**Enki User Guide**

**I. Enki Commands**

This section describes each command available for use within the Enki framework. Note: in Windows environments, substitute the .cmd file extension for each script.

**I.a. Enki search application**

I.a.1. Description: This is command launches an Enki search on the given executable script. The format of executable scripts will be described in more detail in Section II.

I.a.2. Usage: *enki.sh executable\_path [options]*

I.a.3. Options:

--help Displays a brief description of all command-line options.

--debug Provides extra information in the event of a failure, for debugging purposes.

--db Stores search results in a SQL-lite database file. This is required for use with the web application. Note: to use this option, you must first run `enki\_database.sh --create` to create a database file. (Default: False)

--silent Suppresses all console output. (Default: False)

--archive-label A custom label for the archive, displayed in the web application. (Default: None)

--archive-percentile A percentile threshold for which individuals to consider for archiving in each generation. When committing a population to the archive, each individual is ranked against the archive, and only those that exceed this percentile will be combined with the archive. (Default: 0.75)

--archive-size The maximum size for the archive. When the archive reaches this limit, new individuals may still be added to the archive, but the combined archive will be truncated by discarded the least ranking individuals from the combined archive. (Default: 10)

--distance-type The type of distance metric to use for computing novelty scores. Options are “manhattan” and “euclidean.” (Default: euclidean)

--early-termination Terminates a search as soon as the archive is full. (Default: False)

--enable-mutation Enables mutation in the evolution search procedure. (Default: True)

--enable-recombination Enables recombination in the evolution search procedure. (Default: True)

--mutation-rate The probability for any single value in a genome to be mutated. (Default: 0.25)

--mutation-shift The degree of mutation allowed for any single mutated value in a genome. (Default: 0.2)

--num-generation The number of generations to executed in the evolution search procedure. (Default: 10)

--num-nearest-neighbors The number of nearest neighbors to consider when computing novelty scores. (Default: 3)

--num-processes The number of processes to use on the local processor when evaluating a generation. (Default: 1)

--output-path Indicates where to save the resulting archive data. Options include JSON or Python Pickle formats. (Default: None)

--population-size The number of individuals to include in each generation of the evolution search procedure. (Default: 10)

--random-seed Sets a fixed random seed for the search. (Default: None)

--replay-path A file path to an existing Enki archive. Using this option will perform another search, selecting the same identical individuals. (Default: None)

--save-often Using this option will force Enki to save the archive after each generation. (Default: False)

--search-bias Indicates the direction Enki should rank each individuals. Options are “max” or “min.” (Default: max)

--search-objective Indicates the search objective use for ranking each individual. Options include “fitness,” “novelty,” and “none.” (Default: novelty)

--search-type Indicates the search method to be used by Enki. Options include “evolution” and “random.” (Default: evolution)

--tournament-size Sets how many comparisons to use during tournament selection in the evolution search procedure. (Default: 3)

--track-evaluated Forces Enki to keep track of each individual evaluated during the search. This will reduce redundant computation when individuals are generated more than once. This will also include each evaluated individual in the output archive file, in addition to the archived individuals. If memory resources are limited, try setting this option to False. (Default: True)

--track-generations Forces Enki to keep a record of the individuals archived after each generation. If memory resources are limited, try setting this option to False. (Default: True)

**I.b. Enki database command**

I.b.1. Description: This is command allows the user to manage a SQL-lite database file for use by Enki.

I.b.2. Usage: *enki\_database.sh [options]*

I.b.3. Options:

--help Displays a brief description of all command-line options.

--add-archive Adds an existing archive data file to the database.

--create Creates a new SQL-lite database file in the current directory.

--debug Provides extra information in the event of a failure, for debugging purposes.

--destroy Removes all data from the database.

--list Displays a list of all Enki runs stored in the database.

--pickle Creates a pickled archive data file for each Enki run stored in the database.

--reset Destroys the existing database file, and re-creates a new database file.

--vacuum After extended usage, the SQL-lite database file does not automatically shrink when records have been removed. Use this option to reduce the size of the database file.

**I.c. Enki executable launcher command**

I.c.1. Description: This is command will execute a single instance of the given executable script with random inputs. This is useful for debugging and evaluation purposes, when developing an executable script.

I.c.2. Usage: *enki\_launcher.sh executable\_path*

I.c.3. Options:

--help Displays a brief description of all command-line options.

**I.d. Enki plotter command**

I.d.1. Description: This is command will generate plots from an archive data file.

I.d.2. Usage: *enki\_plotter.sh archive\_path*

I.d.3. Options:

--help Displays a brief description of all command-line options.

--debug Provides extra information in the event of a failure, for debugging purposes.

--generation A single generation in the archive to plot. (Default: None)

--mds-radius A fixed plot radius to use for MDS plots. (Default: None)

--plot-all Includes all evaluated individuals in a plot, instead of only the archived individuals.

--plot-all-up-to Includes all individuals evaluated up to the selected generation in each plot.

--show Immediately displays plots instead of saving them to a file.

**I.e. Enki reader command**

I.e.1. Description: This is command will display the contents of an archive data file in the console. Running without any options will displays the min, max, and average score for each generation of the search. Adding an option to select a generation by its number will displays scores for each individual in the selected generation. Adding an option to select an individual by its number will display details about the selected individual.

I.e.2. Usage: *enki\_reader.sh archive\_path*

I.e.3. Options:

--help Displays a brief description of all command-line options.

--debug Provides extra information in the event of a failure, for debugging purposes.

--generation Displays results for the given generation. (Default: None)

--individual Displays results for the given individual. (Default: None)

**I.f. Enki sandbox command**

I.e.1. Description: This is command will create a sample sandbox directory with a sample test executable script.

I.e.2. Usage: *enki\_sandbox.sh [options]*

I.e.3. Options:

--create Creates the sandbox directory.

**I.g. Enki web application command**

I.e.1. Description: This is command will create a web server on the local host for the Enki web application and also launch a web browser pointing to the application.

I.e.2. Usage: *enki\_web.sh [options]*

I.e.3. Options:

--host The IP address for hosting the web server. (Default: 127.0.0.1)

--port The port for hosting the web server. (Default: 5000)

**II. Enki Executable Scripts**

This section describes the basic format expected by an Enki executable script. Enki requires an executable script for each search. This script defines the search parameters, evaluation parameters, and evaluation procedure for the application being assessed by Enki. The script is expected in the format of an executable Python script. It must contain a class that is derived from the *EnkiExecutable* object, and this class must define an *operation\_specification* property, *behavior\_specification* property, and a *execute* function. During Enki’s search procedure, a separate instance of this class will be instantiated for each individual in the population.

**II.a. Operation Specification**

The operation specification defines the search space for Enki. It is expected in the form of a Python dictionary, where each entry is a separate operational parameter for the system being assessed by Enki. The key for each entry in the dictionary will be the name of the operational parameter, and the value for each entry will be a range of permissible values for the operational parameter. For example. `’x’: [0, 100]` defines an operational parameter named ‘x’ and indicates to Enki that any value between 0 and 100 is permitted for x.

Enki supports both numeric ranges or lists of values for each operational parameter. When only two values are supplied, a numeric range is assumed and any value within that range may be selected. When more than two values are supplied, Enki will only select values within the provided list of values. For example, `’color’: [‘red’, ‘blue’, ‘green’]` will define an operational parameter named ‘color’ and Enki will select any of the values ‘red,’ ‘blue,’ or ‘green.’

**II.b. Behavior Specification**

The behavior specification defines which values Enki will consider when ranking each individual set of operational parameters selected during its search. These parameters may or may not include the parameters defined in the operation specification. The format of this property is identical to the operation specification: a Python dictionary with each entry defining a parameter name and permissible values.

**II.c. Execute function**

The execute function defines how Enki will evaluate each selected set of values for the given operational parameters. Each value for each key will be within the ranges defined in the operational specification. The intention is for the execute function to set up an environment that reflects the provided operational values, execute some functionality under that context, return the results. It takes as input a dictionary of values with keys matching the operational specification, and it is expected to return a dictionary of values with keys matching the behavior specification.

**II.d. Example script.**

The following script provides a bare-bones example of how to construct an executable script. This executable takes two values: *x* and *y*, which represent a Cartesian coordinate. The execute function then computes the distance of the given coordinate from the origin (0, 0) and returns this distance as *r*. The operation specification defines the range of values permitted for *x* and *y*, and the behavior specification defines the range of values expected by *r*. Executing an Enki search on this script will return an archive of selected *x* and *y* values that result in a diverse range of *r* values.

For example, running the below script with Enki with the following command:

*enki.sh sandbox\TestExecutable.py –archive-size=3 --random-seed=1 --output-path=test.json*

will result in the following individuals selected for the output archive:

1. x,y=(-0.99980, 0.07614), r=1.00269

2. x,y=(-0.01533, 0.11738), r=0.11838

3. x,y=(-0.40034, 0.30854), r=0.50543

Note: the archived *x* and *y* values have result in very different *r* values.

# -\*- coding: utf-8 -\*-  
*"""  
  
This is a simple example test executable script for use with Enki.  
  
Notes: Every executable script requires a class to be defined that is inherited from the EnkiExecutable class. This  
class must include three things:  
 1.) An operation\_specification property that returns a dictionary of values for Enki to explore.  
 2.) A behavior\_specification property that returns a dictionary of values for Enki to diversify.  
 3.) A execute function that takes a dictionary of values that matches operation\_specification and returns a  
 dictionary of values that matches behavior\_specification. The purpose of this function is to evaluate any given  
 system within a selected set of operational values and monitor its behavior under the corresponding condition.  
"""*import numpy as np  
from enki.core.interface import EnkiExecutable  
  
  
class TestExecutable(EnkiExecutable):  
 *"""This is a simple example test executable for use with Enki. It allows Enki to search a space of possible  
 (x, y) values, evaluate the distance of each selected point from the origin (0, 0), and diversify selected points  
 based on the their distance from the origin.  
  
 """* @property  
 def operation\_specification(self):  
 *"""This property specifies each operational parameter for the system with possible value ranges for Enki to  
 explore. In this example, there are two parameters "x" and "y," which represent a coordinate position. Each  
 parameter can take values between -1 and 1. Enki also supports lists of discrete values as well.* ***:return****: a dict of operational parameters with value ranges  
 """* return {  
 'x': [-1.0, 1.0],  
 'y': [-1.0, 1.0],  
 }  
  
 @property  
 def behavior\_specification(self):  
 *"""This property specifies each behavior parameter for the system with possible value ranges for Enki to  
 diversify. In this example, there is one parameter "r," which represents the distance of a point from the  
 origin. This parameter can take values between 0 and sqrt(2).* ***:return****: a dict of behavior parameters with value ranges  
 """* return {  
 'r': [0.0, 1.4142],  
 }  
  
 def execute(self, operation\_values):  
 *"""This function specifies how Enki evaluates each selected individual in its search. It is given a dictionary  
 of values matching the operation\_specification. (Each parameter in this dictionary will have a single value  
 sampled from the ranges indicated in the operation\_specification.) This function can be defined to do anything  
 you wish, i.e. it make call and execute any external system, however it must return a dictionary of behavior  
 values matching the defined behavior\_specification.* ***:param*** *operation\_values: a dict of operation values with the same parameters as operation\_specification* ***:return****: a dict of behavior values with the same parameters as behavior\_specification  
 """* # get operation values:  
 x = operation\_values['x']  
 y = operation\_values['y']  
  
 # execute some stuff  
 # for example: compute distance to origin within the current operational context.  
 r = np.sqrt(x \* x + y \* y)  
  
 # pack behavior values  
 behavior\_values = {  
 'r': r  
 }  
  
 # return behavior values  
 return behavior\_values

**III. Enki Archive Data**

The section describes the format of the archive data file output by Enki. Enki can output archive data in either JSON or Python pickle formats. The Python pickle format is binary and more compact. However, the JSON format is in a text format, and therefore, readable by any common text editor.

**III.a. Archive contents**

An archive data file contains the following information:

III.a.1. *executable\_path*: a file path to the executable used by Enki to generate the archive data file.

III.a.2. *executable\_name*: a stripped-down name of the executable used by Enki to generate the archive data file.

III.a.3. *operation\_specification*: a copy of the operation specification used by Enki to generate the archive data file.

III.a.4. *behavior\_specification*: a copy of the behavior specification used by Enki to generate the archive data file.

III.a.5. *settings*: a copy of all of the configuration options used by Enki to generate the archive data file, e.g., the archive size, mutation rate, etc.

III.a.6. *evaluated\_individuals*: A list of each individual evaluated by Enki when generating the archive data file.

III.a.6.1. Each individual includes an *id*, *birth\_history*, *score*, *operation\_values*, and *behavior\_values*.

III.a.7. *archived\_individuals*: A list of each generation of individuals archived by Enki when generating the archive data file.

III.a.7.1. Each individual includes an *id*, *birth\_history*, *score*, *operation\_values*, and *behavior\_values*.

**III.b. Using Enki’s built-in ArchiveFileReader**

An ArchiveFileReader object is provided for reading Enki archive files. See the code snippet below for example usage.

from enki.core.io import ArchiveFileReader

reader = ArchiveFileReader(archive\_path)  
archive = reader.read\_archive()

individuals = archive.archived\_individuals

The final line returns a list of the individuals selected for the last generation of Enki’s search.