TRAFIKVERKET

MPS Documentation

Jon Bergström

Compose IT

Document title: MPS Documentation
Created by: Jon Bergström at Compose IT
Document type: Memo
Case number: 2010/17376

Date of publication: 2015-04-01
Publisher: Trafikverket (The Swedish Transport Administration)
Contact person: Petter Hill
Responsible: Peo Nordlöf
Distributor: Trafikverket (The Swedish Transport Administration)

Content

MPS.jar

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 LogMod, or in later iterations LogMod4RCM, as input data.

The program also has some secondary purposes. One is to extract data form the Linear Programming output and generate marginal cost LOS files as input to LogMod4RCM. Another is to generate LPX files by merging previous LogMod output with alternative solutions according to the Linear Programming output. A third is to make weighted copies of LOS matrices for each STAN group.

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

Program control

Command line

Example:

```
java -Xmx2024m -Xms2024m -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, eg “C:\Program Files\Java\jre7\bin\java.exe”.

Command line parameters:

The Xmx and Xms are optional, and sets memory allocation and heap size.

The part –jar MPS.jar tells the java virtual machine to execute the MPS program.

JCMW are codes to determine which program steps should be executed.

Mps.ctl is the name of the file containing input parameters for the MPS program.

Finally ITR=1 tells the program the number of the iteration running.

Control file

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

Control file parameters:

Example:

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.

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.

CC_NAMEROOT=CCS

Optional. The base for the ChainChoi output files, usually “ChainChoi” as in “ChainChoi01_01.out”. If omitted the default value is “ChainChoi”.

UTIL_PERCENT=50

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.

SPANNINGTREE=RailCapMgmt\PathTreeRail.txt

Input - from rail assignment. The file containing spanning tree data.

NODESLIST=RailCapMgmt\Nodes_List.txt

Input- List of nodes in the network with Emme and Voyager numbers.

NODES=RailCapMgmt\PathTreeNodeUse.txt

Input - List of Emme numbered nodes with their use, where all nodes using rail can be extracted.

LINKLIST=RailCapMgmt\Links_List.txt

Input- list of links in the network with ID link.

CAPLINKS=RailCapMgmt\RailLinkCapacitiesBidirectional_STD.dat

List of links with Emme/Voyager numbering system and bidirectional capacity per day

JLIST=RailCapMgmt\JLISTA.dat [output]

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.

MPS=RailCapMgmt\LP_Rail_LP[i].MPS [output]

Output. The MPS file, input for the LP program.

RAILDEFS=RailCapMgmt\LPRail_defs.dat

Output. Aggregated flows per capacitated link.

COLGEN=RailCapMgmt\ColumnData\ColGen2LP[v]-[k].dat

Output. Intermediate file for the process

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

Output. Intermediate file for the process

LPMERGE=RailCapMgmt\ColumnMerge\LP_Rail[v].dat

Output. Intermediate file for the process

CGMERGE=RailCapMgmt\ColumnMerge\ColGen2LP[v].dat

Output. Intermediate file for the process

LPSOLUTION=RailCapMgmt\LP_Rail_LP[i].out

Output from LPSolve. Input for generation marginal cost matrices.

EXTRACT_OUTPUT=extract\output

The folder containing the output files from Extract, file types OD_TonnesNNN_0.314, OD_VhclNNN_0.314, and OD_EmpNNN_0.314. This parameter is redundant if the following six parameters are specified. These files are used to calculate empty vehicle flows as described by Edwards (2015) sections 1.2 and Appendix B.

EMPTIES_0= extract\output \OD_Emp[f]_0.314

Extract output file

VHCL_0=EXTRACT\OUTPUT\OD_Vhcl[f]_0.314

Extract output file

EMPTIES_X=EXTRACT\OUTPUT\OD_Emp[f]_LPX.314

Extract output file

VHCL_X=EXTRACT\OUTPUT\OD_Vhcl[f]_LPX.314

Extract output file

ADDON_0=EXTRACT\OUTPUT\ODEmpAddOn_0.300

Flows aggregated from Extract output used for calculating AddOn factors

ADDON_X=EXTRACT\OUTPUT\ODEmpAddOn_LPX.300

Flows aggregated from Extract output used for calculating AddOn factors

DFR=250

Day factor rail

CHECKLINK=34

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.

IO_LOG=RCM\IOLog_LP[i].log

Optional. 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.

OPTIONAL PARAMETERS FOR SETTING ARRAY SIZES

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_POSSOLS

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_JKEYS

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.

The following five parameters are used by the class SpanningTree. . If the program fails with a message of ArrayIndexOutOfBounds in this class increasing one ore more of these parameters may fix the problem.

MAX_CAPLINKS

Number of rows from CAPLINKS file. Default 1000. Used by the class SpanningTree.

MAX_INTLINKNO

Maximum internal link number in LINKLIST file. Default 150 000. Used by the class SpanningTree.

MAX_VOYNODE

Maximum Voyager node number in the NODESLIST file. Default 50 000. Used by the class SpanningTree.

MAX_EMMESPAN

Maximum size of the span of Emme zones. Default 300 000. Used by the class SpanningTree.

MAX_SPANNODE

Span of nodes in Spanning Tree. Default 2000. Used by the class SpanningTree.

Parameters with replaced fields in file names:

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.

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

[f] is replaced by vehicle number.

Examples:

MPS=RailCapMgmt\LP_Rail_LP[i].MPS means that for iteration 0 the file name will be LP_Rail_LP0.MPS.

LPRAIL=RailCapMgmt\ColumnData\LP_Rail[v]-[k].dat. For commodity 8 and column2 the file will be named LP_Rail08-2.dat.

Program 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.

MPS flow and files

Paths can vary between setups, here they refer to a typical setup.

Iteration 0

Step	Input	Output
MPS J	ChainChoi\OUTPUT\ChainChoiNN_01.out, for all commodities NN	RailCapMgmnt\JLISTA.dat
MPS C	ChainChoiNN_0X.out- for all commodities NN and outfiles X (1<=X<=5) Extract\OUTPUT\OD_EmpXXX_0.314 Extract\OUTPUT\OD_VhclXXX_0.314 Extract\OUTPUT\OD_TonnesXXX_0.314 to calculate OD bsed AddOn factors RailCapMgmnt\PathTreeNodeUse.txt for a list of rail nodes SpanningTree files	RailCapMgmnt\ColumnData\ColGen2LPNN-1.dat RailCapMgmnt\ColumnData\LP_RailNN-1.dat for all commodities NN RailCapMgmnt\LP_Rail_defs.dat Extract\OUTPUT\ALL_Shares_PUT_OD_Vhcl000_0.314
MPS M	RailCapMgmnt\ColumnData\ColGen2LPNN-1.dat RailCapMgmnt\ColumnData\LP_RailNN-1.dat for all commodities NN	RailCapMgmnt\ColumnMerge\ColGen2LPNN.dat RailCapMgmnt\ColumnMerge\LP_RailNN.dat for all commodities NN
MPS W	RailCapMgmnt\LP_Rail_defs.dat RailCapMgmnt\ColumnMerge\LP_RailNN.dat	RailCapMgmnt\LP_Rail_LP0.MPS
LPSolve	RailCapMgmnt\LP_Rail_LP0.MPS	RailCapMgmnt\LP_Rail_LP0.out
MPS L	RailCapMgmnt\LP_Rail_LP0.out INPUT\LOS\vXXX_dist.314 for vehicles XXX 201 – 209 SpanningTree files	INPUT\LOS\vXXX_MC.314 for vehicles XXX 201 - 209
MPS X	RailCapMgmnt\JLISTA.dat ChainChoi\OUTPUT\ChainChoiNN_01.out RailCapMgmnt\LP_Rail_LP0.out	ChainChoi\OUTPUT\ChainChoiNN_01LPX.out for all commodities NN

Program structure

The program is developed in Eclipse Java IDE.

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

Calculates addo n 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 flows to RAILDEFS file.

JlistaWrite.java

Writes the JLIST file by extracting all chains having one or more rail legs from the ChainChoiNN_01.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 ChainChoiNN_01LPX.out files by combining ChainChoiNN_01.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.

StanLosWriter.java

Copies all LOS files from LOS directory to each of twelve subdirectories, one for each STAN group. If subdirectories don't exist they are created. TIMEH LOS files for vessels are then recalculated using port are factors from file CalibrationParameters.txt in INPUT directory.

Locks.java

Reads the LOCKS file, and keeps track of which transport chains are locked. These chains are omitted from the MPS file.

References

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

TRAFIKVERKET

Trafikverket, Box 388, 831 25 Östersund. Besöksadress: Kyrkgatan 43 B

Telefon: 0771-921 921, Texttelefon: 010-123 99 97

www.trafikverket.se