Adaptation of the GAMS-format model specs for MCMA

Mar-20-2019, 14:36

This note is based on the lessons from the pilot exploration of the MCMA-Message_ix interface made in 2017 & 2018 and suggests modifications of Message_ix aimed at easing multi-criteria analysis of models specified in Message_ix. This is a substantially extended and restructured version of the original (2017) note; in particular, the proposal of the GAMS-MCMA interface was updated based on the 2018 YSSP study and the current attempts to use MCMA with the Indus model, both models developed in Message_ix. Moreover, the note includes now the MCMA architecture outline. The proposed interface focuses on Message_ix but it shall also work for models developed in the standard GAMS and continue to work with models represented by the MPS-format files.

Section 4.4 written by Volker in 2017 requires revision to account for the Message_ix modifications since 2017.

1 Introduction

The note summarizes the requirements for, and proposed implementation of the MCMA–Message_ix interface. The GAMS-format model instance specification file is further on referred to as the (core) model specs.[1] We aim at keeping the required modifications of the standard Message_ix specification at a minimum, however some modifications are necessary in order to keep MCMA applicability to models developed with other than Message_ix modeling environments.

The GAMS-format model specification integrates three elements of modeling process:

1. Core model instance specification composed of:
   - a version of specification of variables and relations (constraints);
   - a version of data defining values of the model parameters.
2. Specification of the optimization objective and optimization solver.
3. Specification of the post-optimization processing.

The note is written from the perspective of the Message_ix modifications;[2] therefore, the main part of the note is structured according to the above summarized elements of the GAMS specification. The main part is preceded by the outline of the approach and followed by the summary and supplementary material. Thus, the note consists of the following sections:

- Outline of the approach (Sec. 2).
- GAMS-format model specification (Sec. 3).
  - ◇ The basic assumptions (Sec. 3.1).
  - ◇ Core model requirements (Sec. 3.2).
  - ◇ Optimization of the merged model (Sec. 3.3).
  - ◇ Post-optimization processing (Sec. 3.4).
  - ◇ Technical requirements (Sec. 4).
  - ◇ Summary of the requirements for GAMS model specs (Sec. 4.6).

[1] Actually, this term is used here for an model instance, i.e., a given version of the symbolic model symbolic and a selected version of data used for the model parameter.

[2] The corresponding MCMA modifications have already been implemented and tested.

Supplementary material is available in Appendices:

- Example of specification of outcome variables (Appendix A.1).
- Sample of `_mc.gms` file (Appendix A.2).
- Shared space for ENE applications of MCMA (Appendix B).
- Functionality and architecture of the MCMA tool (Appendix C).

2 Outline of the approach

MCMA extends single-criterion model functionality by replacing optimization criterion of the core model instance (further-on called core model) by maximization of the Scalarizing Achievement Function (SF), the function of the outcome variables selected to serve as criteria in a specific analysis. The SF is interactively parametrized by the users who specify preferences for criteria values. Thus, the original core model is not modified; instead, it is merged with a dynamically (for each specification of preferences) generated small MC-submodel defining the SF.

For effective application of MCMA to a core model instance its specification has to conform to the requirements presented in Sec. 3. These requirements are easy to follow and do not influence single-criterion (and other types of) model analysis.

To provide the context of the approach we now briefly summarize the Mathematical Programming view on single-criterion and multi-criteria optimization of LP models.

2.1 Preliminaries

From the mathematical programming point of view, a model represents the corresponding problem by two types of entities: variables and relations between them, all these entities but one are treated the same way, i.e., regardless of their role in the problem representation. Many problems are described by linear, often dynamic and spatial, models. A standard mathematical programming formulation of such models takes the form:

$$\underline{\mathbf{b}} \leq \mathbf{A} \cdot \mathbf{x} \leq \overline{\mathbf{b}} \tag{1}$$

where vector $\mathbf{x}$ is composed of all model variables, the matrix $\mathbf{A}$, as well as vectors $\underline{\mathbf{b}}$ and $\overline{\mathbf{b}}$ are the model parameters. For the brevity sake we use the standard compact formulation (1) that covers also the bound-type constraints of the core model.

For a properly specified decision-making problem the system of relations (1) has infinitely many solutions. Thus one needs a criterion for selecting a solution that fits best the user preferences. Models of complex problems involve many variables but in model analysis one typically focuses on a small subset of all model variables. In particular, the preferences are typically represented as a function of a subset of all model variables called criteria (aka outcomes, indicators, etc). In this note we use the terms outcomes and criteria, depending on the context, as explained below.

To ease the discussion we shall, depending on the context, consider either all model variables as one compound[3] variable $\mathbf{x}$ or its split into components according to the roles diverse variables represent:

$$\mathbf{x} = \{\mathbf{y}, \mathbf{z}\}, \tag{2}$$

where vectors $\mathbf{y}$ and $\mathbf{z}$ are composed of variables representing outcomes and all other model variables, respectively. The criteria, denoted by $\mathbf{q}$, are interactively selected from outcomes $\mathbf{q}$ by the each MCMA user; therefore, $\mathbf{q} \in \mathbf{y}$.

[3]Compound variable means a (multidimensional) vector of variables.

2.2 Single-criterion optimization

Mathematical Programming Problem (MPP)[4] of a linear model (traditionally called LP or LPP) can be formulated as:

$$\text{minimize } \{obj \;=\; f(\mathbf{y})\} \tag{3}$$

subject to:

$$\underline{\mathbf{b}} \leq \mathbf{A} \cdot \begin{bmatrix} \mathbf{y} \\ \mathbf{z} \end{bmatrix} \leq \overline{\mathbf{b}}, \qquad \text{and} \tag{4}$$

$$\underline{\mathbf{y}} \leq \mathbf{y} \leq \overline{\mathbf{y}} \qquad \text{(optional)}, \tag{5}$$

where:

- the optimization criterion obj is defined by (3); the function $f(\mathbf{y})$ takes diverse forms, depending on the applied representation of preferences, e.g., a selected criterion, or a utility function, or a weighted sum of criteria, or a composite criterion defined as a function of outcome variables and penalty terms, etc;
- reformulation (4) of the core-model (1) with taking into account (2);
- optional bounds (5) on criteria values augmenting a representation of preferences through the optimization criterion (3); approaches based on application of sequences of lower/upper bounds in (5) is called *parametric optimization* and used as a surrogate of MCA.

2.3 Multi-Criteria submodel

MCMA exploits the concept of maximization of a Scalarizing Achievement Function (SF) that represents the user preferences; the SF is interactively parametrized by the user during the analysis. While the SF interpretation is intuitive and easy, its specification in terms of mathematical programming is not straightforward; therefore the SF is specified through an auxiliary LP model, further on called the MC sub-model, which is generated for each preference specification. The MC sub-model is merged with the core model (4) and optimized with the same solver as used for the single-criterion optimization of the core model. Therefore, MCMA does not involve any modification of the core model (4), except of skipping (5) (because including bounds on criteria values would cut-off a part of the set of feasible solutions of the core model).

The MC-submodel defines small sets of own variables and relations, as well as uses the core-model variables representing criteria $\mathbf{q} \in \mathbf{y}$. We stress that the merged (MC-submodel and core model) model has the same set of feasible solutions as the core model (4).

The MC-submodel takes the form:

$$\text{maximize } \{sf = SF(\mathbf{q})\} \tag{6}$$

subject to:

$$\underline{\mathbf{d}} \leq \mathbf{D} \cdot \begin{bmatrix} sf \\ \mathbf{v} \\ \mathbf{q} \in \mathbf{y} \end{bmatrix} \leq \overline{\mathbf{d}}, \tag{7}$$

where:

- variable sf is equal to the value of $SF(\mathbf{q})$; discussion of the $SF(\cdot)$ interpretation and specification is beyond the scope of this note;[5]
- the auxiliary variables $\mathbf{v}$ are generated for defining the sf and for internal scaling of the criteria values;
- variables $\mathbf{q}$ represent criteria and are shared with the core model (4);[6]

[4] Aka optimization problem.

[5] Detailed discussion of the SF is available e.g., in [3], an overview in [8], methodological background in [12].

[6] Note that the other (not selected to be criteria) outcome variables (i.e., $\mathbf{y} \setminus \mathbf{q}$) are not included into the MC-submodel.

• $\mathbf{D}, \underline{\mathbf{d}}, \overline{\mathbf{d}}$ are parameters of the MC-submodel.

Note that relations (6) and (7) define an optimization submodel that needs to be merged with a core model defining all logical and physical relations that variables $\mathbf{q}$ (representing criteria) have to conform to.

2.4 Merged MC-submodel and a core model

The MC-submodel and the core-model are merged into an instance of optimization task (shown in Figure 2 on page 13 as *Optimization Instance*). The structure of the merged model's constraints is illustrated by the merged model Jacobian shown in Figure 1. The optimization criterion of the merged model is defined by (6).

	Variables			
merged models' variables	sf	$\mathbf{v}$	$\mathbf{y}$	$\mathbf{z}$
MC-submodel variables	sf	$\mathbf{v}$	$\mathbf{q} \in \mathbf{y}$	
core-submodel variables			$\mathbf{y}$	$\mathbf{z}$
MC-submodel parameters	$\mathbf{D}$			0
core-model parameters	0		$\mathbf{A}$	

Figure 1: Structure of the Jacobian of the merged MC-submodel and the core model.

The numbers of rows and columns of the MC-submodel are small, usually between 15 and 50, depending on the criteria number and the needs of adaptive criteria scaling. Therefore, the computational requirements of the MCMA are practically the same as of single-criterion optimization.

To complete the outline of the merged MC and core models let us comment on sharing the $\mathbf{q}$ between the MC-submodel and the core model. This is easily achieved by using the same names of $\mathbf{q}$ variables in both models, i.e., MC-submodel and the core-model instance. To assure this consistency, the MCMA user selects the criteria $\mathbf{q}$ from the list of outcome variables $\mathbf{y}$ specified in the core model. In order to ease the selection the names of $\mathbf{y}$ should be provided in the master.cfg configuration file (see Sec. 4.5).

3 GAMS-format model specification

Specification of the GAMS-format model prepared for MCMA should contain the following elements, each discussed in the corresponding section:
• Specification of the outcome variables $\mathbf{y}$ (Sec. 3.2);
• Declaration of the merged model and its optimization (Sec. 3.3);
• Post-optimization processing (Sec. 3.4).
Moreover, the GAMS-format specs needs to conform to the technical requirements summarized in Sec. 4.

3.1 The basic assumptions

Although the proposed interface focuses on enabling MCMA of models developed in `Message`_*ix*, it also aims at such core model specification that as well supports diverse types of model analysis, in particular enables use of the specification without any modifications for:
• single-criterion optimization,
• parametric single-criterion optimization in form of (5), and

• multi-criteria analysis with the MCMA.

The selection of each of these model analysis types shall be done by specification of the corresponding option in the GAMS execution command but should not require any modifications of the model specification. Such an approach shall greatly ease the model analysis process (composed of single- and multi-criteria optimizations) and well as support consistency of both types of analysis. Moreover, the given specification shall be self-contained, i.e., the optimization can be run on any computer running GAMS without accessing the model development environment.

MCMA executes GAMS[7] through the command:

```
gams master.gms -ll=2 -lo=2 --mcma=_mc.gms
```

Therefore, it is easy to implement *a conditional compilation* of parts of the GAMS specs; this can be done by enclosing the corresponding parts between the following pairs of the GAMS commands:

• to include only for MCMA: `$If not set mcma $GOTO label_a` and `$LABEL label_a`
• to exclude only for MCMA: `$If set mcma $GOTO label_b` and `$LABEL label_b`

Note that such a conditional compilation not only enables using one GAMS-specs for both single- and multi-criteria analysis but it also can be used (together with diverse options replacing the command option `--mcma`) for defining diverse single-criterion optimization tasks, e.g., for diverse objective functions or parametric optimizations.

3.2 Specification of the outcome variables

As pointed out in Sec. 2.1, outcome variables are typically specified[8] in models developed for facilitating model-based decision analysis and support. There are at least three popular model specification approaches in which some outcome variables are not explicitly defined but it is easy to define them through simple modifications of the specification:

• Single-criterion optimization objective (goal function) is often defined in LP as a neutral row. Such a row can be duplicated as a right-hand side of equation defining the corresponding outcome variable.
• Single-criterion optimization objective is sometimes defined as a relation composed of diverse elements (such as components of costs or penalty terms or weighted sum of sub-criteria, etc). In such situations a number of outcome variables can be defined by assignments to the corresponding element of such a composite objective.
• Another approach replaces bounds (5) on outcomes by defining lower/upper bounds on constraints representing the corresponding outcomes; e.g., total emissions or use of resources. Also such constraints can be replaced by explicit definitions of outcome variables.

3.2.1 Naming convention for outcome variables

In order to avoid conflicts in names of variables and constraints of the merged MMP, and to enable the outcome variables[9] of core model to represent criteria in MCMA, the following two naming **requirements** need to be met in the GAMS model specification:

1. The first three characters of all names of variables and constraints must differ from the `mC_` string.
2. Names of outcome variables have to conform to the following requirements:
   • the variable name length is maximum 8 characters;
   • the variable is not indexed.

3.2.2 Example of mcma_variables.gms

As an example, below the three blocks of code that need to be included in mcma_variables.gms are provided for a MESSAGE*ix* model with three criteria, i.e. cumulative discounted system costs, cumulative

[7]For the MC-submodel integrated with the core model.
[8]Although rarely called *outcome* variables.
[9]A typically small subset of all variables to be presented for interactive selection of criteria.

GHG emissions and cumulative water use. In addition, the file mcma_variables.gms may contain any additional (and syntactically correct) definitions, e.g., an additional MODEL statement for testing single criteria model versions with the newly defined variables as optimization criterion.

Declaration of equations in which the criteria variables are defined:

```
Equations
COST_CUMULATIVE summation of cumulative total discounted system costs
EMISSION_CUMULATIVE summation of cumulative total GHG emissions
WATER_CUMULATIVE  summation of cumulative total water consumption
;
```

Definition of (non-indexed) variables with a maximum of eight characters:

```
Variables
         CUMCOST
         CUMGHG
         CUMWATER
;
```

Definition of equations (example only for one equation fully spelled out, the other two replaced by . . .):

```
COST_CUMULATIVE..
    CUMCOST =E=
    sum((node,year), discountfactor(year) * COST_NODAL(node, year))
;
EMISSION_CUMULATIVE..  ...
WATER_CUMULATIVE..  ...
```

3.2.3 Guidelines for specification of outcome variables

In order to enable an appropriate MCMA of the core model the model specification should conform to the following **requirements**:

1. The core model should not define bounds on outcome variables that represent preferences (e.g., acceptable range of values for each outcome), see (5) and the associated discussion in Sec. 2.2; such bounds should only be defined if needed for representation of logical or physical constraints on values of outcomes.[10]
2. For each potential[11] criterion a corresponding outcome variable needs to be defined. Specification of all suitable outcome variables not only enables specification of diverse MCMA instances but also can provide additional characteristics of model solutions (outcome variables, even if not used as criteria, typically represent diverse informative metrics or indices).
3. Each outcome variable should either have a precisely defined measurement unit (e.g., monetary or physical) or be an established indicator. In other words, values of such variables should have for the model analysts a clear interpretation in terms of the corresponding unit. This feature is important for preference specification during the MCMA. Therefore, we suggest to refrain from defining outcome variables representing e.g., a composite objective or aggregated metrics unless their values have clear meaning.

[10]Recall that the GF of traditional optimization is typically defined by a neutral row, i.e., there is no constraint on optimization objective.

[11]During the model specification many potential criteria can be considered. Choice of a criteria set is interactively done for each MCMA instance (see Sec. C.1). Criteria selection for subsequent MCMA instances often depends on analysis of earlier defined instances. Therefore, it is rational to include into the core model specification definitions of possibly all outcome variables (note that defining another MCMA instance is by far easier than preparation of a new model instance and then starting a new MCA analysis).

4. One should define in the model specification as many outcome variables as helpful for evaluating different aspects of the model solution. Such definitions are very easy during the model development and do not increase computational requirements. A surplus (compared to a typically small number of outcomes initially considered for criteria) of outcome variables increase flexibility and efficiency of model instance analysis; moreover, outcome variables are also helpful in model verification.
5. Finally, it is recommended to include into the process of model verification runs of selfish optimization, i.e., to optimize each outcome variables separately.

3.3 Declaration of the merged model and its optimization

MCMA solver generates for each specification of the user preference the MC-submodel in the GAMS-format and stores in a file specified with the `--mcma` option of the run command:

```
gams master.gms -ll=2 -lo=2 --mcma=_mc.gms
```

Example of the `_mc.gms` file is provided in Appendix A.2. Therefore, merging the core model with the MC-submodel can be easily implemented by inserting just before the `SOLVE` statement (used for single-criterion optimization of the core model) the following statement:

```
$If set mcma $INCLUDE %mcma%
```

The included file contains also the declarations of the name of the merged MC-submodel and of the optimization task. In order to suppress (within MCMA optimization runs) single-criterion optimization of the core model its `SOLVE` statement(s) should be conditionally excluded from the compilation, e.g., in the way suggested in Sec. 3.1.

The core model specification should conform to the following **requirements**:

1. All specified entities (sets, parameters, variables, equations) of the model will be used in MCMA. This is equivalent to the statement (included in the `_mc.gms` file, see it sample in Sec. A.2):
   `MODEL model_name / all / ;`
   Therefore, parts of the specification not needed for the MCMA of the core model should be excluded from the compilation, e.g., as suggested in Sec. 3.1.
2. Specification of the model instance has to be self-contained. In particular, there is no possibility of passing to MCMA arguments of the GAMS-execution command-line (e.g., for selecting a data set). To replace the command-line option, e.g., for the GDX data file one can include in MESSAGE-MCA_master.gms the statement `$SETGLOBAL`data.
3. For the data-path (full directory name) separators the only / (slash) character should be used. Although also the \ (backslash) character works on a PC-version of GAMS it does not work on unix machines. Note that the slash separator works with GAMS correctly on all platforms.

3.4 Post-optimization processing

The post-processing specified with the core model is executed after successful optimization and the generated files are stored in the `wdir` or in its optional `output` subdirectory.

MC-submodel defines its postprocessing also in the `_mc.gms` file; therefore, several MCMA specific files are stored in each `wdir`. Names of all files generated by or for MCMA start with the underscore (_) character, thus are easily recognized.

4 Technical requirements for the model specs

The GAMS-format core model specification can be provided in two ways: either as a single file or as a zip-archive. The corresponding technical (i.e., other than those concerning core model specification, which are discussed above) requirements for each of these ways are presented below. We precede this presentation by summarizing the requirements common for both ways.

4.1 General requirements for GAMS-format specs

1. In order to avoid conflicts with names of files generated by MCMA, the core-model files should not start with the underscore (_) character.
2. Names of the provided files should:
   - be composed of Latin letters, optionally include digits, and/or the underscore (_) character and/or a single dot (.);
   - NOT include blanks (spaces, tabs);
   - NOT include non-alphanumeric characters, except of the two listed above.

4.2 Requirements for preparing a single-file model instance

A single-file specification can be used for small models. In addition to the requirements specified above only two obvious **requirements** should be met:

1. The model specs should be defined in one file (i.e., no `$INCLUDE` statements can be used for integrating model instance entities).
2. The file name with the model instance specs should have extension `gms` (e.g., nexus4.gms). The root name should be no longer than 12 characters, shall have a letter as first character, and must not contains spaces.

For single-file model specifications MCMA derives names of outcome variables through parsing the specification and selecting names of non-indexed variables that conforms to the naming convention described in Sec. 3.2.1.

4.3 Requirements for preparing a zipped-archive

Models having complex specification and/or large data sets are often specified in GAMS through several files, optionally organized in directory/folder structure. In order to rationally support diverse structures of such files we have defined simple requirements for the corresponding zip-archive. In addition to the requirements presented above, the following **requirements** should be met:

1. The files can be organized into a directory (folder) structure.
2. Names of all files (including directories) shall conform to the file-name requirements described in Sec. 4.2.
3. All files shall be zipped into one file. Zip-archives nested into the uploaded archive will not be processed by MCMA. The zip-archive should have the file name extension `zip` (e.g., an uploaded archive named as nexus7.zip). The root of the archive name should be no longer than 12 characters, and conform to file-naming convention (see Sec. 4.1).
4. Two files located in the root directory of the uploaded archive have the prescribed names and the required content:
   - `master.gms` containing the master (main) part of the model specification. This file will be become the argument of the GAMS solver call. `Message`_*ix* typically names the master file either `MESSAGE_master.gms` or `MESSAGE-MCA_master.gms`. Therefore, renaming such a file to `master.gms` fulfils this requirement.
   - `master.cfg` with configuration data for the MCMA analyses (see Sec. 4.5 for details).
5. Optional directory named `output` is considered as a place-holder for full solution of optimization run of each MCMA iteration. Such directory, if provided in zip-archive will be ignored. In any case, for each iteration an `output` directory will be created in the corresponding working directory, and made available for optional[12] location of the full

[12]If defined in the provided problem specification.

solution (specified in the post-processing part of the GAMS specs) of the corresponding optimization run.

6. The uploaded zip-archive should contain only files needed for the core model specification. Consider that all files included in the uploaded archived are used in every (typically several hundreds of) MCMA iteration. Therefore, including into the archive redundant files (especially results of analysis, logs, tests, or diverse versions of specification, etc) causes substantial overhead.

4.4 Comments on using the MESSAGE*ix*-based models

This Section was written by Volker in 2017. We need to check if:

- *it is consistent with the current* `Message_ix` *version;*
- *we indeed need different master.gms for single-criterion and MCMA; maybe conditional compilation proposed above would be a better solution?*

When applying the interactive MCMA tool with MESSAGE*ix*-based models the requirements for models consisting of multiple files that are uploaded as a zip archive hold (see Section 4.3). In order to prepare the MESSAGE*ix* code for use with MCMA tool, basically three modifications to the standard code need to be made:

- MESSAGE-MCA_master.gms (renamed to master.gms, see below) should be used as the entry point for the model code (instead of MESSAGE_master.gms),
- in MESSAGE-MCA_master.gms the GDX data file for the MCMA variable needs to be specified via the command `$SETGLOBAL`data, and
- a file called mcma_variables.gms in which the outcome variables[13] are defined needs to be created.

When calling MESSAGE*ix* via MESSAGE-MCA_master.gms, the file MESSAGE-MCA_run.gms is used to execute the various blocks of GAMS code that are part of MESSAGE*ix*. In the file MESSAGE-MCA_run.gms, after the definition of the core model, the file mcma_variables.gms is included. As mentioned above, when creating the zip-archive for upload to the MCMA, the file MESSAGE-MCA_master.gms needs to be renamed to master.gms. The mcma_variables.gms includes customized GAMS code that in a set of equations defines outcome variables, i.e., the variables out of which criteria will be selected interactively in MCMA. As described in Section 3.2.1 these outcome variables should conform to some requirements, particularly their length is limited to eight characters and they should not be indexed.

Further, the file mcma_variables.gms should be self-contained in the sense that the file includes the definition of variable names, as well as the declaration and definition of the equations which are used in the assignment of the variables.

4.4.1 Files to be included in the zip archive

In principle the content of the `model` folder of the `message_ix` git repository can be included in the zip archive. While considering which files include in the zip-archive note that MCMA consists of many iterations. For each iteration a working directory is created and a corresponding optimization is run in this directory. For efficiency, the zip-archive is not copied there; instead symbolic links are created from the working directory to each file/directory of the zip-archive. Therefore all files needed for the model instance definition need to be in the archive. However, it should be noted that possibly not all files and subfolders in the `message_ix`-git folder are needed. For instance, at present the `MACRO` sub-folder under `model` is generally not needed and can be removed from the zip-archive if not needed for analysis of results of each iteration.

In order to assure correct and efficient runs the following guidelines should be observed:

[13]See Sec. 3.2.1 for requirements that names of such variables should conform to, and Sec. 3.2.3 for description of outcome variables.

1. The GDX data file needs to be included in the `data` sub-folder of `model` for the model to successfully run.
2. Log files from possible runs of `master.gms` (typically named `master.l??`) should not be included in the zip-archive.[14]
3. MESSAGE*ix* by default stores its results in GDX format in the `output` sub-folder. Including this sub-folder in zip-archive is optional. However, if it is included then it will be removed by the MCMA. In any case, an empty `output` sub-folder is created in working directory created for each MCMA iteration, and the results of the corresponding optimization can be stored there.
4. Other than log-files redundant (i.e., not needed for specification of the model instance) files can be included in zip-archive (and will be linked to each working-directory). However, including many such files is likely undesired.

4.5 Content of the `master.cfg` file

Two types of data (namely, the list of outcome variables names and the approximate dimension of the single-criterion optimization problem) should be provided. This is done by a small, free-format text[15] file named `master.cfg` and is composed of the corresponding two parts separated by an empty line:

1. List of names of outcome variables, i.e., model variables that shall be available for interactive selection of criteria (for each MCMA instance). Each name should be defined in a separate line. Optional, but recommended, content (after a space separating it from the name) is considered as comment, i.e., is not be processed MCMA but it is useful for a quick reference of each variable meaning, especially if the names are not self-explanatory.
2. Approximate dimensions[16] of the single-criterion optimization model, specified by two pairs (each in separate line) of an integer number and a keyword denoting either `rows` or `cols`, respectively. The numbers correspond to thousands of rows/columns. E.g., the following two lines define the dimension of a model composed of less than 1000 rows, and $1000 \leq \text{cols} < 2000$:

```
0 rows
1 cols
```

Lines with the first character # followed by a space are considered as comments that offer optional ad-hoc documentation.

Annotated `master.cfg` sample is included in the pilot example outlined in Sec. B.2 and in Appendix A.3.

We should discuss whether such a file can be easily generated by `Message_ix` *or should be prepared "manually".*

4.6 Summary of the requirements for GAMS model specs

Maybe would be useful here?

A Examples of files

A.1 Example of specification of outcome variables

to be added here.

[14] In the next revision of MCMA the `master.l??` files will also (like the `output` now) be removed.
[15] I.e., spaces serve as word separators, initial and trailing spaces are ignored.
[16] This information is used by the MCMA task manager for queueing the optimization tasks.

A.2 Sample of _mc.gms file

The content of `_mc.gms` file copied below was generated by MCMA solver for the simplest case, i.e., selfish optimization of one (CUMCOST) criterion. In such case the MC-submodel is composed of only one variable and one constraint. Typical MC-submodels are composed of several dozens of variables and constraints. Note that declaration of the CUMCOST variable is commented because all outcome variables should be declared in the core model.

The `_mc.gms` files are composed of three parts:

- specification of the MC-submodel
- specification of the optimization task (which includes the core model)
- specification of post-optimization processing of the MC-submodel (post-optimization processing of the core model, if desired, is specified in the core model). Note that in this part also value of the other criterion (CUMINV) is reported although this criterion is not included in the SF for the selfish optimization of the CUMCOST criterion.

```
**** MC-submodel specs *************************************************
*    Model variables (defined in model.gms) are commented.

Variables
*        CUMCOST
*   Auxiliary MC variable mC_gf_c (defined by the modified GF) to be max.:
        mC_gf_c ;

Equations
        mC_gf_r ;

*   Modified GF: now defines aux. var. mC_gf_c (to be max.)
mC_gf_r  ..  +1.00000e+00 * CUMCOST   =e= -1.00000e+00 * mC_gf_c ;

**** Optimization task specs ********************************************

Model LP_MC_PART /all/ ;
Solve LP_MC_PART using lp maximizing mC_gf_c ;

* End of the MC-part of LP ****************

**** MC-submodel post-processing ****************************************

* Put MC-part solution
FILE mc_out / "/p/ime/smt_work/mcma_tst/files/00545/1244/2468/_mc.sol" / ;
mc_out.nd = 5;
mc_out.nr = 2;
mc_out.nw = 12;
mc_out.tw = 9;
put mc_out;

put 'mC_gf_c'; put mC_gf_c.l / ;
put 'CUMCOST'; put CUMCOST.l / ;
put 'CUMINV'; put CUMINV.l / ;

putclose mc_out;
```

A.3 Example of the `master.cfg` file

To be added here.

B Shared space for ENE Program applications of MCMA

Working space for the ENE program applications of MCMA has been created at the `/p/ene` file system. To ease testing and bug fixing it is recommended (but not necessary) to copy there model instances prepared for MCMA. We will soon provide there a link to `wdir` directories which shall ease discussions on the problems and access to full solutions of the underlying optimization tasks.

B.1 Directory location and structure

The directory `/p/ene/mca` has been created on the unix file system as the ENE collaboration space for the MCMA applications. This directory is mounted on PCs as `P:\ene.mca`.[17]

The directory has currently the following subdirectories with the corresponding suggested use:

- *examples* - for generic examples of using MCMA and MCAA.
- *models* - for model instances; each model should be placed in a separate sub-directory. Please use unix-style dir/file naming convention (short names starting with a letter and composed of only letters and digits (if really desired than an underscore and a dot can be used). Currently[18] there are two directories with models.
- *tests* - for exploring/storing diverse tests. Currently tests of Cplex numerical problems are stored there.

B.2 Pilot example

The pilot example for exploring the MCMA interface to model instances generated from the MESSAGE-IX was prepared by Volker on Apr. 5th 2017 as the zip-archive. Based on this example the test case was prepared by:

- renaming file `MESSAGE-MCA_master.gms` to `master.gms`
- adding the configuration file `master.cfg`).
- creating zip-achieve `volker_apr.zip` containing only the needed files.

The created zip-archive was stored in `/p/ene/mca/examples/gams17` and was used for testing the MCMA-GAMS interface described in this note. Note that the zip-archive contains the annotated configuration file `master.cfg`), which can easily be adapted to other model instances.

C Functionality and architecture of the MCMA tool

Providing the required MCMA functionality for supporting multiple-criteria model analysis demands integration of many components developed for various needs and by diverse developers. The implemented infrastructure is therefore complex and has hierarchical modular structure. However, the users typically prefer to neither explore software architecture nor be involved in software configuration and maintenance. In order to meet these typical preferences, MCMA computational infrastructure is transparent for the users, who access the needed functionality through an easy User Interface (UI) provided through commonly used web browsers. Some readers however, might be curious about the MCMA computational infrastructure; therefore we start the MCMA tool description with the overview of main MCMA components, and follow (in Section C.3 with the description of MCMA functions directly controlled by the users.

C.1 Stages of the MCMA

The MCMA structures the analysis into three stages (each accessible through the corresponding button of the top MCMA menu, cf Section C.3):

[17] The PC mount has been so far arranged for only few ENE colleagues. If you want to have this directory mounted on your PC account, then please create the MIS ticket and make Pat Wagner its co-owner.

[18] MM: update this (it was current in 2017).

1. Problem. In this stage a core model instance is processed into the form suitable for MCMA. The model instance defines a set of feasible solutions as well as outcome variables out of which subsets are interactively selected to serve as criteria. At this stage the user uploads a file with the model instance specification. The uploaded file is processed in the background; if no processing errors occur then the problem (i.e., the model instance) becomes available for MCA.
2. Instance. For a given problem several MCA instances can be defined. An instance is defined by interactively selected (from the list of the model outcome variables) criteria; For each criterion the user interactively defines its type (minimize or maximize) and optionally the criterion name (by default the selected variable name is used as the criterion name). After an instance is defined, the corresponding utopia and approximation of nadir values are automatically computed. Next, the initial analysis is automatically generated. For large models this stage can take several hours (or even days). After completion of this stage, the user can start interactive analyses.
3. Analysis. Several analyses can be generated for each instance (the first one is generated automatically; the user can easily generate more analyses through a simple form). Each analysis is composed of iterations. Each iteration is defined by the user preferences specified interactively in terms of Aspiration/Reservation (A/R) values. Several iterations (for each analysis) are generated automatically in order to provide an initial view on the criteria trade-offs. For each iteration (after the corresponding preferences are defined) the underlying parameterized optimization task (OT) is generated. The OT consists of the problem core model merged with the LP submodel corresponding to the multiple-criteria optimization defined for the A/R values specified by the user. Each OT is run in a separate `wdir` (working directory) where the optimization process logs and full solution are stored. The criteria values are available through the MCMA; values of all model variables are available in the `wdir`, if the GAMS-format model specification includes generation of solution (and possibly other desired information).

C.2 Architecture of MCMA computational infrastructure

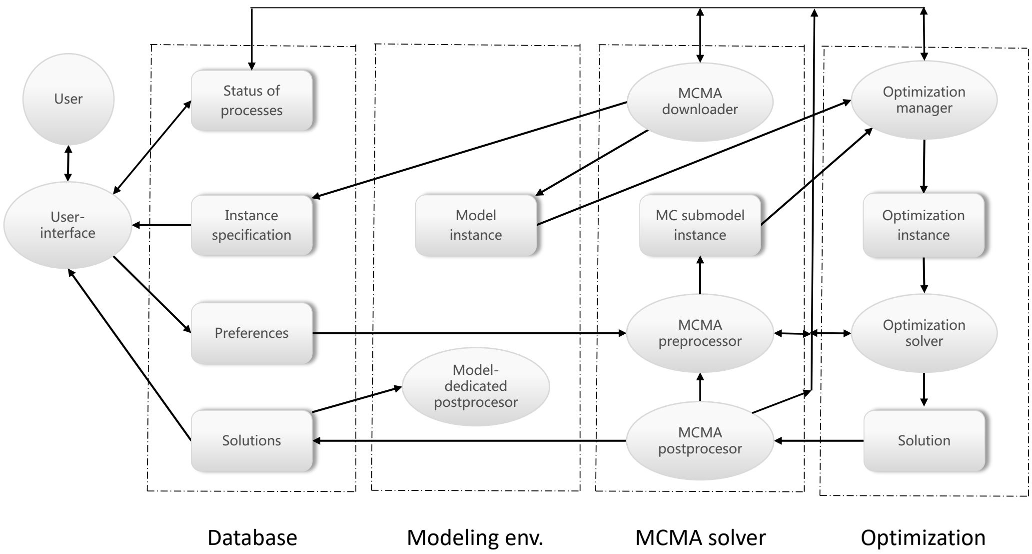

Figure 2: Modular infrastructure for multiple-criteria model analysis.

The main hindrance in wide applications of the MCMA methodology is the amount of work and expertise needed for such applications; the architecture described here illustrates well this issue. Traditional methods of model analysis are relatively easy to use with standard optimization tools, especially if integrated with the modeling environment used for the model development. The environment actually implemented and described in this paper removes this hindrance by making the multiple-criteria model analysis even easier to apply than to use traditional optimization-based environments for iterative model analysis that requires advanced modeling skills.

To enable use of MCMA with models developed in diverse modeling environments, the MCMA architecture shown in Figure 2 consists of four modular and interlinked components that are linked (in a way transparent for the MCMA users) with modeling environments used for development and maintenance of the model instances uploaded to MCMA for analysis:

- The user-interface.
- Database handling all data needed for all MCMA processes.
- MCMA solver.
- Optimization module.

The fifth top-component shown in Figure 2, labeled *Modeling env.*, does not belong to the MCMA infrastructure; however, we show it because the user needs a modeling tool for the model development.

The short characteristics of the top-level components are as follows:

UI: The user communicates with the MCMA exclusively through the User Interface (UI) application, implemented in Java, installed at a Tomcat servlet container, thus providing users with the MCMA interface through Web-browsers. The UI is presented in Section C.3.

DB: Dedicated data-base, manages all persistent data of MCMA. The DB is implemented as an instance of an RDBMS (relational database management system). The schema of the MCMA DB is far too complex to be even outlined in this paper. We only list below examples of data to illustrate the data scope:

- Users and user groups with privileges of members.
- Configuration of the MCMA components; e.g., of solvers to specify functionality options available for diverse users and applications.
- Status of all processes run by the MCMA components. This data provides a back-bone for organizing the MCMA workflows.
- Specifications of uploaded model instances.
- Parameters of the preferences defined by the user.
- The MC-submodel solutions of each analysis iteration.

We also point out that handling MCMA component configuration data through a RDBMS greatly improves robustness and maintenance of such rather complex systems.

Modeling environment: the two-way linkage between the model development environment and MCMA is composed of:

- Model instance conforming to the requirements summarized in Section 4 is interactively uploaded to MCMA.
- Solution of each MCMA iteration is provided to the user for optional, model-dedicated postprocessing of analysis results.

Three formats of model instances are supported:

- the standard MPS-format;[19]
- the GAMS-format[20] provided as a single *.gms file or as a structured collection of GAMS
- a structured collection of GAMS-based model specification and data files generated from the `Message_`*ix* platform [5].

Solutions are provided in formats corresponding to the model instance, i.e., either a standard MPS-format output file or the format defined in GAMS specification for the output.

MCMA-solver: Dedicated solver, written in C++, transparent for the MCMA users. It: (1) processes the uploaded model instance, (2) generates the MC-submodel instances for user-defined preferences, (3) prepares data and working space for optimization solver, (4) queues the optimization tasks, and (5) postprocesses optimization results for making them available to the user through the UI.

Optimization: The dedicated task manager handles optimization jobs generated and queued by MCMA-

[19]MPS (from: Mathematical Programming System) widely used file format for specification of linear- and mixed-integer-programming problems.

[20]General Algebraic Modeling System (GAMS), see e.g., [2].

solver, i.e., allocates each of them to one of optimization solvers distributed over the workstation network. MCMA uses the same solvers as the single criterion optimization of the corresponding core models. Before the selected solver is executed, a dedicated application merges the MC-submodel and core-model instances.

The workflows between elements of the MCMA infrastructure are actually hidden from the MCMA users, who control the flows only through the UI described in Section C.3. Therefore, we only briefly summarize the basic workflows and actions triggered by specification of preferences for each iteration:

1. Preference parameters are stored in the DB and the iteration status is updated in the DB; then the MCMA-solver is called, and the user may either wait for the solution, or switch to another iteration.
2. The MCMA-solver reads the iteration data from the DB, generates the MC-submodel, stores it on the server file-system, and updates the iteration status in the DB, which queues the corresponding optimization task.
3. The optimization manager allocates the task execution. The manager is actually a daemon-type application, i.e., it runs in the background, frequently checks the queued tasks, and allocates them on available servers and solvers whenever resources allow.
4. MCMA solvers are of three types:
   - Preprocessor: it merges the MC-LP with the model instance representation and generates input files for the selected optimizer, executes the suitable optimizer, waits until optimization finishes, and then calls the postprocessor.
   - Optimizers: solvers of the optimization problems.
   - Postprocessor: extracts from the optimization results solution of the MC-LP part, stores it in the DB, and calls back the MCMA-solver. The full solution remains available for the user.
5. MCMA-solver processes the solution, in particular prepares data for generation of the chart presenter to the user.

Each MCMA application updates, at the beginning and at the end of execution, the task status in the DB. Thus, each application can check the status and provide the user with the corresponding information, e.g., about execution stage for yet unfinished jobs, or charts and values for finished iterations.

C.3 User interface

Following the Structured Modeling (SM) paradigm and the corresponding model instance analysis cycle, the MCMA processes are structured to help the users in effective and efficient analysis through a simple, error-tolerant interface. The User Interface (UI) summarized here provides users with flexible control of the all analysis elements, and thus enables organization of the process according to diverse and changing users' needs. The interface is available through Web-browsers, thus allows for the anytime-anywhere access; in particular, the analysis can be paused anytime; the defined tasks are anyway processed in background by servers, results stored and made available whenever the user decides to continue analysis.

C.3.1 User control of the top-level functionality

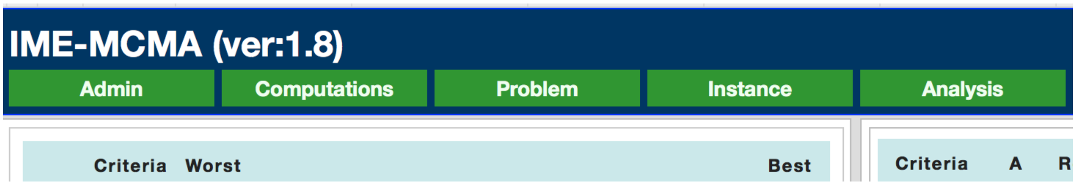

Figure 3: User-interface to control the MCMA workflows.

The main five top-level functions of the UI are accessed through the green buttons shown in Figure 3; each of them opens access to the underlying structure of dialogues. Here, we only outline the provided functionality:

Admin: Each MCMA user has private space for handling model instances and results of analysis. However, a group of users may share their models and results. Therefore, MCMA supports administration of user groups, and privileges of group members.

Computations: MCMA provides information on the status of optimization tasks that have been generated but not yet finished, see Section C.2 for the summary of the corresponding components.

Problem: The corresponding set of dialogues supports uploading of model instances provided in one of the formats discussed above. The uploaded model instance is considered as the MCA problem.

Instance: For each problem the user may define several analysis instances. Each MCA instance is specified by the interactively defined criteria. The definition of each criterion is composed of the following selected:

- Corresponding outcome variables. The list of such variables is extracted from the uploaded model instance; filters for names of variables support selection for large models.
- Criterion type (either minimized or maximized).
- Criterion name. This is optional because the criterion name is initialized to be the same as the name of the corresponding outcome variable.

Analysis: The user may define for each MCMA instance several analyses. Each analysis is composed of iterations outlined above and discussed detail below. Defining several analyses is especially useful for extensive MCMA in which each analysis is composed of many iterations; moreover, separate analyses can have different focus and/or be done by different users.

Contact: Reporting problems and questions is supported by the corresponding white button (at the top right corner of the blue control panel shown in Figure 5). The user comments are stored in the DB together with automatically assembled information about the situation in which the contact is used. Thus, the developers can easily recreate the situation for exploring and handling the reported problem.

The interactive specification of preferences is the main MCMA activity. Therefore we comment below on this part in more detail than on the other MCMA elements.

C.3.2 Preparatory computations

Before providing the user with access to the interactive analysis, MCMA performs several background tasks in order to provide the user with initial information on the Pareto set. These tasks are automatically generated and run after the user defines a new instance or a new analysis (the initial analysis is also generated automatically). We briefly summarize these background computations.

For each new instance the utopia and nadir values are computed. This requires $4 \star N$ automatically generated optimization tasks, where N is the criteria number. First N tasks compute the utopia values, for each criterion by the selfish optimization of the corresponding outcome variable. Next $3 \star N$ tasks sequentially improve approximation of the nadir values. After completing these tasks, the initial analysis is automatically created.

For each new analysis $N+1$ iterations are automatically generated to provide the user with initial set of solutions. This set consists of N iterations of selfish optimizations (i.e., for each only one criterion is active), and one iteration with the so-called compromise preferences, i.e., $\mathbf{q}^a$ and $\mathbf{q}^r$ set for each criterion at equal (in terms of the fractions of utopia and nadir range) values. Thus the user starts specification of his/her preferences with knowledge about the extreme (selfish-criterion) and compromise preferences.

C.3.3 Interactive specification of preferences

After the automatically generated iterations are completed, the user takes full control of further iterations. The main interaction screen is composed of the two panels shown in Figures 4 and 5, which illustrate the left and right parts of the screen, respectively: (1) the left-side chart shows the distribution of criteria values, and (2) the right-side control panel supports specification of preferences for next iteration, as well the selection of the displayed iteration.

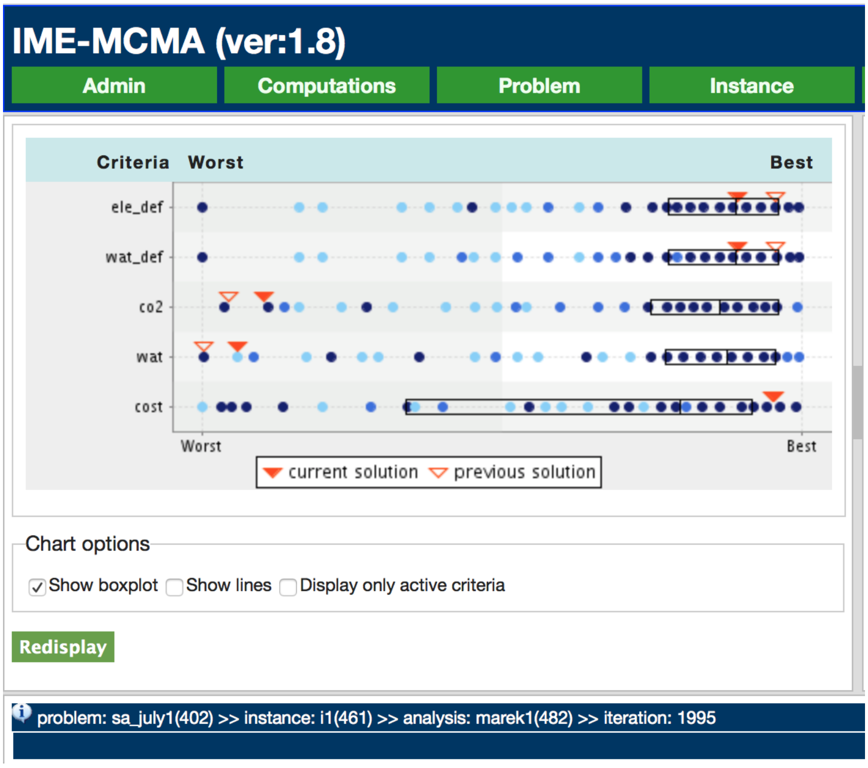

Figure 4: Distribution of the criteria values.

The user can select, using the choice list shown at the bottom of Figure 5, any iteration as the basis for further analysis. For the selected iteration the user, typically, first explores the Pareto-solutions obtained in the previous iterations of the current analysis, and then specifies preferences for the next iteration.

The criteria values of the previously obtained (within the same analysis) solutions are presented in the chart composed of normalized parallel coordinates shown in Figure 4. Each dot corresponds to a solution; dots in darker colors represent aggregates of several solutions having similar criterion values. The details of the underlying solution(s) is displayed (as a hint, not shown in the Figure) on demand, when the user points to a selected solution. The boxes cover the two middle quartiles of solutions; the best and worst 25% of solutions are displayed on the right- and left-sides of the corresponding box, respectively.

The red triangles shown in the chart point to the criteria values of the current (solid) and previous (empty) solutions, respectively. Thus the user easily sees which (and by how much) some criteria were improved or compromised (compared to the previous solution) by the recently specified preferences.

The bottom blue panel provides information about the model and analysis instances, as well as the current iteration of the analysis. The little light-blue icon shown at the left corner (marked by the letter *i*) enables access to the full information on the corresponding optimization run, and the solution provided by the solver.

Preferences for a new desired trade-off between criteria values are then expressed through aspiration and reservation values for each criterion, respectively. While defining new preferences one should consider that the basis solution is Pareto-efficient, i.e., an improvement of one criterion (or more criteria) is possible only, if at least one other criterion will worsen. In other words, the user shall decide which criteria he/she wants to improve and which to compromised to make the desired improvement possible. An improvement of a criterion performance can be triggered by setting a more ambitious (closer to the corresponding utopia value) reservation value for this criterion, optionally augmented by also higher aspiration. Also optionally, one can select a criterion (or criteria) to compromise; this can be done by relaxing (i.e., worsening) the corresponding reservation value(s). In such a way the user preferences are defined for each iteration.

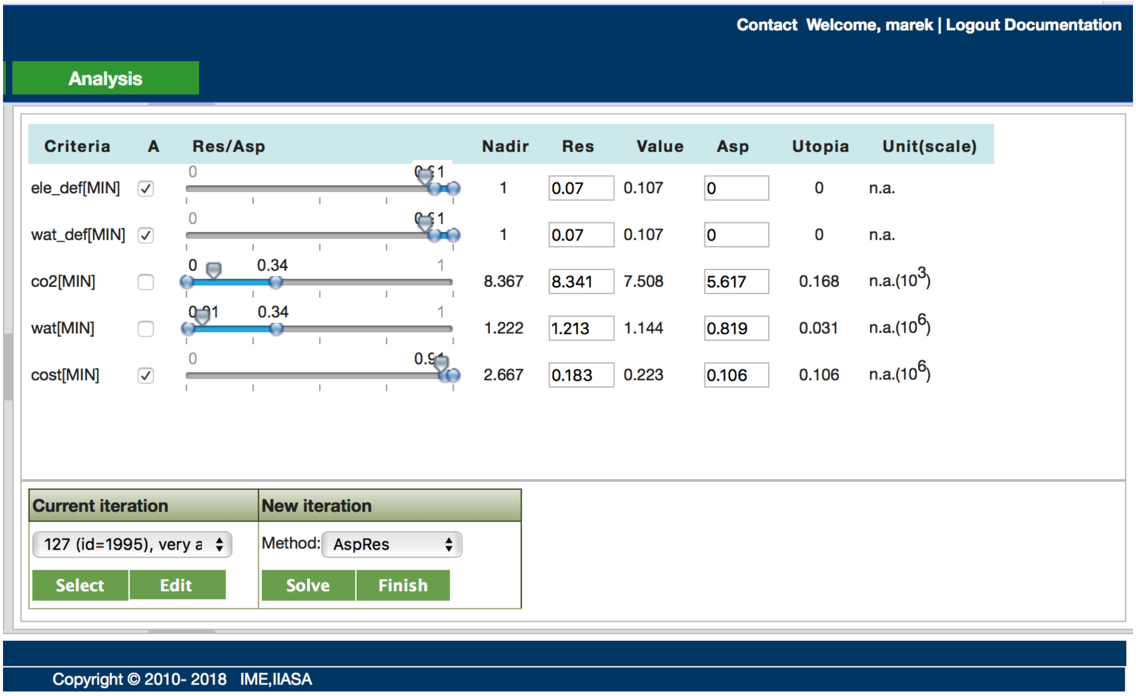

Figure 5: Specification of preferences in terms of criteria reservation and aspiration values.

For any given preferences, the multi-criteria problem is represented by an auxiliary parametric single-objective optimization problem defined through the achievement scalarizing function (6); solution of the corresponding optimization problem provides a Pareto-solution best fitting the user preferences.

Typically, the MCMA users explore various areas of the Pareto frontier (e.g., cheap and expensive having the corresponding bad and good values of environmental criteria) before deciding which compromises between the criteria values fit best their preferences. Examples of such exploration and methodological background on the Pareto set analysis is available e.g., in [6, 11, 7, 9, 1, 12, 10, 4].

References

[1] J. Antoine, G. Fischer, and M. Makowski. Multiple criteria land use analysis. *Applied Mathematics and Computation*, 83(2–3):195–215, 1997. available also as IIASA's RR-98-05.

[2] A. Brooke, D. Kendrick, and A. Meeraus. GAMS*, A User's Guide, release 2.25*. The Scientific Press, Redwood City, 1992.

[3] J. Granat and M. Makowski. ISAAP – Interactive Specification and Analysis of Aspiration-Based Preferences. Interim Report IR-98-052, International Institute for Applied Systems Analysis, Laxenburg, Austria, 1998. Available on-line from `http://www.iiasa.ac.at/~marek/pubs`.

[4] J. Granat and M. Makowski. Interactive Specification and Analysis of Aspiration-Based Preferences. *European J. Oper. Res.*, 122(2):469–485, 2000. available also as IIASA's RR-00-09.

[5] D. Huppmann, M. Gidden, O. Fricko, P. Kolp, C. Orthofer, M. Pimmer, N. Kushin, and A. Vinca. The messageix integrated assessment model and the ix modeling platform (ixmp). *Environmental Modelling & Software*, 112:143–156, 2019. DOI:10.1016/j.envsoft.2018.11.012.

[6] M. Lehtveer, M. Makowski, D. McCollum, and M. Strubegger. Multi-criteria analysis of nuclear power in the global energy system: Assessing trade-offs between simultaneously attainable economic, environmental and social goals. *Energy Strategy Reviews*, 8:45–55, 2015. DOI 10.1016/j.esr.2015.09.004.

[7] M. Makowski. Modeling paradigms applied to the analysis of European air quality. *European J. Oper. Res.*, 122(2):219–241, 2000. available also as IIASA's RR-00-06.

[8] M. Makowski. Management of attainable tradeoffs between conflicting goals. *Journal of Computers*, 4(10):1033–1042, 2009. ISSN 1796-203X.

[9] M. Makowski and L. Somlyódy. River basin water quality management. *Mathematical Modelling Theory and Applications*, 9:311–332, 2000.

[10] M. Makowski and A. Wierzbicki. Modeling knowledge: Model-based decision support and soft computations. In X. Yu and J. Kacprzyk, editors, *Applied Decision Support with Soft Computing*, volume 124 of *Series: Studies in Fuzziness and Soft Computing*, pages 3–60. Springer-Verlag, Berlin, New York, 2003. ISBN 3-540-02491-3, draft version available from `http://www.iiasa.ac.at/~marek/pubs/prepub.html`.

[11] S. Parkinson, M. Makowski, V. Krey, K. Sedraoui, A. Almasoud, N. Djilali, and K. Riahi. A multi-criteria model analysis framework for assessing integrated water-energy system transformation pathways. *Applied Energy*, 210(15 January):477–486, 2018. http://dx.doi.org/10.1016/j.apenergy.2016.12.142.

[12] A. Wierzbicki, M. Makowski, and J. Wessels, editors. *Model-Based Decision Support Methodology with Environmental Applications*. Series: Mathematical Modeling and Applications. Kluwer Academic Publishers, Dordrecht, 2000. ISBN 0-7923-6327-2.