TRAFIKVERKET

MPS Samgods 1.2.2

Trafikverket
Trafikverket, Box 388, 831 25 Östersund. Besöksadress: Kyrkgatan 43 B.
Telefon: 0771-921 921, Texttelefon: 010-123 99 97

Dokumenttitel: MPS Documentation
Författare: Jonas Andersson, Mathias Ljung
Dokumentdatum: 2024-02-28

Kontaktperson: Petter Hill, PLep

Content

MPS.jar version 1.2.2

This document describes the java program MPS version 1.2.2 used in Samgods version 1.2.2. The main purpose is to provide information about the program function, control, IO and structure. Finally, a short list of changes and a reference list is provided.

Program function

The main purpose of the program is to generate a standard input file in MPS format for the Linear Programming software, as described in Edwards (2015) chapter 4. MPS uses the output from standard LogMod, as input data.

The program also has some secondary purposes. One is to extract data from the Linear Programming output and generate marginal cost LOS files as input to LogMod. Another is to generate LPX files by merging previous LogMod output with alternative solutions according to the Linear Programming output.

The program is a console application. It is controlled by command line parameters, and a control file containing various input parameters.

Program control

Parameters given in the command line call control the program, both from information in the call itself and from information in a control file.

Command line

Example:

```
java -Xmx8g -Xms4048m -jar MPS.jar JCMW mps.ctl ITR=1
```

MPS is a java application so the first part is the command ”java” which calls up the java virtual machine. If necessary this part can contain a search path, e.g. “C:\Program Files\Java\jre7\bin\java.exe”.

Here, the following command line parameters are available:

Xmx/Xms	are optional parameters, and sets memory allocation and heap size (in the example above 8 GB and 4 GB respectively). It is recommended to use a minimum of 4 GB.
–jar MPS.jar	is the part tells the java virtual machine to execute the MPS program.
JCMW	is code(s) determining which program steps to execute.
MPS.ctl	is the name of the file containing input parameters for the MPS program.
ITR=1	tells the program the number of the iteration running.

Program step control

The following program steps can be required using the command line letters:

J	Resets the iteration log, and generates the JLIST file (a file with superindices mapping all LP-model variables to standard LogMod product number and keys in result files)
C	Generates the COLGEN and LPRAIL files (setup of input data for columns in the LP-model).
M	Merges the COLGEN and LPRAIL files. (combines the results from different iterations into a common file per product group).
W	Generates the MPS file.
L	Writes marginal cost LOS files using the linear programming output. Also adds a marginal cost column to the CAPLINKS file.
X	Writes LPX files.

Control file

The control file, with extension *ctl*, provide further information. The control file parameters are explained in Table 1. Required parameters are marked with black text. Optional parameters are marked with blue text.

The control file may contain comments on separate lines. Comments are added by putting a dash at the beginning of the line, for example:

- directories

Table 1 Control file parameters, including a description and example values. The notation [i], [v], [k] and [f] refers to replaced fields in file names and is marked with a*.

Parameters (with example values)	**Description**
CAPCOST=998801	The capacity cost value that is written to the MPS file. The unit is SEK.
CAPLINKS=RCM\RailLinkCapacitiesBidirectional_STD.dat	List of links with Emme/Voyager numbering system and bidirectional capacity per day.
CC_NAMEROOT=CCS	The base for the ChainChoi output files, usually "ChainChoi" as in "ChainChoi01STD.out". If omitted the default value is "ChainChoi".
*CGMERGE=RCM\ColumnMerge\ColGen2LP[v].dat**	Output. Intermediate file for the process.
CHECKLINK=51,82,84,85,234,274,272	This generates a file named ChecklinkYY_ItrX.dat containing all flows for the specified link (YY = link number, X = iteration number). Either a single link, or a comma separated list of links. One file is written for each link in the list.

CHECKKEY=2 21220	Causes a check of overflow calculations to be written to a file CheckKey_2_21220.log where 2 is commodity and 21220 is key in commodity
*COLGEN=RCM\ColumnData\ColGen2LP[v]-[k].dat**	Output. Intermediate file for the process
CSTVARI=0.5	If value is higher than 0.00001 the stochastic model is run.
DFR=250	Day factor rail (number of working days per year)
EMPTYFRAC=Extract\emptyfrac.dat	Path to emptyfraction file
ERRLOG=RCM/mps_error.log	If the alternative ERRLOG is used the log is not rewritten.
ERRORLOG=RCM/mps_error.log	Errors are logged to this file. If the ERRORLOG parameter is omitted the file is called Error.log and placed in the folder specified by the PATH parameter. If the keyword ERRORLOG is used the log is rewritten at the beginning of the J step.
EXTRA_CAPCOST=1000000000	The cost of acquiring extra capacity. The unit is SEK.
*IO_LOG=RCM\IOLog_LP[i].log**	The line IO_LOG=RCM\IOLog_LP[i].log in the ctl file will cause all names of input and output files to be written to file, together with ctl settings.
ITR_LOG=LP_iter.log	Output - A log file where all steps in the iteration process are logged. It is rewritten when running the J step.
JLIST=RCM\JLISTA.dat	Output. A list of all first solutions from ChainChoi in files ChainChoiNN_01.out having at least one rail leg. The list is generated in the first step of the MPS program. The file also has a column for marking chains with low flow and locked chains. Low flows are marked as 0, locked flows as 2. Other flows are marked 1.
LINKLIST=RCM\Links_List.txt	Input- list of links in the network with ID link followed by Voyager from and to nodes and Emme from and to nodes respectively. Blank or comma separated.
LOCKS=STD *LOCKS=RCM\LockedSTDLogMod_Soln.txt*	LOCKS parameter can take two types of values, either a filename, or the value STD. If the value is STD locked chains are read from the STDLogMod output files LockedXX.log in folder ChainChoi\OUTPUT. The parameter is mandatory if locked flows are present.

*LOWFLOWS=RCM\LowFlows_LP[i].dat**	File where flows below SHIPCUT are aggregated.
*LPMERGE=RailCapMgmt\ColumnMerge\LP_Rail[v].dat**	Output. Intermediate file for the process.
*LPRAIL=RCM\ColumnData\LP_Rail[v]-[k].dat**	Output. Intermediate file for the process.
LPSOFT=CLP	Dual values for the marginal cost LOS files are read from an output file produced by CLP.
*LPSOLUTION=RCM\LP_Rail_LP[i].out**	Output file from CLP. Input for generating marginal cost matrices.
MAX_CAPLINKS=1000	Number of rows from CAPLINKS file. Default 1000. Used by the class SpanningTree.
MAX_CMD=16	Sets the highest commodity number. Default 16.
MAX_EMMESPAN=300000	Maximum size of the span of Emme zones. Default 300 000 (700000 – 999999). Used by the class SpanningTree.
MAX_INTLINKNO=150000	Maximum internal link number in LINKLIST file. Default 150 000. Used by the class SpanningTree.
MAX_JKEYS=500000	Maximum number of Superindexes in the JLIST file per commodity. Default 500 000. Used in the methods Write_CG_LP_Itr0 and Write_CG_LP_ItrX. If either of these methods fail with an ArrayIndexOutOfBounds error increasing this parameter may fix the problem.
MAX_POSSOLS=150000	Maximum number of positive solutions in the Linear Programming output. Used when writing LPX files in iterations 1 or higher. Default 1 500 000. If the program fails with a message of ArrayIndexOutOfBounds in method WriteLPX increasing this parameter may fix the problem.
MAX_SKEYS	Maximum number of super keys to be processed when writing the MPS file. If omitted the maximum super key number in JLIST is used which in most cases is the best option.
MAX_SPANNODE=2000	Span of nodes in Spanning Tree. Default 2000. Used by the class SpanningTree.
MAX_VOYNODE=50000	Maximum Voyager node number in the NODESLIST file. Default 50 000. Used by the class SpanningTree.
*MPS=RCM\LP_Rail_LP[i].MPS**	Output. The MPS file, input for the LP program.

NODESLIST=RCM\Nodes_List.txt	Input- List of nodes in the network with Emme and Voyager numbers. Blank or comma separated.
PATH=C:\workfolder	PATH is the basis for all paths used by mps.jar. It may be relative or absolute. It MUST point at the directory on top of the ChainChoi-directory NB! All other paths in the ctl-file are relative to the folder set by the PATH keyword.
RAILDEFS=RCM\LPRail_defs.dat	Output. Aggregated flows per capacitated link., i e the annual capacity (number of trains per year).
RAILMODES= GHhIiTUDdEF	List of rail modes in ChainChoi output. The example list is the default, the keyword is needed only if new rail modes are used.
SHIPCUT=800	Cutoff for shipment size in kg. For transports with shipment sizes below this level, alternative transport solution are not generated after iteration 0.
SPANNINGTREE=RCM\PathTreeRail.txt	Input - from rail assignment. The file containing spanning tree data. RCM is an acronym for Rail Capacity Management. If the file name extension is .cmp the program expects a spanning tree in compact format.
VEHICLES=201,202,203,204,205,206,207,208,209	List of all train types in the ChainChoi output files. The above list is the default, the keyword is needed only if new train types are added. A current list is VEHICLES=201,202,203,204,205,206,207,208,209,210,211,212. The number of train types in this parameter is also used to determine the number of train types the program handles, and to allocate variables for this number of train types.
UTIL_PERCENT=30	Cut-off value. Utilization ratio for capacitated links at which the link is considered low capacity. The links with utilization ratio below the UTIL_PERCENT value will not be considered in optimization problem since they are redundant. It is a cut-off criteria.

Some parameters in the control file contain [i], [v], [k] and [f]. The following fields are replaced by the appropriate integer during the run, when parsing file names.

[i] is replaced by iteration number.

[k] is replaced by column number. Iteration 0 generates column number 0 and 1, iteration1 generates column 2, and so on. Note! Column 0 is not entered explicitly into the LP-model, but it serves as a reference for computing the impact of entering other columns into the solution.

[v] is replaced by commodity number in steps that loop over all the output files where LogMod produce one output file per commodity.

[f] is replaced by vehicle number.

For example,

MPS=RCM\LP_Rail_LP[i].MPS

means that for iteration 0 the file name will be LP_Rail_LP0.MPS, and for commodity 8 and column 2 the file will be named LP_Rail08-2.dat since

LPRAIL=RCM\ColumnData\LP_Rail[v]-[k].dat.

Program IO

The check file

When a program run is finished without error the file mps_ok.chk is written to the current directory. The file is empty but its presence in the current directory is an indicator of successful execution that can be used by other programs.

Input and output files

This is the data flow in the MPS program. Paths can vary between setups, here they refer to a typical setup for iteration 0 in Table 2.

Table 2 Program input and output flow.

Step	Input	Output
MPS J	ChainChoi\OUTPUT\ChainChoi*NN*STD.out, for all commodities *NN*	RCM\JLISTA.dat
MPS C	ChainChoi*NNmmm*.out- for all commodities *NN* and output files *mmm* (== STD/RCM) SpanningTree files	RCM\ColumnData\ColGen2LP*NN*-1.dat RCM\ColumnData\LP_Rail*NN*-1.dat for all commodities *NN* RCM\LP_Rail_defs.dat Extract\OUTPUT\ALL_Shares_PUT_OD_Vhcl000_0.314

MPS M	RCM\ColumnData\ColGen2LP*NN*-1.dat RCM\ColumnData\LP_Rail*NN*-1.dat for all commodities *NN*	RCM\ColumnMerge\ColGen2LP*NN*.dat RCM\ColumnMerge\LP_Rail*NN*.dat for all commodities *NN*
MPS W	RCM\LP_Rail_defs.dat RCM\ColumnMerge\LP_Rail*NN*.dat	RCM\LP_Rail_LP0.MPS
CLP	RCM\LP_Rail_LP0.MPS	RCM\LP_Rail_LP0.out
MPS L	RCM\LP_Rail_LP0.out INPUT\LOS\v*XXX*_dist.314 for vehicles *XXX* 201 – 209 SpanningTree files	INPUT\LOS\v*XXX*_MC.314 for vehicles *XXX* 201 – 209
MPS X	RCM\JLISTA.dat ChainChoi\OUTPUT\ChainChoi*NN*STD.out RCM\ColumnMerge\ColGen2LP*NN*.dat RCM\LP_Rail_LP0.out	ChainChoi\OUTPUT\ChainChoi*NN*LPX.out for all commodities *NN*

Program structure

The program is originally developed in Eclipse Java IDE. The program was converted to Netbeans IDE for Samgods version 1.2.

Main processes

MPS.java

This class holds the java main procedure. It reads the control file, and branches out according to the given command line parameters.

AddOnCalculator.java

Returns addon factors for empty train wagon flows, as described by Edwards (2015) chapter 4.

CG_LP_write.java

Writes the RAILDEFS and COLGEN files. Also writes aggregated (annual) flows to RAILDEFS file.

JlistaWrite.java

Writes the JLIST file by extracting all chains having one or more rail legs from the ChainChoiNNSTD/RCM.out files.

LP_Merge.java

This class merges the column data files found in the ColumnData folder ColGen2LPNN-X.dat and LP_RailNN-X.dat where X is column number are merged to ColGen2LPNN.dat and LP_RailNN.dat respectively

MPSWriter.java

Writes the MPS file.

SpanningTree.java

Reads and contains data for the spanning tree. Returns the path for a given connection.

WriteLOS.java

This class writes the marginal cost LOS files that are used by LogMod4RCM.

WriteLPX.java

Generates the ChainChoi*NN*LPX.out files by combining ChainChoi*NN*STD/RCM.out with the alternative solution from the CGMERGE file, according to the factor in the LPSOLUTION file.

Utility classes

Funcs.java

A utility class that has various functions for string handling etc.

CtlReader.java

A utility class that reads and holds the values from the control file.

LgFileReader.java

A standard FileReader with the additional function that it logs the name of the file read.

LgFileWriter.java

A standard FileWriter with the additional function that it logs the name of the file written.

Locks.java

Reads the LOCKS file, and keeps track of which transport chains are locked. These chains are omitted from the MPS file. For information about the content of this file please see de Bok et al (2020).

Lists.java

Lists keeps track of various input data.

Changes in MPS 1.2.1 and 1.2.2

- Added support for four-digit capacity numbers, i.e. 9999 capacity conditions.
- Minor update for multiple transfer points within the same chain type

Changes in MPS 1.2

New features:

- Stochastic approach for RCM.
- Added an alternative solver for the linear programming problem that can be used by setting the LPSOFT parameter. Example: LPSOFT=CLP.

Changes in functionality:

- MPS now handles 16 commodities instead of 35. The default value for parameter MAX_CMD has been changed from 35 to 16.

Enhancements:

- Enhancement of the locks functionality that now handles multiple legs in a more secure way.
- MPS now handles null references and closing of streams in an improved faction, preventing memory leakage.
- Improved logging to error-files.

References

Henrik Edwards (2015): Railway Capacity Management for Samgods Using Linear Programming

Michiel de Bok et al (2020): Program documentation for the logistics model for Sweden

TRAFIKVERKET

Trafikverket, Box 388, 831 25 Östersund. Besöksadress: Kyrkgatan 43 B.
Telefon: 0771-921 921, Texttelefon: 010-123 99 97
www.trafikverket.se