

Energy Technology Systems Analysis Programme

<http://www.etsap.org/tools.htm>

Documentation for the TIMES Model

PART III

April 2005

Authors:

Richard Loulou

Uwe Remne

Amit Kanudia

Antti Lehtila

Gary Goldstein

General Introduction

This documentation is composed of three Parts.

Part I comprises eight chapters constituting a general description of the TIMES paradigm, with emphasis on the model's general structure and its economic significance. Part I also includes a simplified mathematical formulation of TIMES, a chapter comparing it to the MARKAL model, pointing to similarities and differences, and chapters describing new model options.

Part II comprises 7 chapters that constitute a comprehensive reference manual intended for the technically minded modeler or programmer looking for an in-depth understanding of the complete model details, in particular the relationship between the input data and the model mathematics, or contemplating making changes to the model's equations. Part II includes a full description of the sets, attributes, variables, and equations of the TIMES model.

Part III describes the GAMS control statements required to run the TIMES model. GAMS is a modeling language that translates a TIMES database into the Linear Programming matrix, and then submits this LP to an optimizer and generates the result files. In addition to the GAMS program, two model interfaces (VEDA-FE and VEDA-BE) are used to create, browse, and modify the input data, and to explore and further process the model's results. The two VEDA interfaces are described in detail in their own user's guides.

PART III: GAMS IMPLEMENTATION

TABLE OF CONTENTS FOR PART III

1.	TIMES SOURCE CODE	5
2.	THE TIMES RUN FILE	9
3.	THE EXTENSION OPTION	13
4.	THE TIMES REDUCTION ALGORITHM	15

Introduction

TIMES is written in a modular fashion employing the General Algebraic Modeling System (GAMS)¹. In this sections first the structure of the TIMES code is presented, followed by a description of the information contained in the so-called <Case>.run file, which is a GAMS command script that initiates and controls each model run. Each model run is tagged with the user provided Case name. Each case is composed of any number of Scenario data files (<scenario>.DD/DDS) created from VEDA-FE, the TIMES input data management system. Finally, a short introduction to the extension mechanism, which allows the flexible linking of model extensions (e.g. for lumpy investments) to a model run, is given.

1. TIMES source code²

The structure of the TIMES source code is shown in Figure 1. To run a TIMES model, the user has to provide two files the <case>.run file, which has is passed to GAMS to initiate a model run, and the data dictionary <scenario>.dd file(s), which contains the user input sets and parameters to fully describe the energy system to be analyzed. As a result of a model run a listing file (<case>.lst) and a <case>.gdx file (GAMS dynamic data exchange file) are created. The <scenario>.lst file may contain an echo print of the GAMS source code and the input data, a listing of the concrete model equations and variables, error messages, model statistics, model status and solution dump. The amount of information displayed in the listing file can be adjusted by the user through GAMS options in the <case>.run file. The <case>.gdx file is an internal GAMS file that contains all the model input data and results. It is processed according to the information provided in the TIMES2VEDA.VDD to create results input files for the VEDA-BE software to analyse the model results³. In addition to these two output files, TIMES may create a file called **qa_check.log** to inform the user of possible errors or inconsistencies in the model formulation. The **qa_check.log** file should be examined by the user on a regular basis to make sure no “surprises” have crept into a model.

During a TIMES model run various tasks are performed, which are shortly summarized below.

- **GAMS Compile:** GAMS operates as a two-phase compile then execute system. As such it first reads and assembles all the control, data, and code files into a ready executable; substituting for all GAMS environment switches and subroutine parameter references (the %EnvVar% and %Param references in the source code). If there are inconsistencies in input data they may result in compile-time errors (e.g., \$170 for a domain definition error), causing a run to stop.
- **Initialisation:** Upon completion of the compile step all possible GAMS sets and parameters of the TIMES model generator are declared and initialised, then established for this instance of the model from the user’s data dictionary file(s) (<case>.dd⁴),. The corresponding files are marked yellow in Figure 1. The file

¹ Anthony Brooke, David Kendrick, Alexander Meeraus, and Ramesh Raman, GAMS – A User’s Guide, December 1998.

² The TIMES source code is available from the ETSAP Primary Systems Coordinator. Contact Gary Goldstein, ggoldstein@irglt.com, for more information.

³ The basics of the TIMES2VEDA.VDD control file and the use of the result analysis software VEDA-BE are described separately.

⁴ For simplicity, it has been assumed in this description that the name of the *.run file and the *.dd file are the same (<case_name>). The names of the two files can, however, be

units.def contains the short names for the units that can be used by the modeller. Therefore, one should ensure that the units used in the case study are mentioned in the file **units.def** and eventually expand the unit definitions listed there. The file **maplists.def** contains group declarations for process and commodity groups that might be adjusted by the user with the exceptions listed in **Error! Reference source not found. (com_type, prc_grp)**.

- **Execution:** After having prepared the run, the **maindrv.mod** controls all the remaining tasks of the model run. The basic steps are described here.
 - **Preprocessing:** One major task is the preprocessing of the model input data. During preprocessing control sets defining the valid domain of parameters, equations and variables are generated (e.g., for which periods each process is available, at what timeslice level (after inheritance) is each commodity tracked and does each process operate), input parameters are inter-/extrapolated and time-slice specific input parameters are inherited/aggregated to the correct timeslice level as required by the model generator. The files related to the preprocessing task are shown in light blue in Figure 1.
 - **Coefficient calculation:** A core activity of the model generator is the proper derivation of the actual coefficients used in the model equations. In some cases coefficients correspond directly to input data (e.g., FLO_SHAR to the flow variables), but in other cases they must be transformed. For each, the investment cost (NCAP_ICOST) must be annualized, spread for the economic lifetime and discounted before being applied to the investment variable (VAR_NCAP) in the objective function (EQ_OBJ), and the coefficients in the capacity transfer constraint (EQ_CPT) that must take into account the spread and timing of the capacity build-up and its technical lifetime, or fraction thereof. The corresponding files are shown in light green in Figure 1.
 - **Generation of model equations:** Once all the coefficients are prepared the file **eqmain.mod** controls the generation of the model equations. It calls the individual GAMS routines responsible for the actual generation of the equations of this particular instance of the TIMES model. The generation of the equations is controlled by sets and/or parameters carefully assembled by the pre-processor to ensure that no superfluous equations or matrix intersections are generated.
 - **Setting variable bounds:** The task of applying bounds to the model variables corresponding to user input parameters is handled by the **bndmain.mod** file. In some cases it is not appropriate to apply bounds directly to individual variables, but instead of applying a bound may require the generation of an equation (e.g. the equation **EQ(I)_ACTBND** is created when an annual activity bound is specified for a process having a diurnal timeslice resolution).
 - **Solving the model:** The steps described so far are summarized in GAMS language as compilation of the source code and assembly of the model. After construction of the actual matrix (rows, columns, intersections and bounds) the problem is passed to a solver optimized employing the appropriate technique (LP, MIP, or NLP). The solver returns the solution of

different. The listing file generated by GAMS has always the name of the *.run file. The name of the *.gdx file can be chosen by the user on the command line calling GAMS (e.g. **gams mymodel.run gdx = myresults** will result in a gdx called **myresults.gdx**).

the optimisation back to GAMS. The information regarding the solver status is written by TIMES in a text file called **end_gams.sta**; which allows the user to quickly check whether the optimisation run was successful or not without having to go through the listing file. Information from this file is displayed by VEDA-FE at the completion of the run.

- **Reporting:** Based on the optimal solution the reporting routines (orange files in Figure 1) calculate result parameters, e.g. annual cost information by cost type, year and technology or commodity. These result parameters together with the solution values of the variables and equations (both primal and dual), as well as selected input data, are assembled in the **<case>.gdx** file. The gdx file is then processed by the GAMS **gdx2veda.exe** utility according to the directives contained in **times2veda.vdd** control file to generate files for the result analysis software **VEDA-BE**⁵.

⁵ The basics of the TIMES2VEDA.VDD control file and the use of the result analysis software **VEDA-BE** are described in a separate document (in preparation, interim document available from amit@kanors.com).

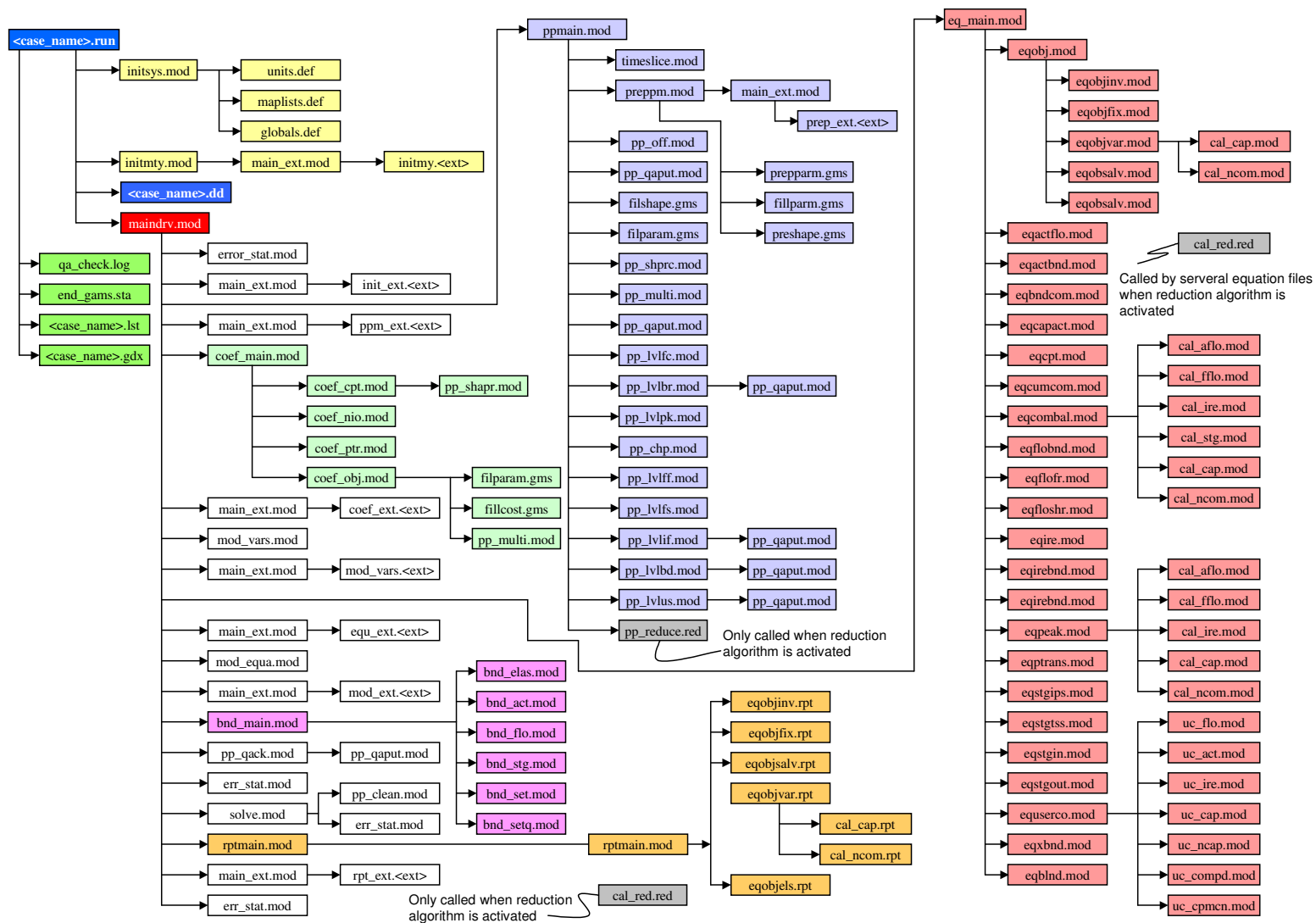


Figure 1.1: File structure of the TIMES code

2. The TIMES RUN file

To start a model run in TIMES, a driving Command File (**vtrun.cmd**) and the **<case>.run** file are assembled by VEDA-FE. The vtrun.cmd script file calls GAMS referring to the **<case>.run** file and identifying the location of the TIMES source code using the following line:

```
Call gams <case>.run idir=<source_code_folder>.gdx =
<case_name>.
```

The contents of the **<case>.run** file are displayed in Figure 2.

```
*$TITLE TIMES -- VERSION 1.3.4
*-----
* The option statements are used to set various global system parameters
* control output detail, solution process and the layout of displays.
*-----
OPTION LIMROW=200000, LIMCOL=20000, SOLPRINT=ON, ITERLIM=50000;
OPTION RESLIM=50000, PROFILE=1, SOLVEOPT=REPLACE;
OPTION LP=CPLEX;
OPTION SYSOUT=ON;
*-----
* Dollar control option
*-----
* turning off echoing put file to listing file
$OFFLISTING
* allowing multiple set definitions
$ONMULTI
*-----
* Definition of several control switches
*-----
* if set to 'YES' similar objective function and handling of capacity
* as in MARKAL
$SET VALIDATE 'NO'

* turn-on/off extended quality assurance checks by setting DEBUG
* to 'YES'/'NO'
$ SET DEBUG 'NO'

* turn-on/off generation of result parameters for VEDA
$ SET SOLVEDA 'YES'

* turn-off generation of import files for old VEDA-BE Version3 by setting
* VEDAADD to 'YES'
$ SET VEDAADD 'YES'

* turn-on/off solver call, set to 'NO' to test for example the compilation
* and matrix generation
$ SET SOLVE_NOW 'YES'

* setting the name of the model
$ SET MODEL_NAME 'TIMES'

* turn-on/off the use of the reduction algorithm
$ SET REDUCE 'YES'

* turn-on/off the use of endogenous technological learning
$ SET ETL 'NO'

* turn-on/off the use of elastic demands
$ SET TIMESD 'NO'

* turn-on/off the use of lumpy investment (discrete investment decisions)
$ SET DSC 'NO'

* solve model from scratch or restart from restart files
*$SET STARTRUN 'RESTART'
$ SET STARTRUN 'SCRATCH'
```

```

*-----
* Definition of timeslices
*-----
* the times-slices MUST come here to ensure ordering is correct in storage
* equations
SET ALL_TS
/
ANNUAL 'Annual'
SP 'Spring'
SU 'Summer'
FA 'Fall'
WI 'Winter'
SPD 'Spring Day'
SPN 'Spring Night'
SUD 'Summer Day'
SUN 'Summer Night'
FAD 'Fall Day'
FAN 'Fall Night'
WID 'Winter Day'
WIN 'Winter Night'
/
*-----
* perform fixed declarations
$BATINCLUDE initsys.mod
*-----
* declare the (system/user) empties
* to use the extension mechanism add extensions, e.g.:
* $BATINCLUDE initmty.mod DSC to use the discrete capacity formulation
$BATINCLUDE initmty.mod
*-----
* accept the actual scenario data
$ BATINCLUDE example.dd
*-----
$ SET RUN_NAME 'BASE'
;
*-----
* Name of scenario and scenario description for VEDA-BE
SET SCENCASE
/
BASE 'Base scenario of the example model'
;/
*-----
* global discount year
q_dyear = 1988.5;
*-----
* calling maindrv mod
$BATINCLUDE maindrv.mod mod

```

Figure 2: Example for a TIMES <case>.run file

At the beginning a <case>.run file some Option control statements, that influence the information output and the solution process, e.g. the choice of solver, are provided. For further available options the user should consult the GAMS manual. Then dollar control options regarding the echoing of the source code (\$ON/OFFLISTING) and the multiple definitions of sets and parameters (\$ONMULTI) are given. Further possible dollar control options are also described in the GAMS manual.

Afterwards the content of several so-called TIMES dollar control (or environment) switches are specified. Within the source code the use of these control switches in combination with queries enables the model to skip or activate specific parts of the code. Thus it is possible to turn-on/off variants of the code, e.g. the use of the reduction algorithm, without changing the input data. The meaning of the different control switches is shown in Table 1.

Table 1: Dollar control (environment) switches in TIMES

ID	Description
BOTIME,	The available time span for allyear can be altered by the two dollar

ID	Description
EOTIME	control parameters BOTIME and EOTIME. Currently EOTIME is set to 1850 and EOTIME to 2200. All years related to the data and model must lie within this range.
DEBUG	Dump out all user/system data structures into a file, and turn on extended quality assurance checks. Activated by means of \$SET DEBUG 'YES'.
DSC	Activating the lumpy (discrete) investments option, by means of \$SET DSC 'YES', results in a MIP problem; requiring a MIP solver (see Section 3 on the lumpy investments extension mechanism)
ETL	Activating the endogenous technology learning feature, by means of \$SET ELT 'YES', results in a MIP problem; requiring a MIP solver.
MID_YEAR	Triggers mid-year discounting (see Part II, Section 5.2), by \$SET MID_YEAR 'YES', otherwise costs are discounted to the beginning of the year.
REDUCE	Activate the reduction algorithm by means of \$SET REDUCE 'YES' (see Section 4 in PART III).
RUN_NAME	The <case> name for this run, used for <case>.VD* files passed to VEDA-BE.
SOLVE_NOW	If the user wishes to only check the input data and compile the source code, but not solve the model \$SET SOLVE_NOW 'NO' can be specified.
SOLVEDA	The solution dump for VEDA-FE is activated by providing \$SET SOLVEDA 'YES'.
TIMESED	To activate the elastic demand formulation \$SET TIMESED 'YES' is provided.
VALIDATE	Usage of MARKAL like formulation of objective function, capacity etc. is turned-on by specifying \$SET VALIDATE 'YES'.
VEDAVDD	Control whether to generate the VEDA-BE files in VEDA3 format, or by means of the GDX2VEDA GAMS utility according to the VEDA2GAMS.VDD file.
XTQA	To activate the extended QA checks \$SET XTAQ 'YES' can be provided. This is highly recommended during the initial phases of developing a model and can be very useful for debugging the model later as well.

The definition of the set of all timeslices (**all_ts**) used in the model has to be done before any other declarations carried out in the initialisation file **initsys.mod**. This is necessary to ensure the correct ordering of the timeslices for seasonal, weekly or daynite storage processes. Therefore, the order of the timeslices given in the **<case>.run** file should accurately represent the sequence in which the storage processes operate, e.g. from winter to spring, from spring to summer, etc.

After the definition of the timeslices the files **initsys.mod** and **inimty.mod**, which are responsible for the declaration and initialisation of all sets and parameters of the model generator, are included.

The line containing the include command for the file **inimty.mod** can be supplemented by calls for additional extensions that trigger the use of additional special equations or report routines. The use of these extension options are described in more detail in the following section.

Afterwards the data dictionary file(s) (**example.dd** in Figure 2) containing the user input sets and parameters, i.e. the description of the analysed energy system, is included. It is

normally advisable to segregate user data into “packets” as scenarios, where there is a single Base scenario containing the core descriptions of the energy system being studied and a series of alternate scenario depicting other aspects of the system. For example, one **<scenario>.dd** file contains the description of the energy system for a reference scenario, and additional **<alt_scenario>.DDS** files containing additions or changes relative to the reference file, for example CO₂ mitigation targets for a reduction scenario. Each of these **DD/DDS** files needs to explicitly brought in by means of a **\$INCLUDE <scenario>.DD/DDS** command in the **<case>.run** file.

The dollar control switch **RUN_NAME** contains the short name of the scenario, and is used for the name of the results files (**<case>.VD***) passed to **VEDA-BE**. The set **SCENCASE** is used to add a scenario description to be displayed in the **VEDA-BE** software when analysing the results. Therefore the set element in **SCENCASE** should be identical to the short name of the scenario in the control switch **RUN_NAME**. The scenario description in the set **SCENCASE** is given in quotes.

The last line of the **<case>.run** file invokes the file main driver routine (**maindrv.mod**) that initiates all the remaining tasks related to the model run (preprocessing, coefficient calculation, setting of bounds, equation generation, solution, reporting). Thus any information provided after the inclusion of the **maindrv.mod** file will not be considered in the model run.

3. The extension option

The extension options allow the user to link in additional equations or reporting routines to the standard TIMES code, e.g. the DSC extension for using lumpy investments. The entire information relevant to the extensions are isolated in separate files from the standard TIMES code. These files are identified by their extensions, e.g. *.dsc for lumpy investments or *.cli for the climate module. The extension mechanism allows the TIMES programmer to add new features to the model generator, and test them, with only minimal hooks provided in the standard TIMES code. It is also possible to have different variants of an equation type, for example of the market share equation, or to choose between different reporting routines, for example adding detailed cost reporting. The extension options currently available in TIMES are summarized in Table 2.

Table 2: Extension options in TIMES

Extension	Description
CLI	The climate module estimates change in CO ₂ concentrations in the atmosphere, the upper ocean including the biosphere and the lower ocean, calculates the change in radiative forcing and the induced change in global mean surface temperature. It is activated by passing the extension CLI to the file initmtty.mod in the <case>.run file resulting in \$BATINCLUDE initmtty.mod CLI
DSC	Option to use lumpy investment formulation; since the usage of the discrete investment options leads to a Mixed-Integer Programming (MIP) problem, the solve statement in the file solve.mod has to be altered. To activate this extension the \$SET DSC 'YES' control switch needs to be provided in the <case>.run file and the extension CLI has to be passed to the file initmtty.mod resulting in \$BATINCLUDE initmtty.mod DSC
IER	Several extensions of the equation system introduced specifically for modelling needs by IER (market/product share constraints, backpressure/condensing mode full load hours). It is activated by passing the extension IER to the file initmtty.mod in the <case>.run file resulting in \$BATINCLUDE initmtty.mod IER
VTT	Several extensions of the equation system introduced specifically for modelling needs by VTT (market/product share constraints, commodity-specific availabilities, generalized flow share equation). It is activated by passing the extension VTT to the file initmtty.mod in the <case>.run file resulting in \$BATINCLUDE initmtty.mod IER

The extension(s) that are to be included in the current model run need to be activated in the **<case>.run** file, and passed to **inimty.mod**, e.g.

```
$BATINCLUDE initmtty.mod DSC CLI
```

As shown above it is possible to add several extension in the \$BATINCLUDE line above at the same time, in this case the lumpy investment option and the climate module are incorporated with the standard TIMES code.

The GAMS source code related to an extension <ext> has to be structured by using the following file structure in order to allow the model generator to recognize the extension⁶ (see also Figure 1). The placeholder <ext> stands for the extension name, e.g. CLI in case of the climate module extension.

- **initmty.<ext>**: contains the declaration of new sets and parameters, which are only used in the context of the extension;
- **init_ext.<ext>**: contains the initialisation and assignment of default values for the new sets and parameters defined in **initmty.<ext>**;
- **coef_ext.<ext>**: contains coefficient calculations used in the equations or reporting routines of the extension; it might contain calls to the inter-/extrapolation routines (**prepparm.mod**, **fillparm.mod**);
- **mod_vars.<ext>**: contains the declaration of new variables;
- **equ_ext.<ext>**: contains new equations of the extension;
- **mod_ext.<ext>**: adds the new defined equations to the model;
- **rpt_ext.<ext>**: contains new reporting routines.

Not of all these files have to be provided when developing a new extension, if for example no new variables or no new report routines are needed, these files can be omitted.

⁶ This structure is only of interest for those modellers who want to programme their own extensions. The modeller who uses an extension in his model does not need to know these programming details.

4. The TIMES Reduction algorithm

The motivation of the reduction algorithm is to reduce the number of equations and variables generated by the TIMES model. Thus hopefully the memory usage should be reduced and the solution time improved. An example for a situation where model size can be reduced is a process with one input and one output flow, where the output flow variable can be replaced by the input variable times the efficiency. Thus the model can be reduced by one variable (output flow variable) and one equation (transformation equation relating input and output flow).

Implemented reduction measures

1. Process without capacity related parameters does not need capacity variables:
 - **No capacity variables VAR_CAP and VAR_NCAP created.**
 - **No EQL_CAPACT equation created.**
2. Primary commodity group consists of only one commodity:
 - **Flow variable VAR_FLO of primary commodity is replaced by activity variable.**
 - **No EQ_ACTFLO equation defining the activity variable created.**
3. Exchange process imports/exports only one commodity:
 - **Import/Export flow VAR_IRE can be replaced by activity variable (might not be true if exchange process has an efficiency).**
 - **No EQ_ACTFLO equation defining the activity variable created.**
4. Process with one input and one output commodity:
 - **One of the two flows has to define the activity variable. The other flow variable can be replaced by the activity variable multiplied/divided by the efficiency.**
 - **No EQ_PTRANS equation created.**
5. An emission flow of a process can be replaced by the sum of the fossil flows multiplied by the corresponding emission factor:
 - **No flow variables for the emissions created**
 - **No EQ_PTRANS equation for the emission factor.**
6. Upper/fixed activity bound ACT_BND of zero on a higher timeslice level than the process timeslice level is replaced by activity bounds on the process timeslice level. Thus no EQG/E_ACTBND equation is created.
7. Process with upper/fixed activity bound of zero cannot be used in current period. Hence, all flow variables of this process are forced to zero and need not be generated in the current period. Also EQ_ACTFLO and Eqx_CAPACT are not being generated. If the output commodities of this process can only be produced by this process, also the processes consuming these commodities are forced to be idle, when no other input fuel alternative exists.
8. When a FLO_FUNC parameter between two commodities is defined and one of these two commodities defines the activity of the process, the other flow variable can be replaced by the activity variable being multiplied/divided by the FLO_FUNC parameter.
 - **One flow variable is replaced.**
 - **No EQ_PTRANS equation for the FLO_FUNC parameter is created.**

Implementation

To make use of the reduction algorithm one has to define the environment variable \$SET REDUCE 'YES'/'NO' in order to turn on/off the reduction. This environment variable controls in each equation, where the flow variable occurs, whether it should be replaced by some other term or not.

The possibility of reduction measures is checked in the file pp_reduce.red. If reduction is turned on, flow variables that can be replaced are substituted by a term defined in cal_red.red. The substitution expression for the import/export variable VAR_IRE is directly given in the corresponding equations. In addition the \$control statement controlling the generation of the equations EQ_PTRANS, EQ_ACTFLO, Eqx_CAPACT has been altered. Also bnd_act.mod has been changed to implement point 6 above.

To recover the solution values of the substituted variables corresponding parameters (PAR_FLO for VAR_FLO.L, DPAR_FLO for VAR_FLO.M, PAR_IRE for VAR_IRE.L and DPAR_IRE for VAR_IRE.M) are calculated in the IER report extension and are then written to the veda file.

Results

In the following the contents of the listing files displaying the solution and solver statistics for model runs of the German model with and without reduction algorithm are given.

Solution statistics for model run **without reduction**:

```
MODEL STATISTICS

BLOCKS OF EQUATIONS      80      SINGLE EQUATIONS      426255
BLOCKS OF VARIABLES     17      SINGLE VARIABLES      418191
NON ZERO ELEMENTS     1694576

GENERATION TIME          =          138.609 SECONDS   276.2 Mb
WIN207-133

EXECUTION TIME          =          624.484 SECONDS   276.2 Mb
WIN207-133
Solution Report          SOLVE TIMES Using LP From line 523210

---- 1 EXEC-INIT          0.000          0.000 SECS
172.3 Mb

              S O L V E          S U M M A R Y

MODEL   TIMES          OBJECTIVE  OBJz
TYPE   LP              DIRECTION  MINIMIZE
SOLVER CPLEX          FROM LINE  523210

**** SOLVER STATUS      1 NORMAL COMPLETION
**** MODEL STATUS      1 OPTIMAL
**** OBJECTIVE VALUE    16041290.3075

RESOURCE USAGE, LIMIT    9503.546          50000.000
ITERATION COUNT, LIMIT  65              60000
```

Due to problems with current GAMS installation
solution time not correct, see solver statistics below

Solver statistics for model run **without reduction**:

```

Starting Cplex...
Presolve has eliminated 240201 rows and 239990 columns...
Presolve has eliminated 241727 rows and 240594 columns...
Aggregator has done 108495 substitutions...
Tried aggregator 1 time.
LP Presolve eliminated 241748 rows and 240644 columns.
Aggregator did 108495 substitutions.
Reduced LP has 76012 rows, 69052 columns, and 560638
nonzeros.
Presolve time = 14.91 sec.
Number of nonzeros in lower triangle of A*A' = 1367851
Elapsed ordering time = 20.09 sec.
Elapsed ordering time = 53.23 sec.
Elapsed ordering time = 68.52 sec.
Using Nested Dissection ordering
Total time for automatic ordering = 75.34 sec.
Summary statistics for Cholesky factor:
  Rows in Factor          = 76012
  Integer space required  = 484428
  Total non-zeros in factor = 6559889
  Total FP ops to factor  = 2212538823
  Itn      Primal Obj      Dual Obj  Prim Inf Upper Inf
Dual Inf
  0  2.4994054e+009 -1.6718665e+007 1.86e+009 1.07e+007
1.23e+008
  1  1.8371633e+009  9.1021633e+009 1.15e+009 6.66e+006
9.41e+007
  ...
  ...
  65 1.6041329e+007  1.6040001e+007 1.19e-002 6.72e-012
3.11e-006
  Barrier time = 655.16 sec.

Primal crossover.
Primal: Fixing 57698 variables.
  57697  PMoves:      Infeasibility  1.17554189e-001
Objective 1.60413240e+007
  ...
  0  PMoves:      Infeasibility  6.70898953e-003
Objective 1.60412937e+007
Primal: Pushed 10575, exchanged 47123.
Dual: Fixing 31911 variables.
  31910  DMoves:      Infeasibility  1.66352741e+005
Objective 1.60411838e+007
  ...
  0  DMoves:      Infeasibility  4.59722403e+002
Objective 1.60412935e+007
Dual: Pushed 22512, exchanged 9399.
Using devex.

Iteration log . . .
Iteration: 1 Scaled infeas = 0.000000
Iteration: 2 Objective = 16041293.694405
Removing shift (14).

```

```

Iteration: 472 Scaled infeas = 4.540451
Elapsed time = 1347.41 sec. (1000 iterations)
Iteration: 1089 Scaled infeas = 0.003932
Iteration: 1538 Objective = 16041316.720697
Elapsed time = 1370.53 sec. (2000 iterations)
Iteration: 2049 Objective = 16041293.901022
Iteration: 2511 Objective = 16041290.307534
Total crossover time = 745.50 sec.

```

Optimal solution found Total solver time 1409 sec without reduction.

Simplex iterations after crossover: 2513

Objective : 16041290.307527

Solution statistics for model run with reduction:

MODEL STATISTICS

```

BLOCKS OF EQUATIONS      80      SINGLE EQUATIONS      146586
BLOCKS OF VARIABLES     16      SINGLE VARIABLES      153737
NON ZERO ELEMENTS     1070319

```

GENERATION TIME = 140.531 SECONDS 202.9 Mb
WIN207-133

EXECUTION TIME = 634.391 SECONDS 202.9 Mb
WIN207-133

Solution Report SOLVE TIMES Using LP From line 526152

---- 1 EXEC-INIT 0.000 0.000 SECS
157.7 Mb

S O L V E S U M M A R Y

```

MODEL   TIMES      OBJECTIVE  OBJz
TYPE    LP          DIRECTION  MINIMIZE
SOLVER  CPLEX       FROM LINE  526152

```

```

**** SOLVER STATUS      1 NORMAL COMPLETION
**** MODEL STATUS      1 OPTIMAL
**** OBJECTIVE VALUE    16039728.1279

```

```

RESOURCE USAGE, LIMIT  2263.843  50000.000
ITERATION COUNT, LIMIT 63         60000

```

Due to problems with current GAMS installation solution time not correct, see solver statistics below

Solver statistics for model run with reduction:

```

Starting Cplex...
Presolve has eliminated 28920 rows and 43692 columns...
Aggregator has done 40926 substitutions...
Tried aggregator 1 time.
LP Presolve eliminated 29772 rows and 43955 columns.

```

Aggregator did 40926 substitutions.
Reduced LP has 75888 rows, 68856 columns, and 560003 nonzeros.

Presolve time = 9.03 sec.
Number of nonzeros in lower triangle of A*A' = 1367308
Elapsed ordering time = 23.33 sec.
Elapsed ordering time = 50.55 sec.
Elapsed ordering time = 65.59 sec.
Using Nested Dissection ordering
Total time for automatic ordering = 71.78 sec.
Summary statistics for Cholesky factor:
Rows in Factor = 75888
Integer space required = 484998
Total non-zeros in factor = 6608576
Total FP ops to factor = 2232053628
Itn Primal Obj Dual Obj Prim Inf Upper Inf
Dual Inf
0 2.4967530e+009 1.0856905e+004 1.87e+009 1.07e+007
8.79e+006
...
63 1.6039729e+007 1.6039725e+007 7.87e-005 4.77e-012
1.64e-006
Barrier time = 631.39 sec.

Primal crossover.
Primal: Fixing 4871 variables.
4870 PMoves: Infeasibility 1.04331194e+001
Objective 1.60397294e+007
...
0 PMoves: Infeasibility 1.27294579e+000
Objective 1.60397305e+007
Primal: Pushed 2841, exchanged 2028.
Dual: Fixing 32435 variables.
32434 DMoves: Infeasibility 2.73758628e+004
Objective 1.60397231e+007
...
0 DMoves: Infeasibility 2.29155343e+004
Objective 1.60397290e+007
Dual: Pushed 21429, exchanged 11006.
Using devex.

Iteration log . . .
Iteration: 1 Objective = 16039730.505061
Removing shift (8).
Total crossover time = 99.19 sec.

Optimal solution found

Total solver time 730 sec without reduction.
--

Simplex iterations after crossover: 255

Objective : 16039728.127920

The number of equations and variables in the reduction is circa 63 % lower than in the non-reduced case. Since the smaller number of equations and variables require less memory, the memory usage in the reduction run decreases by 26 %.

The solution time is in the reduction case by the factor 1.9 lower than the one in the non-reduced model run. The solution time of the barrier algorithm (barrier time in the listings above) is very similar in both runs. The mayor time savings are however gained in the subsequent simplex iterations which are performed in GAMS to convert the interior solution of the barrier algorithm into a basic solution.

For this model a difference in the objective function values of 0.012 % can be observed. With smaller models identical values for the objective function have been obtained. At the moment it is not clear what is causing this deviation.

Pending/Problems

- In some cases (observed with the `canx3.dd`) the reduced problem will produce an “optimal solution with unscaled infeasibilities”.
- Shadow price of non-generated EQ_PTRANS equations are lost.
- Reduced cost of upper/fixed ACT_BND of zero are lost. If one needs this information, one should use a very small number instead, e.g. 1.e-5, as value for the activity bound.