
Program
documentation
for the logistics
model for
Sweden

MICHIEL DE BOK
JAAP BAAK
GERARD DE JONG

March 2015
For Trafikverket

Preface

A logistics module has been developed for both the Swedish national freight model system (Samgods model) and its Norwegian counterpart NEMO. This logistics module introduces logistic elements to the freight model systems, such as the determination of shipment size and the use of consolidation and distribution centres. Just recently a version 2.1 of the logistics model has been documented in the form of a technical report (Significance, 2013), building upon previous technical reports.

This technical document is written for Trafikverket in order to help users to perform analysis with the logistic module. It contains a technical description of the program. The focus is on the logistics model for Sweden. The documentation is not written for programmers in the first place, but mainly for users of the model.

For more information about Significance or this document, please contact Gerard de Jong at:

Significance
Koninginnegracht 23
2514 AB Den Haag
Netherlands
+31-70-3121533
dejong@significance.nl

Contents

Preface.....	iii
CHAPTER 1 Introduction	1
1.1 Introduction	1
1.2 The logistics module	1
1.3 Overview technical documentation	5
CHAPTER 2 User manual.....	7
2.1 Introduction	7
2.2 Installing the logistics module	7
2.3 Running the logistics module.....	9
2.4 Changing input data to implement scenarios.....	10
CHAPTER 3 Program structure.....	15
3.1 Introduction	15
3.2 Program structure	15
3.3 Controlling the iterative procedure.....	19
3.4 input and output to the model.....	20
CHAPTER 4 BuildChain procedure	23
4.1 Introduction	23
4.2 Input files.....	23
4.3 Output files.....	25
4.4 Description	26
4.5 Controlling the BuildChain procedure.....	31
CHAPTER 5 ChainChoice procedure	41
5.1 Introduction	41
5.2 Input files.....	41
5.3 Output files.....	43
5.4 Description	45
5.5 Controlling the ChainChoice procedure	47
CHAPTER 6 Extract procedure	51

6.1	Introduction.....	51
6.2	Input files.....	51
6.3	Output files.....	52
6.4	Controlling the Extract procedure.....	52
References.....		57
Appendix A: Rail Capacity Management.....		58
A.1	Introduction.....	58
A.2	LogMod4RCM.....	58
A.3	LP2CC 60	
Appendix B: Dimensions in the model		62
Appendix C: Parameters and variables		68

1.1 **Introduction**

This document contains a technical program documentation of the logistics model, as developed for Sweden. This chapter gives an overview of the logistics model, version 2.1. The structure of the model is discussed, the assumptions behind the model and the input and output information is given. This information is useful for those wishing to use the logistics model that is part of the Swedish national freight model system.

1.2 **The logistics module**

The logistics model is developed for the Work Group for transport analysis in the Norwegian national transport plan and Trafikverket in Sweden. Actually there are two logistics models with the same structure, one for Norway and one for Sweden. As most freight transport models, the existing Swedish national freight model system (Samgods model) and its Norwegian counterpart (NEMO) are lacking treatment of logistics choices, such as the determination of shipment size, consolidation and distribution. Both in Norway and Sweden a process to update and improve the existing national freight model system is going on. An important part of this is the development of the logistics model.

In 2005/2006, a prototype version of the logistics model was developed, that uses Swedish and Norwegian network and cost data and data on the locations of terminals and distribution centres (at these “nodes” there are transfers between different types of vehicles/vessels and/or between different sizes of vehicles/vessels). The parameters of this prototype were not calibrated to observed data. The purpose of this version was mainly to show the feasibility of the approach.

After having shown that the approach was feasible, a new and extended version of the logistics model has been specified and applied for both Norway and Sweden within the framework of their national freight transport forecasting systems.

The basic mechanism in the transport chain choice is minimisation of a deterministic total annual logistics cost function. Estimation of a random utility-based logistics model on disaggregate data (partly available, partly still to be collected) is foreseen for future years for both countries. We have developed a procedure to calibrate parameters in the cost function to available aggregate data (observed origin-destination flows by mode and commodity type for aggregate zones).

The structure of the national freight model system (either for Sweden or Norway) as a whole is given in Figure 1. The logistics model is only part of this. For Sweden the logistics model consists of all the blue boxes in the figure. For Norway, the logistics model is defined as the blue and yellow boxes.

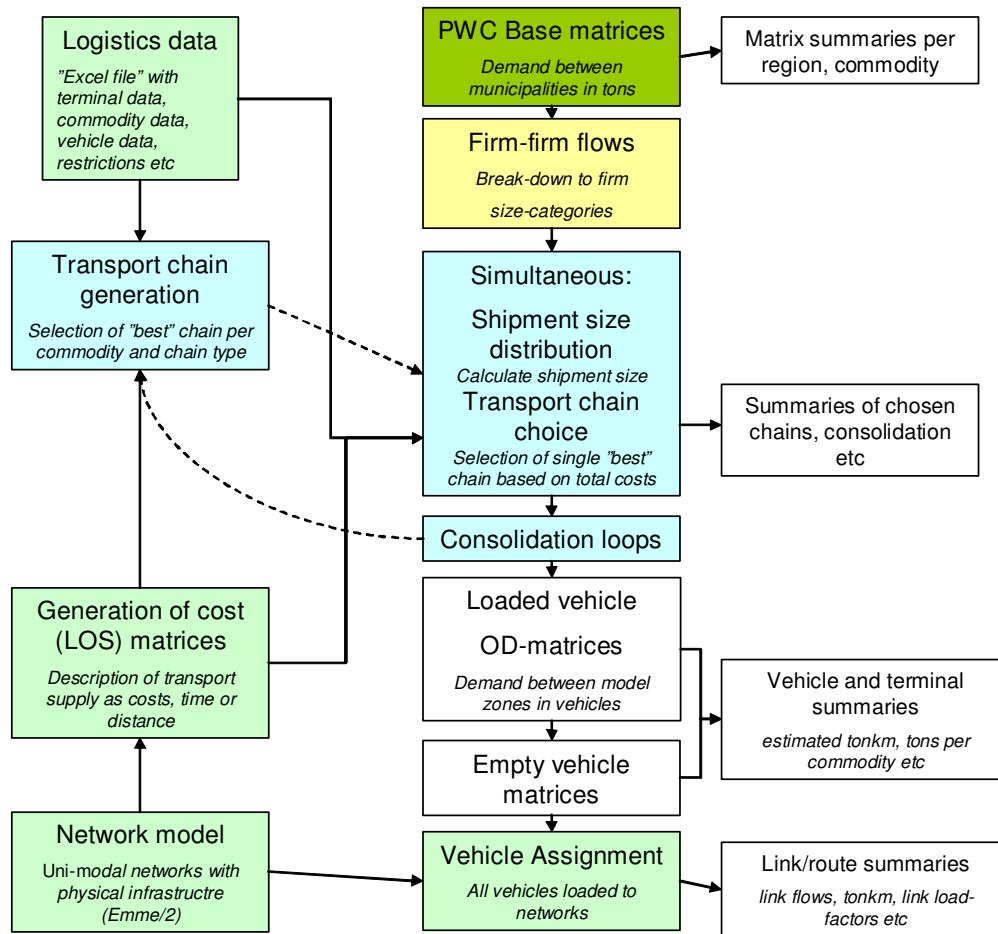


Figure 1-1: Structure of the national freight transport model system with the logistics model (source: John McDaniel, VTI)

Inputs to the logistics model (green, red)

For both countries, the logistics model uses inputs on transport costs, based on time and distance calculated on transport networks, using a network model. Other inputs for the logistics model consider the locations and other attributes (e.g. which commodities can be handled, terminal time and cost) of terminals, commodity-specific costs (e.g. order and storage costs), vehicle data (e.g. capacity and which vehicle types can be used for which commodities) and restrictions (e.g. on draught in ports). These input data are used together with the transport costs to calculate the total logistics costs. The final type of

input of the logistics model refers to the PWC (production-wholesale-consumption) matrices. These are commodity flows from the production zone to the consumption zone (PC flows), also containing flows from production to wholesale and from wholesale to consumption. For Norway, the logistics model takes the zone-to-zone (z2z) PWC flows (base matrices) and in a first step allocates these to firm-to-firm (f2f) flows, which are flows from the sending firm to the receiving firm. These flows may consist of several origin-destination (OD) flows, because a transport chain may be used with multiple modes and/or vehicle types and one or more transshipments along the route. In Sweden there is an external program to generate the f2f flows, which is the last step of the base matrix calculations (Edwards, 2007).

Components of the logistics model (blue, yellow)

The Norwegian model takes as inputs commodity flows from production to consumption zone (that also include the wholesale function). The logistics model then disaggregates these PWC flows to f2f flows. For Sweden the disaggregation to f2f flows is done as part of the base matrix calculation, outside the logistics model, and the logistics model takes these as given.

After this disaggregation, for both countries, the logistics decisions (shipment size, use of consolidation and distribution centres, mode and vehicle/vessel type and loading unit type choice) are simulated at this f2f level (micro-simulation).

In the logistics model there is a choice between a number of transport chains (with one to four legs and with different aggregate modes and different vehicle/vessel types for each leg). The transfer locations between the legs are determined in a separate step: the transport chain generation program (on the left hand side in Figure 1). The transfers can take place within the road system (e.g. from van to truck), or between road, sea, rail and air transport (and the Swedish model also includes transfers within the sea and within the rail system: feeder and main haul transport). The transport chain generation program produces the choice set of available transport chains, but already optimises the locations of transfers within each available type of transport chain.

Then the logistics costs optimisation in the main program (labelled 'simultaneous shipment size distribution and transport ChainChoice') will determine the optimal shipment size, together with the transport chain choice from the choice set of available chains.

We define consolidation centres as locations where goods are transhipped (and possibly stored, but this is not modelled), with small loads getting in and larger loads getting out. Distribution centres are locations where goods are transhipped (and possibly stored), with large loads getting in and small loads getting out. Both consolidation and distribution centres exist not only in road transport, but can also be ports, airports or rail terminals. Each terminal can serve as both consolidation and distribution centre. Whether a shipment can be consolidated with other shipments depends on the amount of cargo shipped from this consolidation centre to the same destination zone as the shipment under consideration. This implies that consolidation is a function of the OD flows, which are endogenous in the logistics model. We solve this endogeneity problem, and determine the

degree of consolidation (or the load factor of the vehicles) between consolidation centres and distribution centres, by using the logistics model in an iterative procedure ('consolidation loops' in Figure 1). This starts with an assumed average consolidation factor for each link, but in a subsequent iteration includes information on the availability of other cargo (based on the available transport chains and port statistics), and in an even further iteration uses the OD flows between consolidation centres predicted in the previous model iteration. The predicted consolidation factor for each link is included in the output of the `ChainChoi` procedure, see section 4.3. Consolidation only takes place within a commodity group. The choice of starting factor has a small but not negligible influence on the final results. To solve this, one could do more than three iterations.

Consolidation in the model can take place at all transfer nodes (terminals, collection/distribution centres) that are included in the model system and along the route ("milk round" or "collection round": a road vehicle goes to several senders in the same zone to collect loads before it goes to the receivers).

Outputs of the logistics model (white)

The output of the model consists of flows between origins and destinations (OD-level), where consolidation and distribution centres (including ports, airports and railway terminals) are also treated as origins and destinations. Furthermore, the model can provide information on total logistics cost (by costs item) between zones, which can be used in trade models or spatial interaction models. The core output of the logistics model is highly disaggregate: at the level of individual f2f flows it gives the shipment size and shipment frequency, the chosen transport chains (number of OD legs, vehicle/vessel type on each leg, transshipment locations) and the logistics costs (by cost item).

Once we have the outputs at the level of the individual f2f flows, these can be aggregated in different ways. The logistics model includes the possibility to produce OD matrices of tonnes and tonne-kilometres by mode and commodity type (but also more aggregate – non-OD- statistics on the total number of tonnes and tonne-kilometres by mode and commodity type), as well as OD matrices of loaded vehicles by vehicle/vessel type, and of loaded and empty road vehicles. These can be used in assignment to the networks, applying the same network model as for the transport cost inputs. Also one can obtain output flows by vehicle type or by terminal. The procedures to calculate these outputs have remained the same as versions 0 (see RAND Europe and SITMA, 2005). This also goes for the calculation of empty vehicles flows. Furthermore, total logistics costs per PWC flow can be produced, for use in the future determination of PWC flows.

Testing of the program

Tests of the program have been carried out by the client groups as well as Significance, looking at the overall mode split, the number of f2f relations and the number of shipments per commodity type, the OD flows and the outcomes for specific PWC relations. The outcomes have been checked against observed data and/or for consistency/plausibility.

1.3 **Overview technical documentation**

This document contains a technical documentation on the implementation of the logistics module. Chapter 2 gives a brief description on how the model can be run on a desktop computer. Chapter 3 explains the structure of the model, how the main run parameters of the program can be set and how the structure can be modified. Chapter 4 describes the BuildChain procedure. An overview is presented of the input and output files, and it is discussed how alternative options are set for this procedure and how this influences the results. Chapter 5 describes the ChainChoice procedure. An overview is presented of the input and output files, and it is discussed how alternative options are set for this procedure and how this influences the results.

2.1 **Introduction**

This chapter describes how the program files and input and output files are set on a desktop computer in order to make calculations with the logistics module. Furthermore it is outlined how the model is run. For a detailed description of options to define scenarios or modify model settings, see the more elaborate descriptions on the model structure and individual components in Chapters 3 to 5.

2.2 **Installing the logistics module**

The logistics module can be run on a desktop computer without any additional software installed. As an indication for the required hardware, on a desktop computer with a 3.4 GHz Core i7 processor and 8 GB of internal memory, the run time of the Swedish logistic model was 5 hours, but this is greatly influenced by the settings in the model. For instance, the Norwegian model was considerably faster, about 2-3 hours, since it includes much less chain types compared to the Swedish model.

It is required to have enough free space available on the hard disk of the computer. The program and input files require 410 MB of free space. To apply the model, each completed simulation requires at least 5,8 GB of free space.

The programs and files that are part of the logistics module can be copied to any directory that is designated to perform model runs. For instance: D:\SIKA\. If all programs and files are copied to this folder correctly, it should have the following files and subfolders available in the structure that is shown below.

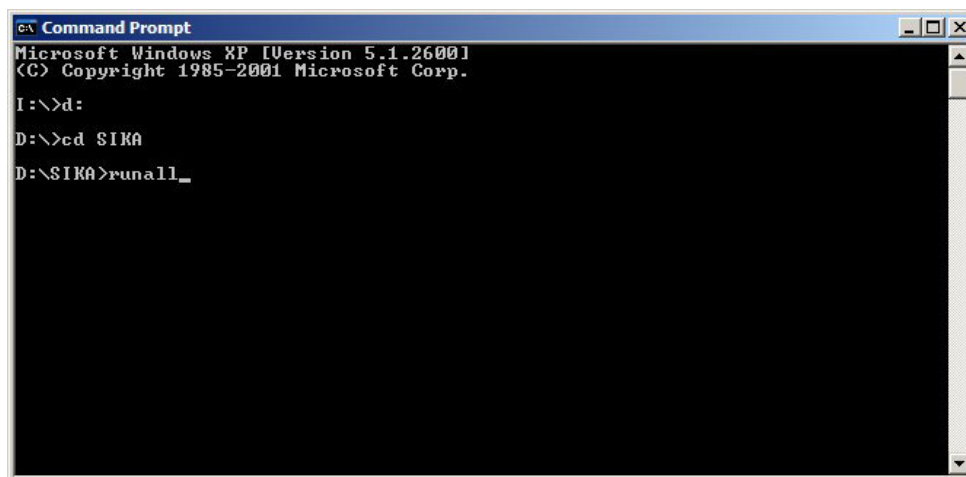
Table 2-1: Overview of the folder structure and main program

Files and folders	Description
\LOG	Contains the log files per commodity for a whole model run (all iterations of BUILDCHAIN and CHAINCHOI)
\BUILDCHAIN	produces a set of potential chains. Folder contains the BUILDCHAIN program file and all required control files.
\BUILDCHAIN\Output	Contains the output files of the BuildChain module.
\CHAINCHOI	determines the single chosen ‘best’ chain. Folder contains the CHAINCHOI program file and all required control files.
\CHAINCHOI\Output	Contains the output files of the ChainChoi module.
\CHAINCHOI\Output\Consol	Contains the consolidation output files of the ChainChoi module.
\EXTRACT	generates vehicle demand matrices. Folder contains the EXTRACT program file and all required control files.
\EXTRACT\output	Contains the output files of the Extract module.
\INPUT	Contains all input files. These input files are stored in the following 4 subfolders:
\INPUT\COST	contains input files with general control data parameters
\INPUT\LOS	contains input files with network link cost matrices. These files are delivered by Samgods from EMME/2
\INPUT\NODES	contains input files with a list of nodes and characteristics of each node, including transfer restrictions
\INPUT\PWC	contains demand matrices in tonnes between zones
commodity.bat	batch file to run all submodules for one specific commodity
runall.bat	batch file to make a model run for all commodities

2.3 Running the logistics module

To make a full run

The logistics module is developed as a console application that is run by starting up the batch job `runall.bat`. This batch job can be run through Windows Explorer, or on a command line. To run the model in Windows explorer, double click the file `runall.bat`. To run the model from the command line, open a console and navigate to the directory where the logistics module is stored, for instance: `D:\SIKA`. On the command line type `'runall'` and press `<enter>`. Next the logistics module is executed in the console, as visualised in Figure 2-1.



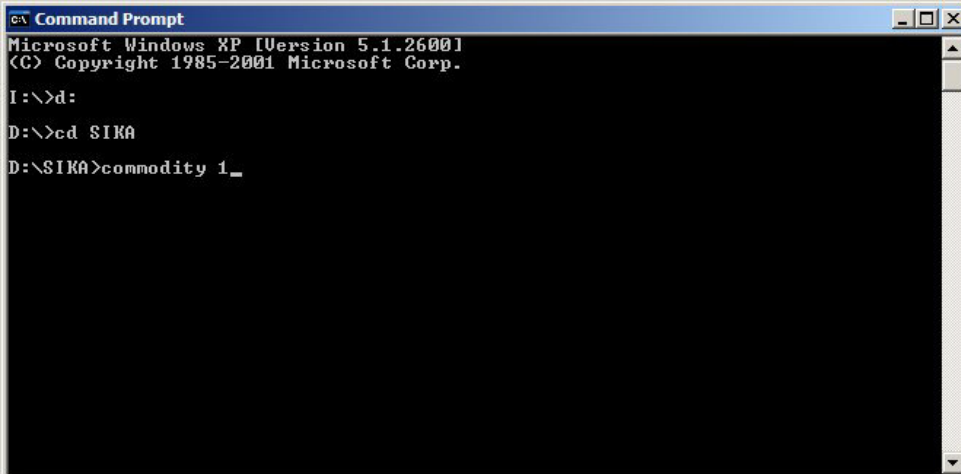
```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

I:\>d:
D:\>cd SIKA
D:\SIKA>runall_
```

Figure 2-1: Running the logistics module from the command line.

To make a run for one commodity

The logistic module can be run for an individual commodity type as well. To run the model for a specific commodity, open a console and navigate to the directory where the logistics module is stored, for instance: `D:\SIKA`. On the command line type `'commodity'` with an additional argument for the commodity type, and press `<enter>`. See where the model is run for commodity type 1 only. See Table A-1 in the appendix for an overview of commodity types.



```

Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

I:\>d:
D:\>cd SIKa
D:\SIKa>commodity 1_

```

Figure 2-2: Running the logistics module for commodity type '1' from the command line.

Controlling and analysing runs

The model settings can be modified through the input files in the `\INPUT` directory, or through the control files that are used to run subprograms. The following section will discuss some typical examples of changing the input data for a model run to test policy options. A detailed description of all input files, how these files can be modified, and what effects can be expected from these changes, is provided in Chapter 4 and Chapter 5 that describe the `BuildChain` and `ChainChoice` procedures.

After a model run, all required output files need still to be extracted from all calculation results with the `EXTRACT` program. To generate the required output files, the batch job `extractall.bat` should be run.

2.4 Changing input data to implement scenarios

Policy options can be analysed by changing the input data and running the model. Here five examples are given of the implementation of typical policy options.

Adding direct access from a node

The nodes in the logistic model consists of zones and terminals. Changing the direct access settings consists of two steps: changing the nodes file, and checking and changing the direct access file(s). Both steps vary whether the terminal already provides direct access to another zone or if it is new in providing direct access. The two steps are explained in more detail below.

First, add direct access from a zone i to a terminal t :

1. Open the nodes file¹ (`\INPUT\NODES\NODES.TXT`)

¹ For a description of the node file see Table 4-3

2. Insert the zone number i in the 3rd column of the record for terminal t .²
3. Close input file and save changes.

Second, check and/or modify the direct access settings in the direct access files (by mode or commodity group):

4. Depending for which transport mode direct access is provided, open the relevant direct access file³. For instance for direct access for sea: `\INPUT\NODES\DIRECTSEA.TXT`⁴
5. Identify the terminal by its node number in the first column of the file.
6. If it already exists: check if the direct access for each commodity group are set correctly.
7. If the terminal doesn't exist in the direct access file:
 - add node code and name of terminal to first two columns.
 - place commodity numbers in the following columns if direct access is provided for that commodity. For instance: to add direct access for commodity 2, add value '2' to the fourth column.
8. Close input file and save changes.
9. Repeat steps 4 to 8 for each relevant transport mode.

Add a terminal

Adding a terminal involves the introduction of a new node to the network. In addition to changing the node files, all network matrices need to be updated too. To add a terminal to the model the following steps should be followed:

1. Open the nodes file (`\INPUT\NODES\NODES.TXT`)⁵
2. Add a new node record to the file under the node corresponding to the zone in which it is located, preserving the sorted order of the nodes. Insert the new node number in the first column⁶.
3. Place a zone name in column 2, starting with the mode type of the terminal, e.g.: "Road: ...".

² Please note that multiple terminals can provide direct access for the same zone. In these cases, the node number of the zone returns more than twice in the 3rd column in the nodes file.

³ For an overview of all direct access files see section 4.2

⁴ It is suggested to open the file in excel so that the data is put into columns according with the format. The tab delimited file format should be maintained when saving the file.

⁵ It is suggested to open the file in excel so that the data is put into columns according with the format. The tab delimited file format should be maintained when saving the file

⁶ The new node number need to correspond to the node numbers in the EMME/2 network.

4. If the terminal provides direct access to existing zones, place the node number of the zone that has direct access to this terminal in column 3.
5. Fill in the attributes for the respective node. For a description of the node attributes see Table 4-3.
6. Close input file and save changes.
7. *EXTERNAL TASK*: Recalculate EMME matrices for network with new node. For an overview of the LOS matrices for distance, domestic distance, time and extra costs by vehicle type see Section 4.2.
8. Update all LOS matrices (in the folder: `\INPUT\LOS\`).

Add a zone

Adding a zone is very much similar to adding a terminal. It involves the introduction of a new node to the network, and changing the node files and all network matrices. To add a zone to the model the following steps should be followed:

1. Open the nodes file (`\INPUT\NODES\NODES.TXT`).
2. Add a new record to the file preserving the sorted order of the nodes. Insert the new node number in the first and the third column.
3. Place a zone name in column 2, starting with: "Zone: ...".
4. Fill in the attributes for the respective node. For a description of the node attributes see Table 4-3.
5. Close input file and save changes.
6. *EXTERNAL TASK*: Recalculate EMME matrices for network with new node. For an overview of the LOS matrices for distance, domestic distance, time and extra costs by vehicle type see Section 4.2.
7. Update all LOS matrices (in the folder: `\INPUT\LOS\`).

Adding a new logistics chain

A new logistic chain can be added by changing the lists of all logistic chains. The following steps can be followed:

1. Open the chain type list (`\INPUT\CHAIINTYPE.LIS`)
2. Insert the character combination for the new chain type.
3. Close input file and save changes.

Chain types with either a roro connection at the begin or end of the chain, or a roro connection with different transport modes on either side, are not accepted by the model.

Adding a vehicle type within a mode

The settings for vehicle type and mode combinations are controlled in three different sets of files: the vehicle data files, and the control files for the `BuildChain` and `ChainChoice` modules. The following steps can be followed in vehicle type v need to be added to mode m :

1. Check if the vehicle type is defined in the cargo specific parameter files
(`\INPUT\COST\VHCLS_GEN_CARGO.TXT`, `VHCLS_DRY_BULK.TXT`
`VHCLS_LIQ_BULK.TXT`)
2. Add the vehicle type number for v , to the list after mode parameter `VHCLM` for mode m to the controlfile `BuildChain%1%.ctl` for commodity group `%I%`.
3. Close control file and save changes.
4. Repeat step two and three for each commodity group that has an additional vehicle type available within a mode.
5. Add the vehicle type number for v , to the list after mode parameter `VHCLM` for mode m to the controlfile `ChainChoi%1%.ctl` for commodity group `%I%`.
6. Close control file and save changes.
7. Repeat step five and six for each commodity group that has an additional vehicle type available within a mode.

3.1 **Introduction**

This chapter will first describe the structure of the model and how this structure is implemented in a program. It is explained what sub-programs the program calls upon (`BuildChain` and `ChainChoice`) and what type of input is required for each component. Next it is explained what run-parameters can be controlled in the main program, and how the iterative procedure can be modified. Appendix B gives an overview of all parameters and variables in the model. This overview describes the parameter or variable, it specifies in which file it is stored, by which module it is used, and the type and range of information (if applicable).

3.2 **Program structure**

The logistics decisions (shipment size, use of consolidation and distribution centres, mode and vehicle/vessel type and loading unit type choice) are simulated at the firm to firm (f2f) level (micro-simulation). The Swedish logistics module uses firm to firm (f2f) commodity flows as input. The initial PWC commodity flows (from production to consumption zone that also include the wholesale function) are disaggregated to f2f flows as part of the base matrix calculation outside the logistics model.

To perform a run with the logistic model, two batch jobs are used. First, `RUNALL.BAT` is used to run the logistic modules for each commodity successively (for the content of this file see Figure 3-1). For Sweden, 35 commodity types are distinguished. An overview of these commodity types is given in the appendix.

Figure 3-1: Contents of [Runall.bat] ⁷.

```
call commodity 1
call commodity 2
call commodity 3
call commodity 4
call commodity 5
call commodity 6
call commodity 7
call commodity 8
call commodity 9
call commodity 10
call commodity 11
call commodity 12
call commodity 13
call commodity 14
call commodity 15
call commodity 16
call commodity 17
call commodity 18
call commodity 19
call commodity 20
call commodity 21
call commodity 22
call commodity 23
call commodity 24
call commodity 25
call commodity 26
call commodity 27
call commodity 28
call commodity 29
rem call commodity 30
call commodity 31
call commodity 32
call commodity 33
call commodity 34
call commodity 35
```

Next, the main procedure `RUNALL.BAT` calls upon the batch job `COMMODITY.BAT` for each commodity group (for the content of this file see Figure 3-3). This logistic model calls upon two logistic components in an iterative procedure, as visualised in Figure 3-2. These two procedures are:

- `BuildChain`: selects the “best” chain per commodity and chain type
- `ChainChoice`: calculates shipment size and selects single ‘best’ chain based on total costs.

Each component will be discussed in detail in Chapter 4 and 5. The function of the modules and how they are used by the logistics program in an iterative procedure is

⁷ Commodity type 30 (mixed and part loads, miscellaneous articles) is not yet implemented in the model, because the input data for this commodity type was not available yet. For this reason, the call upon the batch job for this commodity is inactivated through the ‘rem’ statement.

outlined in Figure 3-2. This iterative procedure is discussed in more detail in section 3.3, but first the function of BuildChain and ChainChoice modules is explained.

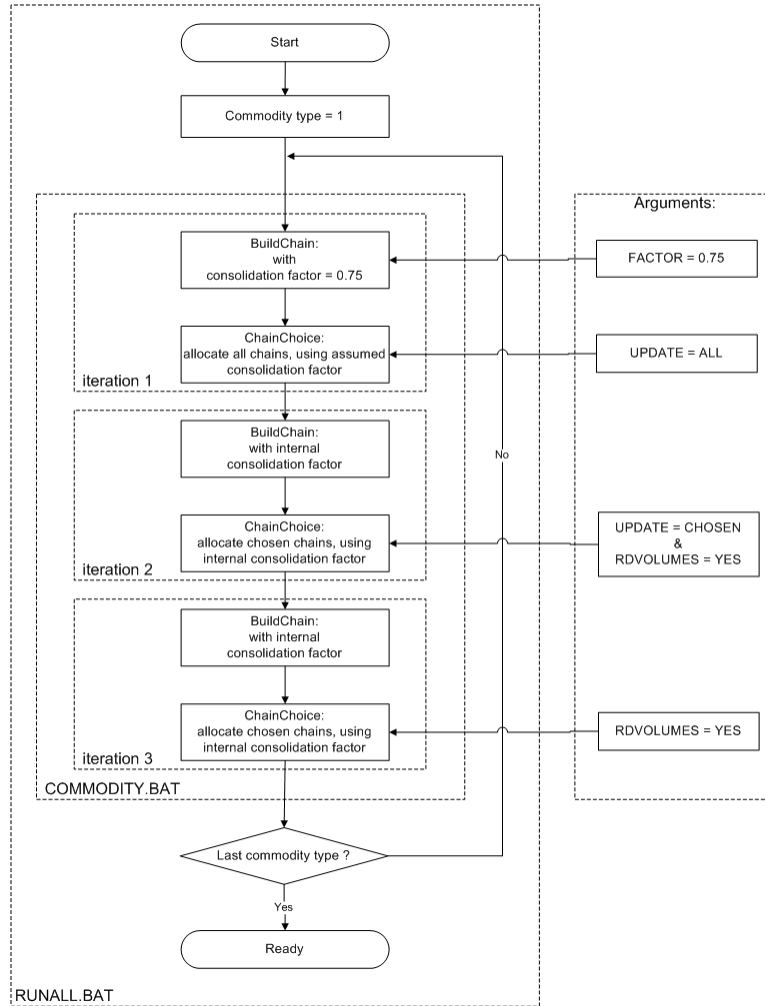


Figure 3-2: Structure of the logistic program

The BuildChain procedure

The BuildChain module selects the “best” chain per commodity and chain type for each origin – destination pair. The module first simulates the choice of the optimal transfer locations for a number of transport chains (with one to four legs and with different aggregate modes for each leg). Transfers can take place within the road system (e.g. from van to truck), or between road, sea, rail and air transport. The Swedish model also includes transfers within the sea and within the rail system: feeder and main haul transport. Hence,

the sea and rail legs can be composed of multiple legs, increasing the maximum possible chain length to 8 legs. The transport chain generation program produces the choice set of available transport chains, but already optimises the locations of transfers within each available type of transport chain. Table A-2 in the appendix contains a full list of chain types that are implemented in the Swedish model.

The ChainChoice procedure

The ChainChoice module calculates the shipment size and selects single 'best' chain based on total costs⁸ for each firm to firm relation. The chain choice procedure accounts for the consolidation and distribution of shipments to optimise the load factor of vehicles. Consolidation centres are locations where goods are transhipped (and possibly stored, but this is not modelled), with small loads getting in and larger loads getting out. Distribution centres are locations where goods are transhipped (and possibly stored), with large loads getting in and small loads getting out. Both consolidation and distribution centres exist not only in road transport, but can also be ports, airports or rail terminals. Whether a shipment can be consolidated at a consolidation centre with other shipments, depends on the OD flows, which are endogenous in the logistics model.

The level of consolidation (or the load factor of the vehicles) between consolidation centres and distribution centres, is determined in an iterative procedure. This happens because the question whether or not a particular shipment will be consolidated on a certain OD leg, depends on whether other cargo (other shipments) will be transported over the same OD leg, so there will be something to consolidate with. However, the latter is an output of the model (also see the Method Report on the Swedish logistics model).

The model derives the level of consolidation of OD flows between consolidation centres from a flow size ranking. This flow size ranking is the product of the allocated OD flow and the observed port output. In the first iteration, no shipments have been allocated yet, so a typical consolidation factor is assumed (this can be set in the BuildChain control files, see section 4.5). In this iteration, ChainChoice allocates shipments to the optimal chain for each chain type (this is set with the argument /UPDATE=CHOSEN, see section 3.3). The calculated consolidation factors from the first iteration are input to the second iteration. If the OD flow has its origin or destination in an important port, the flow is weighed with the observed terminal throughput, to direct more consolidation towards important ports. The OD flows are ranked according to this weighed consolidation factor and the final consolidation level is determined by allocating increased consolidation levels to the ranked OD flows. The range of consolidation levels is set in the ChainChoice control files (see section 5.5; by default the range is 0.05 to 0.95).

In the current model, consolidation only takes place within a commodity group. The choice of starting factor has a small but not negligible influence on the final results. To solve this, one could do more than three iterations. Consolidation in the model can take place at all transfer nodes (terminals, collection/distribution centres) that are included in the model system.

⁸ Second best (and further) chains can be written to file for analysis.

3.3 Controlling the iterative procedure

The structure of the logistic model can be modified or controlled on two aspects: the iteration between the BuildChain and ChainChoice procedures, and the starting value for the consolidation factor. The batch job `Commodity.bat` contains the core of the logistic model and is used to modify the model's structure. The content of this file is visualised in Figure 3-3. Please note that this program calls upon the logistic procedures `buildchain.exe` and `chainchoi.exe`. These procedures require control files and optional arguments.

By default, the iterative procedure consists of three rounds. In the first iteration step the consolidation on the network links is unknown (no logistic decisions have been simulated yet). Therefore, in the first call upon the procedure `buildchain.exe` the consolidation factor is set with the argument `/FACTOR`. This value can be set to any desired level between 0 and 1. It is advised to use the default value of 0.75.

Figure 3-3: Contents of [Commodity.bat]

```
cd buildchain
buildchain.exe buildchain%1.ct1 /FACTOR=0.75 /ITRNO=BC1
cd..
cd ChainChoi
chainchoi.exe chainchoi%1.ct1 /UPDATE=ALL /ITRNO=CC1
cd..
cd buildchain
buildchain.exe buildchain%1.ct1 /ITRNO=BC2
cd..
cd ChainChoi
chainchoi.exe chainchoi%1.ct1 /UPDATE=CHOSEN /RDVOLUMES=YES /ITRNO=CC2
cd..
cd buildchain
buildchain.exe buildchain%1.ct1 /ITRNO=BC3
cd..
cd ChainChoi
chainchoi.exe chainchoi%1.ct1 /RDVOLUMES=YES /ITRNO=CC3
cd..
```

Explanation of flags in [Commodity.bat]:

FACTOR	: exogenous consolidation factor. Range: [0,1] or [empty]. In the first iteration this is set to 0.75, which is advised as representative value. If the flag is missing, the consolidation factor is taken from the previous iteration
UPDATE	: sets the allocation of shipments to chains. ALL shipments are allocated to all available chains CHOSEN shipments are allocated to chosen chains
RDVOLUMES	: tells the chain choice program if OD-flows are available from previous iteration. If so, this argument should be set to YES
ITRNO	: Iteration number to be logged in the log file

The computation of the consolidation factor is set with the argument `/UPDATE` of the `chainchoi.exe` procedure. In the first iteration this argument is set to `ALL`, which means shipments are allocated to all available chains. The set of available chains consists of the optimal chain for each chain type). Based on these allocated shipments, the consolidation factor is determined.

In the second iteration round, the argument for the initial consolidation factors can be excluded from the call upon `buildchain.exe`. Now, the logistic chains are build with the consolidation factors that were computed in the first iteration. When the logistic choices are simulated in `ChainChoi`, the consolidation factors are determined and updated. In this second iteration the `/UPDATE` argument is set to `CHOSEN`, indicating that shipments are only allocated to the chosen chain and used in the computation of the consolidation for the third iteration round. The `ChainChoi` program now takes another argument `/RDVOLUMES=YES`, indicating that OD flows are available from the previous iteration, to calculate the consolidation levels and transport costs for each shipment. The final `ChainChoi` iteration ends with the `/UPDATE` argument blank⁹, in order to keep the consolidation factors that are stored, consistent with the consolidation factors that were used to determine the chain choices.

If desired, the batch job `Commodity.bat` can be extended by adding additional iterations. Any additional iteration requires a call on the `BuildChain` and `ChainChoi` programs. It is advised to call `buildchain.exe` without consolidation factor argument and `chainchoi.exe` with the following arguments: `/UPDATE=CHOSEN /RDVOLUMES=YES`. The last iteration is always run without the consolidation update switch to `/UPDATE=CHOSEN`, so the additional iteration is inserted between the 2nd and 3rd iteration in Figure 3-3. The following lines should be added for each additional iteration:

```
cd ..
cd buildchain
buildchain.exe buildchain%1.ct1 /ITRNO=BC4
cd ..
cd ChainChoi
chainchoi.exe chainchoi%1.ct1 /UPDATE=CHOSEN /RDVOLUMES=YES /ITRNO=CC4
```

3.4 input and output to the model

Inputs to the logistics model

The logistic costs are calculated with the transport costs and several input data. An overview of data that is used as input is given below:

- Transport costs: based on time, distance and extra costs (e.g. tolling) calculated from a network model

⁹ The default value for the `UPDATE` argument is `NONE`

- Terminals, e.g. direct access, location, which commodities can be handled, terminal time and costs
- Commodity-specific costs, e.g. order and storage costs
- Vehicle data, e.g. capacity and which vehicle types can be used for which commodities
- Restrictions, e.g. on draught in ports

Detailed lists of input files are given in the next chapters where the BuildChain and ChainChoice procedures are discussed. In addition to that an overview of all input parameters and variables is provided in Appendix B.

Outputs of the logistics model

The output of the model consists of

- f2f flows, including the shipment size and shipment frequency, the chosen transport chains (number of OD legs, vehicle/vessel type on each leg, transshipment locations) and the logistics costs (by cost item).
- Various aggregates from the f2f flows:
 - OD matrices of tonnes and tonne-kilometres by mode and commodity type (but also more aggregate –non-OD- statistics on the total number of tonnes and tonne-kilometres by mode and commodity type),
 - OD matrices of loaded vehicles by vehicle/vessel type, and of loaded and empty road vehicles. These can be used in assignment to the networks, applying the same network model as for the transport cost inputs.
 - output flows by vehicle type or by terminal.
 - total logistics costs per PWC flow, for use in the future determination of PWC flows.

To generate these outputs a separate procedure is used to process and aggregate the calculated logistic chains that were chosen in the model (see RAND Europe and SITMA, 2005).

A list of input and output files is provided in the following chapters where the two logistic components are described in detail.

4.1 Introduction

The `BuildChain` module builds and selects the “best” chain per commodity and chain type for each origin – destination pair. This Chapter describes the `BuildChain` procedure in the logistic program. First an overview is given of the input and output files. Next the content of the procedure is described. It is indicated how the procedure can be controlled by modifying input files or the control files that define the settings for the simulation.

4.2 Input files

The `BuildChain` uses a large number of input files to generate logistic chains. Below an overview is given of all files that are used. The overview contains a description of the content of all input files. If applicable, the units of the data are presented, or a reference is made to a detailed file description with units of variables elsewhere in this document.

Folder	Filename	Description
\BUILDCHAIN\	Buildchain.ctl	Control file for running the <code>BuildChain</code> procedure, containing common settings for all commodities. Structure of file is explained in Section 4.5
\BUILDCHAIN\	Buildchain1.ctl	control file for running the <code>BuildChain</code> procedure for commodity type 1. Structure of file is explained in Section 4.5
\BUILDCHAIN\
\BUILDCHAIN\	Buildchain35.ctl	control file for running the <code>BuildChain</code> procedure for commodity type 1. Structure of file is explained in Section 4.5
INPUT\COST\	cargo.txt	contains value, holding costs and order costs for all commodities. For a description of the content of the file, and the units of parameters see Section 4.5.
INPUT\COST\	PilotFees.txt	specifies the fees at each terminal [SEK per vehicle]
INPUT\COST\	Vhcls_dry_bulk.txt	contains dry bulk parameters for each vehicle type, such as capacity, cost per hour, costs per km, containerloadtime, etcetera. Content of the file is similar to <code>Vhcls_gen_cargo.txt</code> described in Section 4.5.
INPUT\COST\	Vhcls_gen_cargo.txt	contains general cargo parameters for each vehicle type, such as capacity, cost per hour, costs per km, containerloadtime, etcetera. For a description of the content of the file, and the units of parameters see Section 4.5.
INPUT\COST\	Vhcls_liq_bulk.txt	contains liquid bulk parameters for each vehicle type, such as capacity, cost per hour, costs per km, containerloadtime, etcetera. Content of the file is similar to <code>Vhcls_gen_cargo.txt</code> described in Section 4.5.
INPUT\LOS\	FreqAir.314	frequency for air mode [#transports/week]
INPUT\LOS\	FreqCombi.314	frequency for combi mode [#transports/week]
INPUT\LOS\	FreqContainerVessel.314	frequency for container vessels [#transports/week]
INPUT\LOS\	FreqLorry.314	frequency for lorries [#transports/week]

Folder	Filename	Description
INPUT\LOS\	FreqOtherVessel.314	frequency for other vessels [#transports/week]
INPUT\LOS\	FreqRailFerry.314	frequency for rail ferry [#transports/week]
INPUT\LOS\	FreqRoadFerry.314	frequency for road ferry [#transports/week]
INPUT\LOS\	FreqRoRoVessel.314	frequency for RoRo vessel [#transports/week]
INPUT\LOS\	FreqSystem.314	frequency for system [#transports/week]
INPUT\LOS\	FreqWaggonload.314	frequency for wagonload [#transports/week]
INPUT\LOS\	v101_dist.314	distance matrix for vehicle type 101 [km]
INPUT\LOS\	v101_ddist.314	Domestic distance matrix for vehicle type 101 [km]
INPUT\LOS\	v101_timeh.314	time matrix for vehicle type 101 [hour]
INPUT\LOS\	v101_xkr.314	extra costs matrix for vehicle type 101 [kSEK]
...
INPUT\LOS\	v401_dist.314	distance matrix for vehicle type 401 [km]
INPUT\LOS\	v401_ddist.314	Domestic distance matrix for vehicle type 401 [km]
INPUT\LOS\	v401_timeh.314	time matrix for vehicle type 401 [hour]
INPUT\LOS\	v401_xkr.314	extra costs matrix for vehicle type 401 [kSEK]
INPUT\NODES\	nodes.txt	contains general node information for all commodities. For a description of the content of a node file and the units of parameters see Section 4.5
INPUT\NODES\	transferroadroad.txt	Contains a dummy variable indicating whether or not road-road transfers are allowed
INPUT\NODES\	transferroadtrain.txt	Contains a dummy variable indicating whether or not road-train transfers are allowed
INPUT\NODES\	transferroadsea.txt	Contains a dummy variable indicating whether or not road-sea transfers are allowed
INPUT\NODES\	transferroadcombi.txt	Contains a dummy variable indicating whether or not road-combi transfers are allowed
INPUT\NODES\	transferroadair.txt	Contains a dummy variable indicating whether or not road-air transfers are allowed
INPUT\NODES\	transferroadroadferry.txt	Contains a dummy variable indicating whether or not road-road-ferry transfers are allowed
INPUT\NODES\	Transferseasea.txt	Contains a dummy variable indicating whether or not sea-sea transfers are allowed
INPUT\NODES\	transfercombisea.txt	Contains a dummy variable indicating whether or not Combi-sea transfers are allowed
INPUT\NODES\	transferwagonloadrailferry.txt	Contains a dummy variable indicating whether or not wagonload-railferry transfers are allowed
INPUT\NODES\	transferwagonloadsea.txt	Contains a dummy variable indicating whether or not wagonload-sea transfers are allowed
INPUT\NODES\	transferfeedertrainwagonload.txt	Contains a dummy variable indicating whether or not feeder train-wagonload transfers are allowed
INPUT\NODES\	transfersystemtrainsea.txt	Contains a dummy variable indicating whether or not system train-sea transfers are allowed
INPUT\NODES\	Directsea.txt	Contains a dummy variable indicating whether or not direct sea access is allowed
INPUT\NODES\	directsystemtrain.txt	Contains a dummy variable indicating whether or not direct system train access is allowed
INPUT\NODES\	directfeedertrain.txt	Contains a dummy variable indicating whether or not direct feeder train access is allowed
INPUT\NODES\	directwagonload.txt	Contains a dummy variable indicating whether or not direct wagonload access is allowed
INPUT\NODES\	containerhandling.txt	Contains a dummy variable indicating whether or not direct containers can be handled
INPUT\PWC\	pwc_01.txt	producer demand matrices, with the annual demand (flows from producers or wholesales to consumer between zones) for commodity type 1. [annual demand in tonnes]
...
INPUT\PWC\	pwc_35.txt	producer demand matrices, with the annual demand (flows from producers or wholesales to consumer between zones) for commodity type 1 [annual demand in tonnes]

Folder	Filename	Description
\\BUILDCHAIN\	LOGSELECT.DAT	selection of OD pairs for which the available chains are stored in BuildChain1.log - BuilCHain35.log . To enable this output refer to LOGSELECT parameter. If LOGSELECT=1 it is mandatory provide this file under the BUILDCHAIN folder.
\\BUILDCHAIN\	select.dat	selection of OD pairs for which the available chains are stored in connection1.lst - connection35.lst . To enable this output refer to SELECT and CONNLST parameters.

4.3 Output files

In the overview below, the output files are listed that are generated by the BuildChain procedure. For each commodity type an output file is generated. This file contains the logistic chains between all zone to zone combinations. This result is input to the ChainChoice procedure.

Folder	Filename	Description
\\BUILDCHAIN\OUTPUT\	Chains1.dat	output file with logistic chains for commodity type 1
\\BUILDCHAIN\ OUTPUT\
\\BUILDCHAIN\ OUTPUT\	Chains35.dat	output file with logistic chains for commodity type 35
\\BUILDCHAIN\ OUTPUT\	connection1.lst	List of all available connections for a selection of relations for commodity type 1 ¹⁰
\\BUILDCHAIN\ OUTPUT\
\\BUILDCHAIN\ OUTPUT\	Connection35.lst	List of all available connections for a selection of relations for commodity type 35 ¹⁰
\\BUILDCHAIN\ OUTPUT\	Buildchain1.log	log file with reports of BuildChain process for commodity type 1. See LOGSELEC parameter to control the amount of logged information.
\\BUILDCHAIN\ OUTPUT\
\\BUILDCHAIN\ OUTPUT\	Buildchain35.log	log file with reports of BuildChain process for commodity type 35. See LOGSELEC parameter to control the amount of logged information.

The output files contain a varying number of logistic chains depending on commodity type and availability of transport modes. Below the output is illustrated with a sample of an output file for commodity type 17 (metal products).

Figure 4-1: example of a BuildChain output file: Chain 17.dat, with header description

<i>orig</i>	<i>dest</i>	<i>No.of chains</i>	<i>Chain type</i>	<i>orig</i>	<i>dest</i>	<i>orig id</i>	<i>dest id</i>	<i>time [hour]</i>	<i>dist. [km]</i>	<i>xtra cost [SEK]</i>	<i>freq [#week]</i>	<i>consol. factor [%]</i>
711400	716000	19	C	711400	716000	0	28	0.2709	17.44	0		
			A	711400	716000	0	28	0.2709	17.44	0		
			B	711400	716000	0	28	0.243	17.44	0		
			BSB	711400	718003	0	35	0.2137	18.61	0		
				718003	718004	35	36	0.1499	8.34	0	5	0.05
				718004	716000	36	28	0.2518	18.2	0		
			AFEA	711400	718121	0	53	0.6785	54.89	0		
				718121	718111	53	51	0.0134	0.67	9.112E-50.2		0.05
				718111	886111	51	497	1.88	161.15	0.02094	5	0.5048
				886111	716000	497	28	2.79	223.99	0		
			CGHC	711400	718014	0	43	0.2178	16.67	0		
				718014	718017	43	46	0.144	10.16	0.00132	0.2	0.05
				718017	908411	46	565	2.08	158.41	0.02154	5	0.05
				908411	716000	565	28	1.9	147.12	0		
			ADA	711400	918011	0	581	1.9	150.65	0		

¹⁰ This output is optional. The selection of relations is made in the 'select.dat' file in the \\BUILDCHAIN- directory.

918011	718012	581	41	2.4	185.87	0.03159	5	0.05
718012	716000	41	28	0.316	22.89	0		

The chain output file contains the origin and destination zone (in this case between zone 711400 and 716000). Next the number of chains is specified: 19. For each of these chains, the chain type is given (e.g. C= Heavy Road, non-container), the origin and destination of the respective leg, the origin and destination indices of the leg, the time, distance and extra costs. If applicable, the line contains a frequency and consolidation factor for the leg. This depends on the respective transport mode. The example below shows that for feeder trains (modes E) or wagonload trains (mode F) consolidation factors are computed. For Heavy lorry mode A, for containerised transport, consolidation is not an option. For non-containerised transport, consolidated heavy lorry is coded as mode S. Consolidation with heavy lorry occurs only in the middle of 3 or more legged chains, and can not occur at the destination or origin leg.

The connection list output files specify the available chains for a selection of relations. This selection of relations is optional and controlled in the Select.dat file. The output lists have a simple structure: origin node, available mode, destination node.

Figure 4-2: Sample of a connections list output file

Orig	Mode	Dest
711400	A	711400
711400	A	711401
711400	A	711402
711400	A	711403
711400	A	711500
711400	A	711700
711400	A	712000
711400	A	712300
711400	A	712301

Depending of the LOGSELECT-setting, the log files (*.log) may contain warnings that occurred during execution of the module for the specific commodity type. For instance, a warning is written when no frequency is available; in that case the default frequency is used (the default frequency, DfltFreq, is set in the vehicle cost parameter files, see section 4.5). A warning is also given if the LOS attributes are incomplete (e.g. frequency is given, but the time or cost is not available); in that case the connection is discarded. An example of a log file fragment is given in the figure below:

Figure 4-3: Sample of a log file fragment

Using default frequency: Orig=711400; Dest=711401; Mode=A; Time=0.044197; Dist=2.23; Freq=84
Using default frequency: Orig=711400; Dest=711402; Mode=A; Time=0.072314; Dist=4.29; Freq=84
Using default frequency: Orig=711400; Dest=711403; Mode=A; Time=0.0368; Dist=1.84; Freq=84

4.4 Description

The logistic module first simulates the choice between a number of transport chains (with one to four legs and with different aggregate modes and different vehicle/vessel types for each leg). Transfers can take place within the road system (e.g. from van to truck), or

between road, sea, rail and air transport (and the Swedish model also includes transfers within the sea and within the rail system: feeder and main haul transport). The transport chain generation program produces the choice set of available transport chains, but already optimises the locations of transfers within each available type of transport chain.

The tables in the appendix give an overview of the vehicle types and transport chains for Sweden. The current version of the model distinguishes 35 vehicle types, and 86 transport chains. For each transport chain type, *BuildChain* produces all possible transport chains and their associated logistic costs. For each chain type, the alternative with minimal logistic costs is stored.

The logistic costs function

The logistic costs function is specified in RAND Europe and SITMA (2005) and also specified in the Method report (Significance, 2010). The total annual logistics costs G of commodity k transported between firm m in production zone r and firm n in consumption zone s of shipment size q using logistic chain l :

$$G_{rskmnl} = o_k \cdot (Q_{kmn}/q_{kmn}) + T_{rskql} + Y_{rskl} + (w_k + (d \cdot v_k)) \cdot (q_{mnk}/2) \quad (1)$$

Where:

G : total annual logistics costs

O_k : order costs, dependent of the annual demand, for commodity k . $o_k \cdot (Q_{kmn}/q_k)$ in equation 1

o_k : the constant unit cost per order of commodity k

Q_k : the annual demand (tonnes per year) for commodity k

q_{mnk} : the average shipment size for commodity k transported between firm m and firm n .

T_{rskql} : transport, consolidation and distribution costs between production zone r and consumption zone s , for commodity k , shipment size q , using logistic chain l . T comprises of link-based costs, vehicle/vessel type specific costs, vehicle/vessel pair specific costs, commodity type specific costs. The transport costs function is specified in detail below.

Y_{rskl} : capital costs of goods during transit. $Y_{rskl} = (d \cdot t_{rsl} \cdot v_k \cdot Q_k) / (365 \cdot 24)$

d : the discount rate (per year)

t_{rsl} : the average transport time (in hours) between production zone r and consumption zone s , using logistic chain l .

v_k : the value of the commodity that is transported (in SEK per tonne).

w_k : the storage costs (in SEK per tonne per year) for commodity k .

Please note that the order and holding costs are omitted in the implementation of the logistic costs function in the *BuildChain* procedure. These cost components do not vary across logistic chain alternatives because a fixed average shipment size per commodity type is used in *BuildChain*. Therefore, these costs are irrelevant for the generation of the

optimal chain for each chain type. The ChainChoice procedure does include these components as these costs are affected by the chosen shipment sizes.

The transport costs and capital costs during transit between production zone r and consumption zone s, using logistic chain l is derived by summing the transport costs and capital costs of each leg i in the logistic chain l:

$$T_{rskql} + Y_{rskl} = \sum_{i \in l} (T_{kqi} + Y_{ki}) \quad (2)$$

The transport cost function T_{kqi} is dependent on the transport mode that is used in the leg. In BuildChain a typical vehicle type is used (which is set in the control file), and ChainChoice uses the optimal vehicle type. The transport cost function is specified for container, non-container and ferry transport. The transport cost function for containers consists of a time cost, distance cost, other link costs (e.g. tolling), loading (and unloading) costs, fairway dues, and pilot fees. The computation of the link based transport and capital costs is implemented as follows:

$$\begin{aligned} T_{kqi}^{\text{container}} + Y_{ki} = & n \cdot (t_v^{\text{load:container}} \cdot (f_o^{\text{time}} + f_d^{\text{time}}) + t_i) \cdot c_v^h + q_k \cdot (t_v^{\text{load:container}} \cdot (f_o^{\text{time}} + f_d^{\text{time}}) + t^{\text{wait}} + t_i) \cdot d_k^h \\ & + n \cdot \text{dist}_i \cdot c_v^d \\ & + n \cdot c_i^{\text{oth}} \\ & + (f_o^{\text{cost}} + f_d^{\text{cost}}) \cdot q_k \cdot c_v^{\text{load:container}} \\ & + n \cdot c_v^{\text{fairw}} + q_k \cdot c_v^{\text{fairwT}} \\ & + n \cdot c_v^{\text{pilot}} . \end{aligned} \quad (3a)$$

with: n = number of vehicles, $t_v^{\text{load:container}}$ = vehicle type specific containerload time, t^{wait} = wait time (not at origin and destination) with $t^{\text{wait}} = \frac{0.5 \cdot f \cdot 24}{f}$, f = frequency of the vehicle per week, t_i = link time, c_v^h = vehicle type specific time costs, d_k^h = interest costs per hour per tonne of commodity k calculated as: $d_k^h = v_k \cdot i / (365 \cdot 24)$, i = interest rate per year, dist_i = link distance, c_v^d = vehicle type specific distance costs, c_i^{oth} = other link costs, f_o^{time} = technology factor for terminal time efficiency at origin node, f_d^{time} = technology factor for terminal time efficiency at destination node, f_o^{cost} = technology factor for terminal cost efficiency at origin node, f_d^{cost} = technology factor for terminal cost efficiency at destination node, q_k = shipment size, $c_v^{\text{load:container}}$ = vehicle type specific load costs per tonne for containers, c_v^{fairw} = fairway dues by vehicle, c_v^{fairwT} = fairway dues by tonne, c_v^{pilot} = pilot fee at origin mode.

The Swedish vehicle/vessel type classification (see Table A-2 in the appendix) is based on the assumption that unitised transport can be used with most vehicle/vessel types (exceptions: the first three light/medium road vehicles, system train and airplane cannot be used for container transport; the Kombi train and the container vessels are for container transport only). This means that the cost for the unitised variant includes costs for initial stuffing of the container (at the sender) and final stripping (at the receiver) and that there are differences in the transfer costs (generally speaking container transfers are cheaper than other transfers at consolidation and distribution centres). Therefore, the link costs at the origin and destination of the chain include an additional cost component for stuffing and

stripping. If link i starts at production zone r , the link based costs are increased with the stuffing costs, calculated as:

$$T_{kqi=r}^{\text{container}} = T_{kqi}^{\text{container}} + n \cdot c_v^{\text{stuffing}} \quad (3b)$$

If link i ends at consumption zone s , the link based costs are increased with the stripping costs, calculated as:

$$T_{kqi=s}^{\text{container}} = T_{kqi}^{\text{container}} + n \cdot c_v^{\text{stripping}} \quad (3c)$$

The model assumes equal stuffing and stripping costs, which is set in the control file of the ChainChoice program.

For non-containerised transport the transport function is similar, except some non-container parameters are different:

$$\begin{aligned} T_{kqi}^{\text{non-container}} + Y_{ki} = & n \cdot (t_v^{\text{load;non-container}} \cdot (f_o^{\text{time}} + f_d^{\text{time}}) + t_i) \cdot c_v^h + q_k \cdot (t_v^{\text{load;non-container}} \cdot (f_o^{\text{time}} + f_d^{\text{time}}) + t^{\text{wait}} + t_i) \cdot d_k^h \\ & + n \cdot \text{dist}_i \cdot c_v^d \\ & + n \cdot c_i^{\text{oth}} \\ & + n \cdot c_i^{\text{position}} \\ & + (f_o^{\text{cost}} + f_d^{\text{cost}}) \cdot q_k \cdot c_v^{\text{load;non-container}} \\ & + n \cdot c_v^{\text{fairw}} + q_k \cdot c_v^{\text{fairwT}} \\ & + n \cdot c_v^{\text{pilot}}. \end{aligned} \quad (4)$$

with c_v^{position} = positioning costs by vehicle.

Ferry links have a specific cost function because loading costs, fairway dues and pilot fees do not apply. The categorisation to containerized or non-containerized is derived from the previous leg. If the previous leg was non-containerized, the transport costs at ferry links becomes:

$$\begin{aligned} T_{kqi}^{\text{ferry}} + Y_{ki} = & n_{ro} \cdot (t_v^{\text{load;non-container}} + t^{\text{wait}} + t_i) \cdot (c_v^h + d_k^h) \\ & + n_{ro} \cdot \text{dist}_i \cdot c_v^d. \end{aligned} \quad (5)$$

with n_{ro} = number of vehicles for the shipment on the ferry. If the previous leg was containerized, the transport costs at ferry links becomes:

$$\begin{aligned} T_{kqi}^{\text{ferry}} + Y_{ki} = & n_{ro} \cdot (t_v^{\text{load;container}} + t^{\text{wait}} + t_i) \cdot (c_v^h + d_k^h) \\ & + n_{ro} \cdot \text{dist}_i \cdot c_v^d. \end{aligned} \quad (6)$$

The cost function parameters are set in separate input files and control files to facilitate running policy variants. In the following of this document, when the input files are discussed in more detail references are made to the cost functions specified above.

The annual discount rate can be set in the control file (see the description of the INTEREST parameter in section).

Build logistic chain algorithm

The BuildChain procedure searches for all typical logistic chains and identifies the optimal chain for each of these chain types. The current model distinguishes 67 chain types for Sweden (see Table A-3 for an overview). For each chain type, BuildChain calculates the optimal transfer locations and logistic costs for the logistic chain. In doing so, the algorithm follows a stepwise approach in adding extra legs to chains and analysing the optimal transfer locations.

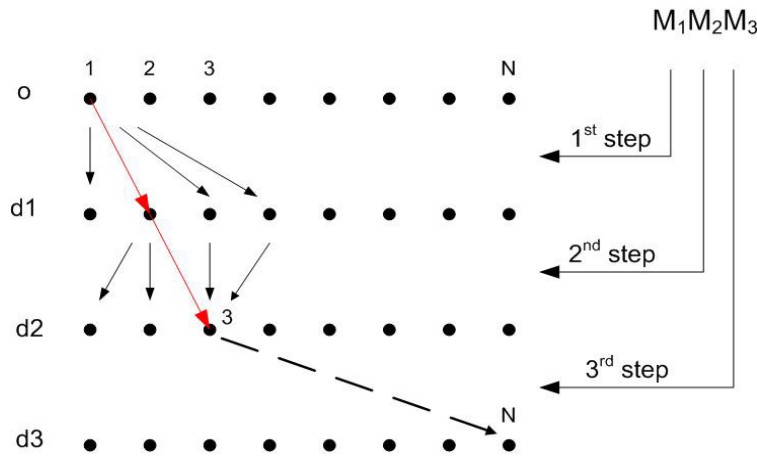


Figure 4-4: Search algorithm, the optimal two leg chain M_1M_2 from origin 1 to destination 3 is indicated in red

The algorithm is based on the assumption that the optimal route to a final destination consists at least of the optimal route to an intermediate destination, if this intermediate destination is on the optimal route to the final destination. The only additional information required is the optimal route between the intermediate destination and the final destination. This approach is explained in Figure 4-4.

For each origin o , the procedure generates chains that can consist of one leg (M_1) to for instance three legs ($M_1M_2M_3$). All transport modes are taken into account. The optimal chain of just one leg (M_1) to each destination is trivial: the alternative with the least logistic costs.

The algorithm generates chains from this origin to each possible destination d , and tries to use the information from the chains that are produced for shorter chains as efficient as

possible. Now, suppose the procedure is searching for the optimal chain of three legs ($M_1M_2M_3$) from origin 1, to destination N, under the condition that the second transfer is made in node number 3. The program has already determined the optimal logistic chain of two legs to this transfer point, as indicated in red in Figure 4-4. It will use this chain as the first two legs of the new three legged chain from origin 1, to N, with a second transfer at node number 3. The program only needs to determine the optimal third leg of this chain. Please note that the program searches for three legged chains from zone 1 to N through all possible transfer nodes, not only through node 3. The optimal two legged chain between this transfer node and zone 1 is already determined by the program.

Please note that the `BuildChain` procedure uses a different shipment size than the `ChainChoice` procedure. The optimal shipment size, q_k in equation (1), is only available after the logistic decisions have been simulated. So `BuildChain` uses a fixed shipment size, representative for the specific commodity type and (possibly) zone, when the set of possible logistic chains are generated. This shipment size is used in all iterations and is set in the `BuildChain` control files (see next Section). For an overview of default values that were implemented in the commodity specific control files, see Table A-1 in the appendix.

4.5 Controlling the BuildChain procedure

The `BuildChain` procedure is controlled through the commodity specific control files. This control file is used to set parameters and define the input files. This section describes what the influence of each parameter is, and will illustrate how the input files can be modified.

Figure 4-5 gives an example of the control file for running the `BuildChain` procedure for commodity 1. The parameters in this file specify parameter values for the procedure or paths to input or output files. The parameters in the control file are described in the overview that follows. This description is not exhaustive but illustrates the most important input files that are used to calculate the logistic costs for the transport chains.

Figure 4-5: Example of a BuildChain control file: [BuildChain1.ctl]

```
INCL=buildchain.ctl
COMMODITY=1
TONNES=dynamic
VHCL=..\Input\Cost\vhcls_dry_bulk.txt
PWC=..\Input\PWC\PWC_01.txt
VHCLA=104
...
VHCLR=401
CHAINS=OUTPUT\Chains1.dat
LOG=OUTPUT\BuildChain1.log
CONNLIST=OUTPUT\connection1.lst
```

Explanation of parameters:

INCL	: reference to the control file that contains the common settings for all commodities. An example of this file is given in Figure 4-6.
LOGSELECT	: Indicator (0/1/2) that controls the amount of information in the log file about incomplete LOS: LOGSELECT= 0 means no information is logged. LOGSELECT= 1 means information is logged for a selection of relations. To specify the list of relations it is mandatory to provide an extra file (LOGSELECT.DAT) as input. Please refer to input file section for further details. LOGSELECT= 2 means information is logged for all relations (not recommended for disk space saving).
COMMODITY	: commodity code [1..35]
TONNES	: If this parameter is a number, then this number will be used as the typical shipment size for this commodity. If this the parameter is set to DYNAMIC_MAX, DYNAMIC_AVERAGE or DYNAMIC_GEOMEAN the typical shipment size will be different for different zones and calculated as the maximum, average or geometric mean of the PC matrix. For an overview of default values per commodity group see Table A-1 in the appendix. This parameter is used to set the logistic cost parameter q (see equation 1) and calculate the logistic costs for the chains [tonnes per shipment]
VHCL	: path to file with vehicle data for the specific commodity type
PWC	: path to file with producer demand matrices for the specific commodity
VHCLA	: Typical vehicle type number for mode A
...	...
VHCLR	: Typical vehicle type number for mode R
CHAINS	: path for output file containing the optimal chains
SELECT	: path to input file containing the collection of relations that should be included in the detailed connection list output. Example SELECT=BUILDCHAIN\SELECT.dat. This parameter works in connection with CONNLST.

CONNLST : path to output file (see Figure 4-2: Sample of a connections list output file) containing all connections on a selection of relations. The selection list is specified by the SELECT specifier.

LOG : path for log file

Figure 4-6: Example of a common BuildChain control file: [BuildChain.ct1]

```
LOGCTL=1
LOGFLS=1
LOGCST=1
LOGSELECT=0
INTEREST=0.1
CONSOL=0.05,0.95
CONSOLA=0.05,0.5
CONSOLB=0.05,0.5
CONSOLC=0.05,0.5
CONSOLS=0.5,0.9
ALL_LORRY_TYPE_CONSOL=1
TYPES=..\Input\chaintype.lis
NODES=..\Input\Nodes
CARGO=..\Input\Cost\cargo.txt
PILOTFEES=..\Input\Cost\pilotfees.txt
LOSDir=..\Input\LOS
CONSOLDIR=..\ChainChoi\OUTPUT
SELECT=BUILDCHAIN>Select.dat
CONNLST=OUTPUT\connection1.lst
```

Explanation of parameters:

LOGCTL : Indicator (0/1) that determines whether or not CTL file settings will be logged in the log file.

LOGFLS : Indicator (0/1) that determines whether or not input file information will be logged in the log file.

LOGCST : Indicator (0/1) that determines whether or not cost parameters will be logged in the log file.

INTEREST : Interest rate used in cost calculations [%/year]

CONSOL : Default consolidation range used when no mode specific range is specified

CONSOL<mode> : Consolidation range for mode <mode>

ALL_LORRY_TYPE_CONSOL : 0/1 switch that determines whether or not consolidation is allowed for all lorry types¹¹ (1 means all lorries are consolidated)

¹¹ Regardless of this setting a separate mode S (consolidated heavy lorry) is distinguished in the mode. Because this mode is only allowed between terminals and on international relations, a consolidation factor range may be used that differs from the other lorry modes (A,B,C).

TYPES	: Path to file containing all available chain types
NODES	: Path to directory containing the nodes input files. Each nodes file contains a single node variable for all commodities. The file names of the nodes files are fixed.
CARGO	: path to file with the cargo values, holding costs and order costs used in the logistic cost function.
PILOTFEES	: path to file with pilot fees.
LOSDIR	:Path to directory containing the Level of service files. The level of service files have fixed names, referring to the applicable vehicle type.
CONSOLIDIR	:Path to directory containing the consolidation factor matrices. The consolidation factor files have fixed names, referring to the applicable mode and commodity.
SELECT	: path to input file containing the collection of relations that should be included in the CONNLST-file.

Explanation of input files:

Below the most important input files are discussed. The content of each file is explained and it is indicated how the input is used by the BuildChain procedure.

The file cargo.txt (Table 4-1) can be used to test different assumptions in the commodity specific holding and order costs, or the average values of commodities. This input file contains the commodity specific parameters in the logistic costs function (see equation 1). The file can be used to modify the assumptions in commodity specific costs. For instance, if the value of a commodity is increased (column 2), this will increase the capital costs on both the inventory at the receiver and on the inventory in transit. This will affect both the chain generation procedure as the chain choice procedure. Increasing the holding or order costs, will affect the logistic choices in the ChainChoice procedure, when the optimal shipment sizes are determined. BuildChain uses a fixed average shipment size, so the order and holding costs do not vary across the logistic chain alternatives.

Table 4-1: Contents of cargo.txt

Column	Parameter	Description
1	Commodity	commodity code [1..35]
2	Value	value of commodity, v_k in equation 1 [SEK / tonne]
3	HoldingCosts	storage costs, w_k in equation 1 [SEK / (tonne*year)]
4	FixedOrderCosts	constant unit cost per order, o_k in equation 1 [SEK]
5	ProportionalOrderCosts	Parameter used for the calculation of the annual demand dependent order costs ¹²

¹² The annual demand dependent order costs are calculated as: $OrderCosts = FixedOrderCosts + ProportionalOrderCosts \times AnnualDemand^{Alpha}$

6	Alpha	Parameter used for the calculation of the annual demand dependent order costs
---	-------	---

The files `vhcls_gen_cargo.txt`, `vhcls_dry_bulk.txt` and `vhcls_liq_bulk.txt` (Table 4-2) contain the vehicle specific costs parameters in the transport costs function (equations 3 to 5). The input parameters include capacity, the time and distance costs for the vehicle type, and a default frequency that is used if no frequency is available from the network cost matrices.

Table 4-2: Contents of `vhcls_gen_cargo.txt`:

Column	Parameter	Description
1	Nr	vehicle type number
2	VesselType	vessel type, for all water transport modes. 0 = not a vessel; 1 = container vessel; 2 = ro-ro vessel; 3 = other vessel.
3	Capacity	capacity of vehicle type. This influences the shipment sizes and required number of vehicles [tonne]
4	Coordination factor	Factor used to catch the fact that the available volume for each vehicle movement will be lower than the calculated annual consolidation volume at the terminal
5	HourCost	link-based time costs for the vehicle, parameter c_v^h in transport costs functions 3 and 4 [SEK per hour per vehicle]
6	KmCost	link-based distance costs for the vehicle, parameter c_v^d in transport costs functions 3 and 4 [SEK per km per vehicle]
7	OnFerryHourCost	link-based costs (time) on a ferry link, parameter c_v^h in transport costs functions 5 [SEK per hour per vehicle]
8	OnFerryKmCost	link-based distance costs on a ferry link, parameter c_v^d in transport costs functions 5 [SEK per km per vehicle]
9	PositioningCost	initial transport costs for vessel to arrive at the loading point for the link, parameter $c_v^{position}$ in transport costs function 4 [SEK per vehicle]
10	DfltFreq	default frequency if frequency is unavailable from LOS input files [#transports/week]
11	ContainerLoadTime	time costs for loading/unloading container transport at distribution and consolidation centres, parameter $t_v^{load;container}$ in transport costs function 3 [hour per vehicle]
12	ContainerLoadCost	costs for loading/unloading container transport at distribution and consolidation centres, parameter $c_v^{load;container}$ in transport costs function 4 [SEK per tonne]
13	NonContainerLoadTime	time costs for loading/unloading non-container transport at distribution and consolidation centres, parameter $t_v^{load;non-container}$ in transport costs function 3 [hour per vehicle]

14	NonContainerLoadCost	costs for loading/unloading non-container transport at distribution and consolidation centres. parameter $c_v^{load;non-container}$ in transport costs function 4 [SEK per tonne]
15	VhclFairwayDues	Fairway costs for vehicle, parameter c_v^{fairw} in transport costs functions 3 and 4 [SEK per vehicle]
16	TonnesFairwayDues	Fairway costs for vehicle parameter c_v^{fairwT} in transport costs functions 3 and 4 [SEK per tonne]

Increasing the time and distance costs of a vehicle type (column 4 and 5) will increase the transport costs for each leg specifically. This will affect the identification of optimal transport chains in BuildChain. Modifying the capacity (column 3) of vehicle types affects the frequency optimisation (in the ChainChoice procedure). This also counts for typical vehicle types as used in BuildChain. Modifying the default frequency in column 6 will only affect those links where no frequency is available from the LOS input files (see Section 4.2). The frequency affects the waiting time t^{wait} in equations 3 to 5. This parameter will affect the optimal logistic chain that was built. Increasing the container or non-container loading time and costs, in columns 7 to 10 will increase the transport costs for each leg specifically. This will affect the identification of optimal transport chains. Increasing the fairway dues by vehicle or tonnes, in columns 11 to 12 will increase the transport costs for each leg specifically. This will affect the identification of optimal transport chains.

The availability of vehicle types depends on the transport mode and is set in the control files (e.g. parameter VHCLA in the Buildchain and ChainChoice control files). The loading and unloading costs for non-container and container transports are not relevant for some vehicle types. The vehicle input files contain the system value “999999” for such irrelevant cases.

The LOS-input files that contain the Level-of-Service between all nodes in the network. These input files are delivered by Samgods in EMME/2 format. The LOS input matrices have fixed names, only the directory where these files are located are input to the BUILDCHAIN module. For a complete list of all files that should be available in the LOS input directory see paragraph 4.2. These LOS files give the travel time, distance, domestic distance or extra travel costs between nodes in the network. Intra zonal distance, time and costs are included as well. BuildChain generates intrazonal ‘chains’ with these intrazonal data. Intrazonal flows can only consist of one leg, so for intrazonal flows only direct chains are generated. The LOS of available modes determines wich direct chain is optimal (usually this is lorry).

Figure 4-7: Sample of distance matrix for vehicle type 101.

```

c EMME/2 Module: 3.14(v9.05) Date: 09-06-10 11:04 User: E061/SIKA.....jm
c Project: Samgods version 1
t matrices
a matrix=mf08 dist 0 distance
711400 711400:5.340
711400 711401:.630
711400 711402:4.290
711400 711403:1.840
    
```

```

711400 711500:12.370
711400 711700:28.780
711400 712000:50.110
711400 712300:17.690
711400 712301:19.540
711400 712500:38.260
711400 712600:42.910
711400 712601:41.780
711400 712602:34.390
711400 712603:42.580

```

The files containing the node variables have fixed names, only the directory where these files are located are input to the BUILDCHAIN module. For a complete list of all files that should be available in the nodes input directory see paragraph 4.2. These files can be used to modify or add a transfer terminal. If transfer locations are added or changed, this can affect the logistic chains that are generated: different chains are found by a change in optimal transfer locations.

The nodes.txt file is a tab-delimited text file. Here the content of the file is explained. For an elaboration on how to add nodes, zones or terminals see the description in section 2.4.

Table 4-3: Contents of nodes.txt:

Column	Parameter	Description
1	Nr	node code
2	Name	node name
3	Zone	zone code of node location, node code of terminal that provides direct access to this zone or empty ('0')
4	Domestic	dummy for domestic location. 1 = domestic, 0 = non-domestic.
5	CostTechnoFac	Cost efficiency factor at terminal, f^{cost} in transport costs functions 3 and 4 [0..1]
6	TimeTechnoFac	Time efficiency factor at terminal, f^{time} in transport costs functions 3 and 4 [0..1]
7	MaxDwtContainerVessel	Load constraint of container vessels that port can handle [tonne].
8	MaxDwtRoroVessel	Load constraint of Roro vessels that port can handle [tonne].
9	MaxDwtOtherVessel	Load constraint of other vessels that port can handle [tonne]
10	SeaOutput	Yearly sea output. This exogenous factor is used to determine a size ranking for sea terminals [tonne]
11	SeaContainerOutput	Yearly sea container output. This exogenous factor is used to determine a size ranking for sea terminals [tonne]
12	AirOutput	Yearly air output. This exogenous factor is used to determine a size ranking for air terminals [tonne]
13	Comments	Column to add a descriptive comment for a node

The first column contains the node number. The second contains the node name, this field is not used in the model. The third column contains a zone number, or zero. If node number (column 1) and zone number (column 3) are equal, the node is considered to be a

zone¹³. If these numbers are different, but the zone number is not equal to zero, the node can be used as a direct access terminal for the zone listed in column 3. The fourth column in nodes.txt contains a domestic dummy. This is used to determine which logistic chains will be available. The logistic network outside Sweden is much less detailed, missing most consolidation or distribution possibilities. To avoid underestimation of consolidated flows to and from Sweden, direct access by consolidated heavy lorries is allowed for the non-domestic zones, indicated with value '0' in column 4. Columns 5 and 6 contain cost and time efficiency factors at the terminal, representative for the technology level of transfer facilities. These technology factors are used as multipliers upon the loading costs (see equations 3 and 4). If the factor is changed from 1.0 to 0.75, the loading costs in the transport costs function are decreased by 25%. Column 7 to 9 contain maximum load constraints. Modifying these values can affect the generation of logistic chains, in particular for logistic chains for flows with larger volumes. If a terminal has no maximum load constraint the value '99999' is used as a maximum. If the load constraint is not relevant (e.g. because the node is a zone) the value '0' is used. Column 10 to 12 contain exogenous (observed) output data for transfer terminals (from the port statistics). The outgoing link flows that are calculated in the logistic module are weighed by this exogenous factor to determine the ranking and level of consolidation (this is described in the Method Report). This is not a policy variable in the model but derived from observed data.

All other input files containing the node variables have the same tab-delimited format. The first column contains the node number, the second column the node name. After that 35 columns follow, one for each commodity. If the dummy variable has the value TRUE, the cell contains the commodity number. If the dummy variable has value FALSE the cell is empty.

The file pilotfees.txt contains the pilotfees at the nodes in the network for specific vehicle types. The first line in the file can be used for comments. The second line contains a file header that specifies the vessel types. The structure of the remainder of the file is shown in Table 4-4. The pilot fee parameters are used as input to the transport costs functions: c_{vt}^{pilot} in equations 3 and 4. Increasing the pilotfee at a node for a specific vehicle type, will increase the transport costs for each outgoing leg from this node for goods carried with that specific vehicle type. This increases the transport costs for these outgoing links and will affect the identification of optimal transport chains.

Table 4-4: Contents of pilotfees.txt:

Column	Parameter	Description
1	Terminal nr	code of zone with sea terminal
2	vehicle type from file header	Pilot fee in SEK per vehicle for applicable vessel type
..		
18	vehicle type from file header	Pilot fee in SEK per vehicle for applicable vessel type

¹³ This procedure is implemented to allow for an efficient data handling.

5.1 Introduction

The `ChainChoice` procedure simulates the logistic decisions and optimizes the logistic costs. In doing so, the shipment size is determined and a choice is made for a transport chain. This Chapter first presents an overview of the input and output files. Next, the procedure is outlined. Next, it is discussed how alternative options are set for this procedure and how these influences the results.

5.2 Input files

The `ChainChoice` uses a large number of input files. Some of these input files are used by the `BuildChain` procedure or calculated in that procedure. Below an overview is provided of all input files that are used.

Table 5-1: Overview of input files

Folder	Filename	Description
CHAINCHOI\	Chainchoi.ctf	Control file for running the <code>BuildChain</code> procedure, containing common settings for all commodities. Structure of file is explained in Section 4.5
CHAINCHOI\	Chainchoi1.ctf	control file for running the <code>ChainChoice</code> procedure for commodity type 1. Structure of file is explained in Section 4.5
CHAINCHOI\	...	
CHAINCHOI\	Chainchoi35.ctf	control file for running the <code>ChainChoice</code> procedure for commodity type 35. Structure of file is explained in Section 4.5
INPUT\COST\	cargo.txt	contains value, holding costs and order costs for all commodities. For a description of the content of the file, and the units of parameters see Section 4.5.
INPUT\COST\	PilotFees.txt	specifies the fees at each terminal [SEK per vehicle]
INPUT\COST\	Vhcls_dry_bulk.txt	contains dry bulk parameters for each vehicle type, such as capacity, cost per hour, costs per km, containerloadtime, etcetera. Content of the file is similar to <code>Vhcls_gen_cargo.txt</code> described in Section 4.5.
INPUT\COST\	Vhcls_gen_cargo.txt	contains general cargo parameters for each vehicle type, such as capacity, cost per hour, costs per km, containerloadtime, etcetera. For a description of the content of the file, and the units of parameters see Section 4.5.
INPUT\COST\	Vhcls_liq_bulk.txt	contains liquid bulk parameters for each vehicle type, such as capacity, cost per hour, costs per km, containerloadtime, etcetera. Content of the file is similar to <code>Vhcls_gen_cargo.txt</code> described in Section 4.5.
INPUT\LOS\	FreqAir.314	frequency for air mode [#transports/week]
INPUT\LOS\	FreqCombi.314	frequency for combi mode [#transports/week]
INPUT\LOS\	FreqContainerVessel.314	frequency for container vessels [#transports/week]
INPUT\LOS\	FreqLorry.314	frequency for lorries [#transports/week]
INPUT\LOS\	FreqOtherVessel.314	frequency for other vessels [#transports/week]

INPUT\LOS\	FreqRailFerry.314	frequency for rail ferry [#transports/week]
INPUT\LOS\	FreqRoadFerry.314	frequency for road ferry [#transports/week]
INPUT\LOS\	FreqRoRoVessel.314	frequency for RoRo vessel [#transports/week]
INPUT\LOS\	FreqSystem.314	frequency for system [#transports/week]
INPUT\LOS\	FreqWaggonload.314	frequency for wagonload [#transports/week]
INPUT\LOS\	v101_dist.314	distance matrix for vehicle type 101 [km]
INPUT\LOS\	v101_ddist.314	Domestic distance matrix for vehicle type 101 [km]
INPUT\LOS\	v101_timeh.314	time matrix for vehicle type 101 [hour]
INPUT\LOS\	v101_xkr.314	extra costs matrix for vehicle type 101 [kSEK]
...
INPUT\LOS\	v401_dist.314	distance matrix for vehicle type 401 [km]
INPUT\LOS\	v401_ddist.314	Domestic distance matrix for vehicle type 401 [km]
INPUT\LOS\	v401_timeh.314	time matrix for vehicle type 401 [hour]
INPUT\LOS\	v401_xkr.314	extra costs matrix for vehicle type 401 [kSEK]
INPUT\NODES\	nodes.txt	contains general node information for all commodities. For a description of the content of a node file and the units of parameters see Section 4.5
INPUT\NODES\	transferroadroad.txt	Contains a dummy variable indicating whether or not road-road transfers are allowed
INPUT\NODES\	transferroadtrain.txt	Contains a dummy variable indicating whether or not road-train transfers are allowed
INPUT\NODES\	transferroadsea.txt	Contains a dummy variable indicating whether or not road-sea transfers are allowed
INPUT\NODES\	transferroadcombi.txt	Contains a dummy variable indicating whether or not road-combi transfers are allowed
INPUT\NODES\	transferroadair.txt	Contains a dummy variable indicating whether or not road-air transfers are allowed
INPUT\NODES\	transferroadroadferry.txt	Contains a dummy variable indicating whether or not road-road-ferry transfers are allowed
INPUT\NODES\	Transferseasea.txt	Contains a dummy variable indicating whether or not sea-sea transfers are allowed
INPUT\NODES\	transfercombisea.txt	Contains a dummy variable indicating whether or not Combi-sea transfers are allowed
INPUT\NODES\	transferwagonloadrailferry.txt	Contains a dummy variable indicating whether or not wagonload-railferry transfers are allowed
INPUT\NODES\	transferwagonloadsea.txt	Contains a dummy variable indicating whether or not wagonload-sea transfers are allowed
INPUT\NODES\	transferfeedertrainwagonload.txt	Contains a dummy variable indicating whether or not feeder train-wagonload transfers are allowed
INPUT\NODES\	transfersystemtrainsea.txt	Contains a dummy variable indicating whether or not system train-sea transfers are allowed
INPUT\NODES\	Directsea.txt	Contains a dummy variable indicating whether or not direct sea access is allowed
INPUT\NODES\	directsystemtrain.txt	Contains a dummy variable indicating whether or not direct system train access is allowed
INPUT\NODES\	directfeedertrain.txt	Contains a dummy variable indicating whether or not direct feeder train access is allowed
INPUT\NODES\	directwagonload.txt	Contains a dummy variable indicating whether or not direct wagonload access is allowed
INPUT\NODES\	containerhandling.txt	Contains a dummy variable indicating whether or not direct containers can be handled
INPUT\PWC\	pwc_01.txt	producer demand matrices, with the annual demand (flows from producers or wholesales to consumer between zones) for commodity type 1. [annual demand in tonnes]
INPUT\PWC\
INPUT\PWC\	pwc_35.txt	producer demand matrices, with the annual demand (flows from producers or wholesales to consumer between zones) for commodity type 1 [annual demand in tonnes]
Buildchain\OUTPUT\	Chains1.dat	logistic chains that were generated for commodity type 1 in the BuildChain program. The structure of this file is explained in Section 4.3
Buildchain\OUTPUT\

Buildchain\OUTPUT\Chains35.dat

logistic chains that were generated for commodity type 1 in the BuildChain program. The structure of this file is explained in Section 4.3

5.3 Output files

The prime result of the `ChainChoice` procedure is logistic chains for commodity flows between an OD pair. These chains include detailed information on the vehicle types that are used and the shipment sizes. Below an overview is given of the output files that are generated for each commodity group. These output data contain disaggregate results. For analysis of aggregate data the post-processing procedure `Extract` is available, that will be described in Chapter 6.

Table 5-2: Overview of output files

DIR	Filename	Description
ChainChoi\OUTPUT\	ChaiChoi1_01.out	contains the single chosen best chain for each f2f flow for commodity type 1. The content of the file is illustrated in Figure 5-1
ChainChoi\OUTPUT\
ChainChoi\OUTPUT\	ChaiChoi35_01.out	contains the single chosen best chain for each f2f flow for commodity type 35. The content of the file is illustrated in Figure 5-1
ChainChoi\OUTPUT\	ChaiChoi1_02.out	contains the second best chain for each f2f flow for commodity type 1. The content of the file is illustrated in Figure 5-1
ChainChoi\OUTPUT\
ChainChoi\OUTPUT\	ChaiChoi35_02.out	contains the second best chain for each f2f flow for commodity type 35. The content of the file is illustrated in Figure 5-1
ChainChoi\OUTPUT\	ChaiChoi1_data_01.out	contains additional data (DirectAccess indicator) for the single chosen best chain for each f2f flow for commodity type 1.
ChainChoi\OUTPUT\
ChainChoi\OUTPUT\	ChaiChoi35_data_01.out	contains additional data (DirectAccess indicator) for the single chosen best chain for each f2f flow for commodity type 35.
ChainChoi\OUTPUT\	ChaiChoi1_data_02.out	contains additional data (LoadingCosts) for the single chosen best chain for each f2f flow for commodity type 1.
ChainChoi\OUTPUT\
ChainChoi\OUTPUT\	ChaiChoi35_data_02.out	contains additional data (LoadingCosts) for the single chosen best chain for each f2f flow for commodity type 35.
ChainChoi\OUTPUT\	ChaiChoi1_data_03.out	contains additional data (WaitTimeCosts) for the single chosen best chain for each f2f flow for commodity type 1.
ChainChoi\OUTPUT\
ChainChoi\OUTPUT\	ChaiChoi35_data_03.out	contains additional data (WaitTimeCosts) for the single chosen best chain for each f2f flow for commodity type 35.
ChainChoi\OUTPUT\	ChaiChoi1_data_04.out	contains additional data (Main ¹⁴ Utilization Rate) for the single chosen best chain for each f2f flow for commodity type 1.
ChainChoi\OUTPUT\
ChainChoi\OUTPUT\	ChaiChoi35_data_04.out	contains additional data (Main Utilization Rate) for the single chosen best chain for each f2f flow for commodity type 35.
ChainChoi\OUTPUT\	ChaiChoi1_data_05.out	contains additional data (NrVhcls main leg) for the single chosen best chain for each f2f flow for commodity type 1.
ChainChoi\OUTPUT\
ChainChoi\OUTPUT\	ChaiChoi35_data_05.out	contains additional data (NrVhcls main leg) for the single chosen best chain for each f2f flow for commodity type 35.
ChainChoi\OUTPUT\	ChaiChoi1_data_08.out	contains information on the conditions that are satisfied in the avail-function for commodity type 1.
ChainChoi\OUTPUT\
ChainChoi\OUTPUT\	ChaiChoi35_data_08.out	contains information on the conditions that are satisfied in the avail-function for commodity type 35.

¹⁴ The main leg refers to the longest leg of a chain

DIR	Filename	Description
ChainChoi\OUTPUT\	ChaiChoi1.log	logfile with description of progress during chain choice procedure for commodity type 1
CHAINCHOI\OUTPUT\...		...
CHAINCHOI\OUTPUT\	ChaiChoi35.log	logfile with description of progress during chain choice procedure for commodity type 35
CHAINCHOI\OUTPUT\	ChaiChoi1.rep	contains an aggregate report for commodity 1
CHAINCHOI\OUTPUT\...		...
CHAINCHOI\OUTPUT\	ChaiChoi35.rep	contains an aggregate report for commodity 35
CHAINCHOI\OUTPUT\	VhclRep01.rep	contains an aggregate costs report for commodity 1
CHAINCHOI\OUTPUT\...		...
CHAINCHOI\OUTPUT\	VhclRep35.rep	contains an aggregate costs report for commodity 35
CHAINCHOI\OUTPUT\	chainchoi1.cst	Contains a detailed cost log for commodity 1 for a selection of relations
CHAINCHOI\OUTPUT\...		...
CHAINCHOI\OUTPUT\	Chainchoi35.cst	Contains a detailed cost log for commodity 1 for a selection of relations
CHAINCHOI\OUTPUT\	consol1_D	contains the level of consolidation of commodity type 1 between node pairs for transport mode D
CHAINCHOI\OUTPUT\...		...
CHAINCHOI\OUTPUT\...		...
CHAINCHOI\OUTPUT\	consol35_R	contains the level of consolidation of commodity type 35 between node pairs for transport mode R
CHAINCHOI\OUTPUT\	Volume1_D	contains the total volume of commodity type 1 between node pairs for transport mode D
CHAINCHOI\OUTPUT\...		...
CHAINCHOI\OUTPUT\...		...
CHAINCHOI\OUTPUT\	Volume 35_R	contains the total volume of commodity type 35 between node pairs for transport mode R

Below the output is illustrated for a sample from the output file for commodity type 17. Again, the chosen chains are illustrated for origin/destination pair 711400 and 716000. The output file shows 4 f2f commodity flows. The second column shows the number of firms in that specific firm size relation group (3x3). In the first group there are 3 f2f flows. Each flow has a size of 1.4191 tonnes. The best frequency that was chosen in the optimization proofs to be 2 shipments. Next the transport and total costs are specified. The chosen chain type is 'A' (=Lorry). The best vehicle type within this transport mode is 104 (=heavy lorry 25-40 ton).

Figure 5-1: Sample of results in output file ChainChoi17.out, with header description

Key	Nrelat	Ton2 reciev. [tonne]	Best freq [#] [SEK]	Best transp. costs [SEK]	Best cost total	Best chain type	orig node	dest node	Best vehicle type	N of vehicles [#vehicles]
33	3	1.4191	2	1559.4	2978.1	A	711400	716000	104	1
34	4	1.1753	2	1550.5	2772.9	A	711400	716000	104	1
35	3	0.9743	2	1543.2	2603.8	A	711400	716000	104	1
36	2	1.6139	2	1566.4	3142.0	A	711400	716000	104	1

Each line in the output file represents one chose logistic chain. Please note that all chosen chains in Figure 5-1 only contain one leg. If a chosen chain consists of multiple legs, the output file has extra columns for a transfer node node, the best vehicle type, and N of vehicles, for each additional link.

The report files (*.rep) contain aggregate reports for a commodity type. It reports statistics such as number of shipments and vehicles average load factors, and average distances. Statistics are aggregated to vehicle type, and to chain types. The structure of the file can not be illustrated in a figure, but the report file contains descriptions that explain the labels of the attributed listed in the files.

The log files (*.log) contain warnings that occurred during execution of the module for the specific commodity type. A warning is written when no chains are available for an OD pair but for which the PWC matrix does specify an OD flow.

5.4 Description

The `ChainChoice` procedure optimizes the logistic costs. This optimization is based on a choice for one of the available transport chains, and a choice for shipment sizes. The optimal shipment sizes are determined by optimizing the consolidation and distribution of commodity flows at transfer points. This optimization is reached by an iterative procedure as described in Chapter 3 and visualised in Figure 3-2.

In the first iteration of the model, that also includes generating the available transport chains, we use a consolidation factor of 75% (this is just a starting point): for all consolidated legs of a transport chains (that is legs coming after a consolidation centre) we assume that 75% of the vehicle capacity is used, and the shipment studied only has to pay a costs proportional to its share in this total load. This is needed to calculate the total logistics cost of transport chains that use rail, sea, airplane or consolidated road vehicles. In the second iteration, the consolidation factor is based on the potential for consolidation and observed terminal throughput data, as described below. In the third iteration, we use the OD legs from the previous model run as starting point for the level of consolidation between terminal pairs.

The flowchart of the procedure is illustrated in Figure 5-2. For a more detailed description of the `ChainChoice` procedure see the Method report on the logistics module (Significance, 2010).

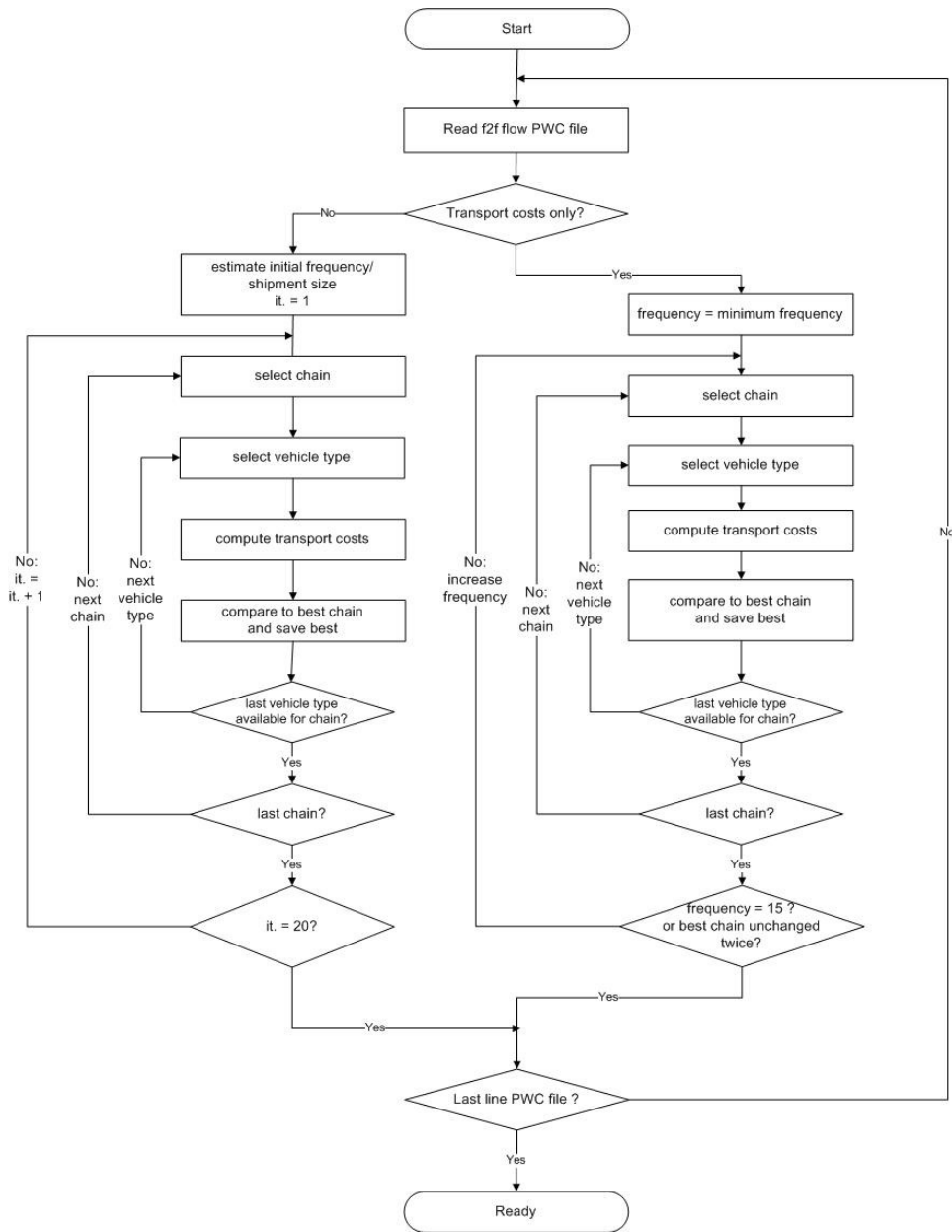


Figure 5-2: Flowchart ChainChoice procedure

5.5 Controlling the ChainChoice procedure

The parameters and input files for the ChainChoice procedure are controlled with commodity specific control files. This section describes the usage of this control file and gives an overview the input files. Below an example is given for the control file for commodity 1.

Figure 5-3: Example of a ChainChoice control file: [Chainchoi1.ct1]

```
INCL=chainchoi.ct1
COMMODITY=1
OPTIP=0
OPTIW=0
MFREQ=1
DIRACC=1,0,0,1,0,0,1,1,1,1
PWC=..\Input\PWC\PWC_01.txt
CHAINS=..\BuildChain\OUTPUT\Chains1.dat
VHCL=..\Input\Cost\vhcls_dry_bulk.txt
VHCLA=104,105
VHCLD=201
VHCLE=202
VHCLF=207,208,209
VHCLB=101,102,103
VHCLC=104,105
VHCLG=202
VHCLH=207,208,209
VHCLI=204
VHCLT=205
VHCLU=206
VHCLM=305,306,307,308,309,310,311,312,313,314,315,316,317
VHCLN=315,316
VHCLQ=317
VHCLP=318,319,320
VHCLQ=321
VHCLR=401
LOADFAC=OUTPUT\ChainChoil.fac
COST=OUTPUT\chainchoil.cst
REP=OUTPUT\ChainChoil.rep
```

```
LOG=OUTPUT\ChainChoil.log
```

Explanation of parameters:

INCL	: reference to the control file that contains the common settings for all commodities. An example of this file is given in Figure 5-4.
COMMODITY	: commodity code
OPTIP	: sets the producer costs in the optimization procedure. 0 = include all costs, 1 = include transport costs only
OPTIW	: sets the wholesale costs in the optimization procedure. 0 = include all costs, 1 = include transport costs only
MFREQ	: minimum frequency that is used when the default frequency is not available in the vehicle costs file (path to this file is specified in parameter VHCL). These frequencies are used as an initial value in the shipment size optimization procedure. [#transport/week]
DIRACC	: switches to make direct access available for specific f2f sizeclass combinations. 0= unavailable, 1 = available. The first character is for singular flows. The second character for small to small firms. The last character for large to large firms
PWC	: path to file with producer demand matrices for the specific commodity.
CHAINS	: path to file with logistic chains for each chain type available to the respective commodity type. This file is generated by the <code>BuildChain</code> procedure. For a description of this file see section 4.3.
VHCL	: path to file with vehicle data for the specific commodity type. Parameters to the logistic cost function (equation 1). This input file is described in detail in section 4.5.
VHCLA	: the vehicle type number for mode A. This value is set by the user but should correspond to the <code>BuildChain</code> control file.
LOADFAC	: Path for output file containing the load factors on each relation
COST	: Path for output file containing detailed cost log for selection of relations
REP	: path for output file with some aggregate statistics.
LOG	: path for logfile that contain error description that have occurred during program execution. If execution has gone well, this file is empty.

Figure 5-4: Example of a common ChainChoice control file: [Chainchoi.ct1]

```

LOGCTL=1
LOGFLS=1
LOGCST=1
LSTCNT=2
STUFF=18
INTEREST=0.1
CONSOL=0.05,0.95
CONSOLA=0.05,0.5
CONSOLB=0.05,0.5
CONSOLC=0.05,0.5
CONSOLS=0.5,0.9
INDIVIDUAL_OD_LEG_OPTIMIZE=1
ALL_LORRY_TYPE_CONSOL=1
MINIMUM_ANNUAL_TONNE_DEMAND_4_FREQ_OPTIMIZE=5.0
DATA=1,2,3,4,5
TYPES=..\Input\chaintype.lis
NODES=..\Input\Nodes
PILOTFEES=..\Input\Cost\pilotfees.txt
CARGO=..\Input\Cost\cargo.txt
LOADDIR=..\Input\Los
OUTDIR=.\OUTPUT
CONSOLDIR=.\OUTPUT\CONSOL
SELECT=Select.dat

```

Explanation of parameters:

LOGCTL	: Indicator (0/1) that determines whether or not CTL file settings will be logged in the log file.
LOGFLS	: Indicator (0/1) that determines whether or not input file information will be logged in the log file.
LOGCST	: Indicator (0/1) that determines whether or not cost parameters will be logged in the log file.
LSTCNT	: sets the number of output files: 1= only best chains; 2 = best chains and second best chains; ...
STUFF	: costs for stuffing and stripping of containers at the origin and destination of a chain [SEK per tonne]
INTEREST	: Interest rate used in cost calculations [%/year]
CONSOL	: gives the default upper and lower bounds for the consolidation factors ranking output
CONSOL<mode>	: gives the default upper and lower bounds for the consolidation factors ranking output for mode <mode>

ALL_LORRY_TYPE_CONSOL	: 0/1 switch that determines whether or not consolidation is allowed for all lorry types ¹⁵
INDIVIDUAL_OD_LEG_OPTIMIZE	: 0/1 switch that determines whether or not the optimization is done for each chain leg individually
MINIMUM_ANNUAL_TONNE_DEMAND_4_FREQ_OPTIMIZE	: Minimum demand for frequency optimization
DATA	: comma separated list of additional output variables of the best chain
TYPES	: Path to file containing all available chain types
NODES	: Path to directory containing the nodes input files. Each nodes file contains a single node variable for all commodities. The file names of the nodes files are fixed.
PILOTFEES	: path to file with pilot fees. Parameters to the logistic cost function (equation 1). For a description see below.
CARGO	: path to file with cargo values, holding costs and order costs. Parameters to the logistic cost function (equation 1). This input file is described in detail in section 4.5.
LOSDIR	:Path to directory containing the Level of service files. The level of service files have fixed names, referring to the applicable vehicle type.
OUTDIR	:Path to output directory. All output files have fixed names.
CONSOLIDIR	: Path to consolidation output directory. All output files have fixed names.
SELECT	: path to input file containing the collection of relations that should be included in the <code>detailed cost log</code> .

Explanation of input files

All input files were already discussed in Chapter 4. For this description see section 4.5

¹⁵ Regardless of this setting a separate mode S (consolidated heavy lorry) is distinguished in the mode. Because this mode is only allowed between terminals and on international relations, a consolidation factor range may be used that differs from the other lorry modes (A,B,C).

6.1 Introduction

The `Extract` procedure generates vehicle demand matrices. This chapter describes the procedure and gives an overview of the input files it uses and what output files are produced. Please note that the output reports are different in the Swedish and Norwegian implementation. The Swedish vehicle matrices are post processed outside of the logistic module. To avoid double processing some output reports were excluded from the Swedish implementation.

6.2 Input files

The `extract` procedure loops through all vehicle types. During its execution it uses the chosen logistic chains from the `ChainChoice` procedure, the LOS files for each vehicle type and a list of all nodes in the network. Below an overview is presented of all files that are used in the `Extract` procedure.

Folder	Filename	Description
\EXTRACT\	emptyfrac.dat	Empty vehicle fractions per vehicle type
\EXTRACT\	Extract101.ctf	control file for running the extract procedure for vehicle type 101. Structure of file is explained in Section 0
\EXTRACT\
\EXTRACT\	Extract401.ctf	control file for running the extract procedure for vehicle type 401. Structure of file is explained in Section 0
INPUT\NODES\	AllNodes.dat	contains a list of all nodes with their ID's and names. Contains no additional information on transfer options.
INPUT\LOS\	v101_dist.314	distance matrix for vehicle type 101 [km]
...
INPUT\LOS\	v401_dist.314	distance matrix for vehicle type 401 [km]
CHAINCHOI\OUTPUT\	Chainchoi1_01.out	contains the single chosen best chain for each f2f flow for commodity type 1. The content of the file is illustrated in Figure 5-1
...
CHAINCHOI\ OUTPUT\	Chainchoi35_01.out	contains the single chosen best chain for each f2f flow for commodity type 35. The content of the file is illustrated in Figure 5-1

6.3 Output files

The Extract procedure generates vehicle matrices (see overview below).

Folder	Filename	Description
\\EXTRACT\ OUTPUT\	ODVhcl101.314	Vehicle matrix for vehicle type 101. Structure of file is explained below
\\EXTRACT\ OUTPUT\
\\EXTRACT\ OUTPUT\	ODVhcl401.314	Vehicle matrix for vehicle type 401. Structure of file is explained below

The vehicle demand matrices are in EMME/2 format. The file specifies the location, date and time of execution, and the control file that was used. The fourth line in the file gives the matrix ID code (in the example below: mf39). A data line contains three pieces of information: origin node, destination node, and number of vehicles. The content of a typical output file is illustrated in Figure 6-1.

Figure 6-1: Sample of output file od_vhcl101.314 the vehicle matrices for vehicle type 101.

```

c D:\SIKA\EXTRACT\Extract.exe 06/12/2007 10:35:02
c Control file: extract101.ct1
t matrices
a matrix=mf11 start 0
711400 711401:27.0000
711400 712000:664.4220
711400 712500:40.5000
711400 712600:165.0000
711400 712700:328.5000
711400 712800:197.6600
711400 713600:1971.5742
711400 714000:224.5102
711400 716000:6.0000
711400 718000:1545.0000
711400 718100:1286.8432
711400 718101:38.0000
711400 718200:24.0000
711400 718400:148.5000
711400 718600:4.5000
711400 718700:6.0000
711400 718800:5385.9671
711400 719200:966.0669
711400 730500:1.5000
.....
    
```

6.4 Controlling the Extract procedure

The extract procedure is controlled with vehicle type specific control files. These control files are used to set the parameters and paths to the necessary input and output files. Below the control file for vehicle type 105 (=heavy lorry, containerized) is illustrated. The chosen chains for all commodity types are all included in the vehicle matrices for a specific vehicle type.

Figure 6-2: Example of an Extract control file: [Extract.ct1]

```

TYP=105
ID=mf15
EMPTY=1
ASYM=50
NFlOWS=34
NODES=..\Input\Nodes\AllNodes.dat
EMPTYFR=emptyfrac.dat
DIST=..\Input\LOS\v105_dist.314
FLOW1=..\chainchoi\Output\chainchoi1.out
FLOW2=..\chainchoi\ Output\chainchoi2.out
...
...
FLOW33=..\chainchoi\ Output\chainchoi34.out
FLOW34=..\chainchoi\ Output\chainchoi35.out
VHCL= Output\OD_Vhcl105.314
TONNES= Output\OD_Tonnes105.314

```

The parameters from the control file are described below.

TYP	: vehicle type number
ID	: matrix name that is written into the EMME/2 output file
EMPTY	: switch for activating/deactivating the calculation of empty vehicles. 0 = empty vehicles are not included, 1 = empty vehicles are derived. The procedure for determining the number of empty vehicles is described below.
ASYM	: The threshold distance above which asymmetric flows will generate empty vehicles.
NFlOWS	: the number of commodity flows that is analysed for vehicle flows. In a standard application this parameter is set to 34, and all commodity flows that are implemented in the current Swedish model are analysed. If a user might decide to determine vehicle matrices for one specific commodity type, the parameter NFlOWS is set to 1. In this setting only one chosen chain path has to be specified: parameter FLOW1. The paths FLOW2 to FLOW34 are abundant in such a case.
NODES	: path to file with node information on all nodes. This input file is similar to the node file described in detail in section 4.5.
EMPTYFR	: path to file with empty vehicle fractions.
DIST	: path to file with distance matrix for the specific vehicle type.
FLOW1	: path to file with chosen logistic chains for commodity type 1. This input file is described in detail in section 5.3.
VHCL	: path for vehicle matrix output file, in EMME/2-format. The content of this file is described in Section 6.3.
TONNES	: path for tonnes matrix output file, in EMME/2-format. The content of this file is similar to the VHCL output file, described in Section 6.3.

Explanation of input files:

The emptyfrac.dat file contains the distance dependent empty vehicle fractions per vehicle type:

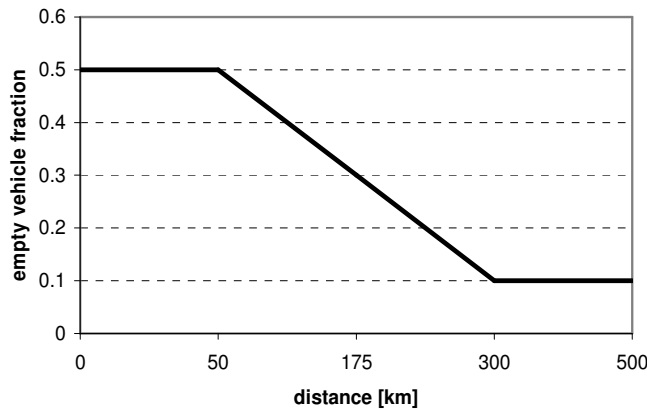
Table 6-1: Contents of emptyfrac.dat:

Column	Parameter	Description
1	Vehicle Type	Vehicle type number
2	Distance	Distance threshold, see explanation below
3	Fraction	Empty vehicle fraction

Below the lowest distance threshold and above the highest distance threshold the empty vehicle fractions are constant. In between two distance thresholds a linear interpolation is used. For example, the following lines in emptyfrac.dat:

```
101 50 0.5
101 300 0.1
```

imply that the empty vehicle fraction for vehicle type 101 equals 0.5 below 50 km and 0.1 above 300 km. In between 50 km and 300 km there is a linear decrease of the empty vehicle fraction from 0.5 to 0.1. This distance function is visualised below.



Empty vehicles

The empty vehicle flow consists of two components. First of all, vehicles may return empty because the reverse flow is unbalanced. This component is taken into account above the ASYM-threshold. On top of that, a fraction of the vehicles will always return empty, even if a flow is perfectly balanced with its reverse flow. This fraction is specified in the emptyfrac.dat file.

To illustrate the calculation consider an OD-relation:

Distance = 70 km
OD-flow = 120 vhcls
DO-flow = 100 vhcls
Empty vehicle fraction at 70 km = 0.1
ASYM threshold = 50

The distance (70) is above the ASYM threshold value (50) so $120-100=20$ vehicle will return empty from D to O due to asymmetric flows. On top of that, 10% of the vehicles will return empty, so the empty vehicle flows are calculated as follows:

OD-flow empties = $0.1*100 = 10$
DO-flow empties = $0.1*120 + 20 = 32$

References

Edwards (2007) Base matrix for SAMGODS and disaggregation of P/C flows to firm to firm aggregates; Vägverket Konsult, Stockholm.

RAND Europe and SITMA (2005) The development of a logistics module in the Norwegian and Swedish national model systems, deliverable 4: final progress report on model development, PM-1968-SIKA, RAND Europe, Leiden.

RAND Europe and SITMA (2006) Documentation and clarification of deliverable 4 and the associated program delivery for a logistics module in the Norwegian and Swedish national model systems, deliverable 4a, PM-2055-SIKA, RAND Europe, Leiden.

Significance (2007) Technical report on the further development of a logistics module in the Norwegian and Swedish national freight model systems, Deliverable 5 for the Samgods group and the working group for transport analysis in the Norwegian national plan, Significance, Leiden.

Significance (2013) Method Report – Logistics Model in the Swedish National Freight Model System (Version 2.1), Deliverable 6B for Trafikverket, Significance, The Hague.

Appendix A: Rail Capacity Management

A.1 Introduction

A transport demand is defined by a certain amount of goods needing to be transported from an origin to a destination. The purpose of the logistic model is to find the most economic transport chain for each transport demand within Sweden as well as from Sweden to destinations abroad, and to Sweden from origins abroad.

The standard logistic model reads all transport demands from PWC matrices. In these matrices all transport volumes, origins and destinations are given. In a first step the program BuildChain finds all possible transport chains. In the next step the program ChainChoi optimizes each of the candidate chains, and selects the most economic one. It also lists the best alternative transport chains.

One limitation of the standard logistic model is that it does not take into account the capacity limitations of rail transport chains. For this purpose the logistic model for rail capacity management, LogMod4RCM, has been developed. At the end of the rail capacity modelling process the new partial solutions needs to be merged with the original solutions from the standard logistic model, and all statistics recalculated. That is the task for the LP2CC program.

A.2 LogMod4RCM

A.2.1 Program purpose

Just like the standard logistic model LogMod4RCM consist of two programs, BuildChain4RCM and ChainChoi4RCM. These programs finds the most economic transport chain in much the same way as the standard logistic model, with two major differences: it only calculates the cost for those transport demands that uses rail in the solutions from the standard logistic model, and it adds a marginal cost calculated to raise the cost for rail transport links where capacity is a limitation. In the resulting transport chains rail transport demand and rail capacity should be in balance.

A.2.2 Function

LogMod4RCM uses the same algorithms to calculate costs and find the most economic transport chain as the standard logistic model. The difference lies in the selection of transport demands, and in the added marginal cost for certain train links. In standard logistic model the transport demands are read from PWC matrix files. LogMod4RCM loops over the transport chains specified in the file named by the JLIST control file parameter. This file lists all transport chains from the standard logistic model solution that uses rail. The transport demands are then read from an output file from the standard logistic model solution, ChainChoiXX_data_06.out.

A.2.3 Program control

Command line parameter

Both BuildChain4RCM and ChainChoi4RCM read the control file name from the command line. The parameter RDVOLUMES=YES indicates that volumes and consolidation factors are read from the CONSOLXX_Y.314 and VOLUMEXX_Y.314 files written by standard LogMod, like the final standard ChainChoi iteration.

Bat file example:

```
cd buildchain
time /t
BuildChain4RCM.exe buildchain01.ctf
cd..
cd ChainChoi
time /t
ChainChoi4RCM.exe chainchoi01.ctf /RDVOLUMES=YES
cd..
time /t
```

Control file parameters

The program uses the ctl files from standard LogMod with a few new parameters added. These parameters are used by both BuildChain4RCM and ChainChoi4RCM.

CCLIST specifies the ChainChoiXX_data_06.out file to use. CHAINSRM specifies the BuildChain4RCM output chains file, also used as input for ChainChoi4RCM.

JLIST specifies the JLISTA.dat file, containing indexes and keys for all chains to recalculate.

Example:

```
CCLIST=..\ChainChoi\OUTPUT\ChainChoi01_data_06.out
CHAINSRM=OUTPUT\Chains01_RCM.dat
JLIST=..\RCM\JLISTA.DAT
```

A.2.4 I/O

Input files

Nodes, costs and vehicle data are read from the same files that standard LogMod uses. Transport demand is read from ChainChoiXX_data_06.out files.

Output files

LogMod4RCM generates the same set of files as standard LogMod, with the label RCM in the file name before the file extension, as in ChainChoi01_01_RCM.out. These files only contain the recalculated chains.

A.3 LP2CC

A.3.1 Program purpose

The LP2CC program merges the new partial solutions needs with the original solutions from the standard logistic model, and all statistics recalculated. It generates a complete set of LogMod output files where the chains that are recalculated in RCM replaces the original chains.

A.3.2 Function

LP2CC uses the transport solutions found by the standard logistic model and the alternative solutions calculated by the rail capacity modelling process. It then recalculates all costs and statistics, and generates a complete new set of output files for the merged solution. The chains are read from the ChaiChoiXX_01LPX.out files, where the optimal LP-solution columns are inserted. Some additional data are also read from ChainChoiXX_data_06.out. These data are then entered into the ChainChoi optimization process as the best chain, and recalculated using the calculation routines from ChainChoi.

A.3.3 Program control

Command line parameter

The only command line parameter is the control file name.

Bat file example

```
cd ChainChoi
LP2CC.exe chainchoi01.ctl
cd..
```

Control file parameters

The program uses the ctl files from ChainChoi. It has no specific ctl file parameters, only parameters also used by ChainChoi.

A.3.4 I/O

Input files

Nodes, cost and vehicle data are read from the same files as ChainChoi uses.

Input for chain calculation is read from ChaiChoiXX_01LPX.out and ChainChoiXX_data_06.out

Output files

The output generated is the full set of output from LogMod, with the label FIN added before the file name extension. The exception is that only one chain is generated for each PWC demand. There are no files corresponding to the standard LogMod output ChainChoiXX_02.out to ChainChoiXX_05.out.

Appendix B: Dimensions in the model

Table A-1: Overview of commodity types:

Nr	Commodity	Aggregate commodity	Average shipment size
1	Cereals	Dry bulk	41.0
2	Potatoes, other vegetables, fresh or frozen, fresh fruit	Dry bulk	3.8
3	Live animals	Dry bulk	3.8
4	Sugar beet	Dry bulk	0.3
5	Timber for paper industry (pulpwood)	Dry bulk	41.2
6	Wood roughly squared or sawn lengthwise, sliced or peeled	Dry bulk	9.2
7	Wood chips and wood waste	Dry bulk	122.8
8	Other wood or cork	Dry bulk	43.4
9	Textiles, textile articles and manmade fibres, other raw animal and vegetable materials	General cargo	0.2
10	Foodstuff and animal fodder	General cargo	1.8
11	Oil seeds and oleaginous fruits and fats	Liquid bulk	14.1
12	Solid mineral fuels	Liquid bulk	164.5
13	Crude petroleum	Liquid bulk	19 739.1
14	Petroleum products	Liquid bulk	103.1
15	Iron ore, iron and steel waste and blast-furnace dust	Dry bulk	4212.2
16	Non-ferrous ores and waste	Dry bulk	135.9
17	Metal products	General cargo	12.9
18	Cement, lime, manufactured building materials	Dry bulk	7.2
19	Earth, sand and gravel	Dry bulk	20.5
20	Other crude and manufactured minerals	Dry bulk	29.1
21	Natural and chemical fertilizers	Dry bulk	55.6
22	Coal chemicals	Liquid bulk	3.2
23	Chemicals other than coal chemicals and tar	Dry bulk	3.1
24	Paper pulp and waste paper	Dry bulk	173.9
25	Transport equipment, whether or not assembled, and parts thereof	General cargo	1.7
26	Manufactures of metal	General cargo	0.9
27	Glass, glassware, ceramic products	General cargo	1.1
28	Paper, paperboard; not manufactures	Dry bulk	23.3
29	Leather textile, clothing, other manufactured articles than paper, paperboard and manufactures thereof	General cargo	0.6
30	Mixed and part loads, miscellaneous articles	General cargo	No PWC flow for this commodity
31	Timber for sawmill	Dry bulk	40.9
32	Machinery, apparatus, engines, whether or not assembled, and parts thereof	General cargo	18.2
33	Paper, paperboard and manufactures thereof	General cargo	0.3

34	Wrapping material, used	Dry bulk	0.6
35	Air freight (2006 model)	General cargo	2.9

Table A-2: Overview of vehicle type numbers, and aggregate modes for container transport and non-container transport.

Aggregate mode		ModeNr	VhclNr	Vehicle type
Containers	Heavy lorry	A	104	Lorry HGV 25-40 ton
			105	Lorry HGV 25-60 ton
	Kombi train	D	201	Kombi train
	Feeder train	E	202	Feeder/shunt train
	Wagonload train	F	203	Wagon load train
	Direct Sea	J	301	Container vessel 5 300 dwt ¹
			302	Container vessel 16 000 dwt ¹
			303	Container vessel 27 200 dwt ¹
			304	Container vessel 100 000 dwt ¹
			305	Other vessel 1 000 dwt
			306	Other vessel 2 500 dwt
			307	Other vessel 3 500 dwt
			308	Other vessel 5 000 dwt
			309	Other vessel 10 000 dwt
			310	Other vessel 20 000 dwt
			311	Other vessel 40 000 dwt
			312	Other vessel 80 000 dwt
			313	Other vessel 100 000 dwt
			314	Other vessel 250 000 dwt
			315	Ro/ro vessel 3 600 dwt
Feeder vessel	K	301	Container vessel 5 300 dwt	
		315	Ro/ro vessel 3 600 dwt	
		316	Ro/ro vessel 6 300 dwt	
Long-Haul vessel	L	303	Container vessel 27 200 dwt	
		304	Container vessel 100 000 dwt	
		317	Ro/ro vessel 10 000 dwt	
Non-Containers	Light Lorry	B	101	Lorry light LGV, ≤ 3,5 ton
			102	Lorry medium 3,5-16 ton
			103	Lorry medium 16-24 ton
	Heavy lorry	C / S ¹⁶	104	Lorry HGV 25-40 ton
			105	Lorry HGV 25-60 ton

¹⁶ Consolidated heavy lorry is coded as mode S in the chains file. Consolidation in heavy lorries is only available on an intermediate leg in a chain with at least three legs.

	Feeder train	G	202	Feeder/shunt train
	Wagonload train	H	203	Wagon load train
	System train	I	204	System train STAX 22,5
		T	205	System train STAX 25
		U	206	System train STAX 30
	Direct Sea	M	305	Other vessel 1 000 dwt
			306	Other vessel 2 500 dwt
			307	Other vessel 3 500 dwt
			308	Other vessel 5 000 dwt
			309	Other vessel 10 000 dwt
			310	Other vessel 20 000 dwt
			311	Other vessel 40 000 dwt
			312	Other vessel 80 000 dwt
			313	Other vessel 100 000 dwt
			314	Other vessel 250 000 dwt
			315	Ro/ro vessel 3 600 dwt
			316	Ro/ro vessel 6 300 dwt
			317	Ro/ro vessel 10 000 dwt
	Feeder vessel	N	315	Ro/ro vessel 3 600 dwt
			316	Ro/ro vessel 6 300 dwt
	Long-Haul vessel	O	317	Ro/ro vessel 10 000 dwt
	Road Ferry	P	318	Road ferry 2 500 dwt
			319	Road ferry 5 000 dwt
			320	Road ferry 7 500 dwt
	Rail Ferry	Q	321	Rail ferry 5 000 dwt
	Plane	R	401	Freight airplane

Table A-3: Transport chains used for Sweden

Number	Potential chain	Explanation
1	A	Direct transport by heavy lorry, using containers (see Table 3)
2	ADA	Heavy-lorry – Kombi-train – heavy lorry, with containers
3	ADJA	Etc.
4	ADJDA	
5	ADKL	
6	AJ	
7	AJA	
8	AJDA	
9	AKL	
10	APA	
11	B	
12	BR	
13	BRB	
14	BS	
15	BSB	
16	C	
17	CGH	
18	CGHC	
19	CGHM	
20	CH	
21	CHG	
22	CHGC	
23	CM	
24	CMC	
25	CMI	
26	CMT	
27	CMU	
28	CPC	
29	CUM	
30	GH	
31	GHC	
32	GHG	
33	GHM	
34	GHMI	

Number	Potential chain	Explanation
35	GHMT	
36	GHMU	
37	GHQH	
38	HC	
39	HG	
40	HGC	
41	I	
42	IM	
43	IMC	
44	IMHG	
45	J	
46	JA	
47	KL	
48	LK	
49	LKA	
50	LKDA	
51	M	
52	MC	
53	MHG	
54	MHGC	
55	MI	
56	MT	
57	MU	
58	RB	
59	SB	
60	T	
61	TM	
62	TMC	
63	TMGH	
64	U	
65	UM	
66	UMC	
67	UMGH	

The typical vehicles/vessels used in BuildChain for each commodity are in Table 5.

Appendix C: Parameters and variables

The overview below specifies all parameters and variables in the model. Each parameter or variable is described for what it is used, in which file it is controlled, by what module it is used, and the type and range of information (if applicable). The real data types do not require a specific precision: the program can handle real numbers with any decimal positions.

Type	Var. / param.	Description	Controlled where	Used by	Type [range]
P	FACTOR	Initial consolidation factor	Commodity.bat	BuildChain (in 1 st iteration)	real [0-1]
P	RDVOLUMES	Boolean indicating if transport volumes are available from previous iteration	Commodity.bat	ChainChoice	Boolean
P	UPDATE	switch to set allocation mechanism for chain assignment of shipments	Commodity.bat	ChainChoice	'All' or 'chosen'
V	COMMODITY	commodity code [1..35]	BuildChain%1%.ctl ChainChoice%1%.ctl	BuildChain ChainChoice	integer [1,□□□]
P	TONNES	Typical shipment size for this commodity. This parameter is used to set the logistic cost parameter q and calculate the logistic costs for the chains [tonnes per shipment]	BuildChain%1%.ctl	BuildChain	real [0,□]
P	INTEREST	Interest rate used in cost calculations [%/year]	BuildChain%1%.ctl	BuildChain	real [0,1]
Path	TYPES	Path to file containing all available chain types	BuildChain%1%.ctl ChainChoice%1%.ctl	BuildChain ChainChoice	character
Path	NODES	Path to directory containing the nodes input files. Each nodes file contains a single node variable for all commodities. The file names of the nodes files are fixed.	BuildChain%1%.ctl ChainChoice%1%.ctl	BuildChain ChainChoice	character
Path	VHCL	path to file with vehicle data for the specific commodity type	BuildChain%1%.ctl ChainChoice%1%.ctl	BuildChain ChainChoice	character
Path	CARGO	path to file with the cargo values, holding costs and order costs used in the logistic cost function.	BuildChain%1%.ctl ChainChoice%1%.ctl	BuildChain ChainChoice	character
Path	PWC	path to file with producer demand matrices for the specific commodity	BuildChain%1%.ctl ChainChoice%1%.ctl	BuildChain ChainChoice	character
Path	PILOTFEES	path to file with pilot fees.	BuildChain%1%.ctl ChainChoice%1%.ctl	BuildChain ChainChoice	character
P	VHCLA	Typical vehicle type number for mode A	BuildChain%1%.ctl ChainChoice%1%.ctl	BuildChain ChainChoice	integer [1,□□]
P	VHCLR	Typical vehicle type number for mode R	BuildChain%1%.ctl ChainChoice%1%.ctl	BuildChain ChainChoice	integer [1,□□]
Path	LOSDIR	Path to directory containing the Level of service files	BuildChain%1%.ctl ChainChoice%1%.ctl	BuildChain ChainChoice	character
Path	CONSOLIDIR	Path to directory containing the consolidation factor matrices. The consolidation factor files have fixed names, referring to the applicable mode and commodity.	BuildChain%1%.ctl	BuildChain	character
Path	CHAINS	path for output file containing the optimal chains	BuildChain%1%.ctl	BuildChain	character
Path	CONNLIST	path to output file containing all connections on a selection of relations	BuildChain%1%.ctl	BuildChain	character
Path	SELECT	path to input file containing the collection of relations that should be included in the CONNLST-file.	BuildChain%1%.ctl	BuildChain	character
Path	LOG	path for log file	BuildChain%1%.ctl	BuildChain	character
V	Value	value of commodity, v_k [SEK / tonne]	cargo.txt	BuildChain	real [0,□□]
V	HoldingCosts	storage costs of commodity, w_k [SEK / (tonne*year)]	cargo.txt	BuildChain	real [0,□□]
V	OrderCosts	constant unit cost per order, o_k [SEK]	cargo.txt	BuildChain	real [0,□□]

Type	Var. / param.	Description	Controlled where	Used by	Type [range]
V	VesselType	vessel type, for all water transport modes. 0 = not a vessel; 1 = container vessel; 2 = ro-ro vessel; 3 = other vessel.	vhcls_gen_cargo.txt, vhcls_dry_bulk.txt, vhcls_liq_bulk.txt	BuildChain	integer [0,3]
V	Capacity	capacity of vehicle type. This influences the shipment sizes and required number of vehicles [tonne]	vhcls_gen_cargo.txt, vhcls_dry_bulk.txt, vhcls_liq_bulk.txt	BuildChain	real [0,□□]
V	HourCost	determine the link-based costs (time) for the vehicle, parameter C_{vt}^h [SEK per hour per vehicle]	vhcls_gen_cargo.txt, vhcls_dry_bulk.txt, vhcls_liq_bulk.txt	BuildChain	real [0,□□]
V	KmCost	determine the link-based costs (distance) for the vehicle C_{vt}^d [SEK per km per vehicle]	vhcls_gen_cargo.txt, vhcls_dry_bulk.txt, vhcls_liq_bulk.txt	BuildChain	real [0,□□]
V	OnFerryHourCost	link-based costs (time) on a ferry link, C_v^h [SEK per hour per vehicle]	vhcls_gen_cargo.txt, vhcls_dry_bulk.txt, vhcls_liq_bulk.txt	BuildChain	real [0,□□]
V	OnFerryKmCost	link-based distance costs on a ferry link, C_v^d [SEK per km per vehicle]	vhcls_gen_cargo.txt, vhcls_dry_bulk.txt, vhcls_liq_bulk.txt	BuildChain	real [0,□□]
V	PositioningCost	initial transport costs for vessel to arrive at the loading point for the link, $C_v^{position}$ [SEK per vehicle]	vhcls_gen_cargo.txt, vhcls_dry_bulk.txt, vhcls_liq_bulk.txt	BuildChain	real [0,□□]
V	DfltFreq	default frequency if frequency is unavailable from LOS input files [#transports/week]	vhcls_gen_cargo.txt, vhcls_dry_bulk.txt, vhcls_liq_bulk.txt	BuildChain	integer [0,□□]
V	Container-LoadTime	time costs for loading/unloading container transport at distribution and consolidation centres $t_{vt}^{load;container}$ [hour per vehicle]	vhcls_gen_cargo.txt, vhcls_dry_bulk.txt, vhcls_liq_bulk.txt	BuildChain	real [0,□□]
V	Container-LoadCost	costs for loading/unloading container transport at distribution and consolidation centres $C_{vt}^{load;container}$ [SEK per tonne]	vhcls_gen_cargo.txt, vhcls_dry_bulk.txt, vhcls_liq_bulk.txt	BuildChain	real [0,□□]
V	NonContainer-LoadTime	time costs for loading/unloading non-container transport at distribution and consolidation centres $t_{vt}^{load;non-container}$ [hour per vehicle]	vhcls_gen_cargo.txt, vhcls_dry_bulk.txt, vhcls_liq_bulk.txt	BuildChain	real [0,□□]
V	NonContainer-LoadCost	costs for loading/unloading non-container transport at distribution and consolidation centres. parameter $C_{vt}^{load;non-container}$ in transport costs function 4 [SEK per tonne]	vhcls_gen_cargo.txt, vhcls_dry_bulk.txt, vhcls_liq_bulk.txt	BuildChain	real [0,□□]
V	Vhcl-FairwayDues	Fairway costs for vehicle C_{vt}^{fairw} [SEK per vehicle]	vhcls_gen_cargo.txt, vhcls_dry_bulk.txt, vhcls_liq_bulk.txt	BuildChain	real [0,□□]
V	Tonnes-FairwayDues	Fairway costs for vehicle C_{vt}^{fairwT} [SEK per tonne]	vhcls_gen_cargo.txt, vhcls_dry_bulk.txt, vhcls_liq_bulk.txt	BuildChain	real [0,□□]
V	frequency	frequency for transport mode [in transports/week]	freq'MODE'.314	BuildChain	real [0,□□]
V	distance	distance between nodes for a vehicle type [in km]	v'vclnr'_dist.314	BuildChain	real [0,□□]
V	ddistance	domestic distance between nodes for a vehicle type [in km]	v'vclnr'_ddist.314	BuildChain	real [0,□□]
V	time	travel time between nodes for a vehicle type [in hour]	v'vclnr'_timeh.314	BuildChain	real [0,□□]
V	extra costs	extra costs between nodes for a vehicle type [in kSEK]	v'vclnr'_xkr.314	BuildChain	real [0,□□]
P	Zone code	zone code of node location, node code of terminal that provides direct access to this zone or empty ('0')	nodes.txt	BuildChain	integer [0,□□]

Type	Var. / param.	Description	Controlled where	Used by	Type [range]
V	Domestic	dummy for domestic location. 1 = domestic, 0 = non-domestic.	nodes.txt	BuildChain	integer [0, 1]
V	CostTechnoFac	Cost efficiency factor at terminal, f^{cost} [0..1]	nodes.txt	BuildChain	real [0, 1]
V	TimeTechnoFac	Time efficiency factor at terminal, f^{time} [0..1]	nodes.txt	BuildChain	real [0, 1]
V	MaxDwtContainerVessel	Load constraint of container vessels that port can handle [tonne].	nodes.txt	BuildChain	real [0, ...]
V	MaxDwtRoroVessel	Load constraint of Roro vessels that port can handle [tonne].	nodes.txt	BuildChain	real [0, ...]
V	MaxDwtOtherVessel	Load constraint of other vessels that port can handle [tonne].	nodes.txt	BuildChain	real [0, ...]
P	SeaOutput	Yearly sea output. This exogenous factor is used to determine a size ranking for sea terminals [tonne]	nodes.txt	BuildChain	[0, ...]
P	SeaContainerOutput	Yearly sea container output. This exogenous factor is used to determine a size ranking for sea terminals [tonne]	nodes.txt	BuildChain	real
P	AirOutput	Yearly air output. This exogenous factor is used to determine a size ranking for air terminals [tonne]	nodes.txt	BuildChain	[0, ...]
V	Transfer dummy's	switch to control the transfer possibilities in nodes by commodity group	transfer'modes'.txt	BuildChain	integer [0, 35]
V	direct access dummy's	switch to control the direct access possibilities in nodes by commodity group	direct'mode'.txt	BuildChain	integer [0, 35]
V	Pilotfee's	pilot fee's for each sea mode vehicle type (17) [in SEK]	pilotfees.txt	BuildChain	real [0, ...]
P	OPTIP	sets the producer costs in the optimization procedure. 0 = include all costs, 1 = include transport costs only	Chainchoi%1%.ctl	ChainChoice	integer [0, 1]
P	OPTIW	sets the wholesale costs in the optimization procedure. 0 = include all costs, 1 = include transport costs only	Chainchoi%1%.ctl	ChainChoice	integer [0, 1]
P	LSTCNT	sets the number of output files= only best chains; 2 = best chains and second best chains; ...	Chainchoi%1%.ctl	ChainChoice	integer [0, ...]
V	STUFF	stuffing costs [SEK per tonne]	Chainchoi%1%.ctl	ChainChoice	real [0, ...]
P	INTEREST	Interest rate used in cost calculations [%/year]	Chainchoi%1%.ctl	ChainChoice	real [0, ...]
P	CONSOL	gives the upper and lower bounds for the consolidation factors ranking output	Chainchoi%1%.ctl	ChainChoice	real [0, ...]
P	MFREQ	minimum frequency that is used when the default frequency is not available in the vehicle costs file. These frequencies are used as an initial value in the shipment size optimization procedure. [#transports/week]	Chainchoi%1%.ctl	ChainChoice	integer [0, ...]
P	DIRACC	switches to make direct access available for specific sizeclass combinations. 0= unavailable, 1 = available. The first character is for singular flows. The second character for small to small firms. The last character for large to large firms	Chainchoi%1%.ctl	ChainChoice	integer [0, ...]
P	DATA	comma separated list of additional output variables of the best chain	Chainchoi%1%.ctl	ChainChoice	integer [0, ...]
Path	CHAINS	path to inputfile with logistic chains for each chain type available to the respective commodity type. This file is generated by the BuildChain procedure.	Chainchoi%1%.ctl	ChainChoice	character
Path	OUTDIR	:Path to output directory. All output files have fixed names.	Chainchoi%1%.ctl	ChainChoice	character
Path	SELECT	path to input file containing the collection of relations that should be included in the CONNLST-file.	Chainchoi%1%.ctl	ChainChoice	character
Path	LOADFAC	Path for output file containing the load factors on each relation	Chainchoi%1%.ctl	ChainChoice	character
Path	COST	Path for output file containing detailed cost log for selection of relations	Chainchoi%1%.ctl	ChainChoice	character
Path	REP	path for output file with some aggregate statistics.	Chainchoi%1%.ctl	ChainChoice	character

Type	Var. / param.	Description	Controlled where	Used by	Type [range]
Path	LOG	path for logfile that contain error description that haveChainchoi%1%.ctl occurred during program execution. If execution has gone well, this file is empty.	Chainchoi%1%.ctl	ChainChoice	character