

User Guide for the optimization tool

SeCoMOLA

Status:

Version 1.1

Copyright © 2016 Helmholtz Centre for Environmental Research – UFZ

Contact person:

Carola Pätzold (carola.paetzold@ufz.de)

Last edited

Date	Edited by	Version	Last edits
10.12.2015 – 24.02.2016	CaP	1.1	Document created and edited

Table of contents

1	Purpose of this document.....	5
2	Modular design and their interactions.....	7
2.1	Generation of the patch ID map and the start individual	8
2.1.1	Based on an ASCII map with the current land uses.....	8
2.1.2	Based on a HRU list.....	10
2.2	Additional configuration option: Generation off the first population inclusive individuals as representatives for extreme land covers	10
2.3	Additional variation operators	11
2.3.1	filter_mutation.....	11
2.3.2	logical_mutation	11
2.4	Additional selection function: constraint_tournament_selection	12
3	Instruction manual	13
3.1	General information and folder structure	13
3.2	How I could start an optimization run?.....	14
3.3	Initialization file config.ini.....	16
3.4	Input files.....	21
3.4.1	HRU information in a csv file	21
3.4.2	The first map as ASCII file.....	21
3.4.3	Text file with transformation information	22
3.4.4	The information of the cover ranges	23
3.4.5	Information of the worst fitness values	23
3.5	Start files for the model server MSG and model1	24
3.6	Specification for the model folders.....	25
3.6.1	Update of the model helping folders	25
3.7	Log files.....	26
3.8	Notes for the usage of the tool on the model server MSG, MSG-HPC, Eve Linux cluster or as external user.....	28
4	Potential errors and instructions for troubleshooting	29

User Guide for the optimization tool SeCoMOLA

5	Differentiation to the following version	29
6	Appendix	30
6.1	Links for the application.....	30
6.2	Links for the inspyred and python user guide	30
6.3	Link for the NSGAII paper	30
6.4	Link for the Eve cluster wiki	30
6.5	Citation and credits.....	31

List of figures

Figure 1: Flowchart of the optimization tool SeCoMOLA.....	5
Figure 2: Visualisation of the modular design	7
Figure 3: Illustration from map abstraction to start individual	9
Figure 4: Start individual based on an own patch ID map	10
Figure 5: Model specific settings in the initialization file config.ini	17
Figure 6: Configuration settings of the optimization algorithm in the initialization file config.ini	19
Figure 7: Information of the first map and for the map generation during the optimization process are set in the initialization file	20
Figure 8: Example for a HRU csv file.....	21
Figure 9: Small example for a land use map in ASCII format and with coded land use categories.....	22
Figure 10: Example of a matrix with transformation information (1 land use change is allowed, else 0)	23
Figure 11: Information of the land cover ranges per land use (in percent)	23
Figure 12: Text file with worst fitness values.....	24
Figure 13: Start file for the MSG server	24
Figure 14: Model structure of the SWAT model	26
Figure 15: Section of the output folder	27

1 Purpose of this document

Under the leadership of Prof. Dr. Ralf Seppelt, a group of technicians and researchers of the department Computational Landscape Ecology developed an optimization tool. The application optimize a given grid based map regarding certain target functions. The optimization is realised by a combination of landscape ecological models and optimization algorithms of the *inspyred* library. One application example could be the optimization of land use management of a defined region with regard to several ecosystem services. The optimization could be based on grid maps or lists of certain spatial units like Hydrological Response Units – HRUs, which are used as input for the integrated models. (ToDo: link to publication if it is published)

inspyred is an free software python library (GNU General Public License version 3.0) with a broad range of algorithms including evolutionary computation, swarm intelligence and neural networks. Furthermore, the library provides relatively simple usage of standardized bio-inspired algorithms for users who needn't many individual modifications.

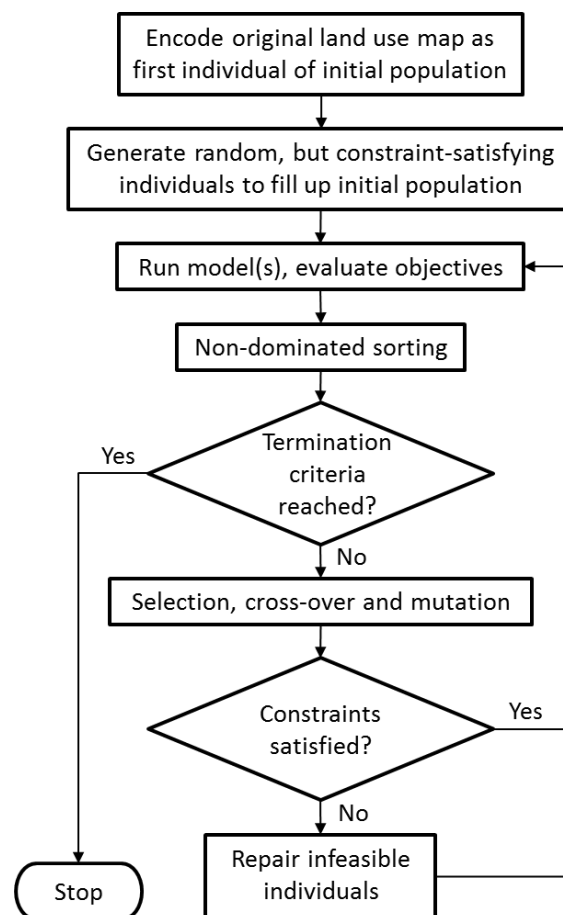


Figure 1: Flowchart of the optimization tool SeCoMOLA

User Guide for the optimization tool SeCoMOLA

The Figure 1 presents the general flowchart of the optimization application SeCoMOLA (**Se**quential **C**onstraint **M**ulti-objective **O**ptimization of **L**and **A**llocation). The modular design and the functionality of the application are described in chapter 2. Who wants to start directly with the usage of the optimization tools finds information about the configuration of the tool or integration of input data and models in chapter 3.

Models can be currently implemented as R script or as a python file. The second option allows the implementation of any model which is separately runnable on the computer or server. In this case simulate the special Python start file a start command over a command line interpreter. You can also adjust the model outputs on the tool requirements over the individual start file. In this way the SWAT (Soil and Water Assessment Tool) model was implemented in the optimization tool.

The optimization tool is also distributed under the GNU General Public License version 3.0. Future developments by the department Computational Landscape Ecology of the UFZ will be commented in the program code and described in this document.

The optimization tool is platform-independent and tested on Windows 7, Windows 2012 R2, Windows Server 2008 and CentOS 6. So you can use the MSG, model1 or the Eve Linux cluster for your optimization runs.

This document is the user guide of the optimization tool.

2 Modular design and their interactions

The tool pass for one optimization run various stages as you can see in Figure 1. Some of these stages will be called repeatedly. The generation of the start individual based on the input data like a raster map or HRU list and the output of the optimization results take place only once. Between these both processes runs a cyclical repetition of several sub processes such as the generation of the next population, start the models or evaluate their results.

The following figure shows the interactions between the individual python files to make this complex procedure possible.

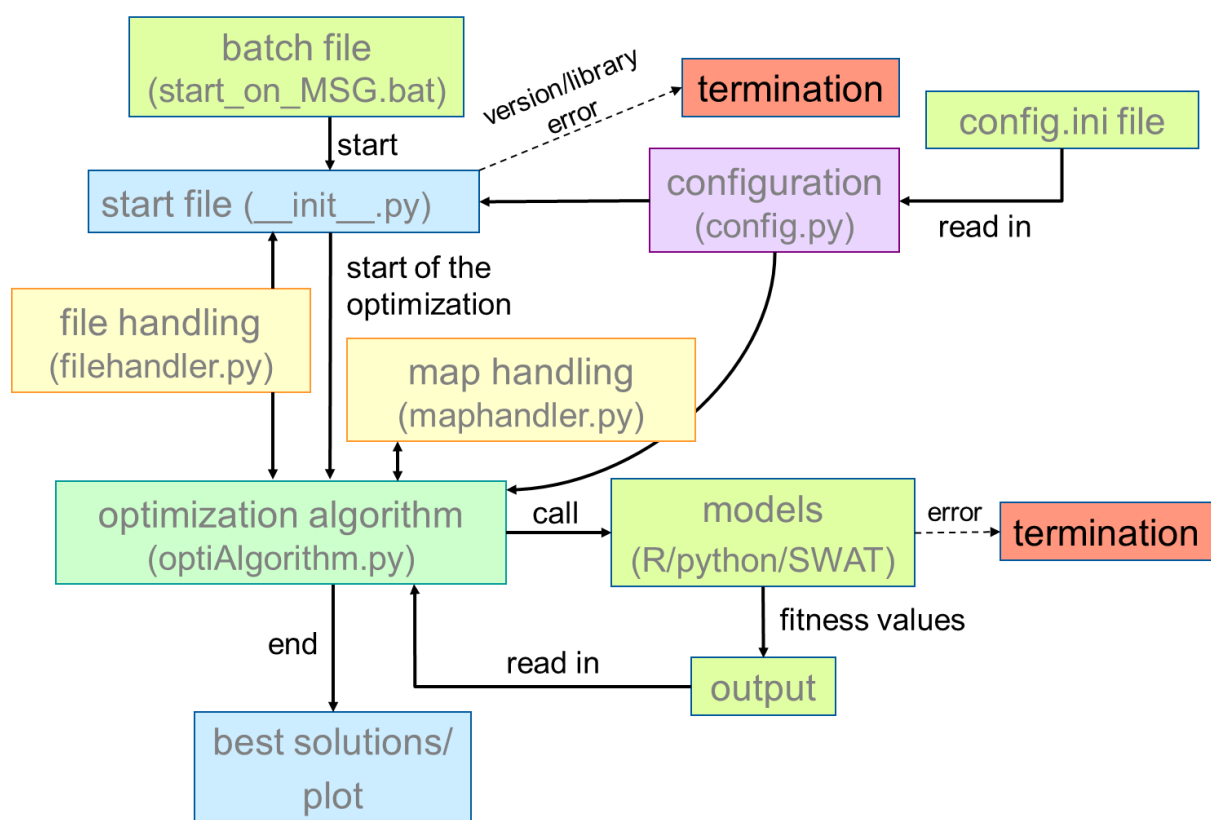


Figure 2: Visualisation of the modular design

The file **config.py** read the parameter settings of the initialisation file config.ini and set the values for the class parameter. The program use default values if necessary parameters are missing in the initialisation file.

The start file **__init__.py** contains the main function of the optimization tool. At the beginning of the function will be checked the python version with all the necessary libraries on their availability (**requirements.py**). If the check wasn't successful, the program terminate with a corresponding error message. Otherwise all log files are initialised, updated the helping folders and start the optimization algorithm. If the optimization process is finished, then the

User Guide for the optimization tool SeCoMOLA

results will be plotted. For the GA is the development of the fitness values during the optimization process plotted and for the NSGA-II the Pareto front of the best solutions.

With the start of the optimization algorithm is the file **optiAlgorithm.py** called. This function starts with the initialisation of the optimisation algorithm and the generation of the first population. For every individual are the models started as parallel calculation processes. For the parallelization is the original models folder copied in helping folders, so that for every individual exists one models folder. After that the individuals are handed over to the model subfolders and the models are started in parallel processes. If the calculations are finished, the collected fitness values will be returned to the optimisation algorithm and the model outputs are written in one file. If the termination criterion isn't fulfilled, then generates the optimization algorithm a new population based on the best individuals of all previous generations and the process starts again. If at least is one criterion fulfilled, then the best solutions with their fitness values are returned and plotted.

The **filehandler.py** is a helping file. It provides several functions for the other files for example the initialisation of the log files of the inspyred library and the optimization tool. Furthermore it contains functions for writing data in other files like log files or updating the files with the current individual or maps in the model folders, calling models and saving created plots as png files.

The **maphandler.py** provides functions which are related with map processing. This includes for example the both new variation operators `filter_mutation` and `logical_mutation`, the map abstraction from the original map to the start individual vector including checks if the constraints are matched and the reconversion from an individual vector to a map.

2.1 Generation of the patch ID map and the start individual

2.1.1 Based on an ASCII map with the current land uses

If you set an ASCII map with the current land uses and transformation information as input, then the program determine the static land use categories at first. Static land use categories are characterised in the transformation matrix with one in the diagonal element and zeros in row and column. If no transformation information exists, then there aren't static land use categories and every land use can change to any other one between 1 and `max_range`. Next step is the derivation of the patch ID map from the original map. This means, that all static and NODATA cells are marked with zeros for the exclusion of the optimization and all other cells are clustered in numbered patches. The patches are determined by scanning row by row and recursive search for related neighbouring cells. Per cell are four or eight

User Guide for the optimization tool SeCoMOLA

neighbouring cells scanned if they aren't checked before. The start individual vector is initialised with length equal to the maximum patch ID. The land use of patch i is written in the element with index $i-1$.

This process is illustrated in the following figure. There are urban and road cells excluded from the optimization and marked in the patch ID map with zero. The abstraction on patch level will find three patches and the cells of the patch at the top left will be indicated with one. The cells of the second patch in the right upper corner will be indicated with two and the cells of the green area with three. Therefore the start individual has three elements. The first two elements filled with the number of cropland and the last one is green area.

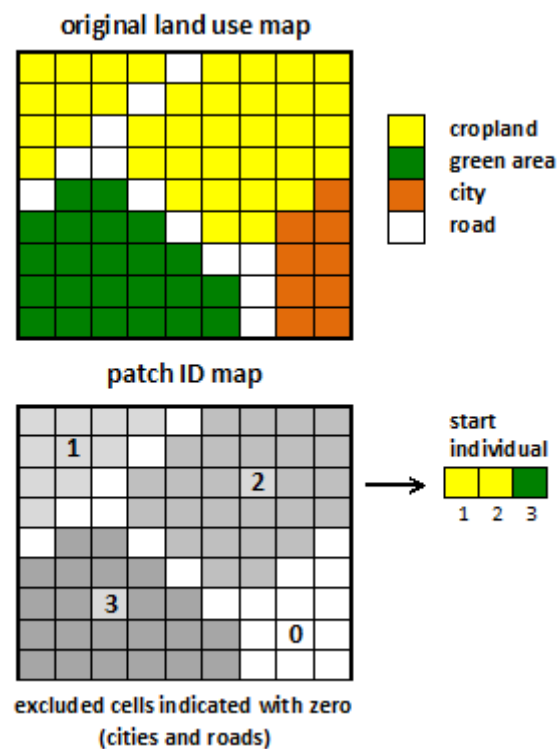


Figure 3: Illustration from map abstraction to start individual

You can use alternatively an own patch ID map in combination with an original map. It could be sensible, if two fields are independently farmed but they have randomly the same land use in the original map. With consideration of an own patch ID map can be omitted the abstraction of the original map into patches. In this case is the first individual generated based on the own patch ID map as shown in Figure 4.

User Guide for the optimization tool SeCoMOLA

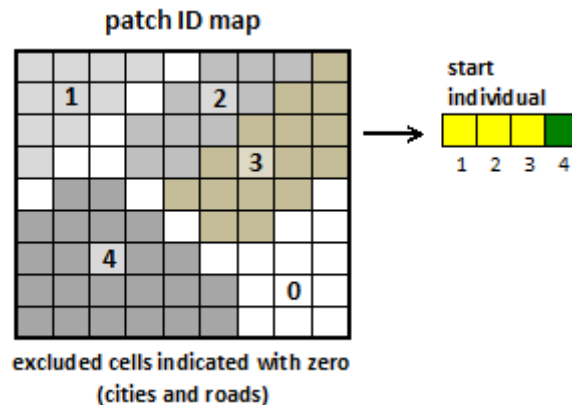


Figure 4: Start individual based on an own patch ID map

2.1.2 Based on a HRU list

If a HRU list is given as csv file with information of the current land uses or crop rotations per HRU, then is the length of the start individual vector equal to the number of HRUs. Each element of the vector gets the current crop rotation of one HRU, where the crop rotation of the n-th HRU is on the vector element with index n-1. So results from Figure 8 the first individual (2,5,6,7,8,9,10,1,1,3,5,6,7) with the length 13. This start individual is transferred to the optimization algorithm. The change of the HRU land uses can take account of constraints given by transformation information or land cover ranges. You can give a patch-ID map or an original map in addition to the HRU list, so you can use combinations of HRU and map based models.

2.2 Additional configuration option: Generation off the first population inclusive individuals as representatives for extreme land covers

A new development of the SeCoMOLA application is the option for a special generation of the first population with individuals as representatives for extreme land covers.

You can enable this option with the parameter *extreme_seeds* in the section for the settings of the optimization algorithm in the configuration file config.ini. Is this parameter “True”, then SeCoMOLA generates per land use category, which is not excluded from the optimization by the transformation information, one representative for minimal and one for maximal land cover. If the start individual is one of the searched representatives, then no other individual will be created for the extremum by this functionality. In this way generated individuals are feasible and meet all constraints. This means for example of an optimization without transformation information and land cover ranges, that per land use one individual is created without this land use category and one which is completely filled with it. If constraints exists,

than will be only representatives generated which maximize and minimize the land cover in consideration of all constraints.

The remaining individuals will be created by the standard generation of the first population.

2.3 Additional variation operators

2.3.1 filter_mutation

The new developed variation operator `filter_mutation` generates individuals based on the transferred candidate from the optimization algorithm which fulfil all constraints from transformation matrix and land cover ranges. Is the transferred candidate is feasible and not used before then is the transferred candidate unchanged returned as new individual. If even one of these conditions is violated, then will be one by one element replaced randomly with one land use category which is an element of the intersection from the possible land uses of the start individual and the transferred candidate. Finally, the new individual is tested if it fulfils the land cover ranges. If it is feasible, it is returned as new individual else the current one is rejected and the total process starts again. The vector elements are filled totally random between one and `max_range` if no constraints exist.

For example the start individual is (2,5,7,1,1,2) and the transformation matrix of Figure 10 is given and `max_range` is eight. The candidate (2,1,7,1,3,2) is transferred by the optimization algorithm. For the first, third, fourth and last element could be used land uses into account only of the transformation information because the land use of the transferred candidate is equal to the one of the start individual. For the other ones could be selected one of the land use categories from the intersection. So the process can use one number of {1,3,5,6,7,8} for the second position. The values six and eight are excluded because the one on the second position of the transferred candidate can only change in odd land use numbers. For the fifth element is one land use of {1,3,5,7} possible. A transformation matrix with ones in the diagonal elements (required for allowed transformation matrices) allows that in any case could be returned at least the start individual as new individual. In the worst case is the start individual equal with the returned one. Depending on the transformation information and the land cover ranges are more combination possibilities allowed.

2.3.2 logical_mutation

The new developed variation operator `logical_mutation` is the principle of the `filter_mutation` function intensified, so that the generated individuals are similar as possible to the given one and repetitions of individuals are not allowed. Both rejected and used individuals are saved in a memory and all elements of the memory are excluded for future individual creations. For

User Guide for the optimization tool SeCoMOLA

this reason, model runs twice with the same starting conditions and consequently same fitness values are avoided. This can reduce the total run time and counteract convergence into a local optimum. If the parameter priority is True, then will be selected preferentially the land use category of the transferred candidate and the new individual is similar as possible to this one. If the intersection of the possible land uses is empty, the algorithm select randomly one only based on the allowed changes from the land use element of the start individual. If no feasible individual could be created, then is a special termination of the optimization algorithm started. If one or more individuals are found then follows the fitness value calculation by the models and the best solutions of the optimization are returned.

The land use numbers per element are selected randomly between 1 and max_range if there are no constraints.

2.4 Additional selection function: constraint_tournament_selection

The implemented constraint_tournament_selection function based on the proposed approach for constraint handling by Deb et al. (2002, A fast and elitist multiobjective GA: NSGA-II). It is a selection operator and a further development of the tournament_selection which is the default setting of the NSGA-II. The special features are the available penalty functions. You can choose between the penalty function with the parameter *penalty_function*. The function has two violation terms:

$$violation = violation_{land\ cover} + violation_{transformation\ information} \quad (1)$$

The violation value of the individual can be calculated with the following two functions:

1. Sum of the differences from the area between the minimal/maximal land cover range and the actual area of the land use categories added to the sum of the patch areas where the transformation matrix is violated

$$\sum_{i=1}^{max_range} Ai + \sum_{n=1}^{length\ individual} An, \quad (2)$$

with land use categories $i = 1, 2, \dots, max_range$ and patches $n = 1, 2, \dots, length\ of\ the\ individual$

2. Quotient of the sum from the added differences of the areas where land cover ranges are violated and the number of the land use categories added to the quotient of the number of patches where the transformation matrix is violated and the length of the individuals (correspond the number of patches which are included in the optimization)

$$\frac{\sum_{i=1}^{max_range} Ai}{max_range} + \frac{\sum_{n=1}^{length\ individual} 1}{length\ of\ the\ individual}, \quad (3)$$

User Guide for the optimization tool SeCoMOLA

with land use categories $i = 1, 2, \dots, \text{max_range}$ and patches $n = 1, 2, \dots$ length of the individual and max_range as number of the land use categories

If the `constraint_tournament_selection` is chosen without a penalty function or only plausible individuals are preselected, then is the normal `tournament_selection` used for the selection of one individual. If one or more individuals of the preselection are infeasible, then is the individual used with the lowest violation value. The selection operators preselect often the same individuals and so the preselected individuals are saved with their violation values in a memory which is used if an individual is selected twice instead of a new calculation.

The evaluation function has been adjusted for the `constraint_tournament_selection`. Only if this special selection method is chosen, the fitness values will be calculated for all individuals regardless if they are feasible or not. Therefore infeasible individuals can be best solutions. For an easier evaluation of the results will be produced at the end of the optimization a second plot and a second csv file only with feasible solutions. The ASCII maps of infeasible best solutions are marked by name as “infeasible”.

3 Instruction manual

3.1 General information and folder structure

The optimization tool is written in the programming language python and modularized designed. The tool need not be installed separately. You can save the tool files at any free memory. The original folder structure contains one main folder with four subfolders. The program, configuration and start files for the usage on model1 or MSG are directly located under the main folder (cf. page 24). The sub folders are:

- input
- inspyred
- models
- output

The input folder contains all files that are needed for a successfully run of the program. This could be a csv file with special vector data like HRU information (Hydrological Response Units), an ASCII map, a text file with the land cover ranges or the worst fitness values (if infeasible individuals are rejected before the models are started) or a text file with the transformation information which land use changes are allowed. The inspyred folder contains the inspyred library. The models are stored as subfolders in the models folder. If modifications of the models are necessary than change it at the original models folder,

User Guide for the optimization tool SeCoMOLA

because the helping folders for the parallelization are created from this one. Chapter 3.6.1 shows you, how you can update the helping folders with the modifications of the models folder. In the output folder will be saved all the files which are created through the optimization tool. This are for example log files, copies of the configuration file and a copy of the created plot.

3.2 How I could start an optimization run?

The following steps are important for a successful start of the optimization algorithm:

1. Adapt the models to fulfil the requirements of the tool
2. Save the models in separately subfolders in the model folder
3. Make the necessary configuration settings on the config.ini file
 - a. Modify the paths
 - b. Specify the model folder, scripts and output files
 - c. Choose a optimization algorithm (GA and NSGA-II are implemented)
 - d. Make the settings of the optimization algorithm
 - e. Make the settings for the creation of a start individual
4. Save or update the necessary files in the input folder (map, land cover ranges, transformation information and so on)
5. Start the run via one of the start files or the command line interpreter. In the latter case it is important that you start the `__init__.py` file with the same python version which you have set in the config.ini file.
6. Close the created plot after a finished optimization run.
7. Check the log files if your models run completely and in a right way, check the optimization process and the plausibility of your results.

Note:

The download folder of the optimization tool provide some example input and model files in the subfolders. You can use these files for your configuration and preparation of the tool.

Note for R models:

In R models will be automatically insert the instruction for setting the working directory at the beginning of the script. Other such existing instructions will be removed. The same way of proceeding is used for the redirection of console outputs which are produced by the R scripts.

User Guide for the optimization tool SeCoMOLA

Important note for model output files:

Please check your model output files, that the only the fitness values, which are needed for the optimization, are listed vertically and no other data exist in this file. If your model creates more than these values, you should prepare the output for the transfer to the optimization tool.

Note for utilizing SWAT models:

- *Provide the TxtInOut folder including an executable file and make sure that the model runs within this folder under the desired operating system*
- *Within the TxtInOut folder copy all mgt-files considered for optimization to an mgt-backup folder*
- *In a folder named “input_params” provide a list of HRUs considered for optimization and pool of all possible management settings (each in a separate mgt file)*
- *If appropriate, adapt start.py and SWATpy.py regarding parameter changes and/or fitness value calculation (one value for each objective)*
- *If SWAT is used in combination with other models, save an HRU-Patch-ID map in the input folder. The HRU-Patch-ID map can be generated by converting the full HRU shapefile from ArcSWAT into ASCII format. (ToDo: should be tested by Michael Strauch)*

3.3 Initialization file config.ini

This program used an initialization file config.ini, where you can make the settings for the optimization tool. A copy of this file will be stored in the output folder.

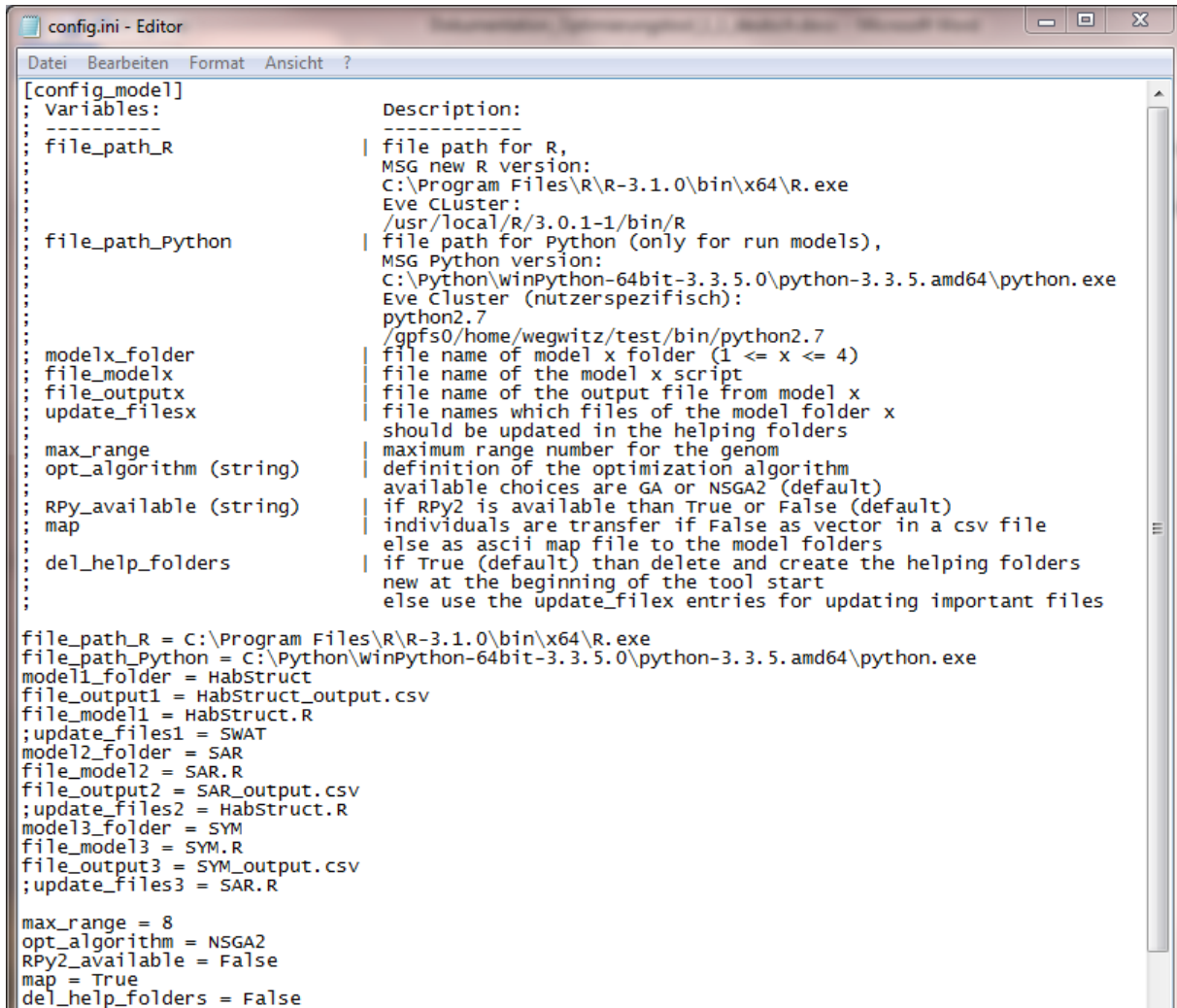
The initialization file is divided in three sections: config_model, config_optimization_algorithm and config_map_analysis.

The first section **config_model** includes model and computer specific information which are important for a successfully run of the program. The variables and their standard values for the model server MSG, model1 and Eve cluster are described in the upper part (Figure 5). These lines are out commented with a semicolon at the beginning of the line. They will not be taken into account for the import of the initialization file.

The following system and model settings are required:

- Path for the R version with the executable R.exe file
- Path for the python version with the executable python.exe file
- Settings for the models (maximal 4 models possible):
 - Name of the model subfolder in the models folder
 - File name of the model start file
 - File name of the model output with the fitness values for the optimization tool
 - As required: information of model files or folders whose should be updated in the helping folders before the optimization starts
- Maximum number of land use options which could be used for the elements of the individuals
- Optimization algorithm (NSGA-II and GA are implemented)
- Information if the RPy2 library is available on the computer (if True, than the R scripts are started in RPy2 environment)
- Information if the individuals of the population should be saved in the model subfolders as vector in a csv file (if False) or as ASCII map (if True)
- Information if the helping folders should be removed and completely new created before the optimization process is started

User Guide for the optimization tool SeCoMOLA



```

config.ini - Editor
Datei Bearbeiten Format Ansicht ?

[config_model]
: Variables: Description:
: file_path_R | file path for R,
: | MSG new R version:
: | C:\Program Files\R\R-3.1.0\bin\x64\R.exe
: | Eve Cluster:
: | /usr/local/R/3.0.1-1/bin/R
: file_path_Python | file path for Python (only for run models),
: | MSG Python version:
: | C:\Python\winPython-64bit-3.3.5.0\python-3.3.5.amd64\python.exe
: | Eve Cluster (nutzerspezifisch):
: | python2.7
: | /gpfso/home/wegwitz/test/bin/python2.7
: modelx_folder | file name of model x folder (1 <= x <= 4)
: file_modelx | file name of the model x script
: file_outputx | file name of the output file from model x
: update_filesx | file names which files of the model folder x
: | should be updated in the helping folders
: | maximum range number for the genom
: max_range | definition of the optimization algorithm
: opt_algorithm (string) | available choices are GA or NSGA2 (default)
: | if RPy2 is available than True or False (default)
: RPy2_available (string) | individuals are transfer if False as vector in a csv file
: map | else as ascii map file to the model folders
: | if True (default) than delete and create the helping folders
: del_help_folders | new at the beginning of the tool start
: | else use the update_filex entries for updating important files
:
file_path_R = C:\Program Files\R\R-3.1.0\bin\x64\R.exe
file_path_Python = C:\Python\winPython-64bit-3.3.5.0\python-3.3.5.amd64\python.exe
model1_folder = HabStruct
file_output1 = HabStruct_output.csv
file_model1 = HabStruct.R
;update_files1 = SWAT
model2_folder = SAR
file_model2 = SAR.R
file_output2 = SAR_output.csv
;update_files2 = HabStruct.R
model3_folder = SYM
file_model3 = SYM.R
file_output3 = SYM_output.csv
;update_files3 = SAR.R
:
max_range = 8
opt_algorithm = NSGA2
RPy2_available = False
map = True
del_help_folders = False
  
```

Figure 5: Model specific settings in the initialization file config.ini

The section with the **settings of the optimization algorithm** is followed by the description of the parameters and their standard values. The standard values for the implemented algorithms are oriented towards the default settings of the inspyred documentation.

Settings, which are set in the initialization file but not usable for the selected optimization algorithms, have no effects on the calculations.

Note:

*If more than one parameter values should be take into account for the calculations for example for „terminator“ with „generation_termination“ and „user_termination“, you need to set the values successively separated with comma but **without** spaces.*

User Guide for the optimization tool SeCoMOLA

You can use the following parameters of the optimization algorithms NSGA-II (see link in chapter 6.3) and GA. In the table you find information about their name and standard value from the inspyred documentation (link in chapter 6.2).

description	name	standard value
population size: Number of created individuals per generation	pop_size	100
optimization direction	maximize	True
selection method	selector	default_selection; NSGA-II: tournament_selection; GA: rank_selection
variation method	variator	default_variation; GA: n_point_crossover, bit_flip_mutation
replacement method	replacer	default_replacement; NSGA-II: nsga_replacement; GA: generational_replacement
migration method	migrator	default_migration
archiving method	archiver	default_archiver; NSGA-II: best_archiver
observer method	observer	file_observer
termination criterion	terminator	default_termination
optional parameter for crossover variation methods	crossover_rate	1.0
optional parameter for crossover variation methods	num_crossover_points	1
optional parameter for variation methods with mutations	mutation_rate	0.1
optional parameter for some of the replacement methods	num_elites	0
optional parameter for some of the termination criterions	min_diversity	0.001
optional parameter for some of the selection methods	num_selected	NSGA-II, GA: 1; truncation_selection: pop_size
optional parameter for some of the selection methods	tournament_size	NSGA-II: 2
optional parameter for some of the variation methods	max_generations	1
optional parameter for some of the termination criterions	max_evaluations	pop_size

In the inspyred documentation are described which parameters are allowed for the individual methods.

User Guide for the optimization tool SeCoMOLA

In addition to the variation methods of the *inspyred* library provide this tool the filter and logical mutation method. These methods are explained in detail in the chapter 0.

Furthermore is a special `constraint_tournament_selection` method implemented which is described in chapter 0.

```
[config_optimization_algorithm]
; available variables see below
pop_size = 18
max_generations = 1
mutation_rate = 0.1
crossover_rate = 1
priority = True
maximize = True
feasible_first_pop = True
extreme_seeds = True

; GA
; terminator = generation_termination,diversity_termination
; NSGA2
terminator = special_termination,generation_termination,diversity_termination
; variator = blend_crossover,gaussian_mutation,filter_mutation
selector = constrained_tournament_selection
penalty_function = 2
variator = n_point_crossover
; variator = n_point_crossover,logical_mutation
; Test
; replacer = crowding_replacement
; selector = rank_selection
; archiver = population_archiver
num_selected = 2
tournament_size = 3
crowding_distance = 3

; Variables:                                Default value:
;-----
pop_size                                | 100
maximize                                | True
selector                                | default_selection
;                                         (NSGA2: tournament_selection;
;                                         GA: rank_selection)
penalty_function                        | 0, 1 or 2 (only for constrained_tournament_selection)
;                                         0: default - tournament_selection without penalty function
;                                         1: sum of differences between cover ranges and real area + sum of patch areas
;                                         2: quotient of the sum of differences between cover ranges and number of
;                                         dynamic land use classes + quotient of the number of patches with violation
;                                         and number of all patches
variator                                | default_variation
;                                         (GA: n_point_crossover,bit_flip_mutation)
;                                         special variator: filter_mutation, logical_mutation
replacer                                | default_replacement
;                                         (NSGA2: nsga_replacement;
;                                         GA: generational_replacement)
migrator                                | default_migration
archiver                                | best_archiver (in inspyred only for NSGA2 as default)
observer                                | file_observer
terminator                              | default_termination
priority                                | True (parameter for logical_variator
;                                         if land use from candidate suggestion is preferentially used)
feasible_first_pop                      | False (if True then the first population
;                                         is generated with the logical_mutation)
extreme_seeds                          | False (if True then for the first population are extreme feasible land cover
;                                         individuals included)
crossover_rate                          | 1.0
num_crossover_points                    | 1
mutation_rate                          | 0.1
num_elites                              | 0
min_diversity                          | 0.001
num_selected                            | 1 (NSGA2, GA, truncation_selection: pop_size)
tournament_size                        | 2 (NSGA2: 2)
max_generations                        | 1
max_evaluations                        | pop_size
crowding_distance                      | 2
```

Figure 6: Configuration settings of the optimization algorithm in the initialization file `config.ini`

You have in addition to all this options the possibility to demand a feasible first population with the parameter `feasible_first_pop`. In this case the individuals will be generated independent of the variation settings with the logical mutation method which guarantees that they fulfil all constraints. All following generations are not influenced by this parameter.

User Guide for the optimization tool SeCoMOLA

In the section for the settings of the **map configuration** are following information possible:

- Information of the first map via csv file with HRU information or as ASCII map (mandatory)
- Transformation matrix of the allowed land use changes as text file
- ASCII map where the individual land use areas are abstracted and coded in patches with patch IDs (alternatively the map is created by the program based on the first ASCII map and transformation information)
- Land cover ranges (minimal/maximal areas per land use)
- Information if four or eight neighbouring cells should be considered for the patch generation
- Text file name of the worst fitness values (required if transformation information or land cover ranges are specified)

The description and the standard values of the individual parameters are listed at the end of the initialization file.

```
[config_map_analysis]
; available variables see below |
file_ASCII_map = land_use.asc
;file_ID_map = patch_ID_map_notrans_inverse.asc
;file_transformation = transformation_info.txt
;file_HRU = Size_HRUS_ff.csv
;file_difference = min_max.txt
four_neighbours = False
file_worst_fitness = worst_fitness_values_maximize.txt

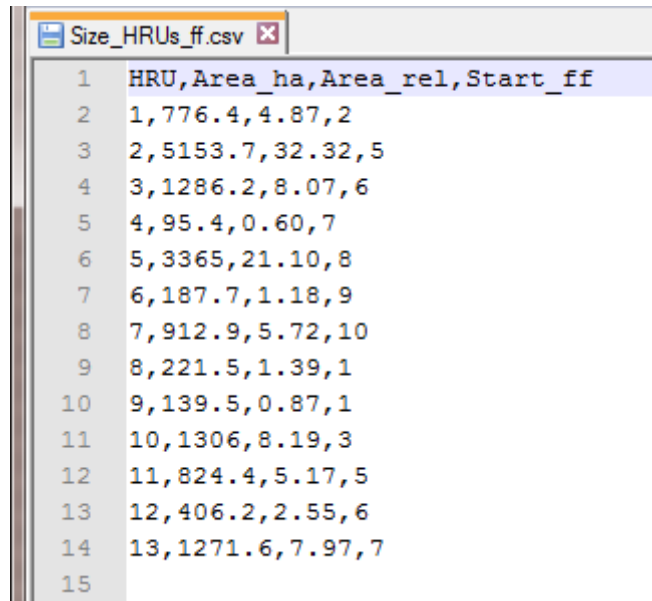
; variables:                Default value/ Description:
;-----
; file_transformation        | file name with characterisation which
;                             | index of the ASCII map can change in which indices
;                             | default: None
; file_difference            | minimal and maximal difference of the
;                             | proportion of all indices of the input map
;                             | default: None
; file_HRU                  | file name of the HRUS
;                             | default: None
; file_ASCII_map            | file name of the ASCII map (original map)
;                             | default: None
; file_ID_map               | file name of the patch ID map
;                             | (if not the implemented generator or an HRU file is used)
;                             | default: None
; file_ID_map               | None
; four_neighbours           | False (analyse of eight neighbour cells)
; file_worst_fitness        | file name with worst fitness values for the individual filter
;                             | default: None
```

Figure 7: Information of the first map and for the map generation during the optimization process are set in the initialization file

3.4 Input files

3.4.1 HRU information in a csv file

In the HRU file are the headers in the first line. In the next lines are the information area and first crop rotation per HRU. The length of the first individual is set of the number of lines from this file minus 1 for the header line.



1	HRU,Area_ha,Area_rel,Start_ff
2	1,776.4,4.87,2
3	2,5153.7,32.32,5
4	3,1286.2,8.07,6
5	4,95.4,0.60,7
6	5,3365,21.10,8
7	6,187.7,1.18,9
8	7,912.9,5.72,10
9	8,221.5,1.39,1
10	9,139.5,0.87,1
11	10,1306,8.19,3
12	11,824.4,5.17,5
13	12,406.2,2.55,6
14	13,1271.6,7.97,7
15	

Figure 8: Example for a HRU csv file

3.4.2 The first map as ASCII file

Instead of the HRU file, you can use a first map as ASCII file. This file can be the real first map or the already coded patch ID map. If you use an ID map than please not that all cells should be zero which are excluded from the optimization process. All other patches should be numbered from 1 to n. You find more information in detail how the patch ID map and the first individual are created from the first map in chapter 2.1.

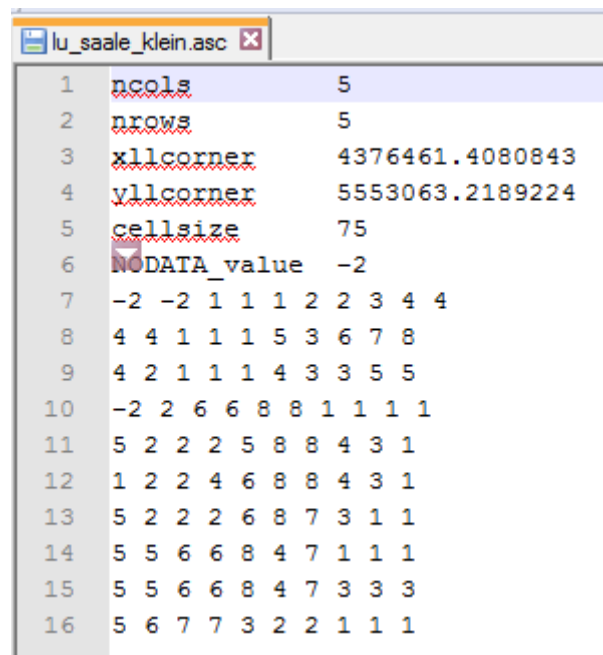
You can find by default in the ESRI ASCII raster format in the upper part of the file the following information:

- ncols – number of cell columns (integer > 0)
- nrows - number of cell rows (integer > 0)
- xllcorner/xllcenter – x coordinate of the original (from the middle or the bottom left corner)
- yllcorner/yllcenter – y coordinate of the original (from the middle or the bottom left corner)

User Guide for the optimization tool SeCoMOLA

- cellsize – cell size > 0
- NODATA_value – value for no data in the output raster (default value is -9999)

Then the individual rows of the original or patch ID map raster follows with the coded land use per cell. The individual cells should be separated with spaces. The rows of the map raster will be read in a matrix which will be used while the optimization process.



```

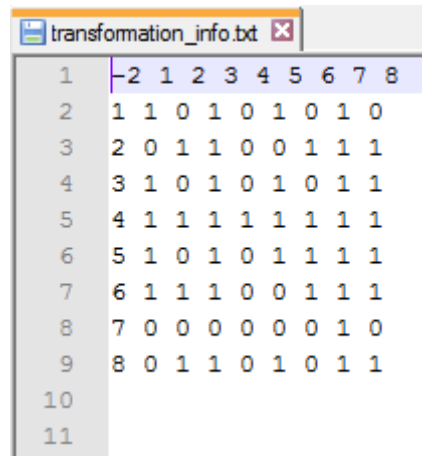
1 ncols 5
2 nrows 5
3 xllcorner 4376461.4080843
4 yllcorner 5553063.2189224
5 cellsize 75
6 NODATA_value -2
7 -2 -2 1 1 1 2 2 3 4 4
8 4 4 1 1 1 5 3 6 7 8
9 4 2 1 1 1 4 3 3 5 5
10 -2 2 6 6 8 8 1 1 1 1
11 5 2 2 2 5 8 8 4 3 1
12 1 2 2 4 6 8 8 4 3 1
13 5 2 2 2 6 8 7 3 1 1
14 5 5 6 6 8 4 7 1 1 1
15 5 5 6 6 8 4 7 3 3 3
16 5 6 7 7 3 2 2 1 1 1
  
```

Figure 9: Small example for a land use map in ASCII format and with coded land use categories

3.4.3 Text file with transformation information

In the text file with the transformation information are the land use coded numbers in the first row and column separated with spaces (the element (0,0) is the NODATA-value). The rest of the matrix is filled with zeros and ones also separated with spaces. One represented that a land use change from the land use of the row to land use of the column is allowed, else zero. The file can be for more land use categories than given by max_range of the initialisation file. Additional land use categories with a number larger than max_range will be ignored by the program.

User Guide for the optimization tool SeCoMOLA

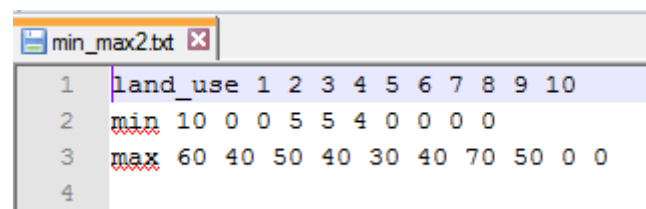


1	-2	1	2	3	4	5	6	7	8
2	1	1	0	1	0	1	0	1	0
3	2	0	1	1	0	0	1	1	1
4	3	1	0	1	0	1	0	1	1
5	4	1	1	1	1	1	1	1	1
6	5	1	0	1	0	1	1	1	1
7	6	1	1	1	0	0	1	1	1
8	7	0	0	0	0	0	0	1	0
9	8	0	1	1	0	1	0	1	1
10									
11									

Figure 10: Example of a matrix with transformation information (1 land use change is allowed, else 0)

3.4.4 The information of the cover ranges

In the text file with the information of the cover ranges per land use category are the land use numbers in the first line. In the following lines are listed the minimal land use area in percent and the maximal area from the land use in the first line. The values are separated with spaces. So for a feasible individual should be the sum of all patch areas with land use i between the minimal and the maximal value. The first column is for the rownames and will be ignored from the program.



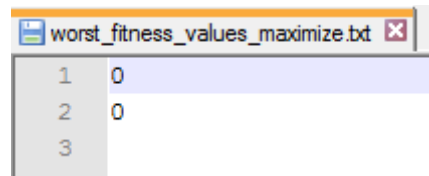
1	land_use	1	2	3	4	5	6	7	8	9	10
2	min	10	0	0	5	5	4	0	0	0	0
3	max	60	40	50	40	30	40	70	50	0	0
4											

Figure 11: Information of the land cover ranges per land use (in percent)

3.4.5 Information of the worst fitness values

If you want to use transformation information or land cover ranges for your optimization, so you need an input file with the worst fitness values. These values will be return to the optimisation algorithm if individuals are infeasible and rejected. In this case the models will not be started for these individuals and you get a better performance. The values should be saved in a text file vertically in the first column and in the order how normally are the return fitness values of the models for the optimisation algorithm. Worst fitness values are characterised with values which are smaller or higher than the expected values of the models depending of the optimization direction. On this way you can be sure that no infeasible individuals are in your best solution of the optimization run.

User Guide for the optimization tool SeCoMOLA



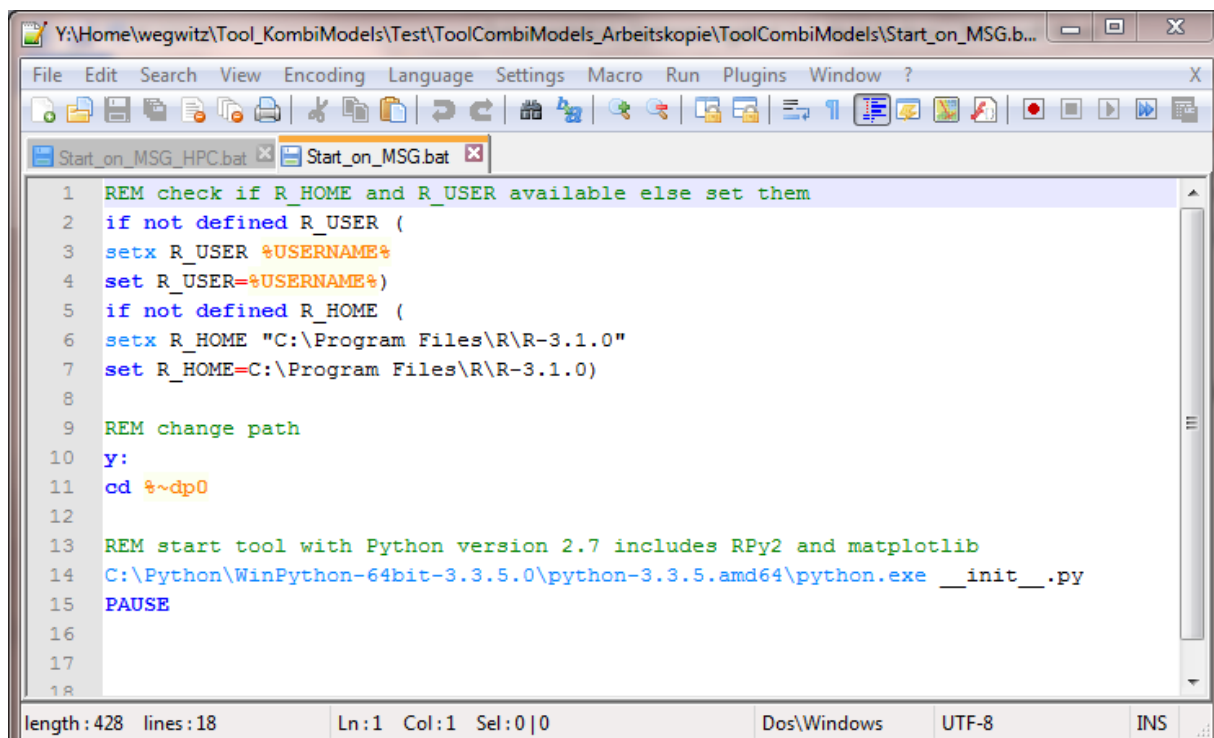
1	0
2	0
3	

Figure 12: Text file with worst fitness values

3.5 Start files for the model server MSG and model1

The start files check via command line interpreter if the environment variables R_HOME and R_USER for R and RPy2 are already set. If the variables not exist than they will be added. Then the directory is changed to the path were the batch files was started. In this directory, the main folder of the optimization tool, will be started the start file `__init__.py` with the python version 3.3 or 2.7 which is the default on the server.

You can find more information in chapter 3.8 how you can start the optimization tool in general on your local computer or a modelling server.



```

1  REM check if R_HOME and R_USER available else set them
2  if not defined R_USER (
3  setx R_USER %USERNAME%
4  set R_USER=%USERNAME%)
5  if not defined R_HOME (
6  setx R_HOME "C:\Program Files\R\R-3.1.0"
7  set R_HOME=C:\Program Files\R\R-3.1.0)
8
9  REM change path
10 y:
11 cd %~dp0
12
13 REM start tool with Python version 2.7 includes RPy2 and matplotlib
14 C:\Python\WinPython-64bit-3.3.5.0\python-3.3.5.amd64\python.exe __init__.py
15 PAUSE
16
17
18
length: 428 lines: 18 Ln: 1 Col: 1 Sel: 0 | 0 Dos\Windows UTF-8 INS

```

Figure 13: Start file for the MSG server

3.6 Specification for the model folders

The start and output file with the fitness values of the individual models have to exist directly in the individual model subfolder of “models”. The fitness values should be saved vertically in the first column of the csv file and without headers. The helping files `genom.csv`, `map.asc` and `console.txt` will be generated automatically from the optimization tool. The current individual will be saved in the `genom.csv` file or as `map` in the `map.asc` file, which is used by the models for the calculation of the fitness values. The console outputs of the models will be redirected in the console text file and these outputs will be collected after the calculations of the current generation in the `model.outputs.txt` file in the output folder of the optimization tool.

The program generates internal copies of the original models folder for the parallel model calculations of the different individuals of one population. Thereby the performance of the total optimization process could be significantly improved, especially for a high population size or long model run times.

3.6.1 Update of the model helping folders

If the helping folders won't be generated new, you have the option for updates of individual file or subfolders of the model folders. Please set the name of the files or folders in the parameter `update_files` of the model if they exist directly in the model subfolders else use the part of the path from the model subfolder to the file or folder.

In the following is a short example (compare with Figure 14) where the file `SWAT2005_ms.exe` exists in the `TxtInOut` subfolder of the SWAT model folder and should be updated:

- `update_filesx = start.py,TxtInOut` is correctly
- `update_filesx = start.py,SWAT2005_ms.exe` would not work because the `SWAT2005_ms.exe` file doesn't exist directly in the model folder
- `update_filesx = start.py,TxtInOut\SWAT2005_ms.exe` is right, because the connection between model folder and `SWAT2005_ms.exe` is given

In this way you can save the time for the new generation of the helping folders if you want to update only individual files or folders or if there are no changes of the model files.

User Guide for the optimization tool SeCoMOLA

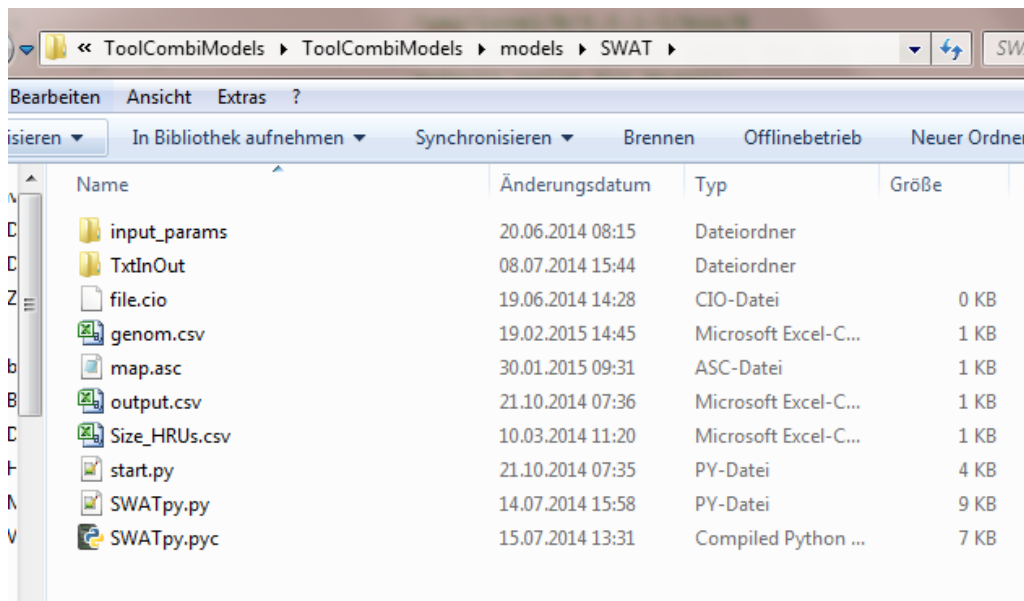


Figure 14: Model structure of the SWAT model

3.7 Log files

In the output folder will be saved all log files which are produced by the program with a time stamp of the start time of the optimization process. These are the following files:

File name	Description
optimization_log.txt	Log file of the SeCoMOLA application
model_outputs.txt	Log file of the model outputs
input_data.txt	Copy of the config.ini file
individuals_file.csv	inspyred log file with all created individuals
inspyred.log	inspyred log file
statistics_file.csv	inspyred log file with statistical evaluation of the populations
best_solutions.csv	File with the best solutions of the optimization run
best_ascii_mapx.asc	Best solutions as ASCII map (if the optimization based on a ASCII map)
patch_ID_map.asc	Patch ID map which was generated by SeCoMOLA
NSGA2_result_plot.png	Result plot of the NSGAII (alternatively as pdf file)

Striking events will be logged automatically in the **optimization_log** file during the optimization process. For example the individuals per generation, the start time of the models, their run times and the fitness values which are transferred to the optimization process will be saved. At the end of the optimization process will be logged the optimization run time, the total run time of the program and the best solutions with the corresponding

User Guide for the optimization tool SeCoMOLA

individuals. This log file gives an overview provide a whether the program runs until the termination and in a correct way.

If a plot of the best solutions was generated, then it is saved as **NSGA2_result_plot** file in png format. You can save the plot alternatively as pdf file. In the most cases the pdf file reduces the file size by 50 percent but the plot is than not anymore lossless scalable. For the optimization algorithm GA is a plot with the inspyred library based on the statistics_file file generated which represents the development of the fitness values during the optimization process. This plot won't be saved in the output folder.

Console outputs which are generated directly by the models will be redirected in the **model_outputs** file of the output folder. In this file is additionally saved the corresponding individual and generation number for every model output for a better structured and understandable file. After the individual follow model path by model path with the output of the model run.

The configuration file config.ini will be copied at the beginning of every optimization run and also saved as **input_data** text file in the output folder.

The three other files individuals_file.csv, statistics_file.csv and inspyred.log are log files which will be generated by the inspyred library. In the **individuals_file.csv** file are solely saved the generated individuals per generation with the corresponding fitness values. The **statistics_file** includes per generation number the population size, the best and worst fitness values, the median, the arithmetic average and the standard deviation of the fitness values. The **inspyred.log** file is the log file of the inspyred library analogous of the optimization_log from the optimization tool. You can check if the optimization algorithm was working as expected and if it was taken all parameters into account which were set in the configuration file. The file **best_solutions.csv** includes the best solutions of the optimization run which information about the individuals and their fitness values.

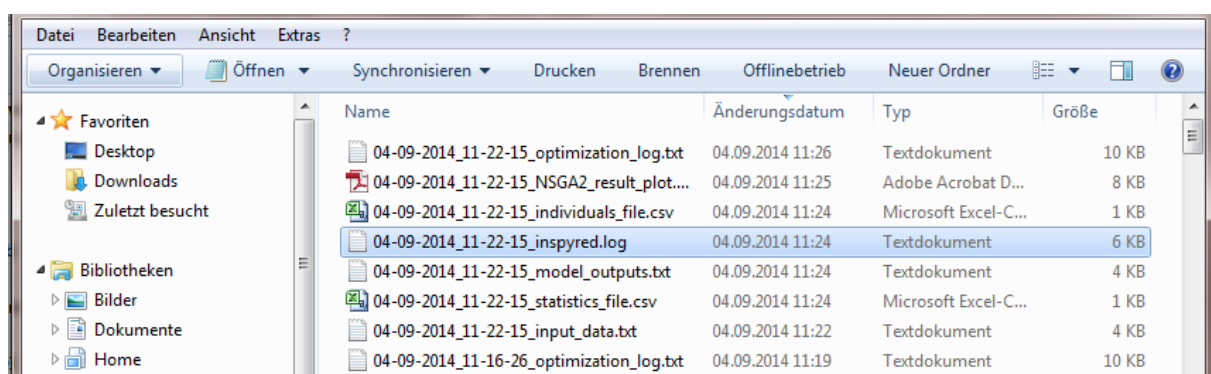


Figure 15: Section of the output folder

User Guide for the optimization tool SeCoMOLA

If the patch-ID map is generated by the program based on an original ASCII map, then is the **patch_ID_map.asc** file saved in the output folder. Furthermore, the best individuals will be converted in maps and saved as **best_ascii_mapx.asc** file if the optimization was based on an original map. x is the number of the individual in the best list.

3.8 Notes for the usage of the tool on the model server MSG, MSG-HPC, Eve Linux cluster or as external user

If you want to use as UFZ employee the model server **MSG** or **MSG-HPC**, then you can save the application folder on your home folder on the Y drive. On this way you could access on the files from both model servers and from your local computer.

As UFZ employee you can also use the **Eve cluster**. Then you have to save the application and the models folders at first on a storage space which is available from the Eve cluster. You can do this for example with WinSCP, if the operating system of your local computer is a Windows version. Furthermore, you need a virtual python environment and an R version. You can find more information in the Wiki of the Eve cluster for example how you can save your data on the Eve cluster and how you can integrate Python and R in your frontend (see appendix 6.4).

(Available only after publishing of the publication): As external user you should download the current SeCoMOLA version on your local computer or an available server. Then check if a python version higher or equal 2.7 with the library matplotlib is installed. If necessary then install it and R if you want to use R models.

You can find all other steps for the configuration of SeCoMOLA and what to do for starting an optimization run in chapter 3.2.

Note:

Please take into account that the different operation systems and R need different path specifications. For Windows you can use backslash for folder separation but on Linux you need slash and for R you can use slash or doubles backslash.

4 Potential errors and instructions for troubleshooting

During the programming of the optimization tool special attention was pay attention that as much as possible meaningful error messages are integrated. For example implemented error messages could appear if a necessary library doesn't exist in the given python path or configuration settings violate predefined rules. This special implemented error messages are printed in the command line interpreter and logged in the optimization_log.txt file. Please do not hesitate to get in touch with your contact person if you get an error message which doesn't help you for solving the problem or a not intercepted system error message appears. So you can help to improve constantly the program and we can extend the helpful error messages or fix problems quickly and efficiently.

5 Differentiation to the following version

This version 1.1 serves as test environment for internal employees. (ToDo: change it, if publication is published -> link to publication and release for external user)

Please do not hesitate to get in touch with the contact person if you have modification requests or indications for an incorrect behaviour of the program.

6 Appendix

6.1 Links for the application

You can find the current version of the tool any time under the following links:

https://svn.ufz.de/optimize/browser/optimisation_tool/trunk?order=name

or

Y:\Gruppen\pof3t12\Inhaltliches\Optimierung\aktuelle_Toolversion

6.2 Links for the inspyred and python user guide

- inspyred: <https://pythonhosted.org/inspyred/reference.html>
- python: <https://docs.python.org/2/>

6.3 Link for the NSGAI paper

http://sci2s.ugr.es/sites/default/files/files/Teaching/OtherPostGraduateCourses/MasterEstructuras/bibliografia/Deb_NSGAI.pdf

6.4 Link for the Eve cluster wiki

http://www.intranet.ufz.de/wiki/eve/index.php/Main_Page

6.5 Citation and credits

SeCoMOLA – Sequential Constraint Multi-Objective Land Allocation

(c) Helmholtz Centre for Environmental Research - UFZ

Department Computational Landscape Ecology

Version 1.1

2016

www.ufz.de

Contact: carola.paetzold@ufz.de or michael.strauch@ufz.de

Credits:

Software developer: Carola Pätzold

Concept and content: Ralf Seppelt, Michael Strauch, Christian Schweitzer

Financial support: Helmholtz Centre for Environmental Research - UFZ

Contact: carola.paetzold@ufz.de or michael.strauch@ufz.de

Citation:

M. Strauch, Anna Cord, C. Schweitzer, R. Seppelt, C. Pätzold (2016) SeCoMOLA. Helmholtz Centre for Environmental Research - UFZ. (ToDo: change if publication is published)