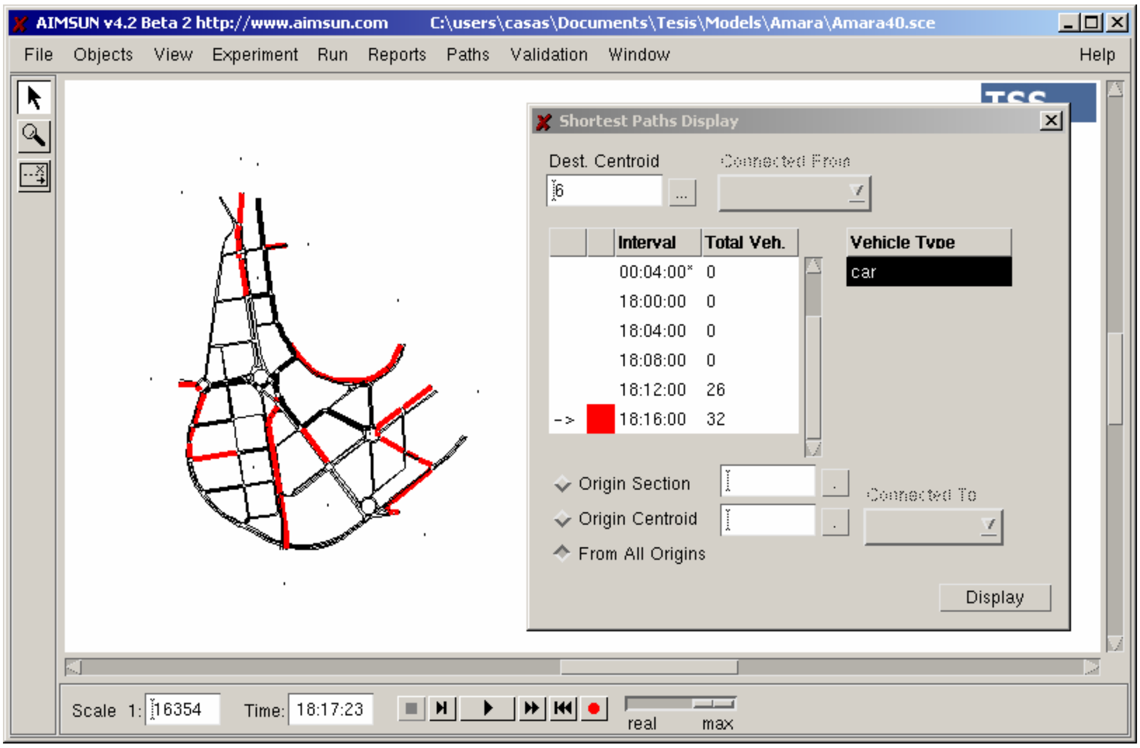


Figure 14.3. Shortest path display from all origins to one destination

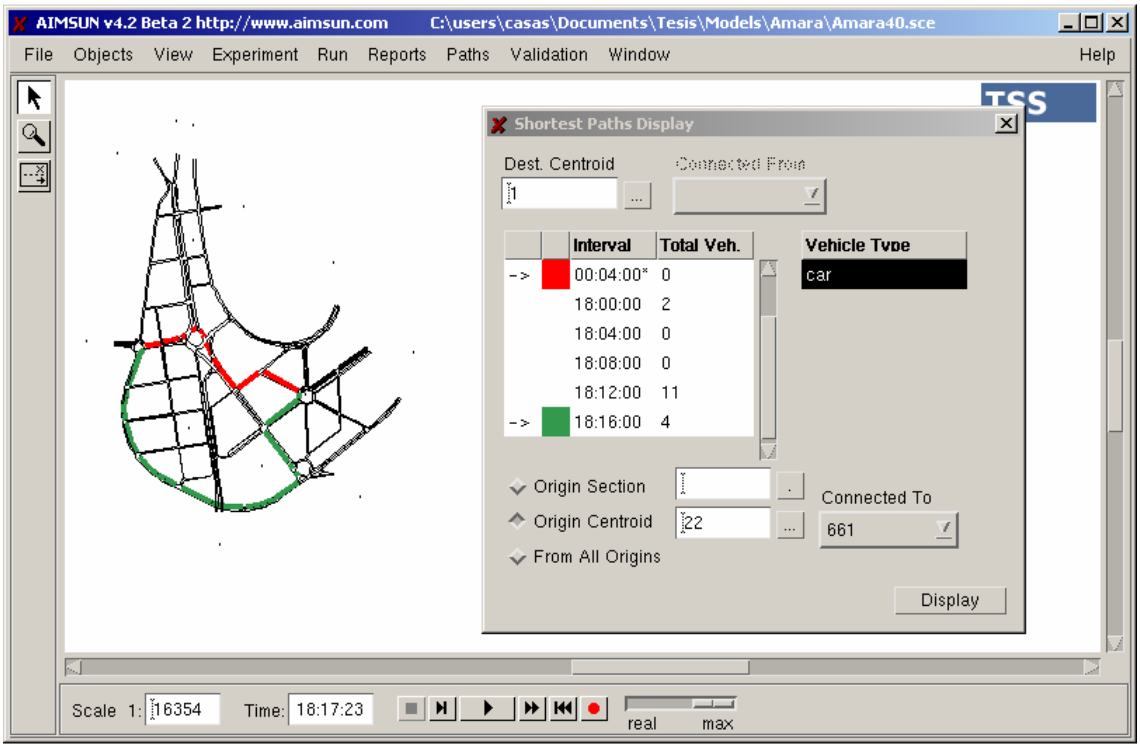


Figure 14.4. Shortest path display from one origin to one destination

### 14.3.1.4 Initial Path Assignment Information

When a vehicle decides to enter the system, it has to decide which path it will follow during its trip. This decision is based on the probabilities in the user-defined assignment and the probabilities calculated using the route choice model, as explained in Section 3.2.2.4.

Initial path assignment is always accessed by origin and destination centroid and it shows a list of user-defined paths. The identifiers and usage probability per vehicle type and the probability of use for all the shortest paths per vehicle type, per entrance section and per exit section, which have been calculated considering the route choice model, are displayed in the estimated assignment section.

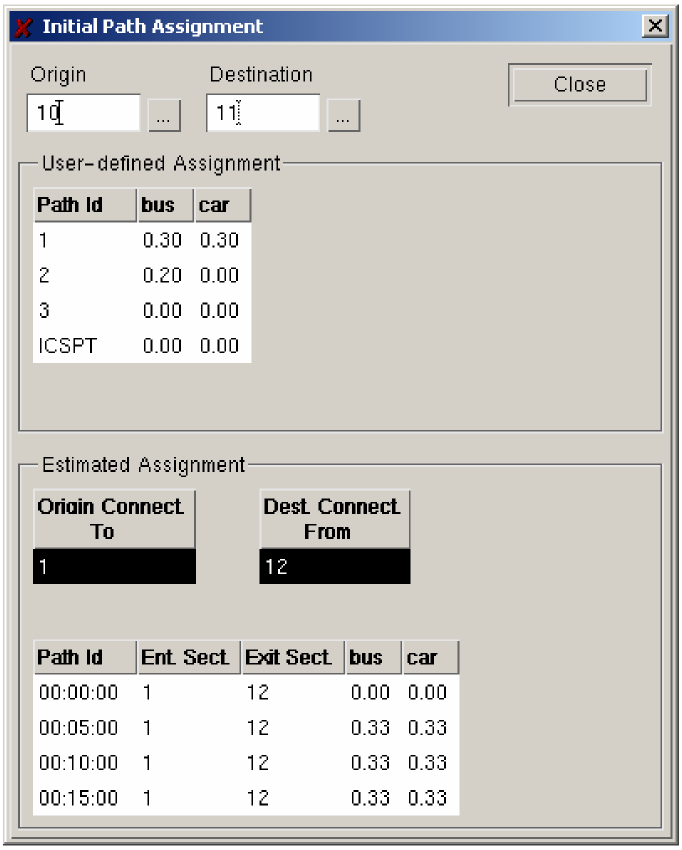


Figure 14.5. Initial assignment information

In the example shown in Figure 14.5, the initial path assignment for an 'unguided' bus that goes from centroid 10 to centroid 11 has a probability of 0.3 of choosing user-defined path 1, 0.2 of choosing user-defined path 2 and 0 of choosing user-defined path 3 and the initial shortest path (ICSP). 50% of buses will follow either user-defined path 1 or 2 (user-defined assignment) and the other 50% will consider the probabilities calculated using the route choice model (estimated assignment). Continuing with the example, one bus has a probability of 0.33 of taking the path calculated at 00:05:00, 0.33 of taking the path calculated at 00:10:00, and so on. The initial assignment for a 'guided' bus considers only the probabilities calculated using the route choice model.

#### 14.3.1.5 En-Route Path Assignment Information

When a vehicle decides to reassign its path during the trip, the decision is taken using the probabilities calculated by the route choice model, as explained in Section 3.2.2.5.

En-Route path assignment information shows the probabilities assigned to each alternative path.

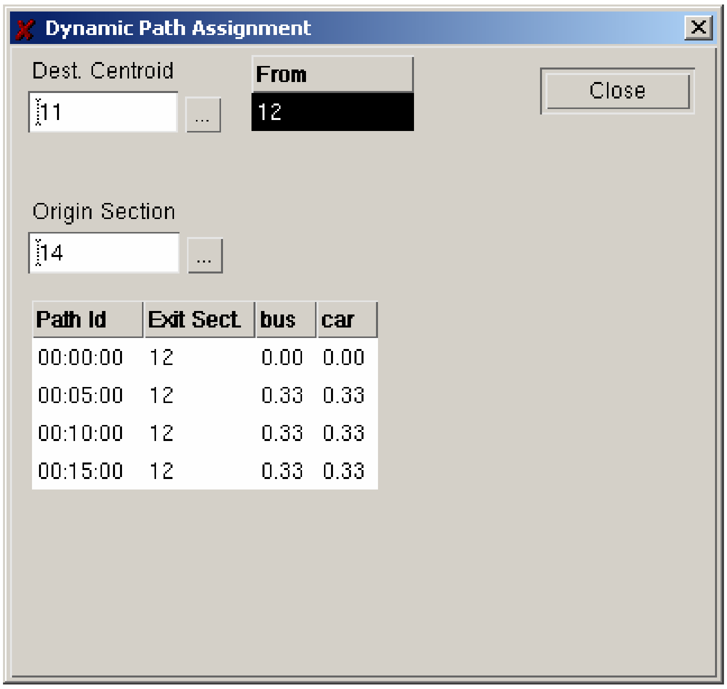


Figure 14.6. En-route path assigment information

In the example shown in Figure 14.6, the en-route path assignment for a bus that goes to centroid 11 and is located in section 14 has a probability of 0.33 of taking the path calculated at 00:05:00, 0.33 of taking the path calculated at 00:10:00, and so on.

## 14.3.2 SIMULATION OUTPUT

The simulation output for validating the dynamic traffic assignment is as follows: display user-defined paths and shortest path calculated and display initial and en-route path assignment. These capabilities are complemented by generating the path information output, generating link cost information and colouring all the vehicles per origin, per destination and both (per origin and destination at the same time).

### 14.3.2.1 PATH INFORMATION OUTPUT

For each shortest path calculated, the path definition (the identifier, calculation time, all links of which it is composed, etc.) and the path statistics (the number of vehicles that have arrived and the average travel time) are the output. When the output is based on ASCII files, the path definition is stored in a file named ‘PathDefinition.PAT’ and the statistical data for each statistical interval is stored in text files named ‘HHhMMmSS.PAT’, where HH:MM:SS corresponds to the simulation time when the report was produced. When the output is stored in a database, the tables that are filled in are ‘PathDef’, ‘PathSect’ and ‘PatSta’. The attributes of each table are as follows:

- **PathDef:** It contains the definition of simulated paths.

*Attributes*:

| Attribute Name | Type | Size | Description |
|---|---|---|---|
| rid | integer | | Replication identifier |
| id | integer | | Path identifier |
| origin | integer | | Origin Centroid identifier |
| dest | integer | | Destination Centroid identifier |
| nbsect | integer | | Total number of sections |
| created | float | | Path Creation Time (seconds ) |
| distance | float | | Path Distance (metres or feet) |

- **PathSect:** It contains all section identifiers that belong to each path.

*Attributes*:

| Attribute Name | Type | Size | Description |
|---|---|---|---|
| rid | integer | | Replication identifier |
| id | integer | | Path identifier |
| sect_pos | integer | | Position of the section |
| sect_id | integer | | Identifier of the section in position sect_pos |

- **PathSta:** It contains the statistics for each path.

*Attributes*:

| Attribute Name | Type | Size | Description |
|---|---|---|---|
| rid | integer | | Replication identifier |
| id | integer | | Path identifier |
| tfrom | integer | | Initial time of the period |
| tto | integer | | Final time of the period |
| ctype | integer | | Identifier of vehicle type.<br>= 0, aggregation of all vehicle type.<br>>0, a specific vehicle type. |
| nbveh | integer | | Number of Vehicles |
| ttime1 | integer | | Mean Travel Time (seconds) |

### 14.3.2.2 LINK COST OUTPUT

For each link and time interval used in the dynamic assignment, the costs are stored in a database, in a table named PastCost. The table is so called because all the link costs can be used as the concept of past costs in a learning mechanism. The table definition is as follows:

| Attribute Name | Type | Size | Description |
|---|---|---|---|
| rid | integer | | Replication identifier |
| fromsect | integer | | Origin Section of the link |
| tosect | integer | | Destination Section of the link |
| tfrom | integer | | Initial time of the period |
| tto | integer | | Final time of the period |
| ctype | integer | | Identifier of vehicle type.<br>= 0, aggregation of all vehicle type.<br>>0, a specific vehicle type. |
| cost | float | | Cost of the link |

### 14.3.2.3 Vehicle Information

During the simulation, the information for each individual vehicle can be accessed; it helps the user to identify his or her origin, destination and the path assigned. In the example shown in Figure 14.7, vehicle 2992 goes from origin centroid 22 to destination centroid 2 following the path calculated at 18:04 and the exit section will be 'Any', that is, the exit section is determined by the shortest path. In the case of a destination centroid, a percentage is fixed to each exit section and then an exit section is assigned to each vehicle (see Section 3.2.2.6 for details).

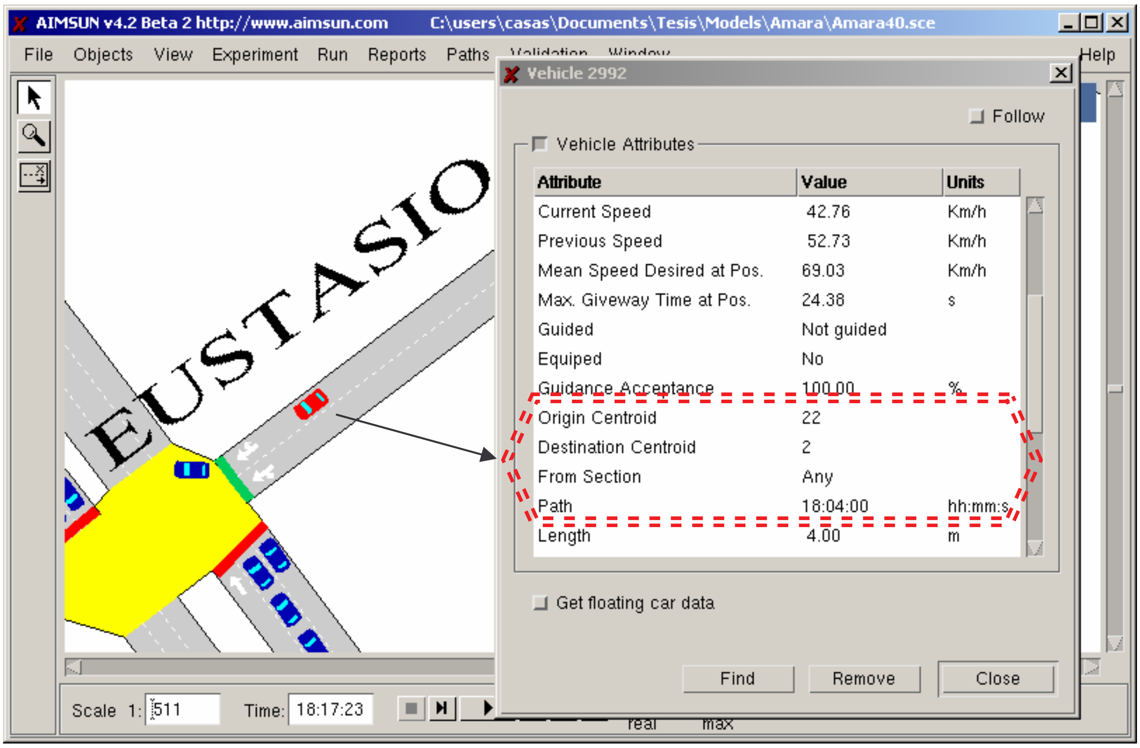


Figure 14.7. Vehicle information

#### 14.3.2.4 VEHICLE COLOURING

During the simulation, the vehicles can be coloured either by origin or destination or both. This helps to identify the origin and destination quickly during the calibration and validation process and to understand the path assigned to each individual vehicle.

Figure 14.8 shows a network in which all vehicles are coloured by OD pair. The colours assigned to their destinations are displayed in the front half of the vehicles and the colour assigned to their origin in the back half of the vehicle.

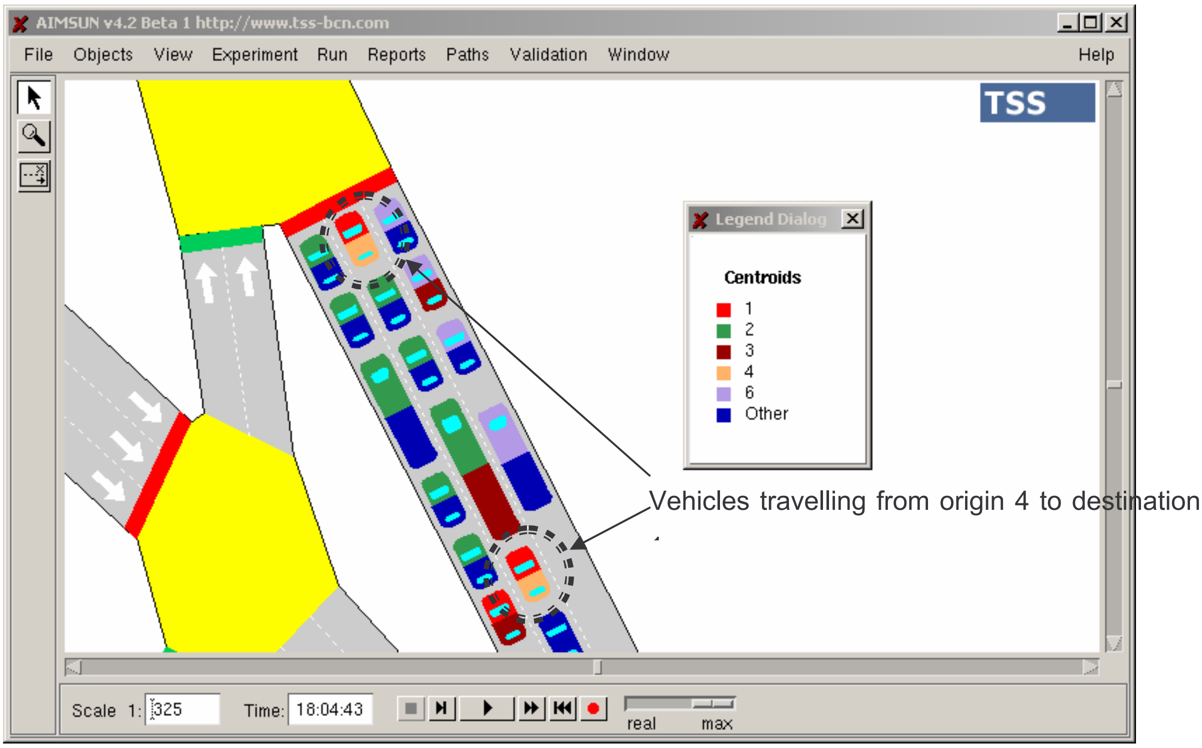


Figure 14.8. Vehicle colouring per OD pair