CIFE CENTER FOR INTEGRATED FACILITY ENGINEERING

OrgSim: A Tool to Simulate Organizational Planning, Coordination, and Information Processing

By

Borge Obel
Richard M. Burton

CIFE Working Paper #053
August, 1998

STANFORD UNIVERSITY

Copyright © 1998 by
Center for Integrated Facility Engineering

If you would like to contact the authors please write to:

c/o CIFE, Civil and Environmental Engineering Dept.,
Stanford University,
Terman Engineering Center
Mail Code: 4020
Stanford, CA 94305-4020

OrgSim

Ver. 1.0

A tool to simulate organizational planning, coordination, and information processing

June 1998

Børge Obel
Department of Management
Odense University, Denmark
Visiting Professor, December 1997 - July 1998
Center for Integrated Facility Engineering (CIFE)
Stanford University, CA, USA

and

Richard M. Burton
The Fuqua School of Business,
Duke University, NC, USA

Abstract

This report is a technical manual for the OrgSim simulation program. OrgSim is a GAMS® program that can simulate planning, coordination, and information processing in an organization. The basis is a mathematical programming presentation of organizational activities. The theoretical background is described in R. B. Burton and B. Obel, *Designing Efficient Organizations: Modeling and Experimentation,* North Holland, 1984 and in Mathematical Contingency Modeling for Organizational Design: Taking Stock, in R. M. Burton and B. Obel, *Design Models for Hierarchical Organizations: Computation, Information and Decentralization,* Kluwer Academic Publishers, Boston, 1995, pp. 3-34. The manual describes the mathematical models using GAMS notation. Additional all files are listed and commented upon. Further, a complete data set for the sample case - HOC (Burton and Obel, 1984, 1995) is given.

Table of Contents

1. Introduction

OrgSim is a simulation tool, which enables simulation of planning, coordination, and information processing in an organization. The basis is a mathematical programming presentation of organizational activities. The theoretical background is described in R. B. Burton and B. Obel, *Designing Efficient Organizations: Modeling and Experimentation,* North Holland, 1984 and in Mathematical Contingency Modeling for Organizational Design: Taking Stock, in R. M. Burton and B. Obel, *Design Models for Hierarchical Organizations: Computation, Information and Decentralization,* Kluwer Academic Publishers, Boston, 1995, pp. 3-34.

Organizational parameters that can be manipulated in OrgSim include: configuration, size, environment, coordination system, information processing, incentives and actor behavior.

This manual describes the theoretical mathematical programming model and the algorithm for information processing and information exchange. The system is based on the decomposition principle of linear programming and includes both the Dantzig-Wolfe algorithm, the ten Kate algorithm as well as a mixture of the two - the Obel-procedure.

The OrgSim system is programmed using GAMS. For specifics about GAMS notation please refer to the GAMS documentation (http://www.gams.com). Academic organization may obtain the educational version of GAMS from the Gams Development Corporation free of charge.

A sample case - the HOC case from Burton and Obel (1984) is used to show how to run OrgSim. Data for this case is also provided with the disk. The specifics of the various files that are needed to run OrgSim are given in the appendices. The files can be downloaded from http://www.busieco.ou.dk/~boe/orgsim. ORGSIM is activated by running GAMS on the MIX2.GMS file (gams mix2.gms) or the Orgsim.bat file.

Before running GAMS on the file MIX2.GMS, data and parameters must be specified in the following files: HOCDATA.INC, FORMAT.GMS, Change-file, Append-file, and Specification-file. The output from the simulation is stored in regular GAMS output files. The file names are specified in mix2.gams. In the Append-file, the percentage of optimality

measure that are discussed in Burton and Obel(1994) is given for each trial. You may use any GAMS option to specify different kinds of output.

The next section presents the overall base model in GAMS format. Then the data for the HOC case is described. The section on the HOC case shows with as little technical discussion as possible how the parameters for a simulation study have to be chosen. The actual files are shown in appendices 5-8.

2. The Base Model in Gams Format

MIX2.GMS solves problems of the following type:

$$\max Z = \sum_{d \in CURDIV} \sum_{\substack{j \in DIVVAR \\ j \le NDIVVAR(d)}} CDIV_{d,j} * XDIV_{d,j}$$

$$+ \sum_{j \in CURHQVAR} CHQ_j * XHQ_j$$

$$- \sum_{j \in CURRES} POVERUSEMP_r * OVERUSEMP_r$$

$$- \sum_{r \in EQURES} PUNDUSEMP_r * UNDUSEMP_r$$

$$- \sum_{r \in CURRES} \sum_{d \in TKUNITS} POVERU,SETK_{d,r} * OVERUSETK_{d,r}$$

$$- \sum_{r \in EQRES} \sum_{d \in TKUNITS} PUNDUSETK_{d,r} * UNDUSETK_{d,r}$$

Subject to the following constraints:

1) for all r ∈ EQRES:

$$\sum_{j} \in CURHQVAR\ AHQ_{r,j} * XHQ_j + \sum_{d \in CURDIV} \sum_{\substack{j \in DIVVAR \\ j \le NDIVVAR(d)}} ADIV_{d,r,j} * XDIV_{d,j}$$

$$-OVERUSEMP_r + UNDUSEMP_r$$

$$- \sum_{d \in TKUNITS} OVERUSETK_{d,r} + \sum_{d \in TKUNITS} UNDUSETK_{d,r} = Amount_r$$

- *For all* $r \in CURRES/EQRES$: $\sum_{j \in CURHQVAR} AHQ_{r,j} * XHQ_j$

$$+ \sum_{d \in CURDIV} \sum_{\substack{j \in DIVVAR \\ j \leq NDIVVAR(d)}} ADIV_{d,r,j} * XDIV_{d,j} - OVERUSEMP_r - \sum_{d \in TKUNITS} OVERUSETK_{d,r}$$

$$\leq Amount_r$$

3) *For all* $d \in DWUNITS$:

$$\sum_{\substack{j \in DIVVAR \\ j \leq NDIVVAR(d)}} CDIV_{d,j} * XDIV_{d,j} \leq BPROFITDW(d)$$

4) *For all* $d \in TKUNITS$:

$$\sum_{\substack{j \in DIVVAR \\ \leq NDIVVAR(d)}} CDIV_{d,j} * XDIV_{d,j} \leq BOUNDTK(c$$

5) 7) -8), 11)-12), 17)-20) in appendix 4.

This problem is equivalent to the model OPTIMAL in appendix 4 if POVERUSETK, PUNDUSETK, POVERUSEMP, PUNDUSEMP, BPROFITOW and BOUNDTK are chosen sufficiently high. The program attempts to solve the above problem by using the Obel algorithm, Obel(1978) (with an added headquarters part that is not decomposed).

When using the Obel algorithm on this problem, the following is observed[1]:

1) The subproblems corresponding to budget controlled divisions are all feasible and bounded
2) The master problem is feasible.
3) The subproblems corresponding to price-controlled divisions are all feasible.

Two problems remain:

1) The master problem might be unbounded.
2) The subproblems corresponding to price-controlled divisions might be unbounded.

1) is coped with by adding a "true upper bound" to the problem called BOUNDMP. This bound can only be active if optimum has not yet been reached and does therefore not effect the convergence of the algorithm. 2) is coped with by adding an upper bound to the optimal objective values in the price controlled divisions called BOBJDW. If BOBJDW is chosen sufficiently high, this will not effect the convergence of the algorithm either.

3. MIX2.GMS used on the HOC-case

In the following the use of the program MIX2.GMS on the HOC-case on page 27 in the book "Designing Efficient Organizations" by Burton and Obel, North Holland, 1984 is discussed. The following models the "two divisions" case. The notation is defined in the appendices.

The first thing to note is that the HOC-case (like many other organization models) involves very large numbers (mill. dollars/mill. units). This creates a big problem for almost

[1]It is assumed that the problem is feasible and bounded and that POVERUSEMP, PUNDUSEMP, POVERUSETK, PUNDUSETK, BOUNDTK and BPROFITOW are chosen as discussed before.

all LP-solvers and it is therefore necessary to "pre-scale" the data. Profit will therefore be measured in "millions of dollars" and the activity levels will be measured in "thousands of units." The profit contribution from the division variables is then:

$$CDIV = \begin{bmatrix} -3.5 & -0.027 & -1.5 & -3 & -2.3 & -3 & -2 \\ 5.04 & 9.6 & 7 & 23 & 17.5 & -0.027 & 0 \end{bmatrix}$$

where $CDIV_{d,j}$ contains the profit contribution (in millions of dollars) pr. unit of division variable j in division d. The variables in the divisions are indexed in the following way:

Y_{CF} ~ variable 1 in division 1
X_{OW} ~ variable 2 in division 1
X_{CF} ~ variable 3 in division 1
X_{TF} ~ variable 4 in division 1
X_{CB} ~ variable 5 in division 1
Y_{CB} ~ variable 6 in division 1
X_{TB} ~ variable 7 in division 1

and in division 2:

X_{CS} ~ variable 1 in division 2
X_{CD} ~ variable 2 in division 2
X_{COT} ~ variable 3 in division 2
X_{TL} ~ variable 4 in division 2
X_{TS} ~ variable 5 in division 2
X_{OT} ~ variable 6 in division 2

The following parameter define the structure of the problem:

NDIV = 2

NDIVVAR ('1') = 7

NDIVVAR ('2') = 6

NHQVAR = 0

NHQCON = 0

NRES = 4 (constraints 1) -4) are the resource-constraints).

EQRESCON (RES) = 1 (all resource constraints are binding).

$$ADIV^1 = \begin{bmatrix} -1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix}$$

(resource usage by division the variables of division 1).

$$ADIV^2 = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$

(resource usage by division the variables of division 2).

AMOUNT (RES) = 0. (The usage of all resources must balance).

Now all data concerning the "technical" division constraints will be initialized. A few points must be made:

1. Since the activity levels are measured in thousands of units, all "right-hand-sides" in the division constraints must be divided by 1000.
2. Constraints 5), 9), 15), 17), 18) 19) and 20) only involve one variable. These constraints can be handled by putting upper and lower bounds on the variables.
3. Constraint 11) is multiplied by 600 on both sides.
4. Constraint 12) is multiplied by 60 on both sides.
5. Constraint 16) is divided by 1000 on both sides.

With 2) in mind, the following is clear:

NDIVCON ('1') = 3
NDIVCON ('2') = 5.

The constraints will be indexed as follows:

(1,1) ~ constraint6) in the Burton and Obel(1984) book.
(1,2) ~ constraint7) in the Burton and Obel(1984) book.
(1,3) ~ constraint8) in the Burton and Obel(1984) book.
(2,1) ~ constraint13) in the Burton and Obel(1984) book.
(2,2) ~ constraint10) in the Burton and Obel(1984) book.
(2,3) ~ constraint11) in the Burton and Obel(1984) book.
(2,4) ~ constraint12) in the Burton and Obel(1984) book.
(2,5) ~ constraint16) in the Burton and Obel(1984) book.

With 1) - 5) in mind, the "right-hand-sides" in the "technical" division constraints will be set to the following:

$$RHSDIV = \begin{bmatrix} 230 & 760 & -110 & 0 & 0 \\ 30 & 500 & 60 & 6 & 85 \end{bmatrix}$$

Also, with 2) - 5) in mind, the "technical" coefficients in the division constraints can be set to the following:

$$BDIV^1 = \begin{bmatrix} 0 & -1 & 5 & 7 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 16 & 0 & 12 \\ 0 & 0 & 1 & 1 & 2 & 0 & 2 \end{bmatrix}$$

$$BDIV^2 = \begin{bmatrix} 0 & 0 & 0 & 6 & 6 & -1 & 0 \\ 8 & 12 & 12 & 0 & 0 & 0 & 0 \\ 1 & 1.5 & 1.5 & 0 & 0 & 0 & 0 \\ 1.08 & 1.92 & 1.4 & 4.6 & 1.5 & 0 & 0 \end{bmatrix}$$

Finally, since no division constraint is binding: EQDIVCON (DIV, DIVCON) = 0

Now all bounds on the division variables will be initialized. First the lower bounds will be set. According to the book, all variables except (1,1) and (1,6) (Y_{CF} and Y_{CB}) should have a lower bound of 0. Constraints 17) and 18) however, gives lower bounds of respectively -5 (thousands) and -2) (thousands) on these 2 variables. This gives:

LODIVVAR (DIV, DIVVAR) = 0 (this gives all variables a lower bound of 0).
LODIVVAR ('1', '1') = -5 (resets the lower bound on variable (1,1) to -5).
LODIVVAR ('1', '6') = -2. (resets the lower bound on variable (1,6) to -2).

Next the upper bounds on the variables will be set. A few points should be noted before doing this:

1. Constraint 9) gives an upper bound of 23 on variable (1,2).
2. Constraint 5) gives an upper bound of 10 on variable (2,2).
3. Constraint 20) gives an upper bound of 3 on variable (2,3).
4. Constraint 14) gives an upper bound of 3 on variable (2,4).
5. Constraint 15) gives an upper bound of 2 on variable (2,5).
6. Constraint 19 gives an upper bound of 3 on variable (2,6).
7. Even though the other variables are not explicitly bounded from above, they should be. The physical environment in which the company operates often dictates natural bounds on the activities. Here a default upper bound of 1000 is chosen.

This gives:

UPDIVVAR (DIV, DIVVAR) = 1000. (sets default upper bounds).
UPDIVVAR ('1', '2') = 23 (upper bound on variable (1,2)).

UPDIVVAR ('2', '2') = 10. (upper bound on variable (2,2)).
UPDIVVAR ('2', '3') = 3 (upper bound on variable (2,3)).
UPDIVVAR ('2', '4') = 3 (upper bound on variable (2,4)).
UPDIVVAR ('2', '5') = 2 (upper bound on variable (2,5)).
UPDIVVAR ('2', '6') = 3 (upper bound on variable (2,6)).

Suppose that both divisions are budget controlled in the first planning period. This means:

DW(DIV) = 0. (this is not explicitly entered in the data file since this is the default value).

Since both divisions are budget controlled, prices for overusing/underusing in a given division must be supplied.

The overuse/underuse variables in the budget-controlled divisions competes with the ordinary division variables for the resources in these divisions budgets. The negative of prices for underusing/overusing the resources in a given division should therefore be lower than the lowest profit contribution from any division variable in that division. The lowest profit contribution in division 1 is -3.5 for variable (1,1). The lowest profit contribution in division 2 is - 0.027 for variable (2,6). The prices for overusing/underusing in the division are therefore chosen to:

POVERUSETK ('1', RES) = 50.
PUNDUSETK ('1', RES) = 50
POVERUSETK ('1', RES) = 10
PUNDUSETK ('2', RES) = 10.

In the master problem (the headquarters problem) the overuse/underuse variables compete with the profit variables for the resources. These profit variables enter the objective function in the master problem with a coefficient of 1. The prices for overusing/underusing the resources in the master problem are therefore set to 10:

POVERUSEMP (RES) = 10.
PUNDUSEMP (RES) = 10

Since both divisions are budget-controlled the master problem moves from being unbounded to being optimal. An upper bound on the optimal objective values in the master problem is therefore needed. Since the optimal objective value to the problem is 178.693, the bound is chosen to 200:

BOUNDMP = 200

The program also needs bounds on the optimal objective values in the budget controlled divisions. The optimal profit contributions from the 2 divisions are:

For division 1: - 171.331 (mill. dollars)
For division 2: 350.024 (mill dollars)
(This gives a total of 178.693)
The bound in division 1 is therefore chosen to -100 and the bound in division 2 is chosen to 400:

BOUNDTK ('1') = -100
BOUNDTK ('2') = 400

Suppose also one wants to supply initial prices and budgets:

PRICE = 1
BUDGET = 1

According to the above, INITP and INITBUD should now be initialized. Here I have chosen the initial prices and budgets to zero however, which are also the default values for these prameters. The initialization of these parameters is therefore omitted.

If data is to be written to an append-file (in a comma separated way) and a change file (to be discussed later) is included then:

APPEND = 1
CHANGE = 1

Finally, suppose that one wants the program to run for 11 iterations in the first planning period (the number of iterations needed to reach optimality) :

NITER = 11

All data in the file HOCDATA.INC has now been entered. Notice that all data have been entered through assignments. Other forms of data entry are also possible (see the GAMS-manual). You should remember that data entry through TABLE-statements and PARAMETER-statements both declares and initializes the given PARAMETER. If you want to enter data through either of these statements you must therefore erase the declaration and initialization of the PARAMETER in the file FORMAT.GMS. Notice also that all statements are terminated with a semicolon (;).

In the following the use the program through more than one planning period is discussed. One then needs to make a file containing the changes from one planning period to the next (since otherwise it will just solve the same problem again and again). The file must follow the following syntax:

```
IF (ORD(PLANPER) EQ 1,
Assignments implementing the changes in data from planning period 1 to planning period
2.);

IF(ORD(PLANPER) EQ 2,
   );
```

.

.

.

.

.

.

IF (ORD(PLANPER)EQ n,

Assignments implementing the changes in data from planning period n to planning period n+1.);

where it's assumed that there are n+1 planning periods in the test problem. Putting assignments *outside* the IF-statement is also allowed. These assignments will be executed after *every* planning period.

Let us implement a change file for the HOC case. The program will be run for 4 planning periods:

In planning period 1 we will run the program as specified before.

In planning period 2, we will run the program as before, except that division 1 will now be price controlled and that the program will run for 14 iterations. The program will need a bound on the profit contribution from division 1 and a bound on the optimal objective value in division 1. As before, the bound on the profit contribution from division 1 is chosen to -100, while the bound n the optimal objective value in division 1 is chosen to 10000. Furthermore, the prices for overusing/underusing the resources in the master problem need to be revised, since the overuse/underuse variables in the master problem now compete with the profit contributions from division 1. These can be as large as -100, and the prices for overusing/underusing in the master problem is therefore chosen to 150.

In planning period 3 we will run the program as before, except that now division 1 is again budget controlled while division 2 is now price controlled. The bound on the profit contribution from division 2 is again chosen to 400, while the bound on the optimal objective value in division 2 is again chosen to 10000. The overuse/underuse variables in the master problem are now competing with the profit contributions from division 2 (as large as 400). The prices for overusing/underusing in the master problem are therefore chosen

to 10. The program will run for 5 iterations.

In planning period 4, the program will be run with both divisions price controlled for 11 iterations. The overuse/underuse variables in the master problem will again compete with the profit contributions from division 1 and the prices for overusing/underusing in the master problem are therefore reset to 150.

The "change file" corresponding to the above description is as follows::

```
IF (ORD(PLANPER) EQ 1,
NITER = 14
DW('1') = 1
DW ('2') =  0
BOBJDW ('1') = 10000
BPROFITDW ('1') = -100
POVERUSEMP (RES) = 150
PUNDUSEMP (RES) = 150
);

IF (ORD(PLANPER) EQ 2,
NITER = 5
DW('1') = 0
DW('2') = 1
BOBJDW ('2') = 10000
BPROFITDW ('2') = 400
POVERUSEMP(RES) = 10
PUNDUSEMP(RES) = 10
);
IF (ORD(PLANPER) EQ 3,
NITER = 11
DW('1') = 1
DW('2') = 1
```

```
POVERUSEMP(RES) = 150
PUNDUSEMP(RES) = 150
);
```

NITER is chosen to secure optimality in the given planning period. There is no way of knowing whether the subproblem corresponding to a price controlled division is feasible with an upper bound on the optimal objective value of 10000. It is known, however, that if a subproblem corresponding to a price controlled division is infeasible, it is because 10000 is too low a number (or of course, if the overall problem is infeasible). In fact, trying to set BOBJDW to 1000, the subproblem corresponding to division 1 turned out to be infeasible. BOBJDW should be chosen as low as possible without any subproblem turning infeasible.

Various possibilities for entering data in the "change-file" exist:

1. Random data:

Suppose that management knows that division 1 "cheats" with the results reported to the headquarter in iteration 1 of planning period 2, but that the "cheating parameter" could be anywhere in the range of 0.5 to 0.9. The statements implementing this could be:

```
IF   (ORD(PLANPER) EQ1,
     BETA ('1', '1') = UNIFORM (0.5, 0.9=
     );
```

2. Restrictions on the budget setting for headquarter:

Suppose that management has to restrict itself from planning period 2 and forwards. It has to set the budgets within a 5% internal of the budgets from the last planning period. In order to implement this, the user has to communicate with the internal budget matrix BUDGET. The statements implementing this is:

```
UPBUDVAR (TKUNITS, RES) $ (BUDGET(TKUNITS, RES) GT 0) =
1.05*BUDGET(TKUNITS, RES)
```

```
LOBUDVAR (TKUNITS, RES) $ (BUDGET (TKUNITS, RES) GT 0) =
0.95*BUDGET(TKUNITS, RES)
UPBUDVAR (TKUNITS, RES) $ (BUDGET(TKUNITS, RES)LT0) =
0.95*BUDGET(TKUNITS,RES)
LOBUDVAR(TKUNITS,RES) $ (BUDGET(TKUNITS,RS) LT0) =
1.05* BUDGET(tKUNITS,RES)
```

A few things should be noted:

1) It should be checked whether the current budget is positive or negative in order to give meaningful upper and lower bounds to the variable BUDVAR. This is what the $-operator does before the equality sign. An assignment is only made if the test inside the parentheses is true.
2) If the budget to a given division of a given resource turned out to be zero, no assignment is made.
3) The statements above are not embedded in an IF-statement, since they must be executed in every planning period.

3. Entering initial prices and initial budgets:

Suppose an initial price of 1 for all resources has to be sent to all price controlled divisions in planning period 2. Suppose also that an initial budget of 2 for all resources has to be sent to all budget controlled divisions in planning period 2. The statements implementing this is the following:

```
IF (ORD(PLANPER)EQ 1
    PRICEI = 1
    BUDGETI = 1
    INITP (RES) = 1
    INITBUD(TKUNITS, RES) = 2
    );
```

4. Eliminating the headquarter:

This can be done by entering the following:

```
NHQVAR = 0
NHQCON = 0
```

The statements above can, of course, also be embedded in an IF-statement.

5. Eliminating the last indexed division:

Suppose that in the 1. planning period there were 5 divisions. The following statement will eliminate division 5 from planning period 2 and forwards:

```
NDIV = 4
```

This statement can, of course, also be embedded in an IF-statement.

In a similar way, one can also eliminate the last indexed resource, the last indexed variable in a given division, the last indexed constraint in a given division, and so on. Likewise, one can add a constraint, a variable, a resource and so on.

6. Changing the status of a constraint:

Suppose that in planning period 1, resource 1 had to balance, but that in planning period 2 and forwards, the use of resource 1 doesn't need to balance. The following will implement this:

```
EQRESCON('1') = 0
```

Above, how to implement various situations through the "change-file" is discussed . As a basic rule *all* data elements can be changed in this file. However, one does have to be careful, especially when deleting the variables and constraints. Remember that it is only possible to delete the *last* indexed constraints and variables.

7. Description of the program

In the following the program is described step by step.

Refer to the listing of the program in Appendix 8.

The first thing the program does is to create the following index sets:

DIV = {1,2 ..... (1. number in MIX2EXE.gms)}

RES ={1,2, ...(2. number)}

DIVVAR = {1,2, ... (2. number)}

DIVCON = {1,2, ... (4. number)}

HQVAR = {1,2, ... (5. number)}

HQCON = {1,2, ... (6. number)}

ITER = {1,2, ... (7. number)}

PLANDER = {8. number}

Remember that *all* internal sets will be drawn as subsets of these sets or of the Cartesian product of these sets, so they have to be large enough to cover *all* relevant indexes. They cannot be initialized empty so they have to contain at least one element.

The next thing that happens is that *all* the user controlled scalars and parameters will be declared and initialized (see appendix 1).

Next - the program MIX2.GMS is called with the 5 given arguments.

What MIX2.GMS does is most naturally explained by going through the "segments" of the program. Two segments in the program are separated by the following line: /* ** .....*** */

Segments

1. Option turning the automatic reprinting of the program in the LST-file off.
2. Other options to minimize the size of the LST-file. They can be set by replacing "OFF" with "ON".
3. The segment does 4 things:

   a) Declares the file APPFILE.

b) Tells the compiler to appends to this file.

c) Declares the file RESULTS

d) Reads the data for the 1. planning period.

4. Writes the name of the test problem

5. Write the headline to the file RESULTS

6. Declares the internal sets CURDIV, CURRES, CURVAR, CURHQVAR, CURHQCON, CURITER, DONEITER and PRINTSET (see appendix 2).

7. Declares the internal sets EQRES, EQDIV and EQHQ (see appendix 2).

8-9) Declares the internal sets LOOPDIV, LOOPITER, TKUNITS AND DWUNITS (see appendix 2).

10. Declares the internal parameters W,L, COEFFBUD, RHSL, PI, PRICE and BUDGET (see appendix 2).

11. Declares all variables used by the program (see appendix 3).

12-13) Declares and defines the relevant constraints in the program (see appendix 4).

14) Declares the models to be used by the program (see appendix 4).

15) Declares and initializes *all* the internal scalars used by the program (see appendix 2).

16) Declares the internal parameters LASTPRICE and LASTBUD (see appendix 2).

17) Initializes PRICE and BUDGET.

18) Starts looping over the planning periods. For p = 1,2, ..., |PLANPER| in turn.

19) Initializes the internal sets CURDIV, CURRES, CURHQVAR, CURHQCON, CURITER, DONEITER and PRINTSET for the current planning period (see appendix 2).

20) Initializes PRICE and BUDGET for the current planning period if the user has chosen to do so (as chosen by the user through the scalars PRICE1, BUDGET1 and SHARERES - See appendix 1).

21) Updates LASTBUD and LASTPRICE.

22) Initializes TKUNITS and DWUNITS for the current planning period (see appendix 2).

23) Initializes EQRES, EQDIV AND EQRES for the current planning period (see

appendix 2).

24) Initializes W, L. RHSL, COEFFBUD and P1 to 0.

25-26) Gives initial primal and dual solutions to the solver (see appendix 3 and 4) before solving the model OPTIMAL. Also sets the upper and lower bounds on the variables.

27) Solves the model OPTIMAL - see appendix 4.

28) Statement directing output towards the file RESULTS.

29) Aborts the program in the cases where OPTIMAL was infeasible or unbounded and writes this information to the file RESULTS.

30) Writes the optimal objective value to the problem OPTIMAL and (if the user has chosen to do so through the user controlled data DIVPRINT and HQPRINT) the optimal solution values of the variables.

31) Sets END to O (not END).

32) Starts looping over the iterations.
For i =1 ... NITER. Stops if END $\neq$ 0.

33) Initializes SUMOBJTK (see appendix 2).

34) Starts looping over the divisions.
For d =1 .... NDIV.

35) Initializes CURDIV and CURVAR (see appendix 2).

36) Checks whether the "looping division" is price controlled.

37-38) Gives a primal and dual initial solution before solving the model DWDIV (see appendix 3).

39) Solves DWDIV (see appendix 4).

40) Prints the optimal objective value of the model DWDIV and (if so chosen by the user through the user controlled parameter DIVPRINT) the optimal values of the variables in the file RESULTS.

41-42) Calculates W and L (see appendix 2).

43) If the "looping division" is budget controlled.

44-45) Gives a primal and dual solution to the solver before solving the problem TKDIV (see appendix 3).

46) Solves TKDIV (see appendix 4).

47) Prints the optimal objective value to the problem TKDIV and (if so chosen by the user through the user controlled parameter DIVPRINT) the optimal values of the variables in the file RESULTS.

48) Calculates SUMOFINF and prints it in the file RESULTS (see appendix 2).

49) Updates SUMOBJTK (see appendix 2).

50) Calculates L (see appendix 2).

51) Calculates P1 (see appendix 2).

52) Calculates RHSL (see appendix 2).

53) Calculates COEFFBUD (see appendix 2).

54) End IF , end LOOP.

55) Updates DONEITER (see appendix 2).

56-57) Gives a primal and dual solution to the solver before solving the model MASTER (see appendix 3) and sets upper and lower bounds on the BUDVAR variables (see appendix 3 and 4).

58) Solves MASTER (see appendix 4).

59) Writes the optimal objective value of the problem MASTER to the file RESULTS.

60) Updates PRICE and BUDGET (see appendix 2).

61) Calculates OBJCONCOMB and TOTPENMUSE (see appendix 2).

62) Prints OBJCONCOMB and TOTPENMUSE (if there are any DW-division) in the file RESULTS.

63) Prints SUMOBJTK in the file RESULTS.

64) Calculates DEGOFOPT (see appendix 2).

65) Prints DEGOFOPT in the file RESULTS.

66) Calculates SUMOFINE and prints it in the file RESULTS (see appendix 2).

67) Prints the optimal values of the headquarter variables (if NHQVAR > 0 *and* HQPRINT $\neq$ 0) IN THE FILE results.

68) Prints the prices and budgets *inside* the loop over the iterations, if so chosen through the user controlled data INITERPRPR and INITERBPR by the user in the file RESULTS (see appendix 1).

69) Calculates TOL (see appendix 2).

70) Updates LASTBUD and LASTPRICE (see appendix 2).

71) Checks whether TOL < SOLTOL. If so END is set to 1, and the loop over the iterations stops.

72) End LOOP.

73) Prints the prices and (if there are any budget controlled divisions) the budgets found in the current planning period in the file RESULTS.

74) Reads the "changefile" if CHANGE ≠ 0.

75) Appends "DEGOFOPT," to the appendfile" if APPEND ≠ 0.

76) End LOOP.

77) Stops writing to the file RESULTS.

Appendix 1: Description of the various user controlled data elements used by the program MIX2.GMS

Below is a description of all the data elements controlled by the user. They can be changed in the data file and/or in the file containing the changes from one planning period to the next. If not changed, they will remain at the default values.

Scalars (numbers):

1) NDIV ~ Number of divisions. Default 0

2) NRES ~ Number of resources. Default 0

3) NHQVAR ~ Number of headquarter variables. Default 0.

4) NHQCON ~ Number of headquarter constraints. Default 0

5) NITER ~ Number of iterations. Default 0

6) BOUNDMP ~ Bound on the optimal objective values in the master problem. Default 10^7

7) PRICE1 ~ "Flag" telling the program whether initial prices are used. A nonzero value implies the use of initial prices, and a value of 0 implies no use. Default 0.

8) BUDGET1 ~ As above, but for budgets. Default 0

9) SHARERES ~ "Flag" telling the program whether the divisions are to share the resources initially or not. A nonzero value implies the sharing of resources and a value of 0 implies no sharing. Default 0.

10) SOLTOL ~ Tolerance for stopping the loop over the iterations (calculated as the absolut change in prices and budgets from one iteration to the next - see appendix 2). Default 10^{-6}

11) CHANGE ~ "Flag" telling the program whether a file implementing the changes from one planning period to the next should be included in the program. A nonzero value implies the inclusion of such a file, a

value of 0 implies no inclusion. Default 0.

12) APPEND ~ "Flag" telling the program whether the degree of optimality at the end of each planning period should be printed in an "append file" (in a comma separated way). A nonzero value implies the printing of these values, and a value of 0 implies the no printing. Default 0.

13) HQPRINT ~ "Flag" telling the program whether the optimal solution values of the headquarter variables should be printed in the file containing the results. A nonzero value implies the printing of these values and a value of 0 implies no printing. Default 0.

14) INITERPRPR ~ "Flag" teling the program whether the optimal price vector (after the solution of the master problem should be printed *inside* the loop over the iterations. A nonzero value implies the printing of these values, and a value of 0 implies no printing. The prices and budgets found at the end of a given planning period will always be printed. Default 0.

Parameters (vectors and matrixes):

1. AHQ ~ Resource usage by the headquarter activities. Delared over RES x HQVAR. $AHQ_{r,j}$, $r \in$ RES *and* $j \in$ HQVAR must contain the amount of resource r that one unit of headquarter variable j uses. Default 0.

2) BHQ ~ Coefficients of the headquarter variables in the "technical" headquarter constraints. Declared over HQCON x HQVAR. $BHQ_{c,j}$, $C \in$ HQCON *and* $J \in$ HQVAR must contain the coefficient for variable j in constraint c. Default 0.

3) CH ~ profit contributions from the headquarter variables. Declared over HQVAR. CHQ_j, $j \in$ HQVAR must contain the profit contribution pr. Unit of variable j. Default 0.

4) ADIV ~ Resource usage by the division activities. Declared over DIV x RES x DIVVAR. ADIV $_{d,r,j}$, d $\in$ DIV, r $\in$ RES *and* j $\in$ DIVVAR must contain the amount of resource r that one unit of variable j in division d uses. Default 0.

5) BDIV ~ Coefficients for division variables in the "technical" division constraints. Declared over DIV x DIVCON x DIVVAR $BDIV_{d,c,j}$, d $\in$ DIV, c $\in$ DIVCON *and* j $\in$ DIVVAR must contain the coefficient for variable j in constraint c in the division d. Default 0.

6) CDIV ~ Profit contributions from division variables. Declared over DIV x DIVVAR. $CDIV_{d,j}$, d $\in$ DIV *and* j $\in$ DIVVAR must contain the profit contribution pr. unit of division variable j in division d. Default 0.

7) EQHQCON ~ Vector declared over HQCON. $EQHQCON_c$, c $\in$ HQCON is a "flag" telling the program whether headquarter constraint c must be binding. $EQHQCON_c$ must be nonzero if constraint c must be binding and 0 otherwise (a "less than or equal" constraint). Default 0.

8) EQRESCON ~ As above, but for the resource constraints declared over RES. Default 0.

9) EQDIVCON ~ As in 8), but for the division constraints. Declared over DIV x DIVCON. Default 0.

10) POVERUSETK ~ Prices (or penalties) for over using the resources in the budget controlled divisions. Declared over DIV x RES. Poverusetk $_{d,r}$, d $\in$DIV *and* r $\in$ RES must contain the price for overusing the budget of resource r in division d by one unit. Default 10^5.

11) PUNDUSETK ~ As above, but for underusing. Declared voer DIV x RES. Default 10^5.

12) POVERUSEMP ~ As in 10), but for the headquarter (appears in the master problem). Declared over RES. Default 10^5

13) PUNDUSEMP ~ As above, but for underusing. Declared over RES. Default 10^5

14) RHSHQ ~ Vector declared over HQCON containing the "right-hand-sides" in the headquarter constraints. $RHSHQ_c$, $C \in$ HQCON must contain the "right-hand-side" in constraint c. Default 0.

15) RHSDIV ~ As above, but for the division constraints. Declared over DIV x DIVCON. Default 0.

16) AMOUNT ~ As in 14), but for the resource constraints. Declared over RES. Default 0. It can be viewed as the amounts available of the resources.

17) NDIVVAR ~ Vector declared over DIV, where $NDIVVAR_d$, $d \in$ DIV, must contain the number of division variables in division d. Default 0.

18) NDIVCON ~ As above, but for the division constraints. Declare over DIV. Default 0.

19) DW ~ Vector of "flags" declared over DIV. Dw_d, $d \in$ DIV must be nonzero if division d is supposed to be price controlled and 0 if division d is supposed to be budget controlled. Default 0.

20) INITP ~ Vector declared over RES, where $INITP_r$, $r \in$ RES must contain the initial price on resource r, if PRICEI is nonzero. Default 0.

21) INITBUD ~ As above, but for the initial budgets (if BUDGETI is nonzero). Declared voer DIV x RES. Default 0.

22) BETA ~ 1-dimensional matrix containing the "cheating parameters". Declared over DIV x ITER. $BETA_{d,i}$, $d \in$ DIV *and* $i \in$ ITER must contain the cheating parameter in division d in iteration i. Default 1 (no cheating).

23) UPHQVAR ~ Vector declared over HQVAR, where $UPHQVAR_j$, $j \in$ HQVAR must contain the upper bound on headquarter variable j. Default INF (no bound).

24) LOHQVAR ~ As above, but for lower bounds. Declared over HQVAR. Default -INF (no bound).

25) UPDIVVAR ~ As in 232), but for the division variables. Declared over DIV x DIVVAR. Default INF (no bound).

26) LODIVVAR ~ As above, but for lower bounds. Declared over DIV x DIVVAR. Default -INF (no bound).

27) DIVPRINT ~ Vector of "flags" declared over DIV. $DIVPRINT_d$, d $\in$ d DIV should be nonzero if the optimal solution values of the variables in division d should be printed in the file containing the results and 0 otherwise. Default 0.

28) INITERBPR ~ Vector of "flags" declared over DIV. $INITERBPR_d$, d $\in$ DIV should be nonzero if the budget for division d should be printed in the file containing the results *inside* the loop over the iterations and 0 otherwise. Default 0.

29) BOUNDTK ~ Vector declared voer DIV. $BOUNDTK_d$ where d is a budget controlled division, should contain a "true upper bound" on the optimal profit contribution from division d. Default 10^6.

30) BPROFITOW ~ Vector declared over DIV, where $BPROFITOW_d$ (d a price controlled division should contain a "true upper bound" on the optimal profit contribution from division d. Default 10^6.

31) BOBJDW ~ Vector declared over DIV, where BOBJDW (d a price controlled division) should contain an upper bound on the optimal objective values in division d. Default 10.

32) UPBUDVAR ~ 2-dimensionel matrix declared over DIV x RES. $UPBUDVAR_{d,r}$, d $\in$ DIV *and* r $\in$ RES must contain the upper bound on the budget of resource r that can be sent to division d (a budget controlled division). Default INF (no bound).

33) LOBUDVAR ~ As above, but for lower bounds. Declared over DIV x RES. Default - INF (no bound)

Comments:

1. If PRICEI = 0, the 0-vector will be used in the 1. planning period and the price vector found in the previous planning period will be used in the others.

2. If BUDGETI = SHARERES = 0, the 0-matrix will be used in the 1. Planning period and the budget matrix found in the previous planning period will be used in the others.

3. The Prices and budgets found in a given planning period will always be printed at the end of that planning period.

4. Even if CHANGE = 0 an empty file should be present in MIX2EXE.GMS and on the operating system.

5. POVERUSETK and PUNDUSETK should be chosen on the basis of the profit contributions from the ordinary division variables (the CDIV parameter). They should be so large that it will be non profitable to overuse/underuse the resources, yet not so large that it will create numerical problems for the solver. A similar remark holds for POVERUSEMP and PUNDUSEMP. The overuse/underuse variable in the master rpoblem competes with:

   The profit contributions from the headquarter variables (the parameter cHQ).

   The profit variables representing profit contribution from the budget controlled divisions (enters the objective function with a coefficient of 1 - see appendix 3 and 4).

6. The profit contributions from vertices generated in the price controlled divisions.

   POVERUSEMP and PUNDUSEMP should be larger than any of the above numbers.

7. There is no way of securing that subproblems corresponding to price controlled divisions are feasible with the numbers chosen in BOBJDW and BPROFITDW. Even if BPROFITDW is chosen as a "true upper bound". It is, however, known (if the overall problem is feasible and BPROFITDW is chosen as "true upper bounds") that a number

for BOBJDW exits so that the subproblems corresponding to price controlled divisions are feasible. My advice is, therefore, to experiment with BOBJDW and choose BPROFITDW as "moderate true upper bounds".

Appendix 2: Description of the various internal sets, parameters and scalars used by the program MIX2.GMS

Internal sets

First comes a description of 10 sets sharing the following properties:

1. They are all initialized at the beginning of each planning period.
2. They are never changed before the next planning period.

The sets are:

1	CURRES $\subseteq$ RES	~	Subset of RES containing the relevant resources for the current planning period. R ε CURRES $\Leftrightarrow$ r ε RES *and* r $\leq$ NRES.
2	CURHQVAR $\subseteq$ HQVAR	~	Subset of HQVAR containing the relevant headquarter variables for the current planning period. J ε CURHQVAR$\Leftrightarrow$ j ε HQVAR *and* j $\leq$ NHQVAR.
3	CURHQCON HQCON	~	Subset of HQCON containing the relevant headquarter constraints for the current planning period. C ε CURHQCON $\Leftrightarrow$ C ε HQCON *and* C $\leq$ NHQCON.
4	CURITER $\subseteq$ ITER	~	Subset of ITER containing the relevant iterations for the current planning period. I ε CURITER $\Leftrightarrow$ I ε ITER *and* i $\leq$ NITER.
5	PRITSET $\subseteq$ div	~	Subset of DIV containing the relevant divisions whose solution should be printed in the current planning period. d ε PRINTSET $\Leftrightarrow$ d ε DIV *and* d $\leq$ DIVPRINT_d $\neq$0.
6	EQRES $\subseteq$ RES	~	Subset of RES containing the resources that must

balance in the current planning period. r ε EQRES ⇔ r ε CURRES *and* $EQRESCON_r \neq 0$.

7 EQDIV⊆ DIV x DIVCON ~ Subset of DIV x DIVCON containing the "technical" division constraints that must balance in the rurrent planning period. (D,c) ε EQDIV ⇔ d ε DIV *and* c ε DIVCON *and* $c \leq NONCON_{d,c} \neq 0$.

8 EQHA ⊆ HQCON ~ Subset of HQCON containing the "technical" headquarter constraints that must balance in the current planning period. C ⊆ EQHQ ⇔ c ε CURHQCON *and* $EQHQCON_c \neq 0$.

9 TKUNITS ⊆ DIV ~ Subset of DIV containing the budget controlled divisions in the current planning period. d ε DIV *and* $d \leq NDIV$ *and* $Dw_d = 0$.

10 DWUNITS ⊆ DIV ~ Subset of DIV containing the price controlled divisions in the current planning period. d ε DWUNITS ⇔ d ε DIV *and* $d \leq NDIV$ *and* $Dw_d \neq 0$.

The next 3 internal sets are changed during a given planning period:

11 DONEITER ⊆ ITER ~ Subset of ITER containing the indexes of the finished iterations at a given moment of execution. Initialized empty at the beginning of each planning period and updated by the current iteration after the solution of all subproblems.

12 CURVAR ⊆ DIVVAR ~ Subset of DIVVAR containing the relevant division variables at a given moment of execution. Before solving the subproblem corresponding to division d, CURVAR is set to the following: j ε CURVAR ⇔ J ε

DIVVAR *and* j ≤ $NDIVVAR_d$.

13 CURDIV ⊆ DIV ~ Subset of DIV containing the relevant division(s) at a given moment of execution. Before solving the model OPTIMAL, CURDIV is set to: d ε NDIV ⇔ d ε DIV *and* d ≤ NDIV. Before solving the subproblem corresponding to division d, CURDIV is set to: CURDIV = {d}.

The last 2 sets are aliases (copies) of their parent set:

14 LOOPDIV ~ Copy of DIV.

15 LOOPITER ~ Copy of ITER.

Internal parameters (vectors and matrixes):

1 W ~ Declared over ITER x DIV. $W_{i,d}$, i ε CURITER *and* d ε DWUNITS will contain the profit contribution from division d sent to the headquarter in iteration i in a given planning period.

2 L ~ Declared over ITER x DIV x RES. $L_{i,d,r}$, i ε CURITER, d ε DIV, d ≤ NDIV and r ε CURRES will contain the resource demand for resource r sent to the headquarter from division d in iteration i.

3 PI ~ Declared over DIV x RES. $Pi_{d,r}$ will contain (after solving a TK-subproblem) the optimal solution value of the dual variable corresponding to "budget constraint" r in TK-division d. It can be viewed as the divisions conception of the price on resource r.

4 COEFFBUD ~ Declared over ITER x DIV x RES. $COEFFBUD_{i,d,r}$, i ε CURITER, d ε TKUNITS *and* r εCURRES will contain the coefficient for the budget variable corresponding to division d and resource r generated in iteration i.

5 RHSL ~ Declared over ITER x DIV. $RHSL_{i,d}$, i ε CURITER *and* d ε TKUNITS will contain the "right-hand-side" in the constraint generated in division d in iteration i.

6 PRICE ~ Declared over RES. Internal price vector.

7 BUDGET ~ Declared over DIV x RES. Internal budget matrix.

8 LASTPRICE ~ Declared over RES. It will contain either the initial price vector in a given planning period or the price vector generated in the last iteration.

9 LASTBUD ~ Declared over DIV x RES. It will contain either the initial budget matrix in a given planning period or the budget matrix generated in the last iteration.

In the following a variable followed by a ".L" will refer to the optimal value of that variable from the last solution involving the variable. Likewise ".M" used on a constraint will mean the optimal value of the dual variable corresponding to that constraint from the last solution involving that constraint (that's GAMS-syntax).

Calculation of W:

Suppose DWDIV has just been solved for division d in iteration i:

$$V_{i,d} = BETA_{d,i} * \sum_{j \varepsilon CURVAR} CDIV_{d,j} * XDIV_{}$$

Calculation of L:

Suppose the subproblem corresponding to division d in iteration i has just been solved:

1. If d ε DWUNITS:

   For all r ε CURRES , do:

$$L_{i,d,r} = BETA_{d,i} * \sum_{j \varepsilon CURVAR} ADIV_{d,r,j} * XDIV.L_{d,j}$$

2 If dε TKUNITS:

   For all r ε EQRES, do:

$$L_{i,d,r} = BETA_{d,i} * (\sum_{j \varepsilon CURVAR} ADIV_{d,r,j} * XDIV.L_{d,j} - OVERUSEtk.l_{d,r} + UNDUSETK.L_{d,r})$$

   For all r ε CURRES\EQRES, DO:

$$L_{i,d,r} = BETA_{d,i} * (\sum_{j \varepsilon CURVAR} ADIV_{d,r,j} * XDIV.L_{d,j} - OVERUSETK_{d,r})$$

Calculation of PI:

Suppose TKDIV has just been solved for division d in iteration i:

   For all r ε EQRES, do:

   $PI_{d,r}$ = BUDEQXON.$M_{d,r}$

   For all r ε CURRES\EQRES, do:

   $PI_{d,r}$ = BUDLECON.$M_{d,r}$

Calculation of COEFFBUD:

Suppose TKDIV has just been solved for division d in iteration i:

For all r ε CURRES, do:

$COEFFBUD_{i,d,r} = PI_{d,r}$

Calculation of RHSL:

Suppose TKDIV has just been solved for division d in iteration i:

$$RHSL_{i,d} = -ZTK.L * BETA_{d,i} + \sum_{r \varepsilon CURRES} L_{i,d,r} * PI_{d,r}$$

Calculation of PRICE:

Suppose the master problem has just been solved:

For all r ε EQRES, do:

$PRICE_r$ = RESEQCON. M_r

For all r ε CURRES\EQRES, do:

PRICEr = RESLECON.M_r

Calculation of BUDGET:

Suppose the master problem has just been solved:

For all d ε TKUNITS, do:

For all r ε CURRES, do:

$BUDGET_{d,r}$ = BUDVAR.$L_{d,r}$

Scalars (numbers):

1 TOL ~ Scalar containing the measure used for stopping the loop over the iterations.

2 END ~ Flag used to stop the loop over the iterations (END is 0 a the start of a given planning period and is set to 1 if

TOL < SOLTOL).

3 SUMOBJTK ~ Scalar that will contain (after solving all subproblems) the sumof the objective values from the TK-divisions.

4 OBJCONCOMB ~ Scalar that will contain (after solving the master problem) the objective value of the convex combination of vertices generated in the DW-divisions.

5 TOTPENMUSE ~ Scalar that will contain (after solving the master problem) the total penalty payed for "misusing" the resources.

6 SUMOFINF ~ Scalar that will contain (after solving either the master problem or a TK-subproblem) the sum of infeasibility in the resource constraints.

Calculation of TOL:

The calculation of TOL is based on the parameters PRICE, BUDGET, LASTPRICE and LASTBUD. Procedure for calculating TOL:

1 Set TOL = 0

2 If there are any DW-divisions, do:

$$Set\ TOL = \sum_{r\ \varepsilon\ CURRES} |PRICE_r - LASTPRICE_r|$$

3 If there are any TK-division, do:

$$SetTOL = TOL + \sum_{d\ \varepsilon\ TKUNITS} \sum_{r\ \varepsilon\ CURRES} |BUDGET_{d,r} - LASTBUD_{d,r}|$$

Calculation of OBJCONCOMB:

$$OBJCONCOMB = \sum_{i \varepsilon DONEITER} \sum_{d \varepsilon DWUNITS} LAMDA.L_{i,d} * W_{i,d}$$

Calculation of TOTPENMUSE:

$$TOTPENMUSE = - \sum_{r \varepsilon EQRES} PUNDUSEMP_r * UNDUSEMP.L_r$$
$$- \sum_{r \varepsilon CURRES} POVERUSEMP_r * OVERUSEMP.L_r$$

Calculation of SUMOFINF:

1 If a TK-subproblem has just been solved:

$$SUMOFINF = \sum_{r \varepsilon EQRES} |OVERUSETK.L_{d,r} - UNDUSETK.L_{d,r}|$$
$$+ \sum_{r \varepsilon CURRES \setminus EQRES} OVERUSETK.L_{d,r}$$

2 If the master problem has just been solved:

$$SUMOFINF = \sum_{r \varepsilon EQRES} |OVERUSEMP.L_r - UNDUSEMP.L_r|$$
$$+ \sum_{r \varepsilon CURRES \setminus EQRES} OVERUSEMP.L_r$$

Calculation of DEGOFOPT:

If ZOPT.L > 0, do:

$$DEGOFOPT = \frac{OBJCONCOMB + SUMOBJTK + TOTPENMUSE}{ZOPT.L} * 100$$

If ZOPT.L < 0 , do:

$$DEGOFOPT = \frac{ZOPT.L}{OBJCONCOMB + SUMOBJTK + TOTPENMUSE} * 100$$

If XOPT.L = 0, DEGOFOPT is set to 0.

Appendix 3: Description of the variables used by the program MIX2.GMS

1 XDIV ~ Division variables. Declared over DIV x DIVVAR. XDIV{d,j}, d ε DIV *and* j ε DIVVAR is division variable j in division d (only used when $j \leq NDIVVAR_d$). $XDIV_{d,j}$ is initialized before each solve to:

a) $LODIVVAR_{d,j}$, if $LODIVVAR_{d,j} \neq -INF$.

b) $UPDIVVAR_{d,j}$, if $LODIVVAR_{d,j} = -INF$ *and* $UPDIVVAR_{d,j} \neq INF$.

c) 0, otherwise.

2) XHQ ~ Headquarter variables. Declared over HQVAR.XHQ_j, j ε HQVAR is headquarter variable j (XHQ is only used when NHQVAR > 0). XHQ_j is initialized before each solve to:

a) $LOHQVAR_j$, if $LOHQVAR_j \neq -INF$

b) $UPHQVAR_j$, if $LOHQVAR_j = -INF$ *and* $UPHQVAR_j \neq INF$.

c) 0, otherwise.

3) BUDVAR ~ Budget variables in the master problem. Declared over DV x RES. $BUDVAR_{d,r}$, d ε TKUNITS *and* r ε CURES represents division d's budget of resource r. $BUDVAR_{d,r}$ is initialized before each solve of the master problem to 0.

4) PROFIT ~ Profit variables in the master problem for the budget controlled divisions. $PROFIT_d$, d ε TKUNITS represents division d's profit contribution. $PROFIT_d$ is initialized to 0 before each solve of the master problem.

5) LAMDA ~ Weight variables in the master problem . Declared over ITER x DIV. $LAMDA_{i,d}$, i ε CURITER *and* d ε DWUNITS represents the weight

given to the vertex generated in division d in iteration i. $LAMDA_{i,d}$ is initialized to 0 before each solve of the master problem.

6) OVERUSETK ~ Overuse variables in the budget controlled division. Declared voer DIV x RES. $OVERUSETK_{d,k}$, d ε TKUNITS *and* r ε CURRES represents division d's overuse of resource r. $OVERUSETK_{d,r}$ is initialized to 0 before each solve of the model TKDIV.

7) UNDUSETK ~ As above, but for underusing. Also declared over DIV x RES. Only relevant for r ε EQRES. $UNDUSETK_{d,r}$ is initialized to 0 before each calculation of the model TKDIV.

8) OVERUSEMP ~ As for OVERUSETK, but for the master problem. Declared over RES. Initialized to 0 before each calculation of the master problem.

9) UNDUSEMP ~ As for UNDUSETK, but for the master problem. Declared over RES and initialized to 0 before each calculation of the master problem.

10) ZDW ~ Objective variable in the price controlled division. Initialized to 0 before each calculation of the model DWDIV.

11) ZTK ~ Objective variable in the budget controlled divisions. Initialized to 0 before each calculation of the model TKDIV.

12) ZOPT ~ Objective variable when solving the overall problem. Initialized to 0 before each calculation of the model OPTIMAL.

13) ZMASTER ~ Objective variable in the master problem. Initialized to 0 before each calculation of the model MASTER.

Besides giving a primal initial solution, the program also gives a dual initial solution:

a. The dual variables corresponding to constraints 1), 4), 7)-12) and 15)-32) are all initialized to 0.
b. The dual variables corresponding to constraints 5)-6) are initialized to:
   i: The previous values of these variables
   ii: 0, if it is he first solve.
c. The dual variables corresponding to constraints 2)-3) and 13)-14) are initialized to the relevant price vector at the given moment of execution.

A further development of the program could be in supplying more information to the solver.

Appendix 4: Definition and discussion of the constraints and models used by the program MIX2.GMS.

The constraints used by the program mix2.gms:

1. OPTOBJ in the program:

$$ZOPT = \sum_{d \in curdiv} \sum_{\substack{j \in DIVVAR \\ j \le NDIVVAR(d)}} CDIV_{dj} * XDIV_{dj}$$

$$+ \sum_{CURHQVAR} CHQ_j * XHQ_j$$

2. Named OPRESEQCON in the program:

   For every r ε EQRES:

$$\sum_{J \in CURHQVAR} AHQ_{rj} * XHQ_j + \sum_{d \varepsilon CURDIV} \sum_{\substack{j \in DIVVAR \\ j \le NDIVVAR(d)}} ADIV_{d,r,j} * XDIV_{dj}$$

$$= AMOUNT_r$$

3 Named OPRESLECON in the program:

$$\sum_{J \in CURHQVAR} AHQ_{rj} * XHQ_j + \sum_{d \varepsilon CURDIV} \sum_{\substack{j \in DIVVAR \\ j \le NDIVVAR(d)}} ADIV_{d,r,j} * XDIV_{dj}$$

$$\le AMOUNT_r$$

4. Named TKOBJ in the program:

   For every d ε CURDIV:

$$\sum_{j \,\varepsilon\, CURVAR} CDIV_{d,j} * XDIV_{d,j} - \sum_{r \,\varepsilon\, EQRES} PUNDUSETK_{d,r} * UNDUSETK_{d,r}$$

$$- \sum_{r \,\varepsilon\, CURRES} POVERUSETK_{d,r} * OVERUSETK_{d,r} = ZTK.$$

5. Named BUDLECON in the program:

   For every d ε CURDIV:

   For every r ε CURRES/EQRES:

$$\sum_{j \,\varepsilon\, CURVAR} ADIV_{d,r,j} * XDIV_{d,j} - OVERUSETK_{d,r} \leq BUDGET_{d,r}.$$

6. Named BUDEQCON in the program

   For every d ε CURDIV:

   For every r ε EQRES:

$$\sum_{j \,\varepsilon\, CURVAR} ADIV_{d,r,j} * XDIV_{d,j} - OVERUSETK_{d,r} = BUDGET_{d,r}.$$

7. Named BDIVLECON in the program:

   For every d ε CURDIV

   For every c ε DIVCON satisfying c ≤ NDIVCON (d) and (d,c) ε DIVXDIVCON\EQDIV:

$$\sum_{\substack{j \,\in\, DIVVAR \\ j \,\leq\, NDIVVAR(d)}} BDIV_{d,c,j} * XDIV_{d,j} \leq RHSDIV_{d,c}.$$

8. Named BDIVEQCON in the program:

   For every d ε CURDIV:

   For every c ε DIVCON satisfying c ≤ NDIVCON (d) and (d,c) ε EQDIV:

$$\sum_{\substack{j \in DIVVAR \\ j \le NDIVVAR(d)}} BDIV_{d,c,j} * XDIV_{d,j} = RHSDIV_{d,c}.$$

9. Named DWOBJ in the program:

   For every d ε CURDIV:

$$\sum_{J \varepsilon curvar} (CDIV_{d,j} - \sum_{r \varepsilon CURRES} PRICE_r * ADIV_{d,r,j} = ZDW.$$

10. Named MASTOBJ in the program:

$$ZMASTER = \sum_{j \varepsilon CURHQVAR} CHQ_j * XHQ_j + \sum_{i \varepsilon DONEITER} \sum_{d \varepsilon DWUNITS} W_{i,d} * LAMDA_{i,d}$$
$$+ \sum_{r \varepsilon TKUNITS} PROFIT_d - \sum_{r \varepsilon CURRES} POVERUSEMP_r * OVERUSEMP_r$$
$$- \sum_{r \varepsilon EQRES} PUNDUSEMP_r * UNDUSEMP_r$$

11. Named BHQLECON in the program:

    For every c ε CURHQCON\EQHQ:

$$\sum_{j \varepsilon CURHQVAR} BHQ_{c,j} * XHQ_j \le RHSHQ_c.$$

12. Named BHQEQCON in the program:

    For every c ε EQHQ:

$$\sum_{j \varepsilon CURHQVAR} BHQ_{c,j} * XHQ_{c,j} = RHSHQ_c.$$

13. Named RESLECON in the program:

    For every r ε CURRES\EQRES:

$$\sum_{j \,\varepsilon\, \text{CURHQVAR}} AHQ_{r,j} * XHQ_j + \sum_{i \,\varepsilon\, DONEITER} \sum_{d \,\varepsilon\, DWUNITS} L_{i,d,r} * LAMDA_{i,d} \cdot$$

$$+ \sum_{d \,\varepsilon\, TKUNITS} BUDVAR_{d,r} - OVERUSEMP_r \leq AMOUNT_r \cdot$$

14. Named RESEQCON in the program:

    For every r ε EQRES:

$$\sum_{j \,\varepsilon\, \text{CURHQVAR}} AHQ_{r,j} * XHQ_j + \sum_{i \,\varepsilon\, DONEITER} \sum_{d \,\varepsilon\, DWUNITS} L_{i,d,r} * LAMDA_{i,d} \cdot$$

$$+ \sum_{d \,\varepsilon\, TKUNITS} BUDVAR_{d,r} - OVERUSEMP_r + UNDUSEMP_r = AMOUNT_r \cdot$$

15. Named LCON in the program:

    For every i ε DONEITER:

    For every d ε TKUNITS:

$$\sum_{j \,\varepsilon\, CURRES} COEFFBUD_{i,d,r} * BUDVAR_{d,r} \geq RHSL_{i,d} \cdot$$

16. Named CONVES in the program:

    For every d ε DWUNITS:

$$\sum_{i \,\varepsilon\, DONEITER} LAMDA_{i,d} = 1.$$

17. Not explicitly named in the program:

    For every j ε CURHQVAR:

    $XHQ_j \geq LOHQVAR_j$

18. Not explicitly named in the program:

For every j ε CURHQVAR:

$XHQ_j \leq UPHQVAR_j$

19. Not explicitly named in the program:

For every d ε CURDIV:

For every j ε DIVVAR satisfying j ≤ NDIVVAR (d):

$XDIV_{d,j} \geq LODIVVAR_{d,j}$

20 . Not explicitly named in the program:

For every d ε CURDIV:

For every j ε DIVVAR satisfying j ≤ NDIVVAR (d)

$XDIV_{d,j} \leq UPDIVVAR_{d,j}$

21. Not explicitly named in the program:

For every i ε DONEITER:

For every d ε DWUNITS:

$LAMDA_{i,d} \geq 0$

22. Not explicitly named in the program:

For every d ε CURDIV:

For every r ε CURRES:

$OVERUSETK_{d,r} \geq 0.$

23. Not explicitly named in the program:

For every d ε CURDIV:

For every r ε EQRES:

$UNDUSETK_{d,r} \geq 0$

24. Not explicitly named in the program:

    For every r ε CURRES:

    $OVERUSEMP_r \geq 0$.

25. Not explicitly named in the program:

    For every r ε EQRES:

    $UNDUSEMP_r \geq 0$.

26. Not explicitly named in the program:

    For every d ε CURDIV:

    ZDW ≤ BOBJDW (d)

27. Not explicitly named in the program:

    For every d ε CURDIV

    ZTK ≤ BOUNDTK(d)

28. Not explicitly named in the program:

    ZMASTER ≤ BOUNDMP

29. Named DWBOUND in the program:

    For every d ε CURDIV:

$$\sum_{j \,\varepsilon\, CURVAR} CDIV_{d,j} * XDIV_{d,j} \leq BPROFITDW_{d}.$$

30. Not explicitly named in the program:
    For every d ε TKUNITS:
    For every r ε CURRES:
    $BUDVAR_{d,r} \leq UPBUDVAR_{d,r}$

31. Not explicitly named in the program:
    For every d ε TKUNITS:
    For every r ε CURRES:
    $BUDVAR_{d,r} \geq LOBUDVAR_{d,r}$

Definition and discussion of the models used by the program:

1. The model OPTIMAL:
   Maximize ZOPT
   subject to:
   1-3, 7-8, 11-12, and 17-20

When OPTIMAL is solved, CURDIV, NDIVVAR, NDIVCON, CDIV, BDIV, RHSDIV, CURHQCON, EQHQ, BHQ, RHSHQ, LOHQVAR, UPHQVAR, LODIVVAR and UPDIVVAR will always contain the relevant data for the current planning period as specified in the data file and/or in the "change-file". Notice that CURDIV will contain *all* divisions in the current planning period.

2. The model DWDIV:
   Maximize ZDW

subject to:

7-9, 19-20, 26 and 29

When DWDIV is solved, CURDIV, NDIVCON, EQDIV, BDIV, PRICE, RHSDIV, NDIVVAR, CDIV, CURRES, ADIV, LODIVVAR, UPDIVVAR, CURVAR, BOBJDW and BPROFITDW will always contain the relevant data at the time of execution when DWDIV is solved. Notice that CURDIV will contain only one division index namely the division index of that division for which DWDIV is solved. CURVAR will contain the indexes from DIVVAR relevant for the division. PRICE will contain either:

a A user supplied price vector.

b The O-vector

c The price vector supplied by the last solution of the master problem.

3. The model TKDIV:

When TKDIV is solved, CURDIV, CURVAR, CDIV, EQRES, BUNDUSETK, POVERUSETK, ADIV, NDIVCON, EQDIV, BUDGET, NDIVVAR, BDIV, RHSDIV, LODIVVAR, UPDIVVAR and BOUNDTK will always contain the relevant at the time of execution when TKDIV solved. BUDGET will contain either:

a. A user supplied budget matrix

b. The 0-matrix.

c. The budget matrix supplied by the last solution of the master problem.

4. The model MASTER:

Maximize ZMASTER

Subject to:

10-18, 21, 24-25, 28 and 30-31

When MASTER is solved, CURHQVAR, CURHQCON, EQHQ, BHQ, RHSHQ, CURRES,

EQRES, AHQ, DONEITER, DWUNITS, L, W, CHQ, TKUNITS, AMOUNT, COEFFBUD, RHSL, LOHQVAR, UPHQVAR, PUNDUSEMP, POVERUSEMP and BOUNDMP will always contain the relevant data at the time of execution when MASTER is solved.

Appendix 5: Data for the Hoc Case - format file

```
SETS
 DIV       /1*%1/        /* IndexSET for the DIVisions (1. argument
                            given by the user in calling format.gms. */
 RES       /1*%2/        /* IndexSET for the RESources (2. argument). */
 DIVVAR    /1*%3/        /* IndexSET for the DIVisionVARiables (3. argument). */
 DIVCON    /1*%4/        /* IndexSET for the DIVisionCONstraints (4. argument). */
 HQVAR     /1*%5/        /* IndexSET for the HeadQuarterVARiables (5. argument
                            - remember it must be at least 1). */
 HQCON     /1*%6/        /* IndexSET for the HeadQuarterCONstraints (6.
                            argument - remember it must be at least 1). */
 ITER                    /1*%7/      /* IndexSET for the ITERations (7.
                                        argument). */
 PLANPER   /1*%8/;       /* IndexSET for the PLANningPERiods (8. argument). */

/* Below comes the DECLARATION and INITIALISATION of the relevant SCALARS in the
program. All SCALARS can be changed in the datafile implementing the data for the 1.
planning period, and also changed in the file implementing the CHANGEs from one
planning period to the next. */

SCALARS
 NDIV      /0/           /* Number of DIVisions. */
 NRES      /0/           /* Number of RESources. */
 NHQVAR    /0/           /* Number of HeadQuarterVARiables. */
 NHQCON    /0/           /* Number of HeadQuarterCONstraints. */
 NITER     /0/           /* Number of ITERations. */
 BOUNDMP   /1.0E+7/      /* BOUND on the optimal objective function values in the
                            MasterProblem. */
```

```
PRICEI      /0/          /* Non-zero if Initial PRICEs are to be set in the planning
                            period and 0 otherwise. */
BUDGETI     /0/          /* As above but for BUDGETs. */
SHARERES    /0/          /* Non-zero if the divisions are to SHARE the RESources
                            in the planning period and 0 otherwise. */
SOLTOL      /1.0E-6/     /* TOLerence for stopping the loop over the iterations
                            (calculated as the absolute change in the prices and
                            budgets from one iteration
                            to the next). */
APPEND      /0/          /* Non-zero if the user wishes the degree of optimality to
                            be written to an "APPEND-file" (in a comma separated
                            way) and 0 otherwise. */
CHANGE      /0/          /* Non-zero if the user wishes to include a "CHANGE-file"
                            implementing the CHANGEs from one planning period
                            to the next and 0 otherwise. */
HQPRINT     /0/          /* Non-zero if the user wishes the optimal values of the
                            HeadQuartervariables to be PRINTed in the "result-file"
                            and 0 otherwise. */
INITERPRPR  /0/;         /* Non-zero if the user wishes the PRices to be PRinted IN
                            the loop over the ITERations and 0 otherwise. The
                            prices will always be printed at the end of each planning
                            period. */

/* DECLARATION of all PARAMETERS relevant for the program. */

PARAMETERS
BOBJDW(DIV)              /* Bound on the optimal OBJectivevalues in the DW-units
                            (price controlled DIVisions). */
BPROFITDW(DIV)           /* Bound on the PROFITcontributions from the DW-units
```

```
                                (price controlled DIVisions). */
BOUNDTK(DIV)                    /* BOUND on the optimal objectivevalues in the TK-units
                                (budget controlled DIVisions). */
AHQ(RES,HQVAR)                  /* RESource usage by the HeadQuarterVARiables. */
BHQ(HQCON,HQVAR)                /* Coefficients for the HeadQuarterVARiables in the
                                "technical" CONstraints in the HeadQuarter. */
CHQ(HQVAR)                      /* Profit contributions from the HeadQuarte rVARiables.
                                */
ADIV(DIV,RES,DIVVAR)            /* RESource usage by the DIVisionVARiables. */
BDIV(DIV,DIVCON,DIVVAR) /* Coefficients for the DIVisionVARiables in the
                                "technical" DIVisionCONstraints. */
CDIV(DIV,DIVVAR)                /* Profit contributions from the DIVisionVARiables. */
EQHQCON(HQCON)                  /* Non-zero if the "technical" HeadQuarter-CONstraint
                                must be binding and 0 otherwise. */
EQDIVCON(DIV,DIVCON)            /* Non-zero if the "technical" DIVisionCONstraint must be
                                binding and 0 otherwise. */
EQRESCON(RES)                   /* Non-zero if the RESourceCONstraint must be binding
                                and zero otherwise. */
POVERUSETK(DIV,RES)             /* Price (or Penalty) for OVERUSing a given Resource in
                                a budgetcontrolled DIVision. */
PUNDUSETK(DIV,RES)              /* As above but for UNDerUSing. */
POVERUSEMP(RES)                 /* Price (or Penalty) for OVERUSing a given Resource in
                                the MasterProblem. */
PUNDUSEMP(RES)                  /* As above but for UNDerUSing. */
RHSHQ(HQCON)                    /* "Right-Hand-Sides" in the "technical"
                                HeadQuarterCONstraints. */
RHSDIV(DIV,DIVCON)              /* "Right-Hand-Sides" in the "technical" DIVisionCON-
                                straints. */
AMOUNT(RES)                     /* AMOUNT available of a given RESource. */
```

```
NDIVVAR(DIV)          /* The Number of DIVisionVARiables in a given DIVision.
                         */
NDIVCON(DIV)          /* The Number of "technical" DIVisionCONstraints in a
                         given DIVision. */
DW(DIV)               /* Non-zero if the given DIVision is supposed to be price-
                         controlled and 0 otherwise. */
INITP(RES)            /* INITial Prices (PRICEI non-zero). */
INITBUD(DIV,RES)      /* INITial BUDgets (BUDGETI non-zero). */
BETA(DIV,ITER)        /* Cheating parameters. */
UPHQVAR(HQVAR)        /* UPper bounds on the HeadQuarterVARiables. */
UPDIVVAR(DIV,DIVVAR)  /* UPper bounds on the DIVisionVARiables. */
LOHQVAR(HQVAR)        /* LOwer bounds on the HeadQuarterVARiables. */
LODIVVAR(DIV,DIVVAR)  /* LOwer bounds on the DIVisionVARiables. */
DIVPRINT(DIV)         /* Non-zero if the user wishes the optimal solution values
                         for the DIVision variables to be PRINTed in the "result-
                         file"and 0 otherwise. */
INITERBPR(DIV)        /* Non-zero if the user whishes the Budget for the given
                         (budget-controlled) DIVision to be PRinted IN the loop
                         over the ITERations and 0 otherwise. The Budgets
                         found at the end of a given planning period will always
                         be printed if there are any budget-controlled DIVisions.
                         */
LOBUDVAR(DIV,RES)     /* LOwer bounds on the BUDgetVARiables. */
UPBUDVAR(DIV,RES);    /* UPper bounds on the BUDgetVARiables. */

/* INITIALIZATION of all PARAMETERS relevant for the program (DEFAULT values).
Again all PARAMETERS can be changed in the data file and in the file implementing the
changes from one planning period to the next. */
```

```
BOBJDW(DIV)=1.0E+6;
BPROFITDW(DIV)=1.0E+6;
BOUNDTK(DIV)=1.0E+6;
AHQ(RES,HQVAR)=0;
BHQ(HQCON,HQVAR)=0;
CHQ(HQVAR)=0;
ADIV(DIV,RES,DIVVAR)=0;
BDIV(DIV,DIVCON,DIVVAR)=0;
CDIV(DIV,DIVVAR)=0;
EQHQCON(HQCON)=0;              /* DEFAULT non-binding. */
EQDIVCON(DIV,DIVCON)=0;       /* ---------------------------- */
EQRESCON(RES)=0;               /* ---------------------------- */
POVERUSETK(DIV,RES)=1.0E+5;
PUNDUSETK(DIV,RES)=1.0E+5;
POVERUSEMP(RES)=1.0E+5;
PUNDUSEMP(RES)=1.0E+5;
RHSHQ(HQCON)=0;
RHSDIV(DIV,DIVCON)=0;
AMOUNT(RES)=0;
NDIVVAR(DIV)=0;                /* NOTICE : MUST BE CHANGED. */
NDIVCON(DIV)=0;                /* ----------------------------------------. */
DW(DIV)=0;                     /* DEFAULT budget-controlled. */
INITP(RES)=0;
INITBUD(DIV,RES)=0;
BETA(DIV,ITER)=1;              /* DEFAULT no cheating. */
UPHQVAR(HQVAR)=INF;            /* DEFAULT no UPper bound. */
UPDIVVAR(DIV,DIVVAR)=INF;     /* ----------------------------. */
LOHQVAR(HQVAR)=-INF;           /* DEFAULT no LOwer bound. */
LODIVVAR(DIV,DIVVAR)=-INF;    /* ----------------------------. */
```

```
DIVPRINT(DIV)=0;                /* DEFAULT no PRINTing of
                                   DIVision variables.*/
INITERBPR(DIV)=0;               /* DEFAULT no PRinting of Budgets IN between
                                   ITERations. */
LOBUDVAR(DIV,RES)=-INF;         /* DEFAULT no LOwer bound. */
UPBUDVAR(DIV,RES)=INF;          /* DEFAULT no Upper bound. */
```

Appendix 6: Data for the Hoc Case - data file

```
/* Data for the 1. planning period for the test problem named HOCCASE. */

/* INITIALISATION of the non-zero elements of the PARAMETER CDIV. */
   CDIV('1','1')=-3.5;
   CDIV('1','2')=-0.027;
   CDIV('1','3')=-1.5;
   CDIV('1','4')=-3;
   CDIV('1','5')=-2.3;
   CDIV('1','6')=-3;
   CDIV('1','7')=-2;
   CDIV('2','1')=5.04;
   CDIV('2','2')=9.6;
   CDIV('2','3')=7;
   CDIV('2','4')=23;
   CDIV('2','5')=17.5;
   CDIV('2','6')=-0.027;

   NDIV=2;                  /* 2 DIVisions. */
   NDIVVAR('1')=7;          /* 7 VARiables in DIVision 1. */
   NDIVVAR('2')=6;          /* 6 VARiables in DIVision 2. */
   NHQVAR=0;                /* No HeadQuarterVARiables. */
   NHQCON=0;                /* No HeadQuarterCONstraints. */
   NRES=4;                  /* 4 RESources. */
   EQRESCON(RES)=1;         /* All RESourceCONstraints must be binding. */

/* INITIALISATION of the non-zero elements of the PARAMETER ADIV. */
   ADIV('1','1','1')=-1;
```

```
   ADIV('1','1','3')=-1;
   ADIV('1','2','5')=-1;
   ADIV('1','2','6')=-1;
   ADIV('1','3','4')=-1;
   ADIV('1','4','7')=-1;
   ADIV('2','1','1')=1;
   ADIV('2','1','2')=1;
   ADIV('2','1','3')=1;
   ADIV('2','2','1')=1;
   ADIV('2','2','2')=1;
   ADIV('2','2','3')=1;
   ADIV('2','3','4')=1;
   ADIV('2','3','5')=1;
   ADIV('2','4','4')=1;
   ADIV('2','4','5')=1;

/* The AMOUNTs available of each RESource is zero. */
   AMOUNT(RES)=0;

   NDIVCON('1')=3;  /* 3 DIVisionCONstraints in DIVision 1. */
   NDIVCON('2')=5;  /* 5 DIVisionCONstraints in DIVision 2. */

/* INITIALISATION of the non-zero elements of the PARAMETER RHSDIV. */
   RHSDIV('1','1')=230;
   RHSDIV('1','2')=760;
   RHSDIV('1','3')=110;
   RHSDIV('2','1')=30;
   RHSDIV('2','2')=500;
   RHSDIV('2','3')=60;
```

```
RHSDIV('2','4')=6;
RHSDIV('2','5')=85;

UPDIVVAR(DIV,DIVVAR)=1000; /* Sets all UPper bounds to 1000. */
UPDIVVAR('1','2')=23;      /* Sets the UPper bound on VARiable (1,2) to 23. */
UPDIVVAR('2','2')=10;      /* Sets the UPper bound on VARiable (2,2) to 10. */
UPDIVVAR('2','3')=3;       /* Sets the UPper bound on VARiable (2,3) to 3. */
UPDIVVAR('2','4')=3;       /* Sets the UPper bound on VARiable (2,4) to 3. */
UPDIVVAR('2','5')=2;       /* Sets the UPper bound on VARiable (2,5) to 2. */
UPDIVVAR('2','6')=3;       /* Sets the UPper bound on VARiable (2,6) to 3. */

LODIVVAR(DIV,DIVVAR)=0;    /* Sets all LOwer bounds to 0. */
LODIVVAR('1','1')=-5;      /* Sets the LOwer bound on VARiable (1,1) to -5. */
LODIVVAR('1','6')=-2;      /* Sets the LOwer bound on VARiable (1,6) to -2. */

EQDIVCON(DIV,DIVCON)=0;    /* No DIVisionCONstraint is binding. */

/* INITIALISATION of the non-zero elements of the PARAMETER BDIV. */
BDIV('1','1','2')=-1;
BDIV('1','1','3')=5;
BDIV('1','1','4')=7;
BDIV('1','2','5')=16;
BDIV('1','2','7')=12;
BDIV('1','3','3')=1;
BDIV('1','3','4')=1;
BDIV('1','3','5')=2;
BDIV('1','3','7')=2;
BDIV('2','1','4')=6;
BDIV('2','1','5')=6;
```

```
BDIV('2','1','6')=-1;
BDIV('2','2','1')=8;
BDIV('2','2','2')=12;
BDIV('2','2','3')=12;
BDIV('2','3','1')=1;
BDIV('2','3','2')=1.5;
BDIV('2','3','3')=1.5;
BDIV('2','4','4')=1;
BDIV('2','4','5')=1;
BDIV('2','5','1')=1.08;
BDIV('2','5','2')=1.92;
BDIV('2','5','3')=1.4;
BDIV('2','5','4')=4.6;
BDIV('2','5','5')=1.5;

NITER=11;          /* 11 ITERations in the 1. planningperiod. */
PRICEI=1;          /* Initial PRICEs are used. */
BUDGETI=1;         /* Initial BUDGETs are used. */
APPEND=1;          /* Data should be written to an "APPEND-file". */
CHANGE=1;          /* A "CHANGE-file" should be included in the program.
                      */

POVERUSETK('1',RES)=50;   /*   Prices for OVERUSing in division
                              1. */
PUNDUSETK('1',RES)=50;    /*   Prices for UNDerUSing in division
                              1. */
POVERUSETK('2',RES)=10;   /*   Prices for OVERUSing in division
                              2. */
PUNDUSETK('2',RES)=10;    /*   Prices for UNDerUSing in division
```

```
                                        2. */
POVERUSEMP(RES)=10;     /*              Prices for OVERUSing in the
                                        MasterProblem. */
PUNDUSEMP(RES)=10;      /*              Prices for UNDerUSing in the
                                        MasterProblem. */
BOUNDMP=200;            /*              BOUND on the optimal
                                        objectivevalues in the Master-
                                        Problem. */
BOUNDTK('1')=-100;      /* BOUND on the optimal objectivevalues in division 1.
                           */
BOUNDTK('2')=400;       /* BOUND on the optimal objectivevalues in division 2.
                           */
```

Appendix 7: Data for the Hoc Case - Change-file

```
/* File containing the changes from 1 planning period to the next. */

/* If we're at the end of planningperiod 1. */
   IF( ORD(PLANPER) EQ 1,

/* Below comes the changes in the data to be implemented from the start of planningperiod
2. */

   NITER=14;                    /*  14 ITERations in the 2. planningperiod. */
   DW('1')=1;                   /*  Division 1 is pricecontrolled. */
   DW('2')=0;                   /*  Division 2 is budgetcontrolled. */
   BOBJDW('1')=10000;           /*  Bound on the optimal OBJectivevalues in division 1
                                    (the pricecontrolled division). */
   BPROFITDW('1')=-100;         /*  Bound on the PROFITcontributions  from division 1
                                    (the pricecontrolled division). */
   POVERUSEMP(RES)=150;         /*  Prices for OVERUSing in the MasterProblem. */
   PUNDUSEMP(RES)=150;          /*  Prices for UNDerUSing in the MasterProblem. */

    );                              /* End-IF. */

/* If we're at the end of planningperiod 2. */
   IF(ORD(PLANPER) EQ 2,

/* Below comes the changes in the data to be implemented from  the beginning of
planningperiod 3. */

  NITER=5;                      /*  5 ITERations in planningperiod 3. */
```

```
   DW('1')=0;                  /* Division 1 is budgetcontrolled. */
   DW('2')=1;                  /* Division 2 is pricecontrolled. */
   BPROFITDW('2')=400;         /* Bound on the PROFITcontributions from division 2
                                  (the pricecontrolled division). */
   BOBJDW('2')=10000;          /* Bound on the optimal OBJectivevalues in division 2
                                  (the pricecontrolled division). */
   POVERUSEMP(RES)=10;         /* Prices for OVERUSing in the MasterProblem. */
   PUNDUSEMP(RES)=10;          /* Prices for UNDerUSing in the MasterProblem. */
     );                           /* End-IF. */

/* If we're at the end of planningperiod 3. */
   IF(ORD(PLANPER) EQ 3,

/* Below comes the changes in the data to be implemented from the beginning of
planningperiod 4. */

   NITER=11;                   /* 11 ITERations in planningperiod 4. */
   DW('1')=1;                  /* Division 1 is pricecontrolled. */
   DW('2')=1;                  /* Division 2 is pricecontrolled. */
   POVERUSEMP(RES)=150;        /* Prices for OVERUSing in the MasterProblem. */
   POVERUSEMP(RES)=150;        /* Prices for UNDerUSing in the MasterProblem. */
     );                        /* End-IF. */
```

Appendix 8: Specification File

```
$ONINLINE
/* The above statement gives allowance to write comments in the program. */

$BATINCLUDE FORMAT.GMS 2 4 7 5 1 1 14 4
/* BAT INCLUsion of the file FORMAT.GMS with the given numbers. The 1.
   number is the maximum number of divisions in any planning period in
   the test problem. The 2. number is the maximum number of resources in
   any planning period in the test problem. The 3. number is the maximum
   number of division variables in any division in any planning period in
   the test problem. The 4. number is the maximum number of division constraints
   in any division in any planning period in the test problem.The 5. number is  the
   maximum of 1 and the maximum number of headquarters variables in any
   planning period in the test problem. The 6. number is the maximum of 1 and the
   maximum number of headquarters constraints in any planning period in the
   test problem. The 7. number is the maximum number of iterations in any
   planning period in the test problem. The last number is the number of
   planning periods in the test problem. All arguments must be present. */

$BATINCLUDE MIX2.GMS HOCCASE HOCDATA.INC HOCCH.INC HOCAPP HOCRES
/* BAT INCLUsion of the file MIX2.GMS with the given arguments. The 1.
   argument is the name of the test problem. The 2. argument is the name
   of the data file (terminated with a .INC). The 3. argument is the name
   of the file containing the statements implementing the changes from
   one planning period to the next (terminated with a .INC). The 4. argument
   is the name of the "append-file". The last argument is the name of the file
   to write the results to. All arguments must be present. */
```

Appendix 9: OrgSim program

```
/* ********************** 1 ********************************** */

$OFFLISTING     /* $-Option telling the GAMS-compiler not to
                   LIST the program in the LST-file (see the GAMS-manual). */

/* ********************** 2 ********************************** */

$OFFSYMXREF
$OFFSYMLIST              /* Various $-OPTIONs and ordinary OPTIONs */
$OFFUELLIST              /* used to minimize the size of the */
$OFFUELXREF              /* LST-file produced by the solver */
OPTION SOLPRINT=OFF;  /* (see the GAMS-manual). */
OPTION LIMROW=0;
OPTION LIMCOL=0;
OPTION SYSOUT=OFF;

/* ********************** 3 ********************************** */

FILE APPFILE  /%4/;          /*  Declaration of the "APPend-FILE" (4. argument given
                                 by the user). */

APPFILE.AP=1;                /*  Option telling the GAMS-compiler to Append to the
                                 FILE APPFILE. */

FILE RESULTS  /%5/;          /*  Declaration of the "RESULTS-file" (5. argument given
                                 by the user). */
```

```
$INCLUDE %2                    /*  INCLUsion of the datafile (2. argument */
                               /* given by the user). */

/* ********************  4 ********************************** */

/* Writing the testname (1. argument given by the user) to the
   "APPend-FILE" if the user has chosen to have one (APPEND non-zero). */

   IF(APPEND,
      PUT APPFILE;
      PUT  '%1,';
      PUTCLOSE APPFILE;
     );

/* ********************  5 ********************************** */

/* Writing the headline to the "RESULTS-file". */

   PUT RESULTS;
   PUT / 'RESULTS FOR THE TESTPROBLEM NAMED %1.' //;
/* ********************  6 ********************************** */

/* DECLARATION of the indexSETS CURDIV, CURRES,CURVAR
   CURHQVAR, CURHQCON, CURITER, DONEITER and PRINTSET. */

SETS
 CURDIV(DIV)                   /*  SubSET of DIV containing the relevant DIVision(s) at
                                   a given moment of execution. */
 CURRES(RES)                   /*  SubSET of RES containing the relevant RESource(s)
```

```
                                    at  a given moment of execution. */
  CURVAR(DIVVAR)            /*  SubSET of DIVVAR containing the relevant DIVision-
                                VARiables at a given moment of execution. */
  CURHQVAR(HQVAR)           /*  SubSET of HQVAR containing the relevant
                                HeadQuarter-VARiables at a given moment of
                                execution. */
  CURHQCON(HQCON)           /*  SubSET of HQCON containing the relevant
                                HeadQuarter-CONstraints at a given moment of
                                execution. */
  CURITER(ITER)             /*  SubSET of ITER containing the relevant ITERations
                                at a given moment of execution. */
  DONEITER(ITER)            /*  SubSET of ITER containing the finished ITERations at
                                a given moment of execution. */
  PRINTSET(DIV);            /*  SubSET of DIV containing the indexes of those
                                DIVisions whose solution values should be PRINTed.
                                */

/* ********************** 7 ********************************** */

/* DECLARATION of the SETS EQRES, EQDIV and EQHQ. */

SETS
  EQRES(RES)                /*  SubSET of RES containing the RESources that must
                                balance (at a given ... ). */
  EQDIV(DIV,DIVCON)         /*  SubSET of the cartesian product of the 2 SETS DIV
                                and DIVCON containing the indexes of those
                                "technical" DIVisionCONstraints that must balance (at
                                a given .... ). */
  EQHQ(HQCON);              /*  SubSET of HQCON containing the indexes of those
```

```
                                HeadQuarterCONstraints that must balance (at a
                                given .... ).

/* ********************* 8 ******************************** */

ALIAS(DIV,LOOPDIV);          /*  Other names for the SETS DIV and ITER */
ALIAS(ITER,LOOPITER);        /*  (used to LOOP over). */

/* ********************* 9 ******************************** */

SETS
 TKUNITS(DIV)                /*  Subset of DIV containing TK-UNITS (at a given ..). */
 DWUNITS(DIV);               /*  Subset of DIV containing DW-UNITS(at a given ..). */

/* ********************* 10 ******************************** */

PARAMETERS
 W(ITER,DIV)                 /*  Profitcontribution from a given DIVision in a given
                                 ITERation (for the DW-UNITS at a given ..). */
 L(ITER,DIV,RES)                 /*          RESource demand from a given
                                          DIVision in a given ITERation (at a
                                          given ..). */
 COEFFBUD(ITER,DIV,RES)          /*          COEFFicients for the
                                          BUDget-variables in the
                                          L-constraints for a given DIVision
                                          in a given ITERation (at a given ..).
                                          */
 RHSL(ITER,DIV)              /*  "Right Hand Sides" in the L-constraints. (at a given ..).
                                 */
```

```
 PI(DIV,RES)              /* The relevant PI-matrix at a given moment of
                             execution. */
 PRICE(RES)               /* Internal PRICEvector. */
 BUDGET(DIV,RES);            /*              Internal BUDGETmatrix. */

/* ********************** 11 ******************************** */

VARIABLES
 XDIV(DIV,DIVVAR)         /* DIVisionVARiables. */
 XHQ(HQVAR)               /* HeadQuarterVARiables. */
 BUDVAR(DIV,RES)          /* BUDgetVARiables. */
 PROFIT(DIV);             /* PROFITvariables in the masterproblem. */

POSITIVE VARIABLES
 LAMDA(ITER,DIV)          /* LAMDAvariables (weights). */
 OVERUSETK(DIV,RES)       /* OVERUSEvariables in the TK-UNITS. */
 UNDUSETK(DIV,RES)        /* UNDerUSEvariables in the TK-UNITS. */
 OVERUSEMP(RES)           /* OVERUSEvariables in the MasterProblem. */
 UNDUSEMP(RES);           /* UNDerUSEvariables in the MasterProblem. */

VARIABLES
 ZTK                      /* Objectivevariable in TK-UNITS. */
 ZDW                      /* Objectivevariable in DW-UNITS. */
 ZMASTER                  /* Objectivevariable in the MASTERproblem. */
 ZOPT;                    /* Objectivevariable for the overall problem (OPTimum).
                             */

/* ********************** 12 ******************************** */
```

```
/* Below comes the DECLARATION of the relevant constraints. */

EQUATIONS
 TKOBJ(DIV)                 /* OBJective functions for the TK-DIVisions. */
 DWOBJ(DIV)                 /* OBJective functions for the DW-DIVisions. */
 MASTOBJ                    /* OBJective function for the MASTerproblem. */
 BUDLECON(DIV,RES)          /* "Less then or Equal" BUDget CONstraints in the
                               TK-DIVisions. */
 BUDEQCON(DIV,RES)          /*                    As above (EQuality CONstraints).
                               */
 BDIVLECON(DIV,DIVCON)      /* "Less then or Equal" "technical" CONstraints in any
                               DIVision for the DIVisionvariables. */
 BDIVEQCON(DIV,DIVCON)      /* As above (EQuality CONstraints). */
 BHQLECON(HQCON)            /* "Less then or Equal" "technical" CONstraints for the
                               HeadQuartervariables. */
 BHQEQCON(HQCON)            /* As above (EQuality CONstraints). */
 RESLECON(RES)              /* "Less then or Equal" RESourceCONstraints in the
                               masterproblem. */
 RESEQCON(RES)              /* As above (EQuality CONstraints). */
 LCON(ITER,DIV)             /* L-CONstraints in the masterproblem (for
                               TK-DIVisions). */
 CONVEX(DIV)                /* CONVEXity-constraints in the masterproblem
                               (DW-DIVisions). */
 OPTOBJ                     /* OBJective function when solving the overall problem
                               (finding OPTimum). */
 OPRESLECON(RES)            /* "Less then or Equal" RESourceCONstraints when
                               solving the overall problem (finding OPtimum). */
 OPRESEQCON(RES)            /* As above (EQuality CONstraints). */
 DWBOUND(DIV);              /* Constraint BOUNDing the profitcontributions in the
```

```
                                DW-DIVisions. */

/* ********************** 13 ********************************** */

/* Below comes the DEFINITION of the relevant constraints. */

/* The SUM of the profitcontributions from all DIVisions, plus the profitcontribution from the
HeadQuarter. */
   OPTOBJ..
    SUM(CURDIV,SUM(DIVVAR$( ORD(DIVVAR) LE NDIVVAR(CURDIV) ),
          CDIV(CURDIV,DIVVAR)*XDIV(CURDIV,DIVVAR)))
   +SUM(CURHQVAR,XHQ(CURHQVAR)*CHQ(CURHQVAR)) =E= ZOPT;

/* For a given RESource belonging to EQRES, the sum of all
   RESourcedemands from the DIVisions for that RESource, plus
   the demand from the HeadQuarter for that RESource must EQual
   the available quantity of that RESource ( AMOUNT(EQRES) ). */
   OPRESEQCON(EQRES)..
    SUM(CURHQVAR,AHQ(EQRES,CURHQVAR)*XHQ(CURHQVAR))
   +SUM(CURDIV,SUM(DIVVAR$( ORD(DIVVAR) LE NDIVVAR(CURDIV) ),
          ADIV(CURDIV,EQRES,DIVVAR)*XDIV(CURDIV,DIVVAR)))
    =E= AMOUNT(EQRES);


/* As above for the "Less-than or Equal" CONstraints. */
   OPRESLECON(CURRES)$( NOT EQRES(CURRES) )..
    SUM(CURHQVAR,AHQ(CURRES,CURHQVAR)*XHQ(CURHQVAR))
   +SUM(CURDIV,SUM(DIVVAR$( ORD(DIVVAR) LE NDIVVAR(CURDIV) ),
          ADIV(CURDIV,CURRES,DIVVAR)*XDIV(CURDIV,DIVVAR)))
```

```
    =L= AMOUNT(CURRES);

/* The profitcontribution from the (CURrent) DIVision, minus the Penalty payed for
OVERUSing or UNDerUSing the RESources */
   TKOBJ(CURDIV)..
    SUM(CURVAR,CDIV(CURDIV,CURVAR)*XDIV(CURDIV,CURVAR))
   -SUM(CURRES,
          POVERUSETK(CURDIV,CURRES)*OVERUSETK(CURDIV,CURRES))
   -SUM(EQRES,
           PUNDUSETK(CURDIV,EQRES)*UNDUSETK(CURDIV,EQRES))
    =E= ZTK;

/* CONstraint telling the DIVision CURDIV to use an amount of "Less then
   or Equal" to the amount specified in BUDGET of a given RESource
   NOT belonging to the RESourceindexset EQRES. If that's not possible
   an amount represented by the VARiable OVERUSETK is subtracted from it's
   use of that RESource. */
   BUDLECON(CURDIV,CURRES)$( NOT EQRES(CURRES) )..
    SUM(CURVAR,ADIV(CURDIV,CURRES,CURVAR)*XDIV(CURDIV,CURVAR))
   -OVERUSETK(CURDIV,CURRES)
   =L= BUDGET(CURDIV,CURRES);

/* As above but for RESources that must balance. A second variable UNDUSETK must be
present here because it's possible to "UNDerUSE" a given RESource. */
   BUDEQCON(CURDIV,EQRES)..
    SUM(CURVAR,ADIV(CURDIV,EQRES,CURVAR)*XDIV(CURDIV,CURVAR))
  -OVERUSETK(CURDIV,EQRES)+UNDUSETK(CURDIV,EQRES)
   =E= BUDGET(CURDIV,EQRES);
```

```
/* "Technical" "Less then or Equal" restrictions in the DIVisions. */
   BDIVLECON(CURDIV,DIVCON)$( (ORD(DIVCON) LE NDIVCON(CURDIV))
                                   AND (NOT EQDIV(CURDIV,DIVCON)) )..
    SUM(DIVVAR$( ORD(DIVVAR) LE NDIVVAR(CURDIV) ),
          BDIV(CURDIV,DIVCON,DIVVAR)*XDIV(CURDIV,DIVVAR))
    =L= RHSDIV(CURDIV,DIVCON);

/* "Technical" binding restrictions in the DIVisions( EQualities ). */
   BDIVEQCON(CURDIV,DIVCON)$( EQDIV(CURDIV,DIVCON) )..
    SUM(DIVVAR$( ORD(DIVVAR) LE NDIVVAR(CURDIV) ),
          BDIV(CURDIV,DIVCON,DIVVAR)*XDIV(CURDIV,DIVVAR))
    =E= RHSDIV(CURDIV,DIVCON);

/* OBJective functions in the DW-UNITS. The profitcontribution from the DIVision CURDIV
minus the cost of the RESources it uses. */
   DWOBJ(CURDIV)..
    SUM(CURVAR, (CDIV(CURDIV,CURVAR)
   - SUM(CURRES,
          PRICE(CURRES)*ADIV(CURDIV,CURRES,CURVAR)))*
          XDIV(CURDIV,CURVAR))
    =E= ZDW;

/* The OBJective function in the MASTerproblem. It's the sum of the following:
     1) Profitcontribution from the HeadQuarter. (If there are any HeadQuarterVARiables).
     2) Profitcontribution from a CONVEX combination of vertices generated in the
        DW-UNITS.
     3) PROFIT contributions from the TK-UNITS.
        4)   Penalty from OVERUSing (UNDerUSing). */
   MASTOBJ..
```

```
    SUM(CURHQVAR, CHQ(CURHQVAR)*XHQ(CURHQVAR))
   +SUM( (DONEITER,DWUNITS),
            W(DONEITER,DWUNITS)*LAMDA(DONEITER,DWUNITS))
   +SUM(TKUNITS,PROFIT(TKUNITS))
   - SUM(CURRES,POVERUSEMP(CURRES)*OVERUSEMP(CURRES))
    -SUM(EQRES,PUNDUSEMP(EQRES)*UNDUSEMP(EQRES))
     =E= ZMASTER;

/* "Tecnical" "Less then or Equal" CONstraints for the HeadQuarterVARiables. */
   BHQLECON(CURHQCON)$( NOT EQHQ(CURHQCON) )..
    SUM(CURHQVAR,BHQ(CURHQCON,CURHQVAR)*XHQ(CURHQVAR))
    =L= RHSHQ(CURHQCON);

/* "Technical" binding CONstraints for the HeadQuartervariables. */
   BHQEQCON(EQHQ)..
    SUM(CURHQVAR,BHQ(EQHQ,CURHQVAR)*XHQ(CURHQVAR))
    =E= RHSHQ(EQHQ);

/* RESource "Less then or EQual" CONstraints in the MasterProblem. */
   RESLECON(CURRES)$( NOT EQRES(CURRES) )..
    SUM(CURHQVAR,AHQ(CURRES,CURHQVAR)*XHQ(CURHQVAR))
   +SUM( (DONEITER,DWUNITS),
           L(DONEITER,DWUNITS,CURRES)*LAMDA(DONEITER,DWUNITS))
   +SUM(TKUNITS,BUDVAR(TKUNITS,CURRES))
    -OVERUSEMP(CURRES)
     =L= AMOUNT(CURRES);

/* RESource binding CONstraints in the MasterProblem. */
   RESEQCON(EQRES)..
```

```
    SUM(CURHQVAR,AHQ(EQRES,CURHQVAR)*XHQ(CURHQVAR))
   +SUM( (DONEITER,DWUNITS),
         L(DONEITER,DWUNITS,EQRES)*LAMDA(DONEITER,DWUNITS))
   +SUM(TKUNITS, BUDVAR(TKUNITS,EQRES))
    -OVERUSEMP(EQRES) + UNDUSEMP(EQRES)
     =E= AMOUNT(EQRES);

/* L-CONstaints.( See notes ). */
   LCON(DONEITER,TKUNITS)..
    SUM(CURRES,COEFFBUD(DONEITER,TKUNITS,CURRES)*
                    BUDVAR(TKUNITS,CURRES))
    -PROFIT(TKUNITS)
     =G= RHSL(DONEITER,TKUNITS);

/* CONVEXity CONstraints. */
   CONVEX(DWUNITS)..
    SUM(DONEITER,LAMDA(DONEITER,DWUNITS)) =E= 1;

/* The profitcontribution from the DIVision CURDIV  must be less than BPROFITDW. */
    DWBOUND(CURDIV)..
     SUM(CURVAR,CDIV(CURDIV,CURVAR)*XDIV(CURDIV,CURVAR))
     =L= BPROFITDW(CURDIV);

/* ********************* 14 ********************************** */

/* DECLARATION of the relevant EQUATIONS for each MODEL. */

   MODEL DWDIV      /DWOBJ,BDIVLECON,BDIVEQCON,DWBOUND/;
   MODEL TKDIV      /TKOBJ,BUDLECON,BDIVLECON,
```

```
                    BUDEQCON,BDIVEQCON/;
  MODEL MASTER    /MASTOBJ,BHQLECON,BHQEQCON,RESLECON,
                    RESEQCON,LCON,CONVEX/;
  MODEL OPTIMAL   /OPTOBJ,BDIVLECON,BDIVEQCON,OPRESEQCON,
                    OPRESLECON,BHQLECON,BHQEQCON/;

/* ********************** 15 ********************************** */

SCALARS
  END             /0/       /* SCALAR telling us when to stop LOOPing over
                               ITERations (when END is non-zero). */
  TOL             /1/       /* SCALAR containing the measure used for stopping
                               the LOOP over the ITERations (when TOL is less than
                               SOLTOL). */
  SUMOBJTK        /0/       /* SCALAR that contains (after solving all subproblems)
                               the SUM of the OBJective values from the TK-units. */
  OBJCONCOMB      /0/       /* SCALAR containing (after solution of the MASTER-
                               problem) the objective value found from the CONvex
                               COMBination identified by the MASTER-problem. */
  DEGOFOPT        /0/       /* SCALAR containing the DEGree of OPTimality. */
  SUMOFINF        /0/       /* SCALAR that will contain (after solving either the
                               masterproblem or a TK-subproblem) the SUM OF
                               INFeasibility. */
  TOTPENMUSE      /0/;      /* SCALAR that will contain (after each solve of the
                               masterproblem) the TOTal PENalty for "MisUSing" the
                               resources. */

/* ********************** 16 ********************************** */
```

```
/* PARAMETERS containing respectivly the BUDgets and the PRICEs from the last
iteration. */

   PARAMETERS
     LASTBUD(DIV,RES)
     LASTPRICE(RES);

/* ********************** 17 ********************************** */

/* INITIALISATION of PRICEs and BUDGETs. */

    PRICE(RES)=0;
    BUDGET(DIV,RES)=0;

/* ********************** 18 ********************************** */

/* LOOPing over the Planning Periods. */

   LOOP(PLANPER,

/* ********************** 19 ********************************** */

/* INITIALISATION of the SETS CURDIV, CURRES, CURHQVAR, CURHQCON,
   CURITER, DONEITER and PRINTSET for the CURrent planningperiod. */

   CURDIV(DIV)=YES$( ORD(DIV) LE NDIV );
   CURRES(RES)=YES$( ORD(RES) LE NRES );
   CURHQVAR(HQVAR)=YES$( ORD(HQVAR) LE NHQVAR );
   CURHQCON(HQCON)=YES$( ORD(HQCON) LE NHQCON );
```

```
   CURITER(ITER)=YES$( ORD(ITER) LE NITER );
   DONEITER(ITER)=NO;
   PRINTSET(DIV)=YES$( DIVPRINT(DIV) AND ( ORD(DIV) LE NDIV ) );

/* ********************* 20 ********************************** */

/* INITIALISATION of PRICEs and BUDGETs as chosen by the user by specifying the 3
options PRICEI,BUDGETI and SHARERES. */

   IF(PRICEI,
      PRICE(CURRES)=INITP(CURRES);
     );
   IF(BUDGETI,
      BUDGET(CURDIV,CURRES)=INITBUD(CURDIV,CURRES);
     );
   IF(SHARERES,
      BUDGET(CURDIV,CURRES)=AMOUNT(CURRES)/CARD(CURDIV);
     );

/* ********************* 21 ********************************** */

/* INITIALISATION of the LASTBUD and LASTPRICE parameters. */

   LASTBUD(CURDIV,CURRES)=BUDGET(CURDIV,CURRES);
   LASTPRICE(CURRES)=PRICE(CURRES);

/* ********************* 22 ********************************** */

/* INITIALISATION of the SETS TKUNITS and DWUNITS for the CURrent
```

```
   planningperiod. */

  TKUNITS(DIV)=YES$( CURDIV(DIV) AND (NOT DW(DIV)) );
   DWUNITS(DIV)=YES$( CURDIV(DIV) AND DW(DIV) );

/* ********************* 23 ********************************** */

/* INITIALISATION of the SETS EQRES, EQDIV and EQHQ for the
   CURrent planningperiod. */

   EQRES(RES)=YES$( CURRES(RES) AND EQRESCON(RES) );
   EQDIV(DIV,DIVCON)=YES$( CURDIV(DIV)
                                  AND (ORD(DIVCON) LE NDIVCON(DIV))
                                  AND EQDIVCON(DIV,DIVCON) );
   EQHQ(HQCON)=YES$( CURHQCON(HQCON) AND EQHQCON(HQCON) );

/* ********************* 24 ********************************** */

/* INITIALISATION of various PARAMETERS used by the program
   for the CURrent planningperiod. */

   W(CURITER,CURDIV)=0;
   L(CURITER,CURDIV,CURRES)=0;
   RHSL(CURITER,CURDIV)=0;
   COEFFBUD(CURITER,CURDIV,CURRES)=0;
   PI(TKUNITS,CURRES)=0;

/********************** 25 ********************************** */
```

```
/* Initialisation of the "fields" on the variables used by the MODEL OPTIMAL in the
CURrent planningperiod. */

   XDIV.LO(CURDIV,DIVVAR)$( ORD(DIVVAR) LE NDIVVAR(CURDIV) )=
   LODIVVAR(CURDIV,DIVVAR);
   XDIV.UP(CURDIV,DIVVAR)$( ORD(DIVVAR) LE NDIVVAR(CURDIV) )=
   UPDIVVAR(CURDIV,DIVVAR);
   XDIV.L(CURDIV,DIVVAR)$( ORD(DIVVAR) LE NDIVVAR(CURDIV) )=0;
   XDIV.L(CURDIV,DIVVAR)$( (ORD(DIVVAR) LE NDIVVAR(CURDIV))
                                   AND (XDIV.UP(CURDIV,DIVVAR) NE INF) )=
   XDIV.UP(CURDIV,DIVVAR);
   XDIV.L(CURDIV,DIVVAR)$( (ORD(DIVVAR) LE NDIVVAR(CURDIV))
                                    AND (XDIV.LO(CURDIV,DIVVAR) NE -INF) )=
   XDIV.LO(CURDIV,DIVVAR);
   XDIV.M(CURDIV,DIVVAR)$( ORD(DIVVAR) LE NDIVVAR(CURDIV) )=0;
   XHQ.LO(CURHQVAR)=LOHQVAR(CURHQVAR);
   XHQ.UP(CURHQVAR)=UPHQVAR(CURHQVAR);
   XHQ.L(CURHQVAR)=0;
   XHQ.L(CURHQVAR)$( XHQ.UP(CURHQVAR) NE INF )=XHQ.UP(CURHQVAR);
   XHQ.L(CURHQVAR)$( XHQ.LO(CURHQVAR) NE -INF )=XHQ.LO(CURHQVAR);
   XHQ.M(CURHQVAR)=0;
   ZOPT.L=0;
   ZOPT.M=0;

/* ********************** 26 ********************************** */

/* Initialisation of the constraint-suffixes on the constraints used by the MODEL OPTIMAL.
*/
```

```
  BDIVLECON.L(CURDIV,DIVCON)$( ORD(DIVCON) LE NDIVCON(CURDIV) )=
  RHSDIV(CURDIV,DIVCON);
   BDIVLECON.M(CURDIV,DIVCON)$( ORD(DIVCON) LE NDIVCON(CURDIV) )=0;
  BDIVEQCON.L(CURDIV,DIVCON)$( ORD(DIVCON) LE NDIVCON(CURDIV) )=
  RHSDIV(CURDIV,DIVCON);
    BDIVEQCON.M(CURDIV,DIVCON)$( ORD(DIVCON) LE NDIVCON(CURDIV) )=0;
 OPRESLECON.L(CURRES)=AMOUNT(CURRES);
 OPRESLECON.M(CURRES)=PRICE(CURRES);
 OPRESEQCON.L(CURRES)=AMOUNT(CURRES);
 OPRESEQCON.M(CURRES)=PRICE(CURRES);
  BHQLECON.L(CURHQCON)=RHSHQ(CURHQCON);
  BHQLECON.M(CURHQCON)=0;
  BHQEQCON.L(CURHQCON)=RHSHQ(CURHQCON);
  BHQEQCON.M(CURHQCON)=0;


/* ********************** 27 ********************************** */

SOLVE OPTIMAL USING LP MAXIMIZING ZOPT;

/* ********************** 28 ********************************** */

PUT RESULTS;  /* Starting to write to the "resultfile" */

/* ********************** 29 ********************************** */

/* ABORTing the program if the overall problem is either unbounded or infeasibel, and
writing this information to the "RESULTS-file". */
```

```
IF(OPTIMAL.MODELSTAT EQ 3 OR OPTIMAL.MODELSTAT EQ 4,
   PUT '*********************************************************' /
        'THE OVERALL PROBLEM IN PLANNING PERIOD '
         PLANPER.TL:0 ' WAS EITHER INFEASIBEL OR UNBOUNDET.' /
        '*********************************************************' /;
   ABORT "";
  );    /* End IF. */

/* ********************* 30 ********************************** */

/* Now it's known that the overall problem is feasibel and boundet.
   The solution found (as chosen by HQPRINT and DIVPRINT) together with
   the optimal objective value is therefore written to the "RESULTS-file". */

  PUT '*************************************************************************' //;
  PUT 'THE OPTIMAL SOLUTION TO THE OVERALL PROBLEM IN PLANNING'
    ' PERIOD '  PLANPER.TL:0  ' IS :'  //  'OBJECTIVE VALUE : ' ZOPT.L:12:6 //;
  PUT$( (NHQVAR GT 0) AND HQPRINT )  'THE OPTIMAL VALUES FOR THE'
                                          ' HQ-VARIABLES ARE :' /;
  LOOP(CURHQVAR$( HQPRINT ),
       PUT 'XHQ(' CURHQVAR.TL:0 ') = ' XHQ.L(CURHQVAR):12:6 /;
      );    /* end-LOOP. */
  PUT$( CARD(PRINTSET) )  // 'THE OPTIMAL VALUES FOR THE '
                              'XDIV-VARIABLES ARE :' /;
  LOOP(PRINTSET,
       LOOP(DIVVAR$( ORD(DIVVAR) LE NDIVVAR(PRINTSET) ),
            PUT 'XDIV(' PRINTSET.TL:0 ',' DIVVAR.TL:0 ') = '
                XDIV.L(PRINTSET,DIVVAR):12:6 /
           );    /* end-LOOP. */
```

```
        );          /* end-LOOP. */
   PUT '*************************************************************************' /;

/* ********************* 31 ********************************** */

/* Initialisation of the END-scalar. */

   END=0;

/* ********************* 32 ********************************** */

/* LOOPING over the ITERations. */

   LOOP(LOOPITER$( (NOT END) AND (ORD(LOOPITER) LE NITER) ),

/* ********************* 33 ********************************** */

/* Initialising SUMOBJTK. */

   SUMOBJTK=0;

/* ********************* 34 ********************************** */

/* LOOPing over all DIVisions. */

   LOOP(LOOPDIV$( ORD(LOOPDIV) LE NDIV ),

/* ********************* 35 ********************************** */
```

```
/* Setting CURDIV and CURVAR to contain the indexes of the CURrent
   DIVision and the CURrent VARiables respectivly. */

   CURDIV(DIV)=YES$(ORD(DIV) EQ ORD(LOOPDIV));
   CURVAR(DIVVAR)=YES$(ORD(DIVVAR) LE NDIVVAR(LOOPDIV));

/* ********************** 36 ********************************** */

/* Treating the " DW-case " */

    IF(DW(LOOPDIV),

/* ********************** 37 ********************************** */

/* Initialising the various fields on the variables used by
   the MODEL DWDIV. */

   ZDW.L=0;
   ZDW.M=0;
   ZDW.UP=BOBJDW(LOOPDIV);
   XDIV.L(LOOPDIV,CURVAR)=0;
   XDIV.L(LOOPDIV,CURVAR)$( XDIV.UP(LOOPDIV,CURVAR) NE INF )=
   XDIV.UP(LOOPDIV,CURVAR);
   XDIV.L(LOOPDIV,CURVAR)$( XDIV.LO(LOOPDIV,CURVAR) NE -INF )=
   XDIV.LO(LOOPDIV,CURVAR);
   XDIV.M(LOOPDIV,CURVAR)=0;

/* ********************** 38 ********************************** */
```

```
/* Initialising the various "constraint-suffixes" for the constraints
   used by the MODEL DWDIV. */

   DWOBJ.L(LOOPDIV)=0;
   DWOBJ.M(LOOPDIV)=0;
    BDIVLECON.L(LOOPDIV,DIVCON)$( ORD(DIVCON) LE NDIVCON(LOOPDIV) )=
   RHSDIV(LOOPDIV,DIVCON);
  BDIVLECON.M(LOOPDIV,DIVCON)$( ORD(DIVCON) LE NDIVCON(LOOPDIV) )=0;
  BDIVEQCON.L(LOOPDIV,DIVCON)$( ORD(DIVCON) LE NDIVCON(LOOPDIV) )=
  RHSDIV(LOOPDIV,DIVCON);
  BDIVEQCON.M(LOOPDIV,DIVCON)$( ORD(DIVCON) LE NDIVCON(LOOPDIV) )=0;
  DWBOUND.L(LOOPDIV)=0;
  DWBOUND.M(LOOPDIV)=0;

/* ********************** 39 ********************************** */

    SOLVE DWDIV MAXIMIZING ZDW USING LP;

/* ********************** 40 ********************************** */

/* It's known that the DW-subproblem was feasibel and boundet.
   The solution (as chosen by DIVPRINT) and the optimal objective value
   is therefore written to the file RESULTS. */

   PUT 'THE RESULTS FOR DIVISION ' LOOPDIV.TL:0 ' IN ITERATION '
         LOOPITER.TL:0 ' OF PLANNING PERIOD '
         PLANPER.TL:0 ' ARE :' // 'OBJECTIVE VALUE : ' ZDW.L:12:6 //;
   PUT$( DIVPRINT(LOOPDIV) ) 'THE OPTIMAL LEVELS OF
                                    THE VARIALES ARE :' /;
```

```
    LOOP(CURVAR$( DIVPRINT(LOOPDIV) ),
            PUT 'XDIV(' LOOPDIV.TL:0 ',' CURVAR.TL:0 ') = '
            PUT XDIV.L(LOOPDIV,CURVAR):12:6 /;
          );    /* end-LOOP. */
  PUT '************************************************************'
        '****************' /;

/* ********************* 41 ********************************** */

/* Assigning values to the W-parameter. */

  W(LOOPITER,LOOPDIV)=BETA(LOOPDIV,LOOPITER)*DWBOUND.L(LOOPDIV);

/* ********************** 42 ********************************** */

/* Assigning values to the L-parameter. */

  L(LOOPITER,LOOPDIV,CURRES)=BETA(LOOPDIV,LOOPITER)*
  SUM(CURVAR,ADIV(LOOPDIV,CURRES,CURVAR)*XDIV.L(LOOPDIV,CURVAR));

/* ********************** 43 ********************************** */

   ELSE     /* Treating the " TK-case " */

/* ********************** 44 ********************************** */

/* Initialising various "fields" on the variables used in the
   MODEL TKDIV. */
   ZTK.L=0;
```

```
   ZTK.M=0;
   ZTK.UP=BOUNDTK(LOOPDIV);
   XDIV.L(LOOPDIV,CURVAR)=0;
   XDIV.L(LOOPDIV,CURVAR)$( XDIV.UP(LOOPDIV,CURVAR) NE INF )=
   XDIV.UP(LOOPDIV,CURVAR);
   XDIV.L(LOOPDIV,CURVAR)$( XDIV.LO(LOOPDIV,CURVAR) NE -INF )=
   XDIV.LO(LOOPDIV,CURVAR);
   XDIV.M(LOOPDIV,CURVAR)=0;
   OVERUSETK.L(LOOPDIV,CURRES)=0;
   UNDUSETK.L(LOOPDIV,CURRES)=0;
   OVERUSETK.M(LOOPDIV,CURRES)=0;
   UNDUSETK.M(LOOPDIV,CURRES)=0;

/* ********************** 45 ********************************** */

/* Initialising various "constraint-suffixes" on the constraints
   used by the MODEL TKDIV. */

   TKOBJ.L(LOOPDIV)=0;
   TKOBJ.M(LOOPDIV)=0;
    BUDLECON.L(LOOPDIV,CURRES)=BUDGET(LOOPDIV,CURRES);
    BUDLECON.M(LOOPDIV,CURRES)=PI(LOOPDIV,CURRES);
    BUDEQCON.L(LOOPDIV,CURRES)=BUDGET(LOOPDIV,CURRES);
    BUDEQCON.M(LOOPDIV,CURRES)=PI(LOOPDIV,CURRES);
    BDIVLECON.L(LOOPDIV,DIVCON)$( ORD(DIVCON) LE NDIVCON(LOOPDIV) )=
   RHSDIV(LOOPDIV,DIVCON);
  BDIVLECON.M(LOOPDIV,DIVCON)$( ORD(DIVCON) LE NDIVCON(LOOPDIV) )=0;
    BDIVEQCON.L(LOOPDIV,DIVCON)$( ORD(DIVCON) LE NDIVCON(LOOPDIV) )=
   RHSDIV(LOOPDIV,DIVCON);
```

```
  BDIVEQCON.M(LOOPDIV,DIVCON)$( ORD(DIVCON) LE NDIVCON(LOOPDIV) )=0;

/* ********************* 46 ********************************** */

    SOLVE TKDIV MAXIMIZING ZTK USING LP;


/* ********************* 47 ********************************** */

/* Writing the optimal solution (as chosen by DIVPRINT) and the optimal
   objective value to the file RESULTS (the TK-subproblem can't be
   infeasibel or unboundet by construction). */

   PUT '************************************************************'
       '****************' //;
  PUT 'THE RESULTS FOR DIVISION ' LOOPDIV.TL:0 ' IN ITERATION
      ' LOOPITER.TL:0    ' OF PLANNING PERIOD ' PLANPER.TL:0 ' ARE :' //
      'OPTIMAL OBJECTIVE VALUE : ' ZTK.L:12:6 //;
  PUT$( DIVPRINT(LOOPDIV) ) 'THE OPTIMAL VALUES OF THE
                                    VARIABLES ARE :' //;
  LOOP(CURVAR$( DIVPRINT(LOOPDIV) ),
        PUT 'XDIV(' LOOPDIV.TL:0 ',' CURVAR.TL:0 ') = '
            XDIV.L(LOOPDIV,CURVAR):12:6 /;
       );

/* ********************* 48 ********************************** */

/* Calculating SUMOFINF, and printing it to the "RESULTS-file". */
```

```
  SUMOFINF = SUM(EQRES,
  ABS(OVERUSETK.L(LOOPDIV,EQRES) - UNDUSETK.L(LOOPDIV,EQRES)))
+SUM(CURRES$( NOT  EQRES(CURRES) ),
        OVERUSETK.L(LOOPDIV,CURRES));
  PUT 'THE SUM OF INFEASIBILITY IS : ' SUMOFINF:12:6 //;
  PUT '*****************************************************************' //;

/* ********************* 49 ********************************** */

/* Updating SUMOBJTK. */

  SUMOBJTK = SUMOBJTK + ZTK.L;

/* ********************** 50 ********************************** */

/* Assigning values to the L-parameter. */

   L(LOOPITER,CURDIV,EQRES)=
   BETA(CURDIV,LOOPITER)*BUDEQCON.L(CURDIV,EQRES);
   L(LOOPITER,CURDIV,CURRES)$( NOT EQRES(CURRES) )=
   BETA(CURDIV,LOOPITER)*BUDLECON.L(CURDIV,CURRES);



/*********************** 51 ********************************** */

/* Assigning values to the PI-vector. */

   PI(LOOPDIV,EQRES)=BUDEQCON.M(LOOPDIV,EQRES);
```

```
   PI(LOOPDIV,CURRES)$( NOT EQRES(CURRES) ) =
   BUDLECON.M(LOOPDIV,CURRES);

/********************** 52 ********************************** */

/* Assigning values to the RHSL-parameter. */

   RHSL(LOOPITER,CURDIV)= -ZTK.L*BETA(CURDIV,LOOPITER)
  +SUM(CURRES, L(LOOPITER,CURDIV,CURRES)*PI(CURDIV,CURRES));

/* ********************** 53 ********************************** */

/* Assigning values to the COEFFBUD-parameter. */

   COEFFBUD(LOOPITER,LOOPDIV,CURRES)=PI(CURDIV,CURRES);

/* ********************** 54 ********************************** */

   ); /* end-IF(DW(LOOPDIV,... */
  ); /* end-LOOP(LOOPDIV,... */

/* ********************** 55 ********************************** */

  DONEITER(LOOPITER)=YES; /* One more ITERation done. */


/* ********************** 56 ********************************** */

/* Initialising various "fields" on the variables used to
```

```
   solve the MODEL MASTER. */

   XHQ.L(CURHQVAR)=0;
   XHQ.L(CURHQVAR)$( XHQ.UP(CURHQVAR) NE INF )=XHQ.UP(CURHQVAR);
   XHQ.L(CURHQVAR)$( XHQ.LO(CURHQVAR) NE -INF )=XHQ.LO(CURHQVAR);
   XHQ.M(CURHQVAR)=0;
   PROFIT.L(TKUNITS)=0;
   PROFIT.M(TKUNITS)=0;
   LAMDA.L(DONEITER,DWUNITS)=0;
   LAMDA.M(DONEITER,DWUNITS)=0;
   OVERUSEMP.L(CURRES)=0;
   OVERUSEMP.M(CURRES)=0;
   UNDUSEMP.L(CURRES)=0;
   UNDUSEMP.M(CURRES)=0;
   ZMASTER.L=0;
   ZMASTER.M=0;
   ZMASTER.UP=BOUNDMP;
   BUDVAR.UP(TKUNITS,CURRES)=UPBUDVAR(TKUNITS,CURRES);
   BUDVAR.LO(TKUNITS,CURRES)=LOBUDVAR(TKUNITS,CURRES);
   BUDVAR.L(TKUNITS,CURRES)=BUDGET(TKUNITS,CURRES);
   BUDVAR.M(TKUNITS,CURRES)=0;

/* ********************** 57 ********************************** */

/* Initialising various "constraint-suffixes" on constraints
   used when solving the MODEL MASTER. */
   MASTOBJ.L=0;
   MASTOBJ.M=0;
   BHQLECON.L(CURHQCON)=RHSHQ(CURHQCON);
```

```
   BHQLECON.M(CURHQCON)=0;
   BHQEQCON.L(CURHQCON)=RHSHQ(CURHQCON);
   BHQEQCON.M(CURHQCON)=0;
   RESLECON.L(CURRES)=AMOUNT(CURRES);
   RESLECON.M(CURRES)=PRICE(CURRES);
   RESEQCON.L(CURRES)=AMOUNT(CURRES);
   RESEQCON.M(CURRES)=PRICE(CURRES);
   LCON.L(DONEITER,TKUNITS)=RHSL(DONEITER,TKUNITS);
   LCON.M(DONEITER,TKUNITS)=0;
   CONVEX.L(DWUNITS)=1;
   CONVEX.M(DWUNITS)=0;

/* ********************* 58 ********************************** */

  SOLVE MASTER MAXIMIZING ZMASTER USING LP;

/* ********************* 59 ********************************** */

/* Writing the optimal objective value to the file RESULTS. */

   PUT '************************************************************'
       '****************' //
       'THE RESULTS FOR THE MASTERPROBLEM IN ITERATION '
        LOOPITER.TL:0  ' OF PLANNING PERIOD ' PLANPER.TL:0 ' ARE :' //
       'OPTIMAL OBJECTIVE VALUE : ' ZMASTER.L:12:6 //;

/* ********************* 60 ********************************** */

/* Assigning new BUDGET's and new PRICE's for the next iteration. */
```

```
   BUDGET(TKUNITS,CURRES)=BUDVAR.L(TKUNITS,CURRES);
   PRICE(EQRES) = RESEQCON.M(EQRES);
   PRICE(CURRES)$( NOT EQRES(CURRES) )=RESLECON.M(CURRES);

/* ********************** 61 ********************************** */

/* Calculating OBJCONCOMB and TOTPENMUSE. */

   OBJCONCOMB = SUM( (DONEITER,DWUNITS),
   W(DONEITER,DWUNITS)*LAMDA.L(DONEITER,DWUNITS) );
   TOTPENMUSE =
  -SUM(EQRES,PUNDUSEMP(EQRES)*UNDUSEMP.L(EQRES))
  -SUM(CURRES,POVERUSEMP(CURRES)*OVERUSEMP.L(CURRES));

/* ********************** 62 ********************************** */

/* Writing OBJCONCOMB and TOTPENMUSE to the "RESULTS-file"
   (only if there are DW-units). */

   PUT$( CARD(DWUNITS) )   'THE OBJECTIVE VALUE OF THE CONVEX' /
                           'COMBINATION FOUND BY'
                           ' THE MASTERPROBLEM IS : ' OBJCONCOMB:12:6 //
                           'THE TOTAL PENALTY FOR "MISUSING" THE '
                           'RESOURCES IS : ' TOTPENMUSE:12:6 //;

/* ********************** 63 ********************************** */

/* Writing SUMOBJTK to the "RESULTS-file" (only if there are TK-units). */
```

```
   PUT$( CARD(TKUNITS) )     'THE SUM OF THE OBJECTIVE VALUES FROM' /
                             'THE TK-UNITS IS : ' SUMOBJTK:12:6 //;


/* ********************** 64 ********************************** */

/* Calculating the DEGree OF OPTimality. */

   DEGOFOPT$( ZOPT.L GT 0 )=
   (OBJCONCOMB+SUMOBJTK+TOTPENMUSE)*100/ZOPT.L;
   DEGOFOPT$( ZOPT.L LT 0 )=
   100*ZOPT.L/(OBJCONCOMB+SUMOBJTK+TOTPENMUSE);

/* ********************** 65 ********************************** */

/* Writing DEGOFOPT to the "RESULTS-file". */

   PUT 'THE DEGREE OF OPTIMALITY IS : ' DEGOFOPT:12:6 //;

/* ********************** 66 ********************************** */

/* Calculation of SUMOFINF and writing it to the "RESULTS-file". */

    SUMOFINF =
    SUM(EQRES,ABS(OVERUSEMP.L(EQRES) - UNDUSEMP.L(EQRES)))
  +SUM(CURRES$( NOT EQRES(CURRES) ),OVERUSEMP.L(CURRES));

    PUT 'THE SUM OF INFEASIBILITY IS : ' SUMOFINF:12:6 //;
```

```
/* ********************** 67 ********************************** */

/* Printing the optimal values for the HeadQuarterVARiables as
   chosen by the user through the option HQPRINT. */

   PUT$(HQPRINT AND NHQVAR) 'THE OPTIMAL SOLUTION FOR THE '
                                        'HEADQUARTERVARIABLES ARE :' //;
   LOOP(CURHQVAR$( HQPRINT ),
          PUT 'XHQ(' CURHQVAR.TL:0 ')= ' XHQ.L(CURHQVAR):12:6 /;
         );
   PUT$( HQPRINT ) /;

/* ********************** 68 ********************************** */

/* Printing the PRices and Budgets to the "results-file" as chosen by
   the user through the options INITERPRPR and INITERBPR. */

   LOOP(CURRES$( INITERPRPR ),
          PUT 'PRICE ON RESOURCE ' CURRES.TL:0 ' : ' PRICE(CURRES):12:6 /;
         );
   PUT$( INITERPRPR ) /;
   LOOP(TKUNITS$( INITERBPR(TKUNITS) ),
          LOOP(CURRES,
                 PUT 'DIVISION ' TKUNITS.TL:0 'S BUDGET OF RESOURCE '
                 CURRES.TL:0 ' IS : ' BUDGET(TKUNITS,CURRES):12:6 /;
                );
          PUT /;
         );
  PUT '************************************************************** /;
```

```
/********************** 69 ********************************** */

/* Calculation of the scalar TOL. If it's a pure Ten-Kate problem or
   a pure Danzig-Wolfe problem, only the budgets or the prices respectivly
   are used to calculate TOL */

   TOL = 0;
   TOL$( CARD(DWUNITS) )=
   SUM(CURRES,ABS(PRICE(CURRES)-LASTPRICE(CURRES)));
   TOL$( CARD(TKUNITS) )=TOL+SUM( (TKUNITS,CURRES),
         ABS(BUDGET(TKUNITS,CURRES)-LASTBUD(TKUNITS,CURRES)));

/* ********************** 70 ********************************** */

/* Updating LASTPRICE and LASTBUD. */

   LASTPRICE(CURRES)=PRICE(CURRES);
   LASTBUD(TKUNITS,CURRES)=BUDGET(TKUNITS,CURRES);

/* ********************** 71 ********************************** */

/* Stopping if TOL is smaller then the SCALAR SOLTOL (userspecified). */

   END$( TOL LT SOLTOL ) = 1;

/* ********************** 72 ********************************** */

); /* end-LOOP(LOOPITER,... */
/* ********************** 73 ********************************** */
```

```
/* Printing the BUDGETs and PRICEs found in the current planningperiod
   to the RESULTS-file. */

   PUT RESULTS;
   PUT 'THE PRICES FOUND IN PLANNINGPERIOD ' PLANPER.TL:0 ' ARE :' //;
   LOOP(CURRES,
          PUT 'RESOURCE ' CURRES.TL:0 ' : ' PRICE(CURRES):12:6 /;
        );
    PUT /;
    PUT$( CARD(TKUNITS) ) 'THE BUDGETS FOUND IN PLANNINGPERIOD '
                                    PLANPER.TL:0 ' ARE :' //;
    LOOP(TKUNITS,
          PUT 'FOR DIVISION ' TKUNITS.TL:0 ' THE BUDGET IS :' //;
          LOOP(CURRES,
                PUT 'RESOURCE ' CURRES.TL:0 ' : '
                      BUDGET(TKUNITS,CURRES):12:6 /;
                );
          PUT /;
        );
    PUT /;

/* ********************* 74 ********************************** */

/* Reading change data */
   IF(CHANGE,
      $INCLUDE %3
     );

/* ********************* 75 ********************************** */
```

```
/* Writing the DEGree OF OPTimality to the APPend-FILE, if one
    is specifyied by the user. */
    IF(APPEND,
        PUT APPFILE;
        PUT DEGOFOPT:12:6 ',';
        PUTCLOSE APPFILE;
       );

/* ********************* 76 ********************************** */

); /* end-LOOP(PLANPER,..... */

/* ********************* 77 ********************************** */

PUTCLOSE RESULTS;  /* stop writing to the file RESULTS. */
```

References

R. B. Burton and B. Obel, *Designing Efficient Organizations: Modeling and Experimentation*, North Holland, 1984.

R. B. Burton and B. Obel, Mathematical Contingency Modeling for Organizational Design: Taking Stock, in R. M. Burton and B. Obel, *Design Models for Hierarchical Organizations: Computation, Information and Decentralization*, Kluwer Academic Publishers, Boston, 1995, pp. 3-34.

B. Obel A Note on Mixed Procedures for Decomposing Linear Programming Problems, *Mathematische Operations Forschung und Statistik, Series Optimization,* Vol. 9, No. 4, 1978, pp. 537-544.