

**Quadstone Paramics V5.0
Technical Notes**

Quadstone Limited

Version No. 1.0
19/10/2004

Distribution Classification:
Public Distribution



Paramics is a registered Trademark.

The contents of this document are the copyright of Quadstone Limited. All rights reserved.

19/10/2004/1.0

Project title: Quadstone Paramics v5.0
Document title: Technical Notes
Document Identifier: RRB/pv5.2/rei01
Distribution Classification: Public
Document history:

Personnel	Date	Summary	Version
Paramics Staff	19/10/2004	1st draft	1.0

Approval List:

Ewan Speirs and Richard Braidwood (Quadstone)

Distribution List:

Public Distribution



QUADSTONE

Telephone: **+44 131 220 4491** Facsimile: +44 131 220 4492

<http://www.paramics-online.com>

16 Chester Street Edinburgh EH3 7RA Scotland

Document Ownership and Confidentiality:

This document and all ideas, methodologies, algorithms, design notes, descriptive software text, etc, contained within remain the sole property of Quadstone Limited. This document or any subpart of its contents must not be distributed, in whole or part and in any medium, to any third party not listed on the distribution list provided above. Any non-Quadstone employee listed on the distribution list provided above must seek the express permission of Quadstone before divulging any details of the contents of this document in any way, through any medium, to any third party not listed in the distribution list provide above for the document and its contents.

Table of Contents

Table of Contents	3
1 Technical Notes	4
1.1 Introduction	4
1.2 File reading	4
1.3 Assignment	5
1.3.1 Assignment: Routing Trees	5
1.3.2 Assignment: Familiarity	5
1.3.3 Assignment: Global Weights	5
1.3.4 Assignment: Cost Factors for Link Categories	5
1.3.5 Assignment: Cost Factors for Individual Links	6
1.3.6 Assignment: Restrictions	6
1.3.7 Assignment: Dynamic Assignment	6
1.3.8 Assignment: Cost Perturbation	6
1.3.9 Assignment: Technical Note on Implementation	7
1.4 Simulation Iteration	8
1.4.1 Simulation Iteration: introduction	8
1.4.2 Simulation Iteration: Release Vehicle	8
1.4.3 Simulation Iteration: Set Speed	9
1.4.4 Simulation Iteration: Change Lane	10
1.4.5 Simulation Iteration: Move Forward	10
1.4.6 Simulation Iteration: Transfer Vehicle	10
1.5 Car-Following	10
1.5.1 Car-Following: Target Headway	10
1.5.2 Car-Following: Modes of Acceleration	11
1.5.3 Car-Following: Cruising Modes	12
1.5.4 Car-Following: Braking Mode	12
1.5.5 Car-Following: Acceleration Mode	13
1.6 Lane Changing	13
1.7 Hazards	13
2 Paramics Support	15

1 Technical Notes

1.1 Introduction

The physical layout of the network is represented by nodes, links and a large number of other associated objects that describe the 3-dimensional geometry and the way in which this geometry constrains the motion of vehicles. The demand placed on the network is represented by a mosaic of vertically extruded polygons (the *zone map*) in combination with a table of trip counts of size (the *OD matrix*). The mosaic is overlaid on the physical network in such a way that one or more links of the network fall within each of the extruded polygons. Each element of the demand table represents the number of trips from an origin to a destination in the time period associated with the mosaic-table pair. There can be any number of mosaic-table pairs, but normally only one is modelled at any one time.

Within the simulation vehicles are released from the origin zone and make their way towards the destination zone, with limited knowledge of the network other than their destination and their immediate environment. The simulation progresses by *time-steps*, of user-defined size. Each Driver-Vehicle Unit (DVU) is represented by a set of parameters. In these notes, the term Driver-Vehicle Unit is used instead of vehicle to reinforce the notion that the decisions and the perceptions of the driver are being modelled as well as the physical characteristics of a vehicle.

Of the DVU's parameters, some are static, while the rest are dynamic and are updated at each time-step of the simulation. A DVU's static parameters include:

- type, age of vehicle
- origin, destination
- aggressiveness, awareness

The *type* parameter characterizes the DVU as being one of a number of pre-defined types of vehicles. Each of these types has an associated set of physical parameters, which include:

- colour (for display)
- length, width, weight, height
- maximum speed, maximum acceleration, maximum deceleration
- mode, demand information, familiarity, perturbation

A DVU's dynamic parameters include:

- position (lane and distance from stop line), speed, acceleration
- next turn intention, target lane range, target lane on next link

- exit index (on roundabout)
- time stopped on journey, time stopped on link
- travel time on journey, travel time on link

The Paramics car-following and lane-changing model has been developed over a period of years from 1992 to 2004. It is loosely based on a number of other models, principally the Hans-Thomas Fritzsche paper, "A model for traffic simulation", but in most respects it was created from scratch, with the primary objectives being to demonstrate validity from two points of view:

- using simulation it should show a close correlation to an array of observed numerical data for urban and inter-urban roads in differing conditions around the world.
- using computer graphics it should show a close correlation to visual observations, both on video and "in the mind's eye" (subjective validation).

This appendix will give an overview of key aspects of the Paramics car-following simulation engine; however for reasons of commercial confidentiality will not reveal every detail.

1.2 File reading

The initialization phase reads each of the input files in turn. Many of the files are optional, and execution will continue if an instance of one of these files is not found. Execution will terminate if the either of the nodes or links files is not readable within the current data directory. The data files are read in groups, in an order that is based on interdependencies between the files, and also on issues concerning modularity: some of the Paramics modules require only a subset of the data files, and these subsets are grouped together for this reason.

The groups of files are as follows:

- configuration, pollution, vehicles, behaviour, restrictions
- nodes, hazards, centres, categories, links, kerbs, stoplines
- detectors, beacons, events, zones, car parks, incidents
- priorities, plans, buses
- turn-lanes (junctions, roundabouts, nextlanes)
- annotation

1.3 Assignment

1.3.1 Assignment: Routing Trees

Apart from file reading, a major part of the initialization time is for construction of the base routing trees. Within Paramics, the routing trees are stored as tables of costs to the destination, but the description that follows will refer to trees. At least two routing trees will be constructed, perhaps more, depending on physical restrictions within the network that apply only to certain types of vehicles.

The routing trees within the Paramics model are used to effect the route-choice decisions of each DVU within the network. The route choice decisions are dynamic, in that at any time in the model, a DVU knows only what its next N route choice decisions will be, where N is quite a low number, and what its final destination is. A DVU does not carry with it a route for its entire journey, but makes a decision on each link as to what turn it shall make at the end of the Nth link ahead of its current position.

The exact nature of each route choice decision depends upon which of the assignment options available within Paramics have been selected. There are a number of options, some of which apply only to larger networks where there are obvious alternatives between origin-destination

(OD) pairs. Each of the options builds on the basic strategy of using one or more route cost tables, each of size $N_z M_i$, where N_z is the number of (destination) zones, and M_i is the number of links in the network.

At the most basic level, assignment is done using static link costs, using an all-or-nothing approach.

This assignment model causes all DVUs to choose the same route between each origin and destination, which is too simplistic for large networks, but it is suitable for small networks, where there is often only one route between each OD pair. This assignment model is commonly used on highly constrained networks that are being studied for the purpose of traffic impact assessments, for example. The static cost for each link is set to be the free-flow travel time for that link, where in this context, the free-flow speed is assumed to be the speed limit defined for each link. Note that within Paramics, a free-flow speed does not form part of the input data

set: the free-flow speed is a derived value based on the mean of the speeds that vehicles achieve while aiming to travel at their *target speed*. The car following model is based on a target speed for each DVU. The target speeds are a distribution based on empirical data using the speed limit for each link as a reference point.

The method by which a route *tree* (or *cost table*) is constructed from the link costs is described in the section 1.3.9 Assignment: Technical Note on Implementation

1.3.2 Assignment: Familiarity

To generate a spread of routes for a given origin-destination trip, each DVU in the simulation is assigned a value of familiarity. Two routing trees are constructed, where one is based a value of *signposted* link costs, and the other is based on *real* link costs. This attempts to model the extra knowledge a familiar driver would have in a network, where a back-road or side street might offer a better route to the destination. In the model, an unfamiliar driver will follow only the signposted or *major* routes. The signposting is affected in Paramics by marking certain links as being *major* while the others are marked as *minor*.

1.3.3 Assignment: Global Weights

To compensate for a difference in perceived costs that take the difference in distance and difference in road price (or toll) into account the user can apply global weights to the cost calculation for each link. Three coefficients can be defined, and these are used to calculate the cost for a link, measured in minutes, as follows:

$$\text{cost} = aT + bD + cP$$

where:

- a is unit-less, and is a weighting. Default 1.0
- b is in minutes per kilometre. Default 0.0
- c is in minutes per unit of monetary cost. Default 0.0
- T is the free-flow travel time (minutes)
- D is the length of the link (km)
- P is the price of the toll (monetary cost units)

The length of a link is calculated automatically, but the price of a toll for any link must be specified explicitly by the user.

1.3.4 Assignment: Cost Factors for Link Categories

To aid the calibration of a network to observed data, cost factors can be applied to each category of link to account for variations to the true cost of a link that are implied by signing in reality. This type of cost factor is seen only by unfamiliar drivers, with the assumption that familiar drivers will know the true cost of a link and will use short-cuts and rat-runs through the network if applicable. The proportion of drivers in any particular vehicle type that are familiar is a user-defined parameter. The default value is 0.85. It is possible, however, to apply the Category cost factors to both familiar and unfamiliar drivers using the **Category Cost factors Visible to All** option (see Assignment: page **Error! Bookmark not defined.**).

1.3.5 Assignment: Cost Factors for Individual Links

As a further aid to calibration, link-specific cost factors can be applied which will be recognised by the entire vehicle population. This can be used to account for unmodelled objects that affect the free-flow speed, or otherwise change the basic cost of a link. However, these cost factors will not change the travel speed or travel time along a link directly, so if a cost factor is used to account for unmodelled pedestrian crossings or the like, a good assignment model may result, but the travel times of the vehicles that do choose the factored route will be lower than expected, and in all probability lower than observed.

1.3.6 Assignment: Restrictions

Facilities exist within Paramics to create a number of physical restrictions within the network. For example, it is possible to specify a maximum height restriction for a link, which could be used to model a low bridge. Each vehicle within the simulation has an associated height, so in order to ensure that vehicles that are too tall for the low bridge use a different route to their destination, it is necessary to build another tree for these vehicles with the cost for the restricted link set to infinity. The total number of additional route trees constructed will depend on the number of restrictions, and the physical parameters of the vehicle types within the simulation: restrictions may be grouped into sets, where one set of restrictions affects one type of vehicle. Height was used in the example above, but the same could be said for width or weight. Minimum restrictions can also be used to model, for example, loading ramps used by good vehicles only or roads that ban two-wheeled vehicles

1.3.7 Assignment: Dynamic Assignment

Wardrop's principle states "under equilibrium conditions traffic arranges itself in congested networks such that all used routes between an OD pair have equal and minimum costs while all unused routes have greater or equal costs". While this holds some truth, few would suggest that this ideal equilibrium ever comes about, especially as network demand is almost always dynamic and time-variant, and the congestion patterns that result are affected by the time-variant state of the road network, due to weather conditions, road-works and incidents.

The driving force of the Paramics simulation model is an OD matrix applied to a zone map combined with a time-varying profile. This means that the demand on the network between each OD pair can vary in time and can also vary relative to other OD pairs. This leads to a congestion pattern that is also time-variant. To model the route-choice decisions that drivers would make based on their knowledge of a time-varying congestion pattern, the user can enable cost-table recalculation on a regular basis, perhaps every five minutes of simulation time. The cost recalculation option, when selected, uses mean simulated travel times for links, rather than estimated free-flow travel times. This revised travel time cost can then be injected

back into the weighted and factored link cost calculation used previously to create a new routing tree.

The cost feedback option within Paramics is of limited use in isolation, as it will almost certainly result in oscillation between alternative routes if it is applied to all drivers. However, only the route tree for familiar drivers is recalculated at each stage: unfamiliar drivers will still follow the signposted routes on the links marked as being major. The ratio of unfamiliar to familiar drivers will determine the damping factor in the feedback control loop: a higher ratio will result in a reduced likelihood of instability. In order to control any prospective instability the **Feedback Decay** and **Feedback Smoothing** factors (see Assignment: page **Error! Bookmark not defined.**) are available as control parameters.

The justification behind this method is that familiar drivers will have developed experience over time of the true costs of each of the possible routes in the network, and cost feedback and dynamic route recalculation aims to model this phenomenon. It is possible within Paramics to run the model with cost feedback enabled for a period of time, and then save the link costs to file. These link costs can be used as background or base costs that can be loaded in for subsequent runs of the simulation. However, it should be pointed out that cost feedback within Paramics leads to equilibrium only if the time profile of the demand applied to the network is completely flat. For a realistic simulation, it is almost always necessary to model the peaks and troughs of demand, and unless these variations happen at exactly the same time for every origin-destination pair, there will never be a state of equilibrium within the network.

The use of route cost feedback within Paramics is enhanced by the use of the next option, cost perturbation.

1.3.8 Assignment: Cost Perturbation

The cost perturbation model within Paramics utilizes a simple (non-hierarchical) logit model. A random error factor (random noise) is added to the total perceived cost of each route option as a way of modelling the variations introduced by differences in perception of cost. This *stochastic all-or-nothing* model is a variation on the Burrell model, which divides the total population into a number of subgroups, and assigns a random factor to each, so that all DVUs in any one subgroup will follow the same route. The difference with the Paramics model is that each vehicle is assigned its own random error factor. While providing a reasonably realistic distribution of demand among the alternatives it should be pointed out that this perturbation model is not completely justifiable as being an accurate representation of each driver's actions in a real network. A true representation would take account of many other factors including previous experience of driving a particular route at a particular time of day. However, any model will only ever be an approximation of real life, and if the distribution of demand produced by this model is satisfactory, then perhaps that is sufficient.

Within Paramics, the amount of random noise can be varied by applying one of two available algorithms. The standard option is referred to as the *percentage* algorithm and can be expressed as:

$$C' = \left(\frac{100-P}{100} + N_1 \right) C$$

where C' is the perceived cost, C is the original cost, P is the perturbation factor and N_1 is a random number in the range 0 to $\frac{2P}{100}$. Alternatively, a square root algorithm can be used, this is expressed as:

$$C' = C + \frac{(N_2 - 5)P\sqrt{C}}{500}$$

where C' , C and P are the same as before and N_2 is a random number in the range 0 to 10

1.3.9 Assignment: Technical Note on Implementation

The routing *trees* in Paramics are built using an algorithm that is simple and reasonably fast. It is not as elegant as some better-known algorithms (such as Dijkstra's algorithm) but it was derived from first principles to cope with the more unusual constraints of a traffic network as compared to a classic directed graph. The nodes in a Paramics network represent junctions. At any particular junction a turn from an inward link to an outward link may be barred, which translates to an infinite cost for that manoeuvre. In addition, if cost feedback is enabled the cost of making a particular manoeuvre (the turning penalty) may be set to a finite, non-zero value, to reflect the cost associated with joining a particular stream of traffic on the inward link. Therefore, the total cost from an origin node to a destination node is the sum of the costs for each of the links on the route plus the sum of the costs for the turns between the links. A classical algorithm, such as Dijkstra's, holds a cost value to each destination at each node.

To implement effectively the requirements of the cost constraints outlined above, the algorithm used within Paramics holds a cost value to each destination for each *link*, and this cost is the cost of travelling from the *beginning* of that link to the destination node. To calculate the cost value for every link for each destination the algorithm works using a repetitive sweep technique that tests each of the links in the network in turn. Initially, all links are set to have an infinite cost, apart from those links that are connected to any node that lies within the destination zone. On

each sweep, each link is examined in turn: if the sum of the costs of the inward link, the appropriate turn and the present value of cost-to-destination of the current link is less than the value stored previously in the inward link then the value in the inward link is modified. The efficiency of the algorithm is improved by including a flag variable for each link that is set when the cost value for that link is changed and cleared when the modified value has been propagated backwards to the inward links. The algorithm terminates when a sweep completes without having modified the costs on any link. The algorithm is shown as pseudo-code in Figure 1.

1.4 Simulation Iteration

1.4.1 Simulation Iteration: introduction

Following initialization the iterative simulation process (the *main simulation loop*) is started. Every iteration or time-step of the simulation a number of processes are carried out, each of which is detailed below. The simulation process continues until halted by the user, or until a specified end time is reached. The process runs as fast as it is able to on the hardware available, so that the speed-up (the ratio of simulation time to elapsed real time) depends on the hardware and the selected time-step, as well as on the number of vehicles being simulated at each timestep, and to a lesser extent the number of nodes and links in the network.

$DC[Z, L]$ is the cost of getting to destination Z from link L
 $C[L]$ is the free-flow travel cost of link L
 $TC[K, L]$ is the cost of turning from link K to link L
 $F[L]$ is the flag for each link marking its value as having been changed
 W is a flag indicating that at least one link has been changed

```

for Z in {all destination zones}
  for L in {all links in the network}
    clear flag F[L]
  for N in {all nodes that lie within Z}
    for L in {all links flowing into N}
      DC[Z,L] = C[L]
      set flag F[L]
    set flag W
  while W is set
    clear flag W
    for L in {all links}
      if flag F[L] is set
        for K in {all links that flow into L}
          if DC[Z,K]>DC[Z,L]+TC[K,L] + C[K]
            then
              DC[Z,K]=DC[Z,L]+TC[K,L] +
                C[K]
            set flag F[K]
            set flag W
  
```

Pseudo-Code for the Paramics Route Cost Algorithm

1.4.2 Simulation Iteration: Release Vehicle

The first process in the main simulation loop releases vehicles onto the network, according to the OD matrix defined. Vehicles are released onto any link whose midpoint lies within the origin zone. Vehicles may be released at any point on the link (normally at the centre of the largest space) that attempts to simulate parked

vehicles moving away from the kerb, out of driveways and emerging from unmodelled side-streets. The number of vehicles released onto each link depends on the value of the *release rate* parameter assigned to that link. This parameter is initialised automatically based on the length of the link, but can be set by the user to reflect the situation on the ground.

The vehicle release algorithm uses a number generated from a uniform random distribution to determine the time between releases. The technique is best illustrated by example. Assume a 4-zone network (with zones A,B,C,D) and that the OD matrix stipulates that 200, 300 and 400 vehicles travel every hour (or alternatively, every 3600 seconds) from zone A to zones B, C and D respectively. Summing these values shows that 900 vehicles in total leave zone A every hour. Every second a random number R is generated from uniform distribution having a range [1,3600]. If R lies in the range [1,200] (the first 200 numbers) then a vehicle is generated going from A to B. If R lies in the range [201,500] (the next 300 numbers) then a vehicle is generated going from A to C. If R lies in the range [501,900] (the next 400 numbers) then a vehicle is generated going from A to D. If R lies in the range [901,3600] then no vehicle is generated. On average, over the space of an hour the correct number of vehicles will be generated going from zone A to zones B, C and D respectively.

The distribution of T_R , the time between releases, measured in seconds, has a negative binomial form. That is to say that $P(n) = q(1-q)^{n-1}$ where $P(n)$ is the probability of $T_R = n$ and q is the probability of a vehicle being released in any one second.

Within Paramics a **Marsaglia** random number generator is used which is a marked improvement on the standard pseudo-random number generation functions commonly included as standard with programming environments (such as `rand()` in the C Programming Language). The **Mersenne** random number generator is also available.

One of the main points of using microscopic simulation is to highlight quickly and easily any areas of a traffic model where demands exceeds capacity. Clearly if there is not enough capacity within an origin zone to accept all the vehicles that require to be released according to the specification in the OD matrix then there is either a problem with the representation of the network, or the figures in the matrix do not reflect reality with a sufficient degree of accuracy.

If this is the case within Paramics, a virtual stack of vehicles is created that represents the difference between the demand specified in the matrix and the capacity of the network within the origin zone. If the demand subsequently falls (because a demand profile is being used) the vehicles on the stack will be released as soon as space becomes available in the network. This has the effect of smoothing out any 'spikes' in the profile. The size of any such stacks can be visualized on the Paramics GUI, using the **Hotspots** option or the **Release Quotas** display.

The user can verify the number of vehicles being released into the network by using one of the measurement options: `generate release counts`. This records a matrix

of vehicles actual released, in OD format, with rows and columns summed, that can be compared directly with the input OD matrix.

If the vehicle generation algorithm decides that a vehicle should be released from an origin zone, a link is then chosen out of all the possible links within the zone, using a similar technique. However, there are further constraints: a link L is chosen for releasing a vehicle to destination Z only if the cost, $DC[Z,L]$ is finite - i.e. it is possible to reach zone Z from a starting position on link L . Sometimes a link will be chosen that is not part of the least cost route to Z , looking at the origin zone as a whole, but this is intentional: it is intended to represent the case where a vehicle has been parked on the wrong side of the street and must be driven around the block in order to start heading towards the ultimate destination. The release parameters and release quotas can be altered using the Programmer module and options in the Modeller module, such as **Release to Shortest Path**.

It is possible to maintain the same release distribution by implementing the **Split Seeds** option (see **Error! Reference source not found.**: page **Error! Bookmark not defined.**) if no edits are made to the demand release or zone/link configuration within the network. This can be used to test scenarios for which the user wishes identical release profiles during the simulation.

1.4.3 Simulation Iteration: Set Speed

The second process in the main simulation loop sets the speed of every vehicle on every link within the network. The speed-setting process takes account of several models that can be defined by the user through the API, such as the car-following model, the lane-changing model, the fixed-point acceleration model and the gradient acceleration model. The general principle used within the speed-setting algorithm is that a number of constraints are defined and an acceleration value is calculated for each constraint. The lowest of all the acceleration values, which may be negative, i.e. a deceleration, is then applied to the current speed to determine the speed of the vehicle in the next time-step. The order in which the constraints are considered is not important as it is always the lowest value that is used. The constraints are as follows:

- The speed of the vehicle in front, either on this link or on the next or subsequent links.
- The Paramics car following model is described in more detail in section 1.5.
- The target speed for the link. This is a value that is based on the speed limit and modulated by the value of the behaviour parameters assigned to the DVU. Road curvature is also used in this calculation.
- The speed of the nearest vehicle in the adjacent lane, looking forward. This is a constraint for a DVU if a stimulus to attempt a lane-changing

manoeuvre has been forthcoming from the lane-changing model. The lane-changing model is described in Section 1.4.4

- The target speed at the end of the current link or any of the subsequent links, if the subsequent links are so short the DVU should start to change speed at the current point to attain the target speed given the physical constraints on deceleration, or acceleration. The target speed may be zero, if there is a Stop Line or a traffic signal on red, or may be nonzero or low if the user has defined the *end speed* parameter for the link. The *end speed* parameter allows point speed restrictions such as traffic calming measures to be modelled. The points at which a DVU must take account of traffic priorities are termed hazards, and are a subset of the nodes in the network. Some nodes in the network may just join two links, for geometrical alignment, and no junction is represented by these nodes. Other nodes may represent nodes at which DVUs approaching from one direction may have to pay attention to traffic priorities, but those from another direction have major priority and will free-flow through the node. Hazards are described more fully on page 13. The point at which a DVU sees a hazard is controlled by the signposting for that hazard. By default this distance is 250 metres for urban roads and 750 metres for highway roads. The signposting distance within Paramics is intended to enable the modelling of the weaving that occurs at the point where drivers discover which lane they should be in for their next turn. Of course, this would be the case only for drivers unfamiliar with the network, as familiar drivers would know which lane to adopt much further in advance. An end speed may also be set automatically for sharp changes in geometry at the end of a link, using the same technique as for road curvature
- A maximum passing speed based on the speed of the vehicle in the adjacent lane. This models the *friction* that exists between lanes of traffic: stopped or slow-moving traffic in one lane tends to slow down passing traffic in the adjacent lane because of the threat of a vehicle moving out suddenly from the stopped traffic. In Paramics Version 1.50 the lane friction parameters are pre-set. Lane friction is not effective on links which have the *no lane change* flag set.
- A courtesy factor that allows *zipping*: where there is a lane drop, DVUs on the second-to-outside lane will extend the space required by the car following model in such a way as to leave room for a vehicle in the dropped lane to merge. This also applies to the second-to-inside lane, if the lane-drop is on the inside. Each DVU allows this to happen once on each link, resulting in an effective alternating lane advance motion.

- Any approaching bus-stop, if the vehicle is a bus.

1.4.4 Simulation Iteration: Change Lane

The lane-changing model is prioritized into two levels: urgent and non-urgent. Within Paramics, a DVU will attempt to execute a lane-change manoeuvre as a response to either

- a single urgent stimulus.
- a series of (nominally) five contiguous and consistent non-urgent stimuli. This prevents lane-changing events from occurring within the model due to rogue stimuli produced by unsuitable transient conditions.

An urgent stimulus is generated if a DVU finds itself outside its target range of lanes. Near to a hazard (see page 13), the target range is controlled by the number of lanes available on the exit link appropriate to the DVU's choice of route. At all times the target range is adjusted subject to the behaviour parameters associated with the DVU: a higher level of aggression causes a DVU to move to the outer (higher speed) lanes, a higher level of awareness causes a vehicle to adopt the target lane for an impending turn sooner.

An urgent stimulus is also generated if the vehicle is caught in a stationary line of traffic (as a result of an incident, for example).

A non-urgent stimulus can be generated by a number of conditions, which are themselves prioritized, as follows:

- Move in or out because of constraints imposed by a fixed physical object such as a ramp joining, or a climbing lane.
- Move in or out as suggested by free-flow lane-changing model. This can be defined as part of the user API, or the standard free-flow model can be used.
- Move in or out on an urban road in such a way as to spread the total demand over the available road space. In the absence of other stimuli this prevents false congestion building up.

Note that these conditions describe what is necessary for a DVU to receive a stimulus to attempt to change lane in either direction. The actual lane-changing manoeuvre will not occur unless a suitable gap exists. The function defining the suitability of a gap (the *gap-acceptance* function) can either be defined by the user, or the standard function can be used.

1.4.5 Simulation Iteration: Move Forward

Having set the speed, each DVU is moved forward by an amount equal to the speed multiplied by the current time-step. A check is then made if this movement has taken the DVU into a signposting zone, and if so, the target range of lanes is set. The target range of lanes is described in Section 1.4.4.

1.4.6 Simulation Iteration: Transfer Vehicle

For each DVU, if the most recent movement has taken it within one time-step of the end of the current link, travelling at the current speed, an attempt to transfer it onto its next link will be made. Each link has an exit and entry point, and between the exit point of one link and the entry point of another link a vehicle is modelled in two dimensions, rather than one, and is said to be *on the node*.

Transfer attempts are made from the receiving link's point of view. If there is blocking back on the receiving link then no vehicle will be transferred. Otherwise, all vehicles that have registered an interest in transferring to the receiving link in a particular time-step are evaluated in terms of their priority. Vehicles on an amber light have highest priority, followed by the sequence of priorities Major, Medium, Minor. Once a vehicle has been selected, a check is made for conflicting traffic streams, which if found will halt the transfer process.

A successful transfer to a new link updates a number of parameters associated with the vehicle, including the following:

- the target lane range
- the next link (for route choice)
- the target lane on the next link
- the exit index if the vehicle is on a roundabout
- the blocking-back flag for the receiving link, depending on the space remaining

1.5 Car-Following

1.5.1 Car-Following: Target Headway

Each Driver-Vehicle Unit (DVU) in the Paramics simulation has a target headway. The mean value for target headway is 1 second by default, however this can be specified by the user in the **Configuration Manager**.

The target headway for each DVU varies around the **mean target headway** parameter, depending upon the value of certain parameters assigned to the DVU, as shown in the following table:

Condition / Parameter Variation	Factor
Vehicle type: e.g. Heavy goods vehicles	1.8
Presence of single-lane highway (no lane changing possible, as in road-works), Close to motorway merge	1.5
Close to motorway merge (accept smaller headway for limited time)	0.5
Close to traffic signals: straight ahead	0.5
Close to traffic signals: turning left	1.1
Aggressiveness (everywhere) $Ag \leq 4$	$\frac{36 - 5Ag}{16}$
$Ag \geq 4$	$\frac{12 - Ag}{8}$
Awareness (Near Lane-Drop) $Aw \leq 4$	$\frac{Aw + 4}{8}$
$Aw \geq 4$	$Aw - 3$

In terms of driver behaviour, a high aggression value will cause a DVU to accept a smaller headway. Similarly, a high awareness value will affect the use of a longer headway when approaching a lane drop in order to allow DVUs in other lanes to merge more easily.

If not constrained by an approaching junction, a DVU will vary its speed in order to attain its target headway with the DVU in front.

The reaction time of the driver is simulated by basing the calculation of the necessary acceleration/deceleration on the speed at which the DVU in front was travelling at some time in the past.

A default mean reaction time of one second is used, and this is modelled by giving each DVU a memory, so that it carries with it not only its current speed and position, but a record of its speed and position for a specified a number of

timesteps in the past. This is known as the **speed memory** and is set using the **Configuration Manager**, however greater values are recommended if a greater number of timesteps per second are used. The default value is 3, however greater accuracy can be achieved by maintaining a large number for the **speed memory**. However, as with many simulation data parameters there is a trade-off between accuracy and operational performance.

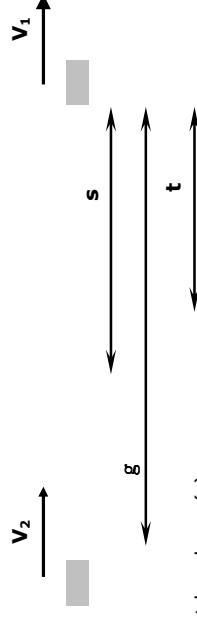
It is recommended that the speed memory value is set so that then number of timesteps (in seconds) is equal to or greater than the mean target headway + 50%.

The effect of this reaction 'lag', due to the application of a reaction time, results in the accurate simulation of backward travelling shock waves. In tests shock waves were found to travel upstream at approximately 11 km/h.

Reducing the driver reaction time is a considerable factor when considering the throughput of vehicles along a link.

1.5.2 Car-Following: Modes of Acceleration

A DVU changes its speed according to its perception of the speed of the DVU in front. These changes are normally smooth, following linear functions, but may be abrupt following the detection of one of two binary signals. These signals are visible brake lights and perceptible acceleration (of the DVU immediately ahead). There are therefore three modes of following within the Paramics model, referred to as *braking*, *cruising* and *acceleration* modes.



- h: target headway (s)
- s: target separation (m)
- t: adjusted target separation (m)
- g: current gap (m)

Figure 2: Diagrammatic representation of target point for car-following algorithm

For all modes of following, the concept of a target point is used. This point is based on a position at a distance *s* behind the leading DVU, the target point is then **adjusted** to improve the car following behaviour. The distance *s* is the target separation distance and is shown in Figure 2. Note that the dimension of *h* is seconds, and the dimensions of *s*, *t* and *g* are meters. Note also that the current perceived speed is the *actual* speed at some time in the past, due to the influence of the reaction time modelling.

The distance is calculated as follows:

$$s = hV_1$$

Where

$$\Delta V = V_1 - V_2$$

However, in order to pull DVUs together at a faster rate than would be the case with linear acceleration alone, the **adjusted** target point position is calculated by:

$$t = \frac{s^2}{g}$$

where g is the current distance between the DVUs.

In addition to the use of an adjusted target point, a **bunching acceleration**, *c*, is also used to bring DVUs together rapidly.

where $k_1 = 1.0s^{-2}$. Clearly this term decreases to zero as DVUs close in toward the minimum separation of 2 meters.

1.5.3 Car-Following: Cruising Modes

There are five discrete areas A, B, C, D and E in the headway/velocity-difference phase space, as shown in Figure 3. Each of these regions has a separate expression for acceleration, expressed as a , and derived below. Of these five, three correspond to conditions where the DVU ahead is cruising:

- In region A, the following DVU has overshoot the target point (the headway is less than the target value), and an attempt is made to achieve the target speed as quickly as possible, i. e. as fast as the physical constraints of the DVU allow.
- In region B, the leading DVU is pulling away from the following DVU
- In region C, the DVUs are at a constant separation or coming together.

The base acceleration values (with dimension ms^{-2}) for each of these regions are as follows:

$$a_A = k_2\Delta V + k_3 \frac{g-t}{g}$$

$$a_B = k_2\Delta V + k_1 \frac{g-t}{t}$$

$$a_C = c - \frac{(\Delta V)^2}{g-t}$$

Where $k_1 = 1.0s^{-2}$ as before, $k_2 = 1.0s^{-1}$ and $k_3 = 0.005s^{-2}$

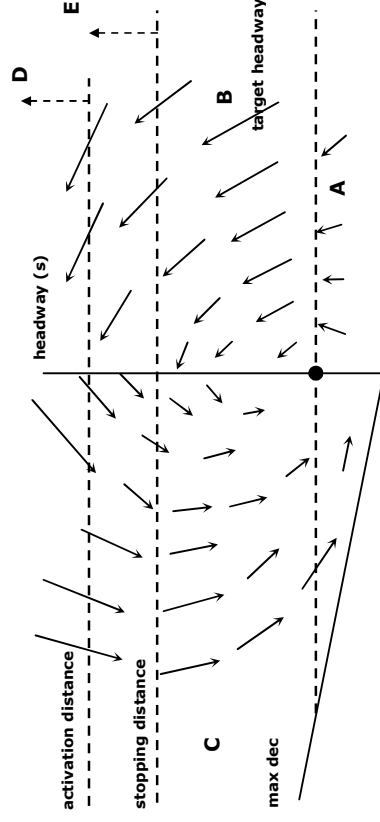


Figure 3: Pictorial representation of headway/velocity-difference phase space, and the resultant acceleration values produced by the Paramics car-following algorithm. The magnitude of an arrow is a representation of the acceleration derived by the car-following algorithm for a given $(\Delta V, h)$ pair. Note that the arrows are not to scale.

1.5.4 Car-Following: Braking Mode

When the DVU ahead is perceived to be braking (its deceleration is greater than a certain threshold), its perceived speed is decreased by an amount dependent on its maximum deceleration rate. This models a driver's expectation that if the DVU ahead is braking, its speed in the next time step will be considerably less than at the current time-step. The method of application of speed difference and current separation to acceleration ensures that a DVU will over-compensate if the DVU ahead is braking, and that this over-compensation will increase as the distance between the DVUs decreases. This, and the time-lag introduced by modelling reaction time results in shock-wave characteristics as seen typically in highway traffic flow.

However, because the speed of the DVU ahead is predicted, and may have a resultant value of zero, a threshold is used to test whether the following DVU is close enough to be in danger of collision. If not, the acceleration is set to a positive value. This corresponds to region D in Figure 3.

So

$$a_D = k_3$$

where $k_3 = 1:0ms^{-2}$

1.5.5 Car-Following: Acceleration Mode

If the DVU ahead is perceived to be accelerating at a high rate, and is more than the following DVU's safe stopping distance away, acceleration is set to the maximum value. This corresponds to region E.

$$a_E = a_{MAX}$$

1.6 Lane Changing

Lane changing in the Paramics model is done using two devices:

- a gap-acceptance policy
- a historical record of suitable gap availability

The gap acceptance policy is linked with the car-following model, in that the accepted gap is based on the target headway. In the following description, the term *target lane* will be used to describe the lane that a DVU is aiming to move into from its current lane.

If traffic in both the current lane and the target lane for a DVU is moving at a constant speed, a gap must exist both in front and behind the position it would occupy that is at least as large as the target headway for that DVU. If there is a speed difference between the two lanes, this expression is extended to take into account the time it would take for the DVU to attain the speed of the DVU it would follow in the target lane.

So, if DVU_0 is the DVU under consideration, and DVU_1 and DVU_2 are the vehicles in the target lane that are ahead and behind of the position DVU_0 would occupy, then both of the following must be true for the lane changing manoeuvre to take place:

$$g_1 > d_{AV1} + hV_1 \quad g_2 > d_{AV2} + hV_2$$

where:

$$d_{AV1} = t_{r0} + \frac{\Delta V1}{D_0}$$

$$d_{AV2} = t_{r0} + \frac{\Delta V_2}{D_2}$$

$$\Delta V1 = V_1 - V_0$$

$$\Delta V2 = V_2 - V_0$$

V_N is the current speed of DVU_N

D_N is the maximum deceleration (braking rate) of DVU_N

g_1 is the gap between the back of DVU_1 and the front of DVU_0

g_2 is the gap between the back of DVU_0 and the front of DVU_2

If both these conditions are true continuously for a period of T_{LC} seconds then the lane-changing manoeuvre will take place. The value of T_{LC} varies depending upon behaviour and location parameters, but is typically in the range 3-6 seconds. The T_{LC} parameter can be altered using the Paramics Programmer module.

1.7 Hazards

The points at which a DVU must take account of traffic priorities are termed hazards, and are a subset of the nodes in the network. A hazard may be any one of:

Diverge one or more lanes leaving the mainline carriageway

Fork a road with N lanes dividing into two roads of n_1 lanes and n_2 lanes,

$$\text{Where: } N = n_1 + n_2$$

Split an either-way choice at ghost island, valid only if $N = 2$; $n_1 = 1$ and $n_2 = 1$

Yield one stream of traffic has priority over the other

Confluence equal priority streams join, with no conflict, as $n = N_1 + N_2$

Ramp merge a distance-limited opportunity to merge sideways into the main stream of traffic

Lane Closure one or more lanes closed

Narrowing lane drop

Widening lane gain

T-Junction a T-junction where this road forms the base of the T.

Cross-Roads a 4-way junction

Left Turn a T-junction where this road forms the right arm of the T

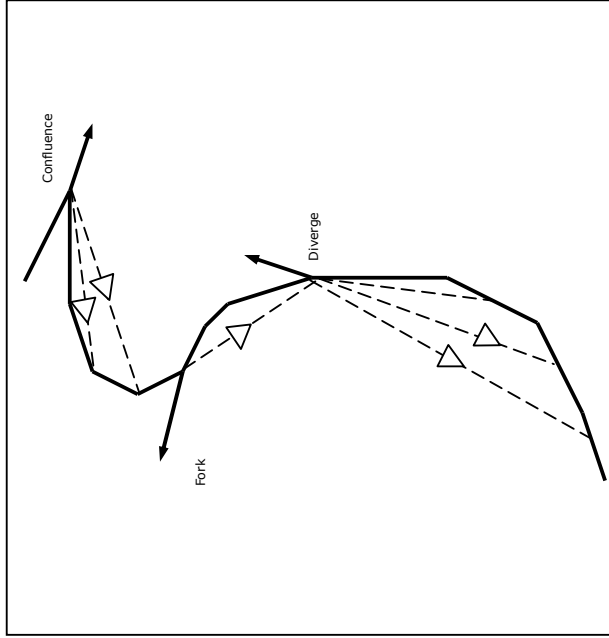
Right Turn a T-junction where this road forms the left arm of the T

Signals a signalised junction

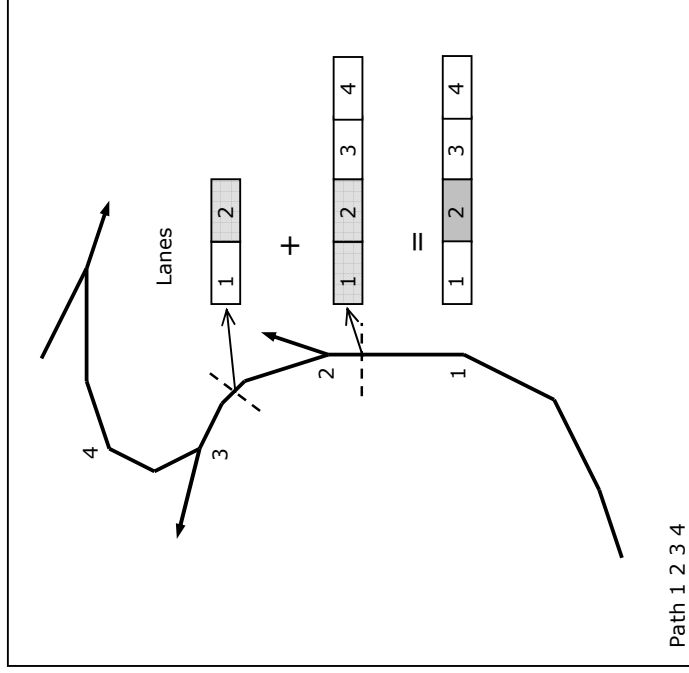
Certain combinations are valid, for example a T-junction may be signalized, or a diverge and a lane closure can be combined to form a representation of a D4M junction.

Any link can have any of its own lanes closed also, and this is held in a field in the main part of the link. In this way, any combination of closed lanes on a local link and on any subsequent link can be dealt with. Another reason for having two separate fields for lane closures is that local lane closures are a 'hard' limitation that must be dealt with immediately, and at all times, whereas closures on an approaching link are slightly less urgent.

A link may have a hazard at its end, or in the case where there is a link shorter than (say) 1000m before the hazard, the link will have a pointer to the link immediately before the hazard. At network setup, hazards are detected automatically, and warnings are propagated backwards. This is shown in the figure below:



Hazard warnings are used to set up a target range of lanes for each vehicle, depending on its destination. On passing a warning (warnings occur at the specified distance from the hazard), the vehicle looks ahead to the next hazard and calculates which lanes are suitable for its next turn. The vehicle then calls the function again, with parameters set up as if it had just passed the next hazard, and finds another range of lanes, suitable after the hazard. This second range is used to modulate the first range, as shown in the figure below:



2 Paramics Support

For more detailed information and online support access Quadstone's Paramics web site at:

<http://www.paramics-online.com>

Quadstone Limited, 16 Chester Street, Edinburgh EH3 7RA, Scotland

Fax : +44 131 220 4492



QUADSTONE