

15. Special Options and Facilities

INTRODUCTION

This section contains a series of notes on special features of **SATURN** and is intended to “explain” their use and/or interpretation rather than to describe the nitty-gritty of how, for example, to set up input files.

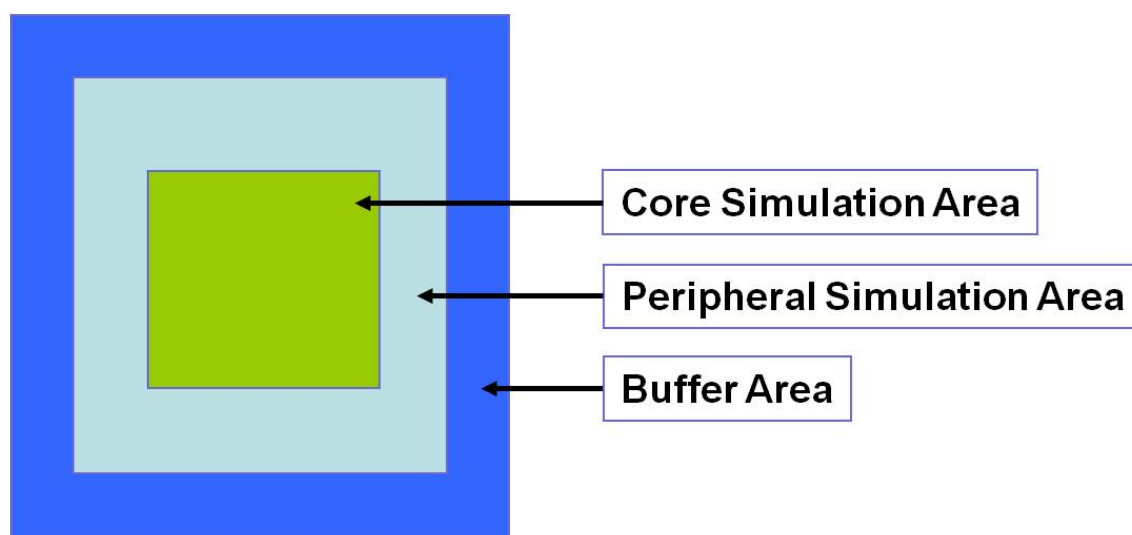
15.1 Network Aggregation and Simplification within Intermediate Bands

N.B. This section, first created in September 2011, replaces a previous section on “How Tutorials” which are no longer available.

15.1.1 General Principles of Network Simplification and/or Aggregation

SATURN networks are very often constructed in the shape of a “doughnut” (see below) where the area of most interest in terms of scheme testing is at or near the heart of the doughnut and the centre of the doughnut is coded as a simulation network with the outside made up of a buffer network (see Section 2.3).

Typical Schematic Diagram of SATURN Network Types



The justification for using the less precise buffer network description is that, if one is interested in analysing schemes at the centre of the network, the resulting impacts within the distant buffer network will be minimal and the extra time and effort required to code and run the full network as simulation cannot be justified.

A further advantage of a buffer network vis a vis a simulation network is that it has better convergence properties due to the fact that it uses “separable” cost-flow curves (see 7.1.3). Conversely simulation networks suffer potential problems of non-convergence due to the fact that, by allowing for within junction interactions, their cost-flow curves are non-separable. Very often this may introduce “noise” into the solution which makes it difficult to accurately assess the impact of relatively small schemes.

SATURN 11 has introduced the possibility of creating an **intermediate** network band (referred to as the Peripheral Simulation Area in the diagram above) which

would lie, geographically, between the central “pure” simulation network and the outer buffer network and which would be modelled at a simpler and/or more aggregate level than the normal simulation but not necessarily as coarse as the buffer network.

Two options are available for the intermediate region:

- 1) Conversion into a Fixed Cost Curve network (FCF);
- 2) Simulation to Buffer Transformation (SBT)

Fixed Cost Curves are described in Sections 15.1.2 to 15.1.6 below and the Simulation to Buffer Transformation in 15.1.7. They are compared in 15.1.8.

15.1.2 Network Simplification using Fixed Cost Curves (FCF)

The FCF transformation retains the essential **geometry** of the simulation network in that it distinguishes between separate turning movements at nodes but with **fixed** - and therefore separable - “cost-flow” or “flow-delay” curves (FCF) for each turning movement which should improve convergence and reduce “noise”. FCF may be thought of as another form of “perturbation assignment” (see 22.2.6) or “diagonalisation” (see 9.1.2).

The essential idea is that, in the intermediate FCF network, the flow-delay curves for the simulation turns are “fixed” after a certain number of simulation-assignment loops (see Fig. 9.1), presumably once a reasonably “good” level of convergence has been reached. Thus, given the general flow-delay equation of the form (see Section 8.4.2):

Equation 8.1 (reproduced)

$$t = AV^n + t_0 \quad V < C \quad (a)$$

$$t = AC^n + t_0 + B*(V - C)/C \quad V \geq C \quad (b)$$

The parameters t_0 , A , n and C are all treated as fixed for individual turns rather than as variables calculated at the end of each new simulation.

A further property of an FCF (“Fixed Cost-Flow”) description is that the same network properties may be applied to both a “do-minimum” and a “do-something” network in order to minimise noise between the two.

Finally we note that the structure of the “assignment network” in which simulation turns are represented by individual “links” is also unchanged under FCF; it is only the nature of the cost-flow curves on these turn-links which has changed. This in turn implies that a basic Frank-Wolfe assignment step will require roughly the same CPU time with or without FCF – although we would expect a reduction in **overall** CPU time with FCF due to a reduced number of assignment-simulation loops.

15.1.3 Modelling FCF nodes within a Simulation Network

Those nodes which are designated as fixed cost-flow within a simulation network are identified (and this is purely a technical detail) by an extra binary bit within an

array containing node properties (DA code 254 to be more precise). The following two sub-sections describe how this may be accomplished; here we describe the modelling differences – and similarities – between a “normal” simulation and an FCF simulation.

Both styles of node simulation start with IN profiles on all input arms (see Section 8.1) and create OUT profiles along with delays. The difference is that the FCF method is based purely and simply on the parameters in equation 8.5 (above and section 8.4.2), rather than by a full simulation of interacting flows over short time unit.

Thus, under FCF, the maximum OUT flow is determined by the minimum of (a) the IN flow and (b) the (fixed) turn capacity (C in equation 8.5 above). The delay is set by use of equation 8.5 for the current flows V.

Note that the modelling of IN/OUT flows ensures that the assigned flows are correctly modelled within the FCF network and that any flow metering (reduced flows downstream of $V > C$ movements) is correctly retained as is the distinction between “demand” and “actual” flows on all intermediate band links.

Equally any blocking back effects are retained under FCF in that the capacities C used in equation 8.5 are the capacities **post** blocking back. However any reductions due to blocking back are fixed and will not change as a result of any flow re-assignment within the intermediate band.

On the other hand a lot of the detailed information that is provided by the normal simulation, for example the “saw-tooth” style queue profiles at signals, lane choice, blocking back factors etc. etc., are either no longer available or else retain their values input at the point of “fixing”. Equally the most essential information which is being passed from the simulation to the assignment, the flow-delay curves, is, by definition, fixed rather than variable by loop. For this reason FCF networks should only be set up once a reasonably stable set of cost-flow curves have been obtained; i.e., that the simulation-assignment convergence is “good”.

15.1.4 Creating a FCF Network using SATCH

The first step in creating a “master” FCF network is to use the standard network cordoning program **SATCH** to “add” FCF nodes to an existing well converged network `old_base.ufs`; e.g.:

```
SATCH old_base control
```

To do so a new logical control parameter DOFCF is set to .TRUE. within &PARAM in `control.dat`. All other inputs in `control.dat`, including the definition of the cut links etc. etc. which define the innermost network, retain the same formats.

With DOFCF = T an additional output binary network file `old_base.ufa` is created - with the (arbitrary) file extension .UFA - which retains the same network topology as `old_base.ufs` but with three components – simulation, FCF and buffer – as opposed to the two original components – simulation and buffer – in `old_base.ufs`. See 12.1.11.

Thus the cordoned network (which is normally created as a separate self-contained network by **SATCH**) defines the innermost pure simulation component of `old_base.ufa`, the remainder of the former simulation network becomes FCF

and the buffer component (if any) is identical between old_base.ufs and old_base.ufa.

Note that neither an output network data file control.kp which normally defines the cordoned network nor a cordoned trip matrix are required in this operation; the sole purpose of running **SATCH** in this fashion is to produce the new three-level network .UFA file. (Therefore, to save unnecessary calculations and CPU, the parameter DOMAT should always be set to .FALSE.)

To complete the first stage the file old_base.ufa should be renamed / copied as, say, new_base.ufn and run through **SATALL** in order to create new_base.ufs. (The reason for using the extension .ufn is that this is the extension required by **SATALL** for an input network.) Provided that old_base.ufs was well converged in the first place and the cost-flow curves for the fixed nodes are stable then the differences between old_base.ufs and new_base.ufs should be minimal. And, hopefully, new_base.ufs will converge much more rapidly.

15.1.5 Creating a FCF Scheme Network using SATNET

Having created a “master” network, e.g., new_base.ufs above, in which certain simulation nodes have been designated as FCF, that information may be passed to a new “do-something” network to be built by **SATNET** from a .dat file, say scheme.dat, by making use of the UPDATE facilities.

Thus if UPFIL = ‘new_base.ufs’, UPDATE = T and also a new parameter UPFCF = T under &OPTION then not only are all the normal network parameters in scheme.ufn copied from new_base.ufs but equally any simulation nodes which have been designated as FCF in new_base.ufs will also be so designated in scheme.ufn. Plus the FCF flow-delay parameters in new_base.ufs (i.e., t_0 , A, n and C) will also be passed as fixed parameters into scheme.ufn.

The expectation is therefore that the modified scheme network with its added FCF nodes and, consequently, a reduced number of “proper” simulation nodes will converge much better and therefore any comparisons between new_base.ufs and scheme.ufs will have fewer problems with noise.

15.1.6 Viewing FCF Nodes within P1X

Nodes which have been converted to FCF operation may be viewed within **P1X** in several different ways.

Firstly, they may be “selected” such that either only those nodes that have been converted are displayed or vice-versa. See 11.6.5.3.

Secondly, they may be assigned a numerical “node data attribute”, 1 for converted to FCF, 0 for not, and displayed as node data within **P1X** network plots and/or processed as a node data column within **SATDB**. See 11.6.5.1 and/or 11.10.5.

Finally, the standard “print” listing of node properties includes a line to indicate FCF operation for that node.

15.1.7 Simulation Buffer Transformation (SBT): Conversion to a Buffer Network

The second method to reduce simulation-based noise in an intermediate network band is to convert that band from simulation into a pure buffer network format with

the inner segment remaining as simulation. So in this case we will still have the traditional simulation/buffer “doughnut” but with an extended buffer component.

We refer to this method as SBT – Simulation to Buffer Transformation.

The SBT transformation may be accomplished by a combination of applications of SATCH, SATBUF and SATCCS (12.1, 15.8.2 and 15.8.3 respectively) plus some text file editing to produce a suitably updated network .dat file. Thus, assuming that we start from old_base.ufs, we proceed as follows:

- 1) SATBUF old_base: to create old_base.buf with **all** simulation links in old_base.ufs converted to buffer format;
- 2) SATCCS old_base: to create old_base.map with all simulation centroid connectors converted to buffer format;
- 3) SATCH old_base control: where control.dat includes INCLUDE = T in order to produce \$INCLUDE files control_11111.dat, control_22222.dat and/or control_44444.dat to represent simulation data within the central cordoned area.

At this stage we now have all the necessary components to create a new network data file new_base.dat as follows:

- 4) COPY old_base.dat new_base.dat
- 5) Edit new_base.dat using a standard text editor, e.g., NOTEPAD, in which we:
 - a) Delete the existing 11111, 22222 and 44444 (if it exists) data segments and ...
 - b) ... replace them by \$INCLUDE references to control_11111.dat, control.22222.dat and control_44444.dat.
 - c) Add 2 extra records “\$INCLUDE old_base.buf” and “\$INCLUDE old_base.map” within the 33333 data segment (but do **not** delete any of the existing 33333 data records).

Thus, at the end of the edit, we have a network .dat file in which the 11111, 22222 and 44444 data segments refer specifically to the central (cordoned) network while the 33333 data segment has had extra data added in the appropriate format to represent the (former simulation) links and centroid connectors in the intermediate band.

We note that the two 33333 \$INCLUDE segments added in step c) above will also include the **central** simulation links and centroid connectors converted to buffer format but since the same links etc. also appear in the new 11111 and 22222 data sets they will be ignored by **SATNET** under 33333.

15.1.8 FCF vrs SBT

The two methods described above, FCF and SBT, have common objectives, that is to reduce simulation noise in areas far removed from a particular scheme where major changes would not be anticipated and to improve overall convergence and CPU. They differ in the levels of aggregation applied within the intermediate region.

Thus FCF retains the same basic geometry in the intermediate region whereby each turning movement is still represented by a single link within the assignment network with a (fixed) cost-flow curve and therefore, in terms of route choice, different turning movements from the same entry link influence route choice. By contrast with BCF the distinction between different turns has been removed.

In addition the FCF formulation permits flow metering to be modelled whereas with SBT, as with any buffer network, there is no distinction modelled between demand and actual flows.

We may also note that, to a first approximation, a network with a FCF conversion gives the same results as the original simulation network. (In fact the first assignment after the FCF transformation should give identical results to the next assignment from the pure simulation network since the cost-flow curves are identical; they only diverge thereafter to the extent that the simulated cost-flow curves change). By contrast SBT networks give quite different results immediately since the buffer-link representation is based on a quite different (and arguably less realistic) network representation than the simulated turns.

Therefore, in terms of “realism”, FCF is preferable to SBT.

On the other hand in terms of CPU and convergence SBT is the winner in that the reduced network size (roughly speaking including turns doubles or more the size of the assignment network) leads to faster run times plus, arguably, faster convergence.

Note that the original trip matrix is still valid for the transformed networks, whether under FCF or SBT, since the zone structure has not been changed at all in the new networks.

15.2 Preferences files

All interactive programs require a “preferences” or “initialisation” control file in order to set default values for various parameters. The files are assigned standard names such as P1X0.DAT, MX0.DAT, etc. (i.e. ‘program name + 0’.dat). They consist of a set of namelist-based definitions of purely internal program variables which control, for example, the size of arrows in node graphics. These therefore are the variables whose values are changed by users via the standard menus.

Formal definitions of the valid variable names in each file are not provided nor are they necessary for users. An updated version of any preferences file may be produced by the program via the files sub-menu and these will contain both the input values plus the new values of any parameters changed by the user in that session.

Hence users can “customise” a preferences file to their own individual specifications and any subsequent program runs will use these specifications.

Preferences files exist for the following programs: **P1X**, **SATED**, **SATDB**, **MX** and **SATLOOK**.

By default preferences files are stored in a specific sub-directory set by SAT10KEY.DAT (see Appendix Y) but it is possible to select alternative preference files using the “PREF” keyword in standard .bat files. For example:

```
PIX network PREF C:\DVV\PREFS\JIMBO
```

selects the preferences file JIMBO.DAT in subdirectory DVV\PREFS rather than the default P1X0.DAT.

In certain cases variables which can be namelist-set in SAT10KEY.DAT may be over-written by individual Preferences Files; e.g., GO4IT and KPEXT. In general the values set in SAT10KEY.DAT might be thought to refer to values for an “organisation” as a whole while values set in Preferences Files might be more appropriate to individual users or jobs.

If, however, a program cannot locate a preferences file it is not the end of the earth - or the program. In that case a standard set of default parameter values are adopted.

Note that a slightly different way to customise the set up of an interactive program is to use the “break” option in a key file (see 14.5.5) whereby the necessary commands are contained in the key file which initiates the run but terminates on “break” allowing the user to carry on as per normal from that point.

15.3 Network Updates (The Update Option)

It is very often the case in calibrating a network that successive test networks differ only marginally from previous tests. It is possible to take advantage of this fact by using the output from former runs to provide a realistic starting point for the subsequent run. More specifically, values of the previous flow-delay parameters (including values of the ratios of actual to demand flows, QRF – see 17.2) are extracted by **SATNET** from an “update network” to set initial values for input to the first assignment rather than starting “cold” with not very realistic default values. This has the great advantage of (potentially) significantly reducing the number of simulation-assignment iterations on the second run by making the initial assignment far more realistic.

In addition, post 10.8, selected data relating to the simulation is also extracted from the “update network” rather than setting default values in order to make the first simulation more realistic.

In order to invoke the UPDATE option two steps need to be taken:

- ◆ Set UPDATE to TRUE on &OPTION in the new network DAT file.
- ◆ Input the UFS file from the previous sequence on channel 2 to **SATNET** (which may most easily be done using the parameter UPFILE (see section 6.1) to define the filename).

We note that this procedure is very similar to the PASSQ option (17.3.1) which also (if UPDATE = F) extracts flow-delay data from a previous network file (in this case, the PASSQ file from the previous time period) The difference under UPDATE = T is that **only** flow-delay information is extracted from the update file, not the queues and suppressed traffic as with PASSQ.



Note that both UPDATE and PASSQ may be used at the same time but, if so, they must use two different input .ufs files (parametric filenames UPFILE/FILUP and FILPQ respectively). If only PASSQ is used then there is no option to cancel the flow-delay updates.

The extended **SATURN** procedures may be used here - the command format is illustrated in Section 14.4.2.

Further Notes:

- 1) The second network may in fact be structurally quite different from the first in the sense that new nodes and new links or turns can be introduced. The program is set up in such a way that only information on turns and links common to both networks are carried over. For “new” turns default flow-delay parameters are assumed. Clearly though, the more similar the two networks are, the greater the savings in CPU time.

N.B. Both UPDATE and PASSQ (17.3.1) allow the “pre-network” to have a different structure from the “main network” whereas – at the time of writing – the pre-load option PLOD does not (see 15.5.1). This is likely to change in the future.

- 2) Note that the UPDATE option as described here implies that only the network is updated, although it is also permissible to introduce a different trip matrix at the same time. If, however, one only wishes to change the trip matrix then the appropriate steps are described under the Re-start Facility in Section 15.4.
- 3) In order to ensure that the **first** assignment within the assignment-simulation loop takes full advantage of the improved initial set of flow-delay curves the maximum number of assignment iterations, normally set by the parameter NITA, is set to the maximum of NITA, NITA_S and 25.
- 4) It is possible (post **SATURN** 10.6) for a file to, in effect, update itself in the sense that an “old” UFS file, say net.ufs, may update a “new” network data file net.dat. In other words it is not necessary to re-name the network every time a minor change is made and the results from the previous incarnation are used to the full.

Note as well that, if UPDATE is set to .TRUE., but the UFS file to be updated has not been defined then it is assumed by default that the file to be updated **IS** net.ufs (when the data file is net.dat).

This option is particularly useful when running multiple time periods using SATTPX (17.4.3) since, in that case, each time period has a unique filename (e.g., neta, netb, netc etc.) emanating from a single data filename (e.g., net.dat). The individual time-period filenames will be automatically and correctly invoked if UPDATE = T in net.dat but no specific .ufs filename (e.g., net.ufs) is set.

- 5) The UPDATE option may be very usefully combined – under either path-based or origin-based assignments - with the WSTART option which adds additional information related to path flows and which improves the initial

assignment even more than just having improved cost-flow curves. See 21.3 for further details.

15.4 Updating the Trip Matrix (The Re-start Facility)

The re-start facility allows a user to carry out a full set of **SATALL** simulation-assignment loops when the only difference between the current and a former run is in the trip matrix, for example when **SATME2** is used to estimate successive trip matrices or when the assignment is part of an external demand-supply procedure, possibly using **MX**.

If there are **any** other differences at all apart from the trip matrix, e.g., changes in PASSQ flows, etc. etc., then a re-start in **SATALL** is **not** appropriate.

Re-start is effectively equivalent to UPDATE (Section 15.3), the main distinction being that it is applied directly within **SATALL** whereas UPDATE is applied in **SATNET**. It is also, under path-based or OBA assignment (MET = 1 or 2), equivalent to WSTART = T; i.e., the first assignment uses the paths from the previous assignment as a “perturbation” assignment (see 21.3).

The distinction between a normal run and a re-start is that the former must start with the network build program **SATNET** before commencing the assignment/simulation loops (see Figure 3.1) whereas the latter uses a previous network and starts with the assignment directly. Subsequent assignment/simulation loops are the same thereafter.

The first assignment requires (in effect) as input:

- ◆ The final UFS file from the previous sequence (This file contains the necessary network specifications and parameters.)
- ◆ The latest trip matrix UFM file.

The command

```
SATALL network tripod RESTART
```

carries out a full simulation-assignment loop but taking its input from the previously converged file network.ufs as opposed to network.ufn which comes direct from **SATNET**.

Note that it is the presence of “RESTART” on the command line which initiates the re-start sequence; i.e. no parameters within a .dat or control file need to be set. However an alternative DIY method to set up re-start would be to copy a .ufs file into a .ufn file yourself and create a control file for **SATALL** with the parameter REGO = T; see 7.13.2. Not recommended!

The output version of network.ufs will over-write the original input version and will include the new flows etc. If you wish to retain separate .ufs files from each step it will be necessary to take a copy of each output .ufs file with clearly, different names.

The main advantage of using the re-start facility, apart from being able to skip one execution of **SATNET**, is that the new sequence starts with the flow-delay curves and simulation profiles from the previous run. In the former sense RESTART is

therefore very similar to the use of UPDATE within **SATNET**, although the use of old simulation profiles is exclusive to RESTART.

If the new trip matrix is not much different from the old then the final flow-delay curves, etc. will not be much different either. Hence by starting with good approximations the overall number of assignment/simulation loops can be sharply reduced.

Section 22.2.2 contains further information on RESTART, including its relationship with other similar “kick-start” techniques.

15.5 Pre-Loading Fixed Flows (The “Plod” Option)

The “pre-load” option was introduced at an early stage of **SATURN** development, somewhat as a short-term measure, to deal with the problems of, say, assigning heavy lorries separately from other vehicles. Indeed that particular application, described in 15.5.1, has been largely superseded by the Multiple User Class Assignment option which is more general and more flexible and generally recommended in preference to PLOD (see 7.3). However, as discussed in 15.5.2 and beyond below, a number of other possible applications have emerged over the years.

In simple terms pre-loaded flows are fixed flows introduced onto the network before any assignment takes place but which contribute to the total flows used to calculate costs (times) in the assignment. They are always defined in units of pcus/hr and have no other properties such as being part of a particular user class or vehicle class. However in terms of calculating their total pcu-hrs etc. it is assumed that their travel times are as defined for user class 1.

Note that, in certain circumstances, pre-loaded flows may contribute to exit and/or entry flows on simulation links (see 15.6.2). For example, if a flow of 100 pcus/hr is preloaded on turn A-B-C but no flow is preloaded onto link A-B then a flow of 100 pcus/hr must be added as a downstream entry flow on A-B.

15.5.1 Pre-loading HGV's

The procedure to be followed with heavies plus cars would be to:

- ◆ Set up a “heavies” network and carry out a full **SATURN** run assigning only a matrix of heavy vehicles.
- ◆ Set up the “normal” network with the previous **demand** (i.e., not actual) flows “pre-loaded” onto the network and treated as fixed flows in the same way that buses are.

The second or “normal” network file would have PLOD = T in &OPTION (and, preferably, the name of the pre-loaded file via PLDFIL) whereas the first would have PLOD = F (the default).

In effect the PLOD option allows the heavy lorries to have the first choice of route and implies that whereas lorries can affect the routes subsequently chosen by the “normal” vehicles, the normal vehicles cannot in turn affect lorries. In some circumstances this may not be a totally unrealistic assumption; however allowing

for interaction in both directions would no doubt be preferable and is provided by Multiple User Class Assignment (Section 7.3).

While the lorry network can have different network properties from the “normal” network, e.g., different link speeds, etc., both networks **MUST** be structurally identical, i.e., have the same nodes, links and turns (unless a text file is used’ see 15.5.4). (N.B. PLOD differs from PASSQ and/or UPDATE in this respect: the PASSQ/UPDATE networks may have a different structure from the main network; see notes 1) in section 17.3.1 and section 15.3 respectively.) Hence, strictly speaking it is not (yet) possible to introduce lorry bans by banning turns or removing links in the lorry network. However lorry bans can in fact be introduced by certain relatively simple tricks.

For example, you can effectively ban lorries from a link by giving that link an extremely high travel time in the lorry network (assuming of course that there are alternative routes available). Banned turns may be introduced by coding them as bus-only turns even if there are no buses; the model response to a bus-only turn is to prevent any elements in the trip matrix - in this case lorries - from using those turns.

Some caution must be exercised when using PLOD so that other forms of fixed vehicles are not loaded twice. For example, bus routes should not be coded as part of the lorry network, only as part of the normal network, since any bus flows in the lorry network will be automatically added as fixed flows to the normal network.

Clearly the same basic procedure is carried out with **any** combination of assigned vehicles, not necessarily just lorries and cars.

15.5.2 Pre-loading Distance Minimisers

Another useful application of the PLOD option is to carry out a separate assignment of a trip matrix of “distance minimisers” whose route choice is, by definition, independent of other trips. How of course one defines such a trip matrix in the first place is entirely up to the user. Again distance minimisers may be treated as a separate user class.

It is also quite feasible to do several stages of pre-loading. For example, you can start with the distance ‘minimisers’, pre-load them onto a lorry network - in which case the output flows would consist of both lorries and distance-minimisers - and then pre-load that network onto a normal network. Clearly some care is called for here to choose the best order and to avoid double counting, etc.

15.5.3 Pre-Load Statistics

The assignment network statistics include totals for any pre-loaded trips separately from the over-all totals, but - as of yet - no comparable breakdown is available within the simulation network.

15.5.4 Pre-Loading from a (Text) Data File

It is also possible to input pre-loaded flows from a text-based data file as opposed to a **SATURN** .ufs file (post version 10.4). For example, if you have extensive bus flows but do not wish to code them as individual routes, only to represent their **total** flow across the network, then it may be done by setting up a text file wherein

each record contains: (a) link identification (in standard or free (CSV) format; see below) followed by (b) the corresponding flow in units of pcu/hr.

Alternatively if the pre-load file and the current file have a **different** network structure and pre-loading from a .ufs file is not permitted (paragraph 4, 15.5.1), then the relevant flows may be pre-loaded by first dumping flows from the pre-load file into a text file; e.g., use **SATDB** (11.0.9).

SATURN differentiates between the two by looking at the extension of the input file: if it is .ufs/ufa/etc. it assumes a **SATURN** file, if not it assumes a text data file. See 14.4.4.

The format of the link identification may either follow standard **SATURN** input data conventions, see, for example, 6.10, with node numbers in fixed columns followed by a (single) link flow or both node numbers and flow may be input totally as “free format” or CSV by setting a parameter PLODFF = T in the network &OPTION data segment (see 6.1). By default PLODFF = F.

Note that pre-loaded “links” should normally include both “roads” and “turns” in a simulation network. Including only “roads” will lead to discontinuities in flows at simulation junctions.

Within free-format text files (PLODFF = T) a further &OPTION parameter PLFF3 = T requires that each input record contains 4 fields – A, B, C and flow. Thus links are distinguished from turns by **always** including an explicit third C-node field which is equal to zero for a link and the turn C-node otherwise.; i.e., A,B,0,link-flow(A,B) as opposed to A,B,C,turn-flow(A,B,C).

Alternatively, if PLFF3 = F, then link records require 3 fields (A and B followed by the flow) whereas turn records require 4 fields in total (A, B, C and flow). By default PLFF3 = F.

For fixed column input (PLODFF = F) PLFF3 does not apply since the fixed data columns used for a C node will simply be blank (or zero) for a link and the flow data is in the same (fixed) columns for both links and turns.

See Section 9.12.3 for suggestions as to how the pre-load facility may be used in combination with the parameter ZILCH to carry out a 100% pre-load.

15.5.5 Pre-Loading Bus (PCU) Flows

As noted in Section 5.5.4 it may be possible / convenient to define bus flows as pre-loaded flows or vice-versa. For example, if you have a very large number of low-frequency bus routes it may be simpler to simply aggregate all their individual flows by link/turn and input them as fixed pre-loaded flows rather than go through the hassle of defining individual routes as per 6.9; the impact on the assignment and (with some reservations) the simulation will be identical.

On the other hand, bus flows may have certain properties that distinguish them from other flows, e.g., bus lanes. Equally coding buses as aggregate fixed flows means that you cannot analyse individual route timings etc.

15.6 Comparing Assigned and Observed Flows: GEH Statistics

15.6.1 General Options

It is possible to obtain a number of goodness-of-fit statistics comparing the modelled flows on both links and turns with observed counts in order to check the performance of the model. This can be carried out in several ways:

The most comprehensive and flexible set of comparisons is available within **P1X**, either under Validation (11.7.1) or **SATLOOK** (11.11.13). In these cases the observed flows are taken from the input .ufs file as originally read as 77777 records in the network .dat file. The modelled flows may be defined in a number of different ways in order to match the precise definition of the counts used; e.g. demand or actual flows may be used (actual probably makes more sense in general), bus flows may be included or excluded, a single user class flow may be selected, etc. etc.

Note that Validation, being newer, provides more options than **SATLOOK**. On the other hand **SATLOOK** is probably easier to run with a key file or as part of an extended batch file.

Alternatively, both counts and flows may be read into **SATDB** and the standard statistical options to compare two data columns invoked. Users may wish to define their own difference measures based on the column- manipulation facilities within **SATDB**. Within **SATDB** the user may select either actual or demand flows as preferred.

Finally **P1X** can also display difference statistics graphically under link annotation. Two standard items are "ABS ERRORS" which is the difference between counts and actual flows and "REL ERRORS" which gives the relative differences as a percentage.

15.6.2 GEH Statistics

With one exception the output comparison statistics are standard and straightforward, the exception being what is referred to as "The GEH statistic". This is a statistic, first suggested to me by Geoff Havers of the Greater London Council, which is useful in comparing two different values of flow on a link, V1 and V2. It is defined by:

$$GEH = \sqrt{(V_2 - V_1)^2 / (0.5(V_1 + V_2))}$$

It may most easily be thought of as the square root of the product of the absolute difference, V2-V1, and the relative difference, (V2-V1)/VBAR where the "average flow" VBAR = 0.5*(V1 + V2).

The reason for introducing such a statistic is the inability of either the absolute difference or the relative difference to cope over a wide range of flows. For example an absolute difference of 100 pcu/h may be considered a big difference if the flows are of the order of 100 pcu/h, but would be totally unimportant for flows of the order of several thousand pcu/h. Equally a 10% error in 100 pcu/h would not be important, whereas a 10% error in, say, 3000 pcu/h might mean the difference between building an extra lane or not.

Generally speaking the GEH parameter is less sensitive to such problems since a modeller would probably feel that an error of 20 in 100 would be roughly as bad as an error of 90 in 2,000, and both would have a GEH statistic of, roughly, 2.

The following table gives an indication of various levels of GEH values, both qualitatively and quantitatively:

Value	Comment	Examples	
GEH = 1.0	"Excellent"	+/- 65 in 4,000	+/- 25 in 500
GEH = 2.0	"Good"	+/- 130 in 4,000	+/- 45 in 500
GEH = 5.0	"Acceptable"	+/- 325 in 4,000	+/- 120 in 500
GEH = 10.0	"Rubbish!"	+/- 650 in 4,000	+/- 250 in 500

Thus, as a rule of thumb, in comparing assigned volumes with observed volumes a GEH parameter of 5 or less would indicate an acceptable fit to a traffic modeller, whether it was a difference of 325 to 4,000 or 120 in 500, while links with GEH parameters greater than 10 would probably require closer attention.

It needs to be noted that the GEH statistic is an "intuitive" and "empirical engineering" measure, not necessarily a measure that a professional statistician would recognise or deign to use. However, it should also be noted that the **square** of the GEH parameter is not unlike the well-used chi-square measure of fit, and would be the same if either V1 or V2 (whichever was the 'observed' flow) were used in the denominator. (One reason for taking the average is to avoid possible problems when either V1 or V2 equals zero.)

It is however not particularly useful to take the comparison too far, particularly when comparing modelled to observed flows, since the sum of the GEH^2 values interpreted as a chi-square statistic will almost certainly indicate that the two are significantly, indeed very highly significantly, different and that therefore the model is 'wrong'. From a pure statistical point of view virtually all transport 'models' are wrong in that they fail to reproduce observations. What a transport modeller wants is a model which, although not strictly correct, is adequate for the uses to which it is applied.

A further distinction between GEH and chi-square is that the latter gives a relatively greater weight to larger differences between flows, for example, to "outliers". For example, errors of 63 and 126 in 1000 pcu/hr give GEH values of (approximately) 2 and 4 but chi-squared values of 4 and 16. GEH effectively says that an error of 126 is "twice as bad" as 63, not four times as bad. These differences are reflected in aggregate measures such as the average of all GEH statistics from a set of counts.

With version 10.1 the GEH statistic comparing two database columns in **SATDB/P1X** may be calculated as an explicit function; see 11.10.8.

15.7 Use of SATURN Outside the U.K.

Although **SATURN** has clearly been set up in the U.K. and with U.K. applications in mind it has been programmed in a perfectly general manner so that with minimal changes it could be applied in other countries, e.g. in Australia and New

Zealand using the NOTUK parameter and in countries where vehicles drive on the right using LEFTDR.

15.7.1 The NOTUK Parameter

Setting NOTUK NE 0 in the &PARAM namelist input causes the model to make a number of assumptions concerning priorities for turns coded with one of the priority markers described in Section 6.4.2 which differ slightly from the assumptions made in the U.K. They are as follows:

- ◆ Opposite right-turning vehicles, for example at traffic signals do **not** interfere with one another, whereas in the U.K. it is assumed that they execute a 'hooked' movement.
- ◆ Right-turning vehicles at traffic signals (i.e. turns coded as X) have priority over left-turning vehicles coming from the opposite direction.

The values allowed for NOTUK are:

- ◆ 0 - neither assumption (the default UK value);
- ◆ 1 - assumption (i) only (as in Australia apart from Victoria);
- ◆ 2 - assumption (ii) only;
- ◆ 3 - both (i) and (ii) (as for Victoria and New Zealand)

The "traditional" (i.e., dating back to the 1970's) default value in **SATURN** is 0 implying that opposing right turns in the UK **do** hook and therefore interfere with one another. However in the 21st Century UK the opposite is almost certainly the norm and, paradoxically, a value of NOTUK = 1 would be recommended.

However, setting NOTUK = 1 on **existing** networks may not be a good idea if a large number of individual turns have been given a Priority Modifier D which reverses the definition of hooked/not hooked (see 6.4.2.7). I.e., if you set NOTUK = 1 but do not change XD to X then all those turns will be assumed to hook.

15.7.2 Right-hand Drive: LEFTDR = F

Although clearly designed for British conditions with drive-on-the-left it is equally easy to use **SATURN** for drive-on-the-right. To invoke drive-on-the-right set the parameter LEFTDR to .FALSE either "universally by default" in SAT10KEY.DAT (Appendix Y) or network-specific (6.3.1). Differences occur in the input in that simulation links need to be input in strictly counter-clockwise order for right-hand drive instead of clockwise.

More serious problems might arise with junction types and/or control strategies which are radically different from those used in the U.K., or - more accurately - cannot be represented properly by **SATURN**.

Output differences include writing "right hand" rather than "left hand" etc. in messages and in annotating on the opposite (i.e. "correct") side of the links in graphical displays (where it is very important to have LEFTDR set correctly).

15.8 Using SATURN as a Conventional Assignment Model

15.8.1 Buffer-only networks

As mentioned in Section 5.2 it is possible in the limit to use **SATURN** as a conventional assignment model by defining a network which consists entirely of a buffer network with no simulation nodes. In such a case one would use **SATNET** to build a network file and **SATEASY** to carry out the assignment. Given that there are no simulation nodes there is no necessity to use the simulation stage **SATSIM** and the assignment obtained from one execution of **SATEASY** is a convergent solution within the limit of the convergence parameters set. Note that one could also use **SATALL** instead of **SATEASY**; it carries out the identical assignment procedures and is recommended.

There are a number of reasons why one might wish to use **SATURN** in this way. For example users might wish to model large-scale interurban networks for which junction modelling is not essential. Another example would be the user who wishes to use the matrix update facilities within **SATURN** without necessarily wishing to define all or part of his network in the detail required by **SATURN**. A third example is the use of **SATURN** purely as a network data base.

The ASCII .dat file necessary to define such a network must commence with the three mandatory input records (OPTION namelist, title and PARAM namelist) as described in Section 6, immediately followed by a buffer network "header" record of a 3 in column 1 and the buffer network description terminated by a '99999 card' as specified in Section 6.6. Node co-ordinates, route flows etc. (optionally) follow. The final card in the file must be another 99999 card.

The use of default speed-flow curves within the 333 records (15.9.5) may be extremely useful in buffer-only networks.

Thus a "typical" file might read:

```
&OPTION
&END
THIS IS A PURE BUFFER NETWORK
&PARAM
BCRP=4.0,
LIST=T,
&END
33333
  3   2   28   56 2500  1   100  3.1
 29   2   21   42 1250  2    90
  2   3   28   56 3750  1   100  3.1
C   2   59   10
C   3   60   10
                                     ...
99999
55555
          ..... Co-ordinate data
99999
99999
(End of file)
```

15.8.2 Converting Simulation Networks to Buffer (SATBUF)

For various applications it is sometimes useful to convert a network with a simulation component (either entirely or in part) into a pure buffer network, e.g. to carry out very simple sensitivity testing or to convert it for use in another suite of programs (so, **SATURN** not good enough for you, eh?) Essentially this requires that link cruise times plus junction delays are converted into the best equivalent buffer speed flow curves which, since they cannot distinguish between different turning movements, must of necessity be suitably weighted averages.

The averaging of delays may be carried out using routines within **SATDB** and data for each simulation link dumped to an ASCII file. A special purpose .bat plus .key file is provided to do this. Type

```
SATBUF net
```

to produce an ASCII file net.buf which contains for each simulation link (A,B) a single record containing:

- ◆ Its A-node
- ◆ Its B-node
- ◆ The average free-flow time (in seconds)
- ◆ The average time at link capacity (in seconds)
- ◆ The distance (in metres)
- ◆ The link capacity (in pcu/hr)
- ◆ The weighted flow-delay power n.

The times above include both cruise time along the link plus a flow-weighted average of the delays to each individual exit turn:

$$d = \sum V_i d_i / \sum V_i$$

where:

d_i = delay for turn i

V_i = simulated (actual) flow for turn i

Thus if you have a simulated right turn with a very long delay but (consequently) a very low flow this will have relatively little effect on the delays which would be modelled in the buffer network (so that in a buffer network representation you could expect to overestimate that particular turning movement).

The capacity is that already calculated for each simulation link (see 8.9.4 for further details) while the flow-delay power n is a weighted sum of individual turns as with delays above.

Both the order and the format of the output variables is the correct order required by buffer network input to **SATNET** (section 6.5). Thus the times, capacity and distance are all output as “integers” although they are calculated as “reals”.

Note that **SATBUF** deals only with simulation links, i.e. the 11111 data input, and that users must decide for themselves how to deal with simulation centroid connectors - the 22222 data inputs. One very simple solution, which implies using an editor, is to edit the 22222 records by

- ◆ inserting a C in every column 1
- ◆ deleting all records from column 11 onwards.

This has the effect of producing a correctly formatted set of buffer link records with the zone as A-node and the first simulation node entered as its B-node. Whether this is a good way to re-code centroid connectors is another question.

N.B. If DUTCH = T in the network being “bufferised” then an alternative version of the batch file may be run thus:

SATBUF net DUTCH

in which case the new link A-nodes and B-nodes will appear in column blocks of 10, not 5 in the output file net.buf. (Added in version 10.9.15)

15.8.3 SATCCS: Converting Simulation Centroid Connectors to Buffer

An extra batch file introduced in Release 11.1, code-named SATCCS, performs essentially the same job as SATBUF but operates on simulation centroid connectors instead of simulation links. Thus the command:

SATCCS net

creates an output text file net.map with link data in the 33333 format for simulation centroid connectors **only**.

Thus if a zone Z is connected to simulation link A-B then there will be a record A-Z and another from Z-B with appropriate distances, times, etc. etc.

The intention would normally be that the file map.dat would be included within the 33333 section of a buffer-only network .dat file (either verbatim within the 33333 data segment or, perhaps preferably, as a \$INCLUDE file. See 15.1.5 for such an application.

The procedure uses the “dump map links” option within **P1X** (see 11.4.2.3) but in a purely off-line batch mode and with only simulation centroid connectors selected. No special KEY file is required (unlike SATBUF).

15.9 Converting Conventional Speed-Flow Curves into SATURN Curves

15.9.1 General Principles

It is very often handy for users with existing networks coded in conventional detail to convert their networks into a **SATURN** network by stages. Thus the first step would be to code the existing network as a buffer-only network (presumably using a computer program to carry out the necessary changes in format) with no simulation network, as described in Section 15.8. Preliminary tests may now be carried out with very little coding effort.

Certain problems may arise in converting existing speed-flow curves into the flow-delay relationship as specified by **SATURN** for its buffer network, i.e., an nth order power law for flows less than capacity and a linear relationship for flows above capacity (see Section 5.4 and equation (5.1)). These problems may concern not only the calculation of n – dealt with below – but also problems with the interpretation of parameters such as capacity.

We consider first the range of flows less than capacity, $V < C$. Clearly if the existing curves are already in the form of a power law then the problems here are minimal; the user must simply ensure that the required value of n is set in BCRP if it is constant for all links or is input for each individual link.

If however the existing curves are of a different form it will be necessary to define power-law curves which, in some sense, give a “best fit” to the existing curves. There are many ways in which this can be done, depending both on the definition of “best fit” as well as on the shape of the existing curves.

Different countries may well have different recommended forms. We illustrate here one method which may be used to convert curves of the form recommended by the UK Department of Transport, currently referred to as DfT, but for historical reasons also referred to as DTp.

15.9.2 DFT/DTp Advice Note 1A

DTp (“Advice Note 1A”) recommended curves have the following form:

$$t(V) = d / S(V)$$

$$S(V) = \begin{cases} S_0 & V \leq F \\ S_1 + (S_1 - S_0)(V - F)/(C - F) & F < V \leq C \\ S_1 / (1 + S_1(V - C)/8dC) & V > C \end{cases}$$

Where:

- t is the link time (in hours),
- d is the link distance (in kilometres),
- S is the link speed (in kph),
- V is the link flow (in PCU per hour),
- S_0 is the “free flow” speed,
- S_1 is the speed at capacity,
- F is the maximum flow at which free-flow conditions hold
- C is the capacity

We wish to fit the above curve, in the range $V < C$, with a function:

$$t = t_0 + aV^n$$

The three unknowns, t_0 , a and n, are fitted from the following constraints:

- 1) Free flow times must be the same (hence $t_0 = d/S_0$);

- 2) Capacity times must be identical, and
- 3) The “average” travel times must be the same.

Condition (3) is the critical one for determining n . We define the average travel time to be:

$$\int_0^C t(v) dv / C$$

Hence for **SATURN** the average time is given by:

$$\langle t \rangle = t_0 + aC^{n+1} / (n+1)C$$

$$= t_0 + aC^n / n+1$$

$$= t_0 + (t(C) - t_0) / n+1$$

Integration of the DTp curve gives:

$$\langle t \rangle = t_0 + (1 - F/C) (\ln(S_0/S_1) / (1/t_0 - 1/t_c) - t_0)$$

Hence:

$$n = (r - 1) / ((1 - F/C) * (r \ln r / (r - 1) - 1)) - 1$$

$$\text{where } r = S_0 / S_1 = t_c / t_0$$

A section of FORTRAN code which does the above job is given below:

```
R = S0/S1
XN = 0.0
IF (R.GT.1.0) THEN
  XBOT = (1.0 - F/C) * (R*ALOG(R)/(R - 1.0) - 1.0)
  IF (XBOT.NE.0.0) XN = ((R - 1.0)/XBOT) - 1.0
END IF
```

and the following table gives values of n for ‘typical’ DTp parameters (where the speeds are in kph and the flows/capacities in pcu/hr):

S_0	S_1	F	C	N
90	76	3600	5200	5.89
79	70	3200	4800	5.25
70	57	400	1800	1.76
63	55	400	1400	1.93
50	50	0	600	0.00
80	66	3400	4800	6.33
65	56	2800	4400	4.79

S ₀	S ₁	F	C	N
50	30	1200	2200	4.29
45	25	500	1000	3.96
35	25	350	600	4.40
25	15	250	500	3.81
67	47	0	4000	1.27
61	27	0	3400	1.72

For the range of flows above capacity, DTp and **SATURN** curves both have a linear relationship, although the slope of the curve in **SATURN** is determined from the length of the time period simulated (parameter LTP) while that for the DTp curve is set by the parameter '8' in the above equation which has units of 1/hours. To set up the same slope in **SATURN** it is therefore necessary to set LTP = 15, i.e., 1/4 of an hour since the slope equals $0.5 \cdot LTP$.

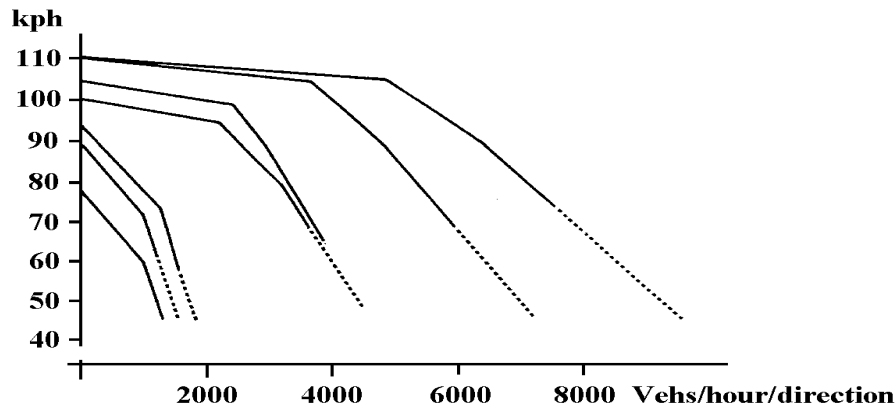
Having set up the buffer network the user may now begin to code parts of the network in the format and detail required for a simulation network, starting with those nodes where the extra detail is most required and working outward as far as may be required. Since any node that appears in both the simulation and buffer networks is ignored in the buffer network nodes may be coded as simulation nodes without having to remove them from the coded buffer network. One advantage of coding **SATURN** networks in this way is that the user gains coding experience by degrees and thereby makes fewer mistakes overall.

Note that in following this procedure the nodes which lie on the boundary between the simulation and buffer networks at any stage **MUST** be included as external nodes in the simulation network unless one uses the AUTOX facility as described in Section 15.12.

15.9.3 COBA 10 Speed-Flow Curves

The form of the DfT-recommended speed-flow curves was replaced in the late 1980's by the so-called "COBA-10 curves" with two sloping linear segments as opposed to Advice Note 1A above which had a flat segment followed by a linear slope. Figure 15.1 below illustrates the new form for flow V less than capacity C. They are still in use to the present day (2007) although the specific numerical values for individual curves are out-of-date.

N.B. The "x-axis" or flow-axis in Fig. 15.1 is specified in units of vehs/hour whereas **SATURN** (see 15.17.1) generally works in terms of PCUs/hr; some conversion may therefore be required if one wishes to **fully** translate COBA curves for use in **SATURN**. See 15.9.4 below.

Figure 15.1 - COBA 10 speed vrs flow curves

The following equation describes the relationship:

$$S(V) = \begin{cases} S_0 + (S_1 - S_0) * (V / F) & V \leq F \\ S_1 + (S_2 - S_1)(V - F) / (C - F) & F < V \leq C \\ S_2 / (1 + S_2(V - C) / 8dC) & V > C \end{cases}$$

Where:

S_0 is the free flow speed

S_1 is the “intermediate” break point speed

S_2 is the speed at capacity C

A “best-fit” value of the power n may then be determined by the equation:

$$n = (R_1 * R_2 - 1) / (B_1 + B_2 - 1) - 1$$

where:

$$B_1 = \frac{(F / C) R_1 \log R_1}{(R_1 - 1)}$$

$$B_2 = \frac{(1 - F / C) R_1 * R_2 \log R_2}{(R_2 - 1)}$$

$$R_1 = S_0 / S_1$$

$$R_2 = S_1 / S_2$$

$$\text{N.B } \lim_{R \rightarrow 1} (R \log R) / (R - 1) = 1$$

and “log” above refers to the “natural log”

Our thanks are due to Yazid Arezki for working out the above formula, thus confirming earlier numerical values calculated by Devon County Council.

In previous versions of the manual, a set of calculated values of n for “standard” UK road classifications were provided in a Table (often referred to as ‘Table 15.9’) with a shorthand description of each road type.

With the release of 10.9.24, the table was **withdrawn** as its inclusion was only intended to illustrate a range of ‘typical’ values of ‘ N ’ which may result, using the formulae above, from piece-wise linear curves. The values used in the table were originally taken from COBA data sets of circa. 1990 and were not, in any sense, **recommended** as up-to-date values for different road types. In practice however, users were applying the Table 15.9 curves without undertaking the necessary critical review required for their specific application.

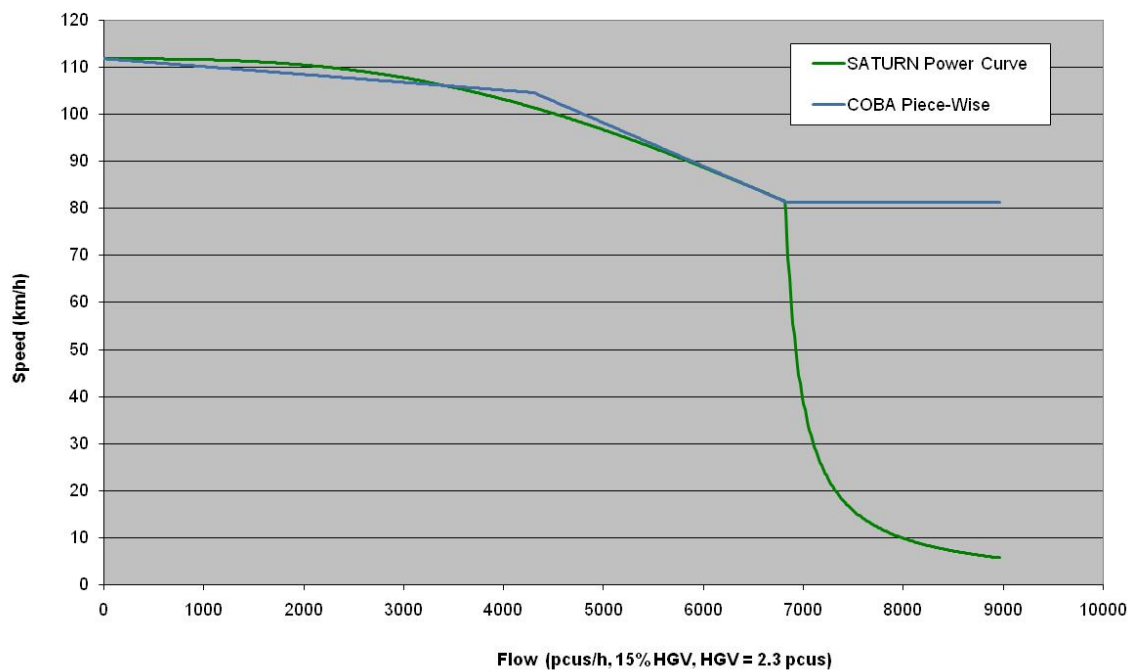
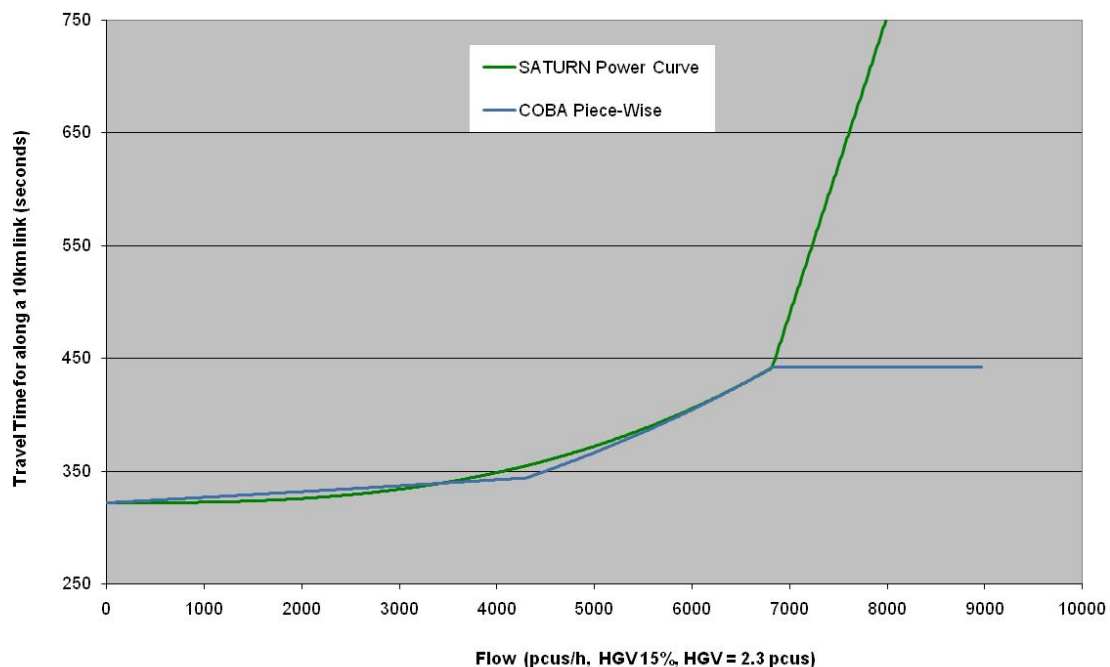
To assist users, an illustrative comparison of a ‘typical’ COBA piece-wise curve and the equivalent **SATURN** Power curve is provided overleaf in Figure 15.2; The data parameters used (listed below) and the best-fit value of N are for a (nominal) dual 3-lane motorway with 15% HGVs. Figure 15.3 re-plots the same data as delays versus flows (which is the form in which it is applied in assignment models).

Further advice is provided to assist in converting curves into a form suitable for **SATURN** in the following section.

The equivalent **SATURN** parameters for the curves **illustrated** above (with various assumptions on the other COBA parameters required) are shown below.

S_0	S_1	S_2	F	C	N	Description
111.8	104.6	81.4	4410	6990	2.80	D3M

Note: speeds S_0 , S_1 and S_2 in km/h whilst breakpoint F and capacity flow C are in pcus/h

Figure 15.2 –COBA11 Piece-Wise v SATURN Power-Based Speed-Flow Curve (15% HGV)**Figure 15.3 – SATURN Dual 3-Lane Motorway Flow-Delay Curve**

15.9.4 Conversion of existing speed-flow curves into SATURN

Extreme care should be exercised when speed-flow curves which have been developed in a different context are “translated” into **SATURN** speed-flow curves. We note, in particular, problems which have arisen in using COBA-10 curves,

whose basic application is within an economic evaluation package, **not** within an assignment model.

Certain of these problems apply more to the use of link capacity-restraint curves within the simulation network (6.4.12 and 8.4.4); most apply to both simulation and buffer networks equally.

The first concerns the question as to whether or not the recommended “capacity” for a link A-B takes into account the existence of (a) intermediate junctions between A and B or (b) the junction at B. In a buffer network both (a) and (b) **should** be included in the capacity used by **SATURN**; in a simulation network (a) should be included but not (b) which is otherwise considered by the simulation of junction B. The problem is therefore one of either double-counting or “zero-counting”. See also 6.4.12.1.

The second problem is one of units. If, as in COBA-10, capacities are normally specified in units of vehicles per hour there may be an assumed percentage of HGV (or other) vehicles within the “vehicles”. Thus, given a link with a flow of 1,000 vehicles/hr of which 15% are HGVs (150 /hr) for which one would wish to attribute a PCU factor of 2.0 PCUs/HGV, the equivalent flow in terms of PCUs/hr would be 1,150.

The third question is what happens to speeds in excess of “capacity”. COBA curves, for example, may assume that speeds above capacity do not reduce but continue to be fixed at their capacity speed. **SATURN** assumes that flows in excess of capacity lead to linearly increasing queues with a consequent linear increase in travel time (/reduction in effective speed) as given in equation (5.1b). Users need to bear this in mind in specifying link capacities (for both simulation and buffer links).

In all three cases it is the responsibility of the user to decide how and by how much to compensate for these effects before using these curves within **SATURN**.

15.9.5 Default Speed-Flow Curves

It is possible to define the speed-flow relationships on buffer links by defining “default” speed flow curve parameters which apply to all buffer links which have the same capacity index. To use this option within a network data file input to **SATNET** you must:

- ◆ Define a set of default speed flow records within the ‘33333’ data records, identified by a ‘D’ in column 1 and with entries for free-flow speed, speed at capacity, capacity, the power ‘n’ and a (non-zero) capacity index in the “normal” fixed columns; see 6.6 for the detailed format;
- ◆ For each buffer link to which the above parameters apply leave blank (or code as zero) the free-flow speed/time, capacity speed/time, capacity and power but include the distance and capacity index. The program then substitutes the default speeds, etc. for the missing records. (In fact it is not even necessary to code the distance if the SHANDY option is in effect; see 15.10.) N.B. It is necessary to leave **all** four of the above entry fields blank/zero; if one of them is included then it assumed that the other entries of zero are all valid entries and the default option is **not** applied.

Thus all links with the same capacity index will have an identical speed-flow curve plus capacity. Note that the actual times need not be identical since these will depend as well on the distance which is coded separately for each link.

One advantage of this option is that you can make “universal” changes to the speed-flow parameters for a set of links by simply changing a single record rather than several. The option should also be extremely useful for networks which are defined by graphical input in some form; here link distances can be calculated from node co-ordinates so that the only input information required from the user (apart from whether a link is one-way or two-way) is an index which determines the remaining parameters.

An example of an input data file using these conventions is illustrated below where “default” indices 1 to 14 are equivalent to the “typical” DfT parameters defined above. Thus link 6-7 has a length of 90 metres but a capacity of 1400, a free-flow speed of 63 kph, etc. as taken from the previous “D” record for capacity index 4.

33333				
D	90	76	5200	5.9 1
D	79	70	4800	5.2 2
D	70	57	1800	1.7 3
D	63	55	1400	1.9 4
D	50	50	600	0.0 5
D	80	66	4800	6.3 6
D	65	56	4400	4.7 7
D	50	30	2200	4.2 8
D	45	25	1000	3.9 9
D	35	25	600	4.4 10
D	25	15	500	3.8 11
D	67	47	4000	1.2 12
D	61	27	3400	1.7 13
D	56	20	1800	1.9 14
....				
6	7		90	4

Further Notes:

- 1) The “D” records can appear anywhere within the 33333 records and can be applied to buffer links that precede them.
- 2) By default (see note 4) below) the five required input data fields (free-flow speed, speed at capacity, capacity, the power ‘n’ and capacity index) must appear within the **same** fixed columns as “normal” buffer links; e.g., the free-flow speed in columns 11-15. But, N.B., note that the required columns differ under DUTCH = T; see 15.20.
- 3) Note that unlike standard buffer records where either speeds or times may be used, the default speed-flow curves are only based on speeds. It is assumed therefore that buffer records which make use of speed-flow curves have an ‘S’ in column 29 (39 under DUTCH = T).
- 4) Problems associated with fixed columns and differences between DUTCH = T or F may be eliminated by setting a parameter DCSV = T under &PARAM in the network .dat file, in which case the 5 necessary fields may appear in free format following the D in column 1. I.e., they must appear in the correct order and be separated by either spaces and/or commas.

- 3) It is quite possible that users would wish to set up curves with characteristics identical except for the link capacity to represent say, dual 2 and dual 3 roads with identical speeds, in which case distinct capacity indices should be used for different lanes. The requirement for link rather than lane capacities should be noted.
- 4) D records are good candidates for inclusion under \$INCLUDE, see 15.30, such that a standard set of default speed flow curves may be recorded in a single file and applied to a wide range of networks.
- 5) Default speed-flow curves may also be applied to simulation links where record 2B (see 6.4.1) excludes any time/speed and capacity data but refers instead to a capacity index which, as with buffer links, defines the link speed-flow curve.

15.9.6 Default Speed-Flow Curves: COBA-10 Formats

An option added in release 10.7 permits default speed-flow curves to be defined directly in terms of COBA-10 speeds and flows such that the best-fit value of the power n is calculated by **SATNET** rather than being input directly by the user.

To invoke this option the default speed-flow records must be altered as follows:

- a) Write 'COBA' in cols. 36-40 (in place of N) (46-50 under DUTCH = T)
- b) Write the speed at the "breakpoint" S_1 in cols. 46-50 (56-60 under DUTCH = T)
- c) Write the breakpoint flow F in cols. 51-55 (61-65 under DUTCH = T)

The calculation of n then follows the equations as given in 15.9.3 where the additional parameters S_0 , S_2 and C as given within the "normal" 33333 fields; see 6.6.

For the time being the option to directly calculate n from COBA-10 curves **only** applies to Default speed-flow curves within the 33333 data records, **not** to individual link records. However, there is no reason why it should not be extended to individual records and, if no problems arise with the above method, it will no doubt be included in the next release.

15.10 The use of Crow-Fly Distances (The SHANDY Option)

15.10.1 General Principles

The SHANDY option (set SHANDY = .TRUE. in the input network .dat file) carries out the following two steps for every input distance for either a simulation or a buffer link:

- ◆ If a positive value has been input it checks this against the crow-fly distance calculated from the input XY co-ordinates and prints a warning message (WARNING 35) if they differ by more than 10 metres in absolute terms AND by more than 5% in relative terms.

- ◆ If a zero (or blank) value has been input it substitutes the crow-fly distance calculated from the input XY co-ordinates and prints a warning message (WARNING 25).

Clearly these steps are only carried out for links where both the A-node and the B-node have been correctly assigned X,Y co-ordinates.

The option works by “pre-reading” the co-ordinate data under the 55555 cards before returning to read the simulation and/or buffer link records. Thus no “interpolated” co-ordinates are available at this stage. If there are no co-ordinates input then the option is cancelled.

In addition, if a GIS file is defined in the network data file (via FILGIS) and that file contains curved link data under 77777 then the crow-fly distances as used to compare against input link distances (SHANDY = T) are calculated point-by-point along the curved links rather than end-to-end directly. See Appendix Z.

Note that this option may be usefully combined with the default speed-flow curve facility described in Section 15.9.5 since the new distance is set BEFORE the speeds are substituted. Thus the free-flow time is obtained from a crow-fly distance divided by the free-flow speed. At a minimum therefore a buffer link record need only contain an A-node, B-node and a capacity index.

It is also an integral part of the **PMAKE** network building options (see section 17) when new links are created.

A summary table comparing actual and crow-fly distances is included near the end of the line printer output file from **SATNET**.

15.10.2 Correcting XYUNIT

In addition an estimate is made of the “correct” value of XYUNIT by comparing crow-fly distances as calculated from the node co-ordinates with the input distances on the .dat file and printed near the end of the .lpn file. If, for example, the crow-fly distances are consistently around 10 times shorter than the coded distances then it is presumed that XYUNIT should be 10 times greater.

15.10.3 CROWCC: Zero Distance Buffer Centroid Connectors

The above rule for replacing an input buffer distance of zero by the crow-fly value traditionally applied to **both** real buffer links and buffer centroid connectors. However, while it may make sense to have a positive distance for “real” links, it may be quite legitimate to have centroid connectors which are purely nominal and therefore have zero distance (plus, presumably, zero time).

An option introduced in version 10.7 allows users the choice as to whether or not buffer centroid connectors may be assigned a distance of zero. Thus, if CROWCC = T (set in &PARAM of a network .dat file) and SHANDY = T a crow-fly distance replaces buffer centroid connectors with an input value of zero. If CROWCC = F an input distance of zero is accepted.

For most users CROWCC = F is likely to be the preferred option. However, the default option prior to 10.7 was effectively T so, for upwards compatibility, the

default value of CROWCC was set to T at that point in time. Subsequently, release 10.9, the default was changed to F.

We further note that setting CROWCC = T may have certain potentially negative consequences for running SATTUBA; see 15.41.5.

15.11 Coding Combined Buffer and Simulation Networks

Problems may arise in coding a network which includes both a simulation and a buffer network, in particular at the interface between the two. The following points may help.

- 1) The simulation network is coded in the normal way with external nodes defined at the edge of the network - either "explicitly" within the 11111 data records or "implicitly" via AUTOX. If the external nodes represent "cordon" or "stub" nodes where the network terminates then they would normally be connected to "cordon" zones, i.e., zones representing all trips entering or leaving the network at these points. These zones should then be included within the 22222 data records (or implicitly via AUTOZ). The precise points of zonal connection will be at the external nodes as described in 16.6.2.

(Alternatively the external connection to the zone may be made via an external simulation link plus an isolated buffer node which is coded under the 33333 data records as described in 16.6.3. However this method is generally **not** recommended as it leads, inter alia, to the same problems with U-turns as described in Section 16.6.4 and 18.9.2.)

However, external simulation nodes may **also** represent points where the simulation network connects continuously into the buffer network and, in this situation, origin/destination zones at the boundary may be connected **either** via the buffer or the simulation network. However, in the latter case, the effect may not be what was desired.

For example, consider the following schematic network where E represents both an external simulation node and a node which is part of the buffer network, S is an internal simulation node and B represents one or more nodes in the buffer network connected to E.

B ----- E ----- S

Let Z be a zone that is connected only via a 22222 record to the (two-way) simulation link E-S and not at all via a 33333 buffer record. In this case trips from the (origin) zone Z can **only** enter the network at E in the direction E-S and, similarly, exit to the (destination) zone Z at E having come from S. They cannot go directly to / come directly from B.

By contrast, if Z were connected to E as a 33333 buffer centroid connector, then the origin trips would enter at E and have an immediate choice between **both** B and S. Equally, the destination trips to Z would exit from E having come from either B or S. In general terms the latter is **probably** what the user would prefer, in which case it is therefore better to define the centroid connector from Z as a buffer connection to E rather than as a simulation connection to E-S; i.e., it should be included within the '33333' cards rather than the '22222' cards described in 6.5 and 6.6.



- 2) The external simulation nodes must also be included in the buffer network with their “buffer-only” connections - i.e., those links to or from other nodes in the buffer network. Thus links such as E-B above would be included under 33333.
- 3) In constructing the joint buffer/simulation network **SATNET** ignores an input buffer link if either of the nodes has already been defined in the simulation network unless both are external simulation nodes. This means that a user progressively re-defining a section of a large network as a simulation network does not need to remove simulation links from the buffer network input. However some care needs to be exercised here that all “inner” nodes have indeed be defined as simulation nodes since otherwise spurious buffer links may creep into the middle of the simulation network.
- 4) On the other hand the “data” on an ignored buffer link, e.g., the time and distance, is not totally ignored in that it is compared to the comparable simulation data in order to check for self-consistency. In addition the buffer link data may be used to supplement the simulation data as explained further in Sections 6.6, 15.13 and 15.14.
- 5) We also note that problems may occur due to U-turns from the simulation network at the simulation/buffer boundary as described in detail in Section 18.9.

15.12 Automatic Network Coding (The AUTOX and AUTOZ Options)

The AUTOX and AUTOZ options are essentially labour-saving devices which remove the necessity for the user to code external simulation nodes explicitly or to code zones at external simulation nodes which are cordon points.

Under AUTOX all nodes defined as simulation A-nodes (i.e., in cols. 5-10 of Card Type 2 (see Section 6.4) but not explicitly defined as simulation nodes themselves are automatically assumed to be external simulation nodes. Thus if node 99 were defined as an A-node as part of the definition of node 22 and not defined elsewhere then node 99 would be added as an external node with node 22 as an A-node (as well as being connected to any other nodes where it was included as an A-node).

The properties of the link from 22 to 99 are inferred from data coded for 22; thus the travel time and distance are the same as those coded for link 99 - 22 (but with default values of 100 metres and 7 seconds if 99-22 had zero capacity), while if none of the turning movements coded at node 22 were into link 22-99 it is assumed that the direction 22-99 does not exist. If however link 22-99 were included as part of the buffer network definition (Section 6-6) then its time and distance as coded there will be used in preference to any default values as described above.

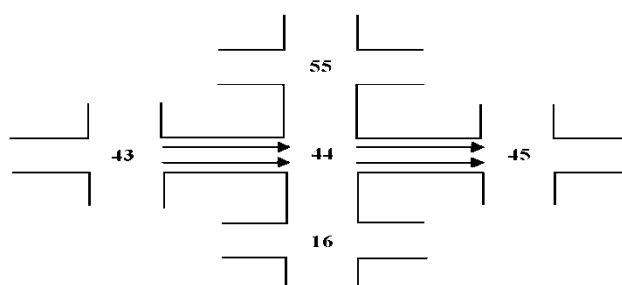
Alternatively, since release 11.3.2, if default values are required (i.e., the 100 metres and 7 seconds, as above) and SHANDY = T then the default distance is calculated as the crow-fly distance and the time is calculated assuming a default cruise speed of 51.42 KPH (32 mph).

It should be stressed that the AUTOX option can be somewhat dangerous to use in that punching errors may go undetected and lead to extra external nodes being

erroneously added. Its use is recommended for very simple networks, for example a network consisting of a single simulation node connected to 'n' uncoded external nodes, set up to simulate an n-way junction in isolation, an example of which is given below, or for networks set up from recoding existing buffer networks or when coding using PMAKE (Section 18).

The AUTOZ option removes the need to explicitly define simulation centroid connectors (6.5) to external simulation nodes by automatically attaching centroid connectors to every link terminating at an external simulation node and assuming that the zone has the same number as the external node. Thus, in the above example where node 99 is an external node connected to internal simulation node 22, a zone numbered 99 would be created, attached to link 99-22 at node 99 in exactly the same way as if a record '99 99 22' were included in the '22222 data' as described under Section 6.5. When using AUTOZ all connections as defined under 22222 should be internal connections, otherwise there will be duplication, and AUTOZ should only be invoked when ALL external nodes are pure cordon points, not when they are links between the simulation and buffer networks. In effect this restricts the AUTOZ option to pure simulation networks without a buffer network.

AUTOX and AUTOZ can both be selected at the same time - and in fact the most useful case for applying both is the case of coding a single simulation junction as they remove the need to code ANY external simulation nodes or zones. An example of coding a 4-way junction, node 44 (as illustrated in Section 16.1), is given in full below. The coding implies that nodes 43, 55, 45 and 16 are external nodes, each one of which is connected to an external zone, also numbered 43, 55, 45 and 16.



Note that the AUTOX option infers that the link 44-43 does not exist (i.e., is one-way from 43 to 44) since there are no turns coded as entering it, and that the time and distance on link 44-45 will be 7 seconds and 100 metres. Equally under AUTOZ zone 43 is entry (or origin) only while zone 45 is exit (or destination) only.

```
&OPTION
&END
NODE 44 CODED ALL ITS OWN
&PARAM
  AUTOX=T,
  AUTOZ=T,
&END
11111
  44    4    3    4    61    85    0    45
      45    0    0    0
      16    2    25   200    0    1700  1  1 1600X 2  2
```

43	2	40	300	1400	1	1	3000	1	2	1200	2	2
55	2	25	220	1400	1	1	2800	1	2			
	19	4	6	43	55	43	45	43	16			
	10	7	4	43	55	16	45					
	25	0	4	16	55	55	0					
	15	5	4	16	55	55	45					
99999												
99999												

15.13 Supplementary Data for Simulation Links Using Buffer Network Inputs

In general all the necessary data for links in the simulation network is defined within the '11111 data cards' described in 6.4; e.g., the link travel time, link distance and number of lanes. It is however possible to use the '33333 data cards' to define extra simulation link data which is not required by the simulation proper but which might be useful under other circumstances. One example of this is the link capacity index which is used to distinguish certain "classes" of links in summary statistics. If a link A-B is included in the buffer network data with a capacity index of, say, 5 but was previously defined as a simulation link, the capacity index of 5 is assumed to apply as well to the simulation link A-B. Using the BEAKER option - see 6.3.1 - the index may also be associated with turns out of A-B; setting BEAKER to .TRUE. is highly recommended.

Similarly any extra "KNOBS" data defined for duplicates of simulation links are also assumed to apply to those links. See Section 15.14.

A further important application concerns "external simulation links", i.e., the simulation link from an internal simulation node A to an external simulation node B. By definition the travel time on the "in-bound" link B-A is fixed, being a simulation link, with - in effect - infinite capacity; any additional delays or capacity restraint on that link are associated with turning movements at A. On the other hand assuming a fixed travel time and infinite capacity on the "out-bound" direction A-B would not be entirely realistic since turning movements at B are not included in the simulation.

It is thus possible to define flow-delay/capacity-restraint relationships on out-bound external simulation links such as A-B above using exactly the same form of link flow-delay curve as is applied to buffer links - see Section 5.4. In order to do so the user must include A-B within the "33333 data cards".

Note that for external links connected directly to cordon zones there is perhaps not much point in worrying about flow-delay since all trips go to the external zone regardless of conditions on the link. However the effect can be important on external links between the simulation and buffer networks, as otherwise it could lead to a situation where there is (effective) capacity restraint in the simulation network and in the buffer network but not at their interface.

In addition if the AUTOX option is used to define external simulation nodes and links - see 15.12 - and the link in question is 1-way outbound (A to B in the above example) so that times and distances are given default values then these default values are over-ridden by any data defined under the 33333 records.

15.14 Extra Link Data (Knobs)

15.14.1 Introduction to Knobs

SATURN allows a variable number of additional data items - referred to as “knobs” - to be input for each link (buffer or simulation) using the ‘33333’ data records (Section 6.6) and/or separate input files to **SATNET**. The data is then stored on the **SATURN** UF files and may, for example, be later displayed using **SATDB** for alpha-numeric output or **P1X** for graphical output.

Knobs, particularly post **SATURN** 10.3, have a number of possible applications. Thus they may be used as components of generalised costs, in particular as tolls, or they may be used to define extra travel times or delays to bus services (15.44). These applications are described below.

Alternatively they may be used to store network data which has no direct impact on traffic assignment, in which case **SATURN** is being used primarily as a network data base - described next.

15.14.2 Data-Base Applications

There are many possible applications of such a data-base. For example one might store the date at which a link was last re-surfaced and thereby produce plots of all links re-surfaced in a given year or range of years using the SELECT facility in **P1X**; equally one might store accident statistics for links. Indeed it is now quite feasible to use **SATURN** purely as a network data-base by simply building a network in which all the “standard” link variables such as time, etc. are ignored and concentrating only on the “extra” data items. From the network build program **SATNET** one could go directly to the display programs **SATDB** and **P1X**.

Once input certain basic algebraic manipulations may also be performed on the data using **SATDB**. For example, if you input accident statistics and calculate link flows you could then calculate and analyse accidents per vehicle.

It is hoped that this facility will encourage other types of **SATURN** users apart from traffic engineers. For example it might allow identical networks to be used for traffic analysis and the analysis of accidents rather, as often seems to be the case with Local Authorities, for two different groups to set up different networks for the same area.

15.14.3 Using Knobs within Generalised Costs

Section 7.11.2 and equation (7.43) describe how the generalised cost of travel as used for traffic assignment may be defined as a linear combination of time, distance and one or more knob data sets. The relative weights are set by PPM, PPK and knob-specific weights specified within the 88888 record set (6.11)

As noted in 7.11.2 **SATURN** makes no further assumption as to what these extra costs are really representing. They might, for example, represent nominal time penalties in units of seconds associated with following a non-signposted route. However the nominal charges will not be included in network statistics of total pcu-hrs.

Note that it is possible to have **negative** values for Knobs data which contribute to generalised cost but only if the **total** link fixed cost does not go negative. See 7.11.2. Negative Knobs values should be used with caution although they may sometimes be useful, for example, to make certain links more attractive to traffic. The weighting coefficient for KNOBS data defined within the 88888 data set (6.11) is defined in the “normal” way as a positive number; i.e., it is not possible to create a negative cost by having positive data with a negative weight.

Note that items of Knobs data which do **not** contribute to link generalised costs (e.g., for applications as described in 15.14.2) should have their 88888 weights input as **zero** (or blank) to avoid being confused with, e.g., tolls.

15.14.4 Using Knobs to Set Tolls (Road Charges)

A particular example of a knob field used to define generalised cost is when the field directly represents monetary charges - tolls. As noted in section 6.11 tolls are indicated by including either a \$ or & symbol in the relevant columns of the 88888 records for that field. If the remaining columns are blank then the assumption is that the knob entry is the “true” charge per link in units of pence but if a numerical factor (apart from 1.0) is also included then the knob entry is factored by that amount. This allows the user to define tolls in purely nominal units, say 1.0 for all links, and then let the 88888 records define the specific toll.

In addition knobs which are explicitly defined as tolls, as opposed to the less well specified effects under 15.14.12, are included in the output network statistics from the assignment which report the total revenue generated by tolls in the same way that total pcu-hrs and total pcu-kms are reported.

For further details as to how tolls are handled within **SATURN** please see Section 20.3.

15.14.5 Creating Knobs Data

In order to use this facility the user must first define the number of data items to be input, the &PARAM namelist parameter KNOBS. Secondly, the link data itself must be input to **SATNET** and that in turn may be in one of three forms:

- 1) as an additional second record for each buffer link with the required number of data fields up to a maximum of 8; see Section 6.6.
- 2) as added data items at the end of the first (and only) buffer link data records;
- 3) as a separate free-standing input file (FILKNB).

Option 3), an external ascii file, is **highly** recommended for ease of use and for avoiding possible errors.

The parameter KONAL (Knobs ON A Line) distinguishes between (i) and (ii): KONAL = F and T respectively. Option (iii) is only used if a file is nominated by the character variable FILKNB (or KNBFIL). If FILKNB is set it is assumed that **no** Knobs data appears in the network .dat file itself (and KONAL is irrelevant).

Note that under (i) the extra record may be entirely blank, in which case it is read as a string of zeros. See 15.29. Equally blank inputs under (ii) are also interpreted as zero's.

15.14.5.1 External KNOBS data files (FILKNB)

The designated file FILKNB (normally) has a standard **SATURN** format (e.g., as for input counts, section 6.10) where each record contains the link/turn identification in fixed column formats in columns 1-15 (1-30 under DUTCH) followed by the knobs data for that link in, essentially, free format, e.g., comma separated.

However, post release 11.2.4, the data records in a KNOBS file may be **entirely** free format (e.g., CSV) by setting an &PARAM parameter FREEKN = T in the network .dat file. In this case the link/turn numbers A, B and/or C do **not** need to be in fixed columns but free format. Note that a third node C must **always** be explicitly included for links either as 0 or, in the case of CSV by “,”

15.14.5.2 KNOBS Data on Centroid Connectors

There is an important distinction between data input “internally” under options (i) and (ii) above and “externally” under (iii). That is that the internal 33333 data may only be defined for **road** links (whether in the simulation or buffer networks) plus buffer simulation connectors whereas data input in an external file may also be defined for **all** components within the **SATURN** assignment networks, e.g., turns (by including 3 nodes) and simulation centroid connectors as well (defined by including a ‘C’ in columns 1, 6 or 11 to identify the zone (columns 1, 11 or 21 under DUTCH)).

Thus to define an outbound centroid connector from a zone Z to node A – where A is either a buffer node or an external simulation node – enter Z in columns 2-5 with a C in column 1 and A in columns 6-10. Reverse the two fields for an inbound centroid connector.

Note that, post 11.1, the requirement to identify zones by a C in an appropriate column may be relaxed by the use of NO333C = T whereby any input node number which is less than or equal MAXZN is assumed to be a zone whether or not a C has been included. This should make it easier to create KNOB files using external packages – but clearly may create problems if zone and node numbers overlap.

Centroid connectors to/from internal simulation links are more complicated and users are advised to consider using Wildcard entries as described below which only require the zone name (plus C) in an appropriate field. Otherwise, to define an outbound centroid connector from Z to link (A,B), enter C+Z in columns 1-5, A in columns 6-10 and B in columns 11-15. For an inbound centroid connector from (A,B) to Z enter A in columns 1-5, B in columns 6-10 and C+Z in columns 11-15.

Post 10.9.5 the above rule has been relaxed so that an outbound centroid connector from zone Z to link (A,B) may be defined by entering Z in columns 1-5 and B **only** in columns 6-10. This, after all, is how the centroid connector appears on the network plots: a dashed line from Z to B. If there is then only one possible link A,B which is so connected the value of A is inferred. However if there are multiple centroid connectors between Z and B the method fails.

Similarly a two field entry A Z will correctly identify a (single) simulation centroid connector from A to Z with the extra node B inferred.

Sound complicated? Stick to wildcard definitions!

15.14.5.3 Wildcard Inputs

In addition KNOBS data read from an external KNOBS file (FILKNB) may (post 10.8) define links using a “wildcard” principle whereby, if an A-node/zone is defined but the B-node columns are left blank (or zero), then the program assumes that the KNOBS data applies to all links **out of** the A-node/zone. Similarly, if the A-node entry is blank (or zero) but the B-node is defined it applies the data to all **entry** links.

In particular this facility is designed to enable users to set entry/exit tolls on zones without having to precisely specify each individual centroid connector to or from a zone. The wildcard principle applies to **all** forms of zones and centroid connectors, i.e., not only zones connected to buffer nodes but also zones which are connected to internal simulation links for which more explicit data inputs (see above) are easy to get wrong.

Note the wildcard principle may also be used to define all entries/exits from a buffer node although not from a simulation node.

In all three cases if data is **not** set for a particular assignment link that value defaults to zero. Thus, in the case of a separate file, not all links need to be included; missing links default to zero.

Note that if a link A-B is included in the 33333 buffer records but has already been coded as part of the simulation network it will be ignored as a buffer link but the ‘KNOBS’ pieces of extra data will be associated with the simulation link. In this respect the capacity index input in columns 43-45 is equivalent to extra data since it too is associated with simulation links.

15.14.6 Storing Knobs: Dirck Access Codes

Once processed by **SATNET** (via whichever format) each “Knob” is stored on the output .ufn file with Dirck Access Codes (Section 15.21): 2303 for the first data field, 2313 for the second, etc. The data thus created may then be displayed using either **P1X** or **SATDB** by referring to those DA codes

15.14.7 Transferring Internal Knobs Data to an External File

As noted earlier (15.14.5) we strongly recommend that KNOBS data be input via an external ascii file “KNBFIL” rather than internally under the 33333 data. In order to assist users who have data stored internally to transfer the data to an external file two useful facilities are provided post 10.8.16.

Firstly, a procedure knobdump.bat, based on **SATDB**, has been created in order to dump existing KNOBS data within a .UFS file (independent of how that data was originally input) into an output ascii file with the correct format for re-input as a KNBFIL.

Secondly, an option has been included within **P1X** Network Editing to delete all existing “second line” KNOBS data from the 33333 data segment.

Thus, by following both the above processes and adding a reference to KNBFIL within &PARAM, the KNOBS data is effectively transferred into a new format.

Note that if the data was originally stored at the **end** of every buffer record (KONAL= T, method 2) in 15.14.5) it is not strictly necessary to delete it from the 33333 data since, if KNBFIL is set, the “end of line” buffer data will never be processed.

15.15 Node-Dependent Parameters: GAP, GAPM, NUC and LCY

As explained in Section 6.4.1 there are four parameters - GAP, GAPM, NUC and LCY - which can be set individually for nodes as opposed to using global default values using data values input on the node record.

15.15.1 GAP and GAPM

GAP and GAPM require little further explanation; they should be used when it is felt that due to the specific physical lay-out of an intersection, gap acceptance is either easier or more difficult than at an “average” intersection. (Node graphics editing within **P1X** may be used to help determine appropriate values.) See also Section 15.22.

We also repeat the warning in Section 15.22 that the default values of both GAP and GAPM are probably highly unrealistic and should be changed, if not on a global basis than certainly on a node-by-node basis.

15.15.2 NUC

Different values of NUC should be used if it is desired to have greater or less resolution of cyclical flow profiles at a particular junction. For example, if the entry profiles to a roundabout are virtually flat there is no particular reason to divide the cycle period into a large number of small time units which will be virtually identical to each other. Here a small value of NUC gives the same results at less computing cost. On the other hand traffic signals with a large number of relatively short stages benefit from the extra resolution of short time units, i.e., a large value of NUC.

Traditionally, and as a very general rule in **SATURN**, our advice has always been to stick to the global values unless there is a very good reason for changing NUC locally. However, post 2007, the benefits of increasing NUC under certain fairly specific local circumstances have become better recognised and extra parameters have been introduced in 10.8 to help deal with these issues.

In particular using larger values of NUC at complex signalised junctions may have benefits for improved convergence.

Thus, for example, with X-turners at traffic signals which are partially blocked by opposing traffic during a green phase, it is important for the simulation to be able to accurately estimate the point during the phase at which gaps begin to appear in the opposing traffic and the X-turns can begin to clear (albeit possibly very slowly). This is particularly so if the green phase is relatively short compared to a time unit and/or the lane is shared.

For example, if NUC is small and the basic time unit is, say, 15 seconds and the duration of a blocked phase is only 10 seconds then the simulated results may differ if the phase is entirely contained within a single 15-second time unit or if it overlaps two time units. (**N.B.** The differences between the two may not look that

large in some respects but they may not be negligible either. For example if the calculated delays were 40 and 45 seconds the “error” of 5 seconds may be small compared to the differences between modelled and observed times; on the other hand a sudden jump of 5 seconds may be relatively very important in terms of convergence between simulation and assignment.)

Thus, for signals with X-turns, we now recommend that NUC should be large enough so that the minimum-length stage time is greater than **three** time units. E.g., if LCY = 100 seconds and the shortest stage time is 7 seconds then NUC should be at least 43 (i.e. the basic time unit should be $7 / 3$ seconds or 2.33 seconds and with LCY of 100, the value of NUC set should be equal to $100 / 2.33$ or 43 rounded to the nearest integer).

Indeed, there is a strong case for setting $NUC = LCY$ at signalised junctions with X-turns so that the time unit corresponds to 1 second in order to achieve maximum “resolution” and every stage transition occurs at an “integer” time unit; see point 3) below under AUTNUC.

In versions of **SATURN** prior to 10.8 there was an upper limit of 25 on the value of NUC both globally and at individual junctions; in 10.8 this has been increased to 125 for individual junctions or for junction types (i.e., $NUCJT()$) but the maximum of 25 is still retained as the global default value. Other relevant changes in 10.8 include:

- 1) Warning 94 and/or Serious Warning 153 have been introduced to detect values of NUC per node which are judged to be too small / seriously too small.
- 2) A subscripted parameter $NUCJT(j)$, $j = 1,5$, has been added to set a default value of NUC for all simulation junctions of type j . NUC continues to function as a global default which may be over-ridden by $NUCJT$ for specific node types.
- 3) If $AUTNUC = T$ then, in processing a network .dat file, **SATNET** will automatically choose an “optimum” value of NUC per node if the default value is judged to be too low (up to the above-mentioned maximum of 125). Thus, in extremis, $AUTNUC$ will set NUC equal to the cycle time for that junction so that one time unit equals one second.

N.B. Increasing NUC for **all** nodes may lead to problems with array dimensions being exceeded and, indeed, this is one reason why in the past users may have been forced to reduce NUC from the (former) default value of 15 down to, say, 10. Indeed, post 11.1, the default value has been decreased to 10. There may therefore be a strong case for using $NUCJT$ to selectively set relatively low values of NUC for, say, roundabouts and priority junctions ($NUCJT(1) = NUCJT(3) = 5$) where resolution is not an issue but larger values for signals ($NUCJT(3) = 25$) such that the overall space requirements are not increased.

Equally increased NUC values also increases the CPU time required to carry out a simulation although, generally, this does not lead to significant increases in overall run times since, particularly in large networks, it is the assignment that takes up almost all the CPU time.

We may further note – see also the final paragraph in 15.15.3 - that having different values of NUC at two adjacent junctions has no real effect on the transfer of cyclic flow profiles between them since CFP profiles will be suitably transformed.

Final thought: Low NUC values do not necessarily lead to “errors” in the simulation; what they do do though is to introduce a certain level of “clunkiness” into the simulation which may be counter-productive in terms of convergence. Increasing NUC values in an existing validated network is unlikely to change it into a non-validated network but it may improve convergence and it may produce noticeable changes at a small number of turns.

15.15.3 LCY – Cycle time

The choice of LCY can however be more important. If all signals in the network operate on the same cycle length then life is simple - all junctions should be simulated using that cycle time and there is no need for any changes from the default LCY. If however different signals operate on different cycle times then, generally speaking, LCY at signals should be set to the local cycle time. Before considering non-signals let us consider the effect of different cycle times.

If A and B are adjacent junctions with equal values of LCY then the OUT cyclical flow profiles at A become the IN profile for link A-B and any “structure” in the profiles is carried forward from A to B. This enables the effect of signal co-ordination between A and B to be modelled. If on the other hand A and B were both signals but on different cycles it follows that at certain times they would be “in phase” and at other times “out of phase”. Rather than trying to model the whole range of possibilities **SATURN** tries to model the “average” behaviour by assuming that if A and B have different values of LCY the A-B IN profile is perfectly flat regardless of what the OUT profiles were like. Clearly this is not a perfect modelling assumption but it has the definite advantage of being easy to implement!

Thus for non-signalised junctions we recommend, as a very general rule of thumb, that LCY should be set equal to the value of the cycle time at the signalised junctions which “most” effects conditions at that junction. This may sound vague - it is intended to be! However most of the time the choice of LCY at non-signalised intersections is unlikely to significantly affect the results so that if there appear to be two “important” controlling signals choose one or the other and don’t worry about it.

At certain points in the standard node/link output table warnings are given if a particular link has **different** values of LCY at its upstream and downstream nodes. In particular the **SATLOOK** table of simulation node properties (11.1.1) contains a line in the link properties which indicates such links. The output is in the form of *s where the higher number of *s, the greater the potential impact. Thus blank implies equal values, * implies unequal with signals at neither end, ** is signals upstream but not downstream, *** is signals downstream and **** is signals at both ends. The logic is that profiles and co-ordination are most important at signals and the number of *s reflects this.

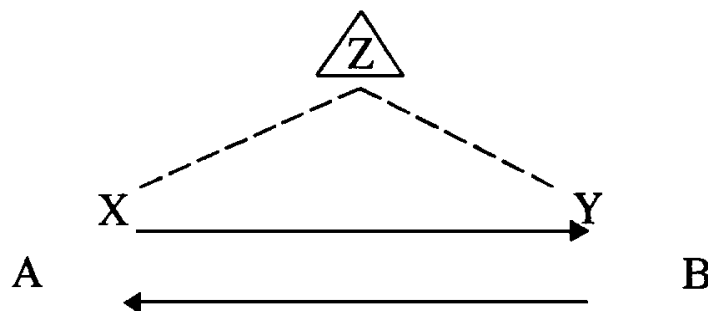
In addition release 11.2.4 introduced a new test to detect a simulation node with, say, $LCY = 60$ while all its immediate neighbours had $LCY = 70$. Serious Warning 183.

Note that having different values of NUC at two adjacent junctions has no real effect on the OUT-IN transformation since an OUT profile evaluated with, say, 10 time units would be suitably expanded into an IN profile with, say, 15 units provided of course that both junctions have the same LCY .

15.16 Simulation Link Flows and Centroid Connectors

15.16.1 Simulation Zone Connectors

Because of the way in which zones in the simulation network are connected to links, not nodes, certain ambiguities may arise with respect to the definition of “link flows”. The various possible definitions are illustrated below for a zone Z which is connected to an internal simulation link AB . X and Y mark the “imaginary” points along AB where trips leave and enter Z .



The “flow” on AB may, in theory, be defined in five different ways; i.e., the flow along:

- AX - the entry flow onto the link,
- XZ - the exit flow to the centroid,
- XY - the “mid-link” flow,
- ZY - the entry flow from the centroid, and
- YB - the arrival flow at the stop line at B

For uniformity throughout **SATURN** we assume that the “link flow” is always taken to be the mid-link flow, i.e. the flow on the link once all traffic destined for the zone has been removed and before any new traffic has joined from the centroid. Thus as defined the link flow is probably lower than the flow that might be observed on the link in reality, although the fact that the link has been connected to a zone implies that the flow level probably does vary along the link.

This therefore is the definition of link flow as used, e.g., in comparing modelled and observed flows (see Section 15.6) and is the (default) link flow as annotated by program **P1X**. By contrast the “ARRIVE FLOW” or “Downstream Flow” as printed out by the FLOW-DELAY tables and illustrated in Table 17.1 corresponds to the stop-line flow, YB above.

15.16.2 Simulation Link Exit/Entry and/or Upstream/Downstream Flows

The previous section has described how flow may exit the network at the upstream end of a simulation link and equally enter the network at the downstream end of a simulation link using centroid connectors.

It is also possible for flows to exit/enter at **either** the upstream or the downstream end of simulation links even if they are not explicitly connected to zones. This possibility arises with bus routes which may originate/terminate at either end of simulation links (dependent on UPBUS; see 6.9.2). It may also arise, less obviously, with either pre-loaded or PASSQ flows (see 15.5 and 17.3.1 respectively) where the rule “flow in equals flow out” may be violated. Thus, in effect, **all** simulation links are potentially bridged by exit/entry links, although only those with explicit centroid connectors are shown as such on, e.g., **P1X** plots.

Please note that this possible ambiguity **ONLY** arises with links in the simulation network and not at all to links in the buffer network where there is only one possible definition of link flow.

Note as well that the distinction between “demand” and “actual” flows as described in Section 17.2 also applies to all the different definitions of flows along a simulation link. Thus the “actual” upstream exit flow on a link may be less than the “demand” upstream exit flow due to queuing upstream.

15.17 Pcu's, Cars, Buses and Vehicles

15.17.1 General Principles

In theory the “units” used to describe traffic flow in **SATURN** can be anything the user likes; e.g., the trip matrix can be units of cars, vehicles, pcu's or whatever. The only real restriction is that the link saturation flows and the trip matrix elements must be defined in terms of the **same** units. In practice however it is **strongly recommended** that trips and saturation flows be defined as pcu's and “most” printed text assumes this to be the case.

Strictly speaking, and for pure buffer networks, flows do not necessarily even need to be defined **per hour**; they could be defined as, e.g., daily flows as long as **all** appropriate commodities such as capacities are defined in the same units. However moving away from hourly rates causes problems for the simulation where the length of the simulated period LTP may only be defined in minutes and the definite assumption is that the flows being simulated are hourly flows.

Note that the same rule also applies to all input counted flows (see 6.10 and 13.1.4); i.e., that they should always be in the same units as all other flows with the presumption being that they are in **pcus/hr**. Equally it applies to all definitions of capacities, e.g., as contained in buffer network speed-flow curves.

One partial exception to the above rule is buses where: (a) the network 66666 definitions of bus routes input frequencies (buses/hour) which are then factored by BUSPCU to give bus flows as pcus/hr, and (b) output bus data is sometimes given in terms of buses and sometimes, e.g., when giving total bus flows on links, in terms of pcu's. Hopefully the text should make it clear what is being printed.

In the case of Multiple User Classes – and stacked O-D trip matrices – it is further assumed that **all** the various stacked matrix levels will be in the same units and that their assigned flows may simply be added together.

Clearly it is vital for users, when importing data into **SATURN** from external sources, to check the units of the external data and to adjust as necessary. For example, this applies to the importing of trip matrices (which may be in vehicles/hr as opposed to PCU/hr), speed-flow curves, etc. etc.

15.17.2 PCU Factors by Vehicle Class

SATURN 10.1 introduced an extra parameter VCPCU (disaggregated directly by vehicle class and indirectly by user class if necessary; see 5.8) which is used to convert pcu's - on the recommended assumption that all trip matrices, capacities etc. are defined in units of pcu/hr - into vehicles.

VCPCU is useful in circumstances such as vehicle emissions when it is more natural to deal with parameters per vehicle rather than per pcu. It is also used in the calculation of toll revenues (see 20.4.1) which are paid by vehicle rather than by pcu.

Post 10.7 **P1X** has an option to annotate individual user class link flows in veh/hr instead of pcus/hr (by factoring the assigned flows by $1/VCPCU$). However it is more difficult to apply the same option to, e.g., total flows where some of its components, passq flows etc. may not be unambiguously identified with a single user or vehicle class and hence pcu-factor.

Note that, by default, all VCPCU factors equal 1.0, in which case it has no direct effect on any **SATURN** outputs.

15.18 Interpolating Routes

Several programs require that "routes" (e.g., bus routes, joy ride routes, etc.) be defined as a sequence of consecutive nodes. For long routes this can be laborious and therefore a simpler method is available if node co-ordinates have been defined whereby the user defines the first and last nodes and the program works out the sequence of nodes which most closely approximates to a straight-line or crow-fly path between the two nodes. The principle can be extended to the case where a route is defined by more than two nodes, the first and last plus any intermediate nodes where there is a decided "kink" in the route.

More specifically if we wish to interpolate a path from node A to node Z we first work out the angle from A to Z, then the angles from A to all its exit nodes B_1, B_2, \dots and choose the B-node whose exit angle is nearest to the A-Z angle. The procedure is then repeated by taking the angle from B to Z and choosing the "nearest" exit C. Exits more than 90° from the desired direction are excluded; it is therefore possible for the algorithm to become "stuck" if there are no exits within 90° . In these cases the user will need to define more nodes within the path.

Alternatively, post 10.9, an alternative interpolation algorithm has been introduced which finds the minimum distance route between A and Z if the first method fails. This requires that a &PARAM parameter MINDER is set TRUE.

Depending on the particular application the interpolated nodes may either only use “real” links - in the sense that one-way links are only used in one direction - or non-directional links.

This facility is particularly useful in defining bus routes, not only in terms of reducing the amount of data to be coded, but also because the route definitions do not need to be altered if the network is changed so that nodes are inserted and/or removed from the original network.

Note, however, that in order to use this facility node co-ordinates **MUST** be defined (although, strictly speaking, only for those nodes which lie on the interpolated path).

WARNING: If two successive nodes to be interpolated are some distance from one another and there are multiple possible routes, interpolation may not necessarily find your desired route; the solution in this case is to define nodes which are much nearer together and for which the route to be interpolated is unambiguous.

Currently interpolated routes may be defined within the following programs:

- ◆ **SATNET** to define bus routes; See note (5), 6.9.2.
- ◆ **SATCH** to define a “spline” of links along which flows are to be cordoned;
- ◆ **P1X** to define bus routes, joy rides and GIS alphanumeric link names.

15.19 Select Link Analysis (SLA)

“Select Link Analysis” is a general term which refers to the identification of specific routes and/or trips assigned to selected links (where in this context “links” may refer to either “roads” or “turns” or “centroid connectors” or even “nodes”) and the calculation of various properties associated with those trips. Thus the analysis may identify, for example:

- ◆ the O-D pairs which use a particular link;
- ◆ the fraction of trips from each O-D on a link;
- ◆ the flows on all other links from the selected trips.

Other forms of analysis are of course feasible. However the central element in select link analysis is the ability to trace the routes generated during the assignment process and to select those that satisfy a particular criterion for further scrutiny. How this is done within **SATURN** is described in Section 15.23.

Select link analysis is a very powerful tool not only for the analysis of schemes but also for the validation of a base year network. In many respects it is the converse of building trees in order to check on an assignment; trees tell you which links are used by specified O-D pairs; select link analysis tells you which O-D pairs use selected links.

Within **SATURN** it is possible to perform select link analysis within several different programs and with slightly different outputs (although very often the same information may be obtained from two or more programs). We therefore first identify the options available:

- 1) **SATPIJA** may be used to undertake a “PIJA analysis” whereby the fraction of trips “P” for each “I-J” movement assigned to link “A” is stored during an assignment. The main purpose of this option is to provide the “PIJA or UFP file” required by **SATME2** and no print facilities are provided within this program. More than one link or turn can be analysed within a single run.
- 2) Program **SATU2** (13.7) can read a PIJA/UFP file and output a matrix of trips (as a UFM file) using selected links. **MX** may then be used to print the trips (with the option of aggregating zone-to-zone trips into sector-to-sector trips and printing).
- 3) Both **P1X** (11.8.1) and **SATDB** (11.10.7.5) can repeat the assignments carried out in the assignment and select trips which either:
 - a) pass through a selected node,
 - b) pass through a selected sequence of nodes in order, or
 - c) pass through one or more of a set of “screen line” links.

Option (b) includes the possibility of either identifying links - by specifying two adjacent nodes - or turns - by identifying three nodes.

Option (c), further explained in (11.10.7.5), allows both conventional “screen lines” in the sense of a closed set of links surrounding a town centre or, more generally, any set of links. The screen lines may be defined either using the link selection facility (11.6.1) or a set of 7777 input data records (6.10) in both **P1X** and **SATDB**. **P1X** also offers two additional methods for defining screen lines – interactively using the mouse or via an external data file (11.8.1.7).

Those trips which satisfy the selection rules are re-loaded and the total assignment pattern of trips before and after they pass through the selected node or nodes is displayed, graphically or as a data base table.

P1X and **SATDB** have further options which duplicate those in **SATU2** to either:

- ◆ output a selected trip matrix UFM file (11.8.1.3);
- ◆ print a sector-to-sector trip matrix.

Generally speaking **P1X** is considerably easier to use than **SATDB**, firstly, through the use of mouse-based link or node selection and, secondly, since the display of the selected flows is MUCH easier to appreciate in a graphical format. On the other hand **SATDB** may be easier to use in conjunction with key files.

SATU2 has effectively been superseded by **P1X** and/or **SATDB** and is less convenient; its use is not recommended.

So the “best buy” recommendation is to use **P1X** in the first instance!

Finally we should note the caveat expressed in Section 15.23.2 that under certain circumstances (e.g., elastic assignment) select link analysis (as with other similar analyses) is based on an approximation and that the select link flows need not be entirely consistent with the “true” flows. In these circumstances the select link results should be viewed as “indicative” rather than exact. The difference statistics

generated within **SATALL** (based on the errors from **all** links) may be used as a rough guide to the errors to be expected on any single link analysed.

15.20 The Dutch Option (Long Node Numbers)

The DUTCH option has been introduced to allow nodes with up to 8-digit node numbers to be defined in buffer networks - so-called because it is common practice in The Netherlands. The major effect of this option is to change a number of the input formats so that node numbers in certain circumstances occupy 10 columns of data input as opposed to 5.

Note that simulation nodes are still restricted to 5 digits (although it is recommended that a maximum of 4 be used so as to keep the formats "neat"). Equally zone numbers are still effectively limited to a maximum of 5 digits (see 5.1.6).

More specifically formats in the following programs are altered as indicated below:

(A) SATNET - SEE SECTION 6.

(A.1) THE BUFFER NETWORK DATA CARDS-- SEE 6.6

Col. 1	A 'C' if the following node refers to a zone.
Cols.2 - 10	The A-node for the link
Col. 11	A 'C' to indicate a zone number following.
Cols.12 - 20	The B-node for the link
Cols.21 – 25	The link time (in seconds) or speed (in kph) at free-flow conditions
Cols. 26 - 30	The link time (in seconds) or speed (in kph) at capacity.
Cols. 31 - 35	The one-way link capacity (in pcus per hour)
Col. 38	A one-way/two-way indicator
Col. 39	An 'S' if speeds were defined above; otherwise times are assumed.
Cols. 41 - 45	The link distance (in metres).
Cols. 46 - 50	The power to be used in the link flow-delay curve
Cols. 53 – 55	A "link index" in the range 0-999
Cols. 56 – 80	(Optionally) up to KNOBS extra data items

(A.2) The Restricted Turns or Links - See 6.7.

Cols. 11 - 20	The B-node, B
Cols. 21 - 30	The C-node, C (blank or 0 in the case of a link)
Cols. 31 - 35	The ban/penalty indicator for user class 1,
Cols. 36 - 40	Ditto, for user class 2,
Cols. 41 - 45	etc.

Cols. 11 - 20 The B-node, B

(A3) NODE AND CO-ORDINATES - SEE 6.8

Cols. 1 A 'C' if columns 2 to 10 contain a zone number.

Cols. 2 - 10 The node or zone number.

Cols. 11 - 15 Its X co-ordinate

Cols. 16 - 20 Its Y co-ordinate

(A4) BUS ROUTES - SEE 6.9

Cols. 2 – 5 The “name” of the route (which must be numeric)

Cols. 6 ‘T’ if the route is two-way and the node order is exactly reversed (in which case the reverse route need not be coded) ; otherwise leave blank

Cols. 7 – 10 The route frequency in buses per hour

Cols. 11 – 15 The number of nodes through which the route passes (i.e. the number of node entries following

Cols. 16 – 25 The first node on the route

Cols. 26 -35 The second node on the route, etc. up to 6 nodes, column 75

If the route passes through more than 6 nodes the list of nodes is continued on a second (or even third) record starting in cols. 16 - 25.

N.B. The strict column formats do not apply if EZBUS = T independent of the value of DUTCH.

(A5) LINK AND/OR TURN COUNTS - SEE 6.10.

Identical changes to (A2) above.

(B) SATPIJA - SEE SECTION 13.2.1.

Link and/or turn counts are specified as under (A.5) and (A.2) above.

Essentially the changes are made anywhere that it is possible that 8-digit buffer node numbers MIGHT be input, but NOT in those areas where only simulation node numbers may be used.

15.21 Referencing Data Arrays Via Dirck Access Codes

15.21.1 General Principles

Certain programs, notably **SATDB** and **P1X**, allow the user to select data by reference to a “Dirck Access Code” as opposed to referring to, say, free-flow travel time by name (“Dirck Access” is a very egotistical pseudonym for “Direct Access” which it tries to replicate). The precise details of Dirck Access files are not important here - the most important point to appreciate is that each data field

stored on a **SATURN** UF file has a code associated with it; free-flow travel time, for example, is coded as 1803 so that asking for free-flow travel time to be annotated in **P1X** causes the program to “read” and annotate record 1803. The same effect can be obtained by referring to 1803 directly.

Note that the final digit in a DA code indicates what “type” of data is stored. Thus all “integer” variables are stored in codes ending with a 4 whereas all “real” numerical data (i.e., numbers which may include a decimal place such as free-flow travel time above) end with a 3 (e.g., 1803). Post 10.7 real arrays may also end with an 8 (and, eventually, integer arrays with a 9).

A second point to note is that the coded data arrays refer to either: (a), simulation links; (b), simulation turns; or (c), assignment network links (which include both (a) and (b) plus all buffer links) so certain DA codes will not be relevant under certain circumstances.

The main reason for introducing code numbers is to increase flexibility without a massive increase in programming effort, particularly since certain data arrays are optional whereas others are mandatory. Thus free-flow travel times are always defined whereas the link flows for user class 4 may not be.

A full list of the Dirck Access codes used within a particular network (*UFS etc.) file may be listed interactively using the auxiliary program **DALOOK** or partial lists generated by **P1X** etc. See also Appendix J for a full list. Each array has a short title associated with it which specifies its contents; these titles are defined in **SATNET** either as default text or as read from an (optional) supplementary file SATTIT.DAT which also gives very useful general information about how DA codes are used.

Note that not all DA arrays will necessarily be useful to users, for example the arrays containing “packed” data will be largely unintelligible (but see 11.10.6).

An explanation of the specific codes relevant to capacities is given in Section 8.9.5 and to times and delays in Section 17.10. See also Appendix J for a full list.

15.21.2 Creating your own DA codes in SATDB

Users may add their own data to .ufs files via **SATDB** referenced by a DA code of their choosing (see 11.10.12) but care must be used not to over-write existing essential DA codes.

This may sometimes lead to problems if the user selects a DA code for output which is “available” in that particular network but which may be used either in different “forms” of networks (e.g., simulation networks use arrays that do not appear in buffer-only networks) or in future versions of **SATURN**. At the present moment array codes in the range 3003 to 3293 are never used and will **not** (barring acts of God, etc. etc.) be used in the future; they are therefore recommended as being exclusively “reserved” for use by users.

15.21.3 Extended Dirck Access Codes

In **SATURN** versions 8.5 and beyond the coding conventions used to identify Dirck Access arrays were “extended” to cope specifically with the problems created by more than 10 user classes where, in effect, all available numerical

codes were used up. Thus class- specific flows were stored in arrays 3803, 3813, 3823, etc. for user classes 1, 2, 3, etc. up to 3893 for user class 10. 3903 however was reserved for something else so that an 11th user class could not be accommodated.

The solution adopted was to add class-specific digits BEFORE the basic code so that with 11 classes the DA flow codes became 3803, 103803, 203803, etc. up to 1103803. The effect was similar to having decimal codes such as 3803.0, 3803.1, 3803.2 but retained the basic principle of integer codes.

At the moment such codes are used in networks with more than 10 user classes, in .UFT files to store data from multiple time periods and, in addition, to extend header records (e.g. 100104 adds extra data to 104) so that old **SATURN** UF files have the same array lengths as the latest one (to ensure upwards compatibility).

15.21.4 DA Codes for Actual User Class Flows

Whereas there are two explicit DA codes used for total demand and total actual flows (4503 and 4513) flows by user class are only stored by **demand**. Thus (see above) the demand flows for user class 1 are stored in DA code 3803, for user class 2 in 3813, etc. etc. and there no arrays/DA codes within .ufs files which directly store actual flows by user class.

However, it is possible, in certain circumstances, to use DA codes 3808, 3818, etc. to obtain **actual** flows by user class 1, 2 etc. For example, **SATDB** will accept such codes as a link data input definition (11.10.2) and they may also be used with DBDUMP (15.46). What actually happens, however, if 3808 is requested is that the actual array read in is 3803 (user class 1 demand flows) but the data is immediately factored down by the global ratio of actual to demand flows. Thus, the end effect is the same as though 3808 was explicitly stored in the .ufs file.

DA codes 3808, 3818, etc. etc. may also be used in **P1X** to create and annotate data using a DA code but individual user class flows – both demand and actual - may also be accessed using items in the “Flow” list. Note that, within this list, user classes 1, 2 and 3 are always explicitly listed along with, if there are more than 3 user classes, a single “designated” user class for which the demand/actual flow may be obtained. By changing the definition of the “designated” user class within an Options sub-menu flows for **any** user class may be obtained, rather than having to use, say, code 3858 to get UC 6 flows.

15.22 Choice of Gap Parameters

The choice of the parameters GAP, GAPR and GAPM can have a very strong influence on the capacities and delays given by the **SATURN** simulation model and some care should be exercised in their choice. In particular the user may wish to set parameters such that the **SATURN** output is similar to that given by other models, in particular models for isolated junctions such as the TRRL programs ARCADY, PICADY and OSCADY. By a judicious choice of parameters this can be achieved.

The role of the gap parameters in setting the capacity of a give-way movement is explained in Section 8.2.2. In the simplest possible case of a minor arm opposed by one major arm (e.g., at a T-junction or any arm at a roundabout) the capacity C_m of the minor arm is given by the equation:

Equation 15.2

$$C_m = S_m (1 - V_M / S_M)^{G S_M}$$

where S_m is the saturation flow of the minor arm, V_M and S_M are the flow and saturation flow of the major arm and G is the gap value.

Hence C goes from a maximum S_m equal to its saturation flow at zero opposing flows down to zero at $V_M = S_M$ (or, strictly speaking CAPMIN; see 8.2.3) with a power defined by $G.S_M$. In general TRRL models predict a linear relationship between C and V_M so that in order to reproduce this same form in **SATURN** it is necessary to set $G = 1/V_M$, i.e., set the gap parameter GAP to the inverse of the saturation flow of the controlling arm(s).

Very often the GAP values derived in this way seem small, particularly when interpreted strictly as a gap in traffic that entry traffic would “accept”. For example if the controlling saturation flow were 3,600 pcu/hr then GAP should be one second.

It must however be appreciated that GAP is essentially a parameter fed into a model. Such models are only approximations to reality and contain a number of intrinsic errors (“specification errors”) which, to a certain extent, can be corrected or counter-balanced by changes to the model parameters. For example, the gap acceptance model in **SATURN** assumes random (Poisson) cross traffic (ignoring for the moment cyclical effects) whereas in reality one knows that traffic tends to come in surges, the effect of this being that the random model tends to under-estimate capacity. Empirically, if we accept the TRRL relationships as “correct”, then the best value to choose for GAP is $1/S_M$.

We may also note that the standard default value of GAP set by **SATURN** is 5.0 seconds which is almost certainly on the high side, causing the **SATURN** simulation to under-estimate capacities. The reasons for choosing 5.0 as a default in the first place are largely historical and arbitrary. The reasons for not changing it since are (a) the fact that best values almost certainly vary from one to another (hence it was made a junction-specific parameter in later versions of **SATURN**); and (b) setting the default to a more reasonable value might discourage users from deciding on more suitable values.

The same principles apply to the choice of GAPR and GAPM; i.e., that they are first and foremost model parameters which should be interpreted only loosely as acceptable gaps. However with priority junctions it is difficult to choose a single value of GAP which makes the dependence on ALL major flows linear since each major flow may have a different saturation flow.

15.23 Re-constructing Assignment Routes: The SAVEIT Option and UFC Files

15.23.1 General Principles

While the most important function of assignment is to obtain estimates of flows on links it is very often equally important to be able to analyse in detail the O-D **routes** used to obtain those flows.

Examples of analysis options which make use of O-D routes include:

- ◆ Building minimum cost routes in **SATLOOK**, **SATDB** and **P1X**.
- ◆ Repeating full loadings of complete trip matrices in **SATDB** (11.10.7.4).
- ◆ Select link assignments in **SATDB** and/or **PIX** (15.19).
- ◆ Cordoning matrices within a sub-network (**SATCH**, 12.1).
- ◆ Producing a PIJA file using **SATPIJA** (13.1.2)
- ◆ Producing a file of route flows (**SATPIG**, 12.6)
- ◆ Turning flows at buffer nodes (15.36).
- ◆ Producing cost and/or skimmed matrices (15.27)

In order to obtain the route flows necessary to carry out such analyses a parameter SAVEIT must be set to .TRUE. in the final assignment in **SATALL** (or **SATEASY**), most easily done by declaring it to be .TRUE. (its default) under &PARAM in the .DAT file input to **SATNET**.

There are then three different methods by which the O-D route information is preserved under SAVEIT dependent (mainly) upon the assignment method (MET) used:

- (1) Remembering the costs used on each Frank-Wolfe iteration and their weights in order to be able to re-construct each individual route by re-building trees (MET = 0 only);
- (2) Explicitly storing flows per individual O-D route (path) (path-based assignment, MET = 1 only);
- (3) Storing a “bush” of splitting factors per individual origin/user class from which individual O-D route flows may be calculated by a single pass (OBA and/or Frank-Wolfe with extra steps added).

Methods (2) and (3) are generally considerably **faster** than method (1) in terms of route flow analyses but may require extra memory to store the required data and/or extra CPU to create them in the first place. The following 4 sub-sections, 15.23.2 through 15.23.5, deal exclusively with method (1); the equivalent information for methods (2) and (3) is given, briefly, in 15.23.6 and, in more detail, in Sections 21.4 and 22.5.2.

Under method (1), for every assignment iteration within the Frank-Wolfe algorithm the complete set of link “costs” used to construct minimum cost routes is preserved in a separate “UFC” binary file. These costs may be used later in order to re-construct the specific “trees” from each iteration and thereby re-construct the specific O-D routes from that iteration for further analysis.

The filename convention is that if the main network file produced by **SATALL** is net.ufs then the cost file will be net.ufc. In addition the .ufc files record the “weights” as used by each iteration in the final solution (see 7.1.2)..

Not creating a .ufc file will not affect the normal analysis or use of the .ufs file, except clearly, if .ufc does not exist, then none of the above mentioned analyses may be invoked.

Note that .ufc files etc. are **only** used under the standard Frank-Wolfe link-based algorithms (MET = 0; see Section 21).

Under method (3) the route information is stored within a “UFO” binary file; see 15.23.6.

15.23.2 SAVEIT/UFC as an Approximation: The SAVEIT Assignment

Under certain (fairly restricted) conditions the routes and costs stored in .ufc files under SAVEIT will be identical to those used to carry out the actual assignment within the assignment-simulation loops:

- (1) a buffer network with a fixed trip matrix,
- (2) post 10.9, a simulation network where MASL = 1 (i.e., **SATALL** has only been through a single assignment and the routes/costs used on that assignment are retained), or
- (3) UFC109 = T and the total number of assignment iterations is relatively small (see 15.23.3 below).

Otherwise – and very often the above conditions are **not** satisfied - an extra “SAVEIT assignment” needs to be carried out by **SATALL** in order to re-create route flows and to create the .UFC cost file.

Thus the SAVEIT assignment is a final complete Frank-Wolfe assignment stage carried out at the end of the simulation-assignment loops using the final set of speed-flow curves and starting, in effect, with a “blank sheet of paper”; e.g., the initial all-or-nothing assignment uses free flow costs. The set of iterative costs and weights stored in the .UFC file and used in the subsequent analyses will be those derived from the “SAVEIT assignment” as opposed to those from the “true” assignment. (Specifically under elastic assignment the final assignment uses the fixed trip matrix generated by the final elastic assignment.)

Almost all options which may be used to “improve” the normal assignment within the assignment-simulation loops may also be invoked by the SAVEIT assignment – for example, an aggregated SPIDER network may be used under SAVEIT as well as the normal assignments to reduce CPU – but there are also certain “improvements” that are only feasible within SAVEIT. In particular, the “trick” to eliminate zero-flow links within a SPIDER assignment (see 15.56.5.3) may be used to great effect within a SAVEIT assignment.

In addition, post release 11.2, a SAVEIT assignment may use an Incremental Assignment (7.11.12) to initialise Frank-Wolfe Assignment with the objective of reducing the onset of residual paths. To invoke incremental assignment set the parameter INKS_S in the network .dat file to, say, 4 to request 4 increments. Empirical tests to determine whether or not the method is effective are under way so it should be considered as an **experimental** option.

While, in principle, the SAVEIT assignment should converge to an identical solution to the full assignment in practice, due to lack of convergence, etc. it is

only an approximation and the flows and assigned routes etc. differ. Although the higher the level of assignment convergence, the better the approximation will be. See 15.23.4 for a discussion as to how the parameters NITA_S and UNCRTS may be set to ensure optimum convergence.

The differences between “true” and SAVEIT assignments can have important implications for, e.g., skimmed matrices as used for economic evaluation or variable demand models (see 7.8.6 and 15.27.6), Select Link Analysis (see 15.19) or PIJA analysis (13.3.12). For example, the total flows on a link generated by a select link analysis may not exactly equal those generated by the original assignment.

However, it also needs to be borne in mind, that the “errors” associated with SAVEIT are just one **extra** source of “noise” to be added to the non-convergence errors from **SATALL** proper (see 2.1 and 9.5). Essentially the SAVEIT assignment is an approximation to an approximation. Therefore, a “perfect” SAVEIT assignment is not a guarantee of an “error-free” economic evaluation although it may help.

15.23.2.1 Comparison Statistics: SAVEIT vrs Original Assignment

In order to assess the consistency of the two different assignments a set of difference statistics is generated and printed at the end of the SAVEIT assignment comparing the SAVEIT link flows with the “true” assigned link flows. These include:

- ◆ The average GEH difference statistic comparing the as-assigned flows and the SAVEIT flows;
- ◆ The mean average absolute difference between the flows (expressed as a percentage);
- ◆ The relative standard deviation (%);
- ◆ The average absolute difference in pcu/hr;
- ◆ The percentage difference between the total pcu-hrs calculated using the assigned and SAVEIT flows;
- ◆ Ditto using distance instead of time;
- ◆ Ditto using assignment cost instead of time.

The last three statistics (new in 10.6) may be thought of as “weighted” differences of the link flows, weighted by time, distance or cost.

If the two assignments are self-consistent all the above statistics will equal zero; larger values imply that the various options listed in 15.23.1 will only generate approximate answers and the statistics give a quantitative estimate of that approximation.

The difference statistics are also held in the output .ufs files and may be examined using the analysis option 8 within **SATLOOK** and/or under Analysis etc. in **P1X** (11.8.4.7).

15.23.3 UFC109/UFC111: Alternative UFC files

In order to remove some of the uncertainties associated with SAVEIT assignments an option has been introduced in release 10.9 to create .UFC files which reproduce **exactly** the iterative costs used by Frank-Wolfe and in a more space-efficient format. It requires that a logical parameter UFC109 is set to .TRUE. under namelist &PARAM in network .dat files. Thus if UFC109 = T (default = F) two things happen:

- (1) The costs stored are those from the “actual” assignment (see 15.23.3.1);
- (2) Times, not costs, are stored under MUC (see 15.23.3.2).

In addition, post release 11.2, the second option is independently controlled by an alternative parameter UFC111 such that, if UFC111 = T, then times are always output, not costs, independent of the value of UFC109.

15.23.3.1 UFC109 = T: Storing “True” Frank-Wolfe Iteration Costs

The costs/times are stored as a “rolling summation” of **all** Frank-Wolfe iterations over **all** simulation-assignment loops (up to certain limits – see below) instead of re-creating the assigned route flows by an extra SAVEIT assignment. Similarly the “weights” per iteration take into account not only the weights during the assignment stage itself (see equation 7.2b) but also any averaging between assignment simulation loops associated with DIDDLE, AUTOK, etc.

This has the benefit that any secondary analysis, e.g., skimming, **SATME2** (see 13.3.13), etc. based on routes is **exact**, not an approximation, since it reproduces the precise routes used in the full assignment. This therefore reduces (but not entirely removes) some of the problems associated with, e.g., the uniqueness of skimmed matrices (see note 6, 15.27.6).

The disbenefit is that there may be many more rolling iterations in total than there would be in a SAVEIT assignment which means that: (a) the .UFC files are larger **and** (b) **any secondary analyses take proportionately longer**.

However, if the **total** number of rolling FW iterations becomes too large (greater than a &PARAM parameter NITA_C, default 256) then we revert to an extra SAVEIT assignment in any case.

Note that the total number of Frank-Wolfe iterations aggregated over all assignment-simulation loops depends on the rate of convergence as well as the values set for MASL and NITA. Thus, if convergence is slow and the maximum number of loops MASL is used and the maximum number of iterations per loop are also used, then the total number of iterations equals MASL times NITA. Therefore, reducing MASL and/or NITA will make it more likely that the option will be used (although achieving an acceptable level of convergence is certainly a more important objective). See Section 9.5 for advice on improving convergence and 9.5.4 on the choice of NITA.

15.23.3.2 UFC109 or UFC111 = T: Storing UFC Link Times under MUC

Under multiple user classes if either UFC109 or UFC111 = T the output .UFC file stores the sets of link **times** per iteration as opposed to link (generalised) costs by

both iteration **and** user class. This is possible because the times per iteration are constant between all user classes on the same Frank-Wolfe iteration; the costs may differ but only because the fixed costs per user class differ and, since the fixed costs are fixed throughout (by definition), it is straightforward to construct the costs per iteration/UC by adding (.UFC stored) times per iteration to fixed costs by user class.

This means that the .UFC files produced under UFC109/111 will be reduced in size by a factor of 1/NOMADS with only a very small overhead in reconstructing costs as required for secondary analysis. However, for a single user class we continue to store cost and there is no reduction in size.

Note that the default value of UFC111 is T, meaning that, post 11.2, the option to output times rather than costs is virtually **always** invoked for MUC UFC files and indeed UFC111 should only be set to F for test purposes.

15.23.4 NITA_S and UNCRTS: Accuracy of SAVEIT/UFC Assignments

The final SAVEIT assignment which creates the .ufc file and the corresponding route flows may use a different maximum number of Frank-Wolfe iterations via the parameter NITA_S. Thus if NITA_S is non-zero it replaces the value of NITA (see 7.1.5) in the final assignment. This option is useful if you have been using a relatively low value of NITA within the assignment-simulation loops (not a bad idea with DIDDLE; see 9.5.2) but you want a more representative single final assignment.

Increasing NITA_S leads to improved convergence of the SAVEIT assignment and therefore should reduce (but not eliminate) the problems of approximation referred to above, 15.23.2. Thus the default value of NITA_S is 99 whereas the default value of NITA is only 20. Users frequently increase NITA_S to even larger values, e.g., 256.

For similar reasons the value of UNCRTS which also controls the number of iterations undertaken by a Frank-Wolfe assignment (see 7.1.5) may also need to be reduced in order to prevent the SAVEIT assignment terminating prematurely. Post release 11 it is set equal to the best GAP value (9.9.1.2) achieved by the main simulation-assignment loops such that the convergence in the main and SAVEIT assignments will be roughly comparable. (Prior to 11.1 UNCRTS was set equal to the **final** value of UNCRTS set during the assignment proper under AUTONA (note 4), 9.5.4) and which is generally a lower value than the GAP; this could therefore lead to extremely long SAVEIT assignments for no appreciable gain in overall accuracy.)

In addition, to improve convergence, **SATURN** also automatically sets PARTAN assignment under SAVEIT (for single user classes) and allows PARTAN as an **option** for MUC SAVEIT assignments via a parameter SPARTA. See 7.11.7 and 15.57.6. Note that a side benefit of using PARTAN/SPARTA is that, by reducing the number of Frank-Wolfe iterations required to produce a SAVEIT solution, all subsequent analysis steps that use UFC files (e.g., SATPIJA, SLA, etc.) will become correspondingly faster.

15.23.5 SATUFC – Re-creating .UFC files

A .ufc file may also be created **after** the original run of **SATALL** using a procedure **SATUFC** introduced in **SATURN** 10.6. Basically **SATUFC** reads a .ufs file and extracts the necessary information to carry out a SAVEIT-style assignment – with the added proviso that the value of NITA_S can be set as a purely numerical value on the command line. E.g.:

```
SATUFC net 30
```

will produce an output file net.ufc based on NITA_S = 30 (independent of the value in net.ufs). If a numerical parameter is not used NITA_S is simply taken from net.ufs.

In fact **SATUFC** is not a separate program, it is simply a run of **SATALL** with, effectively, zero assignment-simulation loops and only the SAVEIT step included. As a consequence it therefore requires that the original trip matrix .ufm file is available.

However, post 10.8, if the original network were based on an elastic assignment the “SAVEIT assignment” uses a **fixed** trip matrix assignment algorithm, in which case it uses the **output** trip matrix (i.e., ROADIJ) from the original run rather than the original input trip matrix file (FILTIJ). These algorithms are generally faster and do not suffer from problems of terminating early.

SATUFC has several advantages:

- ◆ If the original .ufc file did not converge sufficiently well a better level of convergence may be achieved by increasing NITA_S.
- ◆ It may also be computationally efficient to set SAVEIT = F in the original network and to only run **SATUFC** when a .ufc file is actually required. (If NITA is very small and NITA_S large the SAVEIT step may take virtually as long as the original assignment-simulation loops.)
- ◆ UFS files can be sent without their .ufc files as the latter can be easily re-created.
- ◆ By adding an argument UFO to the command line a .UFO file may be created at the same time as the .UFC file (see 15.23.6 below).
- ◆ Post 10.8 it also updates the .ufs file to correctly set SAVEIT and/or SAVUFO to T so that subsequent analysis programs such as **P1X** will “know” that the .UFC/.UFO files should exist.
- ◆ Post 11.1 if the original network were run using OBA then SATUFC will, in effect, generate a set of O-D paths in a .UFC file which approximate the same final set of link flows. Thus the (newly created) .UFC file and the (original) .UFO file fulfil similar functions in terms of post-assignment analysis, e.g., select link analysis, but the .UFC file has the advantage that it may be used in certain programs such as **SATPIG** where the .UFO file may not.
- ◆ Similarly a .UFC file may be generated from a path-based assignment where the .UFQ files which store the paths may also not be suitable for all forms of post-assignment analysis.

- ◆ Note that if using SATUFC with either a path-based or OBA original solution it may make sense to copy and rename the original .ufs file so the two sets of solutions may be used independently.

15.23.6 Alternative Formats for Saving O-D Routes: UFO and UFQ files

The .ufc files described above are relevant to assignments done using the Frank-Wolfe link-based algorithms (MET = 0). Path-based and origin-based assignments use their own particular techniques to preserve route flow information and, in addition, link-based routes may also be converted into an “extended” UFO (OBA) format.

Thus path-based assignment (see 21.4) stores the exact path data in .UFQ files while OBA stores the equivalent “splitting factors” in .UFO files (see 22.5.2). In both cases the information saved is “exact” unlike the link-based .UFC files which (see 15.23.2 above) may be an approximation based on an extra SAVEIT assignment.

We note that path-based UFQ files are restricted to single user class assignments only so that, in terms of practical applications, they are not really relevant and they are not considered further.

In principle the same forms of analyses (such as those listed in 15.23.1) may be carried out under all 3 methods, although the precise algorithms used to do so may differ. Thus .UFC-based algorithms based on a Frank-Wolfe assignment recreate each individual O-D path used in the assignment in order to analyse them as appropriate, path-based algorithms analyse explicitly saved paths in the same way while UFO-based algorithms use “splitting factors” in a “single-pass” per origin without explicitly re-creating O-D paths (see 22.5.2). Note that, in practice, some of the necessary programming work may not have been done yet for certain combinations of method and analysis.

15.23.6.1 UFO vrs UFC

If, say, both .UFO and .UFC files are available for a particular network the user may have the choice as to which to use (for example carrying out a Select Link Analysis in **P1X**, 15.19, PIJA calculations, 13.3.14, or **SATCH** matrix cordoning, 12.1.12). In such cases the use of .UFO files is generally **strongly** recommended as they are considerably faster than using .UFC.

The choice of UFO vrs UFC is generally controlled by a namelist parameter USEUFO = T or F respectively which may be set in either network .dat files (which sets the general default; see 6.3.1), SATPIJA or SATCH control files (13.3.14 and 12.1.12).

In addition the UFO format used for OBA (22.5.2) may also be adapted for use with link-based Frank-Wolfe assignments; essentially the path flows in the UFC file are converted into the equivalent (or, strictly speaking, nearest equivalent) acyclic flows which are then stored in a UFO format. See 22.5.3.

UFO files, as explained in Section 22.5, have (at least) one major advantage over UFC files in that they enable “warm start assignments” to be used under all possible conditions. In addition the same analysis application may run much faster with UFO than UFC files (since any analysis of all O-D routes with UFC contains a

loop over the number of iterations N which UFO avoids so that, in principle, it may be N-times faster).

The downside of creating a .UFO file is that, if the Frank-Wolfe solution is not particularly well converged and contains many examples of cyclical flows, eliminating those cyclical flows may lead to a significantly different set of flows (although arguably they may be nearer to a true Wardrop Equilibrium solution) which causes more confusion than benefit.

Finally note that .UFO files may equally be converted into equivalent (or virtually equivalent) .UFC files using SATUFC - see 15.23.5 above – and that an alternative form of .UFO file may be created based on aggregated “spider” networks – see 22.5.3 – which is generally speaking much faster to use for analysis

15.23.7 Creating .UFO files (SAVUFO): Batch File Procedures (SATUFO)

In order to create both a UFO and UFC output file from a (Frank-Wolfe, MET = 0) run of **SATALL** it is normally necessary to have both SAVEIT = T and SAVUFO = T within the original network .dat file. (By contrast OBA always produces a UFO file as long as SAVEIT = T.) The (approximate) algorithms used to create UFO files from Frank-Wolfe assignment are described in Section 22.5.3

However, alternatively, .UFO files may be created “after the fact” if SAVUFO is not T in the initial assignment by either:

- 1) Running the procedure SATUFC (15.23.5) with an extra (text) argument UFO included in the command line, or
- 2) Running a procedure “SATUFO net” to create a file net.UFO on the presumption that the file net.ufc has already been created.

Note that SATUFC and SATUFO are both batch files which call \$SATALL.EXE with particular command line parameters in order to carry out particular functions; i.e., they are not distinct .exe files.

15.23.7.1 SATUFO: Single User Class option

A sub-option within the batch procedure SATUFO.BAT allows a .UFO file to be created for a **single** user class n by using a Command:

```
SATUFO net NOMAD n
```

in which case the output .UFO file will be named net_n.ufo.

15.23.7.2 SATUFO: Multi-core option

The SATUFO process will also take advantage of the multi-core capability within the software if available – see section 15.53 for further details.

15.23.8 Final Comments: The Uniqueness of Route Flows and Other Limitations

In applying the various analyses available within **SATURN** based on specific O-D routes users must appreciate that all such outputs must be taken with a rather large pinch of salt.

Firstly, it must be appreciated that O-D routes are **not** uniquely specified by Wardrop Equilibrium (see 7.1.6) and that the routes generated by the Frank-Wolfe algorithm (plus OBA) are, to a certain extent, arbitrary. Thus, strictly speaking, Wardrop Equilibrium only identifies those O-D routes that **may** be assigned positive flows but not the precise split of traffic between those routes. A simple example is given in Section 7.1.6 to demonstrate the non-uniqueness of origin or user class based flows between two parallel routes even when the total link flows are uniquely specified. The final assigned route flows may simply be due to some minor artefact of the algorithm used.

From which it follows – and this is the important point - that any outputs from a route flow analysis such as a Select Link Analysis or skimming, say, distance from a forest (15.27.6) are also non-unique and, therefore, prone to being arbitrary.

A knock-on impact of skimmed times, distances etc. etc. being non-unique is that any further analyses based on those skimmed matrices becomes non-unique. This therefore may introduce further problems with economic evaluation packages such as TUBA or external demand models (VDM) which are **not** based on generalised cost matrices generated by **SATURN** (which **are** unique); see also 7.8.6.

Secondly, there are potential problems with the “all or nothing” division of O-D routes into “used” and “non-used”. Thus a “rat run” route which includes a large proportion of very low capacity links may be allocated O-D route flows if it is a minimum cost route, whereas an alternative route along a series of high-capacity motorway links may not be used at all if its generalised cost is (just) 1 second above the minimum. Small changes in either the network or trip matrix may reverse either allocation.

On a more positive note one could argue that, given its sequence of operations, the Frank-Wolfe algorithm is unlikely to produce a totally unbalanced or extreme set of O-D route flows. Equally it treats all origins equally and simultaneously and will not therefore produce route flows which are “biased” by origin. Its outputs might therefore be charitably described as “plausible” – but never perfect.

The situation, however, becomes worse if we consider not just the OD route patterns in a single network but any comparison of the route flows in two networks which differ slightly from one another. Here the “noise” generated by the above two problems may render any comparisons extremely tenuous.

To a certain extent the problems with route flows are simply an inevitable consequence of the fact that more you disaggregate data the more unreliably it becomes. There are far more route flows than, say, link flows and the flows per route are much smaller than the flows per link. These problems are, however, aggravated by the problems of non-uniqueness and the lack of an acceptable behavioural model of route choice. What is required, possibly, is an extension to Wardrop Equilibrium to deal with route choice and which would operate at a far more disaggregate level than total link flows. This issue is discussed further in the following section, 15.23.9.

15.23.9 Unique Route Flows: The Principle of Proportionality

One method (in theory at least) to define a unique set of path flows within a Wardrop Equilibrium solution is to require that the path flows satisfy the

“proportionality condition”; see, for example, various articles by BarGera and others.

Proportionality requires that, at any node A where there are more than one exit links that are on minimum cost paths to another node further downstream, i.e., they represent parallel route segments with the same cost, then the proportion of trips using each segment / exit link must be the same for all origins and/or destinations. Thus, with reference to the example of two parallel links as described in Section 7.16, the 50:50 split between the two alternative path segments must be maintained for all origin and/or destination flows.

Proportionality is thus only a statement of conditions which must hold, it does not in itself provide an algorithm for achieving such a solution. However there are certain assignment algorithms which have recently been developed which do generate proportional solutions but which have not reached the stage of finished products.

An alternative “principle” for specifying a unique set of path flows is that of “entropy maximisation” where we choose a set of path flows T_{pij} which maximise the entropy measure (see equation (13.2) in section 13.1.1). To a large extent (as I understand it) proportionality and entropy maximisation generate the same solutions but there may be pathological situations where they differ. To a certain extent the differences are academic since algorithms to solve for either are hard to come by.

15.24 Alternative Link Costs and/or Times for Tree Building

15.24.1 Introduction

P1X, **SATDB** and **SATLOOK** contain various options (with a large degree of overlap) which allow “trees” or minimum cost paths to be built from an origin to a selected destination zone or indeed to all destinations. The “cost” used to build a minimum cost path is defined as a linear combination of time and distance (or distance-related parameters) as follows:

$$c = a_1 t + a_2 d + \sum b_k d_k$$

where c is the cost on a link

t is the link travel time (including any 44444 time penalties)

d is the link distance

m is a monetary toll (if any)

a_1 and a_2 are the values of time and distance respectively (generally set by parameters PPM and PPK and possibly disaggregated by user class)

d_k , $k = 1, \dots, K$ are the additional link “KNOB” properties

b_k , $k = 1, \dots, K$ are conversion factors to reduce them to a common cost (see 7.12.2).

For most applications the KNOBS facility will not be invoked and cost is therefore a linear combination of time, distance and, possibly, tolls. Weighting parameters

PPM and PPK will already have been defined by the user during the **SATURN** runs and, if the user wishes to re-create the same or similar trees, the same values should be selected. These values may also have been user-class specific. However alternative cost trees may also be investigated; e.g., you may run **SATURN** on the basis of minimum time trees but then look at minimum distance trees.

Distance, KNOBS data and tolls are, by definition, fixed. However time may be defined in a number of different ways (e.g., with or without penalties added) and calculated at different points during the programs as elaborated below in 15.24.2, 15.24.3 and 15.24.4.

Note that these discussions refer equally to “time skims” as discussed in 15.27.

15.24.2 Travel Time: Alternative Definitions

“Time”, unless otherwise qualified, refers to the time to traverse a particular (assignment) link by a standard vehicle or pcu. However in certain situations it may be necessary to consider differential times by, say, user class or by bus lane. These are explained further below, 15.24.4

For display purposes, e.g., in **P1X**, time is very often sub-divided into various sub-components, e.g., fixed times, transient delays, queuing delays, etc. etc. Equally link travel times as displayed may or may not include additional delays associated with one or all of the delays from turning movements at the downstream end of the link.

15.24.3 Calculating Times at Different Stages within SATURN

Link travel times are set at 4 different stages within **SATURN** as follows:

- 1) Free flow travel time;
- 2) Times calculated at each assignment iteration;
- 3) Travel time as calculated at the end of the assignment;
- 4) Travel time as calculated (for simulation turns only) by the simulation.

Of these (1), (3) and (4) are generally saved on the **SATURN** UF files but (2) is only (in effect) saved on a UFC file if the SAVEIT option is requested.

Trees may be built using any of the (available) times above. The following notes describe the differences between these times in greater detail and suggest circumstances in which they might be used.

- 1) Free flow times are defined within **SATNET** for all network elements (e.g., links, centroid connectors) EXCEPT for simulation turns whose free flow times are the delays calculated by **SATSIM** with zero flow on each turn. Free flow times are generally used to build the first set of trees in the assignment. These may be thought of as the “ideal” routes.
- 2) At each subsequent assignment iteration the times are set according to equations (5.1) where V is the “current” flow such that the assigned volumes at iteration i are used to set the times (and therefore the costs) at iteration i+1.

Therefore these times would be used to re-construct the routes built at each stage of the assignment procedure; they constitute the “real” routes as assigned.

(N.B. The times as defined above are only preserved on a UFC file if SAVEIT = T, so that only then can actual routes be re-created. In actual fact what are saved are NOT the times but the costs, i.e., including the fixed components appropriately weighted. Therefore it is not possible to use, say, the second set of link times in a new definition of generalised cost, although there is probably very little reason why one should ever want to do that - the important thing is to be able to re-create the routes to which trips were assigned at each iteration.)

- 3) At the conclusion of the assignment, times are also calculated using (5.1) with V equal to the final assigned flows. These are therefore the “best” times as calculated by the assignment but, somewhat perversely, they are never used by the assignment to build routes. These times should therefore be used to calculate the minimum cost routes at convergence, e.g., to calculate an O-D cost matrix for evaluation purposes.

(N.B. Although routes calculated using the above times were not necessarily generated by the assignment this is not to say that they definitely were NOT. Indeed in most cases the final routes will correspond to one and probably more than one of the routes actually generated so that they are not necessarily unrepresentative. However some care should be exercised in their analysis.)

- 4) When the simulation stage is run after the assignment the delays on turns in the simulation network are re-calculated by simulation. If the model has converged properly these delays should differ by only a small amount from those calculated by the assignment (and be identical in the event of perfect convergence). These times are somewhat more “realistic” than those calculated at the end of the assignment and therefore the “best” estimate of O-D costs would be obtained using these costs. Again the same caveat as above applies to the routes actually calculated using these costs; i.e., they do not necessarily correspond to routes generated by the assignment although they are unlikely to be “unrepresentative” routes.

Times (1), (3) and (4) can be selected by the user via menus in **SATDB**, **SATLOOK** and **P1X**. Times (2) are effectively only available through the options to “loop” over each iterative set of costs to construct each built tree in turn.

15.24.4 Extended Travel Times

The “basic” link travel times for “cars” as defined in 15.24.2 may need to be extended to include additional time components as required in certain circumstances. The need arises very often when different user classes have different speeds and, in particular, when time is being **skimmed** (see 15.27.7.1) as opposed to being used to set minimum cost routes.

Thus, by default, link time penalties as defined under the network 44444 data records (6.8) are, by default, **included** within the normal definitions of skimmed time on the basis that they are more likely to be “real” times rather than “notional” times added by the user to improve the assignment routings. (Note that the

contribution of penalties is **always** included under the definition of “cost”, e.g., used to define minimum cost routes, since it is an integral component of the fixed link costs)

Equally, any extra travel times associated with specific user classes calculated using the CLICKS options (15.47) are also included by default. (N.B. CLICKS was only introduced in 10.6.)

On the other hand there will definitely be occasions when a user wishes to exclude 44444 penalties and/or CLICKS in the definition of times. Thus in the interactive menus used in **P1X**, **SATLOOK** and **SATDB** to define time for tree building and/or skimming there are “toggle” options to explicitly include or exclude CLICKS and/or penalties.

In addition the include/exclude default options may be user set using parameters USETP and CLICKY in the appropriate preferences files (SATLOOK0.dat etc.). Thus if USETP = T then, if “time” is selected, it will include all 44444 time penalties (for that user class); similarly if CLICKY = T then times defined according to CLICKS are used in preference to “normal” times. Both parameters default to T.

15.24.5 Units of Time and Costs

As explained in 7.11.1 **SATURN** conventionally expresses all costs as “generalised time” as opposed to “generalised cost” within the assignment procedures. The same rule also applies by default to the analysis of the assignment via tree building etc., although with 9.1 options have been introduced to allow costs to be defined in units of, say, pence rather than seconds. This is particularly useful for “skimming” trees as explained in Section 15.27.

Note however that **SATURN** virtually always defines generalised cost internally in units of seconds, e.g., when carrying out elastic or variable demand calculations. Users are therefore strongly recommended to stick with generalised time unless, e.g., they specifically require costs in some other units to be exported to some other evaluation package.

15.25 Stochastic Trees

“Stochastic trees” refers to minimum cost routes between origin and destination zones (hence trees) built on the basis of using random number distributions to generate individual link costs (hence stochastic). It is therefore the process at the heart of stochastic assignment as described in Section 7.2.

All programs that allow the user to build trees - **SATLOOK**, **P1X** and **SATDB** - also enable stochastic trees to be built by setting the parameter “SUZIE” to .TRUE. Parameters SUET, KORN and KOB then control the generation of randomised link costs in the same way as they do within stochastic assignment - see Sections 7.2.3 and 7.2.4.

There are two very general circumstances in which stochastic trees are built:

- 1) To precisely reproduce the routes generated during the stochastic assignment itself; and

- 2) To generate a series of “typical” stochastic routes in order to assess qualitatively the effect of different parameter values.

In order to do (1) it is essential to have invoked the SAVEIT option during the assignment (see Section 15.23) and to then select the desired iteration number or numbers. As long as SUET, KORN and KOB are identical to those values used during the assignment then the routes SHOULD be identical. (But, N.B., this reproducibility is a function of how your program has been compiled vis a vis random numbers since there is a choice between using your own machine-dependent random number generating functions, which are probably NOT reproducible, and using a SATURN-supplied function which is. If your programs were supplied as “executables” from Leeds or Atkins, worry not; if they are home-compiled check.)

Method (2) may be used, for example, to see how large a value SUET can take before the routes generated using, say, minimum time as the basis, become unrealistic.

15.26 Trees, Forests and Arboreta

A “tree” refers to the set of shortest routes from one origin to one (or all) nodes/zones in a network. As such a separate tree is calculated on each iteration within an assignment. A “forest” is therefore an aggregation of all the trees from a single origin over all internal assignment iterations, weighted by the fraction of the trip matrix as ultimately assigned to each iteration.

More precisely the “forest” value for a link is the proportion of trips from a particular origin to a particular destination which use that link. It is therefore virtually identical to the “Pija” factors as used by SATME2, although used in different contexts

Forests are a highly preferable method of analysing O-D routes for the simple fact that they contain information about ALL routes assigned traffic for that O-D pair, as opposed to looking at the single route which is currently minimum cost at the end of the assignment process and which may even not have been used in the assignment itself.

Trees may be built in a number of different ways within P1X, SATDB or SATLOOK, although the graphical methods within P1X (11.8.3) are generally recommended.

Forests may be built in either P1X (graphically) or SATDB (numerically). Equally they may be built under both stochastic and Wardrop equilibrium style assignments. (But see remarks in Section 7.2.4 concerning the consistent re-creation of randomised costs).

By contrast an arboretum is defined to be the set of all different routes used by a single O-D pair; i.e. the complete set of different trees. Thus if an assignment takes 20 iterations it generates 20 trees of which only 5 of these may make up the arboretum. The “arboretum display” option in P1X displays each tree one at a time with data on the fraction of all trips using that route (summed over all iterations or trees). Because it uses fewer displays it is preferable to displaying trees one at a time.

(N.B. An arboretum is essentially a Frank-Wolfe algorithm based construct for which there is no direct equivalent under OBA.)

Note as well that forests and arboreta can only be constructed IF the SAVEIT option is in effect - see Section 15.23.

15.27 Skimming Trees and/or Forests

15.27.1 Minimum Cost Trees and Matrices

Trees (15.26) represent the full set of minimum “cost” O-D paths, where cost is some criteria such as time, distance, monetary cost or, most often, generalised cost (i.e., a weighted combination of two or more components such as time and distance) which the individual O-D paths (or “routes”) minimise. They are called “trees” since, if you plot the set of minimum cost paths from one origin to all destinations, the resulting sub-network resembles a tree which continuously branches outwards from its root/origin such that there is only one possible path to each destination D. The cost along the single path to D is therefore the minimum cost to D and a “tree” is therefore synonymous with “minimum cost O-D path”.

Clearly trees depend on how “cost” is defined: the path that minimises time between O and D is not necessarily the same path that minimises distance nor the one that minimises generalised cost.

Note that in the vast majority of transport models “cost” is synonymous with “generalised cost” so that we may distinguish cost from its sub-components such as time and distance.

A “minimum cost matrix” is the complete matrix of O-D minimum costs as extracted from the tree.

15.27.2 Skimming Trees

To “skim” a tree is to sum a particular “quantity”, e.g., time, distance, toll, etc., etc., link-by-link along the minimum cost paths for each O-D pair. For example, we may wish to calculate the distance along the O-D path which minimises cost. Therefore we distinguish between the quantity which is used to build the tree and the quantity which is skimmed. Skimming is very often an essential step in scheme evaluation.

Within **SATURN** skimmed O-D matrices may be obtained in two different ways: via trees (as described here) or, much more usefully, via “forests” as described in 15.27.3.

Trees may be skimmed to produce “skimmed matrices” as part of the tree-building option 14 within **SATLOOK**; thus the ij-th element in the output matrix is, e.g., the distance from origin i to destination j as summed along the links in the minimum cost (time) path from i to j.

Note that using a “tree” to produce, say, distance skims does not necessarily accurately reflect the average assigned distance for trips between a given O-D pair in those cases where several different routes with several different distances are used within the assignment process. Skimming a tree only gives one particular route distance and indeed other routes may give lower or higher

distances so that the “true” average distance may also be greater or less than a skimmed tree.

The problem becomes more acute with quantities such as tolls which are much more “off or on”. Thus if an O-D pair uses two routes, one of which is tolled and the other is not, then it is somewhat hit or miss whether the single skimmed route gives zero toll or the full toll.

WARNING! Skimming O-D data from single path trees may therefore produce unreliable or misleading results. The only quantity which may be skimmed absolutely unambiguously from, say, the minimum cost tree is cost itself.

In fact the only arguable advantage to skimming a tree as opposed to skimming a forest is that it is much faster – only one tree needs to be built per origin as opposed to a forest skim where separate trees have to be built for each Frank-Wolfe iteration; e.g., it may be 50 or 10 times more time consuming.

Cumulative link/node skims may also be obtained for individual links using the tree building option within **SATDB**; in this case a distance skim, for example, would be the summed distance from the origin to the end of a link. Similarly in **P1X** time and distance skims are automatically accumulated for O-D trees plotted continuously there.

15.27.2.1 Default O-D Costs

In the event that a particular o-d pair is **not** connected, for example due to the origin having no out-bound centroid connectors, the skimmed cell in the matrix takes on a default value of zero by default. Logically it might be argued that, if it is impossible to get from o to d, the default value should be a very **large** value. On the other hand, if the trip matrix contains a positive value in that cell (for whatever reason), multiplying the trip matrix by the skimmed matrix would yield very large numbers for the unconnected cells which may swamp the “correct” cells as part of an economic evaluation of total vehicle-costs. The default value may, however, be changed by the user within the **SATLOOK** interactive menus or via a parameter DEFODC in SATLOOK0.dat.

15.27.3 Skimming Forests

As noted above (15.27.2) skimming the single minimum cost tree to produce, say, distance may be unreliable and/or misleading. An alternative procedure – option 9 within **SATLOOK** - is to skim a “forest” (see 15.26) whereby the distance (say) is calculated using the **exact** routes used on iterations 1, 2, 3, etc. and a correct weighted average distance obtained.

The one basic option within a forest skim is to nominate the quantity to be skimmed. Generally, but not always, the quantity skimmed will be a sub-component of the generalised cost used to define the forest, e.g., time, distance, toll, etc. Alternative options exist to either:

- ◆ construct the skimmed quantity from elements in the base network file;
- ◆ select a link property which is stored in a **SATDB** data base column (which in turn may be read in from an external ascii file) (N.B. This option is only

available when **SATLOOK** is accessed via **P1X** and therefore **SATDB** may be accessed as well; see 11.11.19);

- ♦ if a second network file is defined which is topologically identical to the first, then a DA array from that file may also be nominated. This enables one to, e.g. skim average times on network 1 paths based on the times calculated in network 2 (new in **SATURN** 10.1).

In principle, therefore, it should be possible to set up properties such as fuel consumption or accident rates per link and skim them in order to create O-D matrices of fuel consumption or accident rates.

In addition there are a number of sub-options to modify the precise definition of certain skimmed quantities. For example penalty times input under 44444 may be optionally included or excluded, extra time components associated with specific user/vehicle classes under CLICKS may be in/excluded and times/distances on centroid connectors may be deliberately excluded (XCCSK; see 15.41.5).

Note that a forest skim is only possible with SAVEIT = T and also, since it involves building and skimming one tree per iteration within the "SAVEIT assignment", it may take considerably more cpu time than skimming a tree. However the results obtained will generally speaking be more accurate and are recommended for use in matrix-based evaluation such as TUBA.

In particular a forest skimmed matrix satisfies the condition (but see 15.27.5 below) that:

$$\sum_{ij} T_{ij} S_{ij} = \sum_a V_a S_a$$

where the left hand side of the equation represents the total, say, vehicle-kms summed over ij pairs and the right hand side represents the same quantity summed over links. S_{ij} and S_a refer to the property being skimmed e.g., distance.

Note that some care needs to be exercised in the definition of V_a in the above equation since it must be: (1) the link flows from the trip matrix itself (e.g., excluding any fixed link flows); and (2) demand as opposed to actual flows. We return to this point in the following section.

We may also note that forest and tree skims will give identical results for O-D pairs where the assignment has only generated a single route. Typically this occurs for very near O-D pairs or for very uncongested sections of the network.

For further information on SAVEIT and forests please refer to sections 15.23 and 15.26.

We also note one additional caveat with forest skims which is that, since the path flows generated by a Wardrop Equilibrium assignment are not, strictly speaking, unique (see 7.1.6), neither are skims taken over those paths (see 15.23.8). The example given in 7.1.6 as to which origin(s) pay tolls illustrates the problem – although, in practice, the problem is much more likely to be that the model shows the two origins paying tolls in slightly different proportions rather than the extreme "all or nothing" case where one origin pays tolls and a second does not. Note (7) in 15.27.6 discusses this issue further.

15.27.4 Minimum Cost Matrices vrs Skimmed (Average) Cost Matrices

In effect minimum cost matrices – the matrix of minimum possible costs from O to D 15.27.1) - are a particular example of matrices skimmed from a single tree whereby the quantity which is skimmed (summed) is the same quantity which defined the minimum “cost” paths. However, since the minimum O-D travel costs are obtained as an integral part of the tree-building process, it is not necessary to do a subsequent “skim” which would clearly give identical results.

However it is also possible to construct a matrix of costs by skimming a forest where in this case the “cost” refers to the “generalised cost” used in the assignment process (see 7.11.1) which, in turn, are the same costs upon which the forest of assigned O-D routes are based. In this case the skimmed cost will be the weighted **average** cost over all O-D routes used.

We recall that Wardrop’s Principle requires that at equilibrium all used routes have equal and minimum costs so that – in the case of **perfect** equilibrium - the minimum cost matrix will be identical to the cost matrix obtained by skimming the forest. For less than perfect convergence (the inevitable norm) the minimum costs will be (potentially) less than the costs along **some** assigned routes and therefore less than the average.

In certain respects, assuming the inevitable less than perfect convergence, the forest cost matrix is “better” than the minimum cost matrix since it corresponds to the costs actually incurred according to the assignment; it might, therefore, be better to use within economic evaluation. On the other hand it requires considerably more CPU time to calculate. And, strictly speaking, neither is “correct” in the sense of being equal to the cost matrix which would be obtained under perfect convergence; in general one would expect that the minimum cost matrix would be a slight under-estimate of the “true” ultimate cost matrix and the average would be a slight over-estimate.

Thus, faced with two alternatives, neither of which is correct but one of which is much cheaper to calculate, there is a strong case to be made for choosing the cheaper alternative, i.e., to take cost matrices as equal to the minimum cost matrices rather than the forest skims.

Note, however, that this advice does certainly not apply to any **sub**-components of cost such as time or distance which can only be obtained reliably as skims over forests as opposed to, say, building a minimum time tree or skimming a minimum cost tree.

Finally, there is a third method by which cost matrices may be constructed. Thus, if the cost per link is a weighted combination of, say, time and distance such that:

$$C_a = w_1 t_a + w_2 d_a$$

Then, if we take forest skimmed matrices of time and distance t_{ij} and d_{ij} then we can also obtain the average cost matrix C_{ij} via:

$$C_{ij} = w_1 t_{ij} + w_2 d_{ij}$$

This method is sometimes necessary if, for example, the user wishes to define different values of the weights w_1 and w_2 (se 7.8.6)

15.27.5 Skim/Cost Matrices and Trip Matrices

It needs to be appreciated that skimmed and/or cost matrices (in general) are calculated entirely independently of any considerations of demand and actual flows on individual links. Thus the O-D “distance” in a distance matrix is the sum of the individual link distances along a particular path or paths whether or not the O-D trips use those links in the “current” or a “later” time period. Demand and actual flows play no role.

This may have certain implications for matrix-based evaluation procedures and, in particular, for the interpretation of the “product” of a trip matrix and a skimmed matrix; i.e., under what conditions will the following equation hold:

$$\sum_{ij} T_{ij} S_{ij} = \sum_a V_a S_a$$

where the left hand side of the equation represents the total, say, vehicle-kms summed over ij pairs and the right hand side represents the same quantity summed over links (and displayed in output tables such as Table 17.3 as described in Section 17.9).

Firstly, such an equality never holds if the flows V_a correspond to **actual** link flows and the summation corresponds to, say, total pcu-kms within “this time period” (see Table 17.3). In particular, there is no such thing as an “actual” trip matrix for use on the left-hand side, only a demand matrix.

Secondly, if the matrix S_{ij} has been skimmed from a (single path) **tree** then the equality will not hold in general (unless, most unlikely, all O-D trips have been assigned to single, not multiple, paths).

The equality **may** hold if: (a) T_{ij} is a demand matrix; (b) S_{ij} has been skimmed from a forest and (c) the link summation includes both this and the following time periods (the “Totals” column in Table 17.3).

However, even these conditions may not be sufficient to guarantee **exact** equality. For example, if the link flows V_a are obtained from the “true” model run and the skimmed matrix is based on a forest obtained using an approximate SAVEIT assignment (15.23.2) then the equality will only be approximate, limited by the lack of perfect convergence within both the true and the SAVEIT assignment. More specifically, if S represents time then the parameter AFTERS must equal 0.5; see 17.6.2.

Conditions becoming slightly easier if the quantity S refers to the “generalised cost” used by the assignment as opposed to a particular component such as time or distance (or if, say, the assignment is based on pure time). In this situation the skim matrix S_{ij} is effectively the same whether it is obtained as a minimum cost matrix or as a forest skim to the extent that under perfect Wardrop Equilibrium all used paths have equal and minimum cost. Clearly if the convergence is not perfect then the equality will be only an approximation.

15.27.6 Summary: Minimum and/or Skim Matrices

This section summarises several common sources of confusion experienced by users faced with the problem of producing “cost” and/or skimmed matrices (in general).

- 1) A matrix of, say, O-D distances (times, tolls, etc.) may be produced in at least three different ways (where we assume first that the assignment is **not** based on minimum distance):
 - a) building minimum distance trees,
 - b) skimming distance along the (single path) minimum cost trees,
 - c) skimming average distance along the (forest of) multiple paths used within the assignment.

These correspond to minimum cost matrices (Option 14 in **SATLOOK**), skimmed matrices (Option 14 in **SATLOOK**) and forest skims (Option 9 in **SATLOOK** and/or batch files such as SKIMDIST (15.27.7)) respectively. It is important that users understand how these three types of matrices differ.

- 2) Of the above three methods, the third, i.e., the “forest skim” (15.27.3), is almost certainly the most realistic since it corresponds most closely to the “true” assigned paths, generates fewer problems in terms of uniqueness, reliability etc. etc. (but not **NO** problems – see points 6, 7 and 8 below). However it may take considerably more cpu – see point 9 below.
- 3) The first, “true” minimum distance (etc.), has the advantage of being unambiguously defined but, on the other hand, it may not correspond to a route that would actually be used by drivers. However, it might be useful to compare a “true” minimum distance matrix to the “actual” distance matrix to see how near users are to minimum distance.
- 4) Equally skimming distance (etc.) from a single minimum-cost tree is highly unreliable (see 15.27.2). However skimming a single tree may be much faster than skimming a forest and, if only an approximate answer is required, may be sufficient.
- 5) On the other hand, if we are interested in a “cost” matrix for use, say, in an external demand or evaluation model, where cost refers to the generalised cost used in the **assignment**, then methods (a) and (b) above give identical results while (c) differs only in terms of non-convergence. Indeed, if we have achieved perfect Wardrop Equilibrium, cost should be equal on all routes used and (c) must give the same result as (a) and (b). In practice deviations occur due to a lack of perfect convergence so that (a) must give lower values than (c). The answer which would be obtained ultimately by a perfectly convergent Wardrop Equilibrium solution will (almost certainly) be somewhere between the two values. Method (a) is therefore recommended unless one is specifically interested in the **average** O-D costs.
- 6) When skimming from a forest, under the standard Frank-Wolfe method of assignment as opposed to path-based or OBA, the level of convergence achieved by the extra SAVEIT assignment (if one is required) becomes an issue (15.23.2). Thus the routes generated by a SAVEIT Forest will not, in

general, reproduce the same routes and the same link flows as generated by the assignment proper; they are only an approximation. The differences are reduced by better convergence (i.e., increased values of NITA_S) which leads to more accurate skims but it also means more cpu time. Compromises may be required.

(N.B. These problems do **not** arise if the skimmed forests are based on the **actual** assignment routes as opposed to an extra SAVEIT assignment; e.g., if $UFC109 = T$; see 15.23.3.1)

- 7) Since the precise route flows generated under Wardrop Equilibrium are **not** unique (see 7.1.6 and 15.23.8) neither are the **average** (forest-weighted) O-D times, distances etc. cost components which are skimmed from them, even if convergence were perfect. In addition the average OD speed matrix obtained by dividing average distance by average time would not be unique either. This may have implications for the convergence of linked supply and demand models (7.8.6) or for economic evaluation models where the demand/evaluation model is based on a **different** definition of generalised cost from the assignment model and therefore requires separate skims of time, distance, etc. The problem is most acute if a high degree of convergence is required. It may also have implications for economic evaluation models when applied to relatively small schemes when any source of “noise” in time and distance etc. matrices is a problem.
- 8) The problems of non-uniqueness may be further aggravated by the presence of “residual path flows” in the solution used which may make skimmed quantities considerably more unreliable than, say, flows. See 15.57.
- 9) The problems of non-uniqueness for time and/or distance skims may be particularly evident when comparing skims from two different schemes where, for example, there may be large differences in individual O-D times or distances when none might be otherwise expected. These differences may be due to network 1 using an arbitrarily different set of OD routes from network 2 and, if there is a large degree of variability between the times/distances on the alternative routes (even though they may have identical generalised costs), then there will equally be a high degree of variability in the time/distance skims.
- 10) The degree of variability may also depend on the relative cost “weights” (i.e., PPM and PPK) assigned to time and distance. Thus if PPK were near zero, implying that distance is not very important in choosing minimum cost routes, and an O-D pair is assigned to two routes, one with short distance and one with long, then the skimmed distance could be anywhere between the minimum and maximum distances depending on the essentially arbitrary split between the two routes. On the other hand, the **time** skims in this situation would be far more stable since time would be effectively equal to cost and the costs on the two alternative routes would be equal. Conversely if PPM is small then distance skims will be stable and time skims more variable. Note, therefore, that different user classes which have different values of PPM and PPK may well behave differently.
- 11) For assignments based on Frank-Wolfe (as opposed to OBA) calculating a **minimum** cost matrix requires one tree build operation per origin; skimming

an **average** matrix from a forest requires one tree build operation per origin **per** assignment iteration. Hence it may require 25, 50, 100, etc. times more cpu time depending on the value of NITA_S. For large networks this time may be significant. (Note that this does **not** apply to OBA since skimming under OBA requires only a single pass; see 22.5.6)

15.27.7 Skimming Costs Using .Bat Files (E.g., SATCOST.bat)

A number of useful standard .bat files have been created within **SATURN** in order to simplify and automate the creation of minimum and/or averaged “cost” etc. matrices.

Thus the .bat file **SATCOST** automatically extracts the **minimum** (as opposed to average) cost matrix (as defined in 15.27.1) and is available both within DOS and **SATWIN**. For example, the command:

```
SATCOST net cij
```

Generates the matrix of minimum o-d costs for network net.ufs and stores them in cij.ufm. Type “**SATCOST**” for full filename conventions. This is usefully coupled with elastic assignment to generate cost matrices for external demand models (see 7.8.6).

If the input network contains multiple user classes the calculations include **all** user classes and the output matrix is a stacked matrix, one “level” per user class. Similarly if the input network has an extension .uft., i.e., it represents the outputs of multiple time periods, then the output matrix cij.ufm will be “stacked by blocks” with each “block” (see 10.2.4) representing the o-d costs for a particular time period.

Note that **SATCOST** - and all the variants below - produce cost matrices defined in units of generalised seconds which are therefore compatible with the units used within all elastic or variable demand models.

Equally note that the units are, effectively, O-D travel cost per pcu and bear no relationship to the trip matrix or any factors used, e.g., to factor trip matrices. For example, if you have a model with a single total trip matrix equally divided into six user classes, then **SATCOST** will create a stacked .ufm matrix with six levels, one per user class, each approximately equal to the cost matrix for a single user class. Thus the fact that the trip matrix is divided by six per user class does not imply that the cost matrices are equally factored.

Several alternative bat files are provided based on Forest Skimming (Option 9 within **SATLOOK**; see 11.11.9). In particular:

- ◆ SATC_AV skims average costs from a forest (15.27.3)
- ◆ SATC_MAR skims marginal costs (but only from networks which were assigned under system optimal conditions; see 7.11.9).
- ◆ SATC_TP produces a minimum cost matrix (à la SATCOST) but over multiple time periods (see Section 17).
- ◆ SKIMTIME skims averaged o-d times (in seconds) from a forest as described in 15.27.3.



- ◆ SKIMDIST skims averaged o-d distances (in metres) from a forest.
- ◆ SKIMTOLL skims averaged o-d tolls (in pence; see Section 20.3) from a forest.
- ◆ SKIMPEN skims averaged o-d time penalties (in seconds) as defined within the 44444 data records from a forest.
- ◆ SKIM_ALL combines SKIMTIME, SKIMDIST, SKIMTOLL and/or SKIMPEN into a single routine which skims all 3 or 4 quantities “simultaneously” which means that the CPU time is reduced by a factor of roughly 3/4 (or 2 if there are no tolls included in the definition of generalised cost). Time penalties are only output if they exist.
- ◆ SKIMDA skims a particular property identified by its DA code, hence a general purpose skim routine which could be used to skim time, distances, etc. etc. by using the requisite DA code
- ◆ SATTUBA skims time, distance and/or tolls directly to a series of ascii files in various TUBA formats; see Section 15.41 for further details

For further details on file format conventions etc. please type the names above.

Note also that routines SKIMTIME to SATTUBA as listed above may all take advantage of multiple processors if available; see 15.53.3.2.

15.27.7.1 The Definition of Skimmed “Cost”

The following notes may help clarify exactly how the skimmed “cost” is defined in certain of the above batch files.

Under SKIMTIME, times are equal to the “real” times along links and/or turns plus, by default, any penalty times which may have been added under the 44444 restrictions (see 6.7). However, post 10.6, the 44444 penalties may be **excluded** if a parameter USETP is set to F in the preferences file SATLOOK0.DAT. Equally, if the CLICKS option is being used, the link times will use the CLICKS rules if a parameter CLICKY = T in SATLOOK0.DAT which, by default, it is. See also 15.24.4. Finally skimmed times on centroid connectors may be excluded (by setting them to zero) if a parameter XCCSK = T in SATLOOK0.dat (15.41.5).

In addition times under SKIMTIME are those calculated by the final **simulation** as opposed to those calculated by the final assignment (DA code 4013 rather than 4003).

XCCSK also applies to SKIMDIST; i.e., if T skimmed distances on centroid connectors are assumed to be zero.

Tolls under SKIMTOLL are in units of “pence” (or, strictly speaking, defined by the parameter COINS) and include all monetary toll components whether defined under the 44444 records or as a KNOBS input. See Section 20. (N.B. Since tolls in the sense of explicit monetary tolls were only introduced in version 10.3 SKIMTOLL cannot be used with files created prior to 10.3.)



By contrast with SKIMTOLL SKIMPEN also extracts data from the 44444 data inputs but only those elements which been defined as “times” rather than “money”; i.e., those without a \$ or £ sign. See 6.7.

15.27.7.2 *Skimming Using Aggregated (SPIDER) Networks*

If SPIDER = T and the network has been built using an aggregated network definition (see 15.56) then the algorithm used to build minimum cost trees may be based on either the basic or the aggregated network depending on whether a parameter USESPI = F or T respectively. The default value set by the program is F but it is generally over-written by the value within the preferences file SATLOOK0.DAT (which may be set by the user). Or see the sub-section below for a further method for setting USESPI.

The resultant skimmed matrices are the same whether or not the aggregated network is used; the main difference is that the aggregated method requires significantly less CPU time.

The use of aggregated networks applies to **both** skimming a single tree as well as to forest-based average skimming. For forest-based skims the potential CPU time reductions are significant (e.g., 10 times faster); however, for a single tree build operation per origin/user class, not multiple iterations, CPU time is less of an issue here in absolute terms.

Note, however, that at the time of writing the possibility to use aggregated skimming applies to **most** – but not all – applications of skimming. Its use is being gradually extended and will eventually cover all applications. Information within the .LP files should hopefully make it clear whether it is being applied or not.

For further information see 15.56.7.1.

15.27.7.3 *Command Line Over-rides for, e.g., the use of .UFO files*

The command lines associated with procedures such as SKIMDIST, SKIMTIME etc. etc. may also be used to “over-ride” certain default skimming options. For example, the choice of whether or not to use a Spider Web network representation rather than the normal network (assuming, of course, that the SPIDER network has been created in the first place) is normally controlled by a parameter USESPI described above and which may be set in the preferences file (e.g., SATLOOK0.DAT). However, rather than changing the value of USESPI in the preferences file, the choice may be made by including the “token” USESPI on the command line. Hence:

SKIMDIST net matrix USESPI

Requests a distance skim on net.ufs with skimmed output to matrix.ufm but with the parameter USESPI definitely set to .TRUE. independent of its default value and/or any value set in the preferences file SATLOOK0.DAT.

Similarly the command:

SKIMDIST net matrix USEUFO

Requests the use of a .UFO file for skimming rather than a .UFC file (again assuming, of course, that both .UFO and .UFC files are available). In this case the default choice is set by USEUFO as defined within the network .dat file.

Other command line “tokens” include:

- ◆ USEUFC – use .UFC in preference to .UFO;
- ◆ NOT_USEUFO – do **not** use .UFO (and hence equivalent to USEUFC);
- ◆ NOT_USESPI – do **not** use a Spider Web network and
- ◆ NOT_USEUFC – use .UFO instead of .UFC.

These options were first introduced in release 11.1.11 in July 2012.

15.27.8 Post 10.9.17 Skimming Algorithms (NUSKIM = T)

Releases 10.9.17 and beyond include a new set of algorithms to carry out OD skimming within **SATLOOK** which should be more cpu-efficient than the older versions. Essentially they employ a “once-through” algorithm rather than tracing each O-D path separately.

The new algorithms may be invoked by setting a namelist parameter NUSKIM = T in the preferences file SATLOOK0.DAT. The default is, provisionally, T.

Alternatively the “preferences” option may be invoked in the command line to define an alternative “local” preferences file rather than over-writing the “master” version. For example:

```
SKIM_ALL net mat PREF mylook0.dat
```

Substitutes the preferences file mylook0.dat (which should be in the same folder as net.ufs etc.).

15.28 Variable Program Dimensions

SATURN is available in differently compiled .exe files, each allowing for a different maximum problem size. The smallest standard array size is version B with intermediate versions available up to the largest X7 – further details are listed below:

Array / Level	Simulation Junctions	Assignment Links	Zones
B	500	7,500	400
C	1,000	22,500	800
S	1,500	32,500	1,200
H	2,000	47,500	1,600
K	2,500	60,000	2,000
L	3,000	73,500	2,000
M1	3,500	83,500	2,000

Array / Level	Simulation Junctions	Assignment Links	Zones
M2	4,000	93,500	2,000
M3	4,500	103,500	2,000
N1	5,000	112,500	2,000
N2	9,500	120,000	2,000
N3	21,000	200,000	2,000
N4	23,000	200,000	4,000
X7	30,000	250,000	5,750

The above table provides a quick reference guide to the principle variations between the different licence levels but other constraints – such as the number of simulation links or turns - will also determine the licence level required to run specific models.

Beyond Level 'K', the number of zones available is capped at 2000 to reduce excessive memory requirements. If a larger version is required, please contact Atkins to discuss your specific requirements.

Pre 11.2 several variants of Level 'N3' were created to accommodate the suite of sub-regional models developed by Transport for London with various **bespoke** configurations to accommodate their specific requirements. With the release of 11.2, the internal array dimensions were restructured to provide a new Level 'N4' to meet all their anticipated requirements but within a much smaller RAM footprint. Therefore, there may be some issues of backward compatibility for very large networks using SPIDER Network Aggregation and the standard 'N3' will not necessarily be capable of running the TfL Sub-Regional HAMs and an upgrade to Level 'N4' will be required.

The values of the above dimensions for a particular set of executables may be established via Help/About in the **P1X** menu bar or the full set is contained in the .lpn output files from **SATNET**.

Note that one particular array dimension, that controlling the maximum size of a trip matrix within **SATALL**, may be effectively increased in size by the judicious use of a parameter SPARSE; see 7.11.12.

Further details on the financial implications of upgrading your existing version are may be found on the website (www.saturnsoftware.co.uk).

15.29 Comment Cards and Blank Records in Data Files

In theory any ASCII file used as input to a **SATURN** program, e.g., the .dat file input to **SATNET**, may contain comment cards indicated by a '*' in column 1. Any such records are ignored and the next record read. This complements the use of '*' in the namelist input conventions to indicate comments at the end of a line - see Appendix A. This convention was first introduced in **SATURN** 9.1.



For network data files comment cards are particularly useful for, e.g., identifying specific nodes, inserting comments when changes are made (impresses the QA boys!) or for “editing out” previous coding.

In practice the convention may not be 100% fool-proof as the new rule has meant changing every single “read” statement to check for the “*”; almost certainly some will have been overlooked. It should be fairly obvious when this happens - most likely the program will crash - so the obvious solution is: (a) remove the comment card, and (b) politely alert your friendly **SATURN** agent.

The same convention applies to other input ASCII files - in particular, .key files may have comment lines inserted as may the standard graphics system file “graf.dat”.

Blank lines in input data files are, generally speaking, handled in the same way as comment cards, i.e., if read they are ignored and the next record read in its place. If, on the other hand, they are allowed as input numerical records (see below) they are interpreted by FORTRAN as a string of zeros.

Their (intentional) use is **not** recommended at all, in particular since there are exceptions to the above rule. For example, numerical KNOB data contained on extra lines in network .dat files may legitimately contain all zero entries and be correctly represented by a blank line (15.14.5). Equally key files and GRAF.DAT may contain all-blank records. There may be other examples but we haven’t thought of them yet!

Prior to 10.5 blank lines were not explicitly detected and could give rise to fatal errors, e.g., if they were meant to contain node numbers.

15.30 The Use of Sub-Files within Data Files: \$INCLUDE

Certain data sections within, e.g., network .dat files allow “sub-files” to be referenced by inserting a record containing the characters ‘\$INCLUDE’ starting in column 1 followed by a file name which should be read at that point. For example the sequence:

```
66666
$INCLUDE metro.bus
99999
```

in a network .dat file requests the program to read the bus route data from a file ‘metro.bus’.

Note that the filename need **not** be enclosed in inverted commas, i.e., metro.bus, not ‘metro.bus’, unlike filenames etc. which are specified within Namelist inputs (see Appendix A). However, if they are, the ‘s are removed and a warning printed in the .LP file.

Note that the file “metro.bus” should not contain the opening ‘66666’ record but should contain a closing ‘99999’ record which indicates only that reading reverts to the original file at that point. (Strictly speaking the 99999 record is optional as an end-of-file has the same effect; however the use of 99999 records is strongly recommended if only to positively affirm that this is the end of the desired data.) The original file must therefore also contain a ‘99999’ record in the normal way to indicate the end of a data section.

The facility is available within **SATNET** to read any of sections 1 through 8 in the network input data files. It is being gradually extended to other programs and/or files (e.g., counts in **SATPIJA**, see 13.2.1) and may also be used within Namelist inputs; see Appendix A.

It may also, post 10.9 be “subscripted” so as to apply to a particular time period under multiple time period modelling (PASSQ; see 17.4.4) in **SATNET**. For example:

```
$INCLUDE(1) bus1.dat
$INCLUDE(2) bus2.dat
```

in a network .dat file would indicate that two different sets of bus routes were to be included in the first and second time periods. See Appendix B.

An example of a network .dat file which makes extensive use of sub-files is given below:

```
.....
44444
$INCLUDE 444.DAT
99999
55555
$INCLUDE XY.DAT
99999
66666
$INCLUDE BUS.DAT
99999
77777
$INCLUDE COUNTS.DAT
99999

77777
  45   53   52  826   60
  32   33       1500  70
*  COMMENT
  33   34       1600  80
   7    8       800   90
99999
77777
$INCLUDE COUNTS3.DAT
99999
```

There are many possible benefits from using sub-files. For example if you have a large number of networks in a certain study, all of which have the same co-ordinates, it is much simpler to update a single .xy file than to update every single network file when you wish to make changes. Clearly the resulting .dat files use less disk space as well.

Sub-files may also be created and/or extended interactively using **P1X**; see Section 11.9.2.6 and 11.9.2.7.

Finally we note that it is possible – and often highly desirable – to have effectively the same data appearing in more than one file. For example, data for the same simulation node may appear in several locations such that one may deliberately take precedence over another as part of coding alternative scheme and/or scenarios. See Section 6.15 for advice on using FIFO, TOPUP and DOUBLE options.

15.31 Setting “Optimum” Stage Green Times

15.31.1 Background

A common problem in setting up future-year **SATURN** networks is to determine appropriate signal setting parameters. The same problem does not arise with current networks since, in theory at least, the settings may be observed. However the easy solution of carrying present day settings forward into the future is clearly fraught with errors since there is no guarantee that “good” settings for today’s traffic levels will still be “good” in the future. The same problem also occurs in present-day networks when network changes are tested.

The two main parameters of concern here are stage green times and offsets. Cycle times generally have a smaller influence and anyway can be set as a universal parameter LCY; inter-green times are generally fixed by reasons of safety etc. The question of setting optimum offsets is discussed in Section 12.2 with respect to **SATOFF**.

However the problem of determining optimum stage green times is considerably more complex than that of optimising offsets due to the potentially highly sensitive feedback between stage times (which affect capacities) and flows. Basically if one sets optimum green times for a pattern of flow which is in Wardrop Equilibrium given the “old” green times, those flows will no longer be in equilibrium since those routes which have been allocated more green times will have become faster. We must therefore reassign in order to take account of the latest green times. However this will tend to put more flow down those links that were given extra green time and therefore, if we re-optimize the green time in accordance with the new flows, the more heavily loaded links will tend to be assigned more green time. And more green time tends to mean more flow -a vicious circle is thereby established.

Considerable research work has gone into the investigation of the “Iterative Optimal Approach” whereby a loop is established between Wardrop assignment and signal optimisation using a number of different signal control policies as given in section 15.31.3. Interested readers are referred to the classic tome on the subject “Route Choice and Signal Control”, Avebury Press, by Tom van Vuren and Dirck Van Vliet. Under certain circumstances this approach can lead to considerable reductions in total travel time, eg up to 20% compared to the initial (and therefore potentially arbitrary) settings in the base network. However a closer examination of the process shows that this is often obtained via a process in which flows and green times move in small steps in consistent directions with the process only terminating once the stage times reach their minimum values. Such solutions are also characterised by near “all-or-nothing” flow patterns whereby very high flow rates with corresponding near maximum green rates occur on certain well-defined corridors whereas parallel routes are virtually unused. These solutions argue: (a) a large degree of co-operation between drivers and signal setters and (b) that drivers can detect and react correctly to very small shifts in green times.

It is therefore our belief that such solutions are not entirely realistic and may actually over-estimate the level of performance of a network. Therefore the use of an optimal iterative strategy must be viewed with extreme caution. See also 15.31.4.

On the other hand **some** reaction of signals to altered flow is clearly necessary. Perhaps a good compromise for future year networks is to first set signals using “engineering judgement”, carry out one full assignment followed by a stage time optimisation and one more assignment (where by “assignment” in this case we refer to a full run of **SATURN** with assignment/simulation loops internally).

If the improvements in total network travel time are significant, this procedure could be repeated, always bearing in mind the possibility of producing unrealistic flow and green time patterns and even deterioration of overall travel times.

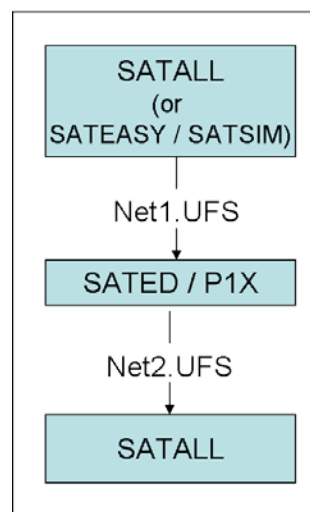
There is another possible need for a fully automated approach; this is the case where a network is not yet in existence, and initial green times can be determined to impose a preferred flow pattern on traffic. The traffic engineer then has the freedom to pursue the iterative loop until the signal settings are found that lead to the lowest network travel times.

15.31.2 Optimum Stage Times using PIX

In order to optimize stage green times a special option has been included within the **P1X** Network Editing options (see 11.9.13) to automatically consider **all** signalised junctions and to optimise all green times (using options as detailed below). This option is to be found within Global Operations on Signals and would normally be followed by the creation of a new UF file and/or .dat file containing the updated signal settings.

N.B. This replaces similar options previously available under option 1 of the now discontinued program **SATED**

An illustration of a “typical” sequence of programs is given below.



Having re-assigned and re-simulated via **SATALL** the option of course exists to loop back through **P1X** in order to re-optimize the signals - subject to the caveats expressed in Section 15.31.1.

The optimisation process may also be carried out at selected nodes only (11.9.13.2).

Alternatively, a new “batch procedure” **SIGOPT** has been introduced in Release 10.8.16 to optimise stage times and/or offsets and which effectively supersedes both **SATED/P1X** in terms of stage times and **SATOFF** in terms of offsets - described in detail in 15.31.6. Thus, in the above diagram, substitute **SIGOPT** for **SATED/P1X**.

15.31.3 Stage Length Optimisation Algorithms

Five basic algorithms to optimise stage green times are provided:

- 1) **SATURN** Equi-saturation.
- 2) Webster.
- 3) Delay minimisation.
- 4) P0.
- 5) **SATURN** Equi-saturation Mark 2.

The first is the traditional algorithm provided for many years within **SATURN**; the last is a recent modification thereof, while the remaining three were first introduced in **SATURN** 9.2, having been converted from versions programmed for **SATURN** 8 as part of a research project. They are provided primarily for experimentation and their reliability cannot be guaranteed. Their choice is governed by the Namelist parameter MYTVV set in the network .dat files with a default, post 10.9, of 5 (previously 1).

The basic equi-saturation policy essentially follows the classic Webster approach of attempting to minimise the maximum volume/capacity ratio by turn by adjusting green splits. There are therefore only minor differences between options 1 and 2 (which mostly occur in complex situations with lane sharing, overlapping stages, etc.)

Delay minimisation, as the name implies, attempts to minimise total vehicle-delay at the intersection. Since it uses analytical approximations to calculate **SATURN** delays it will not necessarily lead to a true optimum.

P0 is based on the elegant principle put forward by Mike Smith (University of York) of equating the product of saturation flow times delay on competing arms. Again, given the complexities at signals as represented within **SATURN**, our version is not necessarily a “pure” application of PO. It differs from the first three options in that it does not explicitly set out to produce a true local optimum but to set the signals such that, in conjunction with the consequent re-assignment of traffic, the total network travel times will be reduced.

Finally equi-saturation Mark 2, as introduced in 10.1, has the same general objective as 1 but uses a different algorithm to achieve it. Experience to date is limited; certainly in certain situations it performs much better but whether there are other situations where it performs worse is not yet certain. Nonetheless it is recommended over 1.

Each algorithm follows an iterative strategy whereby green time is “swapped” between the “best” and “worst” stages, with the amount of green time swapped being the (local) optimum. After each swap the best/worst criteria are recalculated

and the next pair identified. While fairly reliable, such an approach is not guaranteed to produce a global optimum; under certain circumstances the algorithm may “stick” and the apparently best pair for swapping may not in fact lead to any improvement. For this reason the maximum number of iterations is user-set in order to prevent infinite loops.

Further “outer” iterations may also be required since, once new stage times have been generated, a re-simulation of that node may change some of the criteria on which the optimisation was based; e.g. lane sharing may change. A further (user-set) option specifies the maximum number of outer iterations allowed (default 1 since in most cases a re-simulation and re-optimisation has no effect).

Finally it should be noted that the optimisation procedures all assume that stage times are defined by integer seconds as opposed to being continuous variables. Again this implies that the solutions are not “true” global optima, but equally means that they may be insensitive to small changes in junction parameters (e.g. flows) and therefore they converge more rapidly.

15.31.4 Using SIGOPT (and/or SATOFF) within SATALL

As an alternative to optimising stage green times outside the assignment/simulation loop as discussed in 15.31.2 it is also possible to do so within the loop using **SATALL**. Thus setting the parameter SIGOPT = .TRUE in (preferably) the original network .dat file or in the **SATALL** control file results in a “two pass” simulation process within the standard loop. Thus on the first “pass” the simulation uses the current stage green times and the latest assigned flows; it then updates all stage times at signalled junctions independently and re-runs the simulation in a second pass. Statistics describing the degree of changes to the green times appear both on the screen and (in greater detail) in the .LPT file.

Note that the option SIGOPT automatically optimises **all** nodes; there is, as yet, no option to optimise over a subset of signals although this can be done using the batch procedure **SIGOPT** described in 15.31.6 below.

Equally the offsets can be automatically optimised within each simulation by setting **SATOFF** = T – or offset optimisation could be done on its own by setting SIGOPT = F and SATOFF = T (but, see below, this is not recommended).

The choice of optimisation algorithm 1 to 5 above is set by the parameter MYTVV as set in &PARAM either in the **SATNET** .dat file or in the control file to **SATALL**.

It needs to be emphasised that this procedure is largely experimental and we have very little experience so far to compare the results from the above procedure from that using the explicit **SATED-SATALL** loop. Since the updates are more frequent in **SATALL** - one per loop - one might expect to find “better” signal settings and lower travel times using this method - but life is not necessarily straight forward with network models! However using SIGOPT = T within **SATALL** is likely to lead to over-estimates of network performance as noted in 15.31.1 and it should therefore be used with some caution and only if the resulting signal times and flows are carefully analysed for “realism”.

The same note of caution should also be applied to the use of SATOFF = T within **SATALL**. In particular, since the optimum offsets are most sensitive to link cruise times which are (generally) fixed as opposed to turning delays which may vary

considerably with iterations of **SATALL**, the optimum offsets per node may only change once within **SATALL** and the same result could be obtained by using the program **SATOFF** on its own. Using SATOFF = T on its own within **SATALL** with SIGOPT = F has even less to recommend it.

N.B. Optimising stage times (in particular) and/or offsets may lead to significant improvements in the overall convergence of the assignment-simulation loops. See Section 9.1.5.

15.31.4.1 NIPS

On the other hand, using the parameter NIPS to limit the number of times the signal and/or offset optimisation within **SATALL** is called is **STRONGLY** recommended. See 9.12.2. A value of 2 or 3 is recommended.

15.31.5 Preserving and Transferring New Stage Times

Having created new stage times (and/or offsets) by any of the above methods it is natural to wish to include that information within a network .dat file. The best way to do that is to use the Network Editing facilities within **P1X** and, in particular the update option described in 11.9.13.2 and/or rgs files as described in 11.9.14. Alternatively, the batch procedure described in 15.31.6 has options to output an updated .dat file automatically.

Once an updated .dat file has been created you may wish to re-run **SATURN** “from scratch” with the signals and/or offsets fixed at their optimal values. Before you do so be careful that parameters such as SIGOPT or SATOFF have all been “turned off” within the new .dat file. Also note that the “from scratch” results may not be identical to those previously obtained since the new run may follow a slightly different “convergence path” and wind up with slightly different results; only with perfect convergence (unobtainable) would this problem would not arise.

15.31.6 The Batch Procedure SIGOPT

A new “batch procedure” SIGOPT.BAT has been introduced in Release 10.8.16 to optimise stage times and/or offsets in a .ufs file and to create a new output file(s). It effectively supersedes both **P1X** in terms of stage times and **SATOFF** in terms of offsets. Output files may be either (a) .UFS, (b) .DAT and/or (c) .RGS

SIGOPT makes use of existing routines within **P1X** but runs in a non-graphical non-interactive mode such that it resembles any other batch-mode program. It is called via:

```
SIGOPT net KR control KP fildat
```

where control.dat (optional) is an ascii file which sets various options, filenames etc. via Namelist and may also contain a list of selected nodes for optimisation. The full list of parameters with their defaults is listed below.

Fildat (optional) specifies the name of an output (.dat) file containing the revised network .dat file. Fildat may equally be specified as a Namelist parameter within control.dat.

Note that if the input network file net.dat references \$INCLUDE files within the 11111 records then the output file fildat copies these files directly into the new

11111 records; the \$INCLUDE files may be recreated – and potentially renamed – using text editing cut'n'paste. See 11.9.2.1.

The filenames for a new .ufs file and – optionally - a new .dat file must be explicitly set within control.dat but that a file net.rgs is always output with its filename fixed by the input network net.ufs.

PARAMETER	TYPE	DEFAULT	FUNCTION NAME
SIGOPT	LOGICAL	.TRUE.	If .TRUE. optimise green times
SATOFF	LOGICAL	.FALSE.	If .TRUE. an offset optimisation is carried out prior to green time optimisation
SELECT	LOGICAL	.FALSE.	If .TRUE. read a set of selected nodes to be optimised from this file immediately after &END; see also 11.9.13.2
RESIM	LOGICAL	.FALSE.	If .TRUE. a complete simulation is carried out prior to the output of .ufs file
MYTVV	INTEGER	1	Stage time Optimisation algorithm – See 15.31.3
NOPMAX	INTEGER	1	Maximum number of internal iterations used by the signal setting routines; see 15.31.3
MANOFF	INTEGER	0	The signalised simulation node number used as the reference point for all optimum offset set by SATOFF . See 12.2.3.
FILDAT	CHARACTER	Blank	Defines an output .dat file
FILUFS	CHARACTER	Blank	Defines an output .UFS file

RECORD(S) 2 – Selected (Signalised) Nodes

One node number per record in free format terminated by 99999 to select a subset of nodes to be optimised for both stage times and/or offsets.

15.31.7 Using SIGOPT for Base Year Networks

In principle there should be no need to run signal optimisation for base-year networks where the stage times should be directly obtainable from observation and, one would hope, SIGOPT would give very similar times with very little improvement in travel time. However, it may be a very useful “calibration/validation” exercise to check that this is indeed the case since large deviations between observed and “optimised” stage times at an individual node might well be a very good indication that there is something wrong, e.g., that the node has been miscoded or that the assigned flows are well out, etc. etc.

15.31.8 Convergence Statistics for Signal Optimisation

Whether or not the green splits at signals have actually been optimised it is possible to calculate the maximum possible improvement in the V/C ratio per turns at signalised nodes. These calculations are carried out at the end of every run of **SATALL** and the improvements per node are saved on the output .UFS files as

well as the maximum time change per stage in order to achieve optimisation. The .LPT file contains a global summary of the potential improvements.

In addition it is also possible within the **P1X** Convergence Menu (11.15) to list the 10 nodes with the **maximum** potential V/C improvements and to highlight them. Note that those nodes with poorly set stage times are also likely to be the same nodes that cause convergence problems for the assignment-simulation loops and therefore optimising those signals may significantly improve overall convergence. See note 9) in 9.1.5.

15.32 Determining Fuel Consumption

Fuel consumption is an area of major concern to traffic engineers for obvious reasons. It is also an area which, as with emissions (15.33), is probably best handled “post processing”; i.e., users will have their own particular favourite model or formulae for fuel consumption which will require both data from **SATURN** such as flows and/or speeds and exogenous data such as graphs of fuel consumption vrs. speed. In such cases the best option is to dump the required **SATURN** data into, say, a link-based text file using **SATDB** and to pass that data into their own procedures. (Or it may also be feasible for users to set up their own equations / calculations using the data manipulation facilities within **SATDB** (11.10.8.1))

However there are also internal fuel consumption models within **SATURN**. Thus, in order to estimate the total amount of petrol consumed within the simulated network, **SATURN** uses the following equation:

$$f = FLPK * d + FLPH * t + FLPPS * s_1 + FLPSS * s_2$$

where:

f	=	fuel consumption in litres
d	=	total travel distance in vehicle-kilometres
t	=	total delayed (idling) vehicle-hours
s ₁	=	total number of ‘primary’ or ‘full’ stops at an intersection; e.g. where a vehicle arrives at the end of a queue
s ₂	=	total number of ‘secondary’ stops; e.g. stop-starts while a vehicle moves up in a queue

and the “weighting” parameters FLPK etc. have been assigned default values as follows:

FLPK	=	0.07
FLPH	=	1.2
FLPPS	=	0.016
FLPSS	=	0.005

These parameters were all chosen as appropriate figures for an ‘average’ British car in 1981. More details may be found in Ferreira. (“The role of comprehensive

traffic management in energy conservation”, PTRC Summer Annual Meeting, July 1981).

Clearly these figures are now **out-of-date** and take no account, for example, of the breakdown of the flow into various vehicle types. The parameters may be user-set as standard namelist parameters within **SATNET**.

15.33 Determining Emission Statistics

Emissions of harmful pollutants from road traffic are an increasingly important issue for engineers, planners and politicians alike - not to mention the general public who have to live in it! It is also an extremely complicated process, both in terms of actual emissions (e.g., variations between vehicles) and their ultimate dispersion and chemical reactions.

Predicting emissions is, as with fuel consumption (15.32), probably best handled “post processing”; i.e., users will have their own particular favourite model or formulae for calculating emissions which will require both data from **SATURN** such as flows and/or speeds and exogenous data such as meteorological data. In such cases the best option is to dump the required **SATURN** data into, say, a link-based text file using **SATDB** and to pass that data into their own procedures. (Or it may also be feasible for users to set up their own equations / calculations using the data manipulation facilities within **SATDB** (11.10.8.1))

Alternatively, in order to encourage the consideration of pollutant emissions, **SATURN** contains some fairly simple-minded internal procedures for the estimation and display of five standard pollutants: carbon monoxide, carbon dioxide, hydrocarbons, nitrogen oxides and lead. The estimation procedures are similar to those used to estimate fuel consumption, i.e. a linear model with explanatory variables of time, distance, primary and secondary stops. Hence the basic equation for the emission of pollutant i from a link is:

$$E^i = (a_1^i d + a_2^i t_c + a_3^i t_q + a_4^i s_1 + a_5^i s_2) V$$

where:

d is link distance

t_c is average cruise travel time on the link

t_q is the time spent “idling” in queues at junctions

s_1 is number of primary stops per vehicle

s_2 is number of secondary stops per vehicle

V is the vehicle flow

a_1, a_2, \dots are (user-set) coefficients.

It needs to be emphasised that this is an extremely crude model. Moreover the default coefficients given below are even worse! If it gets to within an order magnitude of the “true” answer it will be doing well. The main reason for including it at this stage is to provide, for examples, options in **P1X** to display emissions per link or options in **SATLOOK** to print totals. Improved models with more reliably

calibrated coefficients will undoubtedly follow and users are strongly encouraged to put forward their own models.

Default parameter values for four of the pollutants (excluding CO₂) have been extracted (with some fairly broad brush assumptions; e.g. that a primary stop involves a deceleration from 50 kph to rest and the reverse acceleration) from the data used in the 1988 Leeds PhD dissertation of Athanasios Matzoros (see also A Model of Air Pollution from Road Traffic I and II, A. Matzoros and D. Van Vliet, Transportation Research, pp.315-335, Vol 26A, 1992). Default values are listed below.

Grams / PCU /	Kilometres	Cruise Hour	Idling Hour	Primary Stop	Secondary Stop
Carbon dioxide	70.0	0.00	1200.00	16.000	5.000
Carbon monoxide	0.0	304.80	180.00	2.22	0.444
Nitrogen oxides	0.0	102.60	1.80	0.42	0.084
Hydrocarbons	0.0	57.00	30.00	0.39	0.078
Lead	0.0	0.36	0.09	0.0024	0.0005

Carbon dioxide parameters are extracted from the fuel consumption parameters on the assumption that “most fuel” is converted into carbon dioxide.

Parameter values may be reset by users using the namelist inputs to **SATNET** within a network .dat file. See Section 6.3.3; all parameters are “reals”. Their names are constructed using the following conventions:

- ◆ All names commence with the characters CO, CO2, XNO, HC or PB.
- ◆ The next character is a P (for “per”).
- ◆ The final characters are K (for kilometre), CH (for cruise hour), IH (for idling hour), PS (for primary stop) or SS (for secondary stop).

Thus HCPCH is the variable denoting hydrocarbons emitted (units of grams) per hour cruise time per pcu.

15.34 Estimating Primary and Secondary Stops

While the simulation element within **SATURN** does not explicitly model the exact progression of every vehicle as they move down a link it is possible to infer certain properties of their progression. Thus **SATURN** estimates the number of times on each simulation link that vehicles execute primary and secondary stops.

The distinction between the two forms of stop is basically the following. Imagine a minor arm at a priority junction with a “stop sign” at the end; every vehicle approaching that junction should (must!) come to a complete stand still either at the stop line (if there is no queue) or behind the last vehicle in the queue; that is a primary stop. If there is a queue and vehicles depart from the head of the queue one at a time then vehicles further back will move up by accelerating and then decelerating to a stationary position; these are secondary stops.

Clearly this two-way split does not exactly represent all possible vehicle movements in a queue but it may well be sufficiently good for estimating secondary parameters such as fuel consumption or emissions and for providing a very broad description of the state of a junction.

The rules for estimating primary and secondary stops are, like their definitions, somewhat arbitrary. Thus for minor arms at priority junctions all arriving traffic must make a primary stop if its turn is over capacity or if the queue per lane is greater than 2. If the queue per lane is (in the limit) zero the probability of a primary stop is equal to the calculated probability of there being no gap. For queues per lane between 0 and 2 pcu's a linear relationship is assumed.

Secondary stops are calculated by assuming that all primary stops make a further number of secondary stops equal to the queue length per lane divided by the number of vehicles that can depart from the stop line "in a platoon" once a gap occurs (assumed equal to one over the probability of a gap).

Roundabouts are treated in the same way as minor priority arms.

For major priority arms secondary stops are ignored and a primary stop only occurs if the arm is over capacity or if, at the moment of arrival, the expected queue length per lane is greater than 1.

At signals all arrivals primary stop during a red phase or, during the green phase, if the expected queue is non-zero. Secondary stops occur whenever the lights go red to all vehicles in the queue at that instant.

15.35 Altered Data Formats in .DAT Input Files

Generally data input files to **SATURN** programs are "formatted", meaning that repeated numerical data needs to be in fixed fields or sets of columns with a specific number of decimal places, etc. See Section 2.8.1. These are specified by "format statements" set within the program but which, as explained here, may also be altered by the user.

An example taken from the buffer-network data input to **SATNET**, see 6.6, is given below. Thus the first line specifies a buffer link from node 23 to node 22 in the standard format. The line **FORMAT** (3F7.1... requests a new format, the basic change being that the 3F7.1 implies that the three input fields following the A-node and B-node occupy 7 columns with 1 decimal place as demonstrated by the next data line for link 20 to 21. A line with characters **FORMAT** in columns 1 to 6 but blank thereafter causes the format to revert to its default.

23	22	28	56	2500	2	1000	2.19	125
FORMAT (3F7.1,2X,I1,A1,1X,F5.0,F5.1,2X,I3)								
20	21	28.1	56.1	2500.2	2	1000	2.19	125
FORMAT								
20	21	28	56	2500	2	1000	2.19	125

In principle the change of format facility could be applied almost anywhere: in practice it has only been programmed in a very few places, including the buffer network inputs illustrated above. In this case it has been done to make the data generated under the SATBUF conversion procedure (see 15.8.2) accessible to **SATNET**. Further extensions are planned.

The advantages of being able to change formats are mostly associated with the ability to import data from other suites of programs with formats which do not coincide with those of **SATURN**. Another example of the same basic principle is the parameter XYFORM used by **SATNET**; see 6.3.4.

It should also be stressed that very often data may be read in a variety of forms, provided that it appears within the column boundaries set, and that therefore the formats specified within the Manual may not need to be absolutely strictly adhered to. For example, the format specified in order to read in the “power” for a buffer link speed flow is specified in Section 6.6 as FORMAT F5.1, implying that a single digit appears after the decimal point and that the decimal place must appear in column 39. In practice, since FORTRAN compilers are “forgiving” in terms of input formats, the decimal point may appear in **any** column with **any** number of digits following provided that the whole input appears in columns 36-40. Thus inputs of ‘3’, ‘3.0’, ‘3.123’ will all be correctly read.

In addition, certain inputs which are specified as Integers, e.g., link times and/or speeds with buffer link records, may generally have decimal places included, again with the proviso that the full input appears within the strict column limits.

15.36 Turning Flows at Buffer Nodes

Although **SATURN** does not explicitly differentiate between different exit turning movements from a buffer link in calculating minimum cost routes (unlike the simulation network) when the O-D trips are assigned to paths through the buffer network they do implicitly go through turns and the resulting turn flows may optionally be saved.

To calculate and store buffer turn flows you must have both parameters SAVEIT and REFFUB (which, if you think about, has a rational explanation!) as .TRUE on entry to **SATALL** (the facility is not available in **SATEASY**). The turning flows are calculated by carrying out a final full assignment using the iteration costs stored on the ufc files (see 15.23) and the resulting flows stored in DA array 4953 on the output .ufs file. These may subsequently be accessed using option 2 (look at individual buffer nodes) within **SATLOOK** (11.11.2).

Note the following points:

- 1) Bus routes through the buffer network clearly also make turns; if there are any such bus routes their (pcu) turn flows are calculated within **SATNET** and stored in DA array 943. These are then added to the assigned flows in 4953 which therefore contains total pcu flows.
- 2) The same treatment is not applied to pre-loaded flows (but see note 4 below).
- 3) In multiple-user-class assignment all user classes are combined together in array 4953.
- 4) Since, as explained in 15.23, the routes re-calculated via SAVEIT may be only an approximation to the true routes used in the assignment (due to the effects of DIDDLE or KOMBI for example) the turning flows are “furnished” so that the total exit and entry flows on each arm correspond exactly to

those assigned. This procedure would also account for any “missing” turn flows due to pre-loaded flows.

15.37 Repeated Assignments: Modelling Cold Starts, etc.

The **SATRAP** option within the Assignment/Tree Building sub-menu within **SATDB** repeats a full assignment to the same routes and in the same proportions as in the final assignment (or, strictly speaking, the final assignment as re-created under SAVEIT - see 15.23). Thus re-assigning the original trip matrix should give the same (demand) link flows as already stored on the .ufs file. So why bother?

Firstly, **SATRAP** allows the user to investigate the impact of assigning a different trip matrix to the same routes. One of the sub-options within **SATRAP** allows the user to modify the matrix using random numbers in order to model the impacts of day-to-day variability.

A further option allows the user to assign trips over only a section of the O-D routes defined in terms of distance. For example you may ask for only the first (up to) 500 metres from the origin to be loaded, the obvious application of which is to model vehicle flows when the engine is still cold. Alternatively the flows may only be loaded beyond, say, 500 metres, to represent warm vehicle flows. Adding the two together gives total flows. In the case of the critical distance falling in the middle of a link along an O-D path (as in fact must virtually always occur) the loaded flow is taken pro-rata depending on the length of that link.

15.38 Non-discontinuous Speed-Flow Curves: the Kinky Option

Generally, as described in Section 5.4 and elsewhere, **SATURN** speed-flow or cost-flow curves are assumed to follow a power law for flows up to capacity and to be linear thereafter; equations (5.1a) and (5.1b) respectively. Thus there is a discontinuity in the slope introduced at $V=C$ (although the times or costs themselves are continuous). Generally the discontinuity does not create problems within the assignment algorithms and the shift to a linear form is quite realistic particularly bearing in mind that a power-law curve with, say, power 5 goes very rapidly towards infinity for $V \gg C$, a not very realistic forecast which can have serious consequences for scheme benefits.

However there may be circumstances when the user does wish to extend the simple power law relationship over flows from zero to infinity, for example in modelling networks with so-called “BPR curves” or when doing system-optimal assignment where the discontinuity in slope may be an algorithmic problem (see 7.11.9). This is simply done by setting a parameter KINKY to .FALSE in the network .dat file (or in control files elsewhere) in which case equation (5.1a) holds over the full range of flows for “actual” times and (7.19a) holds for the full range of marginal costs.

The default is .TRUE. and, it needs to be stressed again, the alternative should be used with great caution. In particular it is not recommended to use with KINKY = F with simulation networks. In addition, if KINKY = F, then care should be exercised that parameters that control the power of cost-flow curves such BCRP or PMAX should be less than or equal to 5.0.

In general KINKY = F should **only** be used in research-based applications and, even then, for very specific purposes. For example, it may be useful in studies of system optimal assignment and system optimal tolls; see 7.11.9.

Note that KINKY applies to all cost-flow curves, i.e. both buffer as input and simulation as calculated.

15.39 Bus-only Lanes

Bus-only lanes in **SATURN** represent extra lanes along simulation links which are for the exclusive use of public transport vehicles as coded under the 66666 network data records (6.9). The format specification for identifying bus lanes and whether they are “nearside” or “offside” are given in Section 6.4.9.2.

15.39.1 Flows in bus lanes

Bus flows on a link are assigned to a bus lane – and are therefore removed from “normal” traffic – but only if certain criteria based on the next link in the route are satisfied. Two sets of acceptance criteria are applied: the first is fairly obvious but may be overly strict so that a second rule has been added.

Thus, rule 1, for a nearside bus lane, if a bus route makes a turn at the downstream end of the link which is allowed to use lane 1 (given the input turn/lane specifications on that link) then it is allocated to the bus lane; otherwise it is assumed **not** to use the bus lane. Thus, for example, a bus route which turns right (drive on the left) at the end of a link would not be able to use a nearside bus lane if the normal right turns were only allowed from lane 2. Similarly an offside bus lane may only be used by a route whose exit turn uses the highest (most offside) lane; e.g., left-turning buses would be excluded from an offside bus lane.

The second rule, termed the “1+1 rule”, relaxes the above criteria by increasing both the critical lane and exit turn by 1. Thus, if a bus route takes the **second** exit from the nearside and if that turn can use the **second** lane then that route may use the nearside bus lane on the link. For example, a bus which is going straight ahead (second turn) at a 4-arm junction may use a nearside bus lane if ahead traffic can use lane 2.

Similar rules apply to offside bus lanes but in reverse.

At the moment the model does not fully consider the “continuity” of nearside and offside bus lanes in allocating buses to bus lanes. Thus buses on a nearside bus lane on link AB may transfer seamlessly to an offside bus lane on link BC and back again to a nearside lane on CD. Furthermore it is not, for example, possible to restrict bus lanes to certain “companies” or to require that only trams may use offside lanes.

N.B. The above rules were only added in release 11.1. Prior to that all bus flows were assigned to a bus lane if one were available.

15.39.2 Delays in Bus Lanes

Note that a bus lane in **SATURN** is assumed to go from the upstream “entry line” to the downstream “stop line”; set-back bus lanes are therefore excluded. In addition buses in bus lanes form separate queues and therefore have different

delays from “normal” traffic making the same turning movement. In effect we assume that the exclusive lane continues through the junction to the next link’s entry line - where it may, of course, meet up with a further bus lane.

More specifically the delay to a turn from a bus lane is equated to the minimum delay associated with normal turning traffic; i.e. t_0 in equation (8.5a). Moreover this delay is fixed, independent of the volume of traffic in the bus lane. Travel time/speed along the link itself equals the cruise time for normal traffic and the same link distance is assumed.

Clearly this model is only an approximation which will hopefully be improved with later versions of **SATURN**. It does however include the two most salient features of such bus lanes; i.e. that buses should incur lower queues and delays than other traffic and, perhaps more importantly, that buses in an exclusive lane do not reduce the capacity of other traffic on the same link.

15.39.3 Exits/Entries from Bus Lanes

At the start of a bus lane the bus traffic effectively leaves at the upstream end so that: (a) its flow is an integral part of the previous turn (unless of course this is the start of the route); and (b) its flow is not included as part of the normal flow on the link. At the termination of a bus lane the buses rejoin normal traffic upstream on the following link so that (a) it is not part of the final turn flow but (b) is part of the next link flow.

In some respects bus lanes may be thought of as “tunnels” in that, as far as the rest of the traffic on the network are concerned, buses “disappear” at the upstream start of a sequence of bus-lane links and only “reappear” at the upstream end of the first non bus-lane link.

Information on flows to, on and from bus lanes may be obtained via **SATDB** (11.10.6) with up to 15 levels of flow definition available. In addition a table in the .lpn file output by **SATNET** lists similar information on all links with bus lanes.

15.40 Motorway Weaving Segments

15.40.1 Introduction

Weaving segments on motorways correspond to the situation depicted in the diagram below (Figure 15.4) whereby an entrance ramp onto a motorway is followed by an exit ramp downstream such that traffic entering the motorway and staying on it (Flow 2) will need to “weave” with traffic which is already on the motorway but wishes to take the next exit (Flow 3). This will lead to a reduction in the capacity of the middle segment of the motorway if the fraction of traffic which weaves is high and/or the distance between entry and exit is relatively short.

Figure 15.4 - Fig 2/7 from DMRB Vol. 6, Sect 2, part 1

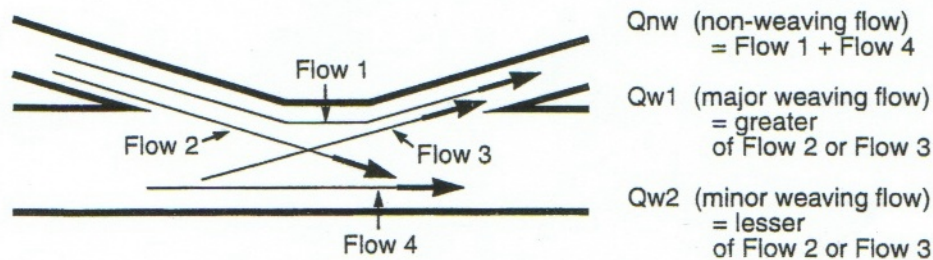


Figure 2/7 : Terms used in Weaving

In these situations **SATURN** uses formulae derived from DMRB (Design Manual for Roads and Bridges) recommendations to reduce the saturation flow (and hence the capacity) of the link (or links) comprising the intermediate segment.

N.B. The treatment below only applies to links coded as part of the simulation network, not buffer. In addition it need not apply only to “motorways” (**SATURN** normally does not know whether a link is motorway or not), although in practice the required geometry of entries and exits is most likely to occur with motorways.

In addition it may **not** be used if the assignment is based on either path-based assignment, OBA or multi-core. In principle it could but it hasn’t been coded; requests to DVV.

15.40.2 Basic Background Theory

Paragraph 2.26 in DMRB Volume 6 Section 2 Part 1 TD 22/92 gives a formula for the number of traffic lanes required for weaving:

Equation 15.3

$$N_r = \frac{1}{D} \left\{ Q_{nw} + Q_{w1} + Q_{w2} \left(2 \frac{L_{\min}}{L_{act}} + 1 \right) \right\}$$

Where:

N_r	=	Number of traffic lanes required
Q_{nw}	=	Total non-weaving flow in vph
Q_{w1}	=	Major weaving flow in vph
Q_{w2}	=	Minor weaving flow in vph
D	=	Maximum mainline flow in vph per lane, referred to as S below.
L_{\min}	=	Desirable minimum weaving length
L_{act}	=	Actual weaving length available (in metres) (referred to simply as L from now on)

(where L_{act} is assumed to be greater than L_{min} and, if not, take $L_{min} = L_{act}$ such that the factor within the bracket multiplying $Q_{w2} = 3$.)

In **SATURN** we need to “invert” this equation since the **actual** number of lanes provided N_a is already specified by the **SATURN** input along with its “natural” saturation flow (which determines capacity) and what we need to know is how much the saturation flow/capacity is reduced by the effect of weaving.

Thus we begin by factoring all the flows Q in Equation 15.2 by a uniform factor F such that the required number of lanes given by Equation 15.2 equals the number of actual lanes N_a (i.e., $F.Q...$ are the flows at capacity) :

Equation 15.4

$$F(Q_{nw} + Q_{w1} + X_f Q_{w2}) = N_a S$$

Where:

- F = required factor
- S = the saturation flow per lane as input by the user and ignoring any effect of weaving (replacing D in Equation 15.2)
- X_f = the extra weight associated with the minor weaving flow ($= 2L_{min}/L_{act} + 1$)

Furthermore at capacity the total unweighted flow should also equal the actual number of lanes times the effective saturation flow S_e :

Equation 15.5

$$F(Q_{nw} + Q_{w1} + Q_{w2}) = N_a S_e$$

$$\text{Let } Q_w = Q_{nw} + Q_{w1} + X_f Q_{w2}$$

Hence from Equation 15.3

Equation 15.6

$$F = N_a \frac{S}{Q_w}$$

Subtracting Equation 15.4 from Equation 15.3 gives:

Equation 15.7

$$Q_{w2} F (X_f - 1) = N_a (S - S_e)$$

whence:

Equation 15.8

$$S_e = S - \frac{Q_{w2} F (X_f - 1)}{N_a}$$

and substituting F from Equation 15.5 into Equation 15.7 gives

Equation 15.9

$$S_e = S \left\{ 1 - \frac{Q_{w2} (X_f - 1)}{Q_w} \right\}$$

Hence the saturation flow is reduced by a factor W:

Equation 15.10

$$W = \frac{S_e}{S} = 1.0 - \frac{Q_{w2} (X_f - 1)}{Q_w}$$

which may be more simply and intuitively written as:

$$W = \frac{Q}{Q_w}$$

where $Q = Q_{nw} + Q_{w1} + Q_{w2}$.

Note that in the “worst possible case” when X_f takes its maximum value of 3.0 and $Q_{nw} = 0$, $Q_{w1} = Q_{w2}$ then the reduction factor $W = 0.5$ which might be thought a bit extreme. Various alternative formulations have been proposed as discussed next.

15.40.3 Extensions and Alternatives to the Basic Theory

A further “feature” of the above model not mentioned above is that the reduction factor is assumed not to apply for weaving lengths in excess of some value L_{max} (typically 3 km.) This will introduce a discontinuity into the multiplier of Q_{w2} , X_f , in that it jumps from a value of $(2L_{max}/L_{min} - 1) > 1$ at $L = L_{max}$ to 1.0 at $L \geq L_{max}$. Recall that the maximum value of X_f is 3.0 at $L \leq L_{min}$.

Two alternatives have been suggested (in addition to retaining the discontinuity):

- (i) Reducing X_f by a constant amount throughout such that it goes smoothly to $X_f(L_{max}) = 1.0$ and is 1.0 beyond.
- (ii) Assume that $X_f(L)$ is a linear function between $L = L_{min}$ and L_{max} going from 3.0 down to 1.0.

More specifically under assumption i) we introduce a correction factor equal to the “normal” value of X_f at $L = L_{max}$ less its desired value of 1.0:

Equation 15.11

$$C = 2.0 \frac{L_{min}}{L_{max}}$$

Hence the full formula for $X_f(L)$ is:

Equation 15.12

$$X_f(L) = \begin{cases} 3.0 - 2.0 \frac{L_{\min}}{L_{\max}} & L < L_{\min} \\ 1.0 + 2.0 \left\{ \frac{L_{\min}}{L} - \frac{L_{\min}}{L_{\max}} \right\} & L_{\min} < L < L_{\max} \\ 1.0 & L > L_{\max} \end{cases}$$

Method ii) may be written:

Equation 15.13

$$X_f(L) = \begin{cases} 3.0 & L < L_{\min} \\ 1.0 + 2.0 \frac{(L - L_{\max})}{(L_{\min} - L_{\max})} & L_{\min} < L < L_{\max} \\ 1.0 & L > L_{\max} \end{cases}$$

Having established X_f (by whatever method) an alternative method for establishing the capacity reduction factor W is to use the formula (as proposed by Philip Barrett of HKBR):

Equation 15.14

$$W = 1.0 / \left(1.0 + \frac{Q_{w2} (X_f - 1)}{Q_w} \right)$$

Which, to a first approximation, is the same as Equation 15.9 for small corrections but is less severe as the effect increases, e.g., as Q_{w2} increases. Thus, whereas Equation 15.9 gives a maximum reduction (minimum W) of 0.5 Equation 15.13 gives 2/3 under the same conditions.

A final extra “rule” is to set a minimum value on the capacity reducing effect of weaving traffic by requiring that, say:

Equation 15.15

$$W \geq W_{\min} = 0.75$$

Which, or which combination, of the above approaches is preferable is very much in the eye of the user. There is very little empirical evidence to say that this equation is “right” and that is “wrong” - what **SATURN** is doing is providing a set of approaches which have been proposed (by experienced modellers!) and let the user decide. And if the user has an alternative approach it should not be too difficult to include alternative formulae within the programs.

15.40.4 Application within SATURN

To apply capacity reductions due to weaving within **SATURN** users must (a) set various parameter values as used in the above equations (strictly speaking

optional as default values are provided), and (b) identify those links where weaving occurs.

15.40.4.1 Network Coding: the W link marker

We note first that a weaving section may consist of either a single link as illustrated in Figure 15.4 connecting the node with the “on ramp” to the node with the “off ramp” or (less frequently) a series of (essentially one way) links connecting the “on” and “off” nodes as illustrated in Figure 15.5.

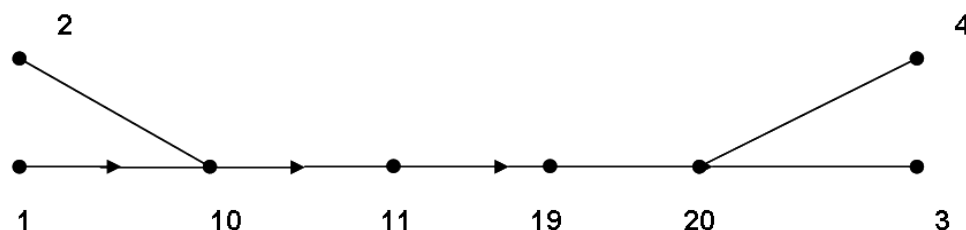
Link identification is accomplished by coding a W within the 4 columns of the simulation link record normally used to specify the number of lanes, i.e., columns 12 to 15 (see Sections 6.4.1 and 6.4.9.4). Thus 3W or W3 would both denote a 3-lane link where weaving takes place.

Note that the link where the W is added is the **middle** link in the weaving section (provided that there is only one intermediate link) and that nothing needs to be added on either the links which enter the weaving segment or that exit (e.g., entry and exit ramps). On the other hand if there are multiple intermediate links as in Fig. 15.3 then a W must be added for **all** those links, otherwise a non-fatal error results and the weaving movement is ignored.

15.40.4.2 Network Geometry

In either case a number of geometrical conditions need to be satisfied.

Figure 15.5 - A weaving section with intermediate links



Thus the “on” or “upstream” node will (normally) be a 3-arm priority junction with 2 one-way in-bound links feeding a single one-way out-bound link; i.e., in-bound motorway and in-bound ramp feeding the out-bound motorway. In more precise geometric terms there must be only one permitted turn from the entry ramp link - the first geometrically possible turn - and only one permitted turn from the “motorway” - the second geometrically possible turn - so that both have the same exit arm. Thus one cannot have an exit from the motorway onto the ramp arm - entry and exit ramps must therefore be defined at distinct nodes.

Equally the “off” or “downstream” node will also be a 3-arm priority junction with one one-way in-bound arm feeding two outbound one-way arms. The entry (motorway) arm must have both its two possible turns defined - the first to the off ramp, the second continuing along the motorway and the off ramp link must be one-way out-bound.

In practice both the on and off nodes will be 3-arm priority nodes as described above. However, strictly speaking, the nodes may have more than 3 arms as long as the extra arms and turns do not interfere with the required geometry. For

example the nodes could include both the entry and exit ramps on either side of a motorway with the two motorway arms being two-way but the turns on one side of the motorway could not cross those on the other side. In general such coding is **not** recommended, particularly if weaving is being modelled: each direction of the motorway should contain distinct nodes and links.

If there are one or more intermediate nodes such as nodes 11 and 19 in Figure 15.5, then each should be essentially a two-arm priority with a single one-way exit feeding a single one-way exit. (Such nodes might be added in order to provide “shape” to the network although, it should be noted, the “shape” may also be obtained via a GIS file; see 5.7).

15.40.4.3 Assignment Calculations

The methods by which the required entry exit flows are monitored with an assignment differ depending on whether not Network Aggregation (15.56) is invoked or not (SPIDER = T or F).

Thus, if Network Aggregation is not invoked (SPIDER = F), the 4 demand entry-exit flows which make up the weaving segment (two possible entries and two possible exits) are continually monitored while the assignment is taking place and, at the end of the assignment and prior to the next simulation, that information is used to calculate the reduction factor as described above. (Strictly speaking only a **single** flow is monitored since the other 3 flows may all be obtained knowing the total demand flows on the entry/exit arms.)

On the other hand, if SPIDER = T and if **all** the nodes within the weaving segment (e.g., nodes 10 to 20 inclusive in Fig. 15.5) are aggregated then all the possible weaving movements 1-3, 1-4, 2-3 and 2-4 will either be distinct aggregated links or part of larger aggregated links. In either case the necessary flows may all be taken directly from aggregated link flows and no extra steps are required during the assignment itself. The method is therefore much more efficient and significantly faster in terms of CPU.

For additional discussion on aggregated networks see 15.56.7.4.

15.40.4.4 Simulation Capacities

Within the simulation the reduction factor is applied to the saturation flows for all turns **out of** the “motorway” links coded as W. Thus, in Figure 15.5, it would be applied to turns 10-11-19, 11-19-20, 19-20-3 and 19-20-4. It would not, however, be applied to the turns corresponding to entry into the first weaving link (i.e., turns 2-10-11 and 1-10-11 in Figure 15.5)

The factor is, in effect, applied immediately after the saturation flows are set and at the same time as the blocking back factor is applied; i.e., step (2) in Section 8.2.1. Note that the reduction factor is equally applied to all turns at intermediate nodes (if any) between the entry and exit nodes.

Note that the weaving reduction is applied **in addition** to any other capacity-reducing effects such as give-ways or blocking back.

15.40.4.5 Simulation Delays

Weaving does not, of necessity, add extra delays to traffic although there are three ways in which extra simulation delays may result.

Firstly, if a link becomes over capacity due to weaving then queuing delays will be imposed.

Secondly, if the weaving links are subject to link-capacity restraint functions (see 6.4.12) then the “link” or “pinch-point” capacity used in equation (6.2) is also reduced in accordance with Equation 15.9 above leading to (in effect) a reduction in cruise speed for a given flow.

Finally, if a Q-marker has been used on an intermediate link (see App. Q), then the capacity used to calculate the V/C ratio in equation (Q.1) is taken **after** the weaving factor *W* has been applied to the saturation flow, thus potentially increasing the delay.

If none of the above three conditions occurs then introducing a weave marker will have virtually no impact on travel times and hence on assignment. Link capacity-restraint speed-flow curves are therefore highly recommended in conjunction with weave markers.

15.40.5 SATURN Namelist Parameters

The following parameters may all be defined within the &PARAM namelist parameters within a network .dat file (Section 6.3) to control the various options within weaving calculations:

- ◆ WLMIN - Minimum length for weaving in metres - L_{\min} in Equation 15.2
- ◆ WLMAX - Maximum length for weaving in metres - L_{\max} in Equation 15.2
- ◆ PHILIP - If .TRUE. use Phil’s formula (Equation 15.13)
- ◆ STUART - If .True. use Stuart’s formula (Equation 15.11); else use (Equation 15.12)

Note that the first two parameters are “reals” while the latter two are “logicals”. The default values are, respectively, 300 metres, 2000 metres, False and True. Thus if the weaving section were 300 metres or less the maximum reduction would be applied to saturation flows; if it were more than 2000 metres than no reduction would apply.

15.40.6 Restrictions

The weaving calculations may not yet be applied to all possible situations within **SATURN**. Thus it will not work with stochastic assignment (SUZIE = T). It should, however, still function with, e.g., elastic assignment or multiple user classes (I Think!).

15.40.7 Link Weaving and W Turn Priority Markers

There are certain obvious parallels between the phenomenon of weaving on a link as described above and of “weaving at a node” as described in 6.4.2.5 and based

on the use of W turn priority markers. In both cases 4 sets of individual flows come together and, depending on the level of “crossing over”, reductions in capacity and increased delays may result.

The precise mechanisms by which these effects are modelled within **SATURN** are different however. Thus W priority markers are modelled essentially as a form of give ways at a single junction controlled by parameters such as GAP whereas the link weaves use quite formulae and quite different parameters and apply over more than one coded node.

Link weaving may be seen as weaving “over a distance” whereas W priority markers represent weaving “at a point” - effectively therefore over much shorter distances. The choice of one form over the other should therefore be partly governed by the distance over which weaving is felt to take place.

15.40.8 Display of Link Weaving Data (E.g., P1X)

Each link which has been coded with a W is assigned a numerical “marker” which indicates, inter alia, its position in the sequence. Thus a value of 1 indicates that the link is the first link beyond the entry ramp in a sequence of more than 1 links (e.g., 10-15 in Figure 15-3), 4 indicates it is the final link before the exit ramp (16-20 in 15-3), 5 that it is the **only** link in the sequence (i.e., both entry and exit) while 2 indicates an intermediate link in a sequence of multiple links (e.g., 15-16).

The markers may be displayed as link annotation data via **P1X** (under “Properties”) or otherwise accessed as a data base item within **SATDB**.

In addition the .lpt files output by **SATALL** print a list of all links where weaving factors have been applied at each assignment-simulation loop with the current values of all relevant data such as Q_{nw} , X_f , etc. The factors may also be displayed in the numerical node information menu in **SATLOOK** (post 10.6).

15.41 SATTUBA

15.41.1 Objectives

SATTUBA is a procedure embedded within **SATLOOK** which enables a set of skimmed cost matrix files to be calculated from a .ufs network file and output in a text format which is compatible with the economic appraisal program TUBA.

More specifically TUBA requires as input a set of matrices giving for each O-D pair:

- ◆ passenger or vehicle trips;
- ◆ distance;
- ◆ time; and
- ◆ (monetary) charges.

These matrices may be further disaggregated by, e.g., user class, time period, trip purpose etc.

Distance, time and/or charge matrices all need to be path-weighted averages, i.e., the travel time averaged over the paths used by each O-D pair as opposed to being, say, the time component along a single minimum generalised cost path or even the time along the minimum time path. In **SATURN** terminology TUBA requires **skimmed** matrices as opposed to cost matrices; see Section 15.27.4. Hence **SATTUBA** requires that the network is set up with SAVEIT = T and the skims are based on forests, not trees.

We note that, as explained in 15.23.2 and 15.27.5, the forest path flows generated by SAVEIT are not necessarily **exactly** equal to the path flows generated during the “true” assignment. Thus quantities such as total pcu-hours, pcu-kms etc. calculated using the skimmed and demand trip matrices – which is, effectively, what TUBA seeks to do – are only approximations. See 15.23.2 for a discussion of how these approximations may be improved.

We further note that, quite apart from numerical uncertainties arising from the method of calculation and/or convergence, there are further theoretical problems in that, in principle, Wardrop Equilibrium does not yield unique values of O-D time, distance or toll. On the other hand it does yield unique values of OD generalised **cost**. See below and sections 7.1.6, 7.8.6 and 15.23.8 for further discussion.

In a wider context it also has to be remembered that the “accuracy” of a skimmed matrix is also affected by the overall convergence of the full model run, not just the SAVEIT accuracy. Counter-intuitive results from economic evaluation techniques such as TUBA or COBA may be simply a consequence of poor convergence in either or both the do-nothing and do-something model runs.

Note that the trip matrix necessary as an input to TUBA may be “dumped” from **MX** using the standard option to dump a matrix as comma-separated (CSV) output; see 10.15.

***N.B.** The problem noted above with respect to the uniqueness of the sub-components of generalised cost (i.e., time, distance etc.) is potentially a problem for **all** economic evaluation procedures. There is therefore a very strong case for basing economic evaluation on the generalised cost as used in the traffic assignment model which has the advantage of being uniquely determined (although there are still problems of convergence accuracy) as opposed to relying on sub-components such as O-D time and distance which are not unique.*

15.41.2 Single User Class Networks

The required matrices for a network with a **single** user class may be produced by a specific bat file sattuba.bat which is run by a command such as:

```
sattuba net
```

which takes as input a network file net.ufs and outputs (up to) 3 matrices in text format:

net_d.txt

net_t.txt

net_m.txt

net_p.txt

where the first matrix contains distances, the second contains times, the third contains toll charges (if any exist) and the fourth contains penalties (if any exist).

Units are the defaults as specified by TUBA: distance is in kilometres, time and penalties are in hours and tolls are in pence. The format used is TUBA "Format 1" - see Appendix C of the TUBA User Manual for more details. (Essentially this outputs all O-D cells, one record per origin, in comma-separated format. If required options to input under formats 2 or 3 could be provided.)

15.41.3 Multiple User Class Networks

If the network has multiple user classes (NOMADS > 1) then separate TUBA data files will probably need to be produced for each individual user class. Thus:

```
sattuba net UC 2
```

processes data for user class 2 from the MUC network file net.ufs to produce:

Net.uc2_d.txt, net.uc2_t.txt, etc. etc.

Equally

```
sattuba net UC *
```

processes data for **all** user classes. See 15.41.4.2.

15.41.4 Options within SATTUBA

We describe here three alternative options within **SATTUBA**: the use of a control file, the use of distinct user classes and alternative output matrix file formats.

15.41.4.1 The 'Control File'

The precise format of the output .txt files may be modified by a number of parameters and/or options contained as Namelist parameters in the **SATLOOK** preferences file satlook0.dat (11.17.2). Alternatively, a different preferences or "control file" may be defined on the command line by, e.g.:

```
sattuba net KR control
```

in which case the file control.dat defines the parameters.

The following namelist variables may be used:

- ◆ EFORM (Logical): If .TRUE. the data is output using E-Formats; Default F.
- ◆ NDPS (Integer): Number of decimal places printed (subject to certain minima); Default 4.
- ◆ USETP (Logical): If .TRUE. 44444 time penalties are included within the skimmed times (See 15.24.4); Default T
- ◆ CLICKY (Logical): If .TRUE. skimmed times by user class include any possible extra times due to CLICKS (See 15.24.4); Default T.

- ◆ XCCSK (Logical) – If .TRUE. times and distances on **all** centroid connectors (effectively only buffer centroids since simulation centroids have zero time and distance by definition) are excluded from the skims by setting them to zero. However any tolls on centroid connectors **are** included as set. See 15.41.5 below.

15.41.4.2 Distinct User Classes

SATTUBA may be used to output files for individual user classes using commands of the form:

```
SATTUBA network UC n
```

which will output matrices of the form network.ucn_t.txt, etc. etc.

If “UC *” is used in the command line then the output matrices represent **all** the possible user classes with matrices of the form network.uc1_t.txt, network.uc2_t.txt, etc. etc.

15.41.4.3 Alternative Matrix Formats

By default **SATTUBA** outputs matrices using TUBA format 1 (CSV); alternatively **SATTUBA0** outputs its matrices as **SATURN** .ufm files whereas **SATTUBA3** outputs them in TUBA Format 3. Otherwise the formats, filenames etc. are the same as under **SATTUBA**.

The number of decimal places used in text output formats, e.g., CSV files, may be user-set via a parameter NDPS in the “standard” **SATLOOK** preferences file satlook0.dat or via user-set file; see 15.41.4. The current default is 4.

Note that there is no **SATTUBA2** procedure since TUBA Format 2 does not make much sense in this context. Equally there is no **SATTUBA1** since that is what **SATTUBA** does.

15.41.5 O-D Speeds in TUBA: XCCSK

We note that TUBA uses the O-D time and distance matrices produced by **SATTUBA** to construct its own internal matrices of average O-D speed which in turn it uses to estimate fuel consumption and vehicle operating costs. Problems may arise if either the time or distance matrices contain “artificial” elements or have certain components missing leading to unrealistic speeds.

Thus, if buffer centroid connectors have been created with either a distance and no time or a time and no distance then the summed O-D times and/or distances may lead to very high or very low speeds. The most frequent (inadvertent) cause of this is when SHANDY = T and CROWCC = T (see 15.10.3), in which case a buffer centroid connector which is defined in the network .dat file with both time and distance fields blank will have its distance set equal to the crow-fly distance but no equivalent time.

One solution is to set CROWCC = F (as recommended and, post 10.9, the default), in which case the distances will **not** be added in the original network file.

An alternative is to use the parameter XCCSK (eXclude CC in SKims) within the **SATTUBA** control file (see 15.41.4 above) to effectively set the time and distances

for **all** centroid connectors equal to zero, in which case they make no contribution to O-D skims which are therefore based entirely on “real” network components only. XCCSK is new in release 10.9 but is “retrospective” in the sense that 10.9 SATTUBA may be applied to .ufs files created prior to 10.9.

Note that XCCSK applies **only** to skims of time and distances, not to skims of, e.g., tolls or generalised costs and that it is also used more widely within time and/or distance skims; see, e.g., SKIMTIME and SKIMDIST in 15.27.7. Its default (F) is set within the preferences file SATLOOK0.DAT.

A further example occurs when two zones both have centroid connectors feeding in/out of the same simulation node, in which case the obvious path consists of an entry connector to the stop-line at the node, a single turn at the junction followed immediately by an exit connector. In this case the O-D pair will have positive time from the turn but zero total distance (since both turns and simulation centroid connectors have zero distance by definition). In this case there are fewer simple remedies within **SATURN**.

N.B. **SATTUBA** is still very much “work in progress” and not all the final essential options have been added. We therefore welcome feedback from users.

15.42 SATCOBA

SATCOBA is a procedure embedded within **SATDB** which enables a sub-network of links to be defined which is compatible with that required by the economic assessment program COBA and, in addition, to output a text file which specifies the network and includes flow data and selected link data as required by COBA in the formats required by COBA.

Alternatively a sub-set of links as would be used by COBA (see paragraph 1 in 15.42.1) may be selected within **SATDB**, after which the user is free to output whatever data they wish in whichever format they wish (as opposed to **SATCOBA** which outputs fixed data in fixed formats).

15.42.1 General Functionality

COBA requires that the network be defined in such a way that: (a) there are no centroid connectors, only “real” links, and (b) all links are “bi-directional”; i.e., if a link (A,B) is included it represents both the link from A to B and that from B to A (or, in the case of a one-way link, only the direction that exists). Thus the first job carried out by **SATCOBA** is to define an appropriate sub-network. Note that within this sub-network node names are those used by **SATURN** but each link is given a unique number which equals its normal link number in the **SATURN** assignment network (so there will be gaps in the numbers). An alternative system of user-set link numbers is described in 15.42.3.

Secondly **SATCOBA** then calculates the total flow per COBA link with the flow for a 2-way link being the sum of its two directional flows. Furthermore, since COBA wants flows over, say, 24 hours (or 12, etc. etc.), the flows are factored by a user-set parameter COBAF which is defined under &PARAM in the original network .dat file (default 1.0) and/or within the **SATCOBA** control file (see 15.42.2) via COBAF1, COBAF2 etc.

By default the flows output are total flows and therefore include all fixed flow etc. contributions, although there are alternative options (MUC and MVC) by which flows by individual user or vehicle classes may be output (see 15.42.2 below). In addition the units may be either pcu/hr or vph.

In addition, since COBA flows would normally be the weighted sum of flows from, say, an AM, off-peak and PM network **SATCOBA** accepts as input one or more “networks” (i.e., time periods) and outputs a single flow which is the weighted sum of each individual network flow. (It is assumed that all networks have the same “topology”.) Alternatively, if the parameter SUMNET = F (15.42.2), each network flow is output separately

Next **SATCOBA** generates a (partial) data set which contains certain fixed data such as the link distances. The full COBA input file requires further information such as lit/unlit which is not available from within a **SATURN** network file on its own.

Finally **SATCOBA** generates a set of turning proportions at each (internal) simulation junction (the “turning matrix” in COBA terminology) with a directionality flow factor for 2-way roads.

Thus the output from **SATCOBA** is a text file (extension .cba) which contains four sections:

- ◆ a network definition section (COBA KEY 042)
- ◆ network flows (KEY 056)
- ◆ network fixed data (KEY 060)
- ◆ the “turning matrix” at each junction (KEY 082)

all in a format specified by COBA and which we need not specify in detail here.

To run **SATCOBA** (which can effectively only be run via its bat file) type:

```
SATCOBA net1 net2 net3 ... KR control
```

where net1.ufs, net2.ufs, net3.ufs ... are the output files from different time periods whose (factored) flows are to be (optionally) added together. The output (text) file would be net1.cba.

A control file, control.dat, which sets/over-writes various parameters may be optionally defined via the bat file. If none is defined a “null” default file, satcoba0.dat, is used which basically accepts all program defaults. See 15.42.2

Like **SATTUBA**, **SATCOBA** is very much “work in progress” - comments on a postcard please to DVV.

15.42.2 The SATCOBA Control File

The control file consists of a standard set of namelist parameters headed by &PARAM and terminated by &END. The default file satcoba0.dat sets all defaults. The following parameters may be set:

Table 15.1 – SATCOBA Namelist Parameters

OPTION	TYPE	DEFAULT	INTERPRETATION
NAMES	Logical	T	If T use standard node names in the output file ; if F use map-based sequential numbers 15.42.6
DEMAND	Logical	T	If T use demand flows ; if F use actual flows
SUMNET	Logical	F	If T add the link flows from each input network and output their sum ; if F output individual flows
MIDLF	Logical	F	If T define simulation link flows mid-link, not downstream
MILES	Logical	F	If T output link distances in miles ; else kilometres
MAJORM	Logical	F	If T the “turning matrix” for all priority junctions is output in the order of a major arm first followed by a minor arm
MUC	Logical	F	If T flows are output separately for up to 3 user classes, all from network 1, in the KEY056 records
MVC	Logical	F	If T flows are output for up to 3 vehicle classes from network 1 in the KEY056 records
PCUS	Logical	T	If T user/vehicle class flows are output in units of pcu/hr; if F they are converted into veh/hr. N.B. This does not apply to total flows, only disaggregate flows.
COBAF1	Real	1.0	Factor to be applied to the flows on input network....
COBAF2	Real	1.0ditto network 2 up to COBAF4
KNOB	Integer	0	The SATNET KNOB field used to define COBA link numbers ; see 15.42.3
FILKNB	Character	Blank	The input file used to define COBA link numbers ; see 15.42.3
FILNOD	Character	Blank	The input file used to define node numbers; see 15.42.6

Notes:

- 1) NAMES will default to F, i.e., sequential numbers, if the standard **SATURN** node names exceed 4 digits since 4 digits is the maximum permitted within COBA format
- 2) MUC = T will only work if (a) only one network is being processed and (b) if the number of user classes is less than or equal to 3 in that network. If not, it is automatically replaced by F. Note that the limit of 3 is due to a limit imposed by COBA in its KEY056 record formats.

- 3) Similarly $MVC = T$ will only work if (a) only one network is being processed and (b) if the number of vehicle classes is less than or equal to 3. Vehicle class flows are obtained by summing over their constituent user class flows. (Clearly both **MUC** and **MVC** cannot be **T** at the same time.)
- 4) Post 10.9 user and/or vehicle class flows may be output as either PCU/hr or vph depending on whether parameter $PCUS = T$ or **F**.

15.42.3 Defining COBA Link Numbers using KNOBS data

The default link numbering system used by **SATURN** to define link numbers in the created cobra-formatted network is, as mentioned above (15.42.1), to use the (essentially arbitrary) numbering system used within **SATURN** assignment networks. However it is also possible for the user to define their own set of link numbers using the “KNOBS” input facility to **SATNET**; see 15.14. This option is controlled by the namelist parameter **KNOB** in the **satcoba** control file (15.42.2).

Thus, if **KNOB** = 1, then the link numbers are those defined within **KNOBS** field 1, etc., etc. If the input **KNOBS** value for a particular link is 0 then that link is not included in the newly created **satcoba** network; this facility therefore allows the user to “select” those links which are to be included in the **coba** files via **KNOBS** data.

The **KNOBS** data is, by default, that input via **SATNET** into the network .ufs files but it may also be input directly into **SATCOBA** via the namelist parameter **FILKNB** (15.42.2) which defines the name of an input file. Format conventions for the file **FILKNB** are as per inputs to **SATNET** (15.14.5).

Note that **KNOBS** data are essentially input and stored as “real” data but when used in this context they are rounded off to the nearest integer

15.42.4 Common COBA Link Numbers in Multiple Networks

One very useful application of using **KNOBS** data to define link numbers is that it allows two (or more) networks (e.g., a do-minimum and a do-something) to use the same definitions of link numbers. To do so the user must first create a “text” data file for the “base” network which contains one record per **COBA** link containing three integers: the link A-node, its B-node and the corresponding link number used to define that link in the output .cba **COBA** file. We propose a standard extension of .cln (Coba Link Number) for such files such that **net.ufs** would produce a file **net.cln**.

To create a .cln file use **SATDB** and choose (starting in the Master Menu):

- 6 – Miscellaneous Data Input
- 12 – **COBA** Network Link Numbers
- 13 – Dump the Full Data Base to an ASCII File (Master Menu)

and create the file with extension .cln. (A batch file to do the job automatically could be created if desired – requests/bribes to DVV!)

To use a file, say net_base.cln, within another network, net_ds.ufs, you must first create a “control file” coba_ctl.dat which might contain:

```
&PARAM
KNOB = 1
KNBFIL = 'net_base.cln'
&END
```

and then run **SATCOBA** via:

```
SATCOBA net_ds KR coba_ctl
```

The output COBA file net_ds.cba would then contain, inter alia, flows on all the links contained in net_base.cln using the same link number conventions. Note that links which are in net_ds but **not** in net_base.cln would not appear in the .cba file. However they are listed in the output .lpd file in order to help the user decide whether to include them somewhere else within the coba file. Equally links which are in net_base.cln but **not** in net_ds would not appear in the output .cba file. Thus the only function of the .cln file is to supply link numbers, not network structure.

15.42.5 Viewing COBA Link Numbers

The link numbers used in the output COBA network may be displayed (Version 10.5 and onwards) via **P1X** and/or **SATDB** but **only** (at the moment) if they are based on **SATURN** assignment link numbers as opposed to user-set KNOBS data. Thus they appear as option 12 under the “Miscellaneous Data Input” sub-menu from the **SATDB** top menu.

In addition the links used may be selected under Link Selection in **SATDB**. (Recall that for 2-way links only one direction is “used”, in general A-B rather than B-A where A has a lower node number than B.)

To display the user-set link numbers simply access the KNOBS data element used, preferably with the links “selected” as above.

15.42.6 Alternative / Sequential COBA Node Numbers

While it is generally preferable to use the “standard” **SATURN** node numbering system within COBA networks it is not always possible. In particular, COBA requires that node number have a maximum of 4 digits so that, if your **SATURN** network uses 5-digit numbers then they will have to be reduced/converted to a system that uses a maximum of 4. (One might well ask why COBA isn’t upgraded to accept 5-digit node numbers rather than **SATURN** having to resolve the problem mais c’est la vie!)

There are two alternative node numbering systems that may be used within **SATCOBA** to avoid 5-digit node numbers.

The first is based on the sequential node numbers as used internally by **P1X** to create “map networks” whereby each sequential number refers either to a zone (the first NCENTS entries) or a junction, whether buffer or simulation. Note that map sequential numbers may be viewed within the **SATDB** node data base as

accessed within **P1X** by selecting the appropriate entry from the list of node attributes.

Sequential numbers are selected within **SATCOBA** by setting NAMES = F in the control file (15.42.2) or, alternatively, it will be done automatically if the maximum node number exceeds 9999.

The second system uses an explicit input file to convert **SATURN** node numbers into a more compressed system – which could indeed be based on pure sequential numbers as above but any arbitrary conversion system may be used. To select this option (a) set NAMES = F as above but (b) define the conversion file filename as FILNOD within the control file (15.42.2).

The conversion file consists of a series of records, one per “real” node (i.e., zones are not included as they do not appear in COBA outputs), each of which contains: (a) a (real) node name (which may exceed 5 digits) and (b) its equivalent output number (4 digits or less). All COBA node number outputs are automatically converted to the new system.

The main advantage of the second system is that it may be applied to any number of different networks, for example a do-minimum and a do-something network, which have (some) different node numbers and therefore different sequential numbers. In particular a new option in **P1X** (see 11.4.2) allows for a file to be created containing the names and sequential number based on a “union” of **all** nodes.

If your network has more than 9999 sequential map numbers it cannot be used by COBA in its current form and you’re in deep doodah! Try a cordon maybe?

15.43 Bitmaps within SATURN

15.43.1 General Principles

Bitmaps are used as inputs within **SATURN** to provide a background to network plots within **P1X**; see 11.3.6. Thus instead of a blank (i.e., white) screen background an image obtained from a .bmp file is used and the network plot is over-written upon it. An example from the central area of York is shown below. Note that in this case the “network window” as nominated by **P1X** is larger than the area covered by the bitmap so that there is a blank surround to the bitmap. Had the bitmap covered a wider region than the network window then the appropriate region of the bitmap would have been selected and suitably expanded. Thus a very useful property of the bitmap displays is that they “move” with the network window.



Within this particular context bitmap files must be of either “.bmp” or “.pcx” format, as opposed to, e.g., .jpg, .gif, etc. formats (although .jpg is allowed as output; see 11.3.6). However other graphical formats may almost certainly be converted into a .bmp format by making use of standard software such as Paint.

Where or how the bitmap file is obtained is not strictly relevant; it might be downloaded from, e.g., OS sources, scanned from a road map, dumped from a GIS software package or even output from a different run of **P1X** for a different network. The important thing is that it be in .bmp format and, equally important, that the “area” which it covers be identifiable.

Thus in order for **P1X** to draw a bitmap background within the windowed area covered by a network plot it is necessary to know (a) the precise area covered by the network window and (b) the full area covered by the .bmp file, in effect the co-ordinates of its 4 corners, so that the degree of overlap between the two may be ascertained. This may not sound too difficult, indeed most of the time it isn't; the tricky thing is being able to obtain the co-ordinates of the bitmap and of the network within the same reference system. (Note that it is the network “window” which “controls” the region plotted and that the bitmap must be manipulated to fit onto the area chosen by the **P1X** network window rather than the other way around.)

Thus for every .bmp file used by **P1X**, say picture.bmp, it is necessary to set up a further (very small) file, named picture.xyb, which specifies the 4 corners of “picture” using the same coordinate system as that used by the network (i.e., the co-ordinates as used within the 55555 network data section and independent of XYUNIT (6.8)). .xyb files consist of a single record containing 4 (real) values in the following order:



- ◆ XMIN - the east-west co-ordinate of the lower left-hand corner;
- ◆ XMAX - ditto for the upper right corner;
- ◆ YMIN - the north-south co-ordinate of the lower left-hand corner;
- ◆ YMAX - ditto for the upper right corner.

Optionally, a second record may be included which contains the “intensity scaling factor” to be applied for that particular bmp image; see 15.43.6.

Note that the “units” of XMIN etc. should be the same as the units of the node X,Y co-ordinates as defined under the 55555 records in the original network .dat file (see 6.8). Thus if XYUNIT = 10.0 so that the co-ordinates are defined to the nearest 10 metres then XMIN etc. should also be the nearest 10 metres. However, as noted in 15.43.2, we strongly recommend that **all** co-ordinates are defined in units of metres to minimise confusion,

The .xyb file may be most conveniently set up the user assuming that the information is known in advance through knowing the source of the image.

Alternatively, if a bitmap is input into **P1X** without a corresponding .xyb file being located, the user is offered the option to “calibrate” the .bmp file as detailed in 15.43.3.

15.43.2 Co-ordinate Systems

Once again, the importance of having a common system of co-ordinates for both the network and the .bmp files cannot be over-emphasised. The simplest method is to base both upon some standard system such as, in the UK context, the Ordnance Survey (OS) co-ordinates with both east-west (X) and north-south (Y) co-ordinates defined to the nearest metre. (Very often the “leading digits” as used by the full OS system may be dropped; e.g., if all your X values begin with, say, 45 followed by 4 digits then it is easier to drop all the 45’s and stick to the final 4 digits.)

Note that defining co-ordinates as metres implies that XYUNIT should be set to 1.0 (its default). And we **strongly** recommend that such a system be adopted throughout.

This means that networks which, for one reason or another, have been defined from their “birth” using OS-based co-ordinates will find it much simpler to use .bmp files than networks which are based on a more arbitrary set of co-ordinates. In the latter situations it is probably far easier in the short term to convert .xyb co-ordinates to the arbitrary system (see 15.43.3 below) rather than trying to convert the original X,Y co-ordinates. However, on the other hand, there are considerable longer-term benefits, e.g., being able to interface with various GIS-based data sources, in using OS-based co-ordinates and it may therefore be a good idea to “bite the bullet” and transform your arbitrary co-ordinates into OS NOW! For further advice on how to do so please contact DVV.

15.43.3 “Calibrating” .bmp files

By “calibration” we refer to the process by which the four corners of a particular .bmp file are established in terms of the co-ordinates used by the network. Ideally,

as we have pointed out above, both should be based on the same system and the coordinates of the four corners should be established a priori. However, in the absence of such information, a procedure has been established within **P1X** to obtain this information.

In order to carry out this procedure the user must be able to identify the (network-based) co-ordinates of two points within the bitmap display. Ideally these two points should be as far away from one another as possible and near one or the other diagonals. Normally the points will correspond to nodes for which the network co-ordinates are known, although in principle they could be any points which can be easily identified in network terms.

Formally the bitmap is displayed with a thin red strip added along the edges and a further red cross displayed in the centre. The user is asked, first, to move and click the mouse over the red cross (in order to confirm the centre point in pixels) and next to click on two points and input their X,Y network co-ordinates. The four corner points may then be easily calculated via a simple linear transformation.

A practical problem which arises in the above procedure is that it is not possible within **P1X** to simultaneously display both the bitmap **and** the network (prior to calibration). We therefore recommend first viewing the bitmap (e.g., enter it within **PMAKE** or use any other graphical system such as Paint) in order to identify the two points(nodes) to be used and then viewing the network in **P1X** and using the X,Y monitoring option under Information to determine - write them down! - the two sets of co-ordinates. Armed with this information you can return to **P1X** and the bitmap display to complete the calibration. Both procedures may be carried out at the same time by having two program windows open - or even two computers!

This process is probably most easily done by using **PMAKE** to select, view and calibrate the .bmp file and then exit the program. Trying to do the same process within **P1X** leads to problems of having both a bitmap and an (incompatible) network both trying to define a network window.

Once calibrated a .xyb file is automatically created (so that picture.bmp spawns a file picture.xyb) and which will from then on be opened at the same time as the .bmp file is opened.

15.43.4 Outputting Bitmaps to Hard Copy Devices

Bitmap files may be included in output hard copy plots in the same way that they appear on the screen but there may be certain restrictions.

Bitmap files are held by **P1X** in internal memory and the array thus used has dimensions (2001 by 2002 by default but may be increased by request) which will normally cover the pixel dimensions set by the screen resolution. However hard copy devices may well have pixel resolutions which exceed the above limits by considerable margins and therefore the program is unable to print "full" background bitmap images to such devices. Currently only the "upper half" of the bmp file is printed (so as to use as much as possible of the available memory); a more permanent "fix" is currently being sought.

Note that a pre-10.6 problem whereby the network and the bmp file were printed with slightly different scales (by 5%) has been corrected.



There is, however, no problem in dumping the current plot plus bitmap display to a .bmp output file or the clipboard and subsequently outputting to a printer (but with some loss of resolution).

15.43.5 Bitmap backgrounds within Node Graphics

In principle a bitmap image could equally well be used as the background to node graphics displays. At the time of writing the necessary co-ordinate transformations have not been worked out but it will happen soon!

15.43.6 Changing the Intensity of Bitmap displays

If the bitmap display is too "intense" it may make the over-printed **P1X** displays difficult to see. This may be corrected by reducing the intensity of the background by setting a "scaling" factor between 0 and 1 (set within Display/Background or via the .xyb file, 15.43.1); the lower the factor, the "whiter" the bitmap. The default scaling factor is 1.0 but may be changed globally via the namelist parameter SCABMP in the **P1X** preferences file p1x0.dat.

This option may be particularly useful within **PMAKE** when a new network is being traced on a bitmap image.

15.43.7 Maximum Bitmap File Sizes

The "size" of a bitmap file which can be read by **P1X**, i.e., the number of pixels in both the horizontal and vertical dimensions, is limited to certain upper limits, e.g., 2003 x 2004. This may create problems for users since it is quite easy to create .bmp files with an almost infinite number of pixels depending on the geographical size of the file (i.e., the number of square kilometres) and its resolution. Some compromises may be necessary within **SATURN**.

For example, if the .bmp file covers an area of 1 km x 1 km with 2,000 pixels in each dimension then one pixel (in the .bmp file) covers 0.5 metres which, for most purposes should provide sufficient resolution. However, if the user "windows in" to a 20 x 20 metre display in order to look at a single junction then each pixel in the original .bmp file covers $0.5/20$ equals $1/40$ -th of the screen and the image would be very "chunky". That problem could be avoided by creating (outside **SATURN**) a .bmp which covered just the 20x20 area with the full resolution of 2,000 x 2,000 pixels. However, that would not be a very useful background for a window of 1 km x 1 km.

It is, of course, possible within **P1X** to prepare several different input .bmp files and to "swap them over" depending on the current window, but it is not highly satisfactory.

Such problems could also be removed by increasing the maximum dimensions which **SATURN** can handle, e.g., to go from 2001 x 2002 to 10,000 x 10,000 and this can be done easily enough when compiling the program. However this may create other problems in that a .bmp file of 10,000 x 10,000 pixels requires 6×10^8 bytes, i.e., 0.6 GigaBytes, within **P1X** which might, in combination with all the other demands from **P1X**, exceed the RAM provided on most machines and therefore slow down the overall execution speed dramatically. In addition, even if there were sufficient internal RAM, the cpu time required to input and manipulate very large .bmp files may still be excessive for most user requirements.

The size and resolution of .bmp files may be easily manipulated using standard Windows graphics packages such as MS Paint or MS Picture Manager for example.

15.44 Defining Extra Bus Travel Times (BUSSPK and BTKNOB)

The travel times associated with bus routes are normally calculated by summing the standard link and/or turn times associated with other vehicles along the route. However it is possible to “supplement” these times to represent the additional effects of, e.g., bus dwell times at stops or bus speeds being slower than cars.

The extra time may be introduced using either:

- (i) an additional time proportional to the total distance over the whole route,
- (ii) explicit link by link extra travel times coded as “knobs”.

In both cases the additional travel times are calculated once and for all per route when the network is built within **SATNET** and then recorded within the .uf* files. This means, for example, that it is not possible to “view” the extra travel times per link using the bus “joy ride” display within **P1X**. The extra times are reported both within the individual route statistics and in the more aggregate statistics under option 6 in **SATLOOK** (but **not**, N.B. in the total pcu-hrs etc. reported under either options 4 or 5 in **SATLOOK**).

Under (i) the proportionality between extra (i.e., stop) time and distance in km is set by the parameter BUSSPK (Bus Stop Seconds Per Kilometre) defined within the &PARAM namelist records in the network .dat file. In the event of there being more than one “bus company” BUSSPK may be subscripted so that, e.g., BUSSPK(3) = 0.04 would set the specific value for bus company 3.

Under (ii) link data must firstly be defined using the KNOBS facility (see 15.14 - any of the 3 input methods may be used) and a proportionality factor BTKNOB(b,k) set > 0 where b refers to a bus company and k to a KNOB data set (1 ... KNOBS). The units of BTKNOB are assumed to be seconds per whatever units that particular knob data field is in. Again BTKNOB may be defined within the &PARAM namelist records (6.3.3).

Note that in using namelist input to set BTKNOB which is a 2-dimensional array you may need to use the array based input, so that:

BTKNOB(3) = 0.3, 0.0, 4.0

would set the elements (3,1), (3,2) and (3,3) to 0.3, 0.0 and 4.0 respectively. See note 17, Appendix A. (In fact BTKNOB is the only variable to which array-based inputs may be applied.)

Both methods are fairly “aggregate”, even crude, in that they do not allow you to define stopping times by links by bus route (unless you have one route per company which is not very practical). However they are a start and all requests for more will be listened to.

15.45 Representing Walk / Pedestrian Networks

Traditionally **SATURN** networks represent roads down which cars travel and their travel speeds are vehicle speeds. However there is no hard and fast reason why every “link” in a **SATURN** network should be a road link and it is quite possible to “fool” **SATURN** into treating a link as though it were part of the road network while clever old you give it characteristics more appropriate to a pedestrian than a car.

How such coding “tricks” are accomplished is of course up to the user. One common method (assuming that the walk networks are superimposed on a simulation network) is to code the walk links and nodes as part of the buffer network with (low) fixed speeds/times, a flat flow-delay curve (power = 0) and, effectively, infinite capacity (e.g., 99999). The walk links are then connected to the simulation network via external simulation nodes. The origin/destination zones to/from which trips exit/enter are then connected to walk nodes rather than simulation links.

Thus an o-d trip with a destination which requires walking will be assigned a route which starts at a “normal” zone and initially follows a set of “normal” (i.e., car) links until it reaches the walk links through which it can reach its destination. At this point, in effect, the car becomes a pedestrian although as far as **SATURN** is concerned nothing has really changed - it is simply finding a route through a network. The same principle works in reverse for trips which start as walk trips.

If, very naturally, the point of transition from car to walk is at a car park note that tolls (and capacities) may be associated with the car park as described in Section 20.5.3.

Certain “presentational” problems may occur in **P1X** if, for obvious reasons, the nodes in the walk network have the same (or very close) co-ordinates to “real” junctions since there will be a high degree of overlap between the walk network and the road network. This may be avoided by assigning a unique set of capacity indices to the walk links (a good idea anyway) and then excluding those capacity indices from the network link plots as described in 11.6.1.4 and/or note 4, 11.6.4.

15.46 DBDUMP & P1XDUMP: Dumping Link Data to Text Files

15.46.1 DBDUMP: Dumping Data via SATDB

A batch file dbdump.bat based on the program **SATDB** has been set up in order to provide a simple method to dump selected link data from a binary .ufs file into an ascii text file. For example, the command:

```
Dbdump net flows.txt 4503
```

dumps the demand flows (DA code 4503) from net.ufs into a file flows.txt following the “rules” described in 11.10.9.

Various options described below are provided to control the precise “format” and contents etc. of the output file.

Tokens on the command line may be divided into two types:

- ◆ DA codes for output data items

◆ Options

DA codes are always given as numerical values; For a full list of the DA codes within a .ufs file please consult Appendix J. Note, in particular, that codes such as 3808 to represent **actual** flow by user class 1 are also permitted (see 15.21.4).

Options are always represented by characters beginning with a \$. They may be further sub-divided into two groups, link types and format.

(i) *Those that select the link types to be output*

Under link selection the following characters indicate link types to be included:

\$SL	Include simulation links (i.e. roads)
\$ST	Include simulation turns
\$SCC	Include simulation centroid connectors
\$BL	Include buffer links
\$BCC	Include buffer centroid connectors

and the composites

\$L	Include all (simulation and buffer) links
\$CC	Include all (simulation and buffer) centroid connectors

Including an X immediately after \$ indicates “exclude” that link type; e.g., \$XST implies exclude simulation turns but leave the other 4 types.

For example:

```
DBDUMP net net.txt 4503 $SL
```

would dump demand flows for simulation links only to file net.txt.

(ii) *Those that control the format*

Further options controlling the output formats, e.g., nodes in fixed columns versus free format (CSV), are being added to match some of the interactive options within **SATDB**. Thus:

\$KP5COL	Nodes are output in fixed columns of 5 or ...
\$KP6COL	... in fixed columns of 6

Note that if the node or zone numbers in the network being dumped exceed 4 digits then the output link format **automatically** allocates 6 columns per node. (The 5-column option may be set by default by defining the parameter KP5COL = T in the preferences file p1x0.dat; KP5COL = F selects 6 columns.)

Furthermore, if the filename of the output file in the command line has an explicit extension .CSV, then it is assumed that the output data format will be CSV rather than fixed columns. For example:

```
DBDUMP net net.csv 4503 $ST
```

would dump demand flows for simulation turns to a CSV formatted file, net.csv.

15.46.2 P1XDUMP: Dumping Data via P1X

A very similar batch file to DBDUMP, P1XDUMP, dumps selected data to a text file but based on P1X internal codes rather than DA codes. For example, the command:

```
P1XDUMP net flows.txt 5
```

dumps the free-flow speeds (P1X internal code 5) from net.ufs into a file flows.txt. See Appendix I for a full list of codes.

To request a particular user class for a user-class dependent variable (such as link flows) the class is indicated by appending 'Un' to the internal code; for example:

```
P1XDUMP net flow_uc4.txt 40U4
```

dumps the (demand) flow for user class 4.

The options \$SL etc. described above under DBDUMP apply equally under P1XDUMP.

15.47 CLICKS: Variable Free Flow Speeds by User Class

15.47.1 General Principles of CLICKS

The "CLICKS" parameters represent a somewhat simplistic method to model the clearly evident fact that on, say, motorways, heavy lorries travel at a lower speed than cars (whether due to speed restrictions or to vehicle characteristics – or both). Setting a parameter CLICKS(2) = 100 signifies that user class 2 vehicles (e.g., lorries) have a maximum speed of 100 kph on all roads (both buffer and simulation).

Input values of CLICKS are included as subscripted variables within &PARAM in the network .dat file; the default of 0.0 signifies that there is no maximum speed restriction for that user class. Units are in kph.

CLICKS only has an impact on **road** links (i.e., buffer and/or simulation roads, not simulation turns and **not** centroid connectors) whose free-flow speed is in excess of the input value(s) of CLICKS. In modelling terms it is represented by a **fixed** time penalty per user class equal to the difference in time between a vehicle travelling at the input free-flow speed and at CLICKS; if the free-flow speed is less than or equal to CLICKS then the time penalty is zero. (And equally if CLICKS is not set the time penalty is zero.) (But see 15.47.3 for an alternative model with **variable** time penalties.)

Generally speaking CLICKS is applied to links which have a speed-flow curve, but they may equally well be applied to links which have a flat speed-flow curve. If the link does have a speed-flow curve then it would be expected that CLICKS would be somewhere in between the maximum free-flow speed and the minimum speed at capacity; if not a Serious Warning 159 is generated.

In general we would expect that CLICKS(1) = 0 on the assumption that user class 1 represents cars and that cars can always travel at the maximum speed, i.e., the

free-flow speed coded for each link, and that there is no need to impose an extra travel time on them. But, if you do want to impose penalties on all user classes, go right ahead!

The fact that the penalty is fixed means that it is included within the travel time (for that user class) under **all** conditions, i.e., all speeds and all flows. To give a simple numerical example, consider a link 1 km long with an input free-flow speed of 120 kph (which presumably applies to cars, user class 1) but with user class 2 (lorries) having been assigned $\text{CLICKS}(2) = 100$. Under free-flow conditions cars take 30 seconds to travel the 1 km. at 120 kph, lorries take 36 seconds at 100 kph. Hence the fixed time penalty is 6 seconds for user class 2.

The end effect is identical to adding a penalty of 6 seconds within the 44444 data records for user class 2 on that particular link; in both cases, the 6 seconds is simply added to the minimum generalized cost for that link as used within the assignment. Therefore, in both cases, the extra time may influence their route choice. However, in practical terms, it is much simpler to set a **single** parameter under &PARAM than to include explicit penalties for every link which requires it. Although, on the other hand, explicit penalties under 44444 can be made much more precise.

A possible modelling disadvantage of assigning a **fixed** penalty may be seen, using the above 1 km long motorway link, by assuming that the speed at capacity for that link has been coded as 40 kph. Under capacity conditions it is perhaps more realistic to assume that both cars and lorries travel at the same bumper-to-bumper speed. In fact the extra 6 second penalty on the lorries will bring their effective speed down to 37.5 kph, an “error” of 2.5 kph. On the other hand it might be argued that even under bumper-to-bumper conditions cars will still have a slight advantage over lorries by being able to weave in and out a bit more and that maybe a differential speed of 2.5 kph is not too bad an estimate of that effect. It is up to user to judge whether or not this represents an “acceptable” model.

At this point, users may well be asking why there cannot be two (or more) different speed-flow curves per link per user class which may differ at free-flow but come together at capacity. In principle, there is no reason why such differential curves could not be defined. Unfortunately, for complicated theoretical reasons, the multiple user class assignment algorithm used within **SATURN** (see 7.3) requires that there can be only **one** cost component which is flow-sensitive and that that component (i.e., time) is common across all user classes. Fixed cost components may, however, differ between user classes (e.g., the evaluation of distance in generalized cost seconds) and the differential time penalties must therefore fall into that category. Otherwise multiple equilibria may occur.

In practical terms, as mentioned above, the use of CLICKS may influence route choice by user class. The extra time penalties will automatically be included within any O-D skim that includes time.

Summary statistics listing the total **extra** travel time in terms of pcu-hours incurred under CLICKS are given in the .lpt files and within the various list options under **SATLOOK** and **P1X**. The outputs are disaggregated by: user class, simulation/buffer, this/next/total time period and by capacity index. In principle these totals could (and possibly should) be added to the cruise time etc. totals which appear in standard output tables; however, for the time being, this has not

been done in order to allow users to think about exactly how they would wish to have the data presented.

In summary setting a value for CLICKS is an extremely simple method to represent differential speeds by user class but some users might feel that the possible “errors” introduced at high flow levels may outweigh the advantages of simplicity.

15.47.2 Disaggregated Levels of CLICKS (KLUNK)

Prior to the release of version 10.7 the value of CLICKS for a particular user class applied equally to **all** links (excluding centroid connectors); post 10.7 CLICKS may be disaggregated either by a link’s capacity index or, ultimately, per individual link. The choice is set by an Integer input parameter KLUNK defined under &PARAM.

Thus, if KLUNK = 0 (the default) CLICKS(u) applies to **all** links for user class u. If KLUNK = 1 then, in effect, CLICKS becomes a two-dimensional array such that CLICKS(v,k) defines the value of CLICKS for **vehicle** class v for all links with capacity index k. If KLUNK = 2 then every individual link can (potentially) have its own unique set of CLICKS values.

N.B. With KLUNK > 0 the values of CLICKS are defined by vehicle class, **not** user class, but, recall from section 5.8, that each user class should be associated with a particular vehicle class (as set on the network 88888 data records) so the general principle that each user class has a maximum speed per link is retained.

The reason for using vehicle classes directly rather than user classes is that there are generally a much smaller number of vehicle classes (e.g., cars, light and heavy lorries) and it is really the type of vehicle that determines maximum speeds, not whether a car driver chooses a route based on minimum time or distance. It also allows users to introduce new definitions of user classes but with the same set of vehicle classes (e.g., split a user class Work into HB Work and NHB Work but both are part of Vehicle Class 1) without having to update CLICKS(.). Plus it allows data files such as FILVSD files (see below) to apply to more than one network as long as both networks use the same conventions for capacity indices and vehicle classes

15.47.2.1 KLUNK = 1 (Disaggregate by Capacity Index)

To input variable values of CLICKS by vehicle class under KLUNK = 1 the user must either:

- (a) prepare a text file (formatted as below) and define its file/pathname via the text parameter FILVSD input under &PARAM in the network .dat file, or
- (b) include extra records within the 33333 buffer data (new with 10.9).

15.47.2.2 FILVSD File Input:

The file must contain one record for each Capacity Index for which CLICKS values are required with the first entry field containing the (integer) Index and the following values containing the (real) maximum speeds for Vehicle Classes 1, 2, 3.... The format is essentially free – each item must be separated by either a space or a comma from its neighbours; i.e., CSV is acceptable. All values for

Capacity Indices **not** included in the file default to zero (i.e., maximum speeds are not applicable) as do missing or zero values per Vehicle Class as input.

For example, if capacity index 1 refers to a road type where vehicle classes 1 and 2 have normal link-dependent speed-flow curves but vehicle class 3 (HGVs maybe) has an upper speed limit of 88 kph the relevant (CSV-formatted) record would read:

1,0,0,88

and the maximum speed of 88 kph would then apply to all user classes associated with vehicle class 3.

The file is terminated either explicitly by 99999 in columns 1 to 5 or simply by the end of the file.

As with CLICKS(1) under KLUNK = 0 we anticipate that CLICKS will not apply to the vehicle class "cars" (generally 1) so that one of the input fields (the first) will be uniformly zero.

N.B. Note that the input maximum speeds are defined by vehicle class, **NOT** by user class.

15.47.2.3 Extra 33333 Data records

To define the maximum speed for a particular combination of vehicle class and capacity index the user must include a record in the network .dat file, similar to default speed-flow records per capacity index (15.9.5), under 33333 (see note (16), section 6.6) with:

- (i) The character V in column 1 followed (in free format) by:
- (ii) The vehicle class
- (iii) The maximum speed (CLICKS) in kph
- (iv) The capacity index

Thus the 3 data fields following V may be in any columns as long as they are separated by either a blank or a comma. However, for "visual" reasons, we would strongly recommend having the vehicle class in columns 2-5, the maximum speed in columns 11-15 and the capacity index in columns 43-45 as done (in the latter two cases) for default speed-flow curves¹. In fact it would make good sense to include any specific maximum speeds by vehicle class/index immediately **after** the equivalent default speed-flow record to make any comparisons of data much more transparent.

Alternatively they might be contained in a separate file referenced by \$INCLUDE.

Note that fields (ii) and (iv) must be integers (as well as, clearly, being valid numbers) but the speed (iii) may be input as a real value. Normal logic checks

¹ Note, if DUTCH=T has been set to permit longer node numbers, the matching columns will be 21-25 and 53-55 respectively

that, e.g., CLICKS speeds are less than normal speeds, etc. etc. are carried out and error messages produced as necessary.

The V-records will be ignored, in the same way that default speed-flow curves by capacity index with a D in column 1 are ignored, when the buffer network is built.

Note as well that if any V- records are included under 33333 and KLUNK = 1 then any reference to FILVSD is ignored; you cannot therefore use both methods to define KLUNK = 1 data at the same time.

15.47.2.4 KLUNK = 2 (Disaggregate by Link)

Not yet implemented. This will probably be done in conjunction with the 33333 input formats above but per link rather than per capacity index. Thus the HGV (say) speed-flow curves will be defined independently per link, e.g., as a flat maximum speed until it intersects with the “normal” speed-flow curves.

15.47.3 Fixed Maximum Speeds: CLIMAX

An alternative to having a **fixed** difference in travel times per user/vehicle class is to specify a fixed maximum **speed** by setting a parameter CLIMAX = T under &PARAM. If CLIMAX = T then it is assumed that the speed-flow curve for a particular user (or vehicle) class is **fixed** at CLICKS independent of the total link flow until the car speed drops below that value, at which point the car and “other” speed-flow curves coincide. In other words, the penalty time imposed under CLICKS is not fixed but gradually reduces from its maximum value at flow equal zero until it goes to zero at the point where the car speed equals CLICKS.

In almost all cases CLIMAX is used to model fixed speeds for HGV's which are less than car speeds under free-flow conditions up to the point where car and HGV speeds become equal. Whether or not this is a better representation of the differences between HGV and car speeds is of course up to the user.

In modelling terms the fixed travel time per link applied to, e.g., HGV's is adjusted within the simulation-assignment loops within **SATALL** at the end of each simulation step, effectively at the same time as the link speed-flow curves per turn are updated for the next assignment step via the simulation,

For example, consider (as in 15.47.1) a link which is (a) 1 km long, (b) has a maximum speed (CLICKS) for HGVs of 100 kph and (c) a speed-flow curve defined with a speed of 120 kph at free-flow. With CLIMAX = F the time penalty for HGVs would be fixed and equal to the difference between $1/100$ and $1/120$ hours, i.e., $36 - 30 = 6$ seconds. With CLIMAX = T the penalty would be initially set to 6 seconds but if, at the end of the first assignment, the flow on that link were sufficient to reduce the car speeds to 110 kph the new HGV penalty would be $1/100 - 1/110 = 3.27$ seconds. If car speeds dropped to less than 100 kph the HGV penalty would go to zero.

By introducing an “interaction” between two different sets of costs and flows on the same link (in the same way that the delays to minor arm flows at a T-junction are affected by and interact with flows on the major arm) an extra complication is introduced into the assignment-simulation loops which, in principle, could lead to non-convergence and/or multiple equilibria. However, compared to the interactions that go on within the normal simulation process, the sort of

interactions we are talking about here are relatively “weak” and, touch wood, should not lead to any extra convergence problems. Statistics to demonstrate the degree of convergence, e.g., the ratio of the total absolute changes in penalties to the total penalties, are printed each time the adjustments are made and should converge to (near) zero.

15.47.3.1 Defining CLIMAX

CLIMAX is a Logical variable set in the network .dat files under &PARAM which is set by individual user classes. Thus CLIMAX(3) = T turns “on” the CLIMAX option for user class 3. The default is .FALSE. for all user classes.

Setting CLIMAX = T (i.e., no subscript) sets CLIMAX(n) = T for **all** user classes n by default unless a specific record is included for an individual user class.

Note that CLIMAX(n) is only relevant if CLICKS(n) has been set. So, unlike CLICKS, there is no problem with having CLIMAX(n) = T for all user classes since normally there should be at least one user class to which CLICKS is not applied. In addition CLIMAX() is a function of user class only and applies to **all** links (or, strictly speaking, all links where CLICKS is less than the free-flow speed).

15.47.4 Link Times Incorporating CLICKS

By default the link travel times generally calculated and, e.g., displayed by **P1X** do **not** include any extra times associated with CLICKS for particular user classes; in effect they represent travel times by cars. Post release 11.3 it is possible to calculate and display an average travel time representing a flow-weighted travel time over all vehicle classes. Thus:

$$T_w = t + \sum_u \Delta t_u V_u / \sum V_u$$

Where t_w is the weighted travel time

T is the “normal” time

Δt_u is the extra travel time for user class u

V_u is the flow for user class u

Where flows are expressed either in units of vehicles/hr or PCU/hr.

The weighted times are calculated within **SATALL** once the assignment-simulation loops have converged and are then stored in DA codes 4008 (weighted by vehicles/hr) and/or 4018 (weighted by PCU/hr). They may then be viewed in, e.g., **P1X** as normal link data.

In the event that all user classes have the same PCU weight, ie., 1.0, both measures of time are identical and 4008 is not included.

Current applications of weighted times include validation of timed routes (see 11.7.2.1) and joyrides (see 11.8.2.3). Further suggestions are most welcome.

15.48 UNIQUE: Combined Queues within the Buffer Network

The option “UNIQUE” was introduced in 10.7 in order to minimise the double-counting of V>C delays in buffer networks in certain circumstances.

More precisely, consider a series of links A-B-C-D... in the buffer network such that traffic on A-B can only exit to C (ignoring U-turns to A), traffic on B-C can only exit to D, etc. etc. Hence all links must be assigned the same demand flow V . If, say, A-B and B-C both had the same capacity C and $V > C$ then, in reality, one would expect that a queue of traffic would form on A-B at a rate $V-C$ but that the capacity on A-B would restrict or “meter” the “actual” flow to B-C to equal C and that therefore there would **not** be a second queue on B-C. Hence there should be a “queuing” delay on the first link but not on any of the flow-metered links downstream.

However, prior to 10.7 and UNIQUE, the same queue build-up and consequent delay was imposed on **all** links A-B, B-C, ..., hence “double-counting” the effects of queuing. But, if UNIQUE is set to T within the &PARAM of the .dat file, the extra delay is imposed at only **one** of the links (that with the minimum capacity which therefore represents the true “bottleneck”).

This option is useful if, say, an existing buffer link A-C is split by a mid-link node B with no other changes and the same link properties apply on both A-B and B-C.

15.49 SATURN Summary Statistics Reporting Tool (SATSTAT)

SATSTAT is a tool used to automatically extract and summarise convergence and summary performance statistics for each network(s) into a CSV file. The resulting CSV files may be then be readily imported into the associated MS Excel spreadsheet and comparisons undertaken between different networks.

SATSTAT consists of two parts: (i) a Fortran 32-bit program to extract the summary statistics; and (ii) an MS Excel spreadsheet to undertake the comparisons.

15.49.1.1 SATSTAT FORTRAN Program

For each UFS network(s) selected, the SATSTAT Fortran Program (v3.00) will:

- ◆ Automatically extract SATURN convergence, assignment/simulation statistics and queues using the standard reports available through SATLOOK, SATDB, P1X etc;
- ◆ Produces summary outputs in MS Excel CSV format of:
 - ◆ Model Convergence (NITA,NITS,%flows, %gaps,%epsilon - SATLOOK option 8);
 - ◆ Model Runtimes (SATNET, Assignment, Simulation - SATLOOK option 8);
 - ◆ Matrix Totals including Origins & Destinations within Buffer/Sim areas BUT excluding INTRA-ZONALS;
 - ◆ Simulation statistics (speed/distance/time within this time period / following time period - SATLOOK option 4)
 - ◆ Assignment / Simulation statistics (speed/distance/time - SATLOOK option 5)
 - ◆ Average Queue in the Simulation Network (SATDB DA Code 1433)

- ◆ Queue at End of Time period in Simulation Network (SATDB DA Code 1483)
- ◆ Average Delay / Vehicle (mins) (calculated)
- ◆ Congestion Index (mins / km) (calculated)

The name of the output file is the network name with a CSV extension (eg EPSOM98M.CSV).

15.49.1.2 SATSTAT Excel Spreadsheet

In conjunction with SATSTAT, the resulting CSV file(s) may be imported into the MS Excel spreadsheet called "Summary Report (v3.02).xls" via the "Import CSV" Visual-Basic macro available in the Summary worksheet. Once the MS Excel macro is run, it will ask for the name of the CSV file to be imported into the spreadsheet and become available in the selection box. If selected, the network stats will appear in the column below both in summary and more detailed form enabling comparisons to be made between different networks. A demonstration is available under Test Networks > Option 4 Run SATSTAT for Epsom.

SATSTAT has been successfully tested on all versions of SATURN 10.xx (ie 10.1 to 10.8). Please note that it does not disaggregate summary statistics by separate user class, only for "TOTAL FLOWS".

15.49.2 Worked Example

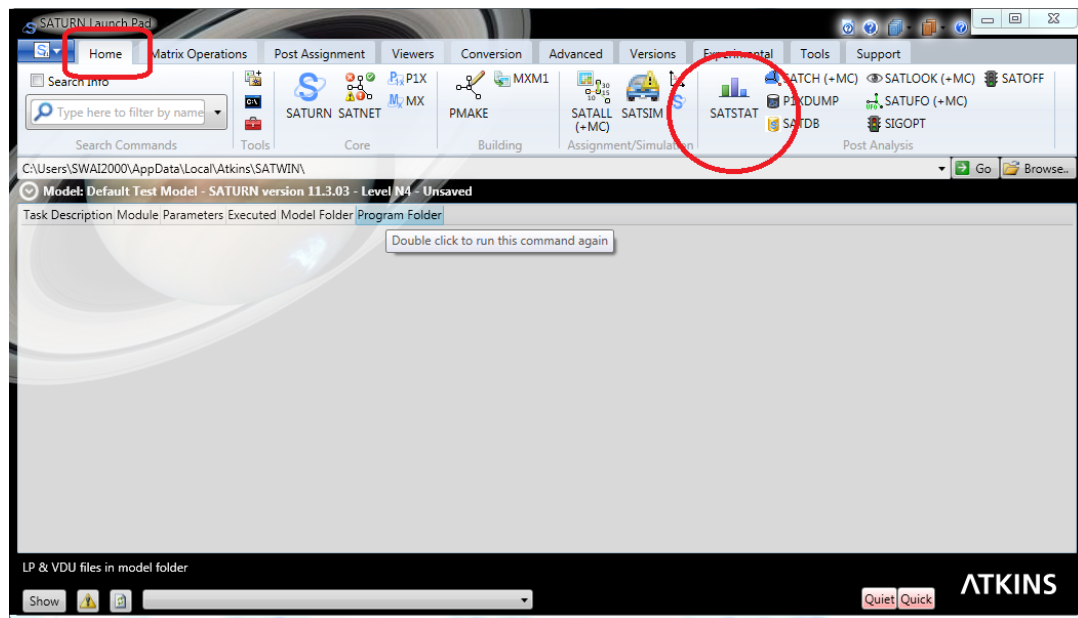
A worked example is provided in the "Test Network" menu, option 4 that will run the SATURN Epsom network for two scenarios (without/with development), run SATSTAT to extract the summary statistics and open MS Excel (using a Workspace) and import the two SATSTAT output CSV files.

The SATURN files are located in the sub-directory called "TEST\DEMO – SATSTAT". There are two sample networks: a reference case called *EPSOM98AXX.DAT* and a development scenario called *EPSOM98RXX.DAT*.

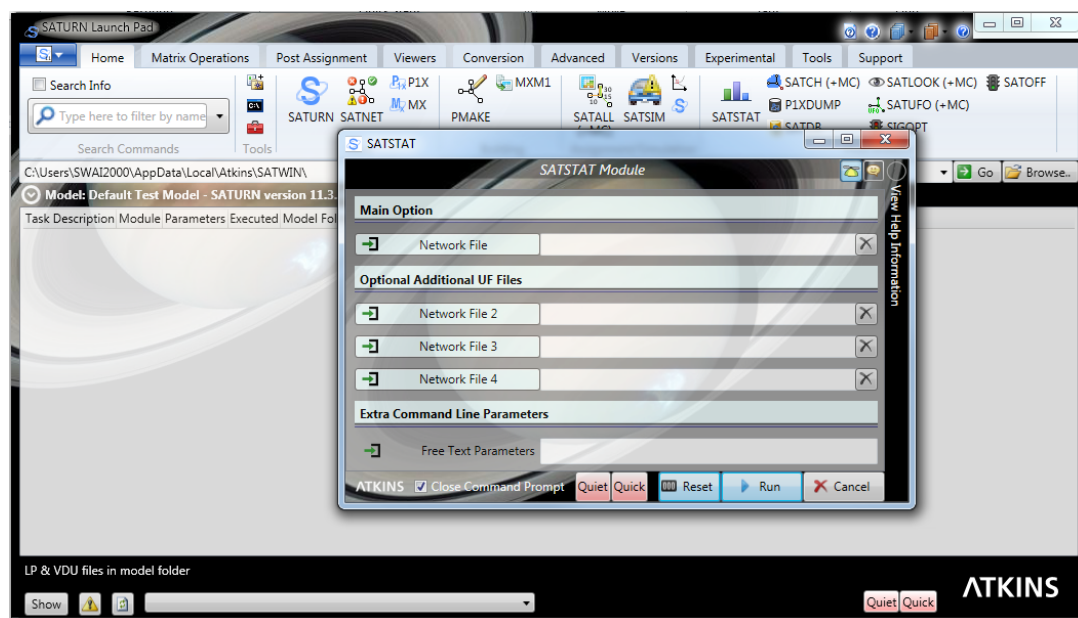
Below we step through the process rather than running it all automatically.

15.49.2.1 Running SATSTAT in SATWIN

The SATSTAT module is selected for running in the usual fashion as shown below for SATWIN11; from the Home tab, we can select SATSTAT.



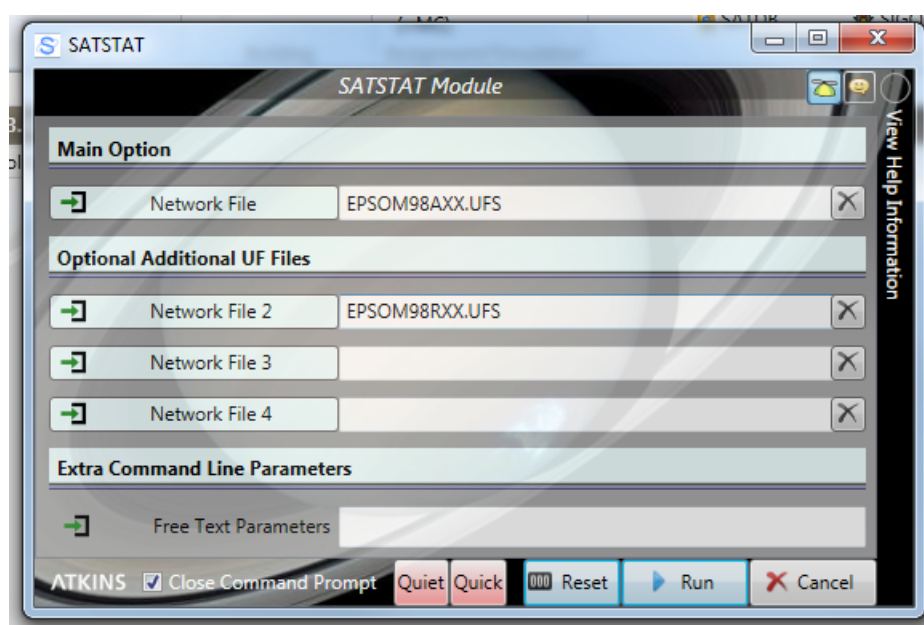
This will load-up the SATSTAT module requesting the network(s) that you wish to extract the summary statistics from:



The two networks we wish to compare are:

- ◆ Reference Case *EPSOM98AXX.UFS*; and
- ◆ Development scenario *EPSOM98RXX.UFS*.

We now select both these networks to run through SATSTAT as shown below. Note that in this example, we have changed the *Working Folder* to “C:\Users\SWAI2000\AppData\Local\Atkins\SATWIN” but the files may be located in any folder.



We now run SATSTAT and the module will now, for each network in turn:

- ◆ Run SATLOOK and SATDB to determine the version of SATURN in use;
- ◆ Run SATLOOK and SATDB to extract the summary statistics; and
- ◆ Run the SATSTAT Fortran program to generate a summary statistics file in CSV format.

The output(s) from the process will be a one (or more) CSV files with the same filename as the network UFS file. In our example, the two output files will be EPSOM98AXX.CSV and EPSOM98RXX.CSV.

15.49.2.2 Using the SATSTAT Spreadsheet

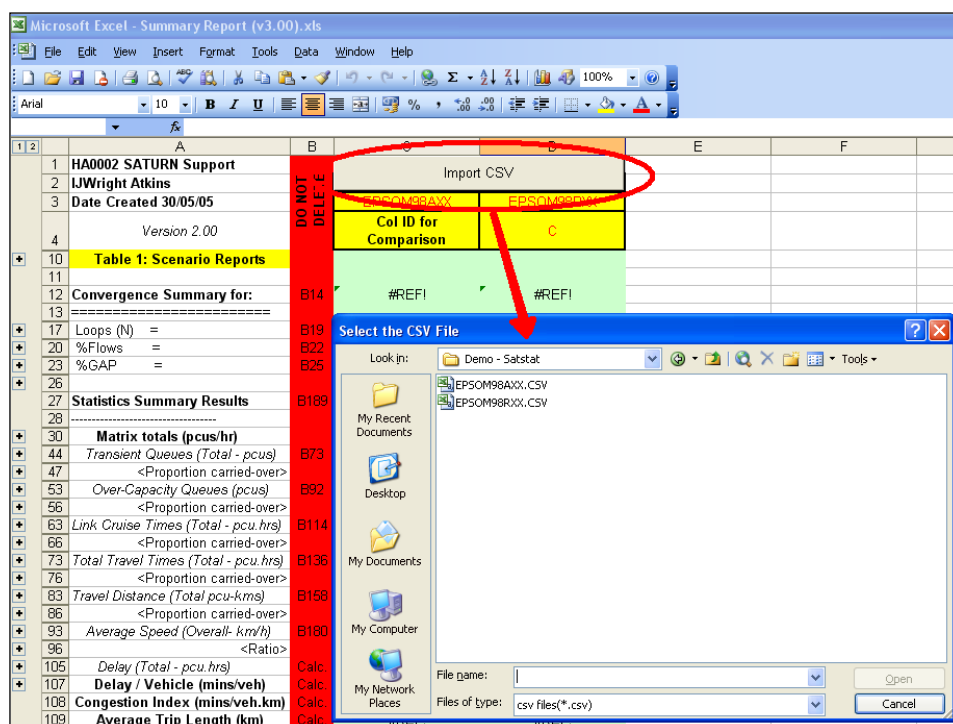
Once the Summary CSV files have been produced, we may import them into the SATSTAT spreadsheet, the latest version of which (at the time of writing) is called "*Summary Report Excel2007 (v4.10).xls*". The process has however not changed since the following example (which refers to "*Summary Report (v3.00).xls*") as shown below was created. The spreadsheet consists of a number of worksheets:

- ◆ A Version Control worksheet – for reference only
- ◆ A Summary worksheet – this is the main report; and
- ◆ A number of imported CSV summary files.

Microsoft Excel - Summary Report (v3.00).xls					
File Edit View Insert Format Tools Data Window Help					
Arial 10 B I U % , .00 .00 100%					
E11					
		A	B	C	D
1	2	HA0002 SATURN Support	DO NOT DELETE	Import CSV	
2		IJWright Atkins		EPSOM98AXX	EPSOM98RXX
3		Date Created 30/05/05		Col ID for Comparison	C
4		Version 2.00			
10		Table 1: Scenario Reports			
11		Convergence Summary for:	B14	EPSOM98AXX.CON	EPSOM98RXX.CON
12		=====			
13					
17		Loops (N) =	B19	12	11
20		%Flows =	B22	94.5	98.1
23		%GAP =	B25	0.302	0.333
26					
27		Statistics Summary Results	B189	EPSOM98AXX.STA	EPSOM98RXX.STA
28		=====			
30		Matrix totals (pcus/hr)		5013.1	5170
44		Transient Queues (Total - pcus)	B73	39.5	196.3
47		<Proportion carried-over>		0.5%	3.0%
53		Over-Capacity Queues (pcus)	B92	1.4	89.5
56		<Proportion carried-over>		0.0%	8.6%
63		Link Cruise Times (Total - pcu.hrs)	B114	117.5	217.7
66		<Proportion carried-over>		0.1%	1.4%
73		Total Travel Times (Total - pcu.hrs)	B136	158.4	503.6
76		<Proportion carried-over>		0.2%	3.3%
83		Travel Distance (Total pcu-kms)	B158	5230.1	8077.7
86		<Proportion carried-over>		0.1%	1.3%
93		Average Speed (Overall- km/h)	B180	33	16
96		<Ratio>		32.4%	41.3%
105		Delay (Total - pcu.hrs)	Calc.	40.9	285.9
107		Delay / Vehicle (mins/veh)	Calc.	0.49	3.32
108		Congestion Index (mins/veh.km)	Calc.	1.82	2.61
109		Average Trip Length (km)	Calc.	1.04	1.56
111		Simulation Queues (pcu/h)			
114		Ave. Queue (DA1433) =	B373	39.36	158.16
115		Queue at End (DA1483) =	B374	5.48	163.7
122		Turn Penalties (pcu-hrs/hr) =	B360	not found	not found
129		Turn Penalties (number) =	B367	not found	not found

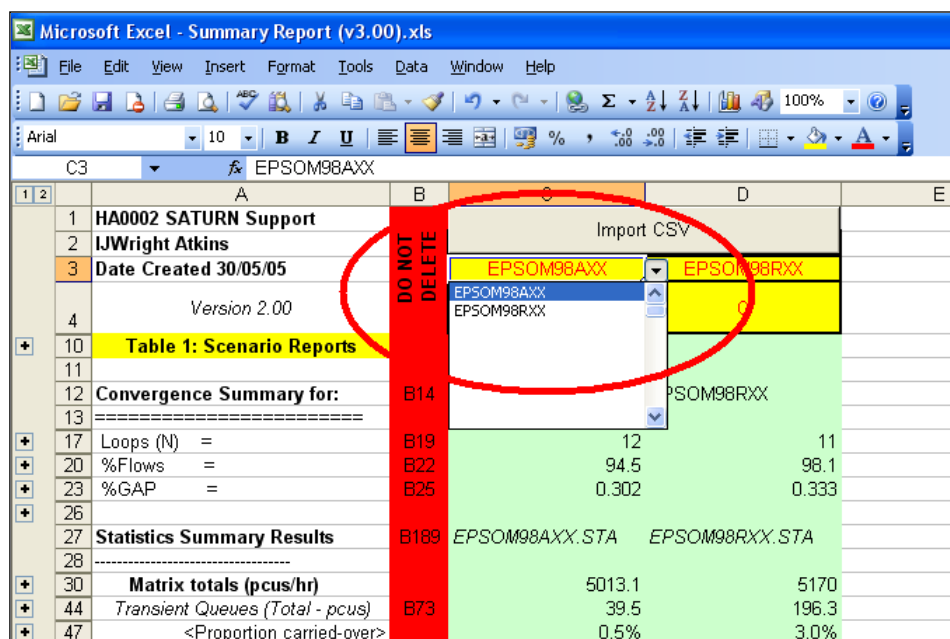
15.49.2.3 Importing CSV Files

- ◆ To import our new Summary Spreadsheet files, we select the *Summary* worksheet and press the *Import CSV* button to bring up the Windows File Open dialogue box:



We may then select each CSV file, in turn, and these will be imported as *new* worksheets into the existing spreadsheet.

Once they have been loaded as new worksheets, we may now select them to be loaded into the reporting columns in the summary worksheet by clicking on the *Filter* box at the top of each column. The summary statistics for that network will then be summarised in the column below.



Once completed, Table 1 'Scenario Reports' shows the summary statistics for two networks side-by-side, as show below.

Microsoft Excel - Summary Report (v3.00).xls					
File Edit View Insert Format Tools Data Window Help					
Arial 10 B I U % , .00 .00 100%					
A2		IJWright Atkins			
1	2	A	B	C	D
1		5048181 SATURN Support	DO NOT DELETE	Import CSV	
2		IJWright Atkins			
3		Date Created 15/03/07		EPSOM98AXX	EPSOM98RXX
4		Version 3.00		Col ID for Comparison	C
10		Table 1: Scenario Reports			
11					
12		Convergence Summary for:	B14	EPSOM98AXX	EPSOM98RXX
13		=====			
17		Loops (N) =	B19	12	11
20		%Flows =	B22	94.5	98.1
23		%GAP =	B25	0.302	0.333
26					
27		Statistics Summary Results	B189	EPSOM98AXX.STA	EPSOM98RXX.STA
28		-----			
30		Matrix totals (pcus/hr)		5013.1	5170
44		Transient Queues (Total - pcus)	B73	39.5	196.3
47		<Proportion carried-over>		0.5%	3.0%
53		Over-Capacity Queues (pcus)	B92	1.4	89.5
56		<Proportion carried-over>		0.0%	8.6%
63		Link Cruise Times (Total - pcu.hrs)	B114	117.5	217.7
66		<Proportion carried-over>		0.1%	1.4%
73		Total Travel Times (Total - pcu.hrs)	B136	158.4	503.6
76		<Proportion carried-over>		0.2%	3.3%
83		Travel Distance (Total pcu-kms)	B158	5230.1	8077.7
86		<Proportion carried-over>		0.1%	1.3%
93		Average Speed (Overall- km/h)	B180	33	16
96		<Ratio>		32.4%	41.3%
105		Delay (Total - pcu.hrs)	Calc.	40.9	285.9
107		Delay / Vehicle (mins/veh)	Calc.	0.49	3.32
108		Congestion Index (mins/veh.km)	Calc.	1.82	2.61
109		Average Trip Length (km)	Calc.	1.04	1.56
111		Simulation Queues (pcu/h)			
114		Ave. Queue (DA1433) =	B373	39.36	158.16
115		Queue at End (DA1483) =	B374	5.48	163.7
122		Turn Penalties (pcu-hrs/hr) =	B360	not found	not found
129		Turn Penalties (number) =	B367	not found	not found

15.49.2.4 Summary Reports

At the highest level, the summary reports are available for:

Convergence in the Assignment-Simulation loop

- ◆ Number of loops undertaken;
- ◆ %Flows achieved
- ◆ %GAP achieved

Summary Statistics (post-simulation for TOTAL flows)

- ◆ Matrix totals excluding intra-zonals;

- ◆ Over-capacity queues;
- ◆ Link cruise times;
- ◆ Total travel times
- ◆ Average speed;
- ◆ Total delay;
- ◆ Average delay per vehicle;
- ◆ Average delay per vehicle-kilometre;
- ◆ Average trip length
- ◆ Average simulation queue;
- ◆ Simulation queue at end of modelled time period; and
- ◆ Turn penalties.

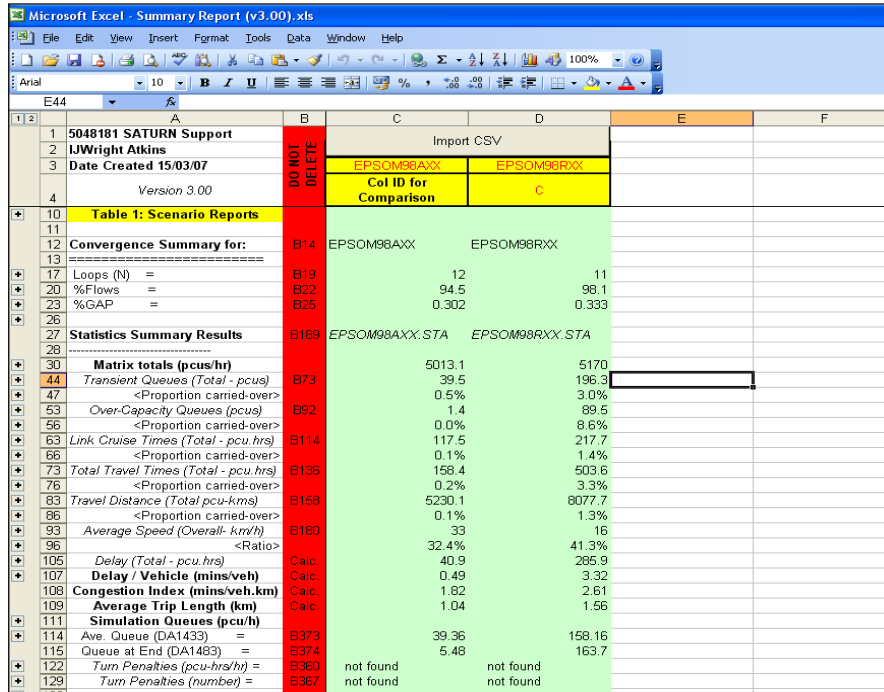
More detailed comparisons are available within the *Summary* worksheet by selecting the second level option to highlight more convergence statistics and, for example, performance for both the modelled hour and the next time period.

A		Congestion Index (mins/veh.km)			
1		Import CSV			
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					
13					
14					
15					
16					
17					
18					
19					
20					
21					
22					
23					
24					
25					
26					
27					
28					
29					
30					
31					
32					
33					
34					
35					
36					
37					
38					
39					
40					
41					
42					
43					
44					
45					
46					
47					
48					
49					
50					
51					
52					
53					
54					
55					
56					

15.49.2.5 Importing Additional Networks

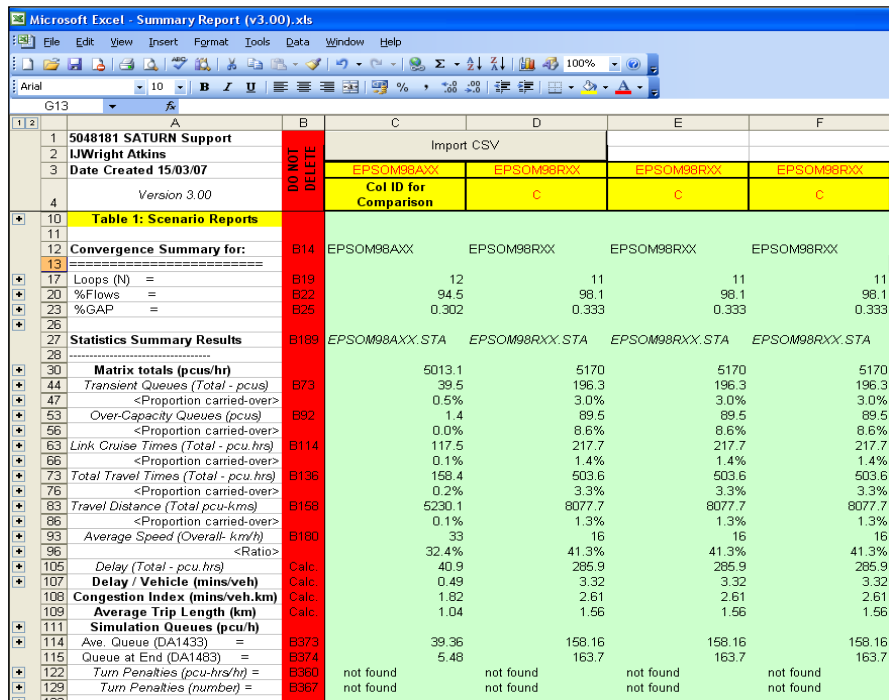
- ◆ This may be repeated for any number of networks. If further reporting columns are required, an existing column may be copied across as shown below.

EXISTING



	A	B	C	D	E	F
1	5048181 SATURN Support	DO NOT DELETE	Import CSV			
2	IJWright Atkins					
3	Date Created 15/03/07		EPSOM98AXX	EPSOM98RXX		
4	Version 3.00		Col ID for Comparison	C		
10	Table 1: Scenario Reports					
11						
12	Convergence Summary for:	B14	EPSOM98AXX	EPSOM98RXX		
13	=====					
17	Loops (N) =	B19	12	11		
20	%Flows =	B22	94.5	98.1		
23	%GAP =	B25	0.302	0.333		
27	Statistics Summary Results	B189	EPSOM98AXX.STA	EPSOM98RXX.STA		
28	=====					
30	Matrix totals (pcus/hr)		5013.1	5170		
44	Transient Queues (Total - pcus)	B73	39.5	196.3		
47	<Proportion carried-over>		0.5%	3.0%		
53	Over-Capacity Queues (pcus)	B92	1.4	89.5		
56	<Proportion carried-over>		0.0%	8.6%		
63	Link Cruise Times (Total - pcu.hrs)	B114	117.5	217.7		
66	<Proportion carried-over>		0.1%	1.4%		
73	Total Travel Times (Total - pcu.hrs)	B136	158.4	503.6		
76	<Proportion carried-over>		0.2%	3.3%		
83	Travel Distance (Total pcu-kms)	B158	5230.1	8077.7		
86	<Proportion carried-over>		0.1%	1.3%		
93	Average Speed (Overall- km/h)	B180	33	16		
96	<Ratio>		32.4%	41.3%		
105	Delay (Total - pcu.hrs)	Calc.	40.9	285.9		
107	Delay / Vehicle (mins/veh)	Calc.	0.49	3.32		
108	Congestion Index (mins/veh.km)	Calc.	1.82	2.61		
109	Average Trip Length (km)	Calc.	1.04	1.56		
111	Simulation Queues (pcu/h)					
114	Ave. Queue (DA1433) =	B373	39.36	158.16		
115	Queue at End (DA1483) =	B374	5.48	163.7		
122	Turn Penalties (pcu-hrs/hr) =	B369	not found	not found		
129	Turn Penalties (number) =	B367	not found	not found		

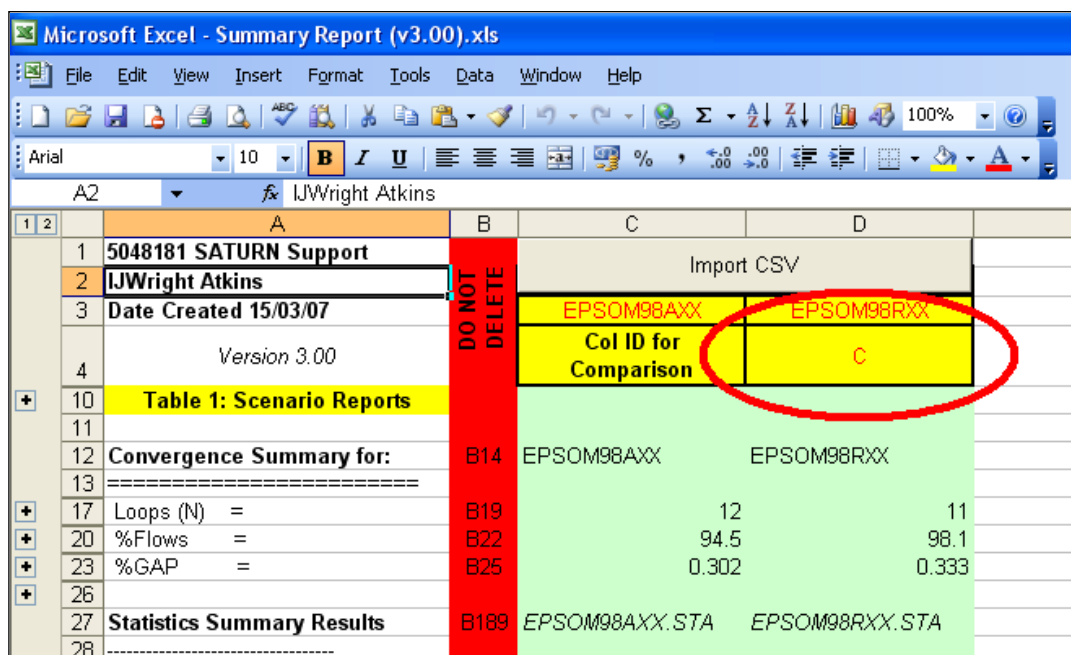
EXTENDED



	A	B	C	D	E	F
1	5048181 SATURN Support	DO NOT DELETE	Import CSV			
2	IJWright Atkins					
3	Date Created 15/03/07		EPSOM98AXX	EPSOM98RXX	EPSOM98RXX	EPSOM98RXX
4	Version 3.00		Col ID for Comparison	C	C	C
10	Table 1: Scenario Reports					
11						
12	Convergence Summary for:	B14	EPSOM98AXX	EPSOM98RXX	EPSOM98RXX	EPSOM98RXX
13	=====					
17	Loops (N) =	B19	12	11	11	11
20	%Flows =	B22	94.5	98.1	98.1	98.1
23	%GAP =	B25	0.302	0.333	0.333	0.333
27	Statistics Summary Results	B189	EPSOM98AXX.STA	EPSOM98RXX.STA	EPSOM98RXX.STA	EPSOM98RXX.STA
28	=====					
30	Matrix totals (pcus/hr)		5013.1	5170	5170	5170
44	Transient Queues (Total - pcus)	B73	39.5	196.3	196.3	196.3
47	<Proportion carried-over>		0.5%	3.0%	3.0%	3.0%
53	Over-Capacity Queues (pcus)	B92	1.4	89.5	89.5	89.5
56	<Proportion carried-over>		0.0%	8.6%	8.6%	8.6%
63	Link Cruise Times (Total - pcu.hrs)	B114	117.5	217.7	217.7	217.7
66	<Proportion carried-over>		0.1%	1.4%	1.4%	1.4%
73	Total Travel Times (Total - pcu.hrs)	B136	158.4	503.6	503.6	503.6
76	<Proportion carried-over>		0.2%	3.3%	3.3%	3.3%
83	Travel Distance (Total pcu-kms)	B158	5230.1	8077.7	8077.7	8077.7
86	<Proportion carried-over>		0.1%	1.3%	1.3%	1.3%
93	Average Speed (Overall- km/h)	B180	33	16	16	16
96	<Ratio>		32.4%	41.3%	41.3%	41.3%
105	Delay (Total - pcu.hrs)	Calc.	40.9	285.9	285.9	285.9
107	Delay / Vehicle (mins/veh)	Calc.	0.49	3.32	3.32	3.32
108	Congestion Index (mins/veh.km)	Calc.	1.82	2.61	2.61	2.61
109	Average Trip Length (km)	Calc.	1.04	1.56	1.56	1.56
111	Simulation Queues (pcu/h)					
114	Ave. Queue (DA1433) =	B373	39.36	158.16	158.16	158.16
115	Queue at End (DA1483) =	B374	5.48	163.7	163.7	163.7
122	Turn Penalties (pcu-hrs/hr) =	B369	not found	not found	not found	not found
129	Turn Penalties (number) =	B367	not found	not found	not found	not found

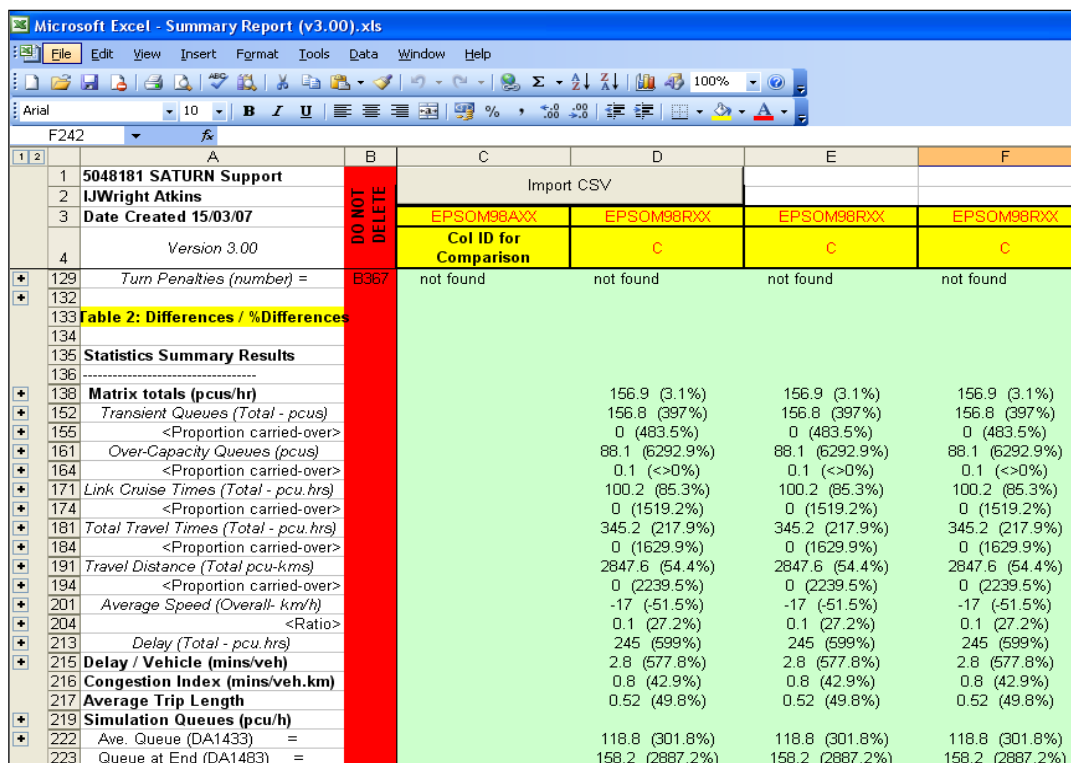
15.49.2.6 Comparing Different Networks

We may also compare the summary statistics for each of the different networks (eg the Development Scenario *EPSOM98RXX* against the Reference Case *EPSOM98AXX*) by specifying the appropriate column ID (ie 'C' in this case).



A		B	C	D
1	5048181 SATURN Support			
2	IJWright Atkins			
3	Date Created 15/03/07			
4	Version 3.00			
10	Table 1: Scenario Reports			
12	Convergence Summary for:	B14	EPSOM98AXX	EPSOM98RXX
17	Loops (N) =	B19	12	11
20	%Flows =	B22	94.5	98.1
23	%GAP =	B25	0.302	0.333
27	Statistics Summary Results	B189	EPSOM98AXX.STA	EPSOM98RXX.STA

Whilst Table 1 will remain unchanged, Table 2 below will now report on the Differences and %Differences in the summary statistics (ie this network **MINUS** the one with the column ID just selected) as shown below.



A		B	C	D	E	F
1	5048181 SATURN Support					
2	IJWright Atkins					
3	Date Created 15/03/07					
4	Version 3.00					
129	Turn Penalties (number) =	B367	not found	not found	not found	not found
133	Table 2: Differences / %Differences					
135	Statistics Summary Results					
138	Matrix totals (pcus/hr)					
152	Transient Queues (Total - pcus)		156.9 (3.1%)	156.9 (3.1%)	156.9 (3.1%)	156.9 (3.1%)
155	<Proportion carried-over>		0 (483.5%)	0 (483.5%)	0 (483.5%)	0 (483.5%)
161	Over-Capacity Queues (pcus)		88.1 (6292.9%)	88.1 (6292.9%)	88.1 (6292.9%)	88.1 (6292.9%)
164	<Proportion carried-over>		0.1 (<0%)	0.1 (<0%)	0.1 (<0%)	0.1 (<0%)
171	Link Cruise Times (Total - pcu.hrs)		100.2 (85.3%)	100.2 (85.3%)	100.2 (85.3%)	100.2 (85.3%)
174	<Proportion carried-over>		0 (1519.2%)	0 (1519.2%)	0 (1519.2%)	0 (1519.2%)
181	Total Travel Times (Total - pcu.hrs)		345.2 (217.9%)	345.2 (217.9%)	345.2 (217.9%)	345.2 (217.9%)
184	<Proportion carried-over>		0 (1629.9%)	0 (1629.9%)	0 (1629.9%)	0 (1629.9%)
191	Travel Distance (Total pcu.kms)		2847.6 (54.4%)	2847.6 (54.4%)	2847.6 (54.4%)	2847.6 (54.4%)
194	<Proportion carried-over>		0 (2239.5%)	0 (2239.5%)	0 (2239.5%)	0 (2239.5%)
201	Average Speed (Overall- km/h)		-17 (-51.5%)	-17 (-51.5%)	-17 (-51.5%)	-17 (-51.5%)
204	<Ratio>		0.1 (27.2%)	0.1 (27.2%)	0.1 (27.2%)	0.1 (27.2%)
213	Delay (Total - pcu.hrs)		245 (599%)	245 (599%)	245 (599%)	245 (599%)
215	Delay / Vehicle (mins/veh)		2.8 (577.8%)	2.8 (577.8%)	2.8 (577.8%)	2.8 (577.8%)
216	Congestion Index (mins/veh.km)		0.8 (42.9%)	0.8 (42.9%)	0.8 (42.9%)	0.8 (42.9%)
217	Average Trip Length		0.52 (49.8%)	0.52 (49.8%)	0.52 (49.8%)	0.52 (49.8%)
219	Simulation Queues (pcu/h)					
222	Ave. Queue (DA1433) =		118.8 (301.8%)	118.8 (301.8%)	118.8 (301.8%)	118.8 (301.8%)
223	Queue at End (DA1483) =		158.2 (2887.2%)	158.2 (2887.2%)	158.2 (2887.2%)	158.2 (2887.2%)

15.49.2.7 SATURN Versions

SATSTAT will work for all versions of SATURN v10 available i.e., v10.1 through to 10.9. It should operate correctly on any UFS files created under these various versions as well as undertaking the summary reporting using any of these versions.

15.50 SATMECC – Marginal Economic Consumer Costs

15.50.1 Basic Theory

The principle of “marginal cost” was introduced in Section 7.11.9 where equation (7.46) defined the marginal cost for a “separable” cost-flow curve, i.e., one where the cost of travel on link *a* is a function only of the flow on link *a*, as:

Equation 15.16

$$\tilde{c}_a(V_a) = c_a(V_a) + V_a \frac{\partial c_a}{\partial V_a}$$

In this section we generalise the concept to allow for “interactions” between different streams of traffic and therefore “non-separable” cost-flow functions as modelled by the simulation stage within **SATURN**. We use the acronym MECC to stand for Marginal External Cost of Congestion.

However, the basic underlying concept of marginal cost is unchanged: it is the extra cost imposed on **all** trips by the addition of one extra **pcu** (N.B. pcu, **not** vehicle) on a particular “link” (where a link may be either a road or a simulated turn). With separable costs the only other vehicles which are affected by an extra vehicle on link *a* are those vehicles already on link *a*; with non-separable costs the affected vehicles may be on other links. Unfortunately, since **SATURN** does not generate explicit non-separable cost-flow curves of the form $c_a(\mathbf{V})$ where \mathbf{V} represents the complete vector of all link flows, we must resort to simulation.

We recall that non-separable or interaction costs only arise from turning movements at simulation nodes. For buffer links and “pure” simulation links there are no direct interactions and equations such as (7.46) may still be applied.

The basic method used to calculate marginal costs for simulation turns is to add one pcu per turn, re-simulate that individual node and to calculate the changed costs on all turns and/or links at that simulation node. It is carried out by a procedure (i.e., .bat file) known as **SATMECC** which, in fact, makes use of special procedures within **SATLOOK**. The procedure outputs an ascii file (details below, 15.50.8) with an extension .mec.

SATMECC was first introduced in 10.8 in 2007 and was developed with the financial support, advice and technical co-operation of GMPTE (Greater Manchester Public Transport Executive) and GMTU (Greater Manchester Transportation Unit) whose inputs are gratefully acknowledged.

15.50.2 Marginal Cost vrs Marginal Time

Generalised **cost** is normally a weighted sum of time, distance and other components such as tolls (see 7.11.1) but, of these, only time is directly affected by flows; adding an extra pcu has no impact on link distance, for example.

Therefore marginal cost is effectively equivalent to marginal time with a “value of time” factor (i.e., PPM) to convert marginal time into marginal cost in exactly the same way that time is converted to cost.

We note briefly at this point, and in more detail later (15.50.5), that the value of time may differ between different user classes and that we may distinguish marginal cost by user class.

Within **SATMECC** outputs are always expressed in terms of marginal **time** (in units of seconds/pcu) and it is up to the user to convert to marginal cost if and when desired. (Indeed it would probably be more accurate to refer to MECT rather than MECC but we retain the more standard convention.)

15.50.3 Marginal Cost Calculations: Incremental Simulation

MECC values per simulation turns are estimated by (a) carrying out a full simulation of the “base” to obtain both base delays and base flows per turn and (b) repeating a full simulation of the node with an additional small increment of flow ΔV (e.g., 1 pcu) added to the turn in question. (But see 15.50.6 for alternative procedures in selected circumstances.)

The total value of MECC may be calculated as:

Equation 15.17

$$MECC_a = \sum_i V_i (d_i(l) - d_i(B)) / \Delta V_a$$

Where i represents a particular turning movement (link) at that junction (including a)

V_i is the (total) **demand** flow for that turn

ΔV_a is the increment of traffic to the current turn a (either positive or negative)

$d_i(l)$ is the simulated delay with the increment ΔV_a

$d_i(B)$ is the delay in the “base” simulation

Notes:

- 1) This definition **excludes** the current cost of link a , i.e., the first term on the right-hand-side of 7.46. However it is not difficult to add this contribution later on as required.
- 2) Strictly speaking Equation 15.16 defines marginal **time**, not cost since we use “unweighted” delays in units of seconds.
- 3) This method can give both positive and negative values of MECC whereas the use of equations with separable cost-flow curves can only yield non-negative values. Negative MECC values may seem counter-intuitive but in fact they occur quite naturally as a result of normal give-way conventions. For example, consider a roundabout with 4 arms (north, south, east and west) with a very heavy flow from east to west which effectively blocks all entry traffic from the south. In these circumstances the only way traffic from the south can enter is when north-south traffic cuts off entry from the east. Hence

an increase in N-S flow can **reduce** the delays from the south leading to a negative contribution to MECC which, in turn, can potentially drive the total MECC negative as well.

15.50.4 Disaggregated Marginal Costs by Turn

The impact of adding an extra pcu on a particular simulation turn may be disaggregated into increased delays to:

- (a) pcus making that particular turn,
- (b) other turning movements from the same arm and
- (c) turns out of other arms at the same junction.

MECC is the sum of all three impacts.

We refer to these three disaggregate contributions as “own-MECC”, “arm-MECC” and “interactive-MECC”.

15.50.5 Disaggregated Marginal Costs by User Class, Vehicle Class, etc.

If we wish to consider the marginal impact of increased flows on a particular link on, e.g., one particular class of flows – as opposed to the **total** flow on links as implied by the definition of V_i in equation (15.16) – it is simply a question of re-defining V_i as the flow for that user class. Equally we could define marginal cost for a particular class of flows such as buses.

Note that user class is defined here in terms of the class of vehicles **affected** as opposed to the class of vehicles which is causing the changes. In particular, if we increase the flow on link a by 1 pcu it does not matter **which** class of vehicles is being increased since, say, 1 pcu of user class 1 has the same effect within the simulation as 1 pcu of user class 2.

Note that in this context it is very important to distinguish between marginal **cost** and marginal **time** since the conversion between the two will depend on the value of time defined for that user class. If – as is most likely the case – users require the total marginal cost in terms of, say, pence as opposed to total marginal time then it is necessary to calculate marginal time for each individual user class, weight that by the appropriate value of time for that class and then sum over all user classes. It is not really possible to define an “average” value of time since the distribution of flows by user class will vary by turn.

By a similar token please note that MECC is always calculated **per pcu** and that different user classes may also have different values of pcu/vehicle. Again it is up to the user to take account of these factors in terms of translating SATMECC outputs into actual toll per vehicle.

15.50.6 Alternative Modifications to Incremental Simulation

There are a variety of circumstances under which the simple “add a pcu” simulation method may give unreliable results (where “unreliable” generally means extremely high absolute values). Therefore a number of “alternatives” have been included within **SATMECC** as follows:

- 1) If the turn is over capacity in the base simulation we do not have to perform a second simulation to calculate MECC since the main impact of an extra pcu on an over-capacity link is essentially to make the queue on that arm one pcu longer rather than changing the flows through the node. Thus we may combine equations (7.46) and (8.5b) (or (8.11b) in the case of shared lanes) to obtain:

Equation 15.18

$$MECC = (LTP/2)V/C$$

- 2) If the turn is “almost at capacity” (strictly $90\% < V/C < 100\%$ then a **negative** increment of 1 pcu is used in the first instance rather than an increase of 1.0. (But see point (7) below.)
- 3) If the turn has zero or very low flow (arbitrarily under 5 pcu/hr) the turn is simulated at flows of, say, 5.0 and 6.0 pcus/hr as opposed to, say, 0.0 and 1.0 pcus/hr since, very often, there can be very highly significant changes between no flow and a very small flow. This is particularly true of X-turns at signals, even more so when they come from a single lane with other shared movements.
- 4) If the addition of +1 pcu takes a turn beyond capacity then the increment is reduced so as to go only “halfway” to capacity.
- 5) In order to minimise any problems of “noise” in the two simulations (if we are looking at two very similar flows any “errors” in delay calculations will be magnified) we convert the value of NUC applied at signalised junctions to a large value. In particular this has proved to be essential for junctions with very short signal phases and/or X-turns.
- 6) If, for whatever reason, a node does **not** converge to the required limits (see 8.3.2) even with any changes to NUC, etc. its convergence is judged to be “poor” and, rather than permit any noise to creep into the calculations, its MECC values are calculated using its separable cost-flow curve and equation (15.15) above. This is probably an underestimate of the total MECC since it exclude any interactions with other turns at that junction but this is felt to be preferable to introducing random errors. In most networks the number of “poorly” converged nodes is probably well under 1%. (A higher percentage of poorly converged nodes may be an indication of slightly dodgy coding practice.)
- 7) As an example of belt’n’braces and to cope with various “noise” problems which empirically are observed to occur even with all the above rules, for priority and signalised junctions, whenever both positive and negative increments are feasible, we carry out **two** increments, both plus and minus 1 pcu, and take the **minimum** of the two MECC values. (In almost all cases the two give virtually equal answers but there are odd examples when one result appears to be unreliable and we prefer to believe the lower values.)
- 8) Turns at simulation dummy nodes experience zero delay by definition and therefore zero marginal costs. They are included in the output .mec files with values of zero. Similarly bus-only links and/or turns are assigned zero MECC values.

15.50.7 Marginal Costs on Links

In addition to calculating marginal costs on simulation turns **SATMECC** also calculates marginal costs for “pure” links (i.e., roads A-B as opposed to turns A-B-C) which are (a) in the buffer network or (b) simulation links with capacity-restraint speed-flow curves. In both cases it is only necessary to use equations (7.47a) and/or (7.47b). Note that simulation links which do **not** have speed-flow curves are excluded.

Links are included within the output .mec file (see 15.50.7) and are identified by a value of 0 in the third node field; i.e, A B 0 as opposed to A B C for simulation turns.

Simulation links without capacity-restraint speed-flow curves are totally excluded as are all centroid connectors.

15.50.8 The SATMECC Batch Control File

A special batch file SATMECC.bat has been set up in order to carry out the calculations detailed above making use of the program **SATLOOK**. Its specification is as follows:

Call: SATMECC network (UC n KR control
 Files: network.ufs Input network file
 Network.mec Output ascii file of MECC values
 Network.LPL Output line printer file from **SATLOOK**
 Control.dat Control file (Optional)

UC n (optional) requests that the calculations be carried out for user class n and the output file will be network.ucn.mec. UC * requests a loop over **all** user classes to produce files network.uc1.mec, network.uc2.mec, etc. etc.

Output .mec files contain 8 “fields” formatted as follows:

Field 1	A-node
Field 2	B-node
Field 3	C-node (for a turn; 0 for a link whether simulation or buffer)
Field 4	The “base” delay to turn A-B-C in seconds
Field 5	The total MECC in seconds (<u>N.B.</u> marginal time , not cost)
Field 6	“Own” MECC
Field 7	“Arm” MECC
Field 8	“Interactive” MECC

For “pure” links A-B (i.e., simulation or buffer links) fields 7 and 8 are blank.

Note that Fields 1, 2 and 3 all occupy 6 columns in order to improve legibility for networks which have up to 5-digit simulation node numbers. This may cause



problems in input to certain **SATURN** programs where the convention is to have 5 columns by default. Note that the latest 10.8 version of **SATDB** allows either 5 or 6 column node fields under the input of miscellaneous text files.

If DUTCH = T **and** the maximum node number used in the buffer network exceeds 5 digits then the first two fields occupy 9 columns each and the third (which is zero by definition) occupies 2 columns (i.e., 0 in column 20).

In general MECC values are printed with two decimal places (with units of seconds) although, in some extreme cases, there may not be sufficient "width" in the format to permit two decimal places, in which case MECC is printed in an E-format.

15.51 Running SATURN within DIADEM

The DIADEM suite of programs has been created by Mott-MacDonald under contract to the DfT to provide demand matrix calculations linked to various traffic assignment programs. In particular Diadem has been linked with **SATURN** such that Diadem calculates vehicle trip matrices which **SATURN** may then assign. See Section 7.4.5 for a discussion of the general VDM principles involved and for suggestions as to how to make the overall process more efficient.

Full documentation on Diadem and its linkages with **SATURN** are provided by the Diadem documentation.

Generally the procedures for running **SATURN** programs within Diadem are controlled by Diadem itself. However, there are various options within **SATURN** programs which may assist in achieving a well-converged solution with minimal cpu and which either users may set themselves in their network .dat files and/or Diadem developers may incorporate within the internal control procedures.

It is highly recommended that Diadem users ensure that they are running the most up to date releases of **SATURN** since some of the features listed below are fairly recent additions.

QUIET – This option enables **SATURN** programs such as **SATALL** to run totally in "the background" without interrupting anything else. See 14.9

NDPS – This controls the number of decimal places used to output skimmed matrices in TUBA Format 2. Large values, e.g., 4, are recommended to avoid convergence problems due to rounding off but, in older releases of **SATURN**, this could cause problems of "overflow" if a numerical skim value required more than 10 columns including decimal places. Corrected in 10.7. The current default number of decimal places is 5. See 10.15.2 and 15.41.4.

DIADEM parameter. Setting DIADEM = T under &OPTION (N.B. **not** &PARAM) in a network .dat file at the same time that UPDATE and/or WSTART are also T means that, if the file to be updated as set in UPFILE does **not** exist, UPDATE and/or WSTART are ignored. Normally a missing UPFILE is a semi-fatal error. In the context of DIADEM this allows the same network .dat file to be used to build a network for both the initial assignment and for later assignments where the UPDATE/WSTART options may be invoked to update/warm start the previous network. See 6.1.

XCL ON COMMAND LINES This feature allows more than 9 arguments per command line. In the particular context of DIADEM this **increases** the number of matrices which may be stacked and unstacked but the number is still potentially limited. See 14.8.

UFMSTACK AND UFMUNSTACK These new bat files allow matrices to be stacked and/or unstacked with, effectively, no limit on the number of levels / user classes which may be accommodated. See 10.20.17 and 10.20.18.

SAVEIT – Unless you specifically need to create skimmed matrices of time, distance, etc. in order to run the demand model within DIADEM (which, in general terms, we do not recommend; see 7.4.5.3 and 7.8.6) **or** you are using a warm start we recommend that you set **SAVEIT = F** and avoid the excess CPU of creating .UFC files which will not be used at the end of each run of **SATALL**. Note that, if required, a .UFC file can be created at the very end by running the procedure SATUFC (see 15.23.5). (N.B. This does not apply under OBA where the overheads associated with **SAVEIT = T** are minimal.)

SKIM_ALL – If, however, the Diadem model requires skimmed matrices of time, distance and/or tolls on each supply-demand loop it will save CPU time to use the simultaneous skimming procedures embedded in **SKIM_ALL** (see 15.27.7) rather than skimming each component separately.

Further general advice on linking **SATURN** with external variable demand models such as Diadem as given in Section 7.4.5.

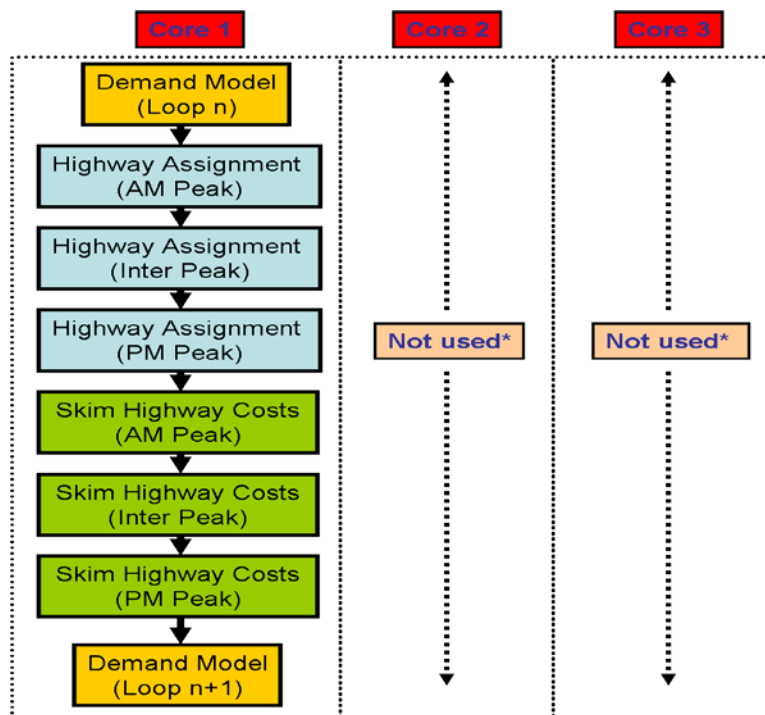
Users may also wish to note that the “%Gap” convergence measure used within Diadem has an equivalent measure within **SATEASY**, i.e., TxCij-AAD as referred to in Section 7.5.5.

15.52 Running SATURN in Parallel

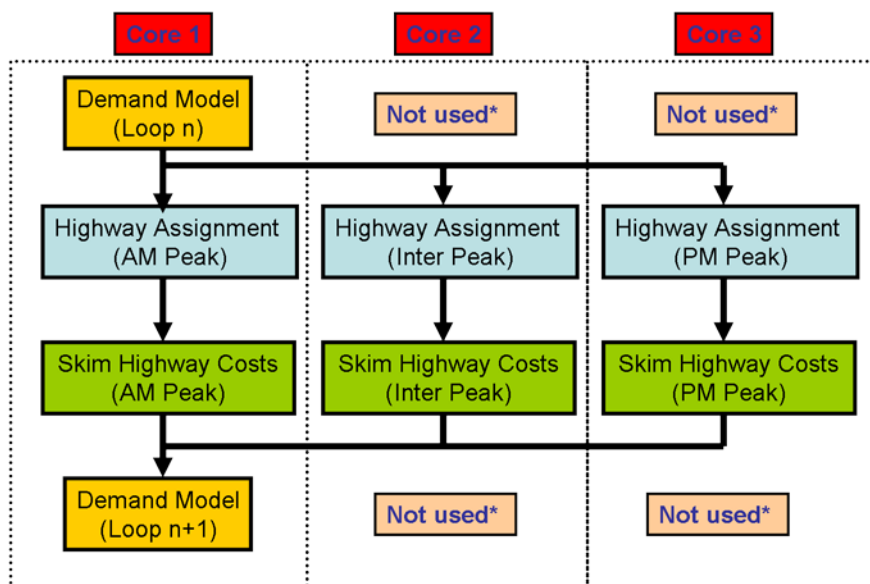
From **SATURN** v10.7 onwards, a new feature was introduced that enabled users to take advantage of desktop PCs with more than one core and/or processor. This was intended as stop-gap measure until development work on modifying the **SATURN** source code to access more than one core was completed during early 2009. Subsequent testing has demonstrated that the process is also compatible SATURN Multi-Core and enables the maximum use of all the cores available.

The software was developed in response to a need to reduce runtimes for demand models where there is a requirement to undertake independent highway assignments by time period (e.g., separate morning peak hour, inter-peak hour and evening peak hour models) and to subsequent skim various permutations of travel costs and/or demand for use in WebTAG-based compliant models.

The recent advances in PC-based desktop hardware has resulted in Intel-based Core2Duo/ Nahlem / Sandy Bridge chips becoming the norm (amongst others) and the existing methods of using **SATURN** to undertake tasks sequentially does not utilise any of the other cores available as illustrated below in Figure 15.6 and Figure 15.7..

Figure 15.6 – Traditional Sequential Operation

* Not used by SATURN

Figure 15.7 – Advantages of Multi-Core Processors

* Not used by SATURN

15.52.1 Additional Programs

Parallel runs can be performed with the aid of two programs: MONITOR and WAIT

15.52.1.1 Monitor

MONITOR takes a snapshot of all running processes in Windows at regular (user specified) intervals and checks if a user specified program is still running. The program terminates when the user specified program is not one of the currently running processes. MONITOR takes two parameters:

- ◆ *Parameter 1* – the **SATURN** program to run in parallel (eg \$SATALL.EXE).
- ◆ *Parameter 2* – the time interval (in seconds) for taking snapshots of running processes to determine if the **SATURN** program specified by *parameter 1* is still running (eg 10).

15.52.1.2 Wait

WAIT (used within a batch file) causes the batch file to pause (where it is called) for a user specified number of seconds before the batch file proceeds to the next command. This introduces a short pause before the next instance of the **SATURN** program that is to run in parallel is called. In other words, if the user wishes to run two instance of \$SATALL, the user should request a small pause between the first and second runs commencing to prevent potential file access errors arising. Note that WAIT has to be used within a batch file – it will not ‘wait’ as a single command line call.

15.52.2 An Example

An example of a batch file to undertake a ‘parallel’ run of the **SATURN** module is annotated below (but the process would also readily work with other **SATURN** modules including \$SATALL, \$SATPIJA, \$SATLOOK, \$SATME2, and \$SATEASY for example).

The batch file should contain the following commands:

```
PATH = C:\SATWIN\XEXES
```

- 1) sets the path to the folder containing SATURN programs (if required)..

```
SET INPUT1=EPSOM5000
```

- 2) sets the input parameter for the first **SATURN** run as an environment variable. In this example, the network and matrix file have the same root name (i.e. EPSOM5000.UFN and EPSOM5000.UFM), hence only one environment variable (i.e. INPUT1) is required. If the root names were different, two environment variables would have been required: one for the UFN file and one for the UFM file.

```
SET INPUT2=EPSOM5001
```

- 3) sets the input parameter for the second **SATURN** run as an environment variable. The root name of the network and matrix file is the same in this example; hence, one environment variable is required as in 2.

```
START SATURN %INPUT1% %INPUT1%
```

- 4) starts the first SATURN run.

```
WAIT 5
```



- 5) Waits for 5 second to avoid simultaneous access to SAT10KEY.DAT

```
START SATURN %INPUT2% %INPUT2%
```

- 6) starts the second SATURN run

```
START /W MONITOR $SATALL 10
```

- 7) starts monitoring \$SATALL.EXE every 10 seconds and **MONITOR** terminates if \$SATALL.EXE is not found in the latest snapshot of running processes controlled by the Windows Operating System.

- 8) **SATURN MULTI-CORE** applications are multi-threaded versions of existing programs that are able to take advantage of the additional processors (either in the form of physical cores or virtual threads) available on most Intel / AMD-powered standard desktop PCs.

The 'start' command will open a new command shell to run SATURN every time it runs. If the user wishes each command shell to be closed at the end of the operation, create a copy of the existing SATURN.BAT and add an extra line at the end saying "EXIT". So for example, if the amended batch file was called "SATURNEXIT.BAT" the revised command line would be:

```
START SATURNEXIT %INPUT2% %INPUT2%
```

15.53 SATURN Multi-Core Applications

SATURN MULTI-CORE applications are multi-threaded versions of existing programs that are able to take advantage of the additional processors (either in the form of physical cores or virtual threads) available on most Intel / AMD-powered standard desktop PCs.

SATURN MULTI-CORE is a **separate, low-cost add-on** to the standard suite and may be accessed through an updated set of executables (and system files). The control of the multi-threaded processors is automatically undertaken by the program and the Operating System.

Multi-processor applications may be sub-divided into two categories:

- those programs that allocate calculations to separate threads internally within the processor(s) and therefore need to be linked with certain routines compiled using IVF as opposed to Salford Fortran, and
- those where the allocation to separate threads is handled at the level of the batch file but the same basic program exe is used for each thread ("**distributed processing**" as previously described in Section 15.52).

Applications under (a) require distinct EXE files, those under (b) require special .bat files identified with the '_MC' suffix.

In principle method a) should be faster and more efficient than method b) but, on the other hand, it requires specially compiled versions of the .exe files whereas method b) uses the standard exe's but with "clever" batch files. Method a) works by allocating tree-building etc. operations by origin between processors, method b) works at a much more aggregate level by allocating, say, calculations per user

class to separate threads (and therefore is most efficient if the number of threads available equals the number of user classes).

15.53.1 Programs Available

SATALL was the first program to be modified to function with multiple parallel processors and was first released with version 10.8.22. Since then, further development work has been undertaken to expand the number of programs available with multi-threaded capability as detailed below. The development work was completed with the release of 11.2.05 in March 2013.

Program	Status	How to Access?	Version	Comment
SATALL	Final Release	Replacement \$SATALL.EXE and set MULTIC=T	v10.8.22 onwards	The assignment routines are multi-threaded internally whereas, the simulation remains unchanged
SATLOOK	Final Release	Replacement \$SATLOOK.EXE and set MULTIC=T	v10.9.22 onwards	Various skimming options may be run using multiple threads in parallel. See 15.53.3.2.
SATUFO	Beta Release	Replacement \$SATUFO.EXE and set MULTIC=T (and/or embedded within \$SATALL.EXE if SAVUFO=T	v11.1.02 onwards	Generation of .UFO from existing .UFC undertaken using multiple threads in parallel. Replaces previous distributed SATUFO_MC process.
SATPIJA_MC	Final Release	New SATPIJA_MC.BAT and \$SATPIJA_MC.EXE	v10.9.22 onwards	Undertaken using a distributed version whereby the PIJA process is split by "blocks" of origins and multiple versions of SATPIJA run for each. A final SATPIJA run combines the individual datasets into a single file. The management of the process is undertaken automatically by the software. See 15.53.3.3.
SATCH_MC	Final Release	New SATCH_MC.BAT and \$SATCH_MC.EXE	v10.9.24 onwards	Multi-distributed as per SATPIJA with each user class undertaken on a separate thread; See 15.53.3.6
(Secondary Analysis)		(See SATUFO above)		(Undertaken using UFO files)

15.53.1.1 SATALL Multi-Core Restrictions

We note that the multi-core facilities within **SATALL** do not (yet) work with every possible combination of assignment options. Thus it does not work with any form of *elastic assignment*, with *stochastic assignment* (SUZIE = T) or with networks which incorporate *Motorway Weaving Segments* (Section 15.40). If

there is sufficient demand to include such options it will be considered for future inclusion.

SATALL also does not work with either *path-based* or *origin-based (OBA) assignment* because the theoretical principles of these algorithms require them to be undertaken in a purely sequential process.

In addition multi-core requires that there is sufficient RAM provided to store the full trip matrix in core; with certain matrices it may therefore be necessary to set a control parameter `SPARSE = F` to select a more efficient system for matrices where more than 50% of the T_{ij} cell values are positive. See 7.11.12.

15.53.1.2 Numerical Differences between Multi-Core and Standard Programs

The development work required the computational intensive sub-routines to be modified so that they may be undertaken in parallel. The work also required the modified source code to be created separately using a different software compiler than the standard release (specifically Intel Virtual Fortran rather than Salford).

An unavoidable consequence of using a different compiler is the introduction of very small differences in the numerical precision that the internal calculations within the assignment are stored in their respective versions. The differences should, generally, be too small to spot but there may be some cases where, like the analogy of a butterfly flapping its wings, the two may give detectable, even significant, differences although each will be perfectly valid solutions. **Consequently, the results from the SATALL MC executable may be different from the standard version.**

However, the other executables that undertake the secondary analysis, should not be affected as they are either (i) only re-building the existing stored paths rather than undertaking new assignments; (ii) undertaking the analysis for each user class in parallel using the same process.

15.53.2 Processors, Cores and Threads

Processors (or 'Central Processor Units' (CPUs) to give them their full name) provide the computing power for the Personal Computer (PC). The processor undertakes the tasks as specified by the Operating System (usually a version of Windows) and the software programs using the Operating System. The processor may have a single or multiple cores with each core capable of running independently to provide additional computing power. Most Desktop PCs will have at least two cores but four is becoming more common with higher-end systems having six or more cores.

Cores and threads are often used interchangeably even though they are fundamentally different. This has implications for the performance gains available from multi-threaded applications.

Cores are physical hardware blocks in the central processor unit (CPU) that can run applications serially whereas threads aren't physical but are software-generated tasks that can be undertaken independently. The computing power of each core is a fixed quantity available for use. In day-to-day applications, not all of the processing power available may be fully used if a software-generated thread is paused or stopped whilst waiting for data so some of the processing

power may be unused and 'wasted'. However, running two threads on the same core enables the second thread to take advantage of this 'spare' processing power whilst the first thread was waiting for data. Whilst running two threads on a single core reduces wastage it is not a substitute for having additional physical cores instead

There are various different propriety names for this technology - Intel's Hyper Threading (HT) technology, available on most of their medium and high-end processors, is probably the most widely known. Intel HT uses this principle whereby each physical core is able to run two threads simultaneously.

As far as **SATURN** Multi-core applications are concerned, its applications will automatically generate N threads (either up to the maximum available as defined by the operating system, the user-defined MCNUM parameter or the maximum number of threads that the application may be broken down into as defined in the batch files) so that tasks may be undertaken simultaneously. The Windows Operating System takes the threads generated by the **SATURN** application(s) and schedules them to run on the threads available. No further user intervention is required.

15.53.3 Performance Gains

The performance gains available are dependent on a large number of variables namely:

- ◆ PC hardware including the processor, operating system and RAM available; and
- ◆ Model size and configuration particularly with the number of zones and user classes.

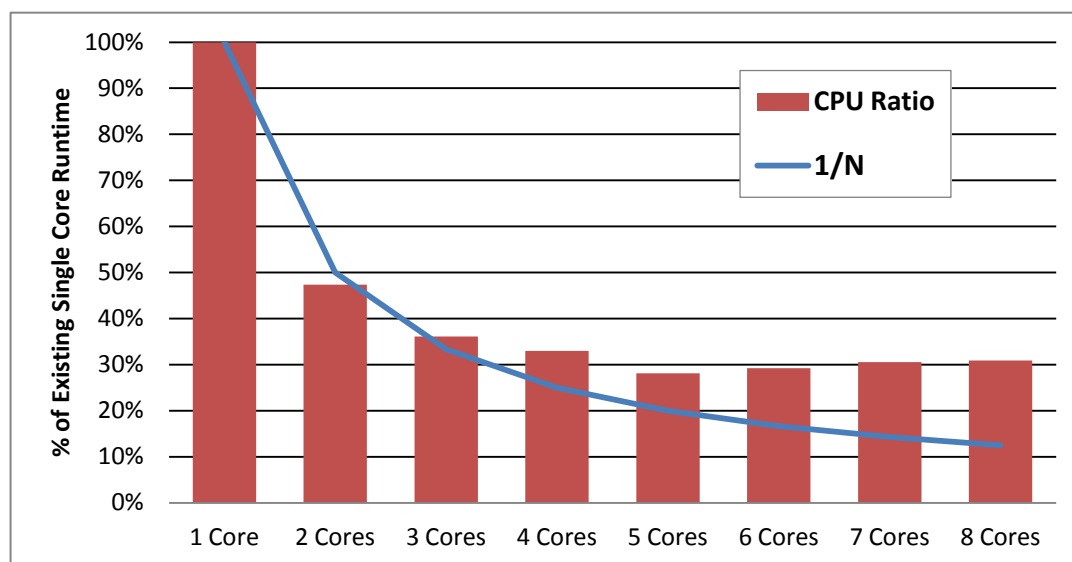
The performance testing across a range of different sized **SATURN** models demonstrated the significant reductions in model runtimes available with **SATURN** Multi-Core. In the following paragraphs, examples are provided for a medium sized model on a high performance desktop PC.

Typically, the multi-threaded applications reduced the overall model runtimes by up to $1 / N$ where N was the number of physical cores available (depending on the size and type of network and the assignment parameters used). For example, on a quad-core machine, the model runtimes on various test networks were reduced by up to a factor of four.

Note that all the tests were undertaken on the same HP XW8600 workstation (2 x Intel Xeon X5450 3GHz with 4Gb RAM running Windows XP 32-bit). The processors provided eight physical cores with each core able to handle one thread each (i.e. no Hyper-threading option was available on these particular processors).

15.53.3.1 SATALL (Multi-threaded)

The performance gains available with the multi-threaded version of **SATALL** are shown below in Figure 15.8. The overall reduction in the total CPU time for **SATALL** was reduced by up to 3.5 times on a Quad-core PC. With an extra fifth core available, further reductions in model runtimes were achieved but with six or more cores, the model runtimes marginally increased in this example.

Figure 15.8 – Example of SATALL Performance (Medium Size Network)

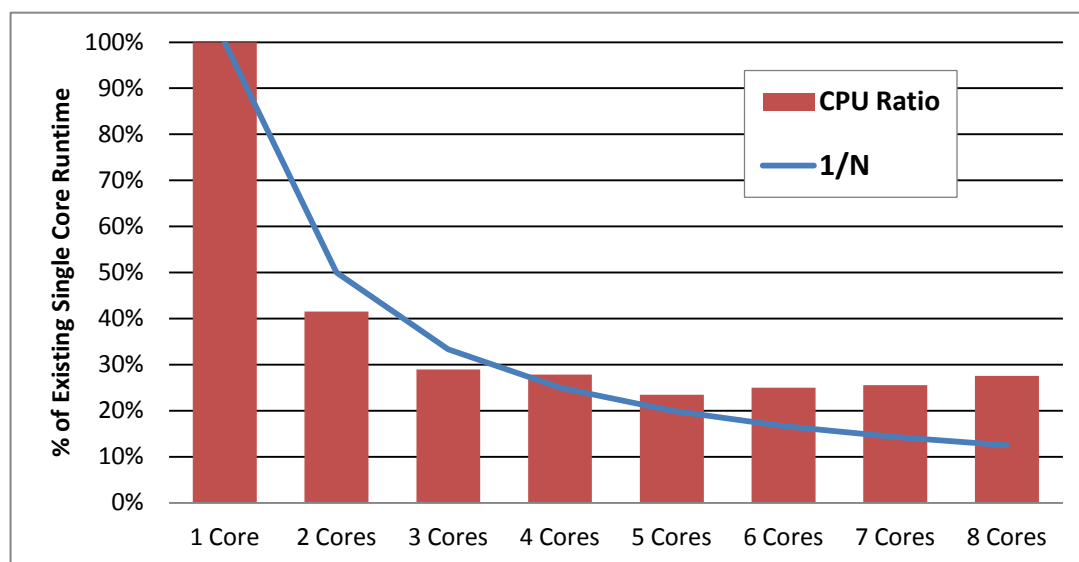
The assignment undertaken using **SATALL** involves an iterative looping process between successive assignment (for tree-building and loading) and simulation (for junction interactions). However, only the main assignment routines are undertaken in parallel and therefore the benefits of **SATALL** Multi-core are dependent on the time taken within the assignment and simulation routines. Similar results were found in other models but the performance gains will be dependent on a large number of variables including the PC hardware available and the SATURN model used.

SATALL Multi-Core is also compatible with Network Aggregation techniques (see Section 15.56) and the performance gains are independent (and hence, typically, multiplicative). Further information may be found in Appendix S.

15.53.3.2 SATLOOK Skims (Multi-threaded)

At present multi-threaded versions of **SATLOOK** may only be run within a limited number of applications / batch files which skim costs; specifically: SKIMTIME, SKIMDIST, SKIMPEN, SKIMTOLL, SKIM_ALL (see 15.27.7) and SATTUBA (see 15.41.1).

The performance gains for such routines are similar to those produced by **SATALL** with reductions of up to 3.5 times on a Quad-core PC (see Figure 15.9 for applications of SKIM_ALL). Performance benefits continued to improve with five cores but there was some erosion of the gains beyond six cores.

Figure 15.9 – SATLOOK Performance (Medium Size Network using SKIM_ALL)

15.53.3.3 SATUFO (Multi-threaded)

SATUFO Multi Core may be used to create a network .UFO file from a .UFC file (see 15.23.7). The same implementation is used as with **SATALL** and **SATLOOK** skimming. The .UFO file may be created as part of the main **SATALL** assignment by setting SAVUFO=T and MULTIC=T or, alternatively, as separate standalone process following the main assignment via the batch file SATUFO.BAT (with MULTIC=T previously used in the main assignment).

15.53.3.4 SATPIJA_MC (Distributed)

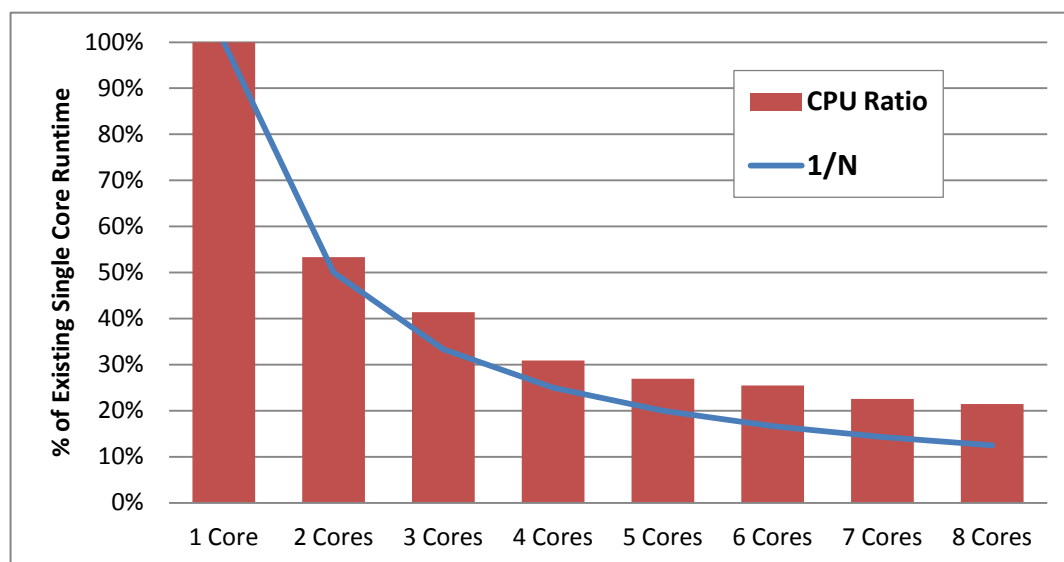
Unlike the SATALL, SATLOOK and SATUFO Multi-Core applications, **SATPIJA** uses a **distributed** approach whereby the creation of the PIJA file from the assignment is split into 'N' blocks of zones (see 13.4.9), with each block undertaken by a separate **run** of **SATPIJA**. Each **SATPIJA** run is undertaken in a separate sub-directory (or 'production folder') and an extra (short) **SATPIJA** run is undertaken at the end to combine the 'N' (smaller) PIJA files into a single file.

The process is automatically controlled by a new **SATPIJA_MC** batch file (13.6.3) so, in theory, there are no changes to either the basic program, \$SATPIJA.EXE, or to its associated batch file, SATPIJA.BAT (13.6.2).

As with **SATALL**, the performance benefits will vary between models and the PC hardware available. The splitting of the production of the PIJA into zones blocks is (currently) undertaken based on the sequential zone numbers and the distribution of trips in the matrix is unlikely to be equally shared between the blocks of zones. In addition, for each of the **SATPIJA** runs, the network, matrix and control files need to be copied to/from the 'production folders' which will incur a performance 'hit'.

The performance of the distributed **SATPIJA_MC** on a **very large SATURN** network is shown below in Figure 15.10. As noted above, the potential benefits will be dependent on the model and PC hardware used.

Figure 15.10 – SATPIJA_MC Performance (Very Large Network)



15.53.3.5 Performance Scaling

As illustrated in the figures above, the practical testing showed that there were (typically) negligible performance benefits over and above the use of five cores. This 'throttling' of the performance arises due to the limited memory available within the internal CPU Level 1/2/3 caches.

Conversely practical testing on other hardware systems (such as Blade servers) shows further performance benefits arising with six or more cores. The performances gains are clearly dependent on the **SATURN** model and computer hardware used.

15.53.3.6 Running More than One Multi-Core Assignment

In Section 15.52, we describe how **SATURN** may be used (and controlled) to undertake parallel operations. The same procedures may be used with **SATURN** Multi-Core without any change to those procedures.

15.53.3.7 SATCH_MC: Distributed Trip Matrix Cordoning

A distributed procedure SATCH_MC may be used to create a multiple user class cordoned trip matrix by creating cordoned matrices **by user class** within separate processors and then finally creating a full stacked trip matrix by stacking the individual sub-matrices using **MX**. See 12.1.6.

15.53.4 Multi-Core Parameters

15.53.4.1 Options

To activate the multi-threaded operations within **SATALL** and **SATLOOK** once installed, the &PARAM namelist parameter **MULTIC** is set to TRUE in the network .dat file and the Windows Operating System handles the allocation of the computational calculations between the available threads available. The value of

MULTIC parameter is stored in the network binary files (.ufn/s) and the value read by **SATALL**, **SATLOOK**, etc. where necessary.

An additional (integer) namelist parameter **MCALG** selects one of a set of optional algorithms which are basically provided for internal testing. We recommend the default value of 1.

A further (integer) &PARAM Namelist parameter **MCNUM** defines the maximum number of core processors to be used on the computer; default 0 (meaning use all available threads). See 15.53.4.2 below.

For the other Multi-Core programs (i.e. **SATPIJA**, **SATCH** and **SATUFO**), the Multi-Core batch files have to be used.

15.53.4.2 Upper Limit on MCNUM Values

The SATURN-MC will require more memory than the standard versions dependent on the number of threads available. Within the software, there is no restriction set on the maximum number of threads that may be used. For the distributed processes, a practical limit of 8 (eight) was coded but for version 11.3 onwards, the limit was increased to 32.

15.54 SATURN CASSINI

15.54.1 Overview

CASSINI is a Visual-Basic program developed to significantly reduce **SATURN** runtimes when **SATURN** is used within a Variable Demand Model such as a simple DIADEM model or a more complex, bespoke modelling system. See Section 7.4.1 for a general discussion of the problems of convergence between supply (i.e., assignment/SATURN) and trip matrix demand models and Section 7.4.5 for a description of the iterative “cobweb” loops between supply and demand models whose runtimes CASSINI seeks to “optimise”.

CASSINI enables the user to automatically adjust the convergence targets set for each run of **SATURN** to match the current level of convergence achieved for the supply-demand “cobweb” loops. Typically, a ‘relaxed’ set of convergence criteria would be set for the initial loops when supply-demand convergence is poor and the trip matrices are still highly uncertain but these would be subsequently tightened as the overall model convergence improves; in other words, reducing the ‘over-convergence’ within the supply model (i.e., **SATURN**).

See section 7.4.5.3 for a more general discussion of the principles applied by CASSINI.

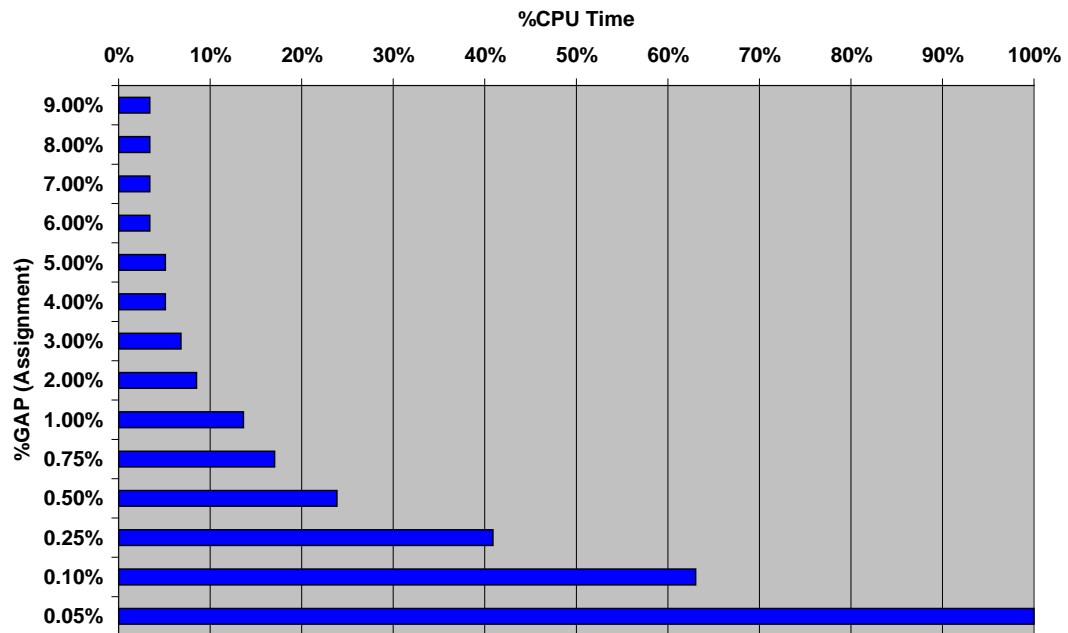
Appendix R contains a copy of the ETC2009 paper that describes the practical; benefits of CASSINI within a full WebTAG-compliant demand model.

15.54.2 Basic Principles

Given a fixed trip matrix **SATURN** uses internal loops between its assignment and simulation sub-models as well as internal iterations within the two sub-models in order to achieve an overall equilibrium solution in terms of path-flow choices as best represented by its “gap value” (see 9.2.1).

A characteristic of the process is a rapid initial descent before a much more gradual approach to a highly converged solution as shown in Figure 15.11 below. In this example, to achieve a %GAP value of 0.05 requires around 20 times the CPU time to achieve a %GAP of 5.0, eight times the time to achieve a %GAP of 1.0 and four times the time to achieve a %GAP of 0.5 respectively. Clearly, significant CPU savings may be achieved by (appropriately) reducing the convergence targets for the **SATURN** highway model where possible.

Figure 15.11 – Typical SATURN Model Convergence Profile



However, **SATURN** may also be embedded within a larger demand model structure (aka VDM Shell) in which the trip matrices are not fixed but are variable and cost-dependent and this larger model structure must also converge to an equilibrium solution (see 7.4.1). Typically some form of cobweb loop between supply (assignment) and demand (see 7.4.5) is used in order to achieve equilibrium between the two sub-models as illustrated in Figure 7.8.

We may quantify the degree of convergence between successive loops of the demand models by a “supply-demand gap value” as given in TAG Unit M.2 and defined by:

$$GAP - SD = \frac{\sum_{ijctm} C(X_{ijctm}) \left| D(C(X_{ijctm})) - X_{ijctm} \right|}{\sum_{ijctm} C(X_{ijctm}) X_{ijctm}} * 100$$

where:

- ◆ X_{ijctm} is the current flow vector or matrix from the model
- ◆ $C(X_{ijctm})$ is the generalised cost vector or matrix obtained by assigning that matrix

- ♦ $D(C(X_{ijctm}))$ is the flow vector or matrix output by the demand model, using the costs $C(X_{ijctm})$ as input; and
- ♦ $ijctm$ represents origin i , destination j , demand segment/user class c , time period t and mode m .

The convergence profile of GAP-SD over cobweb loops is similar to the assignment profile of GAP over internal loops, i.e., decreasing rates of improvement as convergence improves, as shown below in Figure 15.12.

The objective of CASSINI is therefore to minimise the overall CPU time required in order to achieve a satisfactory degree of convergence within both **SATURN** (the supply model) **and** the supply-demand model; i.e., both GAP and GAP-SD must be sufficiently near zero at the end of the process. (We assume here that the CPU time required to run the demand model on its own (i.e., to produce the new set of trip matrices) is effectively fixed per loop and that internal convergence within the (pure) demand model is not an issue.)

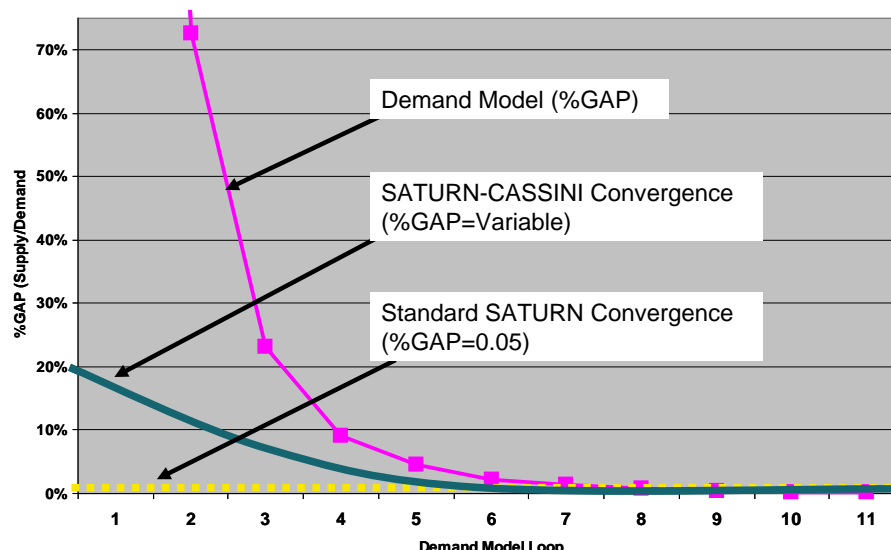
CPU may be reduced by either reducing the time per **SATURN** run and/or by reducing the total number of cobweb loops (or, as it turns out, by reducing the former and not increasing the latter too much). We achieve this by noting that it is not efficient to spend a lot of CPU obtaining a highly internally convergent **SATURN** assignment for a particular trip matrix if that trip matrix is then going to be considerably changed by the next supply-demand loop. For example, there is no point in having link flows accurate to $\pm 0.1\%$ if trip matrix cells are varying by $\pm 10\%$.

We therefore apply a principle of “relaxed convergence” (see 7.4.5.3) by specifying relatively easy convergence criteria for the initial **SATURN** runs when the trip matrix to be assigned is still “in flux” but to tighten up those criteria once the demand trip matrices begin to stabilise. While this may potentially increase the overall number of cobweb loops required to achieve convergence the expectation is that that increase will be more than offset by the CPU saved on earlier loops where internal **SATURN** convergence is much faster.

We may note that this process of “relaxed convergence” is very similar to that used by AUTONA (see 9.5.4) whereby we set “easy” assignment stopping conditions when the assignment-simulation loops are poorly converged (in order to minimise assignment CPU) but tighten them up as the assignment-simulation convergence improves.

By setting a more relaxed highway convergence target for the early cobweb loops using CASSINI, considerable savings in CPU time may be achieved as the ‘over-convergence’ of the highway assignment is reduced. These two convergence profiles are also shown below in Figure 15.12.

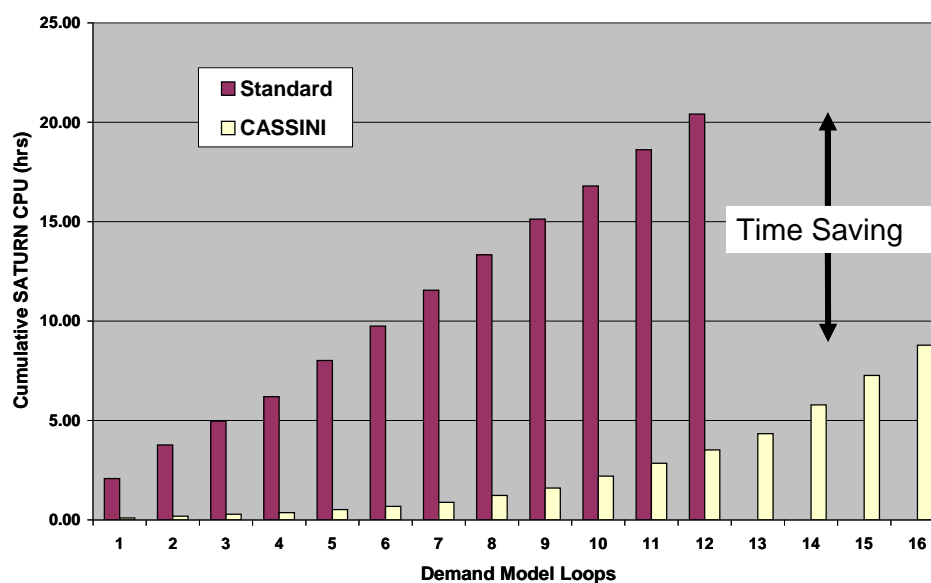
Figure 15.12 – Typical Demand Model Convergence Profile



15.54.3 Performance Gains

With CASSINI introduced, the demand model usually requires a few more loops to achieve the same %GAP-SD value of <0.2 (say) – typically an extra three or four loops reflecting the slower descent in this example. Nevertheless, there was an overall saving of around 50% in the total CPU time required compared to the standard method as shown below in Figure 15.13.

Figure 15.13 - Comparison of Highway Model Runtimes by Demand Loop

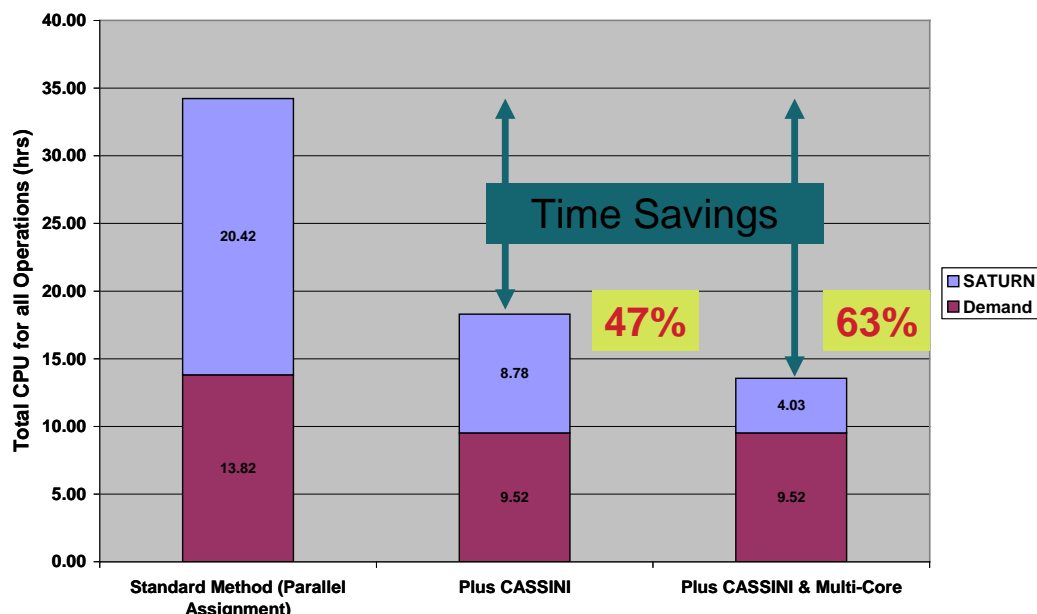


15.54.4 Compatibility with SATURN Multi-Core

CASSINI is fully compatible with SATURN Multi-Core, the new multi-threaded version of the **SATURN** assignment program. The recent testing work using the

GBMF modelling system demonstrated that using Multi-Core has reduced overall CPU times by a further 25% as shown below in Figure 15.14 below.

Figure 15.14 – Performance of CASSINI and SATURN Multi-Core



15.54.5 Convergence Strategies

To operate CASSINI, the user needs to define the convergence strategy that describes how the **SATURN** convergence parameters should change in response to improving convergence in the trip matrices. As shown earlier in Figure 15.12, the convergence parameters adopted for the early loops should be relaxed and progressively tightened as the demand model convergence improves.

If the assignment convergence strategy is too relaxed then the supply-demand model may not converge whereas setting too tight convergence criteria for the initial loops may over-converge the highway assignment and 'waste' CPU time. As such, it is a balancing act but it's better to err on the side of caution and over-converge the assignment (as a fully converged model remains the ultimate goal).

15.54.6 Running SATURN CASSINI

In normal operation, the CASSINI program is usually called internally within **SATNET** and produces a supplementary ASCII data file containing **eXtra Convergence Parameters (.XCP)** which propose new values for the relevant convergence parameters such as MASL, etc. etc. **SATNET** then reads in this new XCP file before fully processing the main network data file - the parameters in the XCP file **overwriting** those contained in the network data file. CASSINI is activated in **SATNET** by setting the parameter CASINI=T under &OPTION.

15.54.6.1 File Inputs

CASSINI requires three input files, namely:



- ◆ An existing **SATURN** Network data file with some additional parameters to control the CASSINI process;
- ◆ A CASSINI Control ASCII file that defines the convergence strategy/strategies to be implemented; and
- ◆ An ASCII report file on the Demand Model convergence (which defaults to a DIADEM output file).

SATURN NETWORK FILE

As noted above, to operate CASSINI, a number of new parameters need to be added to the existing &OPTION section (see 6.1) in the **SATURN** Network data file as described below.

CASINI	If TRUE, CASSINI will be called within SATNET and a number of additional checks will be undertaken to ensure the files named below exist. If any of these files do not exist, a semi-fatal error occurs.
CASTXT	Specifies the type of demand model used and the file format (and other operations) that CASSINI will expect. There are currently two options either: <p>'DIADEM' file format and CASSINI will extract the convergence of the demand model from a standard DIADEM report file a illustrated below – this is the default option, or</p> <p>A simpler 'OTHER' file format with the file consisting of two data fields in CSV format. The first value is the demand model loop number whilst the second value specifies the GAP-SD convergence of the demand model.</p>
FILCAS	The “file name” of the CASSINI Control file defining the convergence strategy to be applied (see below for more information). Default - blank (i.e., no file defined at this stage)
FILGAP	The “file name” of the ASCII CSV file reporting the convergence of the demand model Default – blank (i.e., no file defined at this stage)

Note that the convergence parameters in the **SATURN** network file should be relaxed as these will be applied for the assignment of the first demand model loop. These will be subsequently overwritten by .XCP produced by CASSINI.

CASSINI CONTROL FILE

The convergence strategy is defined in the CASSINI Control file (as specified by FILCAS parameter). The strategy is defined by setting a series of %GAP-SD thresholds which, for a given %GAP-SD (or lower) in the demand model, the user defines the parameters that CASSINI will export to the .XCP file. The parameters are a sub-set of those normally contained in the &PARAM &END section of the

network data file, in particular those that relate to convergence options within **SATALL** as shown in the table below. (N.B. the list may be extended if necessary.)

The current list of parameters that may be changed by CASSINI is as follows (with the new v11.2 parameters in italics):

- ◆ &OPTIONS: UPDATE, WSTART, *DIADEM*
- ◆ &PARAMS: SAVEIT, UFC109, FISTOP, STPGAP, XFSTOP, UNCRTS, MASL, KONSTP, ISTOP, MET, NISTOP, NITA_M, NITA_C, NITA_S, NITS, *NITA, SPIDER, MULTIC, ILOVEU, NITS_M, SAVUFO, AK_MIN*

The user may also provide more than one strategy with the strategy chosen determined by the number of loops undertaken by the demand model. This provides the user with the flexibility to switch between strategies depending on whether the demand model is in an *early* stage (e.g., loops 1 to 5 for example), *middle* stage (e.g., loops 6 to 10) or *late* stage (loops 11 onwards) and approaching completion as illustrated below:

	<i>Early</i>	<i>Middle</i>	<i>Late</i>
Demand Model Loop	1 to 5	6 to 10	11 to 15
KONSTP	1	1	1
STPGAP	10%	2.5%	0.05%
ISTOP	90%	95%	98%
MET	0	0	0
SAVEIT	F	T	T
UNCRTS	10%	2.5%	0.05%
NISTOP	1	1	2
MASL	10	40	80
NITA_S	25	100	250

Each strategy is identified by a “[**LoopThreshold** Y]” where Y is the demand model loop which that strategy is used. If more than one strategy is specified, the Loop Thresholds must be provided in ascending order.

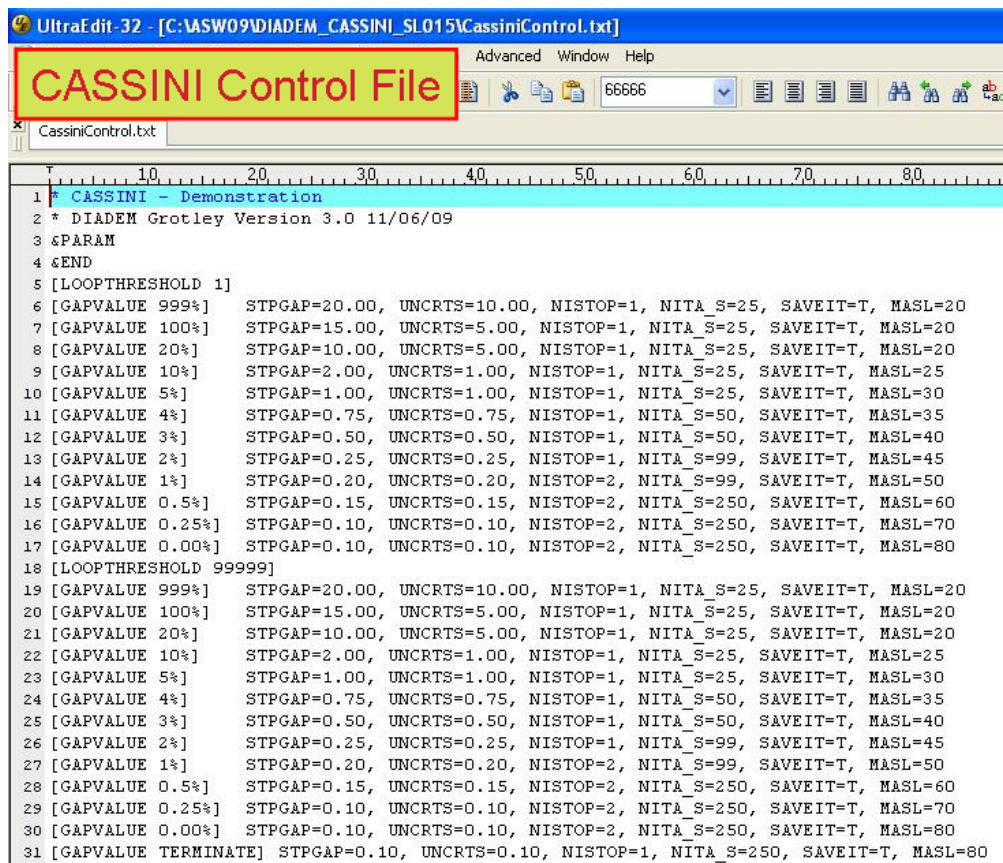
Within each strategy (or LoopThreshold), the following row(s) provide the parameters to be transferred to the .XCP file depending on the GAP-SD value reported in the Demand Model convergence file (FILGAP). Each row starts with **GAPValue** Z% where Z% is the %GAP-SD threshold that identifies the parameter(s) to be adopted for the next loop if the demand model convergence is less than the value of Z. The parameters for each GAP-SD value must be contained on the same row and each parameter separated by a comma.

In operation, CASSINI will:

- ◆ Read the demand model convergence file (as defined by FILGAP) and determine the number of loops undertaken (so far) by the demand model and resulting demand model convergence;
- ◆ Read the CASSINI Control file (as defined by FILCAS)
- ◆ Match the number of demand loops undertaken and the strategy to apply (as defined by the Loop Threshold). So, for example, if there are two strategies: an initial strategy for the first loop and a second defined for loop 5 onwards [i.e., LoopThreshold 5], and four loops have been undertaken so far, the first strategy will be applied;
- ◆ Within that strategy, compare the current demand model GAP-SD value against the various %GAP-SD ranges [GAPVALUE] and export the parameters to the XCP file.

An example of the control file is provided below.

EXAMPLE OF THE CONTROL FILE



```

1 * CASSINI - Demonstration
2 * DIADEM Grotley Version 3.0 11/06/09
3 &PARAM
4 &END
5 [LOOPTHRESHOLD 1]
6 [GAPVALUE 999%] STPGAP=20.00, UNCRTS=10.00, NISTOP=1, NITA_S=25, SAVEIT=T, MASL=20
7 [GAPVALUE 100%] STPGAP=15.00, UNCRTS=5.00, NISTOP=1, NITA_S=25, SAVEIT=T, MASL=20
8 [GAPVALUE 20%] STPGAP=10.00, UNCRTS=5.00, NISTOP=1, NITA_S=25, SAVEIT=T, MASL=20
9 [GAPVALUE 10%] STPGAP=2.00, UNCRTS=1.00, NISTOP=1, NITA_S=25, SAVEIT=T, MASL=25
10 [GAPVALUE 5%] STPGAP=1.00, UNCRTS=1.00, NISTOP=1, NITA_S=25, SAVEIT=T, MASL=30
11 [GAPVALUE 4%] STPGAP=0.75, UNCRTS=0.75, NISTOP=1, NITA_S=50, SAVEIT=T, MASL=35
12 [GAPVALUE 3%] STPGAP=0.50, UNCRTS=0.50, NISTOP=1, NITA_S=50, SAVEIT=T, MASL=40
13 [GAPVALUE 2%] STPGAP=0.25, UNCRTS=0.25, NISTOP=1, NITA_S=99, SAVEIT=T, MASL=45
14 [GAPVALUE 1%] STPGAP=0.20, UNCRTS=0.20, NISTOP=2, NITA_S=99, SAVEIT=T, MASL=50
15 [GAPVALUE 0.5%] STPGAP=0.15, UNCRTS=0.15, NISTOP=2, NITA_S=250, SAVEIT=T, MASL=60
16 [GAPVALUE 0.25%] STPGAP=0.10, UNCRTS=0.10, NISTOP=2, NITA_S=250, SAVEIT=T, MASL=70
17 [GAPVALUE 0.00%] STPGAP=0.10, UNCRTS=0.10, NISTOP=2, NITA_S=250, SAVEIT=T, MASL=80
18 [LOOPTHRESHOLD 99999]
19 [GAPVALUE 999%] STPGAP=20.00, UNCRTS=10.00, NISTOP=1, NITA_S=25, SAVEIT=T, MASL=20
20 [GAPVALUE 100%] STPGAP=15.00, UNCRTS=5.00, NISTOP=1, NITA_S=25, SAVEIT=T, MASL=20
21 [GAPVALUE 20%] STPGAP=10.00, UNCRTS=5.00, NISTOP=1, NITA_S=25, SAVEIT=T, MASL=20
22 [GAPVALUE 10%] STPGAP=2.00, UNCRTS=1.00, NISTOP=1, NITA_S=25, SAVEIT=T, MASL=25
23 [GAPVALUE 5%] STPGAP=1.00, UNCRTS=1.00, NISTOP=1, NITA_S=25, SAVEIT=T, MASL=30
24 [GAPVALUE 4%] STPGAP=0.75, UNCRTS=0.75, NISTOP=1, NITA_S=50, SAVEIT=T, MASL=35
25 [GAPVALUE 3%] STPGAP=0.50, UNCRTS=0.50, NISTOP=1, NITA_S=50, SAVEIT=T, MASL=40
26 [GAPVALUE 2%] STPGAP=0.25, UNCRTS=0.25, NISTOP=1, NITA_S=99, SAVEIT=T, MASL=45
27 [GAPVALUE 1%] STPGAP=0.20, UNCRTS=0.20, NISTOP=2, NITA_S=99, SAVEIT=T, MASL=50
28 [GAPVALUE 0.5%] STPGAP=0.15, UNCRTS=0.15, NISTOP=2, NITA_S=250, SAVEIT=T, MASL=60
29 [GAPVALUE 0.25%] STPGAP=0.10, UNCRTS=0.10, NISTOP=2, NITA_S=250, SAVEIT=T, MASL=70
30 [GAPVALUE 0.00%] STPGAP=0.10, UNCRTS=0.10, NISTOP=2, NITA_S=250, SAVEIT=T, MASL=80
31 [GAPVALUE TERMINATE] STPGAP=0.10, UNCRTS=0.10, NISTOP=1, NITA_S=250, SAVEIT=T, MASL=80

```

EXAMPLE OF THE 'DIADEM' DEMAND MODEL FILE

Microsoft Excel - 4.4C_results.csv

FileEditViewInsert

DIADEM Results File

Type a question for help

100%

10

Reply with Changes...End Review...

PivotTable

G3		8.0056835%																
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
Iteration		Step length	Max flow c	Obj fn	Gap Abs	Rel	Cost stability			Flow stability				Totals	Cost			
Main	Sub						RAAD	AAD	RMS	%<5	RAAD	AAD	RMS	%<5	Trips	Cost		
1	1	0.3	0	3289.929	106751.3	8.01%	0	0	0	0%	0	0	0	0%	73792.98	1333444		
2	1	0.3	-5.3062	4400.637	124263.5	8.84%	0.083895	1.1828	1.906136	50.62%	0.040418	0.003911	0.034065	70.69%	73780.82	1405136		
3	1	0.3	-10.6656	1904.122	84682.3	6.09%	0.033067	0.414765	0.759381	82.44%	0.039986	0.003704	0.039421	72.02%	73754.44	1390225		
4	1	0.3	-5.5305	824.1736	55114.65	4.02%	0.032882	0.423689	0.864608	82.90%	0.029493	0.00266	0.025931	80.78%	73737.05	1371139		
5	1	0.3	2.888983	693.2769	40291.45	2.93%	0.022827	0.306729	0.592542	89.12%	0.019947	0.001841	0.01706	90.04%	73730.31	1375486		
6	1	0.3	4.777853	328.427	31538.26	2.31%	0.015312	0.234167	0.480673	93.24%	0.014037	0.001346	0.015647	95.60%	73722.41	1366421		
7	1	0.3	-1.71161	176.5888	20178.13	1.48%	0.009991	0.143058	0.292238	97.21%	0.011124	0.001056	0.010769	97.10%	73719.51	1367360		
8	1	0.3	2.449214	154.3921	16753.88	1.23%	0.006764	0.099412	0.209481	98.87%	0.007531	0.000681	0.007897	98.87%	73716.99	1363020		
9	1	0.3	-2.21613	99.01255	14737.21	1.08%	0.00891	0.109396	0.219731	97.86%	0.005951	0.000571	0.007384	99.46%	73716.19	1362522		
10	1	0.3	-1.23834	51.8912	10911.21	0.80%	0.004953	0.0769	0.179347	99.32%	0.005552	0.000521	0.005913	99.58%	73715.76	1363784		
11	1	0.3	-1.05451	38.60629	8839.645	0.65%	0.004379	0.059696	0.106201	99.57%	0.00429	0.000391	0.004281	99.93%	73715.17	1362662		

EXAMPLE OF THE 'OTHER' DEMAND MODEL FILE

File Edit Search Project View Format Column Macro Advanced Window Help

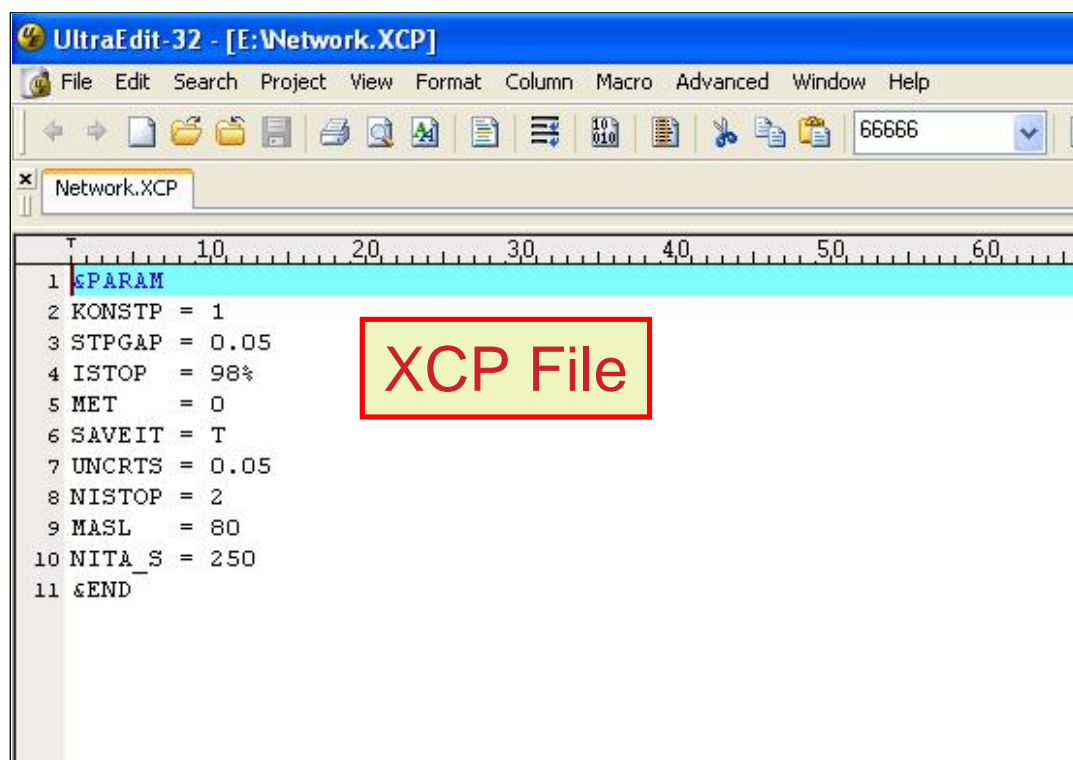
31JRT2137_Conv.txt*

```

1 *Demand/Supply *Gap, 06/02/2010, 16:23:14.09
2 Iteration, Percentage_Gap,
3 i=1, 286.1230,
4 i=2, 70.6745,
5 i=3, 23.6462,
6 i=4, 9.2933,
7 i=5, 4.4406,
8 i=6, 2.2028,
9 i=7, 1.3618,
10 i=8, .8291,
11 i=9, .4435,
12 i=10, .2254,
13 i=11, .1535,
14 i=12, .0959,
15 i=13, .0740,
16 i=14, .0586,
17 i=15, .0481,
18

```


EXAMPLE OF THE XCP FILE

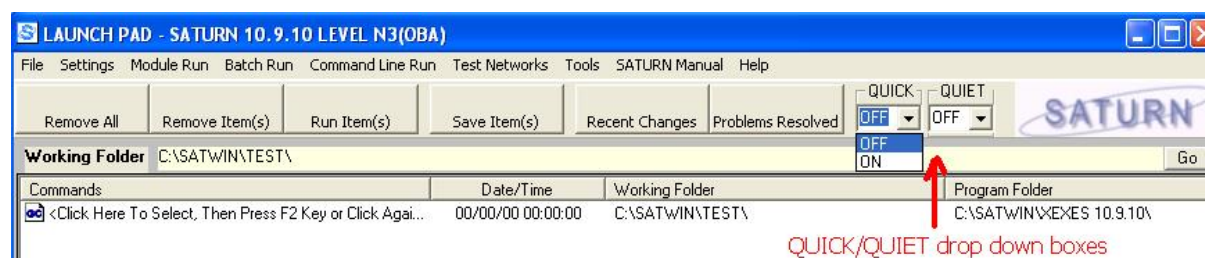


15.55 QUIET & QUICK Options via SATWIN

The QUIET and QUICK options in **SATURN** (see section 14.9 and 14.10) may also be activated via SATWIN10 or SATWIN11. Once QUICK and/or QUIET is toggled 'ON', the option remains active for subsequent **SATURN** commands within SATWIN until they are toggled to 'OFF'. The QUICK and/or QUIET settings in SATWIN are also applied to DOS command line runs created by the SATWIN (ie via the "TOOLS/SATURN DOS Command Shell" menu option). The SATWIN settings can be overwritten if QUICK and/or QUIET is subsequently explicitly set on the command line.

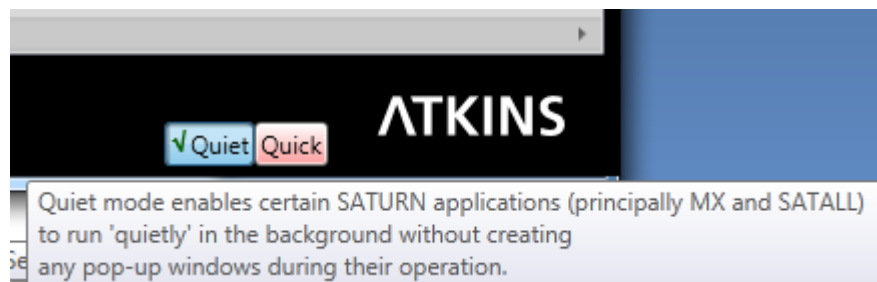
15.55.1 Using SATWIN10

In SATWIN 10 this is done by setting the QUIET and QUICK drop down to ON as shown below.



15.55.2 Using SATWIN11

In SATWIN 11 this is done by pressing the QUIET and/or QUICK buttons located on the bottom right-hand corner of the SATWIN11 interface as shown below.



15.56 Network Aggregation (SPIDER)

15.56.1 Basic Principles

Network aggregation is a technique whereby links and/or nodes in the basic assignment network may be combined together into an equivalent set of aggregated links/nodes with the objective of reducing the cpu time required to carry out the basic assignment steps of tree building and loading.

For example, as illustrated below, a one-way link from A to B followed (in series) by a one-way link from B to C (so that node B has only one entry and one exit) may be replaced by a one-way link from A to C with a cost equal to the sum of the costs on A-B and B-C. Thus we have reduced two links to one link and removed node B while at the same time retaining the same cost of travel between A and C so that, if links A-B + B-C are part of a minimum cost path from a particular origin in the original network, then so is A-C in the aggregated version of the network.

$$A \rightarrow B \rightarrow C \quad \Rightarrow \quad A \rightarrow C$$

The cpu time required to build a minimum cost (shortest path) tree from a single origin to all nodes in a network may be estimated by formulae such as, for the d'Esopo algorithm most commonly used in **SATURN**:

$$T_{\text{cpu}} = a_1 + (a_2 N_{\text{nodes}} + a_3 N_{\text{links}}) (1 + a_4 \text{Sqrt}(N_{\text{nodes}}))$$

See "Improved shortest path algorithms for transport networks" by Dirck Van Vliet, Transportation Research Vol. 12, 7-20 (1978) and reproduced in Appendix T.

Other algorithms may have slightly different functional forms but all share the same basic property of being increasing functions of the number of nodes and the number of links in the network. Similarly the cpu time required to load a single (origin) row of the trip matrix is proportional to the number of destinations times the number of links.

Thus the total time required to carry out a single all-or-nothing assignment step, the basic building block of the Frank-Wolfe algorithm, is an increasing function of (a) the number of (origin) zones, (b) the number of nodes and (c) the number of links. Any reductions in one or all of these should therefore lead to reduced cpu times; network aggregation achieves this by reducing the number of nodes and/or links.

In addition network aggregation reduces the cpu time involved in building trees during post-assignment analysis such as skimming, select link analysis, etc, etc. See Section 15.56.7.

A condensed version of the material that follows was presented at the ETC (European Transport Conference) in Glasgow, 2010, by Wright et al and reproduced in Appendix S (.pdf version only).

15.56.2 Aggregation Techniques

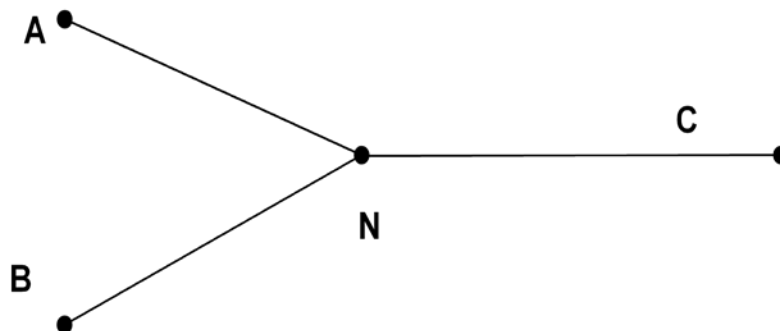
15.56.2.1 2-arm Links in Series

The simplest example of combining two links in series into one has been illustrated above. Clearly the same technique may be applied in both directions when both links A-B and B-C are two-way (assuming that U-turns are banned at B); hence an aggregate link A-C replaces A-B and B-C while C-A replaces C-B and B-A.

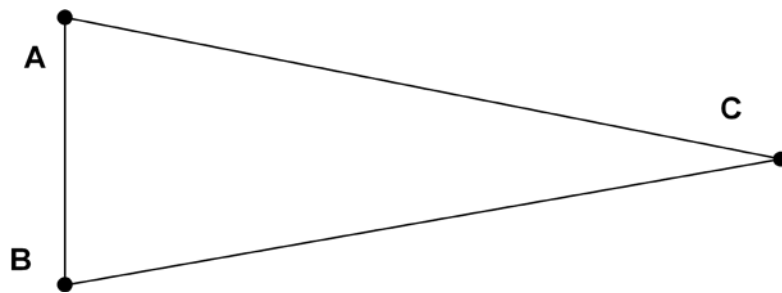
The same technique may clearly be extended to the case where there are a series of more than one two-arm nodes between A and B such that a single link from the start to the exit node replaces all the intermediate links and all the intermediate nodes are removed. (This form of configuration occurs not infrequently in **SATURN** networks when a number of artificial nodes are inserted between two “main” nodes in order to give the link “shape” – although clearly a better method is to define the shape via a .GIS file. In fact a common theme in network aggregation is that the degree of potential aggregation and time savings that are available may depend very sensitively on the coding techniques adopted.)

15.56.2.2 Aggregating Multiple-arm Nodes

It is also possible to eliminate nodes with more than 2 arms, for example a 3-arm node N as illustrated below



May be aggregated into a “triangle”:



Such that the cost on the aggregate link A-C is the sum of the original costs on A-N plus N-C. (Note that it is not necessary to create a "U-turn" link, say, from A to A equivalent to A-N plus N-A since, even though the movement may be valid in the original network, it can never be part of a shortest path tree.)

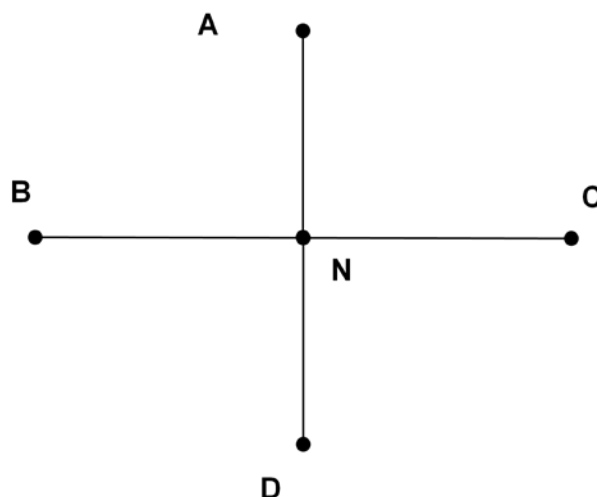
Note that in this case, if all the original links are 2-way, then the original network segment contained 6 links as does the aggregated segment. So, if we have not managed to reduce the number of links, we have at least removed one node which, given the form of equation (15.x), should still lead to an overall reduction in cpu time.

On the other hand if one of the original links were one-way - imagine that A-N were one-way for example - then the original segment has 5 one-way links but the aggregated segment has only 4 (A-C, A-B, B-C and C-B) and therefore the numbers of **both** nodes and links has been reduced.

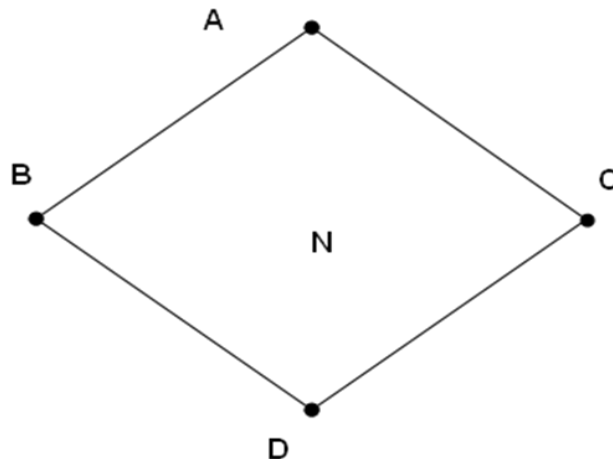
A common example of a 3-arm node might be an entry ramp onto a motorway where A-N would be a one-way entry onto a motorway with one-way links B-N and N-C. In this case 3 links are reduced to 2.

The entry ramp configuration may be generalised to any node that has a single one-way exit and n one-way entries such that $n+1$ links are reduced to n . Equally all nodes with a single entry and multiple exits may be aggregated to save one node and one link.

Indeed a node with any number of entry/exits may always be removed by aggregating pairs of entry/exits. The example of a 4-arm node is illustrated below.



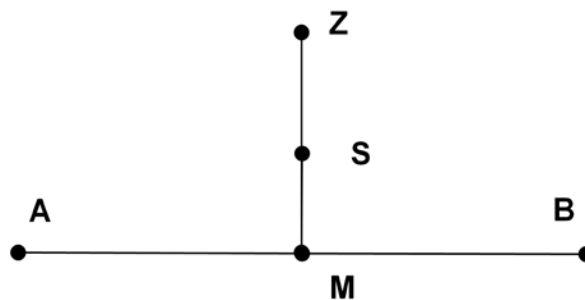
which reduces to:



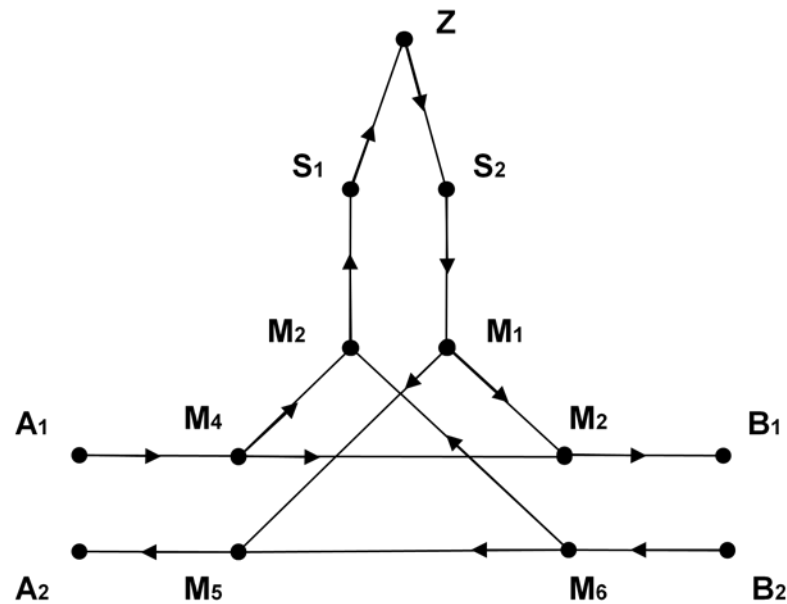
Note that here, if all arms are two-way, then we actually increase the number of links from 8 to 12 in order to remove 1 node, although if one or more of the arms are one-way the increase in links is reduced and may even represent a reduction. (E.g., 2 one-way entry links and 2 one-way exit links reduce 4 links to 2.)

15.56.2.3 Application to “Spigot Zone Connectors”

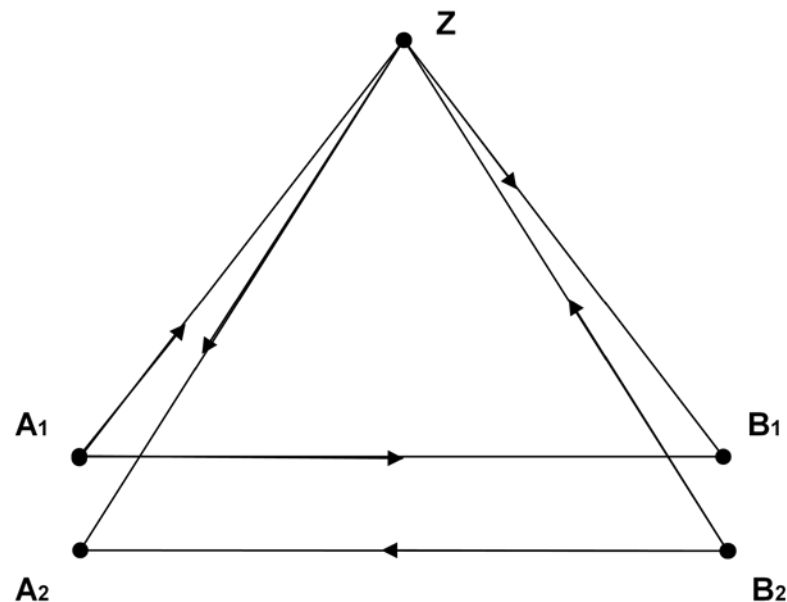
A not uncommon coding “trick” used in SATURN is illustrated below where a zone Z, rather than being connected onto a link A-B directly, is connected by an external simulation “spigot” or “stub” node S which is in turn connected to an “artificial” mid-link node M. (See also Sections 16.6.2 to 16.6.4 and 11.9.4.1)



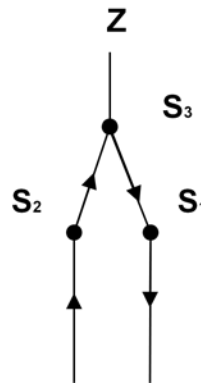
However, in the assignment network representation of this section of the network where “mini nodes” are created at the start and end points of all one-way links, the situation would be as follows:



We therefore note that traffic leaving the network from zone Z has only two possible paths available to it: Z-S1-M1-M3-B2 or Z-S2-M2-M4-A2. Equally there are only two possible paths into Z from A1 or A2. Thus, in this situation we may aggregate the network into 4 aggregate links – Z-B2, Z-A2, B1-Z and A1-Z – while at the same time removing **all** the mini nodes at S and M. (Assuming all links are two-way this removes 8 nodes out of 13 and 8 links out of 14.)



Note that if the spigot node S has been coded as a **buffer** node under 33333 (see Section 16.6.3) then a third mini-node is created at S has shown below. This however does not change the general principle that zone Z is connected via aggregate centroid connectors to nodes A and B but it does increase by 2 the number of assignment links replaced.



We may further note (see also 16.6.4) that the above buffer node connector allows possible U-turns via S_1 - S_3 - S_2 in the full network but that this possibility is explicitly excluded in the aggregate network since, e.g., there is no aggregate link created from A_1 to A_2 which would correspond to a U-turn.

15.56.2.4 Spigot Centroid Connectors in General

A more general variation on the spigot centroid connector configuration occurs when the simulation node M is connected to **more** than 2 other internal simulation nodes. In this situation a “mini aggregation” may be invoked by substituting direct centroid connector links from M1 to Z and from Z to M2 in Figure 15.x with a reduced number of zones and/or links removed. However, see step 2) in 15.56.3, it is still an aggregation step worth doing.

15.56.2.5 Some Properties of Aggregate Networks

The final aggregate network will consist of a sub-set of the original nodes (since none of the steps described above introduce new nodes) plus a set of new aggregate links joining those nodes. Note that the number of zones remains unchanged and therefore the proportion of zones within nodes – as well as the proportion of centroid connector links within links – increases significantly.

In addition an aggregate network may contain a significant number of “duplicate links”, i.e., links with the same A-node and B-node, the reason for which is discussed below.

Both of these slightly unusual network properties may lead to variations in basic tree build algorithms becoming effective; see below.

Note that the sub-set of nodes which are retained within the aggregate network may be selected and therefore highlighted within **P1X**; see 11.6.3.5.

15.56.3 Implementation within SATNET

A (semi-empirical) methodology has been introduced into the network building procedures within **SATNET** to produce an aggregated network, activated if a &PARAM parameter SPIDER is set to .TRUE. (default .FALSE.). It proceeds via a number of successive steps as follows:

- 1) Aggregate certain “priority” nodes, i.e., buffer nodes where there are banned/penalised turns and weaving sections, where aggregation is essential for the modelling (see 15.56.7.3);
- 2) Aggregate all “spigot” centroid connectors (15.56.2);
- 3) Aggregate stub link centroid connectors to external buffer zones (see 15.56.2)
- 4) Remove any bus-only links (since they will never form part of minimum cost paths for trips in the trip matrix)
- 5) Aggregate all nodes which have a single exit with one or more entries (N.B. this will include all “dummy” 2-arm nodes)
- 6) Ditto but with nodes with a single entry and / or more exits
- 7) Aggregate all nodes with, progressively, 3 arms, 4 arms, etc. etc. up to a maximum of MAXSPA arms

Thus at the end of each step a new aggregated network is created and passed to the following step for further aggregation. Steps 5) to 7) are repeated iteratively with the maximum number of arms increased by one on each pass. Thus on the first pass all 3-arm nodes are aggregated in step 7), on pass 2 all 3- and 4-arm nodes are aggregated, etc. etc. The iterative loops are repeated until no further aggregation is feasible or the maximum number of arms per node which may be aggregated reaches MAXSPA as set in &PARAM (with a default value of 15).

A further rule is applied in step 7) which is that a node is only aggregated if, in addition to having less than a certain number of arms, it also satisfies the (highly empirical) rule that:

$$N_{\text{new}} \leq 23 + N_{\text{in}} + N_{\text{out}} + N_{2w} / 2$$

Where: N_{in} = the number of in-bound directional links

N_{out} = out-bound links

N_{2w} = number of two-way arms

N_{new} = number of new direct links that will be created = $N_{\text{in}} * N_{\text{out}} - N_{2w}$

For example, aggregating a 6-arm node with 2 two-way arms, 2 one-way inbound and two one-way outbound would create 14 new links from 8 existing links (i.e., we add 6 links and lose 1 node) but the above rule says to go ahead regardless. In fact this rule allows nodes with up to roughly 20 arms to be removed even if this seems totally counter-intuitive – empirically it saves CPU! And hence the default value of MAXSPA = 15 noted above.

Note that there is a large degree of overlap between some of the steps. Thus the nodes which are aggregated under steps 5) and 6) would also be picked up under step 7) but there may be a benefit to identifying the **simplest** structures and aggregating them first before eliminating the more complex node structures.

Note as well that the number of links per node is not fixed but potentially grows with each successive step. Thus node A may have initially have 4 arms but if one

of its neighbouring nodes B is aggregated then A will have additional links added to all of B's other neighbours.

At the end of the process the aggregated network structure is stored within the .ufn/.ufs files so that it may be optionally used within subsequent assignments and/or analyses.

15.56.4 Implementation within SATALL

Having created an aggregated network within **SATNET** the assignment procedure within **SATALL** may then be based on the aggregated network. Thus the basic Frank-Wolfe algorithm proceeds as normal with the one exception that step 3 (see Section 7.1.2) - build minimum cost trees and load all O-D trips to the minimum cost paths – uses the aggregated network. This in turn involves two extra steps:

- 1) Prior to tree building calculate the current cost of each aggregated link by summing the costs of its constituent links, and
- 2) Post loading transfer the flows onto the aggregate links back onto their constituent normal links to obtain $F_a(n)$.

All other FW steps, e.g. the optimum combination of link flows and the calculations of the objective function in step (4) are all based on the basic network definitions. (Note that steps (1) and (2) above are repeated once per user class for MUC assignment.)

While steps (1) and (2) are an extra overhead on the normal all-or-nothing loading sequence which increases cpu these are compensated by the reductions in cpu time for tree building and loading per origin and, provided that the number of origin zones is large, the latter will always outweigh the former. Indeed, the larger the number of zones, the more cpu time will be saved.

It should also be noted that the aggregate version of Frank-Wolfe may occasionally give different results to the normal version. One reason arises when there are two equal minimum cost routes between two nodes and the one selected is essentially arbitrary. Equal cost routes occur most commonly on the very first iteration where the costs are based on free flow speeds, distances etc. which may well be identical on parallel routes; on later iterations, where flows are essentially continuous variables, equal costs are much less likely. Another reason may be the treatment of possible U-turns at simulation-buffer boundaries. In any event, the final differences should be relatively small and it should be borne in mind that both solutions are equally valid.

15.56.5 Alternative Tree Building Algorithms

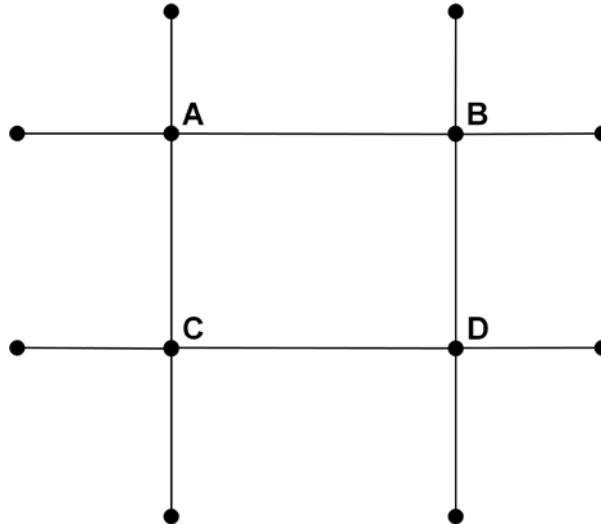
Having established a different “form” of network on which to build trees it should equally be feasible to create different tree building algorithms that take advantage of the new network properties.

15.56.5.1 Duplicate Links

For example, aggregate networks tend to have a large number of duplicate links (i.e., joining the same two nodes together) whereas these are not permitted in “normal” networks. (The reason that they are not permitted in normal networks is

not because they are judged not to exist but due to the technical problems of being able to uniquely identify links by their A-node and B-node; e.g., in a counts file.)

The reason that duplicates can arise in aggregate networks is illustrated below with a segment of a “grid network”.



If node D is aggregated a diagonal link from C to B will be created whose cost is equal to the costs of CD plus DB. Similarly if A is aggregated another diagonal link will be created from C to B with cost CA plus AB. For all origins (with costs fixed) only **one** of the two alternative CB links may possibly appear in the minimum cost tree, that version which has the lower cost. (Note that it is quite possible that **neither** link appears in the minimum cost tree if B has an entirely different back-node).

Thus if we eliminate the more expensive link between C and B prior to tree building we will save time on tree building since only one version of link CB will ever be considered as a candidate for inclusion in the minimum cost tree. Unfortunately it is not possible to totally eliminate one of the duplicates once and for all since the one to be eliminated depends on the current definitions of link costs which change throughout the assignment process. However for a particular iteration of Frank-Wolfe where the costs are fixed it is possible to eliminate the more expensive alternative and therefore save cpu time for each origin zone's tree build.

This modification has therefore been introduced into the Frank-Wolfe algorithm applied to aggregated networks and is found to reduce total cpu significantly. (Duplicate links do not need to appear just as pairs: it is quite feasible for **several** duplicate links to exist between the same two nodes so that eliminating all but one reduces cpu time still further.)

15.56.5.2 *Separate Centroid Connectors from Real Links*

Tree building algorithms are based on repeating a number of very simple steps a very large number of times; any reduction in the basic step sequence (no matter how silly it appears!) may lead to not insignificant reductions in CPU time.

Thus in the aggregate tree building algorithm based on d'Esopo-Pape (see Appendix T) we find empirically that splitting the algorithm into three distinct stages saves a small amount of time. Specifically the stages are:

- 1) Construct the minimum cost paths from the origin zone to all connected nodes. (Since we know that all the connected nodes **must** be added to the loose end table we can do away with that test.)
- 2) Carry on tree building through all “real” links but ignore all out-bound centroid connectors to destination zones.
- 3) For each destination zone consider all entry centroid connectors and choose the minimum cost alternative (Once again we avoid any tests as to whether a minimum cost link requires a loose-end table entry).

The reason that this 3-stage process appears to reduce CPU time for aggregate networks but not necessarily for normal networks is probably associated with the fact that aggregate networks contain a much higher proportion of zones and centroid connectors than normal networks (see 15.56.2 above).

15.56.5.3 Eliminating Zero-flow Links

If we “know” in advance that certain links are never going to feature in minimum cost O-D paths then they may be eliminated before the tree building takes place. In particular if a link has **zero** flow then it can never be part of a used path for an O-D cell with **positive** trips and it may be ruled out a priori.

For example, if we are re-constructing O-D paths post-assignment as part of a Select Link Assignment (see 15.23 and/or 11.8.1) then we are only interested in those paths which carry positive flows and clearly any link which we already know has zero flow cannot be part of those paths. Thus before carrying out SLA we remove all links with zero flow in total.

On the other hand if we are skimming, say, O-D distance or time then it is possible for a link with zero flow to be part of a min-cost O-D path where the O-D itself has zero flow.

Equally during the extra SAVEIT assignment where we have already carried out a full assignment as part of the assignment-simulation loops we know which links are unused and these can be eliminated within SAVEIT. (Although, strictly speaking, it is possible that, due to poor convergence, a link could be used during a SAVEIT assignment when it was never used during the “full” assignment.)

At the moment the “trick” of eliminating zero-flow links is used in the following situations:

1. SAVEIT assignments: see 15.23.2;
2. SAVUFO calculations: see 22.5.3;
3. Select Link Analysis (SLA) with **P1X**: see 11.8.1.12.
4. **SATCH** cordoned matrices; see 15.56.7.2.

Eliminating zero-flow links can substantially reduce CPU time in all instances since, empirically, it appears that over 50% of aggregate spider links may be unused.

N.B. In principle it is possible to apply the same rules to “basic” networks but since, in practice, there are very few if any “proper” links with zero flow then it is not worth the added effort.

15.56.6 Results from Representative Networks

15.56.6.1 Pure Assignment (SATASS only)

We display below a table of results from a randomly selected set of real-life networks in which we give:

- ◆ The number of zones and user classes (which are the same for both basic and aggregated networks)
- ◆ The number of (assignment network) nodes and links in the base network
- ◆ The number of nodes and links in the aggregated network
- ◆ The number of newly created aggregate links which are duplicates (joining the same A- and B-nodes)
- ◆ The total number of equivalent base links which the aggregate links map into
- ◆ The ratio of base/aggregate CPU time for a single Frank-Wolfe assignment (i.e. no SATSIM)

Table 15.2 – Performance Comparison (SATASS CPU time only)

Network	Zones	UC	Original Network		Aggregate Network		Duplicates	Equivalent	CPU Ratio
			Nodes	Links	Nodes	Links			
New Town ⁽⁵⁾	115	2	1,964	3,022	308	2,858	362	24,232	3.2
York	176	1	1,246	2,329	340	3,026	413	16,379	2.2
Horley ⁽⁴⁾	229	2	4,263	6,440	621	6,313	1,063	5,89	3.3
Heysham	299	3	3,758	6,198	692	6,712	949	50,401	1.9
Dorset ⁽²⁾	527	6	3,204	20,116	1,616	18,114	4,693	175,665	9.2
Corby	598	8	13,374	21,229	2,310	27,364	4,187	200,797	13.5
Bristol ⁽¹⁾	600	6	13,515	19,947	1,688	17,675	3,444	177,598	6.7
SALT	804	4	65,183	96,602	8,336	69,240	11,121	564,480	3.9
GMTU	993	1	42,665	63,596	5,264	60,363	14,422	567,896	11.0
East London	1,348	7	30,633	52,277	4,926	53,363	17,028	387,600	8.6
M25	1,417	5	75,178	109,568	9,992	68,584	5,601	556,742	9.6
Central London	1,638	7	28,587	60,325	6,064	61,922	22,645	335,510	6.0
South London ⁽³⁾	2,520	5	57,877	93,905	8,833	96,296	23,593	753,240	2.9
LoHAM	5,624	5	119,534	185,576	21,348	151,776	13,513	962,301	3.75

The networks are arranged in order of increasing number of zones.

It is difficult to draw any universal conclusions from the above table; clearly the improvement in CPU time is a function of certain network coding “idiosyncrasies” (e.g., whether or not stub zone connectors are widely used). There is a tendency for networks with smaller number of zones to be more efficient under aggregation as we might expect since the more zones there are, the more opportunities there are to reduce tree building times compared to the overheads involved in constructing aggregate link costs.

15.56.6.2 Full Assignments (Assignment & Simulation)

Since **SATURN** incorporates both assignment and simulation sub-models and the CPU time for the simulation is unaffected by network aggregation the overall reductions for full runs are less spectacular than those demonstrated for pure assignment sub-models above. Five of the above networks were selected (as identified with suffixes 1 – 5) to compare the overall runtimes for the full **SATURN** assignment using the four Frank-Wolfe-based assignment techniques currently available:

- ◆ Standard Frank-Wolfe (FW);
- ◆ Multi-Core Frank-Wolfe;
- ◆ Frank-Wolfe with Network Aggregation technique; and
- ◆ Multi-Core Frank-Wolfe with Network Aggregation technique.

The assignments were undertaken on the same desktop PC with up to four cores available to the software package. The three main elements of the SATURN assignment are:

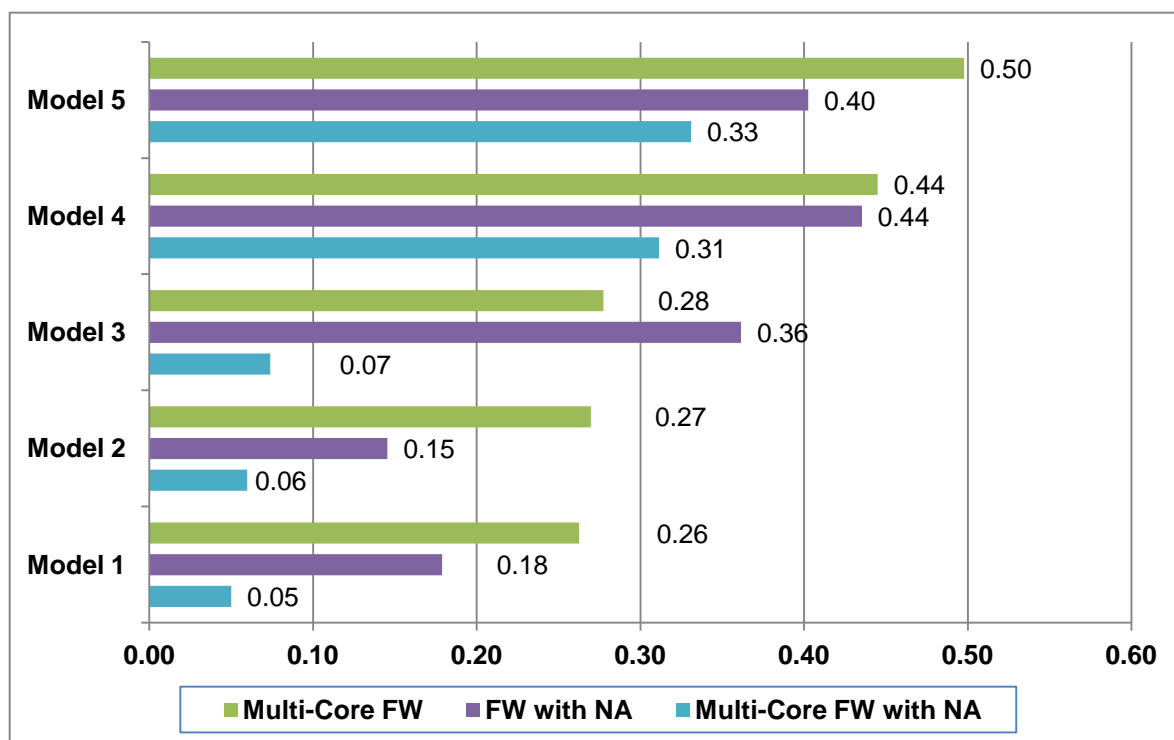
- ◆ Path-building and loading with fixed flow-delay relationships (assignment);
- ◆ Updating the flow-delay relationships representing vehicle interactions at modelled junctions (simulation); and
- ◆ Re-estimation of the final paths and costs for skimming (SAVEIT – if selected).

The first and third elements are multi-threaded whilst the simulation remains a sequential process. Consequently, the reductions in CPU time arising from using both network aggregation and multi-core processes do not directly translate into the same proportional reduction in total CPU time.

Figure 15.15 presents the total CPU times for the FW algorithm combined with the NA and/or Multi-core techniques using the five SATURN models. The total CPU time is normalised with respect to the standard FW algorithm.

The results show that all three techniques - using either Multi-Core and/or Network Aggregation - are at least twice as fast as the existing standard FW algorithm for all five models and, in the best case found, virtually 20 times faster.

The reductions in CPU expenditure achieved by the Multi-Core algorithm or the Network aggregation on its own are broadly comparable with CPU time reducing by a factor of 2 to 2.5 for Model 5, increasing to factors between 4 and 5 for Model 1. In most cases, aggregating the network before assignment is more efficient than distributing the original network across more than one CPU core – the exception is the very large Model 3 network.

Figure 15.15 – CPU Time by Algorithm (Normalised to Standard FW)

As expected, creating a multi-core version of the network aggregation provides a substantial “multiplicative” reduction in CPU time. In all cases, Multi-Core FW with NA reduces the CPU expenditure by factors of between 3 (Model 5) and 20 (Model 1). The overall performance gain is principally determined by the proportion of CPU expended on path building/loading relative to that spent in junction simulation and the re-estimation of paths for the final assignment. The overall reductions in CPU expenditure may be substantial – for example, for Model 3, using Multi-Core FW with NA reduces the model run time (compared to the standard FW technique) from 4.5 hours to less than 30 minutes with the same level of convergence.

Further information on the practical benefits of Network Aggregation techniques may be found in the Appendix S.

15.56.7 Other Applications of Aggregate Networks

Thus far we have concentrated on how aggregated networks may be used to reduce the cpu time required to (a) build minimum cost trees and (b) load O-D trips onto those paths. They may, however, be used effectively in several post-assignment analyses as well as other modelling issues.

15.56.7.1 Tracing Paths in Aggregate Networks

If an analysis option of min-cost O-D paths wishes to trace a path which, in the basic network, follows a link sequence A-B-C-...X-Y-Z then it requires 25 steps. If, on the other hand, the network has been aggregated such that the equivalent aggregated path is A-G-M-R-Z then only 4 steps are required: clearly potentially much faster.

For example, O-D skims of, say, times along a forest may equally be calculated on an aggregated network following the same basic procedures as for standard networks but with one preliminary step: calculate the time (or whatever quantity is to be skimmed) per aggregate link. CPU savings accrue from being able to reconstruct the minimum cost paths per iteration using the much more compact aggregate network such that similar time savings to those illustrated above for assignment should also be achieved for skimming.

See Section 15.27.7.2 for information on aggregate skims within **SATLOOK** as selected by a parameter USESPI.

Aggregated networks are also optionally used for Select Link Analysis within **P1X** - see 11.8.1.12; also controlled by a parameter USESPI..

15.56.7.2 Tracing Paths in Hybrid Networks

For certain applications it is possible to trace paths through a “hybrid network” which consist of a mixture of both aggregate and normal links. Hybrid networks were first introduced in release 11.2.3 in February 2013.

For example, if in the above example of the basic path A-B-C-....X-Y-Z one were doing a select link analysis of link K-L one could analyse a path that used the aggregate links A-G, M-R and R-Z but, for the section G-M which contains the individual link of interest K-L, one could revert to a basic network trace G-H-I-J-K-L-M. Thus the “hybrid network” is formed of aggregate links where no “events of interest” (e.g., a selected link) occur plus “normal” network links in the vicinity of “events”. In so far as the majority of links in the hybrid network are aggregate the number of steps required and hence CPU will be reduced.

The concept of a hybrid network was first used in trip matrix cordoning where the “event” that distinguishes aggregate from normal links is the crossing of a cordon link (either in-bound or out-bound). See 12.1.4, note 13). It is planned to extend the principle to other areas of post-assignment analysis such as SATPIJA and SLA.

We may also note that the concept of a hybrid network may be usefully combined with that of eliminating zero-flow spider links prior to tree building and path tracing as explained above in 15.56.5.3.

15.56.7.3 Banned and/or Penalised Turns at Buffer Nodes

Post release 11.1 it is possible to model banned and/or penalised turning movements at buffer nodes **provided that** the node in question has been aggregated (i.e., removed). These are defined within the 44444 data section using the same formats etc. as for simulation nodes. See section 6.7.

For example, if a turn A-B-C in the buffer network is to be banned then, if and when B is aggregated, the aggregate link from A to C (link A-B plus link B-C) is **not** created. If the turn is penalised then the aggregate link A-C is created but any time A-C is used during tree building then the necessary penalty is added to its cost. The same principles apply if A-C is part of longer aggregate links.

15.56.7.4 Motorway Weaves in Aggregated Networks

The necessary flow calculations required to invoke the motorway weaving rules (see 15.40.4.3) may be obtained far more efficiently using Network Aggregation if all the nodes within the motorway weaving section have been aggregated.

15.56.7.5 High-Priority Nodes for Aggregation

In the cases of both banned turns and motorway weaves in order to insure that the required nodes **are** in fact aggregated (since the final set of nodes to be aggregated is effectively arbitrary) those nodes are assigned a high priority which means that, in terms of steps 5) and 6) in the algorithm described in 15.56.3, they are preferentially aggregated at an early stage of the process, independent of the number of arms per node. Once all the "priority" nodes have been aggregated the process proceeds as described in 15.56.3.

15.56.8 Further Research

Despite the impressive reductions in cpu time achieved with the current techniques there are almost certainly further improvements possible. Thus the rules that are used to determine when a node should be aggregated – and indeed the order in which nodes are considered - are highly empirical and their efficiency is highly dependent on unknown factors, e.g., how many new links will form duplicates which may be subsequently removed. A more "intelligent" set of rules would doubtlessly lead to further improvements in cpu times.

In effect network aggregation may be thought of as a form of "pre-tree" building; in other words, before a minimum cost tree is built from a single origin, a number of minimum cost "sections" are pre-constructed from existing links which then allow the actual tree building to proceed with larger steps. Thus if the node-link sequence A-B-C-D-E-F is repeated as a minimum cost segment for multiple origins then replacing it by a single aggregate link A-F reduces CPU. On the other hand if the aggregated link A-B-C-X-Y-Z never features as part of a minimum cost path then its presence simply wastes CPU.

The "trick" therefore is to selectively aggregate "good" link sequences and to avoid the "bad"; the current rather simple-minded procedure must almost certainly be capable of improvement.

There may also well be more efficient methods for combining links together which are dependent on a particular set of link costs - as opposed to the current network aggregation procedure which aggregates links without regard to costs.

It may also be possible to eliminate a greater number of aggregated links prior to tree building than just removing the more expensive duplicates. For example, one may be able to apply a "triangle rule" which says that if nodes A, B and C form a triangle and $c(A,B) + c(B,C) < c(A,C)$ then link AC may be disregarded in terms of tree building (for that particular set of costs, not necessarily universally).

An alternative, though less rigorous, approach would be to distinguish between "probable" and "improbable" links where an improbable link (A,B) is very unlikely, given its cost and the alternatives from A to B, to be part of any min cost trees but an exact assessment might take longer than the time saved by eliminating that link. Eliminating improbable links will speed up individual Frank-Wolfe iterations

but it would be necessary, near the end of the process, to re-introduce **all** links just to confirm that none of the improbable links should be reclassified. (As long as Frank-Wolfe finds lower cost auxiliary solutions on each iteration it should not matter if it finds the **absolute** minimum cost auxiliary as long as no potential paths are being ignored in perpetuity.)

The above thoughts on eliminating certain links are based on the empirical observation that in most spider web networks less than 50% of all aggregate links are assigned flows so that, had they been eliminated a priori, the ultimate solution would be the same but achieved more quickly.

Several good topics for further research!

15.57 Residual (Incorrect) Path Flows and Restricted Frank-Wolfe Algorithms

15.57.1 Residual Path Flows: Definition

A problem in identifying path flows under the Frank-Wolfe algorithm for solving Wardrop Equilibrium (which does necessarily not apply to other algorithms such as OBA) is that of “residual paths”. A residual path is one which has been generated as a (current) best route on an early iteration of Frank-Wolfe but which, by the end of the algorithm, is very much longer than the current best route but has not been totally removed from the final averaged solution.

Thus, consider a situation where an O-D pair has two alternative routes available: a “short” route that goes through a signalised intersection and a “long” route without potential capacity restrictions. At equilibrium the signals are under capacity and incur relatively minor delays and the all-or-nothing solution with all O-D flow going through the “short” route and none on the “long” route is the correct solution for this particular O-D pair.

However, it may have happened that on an early FW iteration (most likely the second iteration following the initial all-or-nothing assignment to free-flow routes) the signals had become heavily over capacity (due to the routes chosen by alternative OD pairs which later divert elsewhere) and the minimum cost OD route for our particular O-D pair went via the long route on that particular iteration. But on all subsequent iterations the signals are never again as over-saturated and the best O-D route is always via the signals. In this case the final path flow contribution from the long route (as expressed by equation 7.2b) will never be reduced to zero unless (unlikely) a particular value of λ equals 1.0.

The creation of incorrect residual flows is an intrinsic property of the Frank-Wolfe algorithm and is one of the reasons why its convergence rate slows drastically as it approaches convergence.

Apart from slowing down convergence residual flows may also have several other undesired consequences.

15.57.2 The Importance of Residual Flows

The practical impact that residual flows may have on different forms of path analysis (see the list in 15.23.1) can be extremely variable. For example, in Select Link Analysis, if a link has a total flow of 1000.0 pcus/hr of which 0.1 is based on

residual flows then the differences between including or excluding the residual flows is arguably minimal.

On the other hand the impact of a small residual flow may be considerably amplified in certain circumstances. Consider, for example (and this is based on an example found in a real-life network), an O-D pair separated by a single link of 100 metres with signals at the downstream end such that, at equilibrium, the correct solution is an all-or-nothing flow along that link even if the signals turn out to be over capacity. (Since in that case more distant O-D pairs would have options to divert further up/downstream to avoid the congested signals but the only option for the “local” O-D pair might involve a relatively long and costly diversion.) Hence a distance skim along the used path should give 100 m for that O-D pair.

If, however, on the very **first** all-or-nothing Frank-Wolfe free-flow assignment the initial assumption is that the signals are operating under capacity then that first assignment may “optimistically” severely over-assign multiple O-D trips along that link, resulting in the downstream signals have a V/C ratio of, say, 2.0 and a queuing delay (LTP = 60) of 30 minutes. Hence an alternative route of 30 km at an average speed of 60 kph (assuming time-only assignment, PPK = 0) could be lower cost and that path would be selected on the **second** FW iteration and therefore become part of the final solution, even if its contribution may have been diluted to, say, 1% by the end of the Frank-Wolfe iterations. Thus, in terms of distance skims, that path would add $0.01 * 30,000 = 300$ m to the average distance so the skimmed distance would be 400 m, not 100.

In this case a small difference in flow has been magnified to produce a very much larger difference in outputs.

15.57.3 Frank-Wolfe Assignment with Restricted Residual Flows

There appear to be (at least) two alternative methods to minimize the impact of residual flows within Frank-Wolfe assignment:

- (1) Apply Frank-Wolfe assignment as per normal but, in any post-assignment analysis of individual O-D paths, identify any which appear to be “residual” and remove them from the analysis, or
- (2) Attempt to identify and eliminate any potentially likely residual flow paths during the tree building stages within Frank-Wolfe so that any post-assignment analyses may proceed as normal without worrying about possible residual paths.

In effect the first tries to cure the disease once it has occurred, the second tries to inoculate against it.

A semi-empirical method based on method (1) was initially introduced within **SATURN** release 10.9 and is described in the following section, 15.57.4. However, on further reflection, it seems that the method (2) is far more promising but, at present, it has only been applied in preliminary stages. See 15.57.5.

The jury is still out!

15.57.4 Removing Residual Flows Post-Assignment

SATURN release 10.9.12 introduced two (highly experimental) applications which attempt to remove residual flows if they have occurred within the assignment:

- (1) Calculating multiple commodity (i.e., times, distances and tolls) O-D skims in **SATLOOK** (SKIM_ALL, 15.27.7)
- (2) Converting a .UFC O-D route format file into a .UFO format file in **SATALL** (15.23.6 and 22.5.3)

Since any path flow along a non-minimum cost route is, strictly speaking, a residual flow it becomes important to establish a rule to identify an “important” residual flow which needs to be dealt with as opposed to the unimportant flows that may be ignored. Within **SATURN** we use two criteria:

- (i) The absolute difference AD in costs between the cost on a path c_{pij} and its minimum cost c_{ij}^* and
- (ii) The relative difference $RD = (c_{pij} - c_{ij}^*) / c_{ij}^*$

AD and RD are then compared to use-defined parameters (within &PARAM) RESIDD and RESIDR such that if a path (or a portion of a path) satisfies the condition that $AD > RESIDD$ and $RD > RESIDR$ then the path is considered to be a residual flow path.

The default values of both RESIDR and RESIDD are both 0.0 signifying that residual paths are to be ignored. Recommended values might otherwise be $RESIDR = 1.5$ and $RESIDD = 60.0$ (in units of seconds).

N.B. These two options are only available on Beta-release and will almost certainly be replaced by the use of methods to prevent residual paths occurring in the first place.

15.57.5 Avoiding Residual Flows during Frank-Wolfe Assignment

An alternative approach to dealing with residual flows **AFTER** they have been generated by an assignment is to prevent the assignment from generating them in the first place.

The basic idea is, on very early iterations of Frank-Wolfe, to use an **estimate** of what the final converged link costs are likely to be (e.g., from a “warm start” network) to build a minimum cost tree per origin based on the final costs. Then, when building the “proper” FW minimum cost trees links using the current FW costs, exclude any links which, according to the final cost tree, are clearly nowhere near minimum. The expectation is that the extra CPU involved in building two trees instead of one will be justified by the early elimination of “bad” paths and therefore not only reduce residual flows but accelerate convergence.

Improved Frank-Wolfe algorithms which incorporate the above ideas are currently being developed and tested but are not yet available to users.

15.57.6 Avoiding Residual Flows: Choice of Assignment Algorithms

Whereas residual flows may be an intrinsic problem created by the Frank-Wolfe algorithm the same problems do not occur with **all** traffic assignment algorithms. Thus they are virtually non-existent under OBA and very much less common under path-based algorithms (where social-pressure based algorithms, see Appendix H, preferentially remove residual flows).

In addition the Partan variant of Frank-Wolfe (7.11.7) tends to reduce the occurrence of residual flows due to its ability to include “backward steps” which are able to entirely remove the contributions from early iterations. It is therefore **automatically** invoked during a SUC SAVEIT assignment (15.23.4). N.B. To use Partan during the **MUC** SAVEIT assignment you must set the parameter SPARTA = T; it is not automatically invoked as with SUC. See 15.23.4.

Equally the use of incremental assignment (7.11.13) during the initial stages of an assignment should, it is hoped, discourage the build-up of residual paths since, as congestion builds up more slowly on early incremental assignments, the “correct” O-D pairs should be able to choose alternative routes avoiding locally congested links.

In particular the use of incremental assignment is (potentially) recommended to reduce residual paths during SAVEIT assignments (15.23.2). Set the Namelist parameter INKS_S = 4, say, in the network .dat file.

15.58 Error Listing (ERL) Files

15.58.1 Structure and Contents

Version 10.9.17 contains a new feature, Error Listing Files (.ERL), which provide a list of the errors reported within **SATNET** ordered by node number(s) rather than in the order in which they are detected (as they appear in the body of .LPN files) or sorted by error number (as in the .LPN summary statistics).

Thus at the end of **SATNET** a text file with the extension .ERL is created which contains one record per error detected with the following data fields:

- (i) A-node
- (ii) B-node
- (iii) C-node
- (iv) Error number
- (v) A 0/1 identifier (Extra field 1)
- (vi) A second numerical identifier (Extra field 2)
- (vii) The (short) text message associated with the error number

A sample segment of a .ERL file follows:

```
14 10 0 137 0 2 Turn saturation flows per lane differ widely
14 10 27 97 1 0 Opposing X-turns at signals hook (interfere);
27 10 0 137 1 0 Turn saturation flows per lane differ widely
0 11 0 15 0 0 Maximum roundabout turn sat flow exceeds
13 12 0 162 0 1 Multiple turns sharing multiple lanes
```

The records are sorted firstly by B-node, secondly by A-node, thirdly by C-node and finally by error number. If the error is associated purely with a node then the A- and C-node entries are zero; equally if the error is on a link then the C-node is zero while for an error associated with a turn all 3 fields are used. Errors which are **not** associated with nodes, e.g., errors in parameter inputs, do not appear in the .ERL list.

The error number uses the standard numbering system as listed in Appendix L, e.g., all Warnings are in the range 1 -99, all Serious Warnings in the range 101-199, etc. etc.

The first 0/1 extra identifier field is used, at the moment, to distinguish whether the error is new (value = 1) or whether it has occurred in a previous .ERL file, in which case it is set to zero. Thus an .ERL file for a previous run of SATNET may be defined via a Namelist parameter FILERL input under &OPTION in the network .dat file and the errors listed in the new .ERL file are compared to those in the old .ERL in order to identify an exact match.

The second numerical identifier field is also used in association with a matching entry in an input .ERL file but in this case the value is simply copied directly from the value in the old .ERL file. The thought here is that if users wish to “mark” certain error messages as being “OK”, e.g., by writing a 1 in the second field, then the new .ERL file simply carries this information over. If no match is found the second identifier defaults to 0.

Thus the intention is that users might input the .ERL file output by **SATNET** into, say, Excel, and then add their own numerical marks therein before either re-creating a new .ERL file for subsequent use by **SATNET** or inputting the new file directly into **P1X** in order to highlight certain nodes (See 15.58.2). Hence the procedure could be used as part of an “audit trail” where errors which have been checked and approved might be assigned one numerical values and errors which have not been checked could be assigned a different value.

It must be emphasised that at this stage in its development the concept of a .ERL file is still highly fluid and we are very much open to suggestions from users as to the basic format and contents of such files and equally the uses to which they might be put.

15.58.2 Display of ERL Data in P1X

ERL data may be displayed in **P1X** by “highlighting” nodes (see 11.6.5.4) based on the values in either the first or second extra identifier fields described above. The options are entered via menu choices 1st or 2nd ERL Field within the Display sub-menu.

These options differ from the “normal” highlighting procedures which highlight nodes based on **all** errors detected within **SATNET** by basing it only on errors

which have been included within the .ERL file but with extra tests based on values stored in the first and/or second “extra” data fields.

In both cases a “critical” value needs to be defined by the user but its application differs between whether data from Field 1 or 2 is to be used. Thus with Field 1 the test is based on equality; i.e., if you set a critical value of 1 only those error records which have a 1 in Field 1 will be selected. Under 2 the test is “greater than or equal”; i.e., all entries whose Field 2 value \geq the critical value are selected.

In addition the second field differs in that the colour used to highlight the selected nodes depend upon the **value** in the second field. Thus a low value might be displayed with a light colour and progressively higher values with progressively darker colours in order to indicate possible degree of urgency as set by the user. The pens to be used for different numerical values are pre-defined within the program but may be over-written using parameters NP_ERL(n) within the (most recent) preferences file P1X0.DAT.

We repeat the information given above that, at the moment, Field 1 is set as either 0 or 1 within **SATNET** depending on whether an error is “old” or “new”, whereas Field 2 is intended to be manipulated externally by the user via, say, Excel, prior to its use in **P1X**.

15.59 Disaggregate Network Summary Statistics

15.59.1 General Principles

Network statistics such as total PCU-hrs, total PCU-kms etc. are automatically calculated over all links by **SATALL** (and **SATSIM**) with a split between, e.g., simulation links, buffer links, etc. as illustrated by the tables in Sections 17.8 and 17.9. In addition to total flows, flows are always disaggregated into the following categories (if they exist): (1) bus flows, (2) pre-loaded flows, (3) PASSQ flows, (4) all user class flows, and (5) flows exclusively from the trip matrix. Therefore the standard output statistics always include the total PCU-kms by pre-loaded flows or by user class 3, etc. etc.

However it is also possible to optionally obtain a further disaggregation of the same statistics by sub-sets of links and/or by sub-sets of flows, either calculated within **SATALL/SATSIM** or afterwards using **SATLOOK**.

Note that the sub-sets of flows are effectively fixed and used in each level of link disaggregation. Thus the main choices to be made by the user are how to define the disaggregation of links.

Traditionally links were disaggregated into sub-sets according to their capacity indices but, post 11.2.8, it is possible to define a much wider range of criteria to set link sub-sets. For example, links may be grouped into self-contained sectors or “traffic boroughs” (see 5.1.7.1 and 5.1.7.2) and statistics such as total PCU-kms by user class 3 produced per sector or borough.

Indeed these more general link criteria now take precedence over disaggregation by capacity index since capacity indices are generally aimed primarily at setting link speed-flow curves, from which a disaggregation of, say, PCU-kms may not be particularly useful.

15.59.2 Disaggregation within SATALL

At the end of each run of **SATALL** (or **SATSIM**) a complete set of **total** network statistics are calculated and stored within the output .UFS file. In addition, optionally, a further set of **disaggregate** statistics is calculated and stored in .UFS as controlled by user-set Namelist parameters in the network .dat files.

Thus if BYGRUP = T, statistics are calculated by link “groups” (see 5.1.7.3) where the groups are defined either as (a) traffic boroughs if a parameter TFL = T or, if TFL = F, (b) by a “N2G” file set as FILN2G. N2G files are specified further in Section 15.60.2; basically they define an “index” for each node such that links are grouped according to the node index of their B-node. (Which is effectively the way in which links are grouped into traffic boroughs where the “name” of a link’s B-node defines its borough number following TFL rules - see 5.1.7.2.)

Finally if BYGRUP = T, TFL = F, but **no** N2G filename has been set (FILN2G is blank) then **no** disaggregate statistics are calculated within **SATALL**. When this happens a warning message is generated.

If BYGRUP = F then the disaggregation is (potentially) set by the link capacity indices but only if (a) a further namelist parameter BYCAPI = T and (b) capacity indices exist in the network. Since, as explained above, capacity indices may not be all that useful for disaggregate statistics BYCAPI defaults to F.

The disaggregate statistics calculated by **SATALL** and stored within the .UFS file may be viewed only within **SATLOOK** – option 4, then 1 from the main menu; see Section 11.11.4.

15.59.3 Disaggregation within SATLOOK

As mentioned above disaggregate network statistics as (optionally) calculated within **SATALL** may only be accessed using **SATLOOK** (either the standalone version or called from **P1X**). However, it is also possible to calculate similar statistics “on the fly” within **SATLOOK** using not only those criteria available within **SATALL** (e.g., disaggregation into traffic boroughs) but also by a much wider range of possible disaggregation rules – main menu option 4 followed by option 2.

Thus the most basic level of link disaggregation is set by the parameters BYGRUP, TFL and BYCAPI; if any of these three is “toggled” interactively then the link sub-sets will be re-calculated and the disaggregate statistics will be re-calculated and output. In addition the .N2G file which would have been initially set as a network .dat file parameter may be re-defined interactively and the disaggregate data re-calculated

In addition the link selection rules as set within **SATDB** may also be applied “on top” of the normal link disaggregation rules – but only if **SATLOOK** is being accessed via **P1X**.

A further **P1X**-only option allows the “indices” which define sub-sets of links to be set via an existing integer data base column. Alternatively the link indices may be input directly from a “.L2G” text file which gives the required index for each link; see 15.60.4 for formatting rules.

15.60 Node and/or Zone Aggregation Files

15.60.1 General Principles and File Extensions

A set of filename conventions has been drawn up in order to identify ASCII text files which define the “mapping” of one set of node/zone definitions into another. Thus a file with extension .Z2G will contain data which specifies which Groups are to be associated with each Zone, .Z2S maps Zones into Sectors, N2G maps nodes into groups, G2S maps groups into sectors, etc. etc.

The following letters may be used: N for nodes, Z for zones, D for districts, B for boroughs and S for sectors. In addition T represents “text” so that a .G2T file would consist of a series of group names followed by a text description of that group; e.g., “1 Otley”.

As a matter of good practice and common sense it is proposed – although this is not a rigid requirement in **SATURN** – that all *2* (N2G, G2S, etc.) files should (a) have the same “root” filename and (b) be stored in the same folder. In other words, files such as mapping.z2g, mapping.z2s, mapping.g2s, etc. would all be stored in the same folder and therefore have a common root pathname as well as a common filename. The advantage of this is that the user need not define all the possible mapping files since a **SATURN** program could logically infer a file/path name when necessary.

In particular this facility is routinely used with text descriptor files of the form .G2T whereby if the predicted file mapping.g2t can be found it is used to add text names to groups; if not group text names are simply ignored.

15.60.2 FILZ2* – Zone Aggregation (.Z2G)

All files which map zones into more aggregate structures such as groups, sectors, etc. have the same general, very simple format described as follows:

They consist of a series of text records (terminated by a 99999 record) where each record consists of two integers in free format (i.e., including CSV) specifying a zone followed by its group (where we use the terms “zone” and “group” to denote the first and second quantities as in a Z2G file but the same specifications apply equally to all such files)..

Note that numerical “names” must always be used for both the zone and the group - **not** sequential numbers (although very often zone and/or group names are in fact sequential).

Records need not be in numerical order of zones, i.e., the first number given is always increasing, although this is generally the most convenient way to create such files.

Duplication (i.e., assigning the same zone to two different groups) is not allowed (although it may not always be checked).

A hyphen in front of a zone name (negative numbers) may be used to indicate a “range” of zones. Thus two successive records:

```
9      1
-19    2
```

would indicate that all zone names in the range 10 through 19 would be assigned to group 2 (and that zone 9 would be in group 1).

Note that 19 need not necessarily be a valid zone name itself, it simply represents an upper limit, in which case the “true” upper limit would be the maximum zone name lower than 19. The lower value of the range is the previous upper limit plus one. If a negative number is used to indicate an interval the absolute value of the negative number must be greater than the absolute of the previous number in the list. If as above a positive number is used (e.g., 9) to set the previous line that zone name **must** exist.

Therefore it is recommended that you use either all intervals (negative numbers) or include all zone names in the Z2* file using the philosophy that the point of using intervals is for the process not to fail and the point of using a zone by zone list is that you want the process to warn you about missing elements by failing.

Errors occur and are noted if a record does not consist of two integers, if a zone cannot be identified (excluding negative values above) and if some zones are not assigned to groups. These may or may not result in the operation being rejected.

Blank records are allowed and ignored as our comments, i.e., records with a * in column 1.

In order to process a Z2G file the zone names (and their number) must already be known but the set of group names and their total number are only known and fully specified **after** the Z2G file has been processed.

Note that the Z2G format also corresponds to a simplified version of the Records 2 used by the batch file MXM5; see Appendix W.3.

15.60.3 FILN2* - Node Aggregation (.N2G)

Files which map nodes into more aggregate groups follow the same specifications as for zonal aggregation files as described in 15.60.2, two integer values in free format – with the obvious caveat that the first integer value per record is a node number, not a zone number.

The use of negative node numbers to indicate ranges is also allowed.

15.60.4 FILL2* - Link Aggregation (.L2G)

Links may be directly mapped into groups of links (as opposed to using their B-node to define the mapping) via an “L2G” etc. file where the default file extension .L2G signifies a file which gives the mappings of links into “groups”.

L2G files contain 3 free-format integer values per record, the first two being the link A-node and B-node and the third being the group. The use of negative node numbers to indicate “ranges” is **not** permitted with L2G files.

Currently L2G files are only processed within **SATLOOK**; i.e., it is not possible to define an L2G file as a namelist parameter in a network .dat file and have the appropriate aggregation statistics calculated within **SATALL** and stored on .UFS files in the same way that node-based aggregate statistics may be set.




15.60.5 FIL*2T – Text Definitions

Files with extensions of the form Z2T, N2T, etc. etc. are used to supply alpha-numerical titles to zones, nodes as indicated by the first letter in the extension.

They are not, however, generally available to users at present.

15.61 Version Control

JOB NUMBER: 5120257		DOCUMENT REF: Section 15.doc				
Revision	Purpose / Description					
		Originated	Checked	Reviewed	Authorised	Date
10.9.10	SATURN v10.9 Release	DVV	DG	IW	IW	04/09/09
10.9.12	SATURN v10.9 Release (Full)	DVV	DG	IW	IW	31/10/09
10.9.17	Web release – Jun 10	DVV	NP	IW	IW	22/06/10
10.9.22	Web release – Dec 10	DVV	AG	IW	IW	06/12/10
10.9.24	SATURN v10.9 Release (Full)	DVV	AG	IW	IW	31/05/11
11.1.09	SATURN v11.1 Release (Full)	DVV	AG	IW	IW	31/03/12
11.2.01	SATURN v11.2 Beta Release	DVV	JS	IW	IW	07/12/12
11.2.05	SATURN v11.2 Release (Full)	DVV	JS	IW	IW	17/03/13
11.3.03	SATURN v11.3 Release	DVV	EN	IW	IW	30/04/14
11.3.07	SATURN v11.3.07 Release	DVV	DAS	EN	IW	26/09/14
11.3.10	SATURN v11.3.10 Release	DVV	DAS	IW	IW	22/01/15
11.3.12	SATURN v11.3.12 Release	DVV	DAS	IW	IW	22/04/15