

MaMEA User's Guide

Lukas Turcani

December 7, 2016

Contents

1	Introduction	2
2	Installing MaMEA	2
3	Input Files	2
3.1	Format and structure.	2
3.2	Valid input values.	4
3.2.1	List and explanation of valid keywords.	4
3.2.2	Finding available functions and their arguments.	5
4	Running MMEA	5
4.1	From the command line.	5
5	Output	5
6	MMEA as a Library	5

1 Introduction

MMEA (MacroMolecular Evolutionary Algorithm) is a genetic algorithm (GA) for chemistry. It aims to be as general as possible, being suitable for use with any class of molecules. While it does not support all types of molecules out of the box, it is designed to be easily extended. This is made easier by being written in Python. For notes on how to extend MMEA to a new class of molecules or add GA operations, see the developer's guide.

2 Installing MaMEA

3 Input Files

To run MaMEA you need to submit an input file. These contain all the details required to start a GA calculation. Among other things, this may include things like which fitness function you want to use, the number of generations MMEA should make and how big the population should be.

Using an example, the format and structure of input files is explained in section 3.1. A guide to finding out what GA operations and tools are available is in section 3.2.

3.1 Format and structure.

An example input file can be downloaded from https://sabercathost.com/1aoC/mamea_input.in.

Any lines consisting of only whitespace or beginning with “#” are ignored. This is used to create comments in the input file and allows division of it into sections. Notice that there are two kinds of non-comment line in the input file. The first kind has the form

```
keyword; function_name; param1=val1; param2=val2; param3=val3
```

Lines 5, 11, 17 in the example input file are examples of this. Here is line 11 (in case you did not download the file).

```
generational_select_func; stochastic_sampling; use_rank=True
```

This means that in line 11 the keyword is “generational_select_func”, the function name is “stochastic_sampling”, parameter 1's name is “use_rank” and its value is “True”. As you can see, in this line there is only 1 parameter name-value pair.

Lines with this format define functions used by MMEA. It is called the “function definition format”. These functions will correspond to the various GA operations, such as crossover, mating and selection or to things like the fitness function or optimization function (more on this later). The words on these lines are separated from the adjacent ones by a semicolon. Notice that the last word on the line is not followed by a semicolon. The keyword defines

which GA operation the line describes. For example, in line 11 the keyword “generational_select_func” signifies that the line describes which function is to be used for selecting members of the next generation. In line 17 the keyword “parent_select_func” signifies that the line defines the function to be used for selecting parents for crossover. A list of valid keywords is described in section 3.2.

The parameter “function_name” is the name of the function which is to carry out the role designated by the keyword. This must correspond to the name of a function defined within MMEA. For example, if the keyword is “generational_select_func” then “stochastic_sampling” or “roulette” are both valid values for “function_name”, because the module “selection.py” within MMEA defines functions with these names. Indeed, any function defined within this module can be used as the selection function.

Finally, on line 11 “param1” and “val1” correspond to “use_rank” and “True”, respectively. As mentioned before “stochastic_sampling” is the name of a function defined within the “selection.py” module. As is usual for Python functions, it takes arguments. Looking at “stochastic_sampling()” within “selection.py” it can be seen that its arguments are “cls” and “population” and its keyword arguments are “elitism” (defaults to “False”), “truncation” (defaults to “False”), “duplicates” (defaults to “False”) and “use_rank” (defaults to “False”).

In the case of selection functions the first two arguments (“cls” and “population”) are dealt with automatically by MMEA. If the user wants to use one of the selection functions they have to provide values to any of the remaining arguments in the form “param1=val1”. The order in which they are given in the input file does not matter. If the user wants to change the default values of any of the function’s keyword arguments, they may do so using the same notation on the same line. Keyword arguments otherwise do not have to be provided in the input script and will be default initialized to whatever is defined within the source code of MMEA. All of this holds true for other functions defined in the input file, not just selection functions. How a user can easily view what functions and parameters are available, as well as what they mean, is addressed in section 3.2.

All keywords should be defined in the input file once. The exception to this is “mutation_func”.

There is a second kind of non-comment line in the input file of MMEA. These lines have the form

```
keyword=val
```

As you can see, these are much simpler. They define basic constants relevant to the GA. As a result, this is called the “constant definition format”. This includes things like population size, number of matings carried out during each generation and so on. It is merely the keyword, a “=” and the value. No spaces. Valid keywords for these lines are discussed in section 3.2.

3.2 Valid input values.

In order to prevent the need to continuously update this documentation when a new function is added to MMEA, only valid keywords are described here. However, this does not mean a user has to go into the source code to find out what functions they have at their disposal. This section describes how MMEA can be made to output valid function names and parameters for each keyword.

3.2.1 List and explanation of valid keywords.

Here a list of valid keywords, the type of line formatting the keyword requires and which arguments do not need to be provided to the functions. Recall in section 3.1 the selection function did not require the “population” argument to be provided explicitly in the input file. Other functions, such as crossover/mating functions do not require other parameters to be provided. The format here is:

keyword - *line format* - excluded_param1, excluded_param2 - Explanation of keyword.

The keywords are:

init_func - *function definition format* - ga_tools - The function used to create the initial population.

generational_select_func - *function definition format* - population - The function used to select members of the next generation.

parent_select_func - *function definition format* - population - The function used to select parents for crossover.

mutant_select_func - *function definition format* - population - The function used to select individuals for mutation.

mating_func - *function definition format* - macro_mol1, macro_mol2 - The function which carries out the mating/crossover operation.

mutation_func - *function definition format* - macro_mol - The function which carries out the mutation operation.

opt_func - *function definition format* - macro_mol - The function which optimizes the geometry of the molecules.

fitness_func - *function definition format* - macro_mol - The function which calculates fitness.

num_generations - *constant definition format* - The number of generations

which MMEA will create.

num_mutations - *constant definition format* - The number of mutations MMEA will carry out each generation.

num_matings - *constant definition format* - The number of mating/crossover operations MMEA will carry out each generation.

pop_size - *constant definition format* - The size of the population.

mutation_weights - *constant definition format* -

3.2.2 Finding available functions and their arguments.

4 Running MMEA

4.1 From the command line.

5 Output

6 MMEA as a Library