DESDEO

Release 0.9

Multiobjective Optimization Group

May 27, 2021

CONTENTS

1 Packages **3**
- 1.1 Introduction . . . 3
- 1.2 Software . . . 5
- 1.3 Guides and Examples . . . 8
- 1.4 Contributing . . . 8
- 1.5 Glossary . . . 13

Index **15**

Decision Support for computationally Demanding Optimization problems

DESDEO is a free and open-source Python-based framework for developing and experimenting with interactive multiobjective optimization. It contains implementations of some interactive methods and modules that can be utilized to implement further methods.

We welcome you to utilize DESDEO and develop it further with us.

For news and other information related to DESDEO, see the official website.

CHAPTER

ONE

PACKAGES

1.1 Introduction

1.1.1 DESDEO

Decision Support for computationally Demanding Optimization problems

DESDEO is a free and open-source Python-based framework for developing and experimenting with interactive multiobjective optimization.

DESDEO contains implementations of some interactive methods and modules that can be utilized to implement further methods.

We welcome you to utilize DESDEO and develop it further with us.

DESDEO brings interactive methods closer to researchers and practitioners worldwide by providing them with implementations of interactive methods.

DESDEO is part of DEMO (Decision analytics utilizing causal models and multiobjective optimization which is the thematic research area of the University of Jyväskylä.

1.1.2 Mission

The mission of DESDEO is to increase awareness of the benefits of interactive methods make interactive methods more easily available and applicable. Thanks to the open architecture, interactive methods are easier to be utilized and further developed. The framework consists of reusable components that can be utilized for implementing new methods or modifying the existing methods. The framework is released under a permissive open source license.

1.1.3 Multiobjective optimization

In multiobjective optimization, several conflicting objective functions are to be optimized simultaneously. Because of the conflicting nature of the objectives, it is not possible to obtain individual optima of the objectives simultaneously but one must trade-off between the objectives.

1.1.4 Interactive methods in multiobjective optimization

Interactive methods are iterative by nature where a decision maker (who has substance knowledge) can direct the solution process with one's preference information to find the most preferred balance between the objectives. In interactive methods, the amount of information to be considered at a time is limited and, thus, the cognitive load set on the decision maker is not too demanding. Furthermore, the decision maker learns about the interdependencies among the objectives and also the feasibility of one's preferences.

1.1.5 The Research Projects Behind DESDEO

About the Multiobjective Optimization Group

The Multiobjective Optimization Group developes theory, methodology and open-source computer implementations for solving real-world decision-making problems. Most of the research concentrates on multiobjective optimization (MO) in which multiple conflicting objectives are optimized simultaneously and a decision maker (DM) is supported in finding a preferred compromise.

About the DESDEO research project

DESDEO contains implementations of some interactive methods and modules that can be utilized to implement further methods. DESDEO brings interactive methods closer to researchers and practitioners world-wide, by providing them with implementations of interactive methods.

Interactive methods are useful tools for decision support in finding the most preferred balance among conflicting objectives. They support the decision maker in gaining insight in the trade-offs among the conflicting objectives. The decision maker can also conveniently learn about the feasibility of one's preferences and update them, if needed.

DESDEO is part of DEMO (Decision analytics utilizing causal models and multiobjective optimization) which is the thematic research area of the University of Jyväskylä (jyu.fi/demo).

We welcome you to utilize DESDEO and develop it further with us.

About DAEMON

The mission of DAEMON is method and software development for making better data-driven decisions. The project considers data and decision problems from selected fields as cases to inspire the research and demonstrate the added value.

In DAEMON, we support optimizing conflicting objectives simultaneously by applying interactive multiobjective optimization methods, where a decision maker (DM) incorporates one's domain expertise and preferences in the solution process. Overall, we model and formulate optimization problems based on data, so that DMs can identify effective strategies to better understand trade-offs and balance between conflicting objectives. In this, we incorporate machine learning tools, visualize trade-offs to DMs and consider uncertainties affecting the decisions.

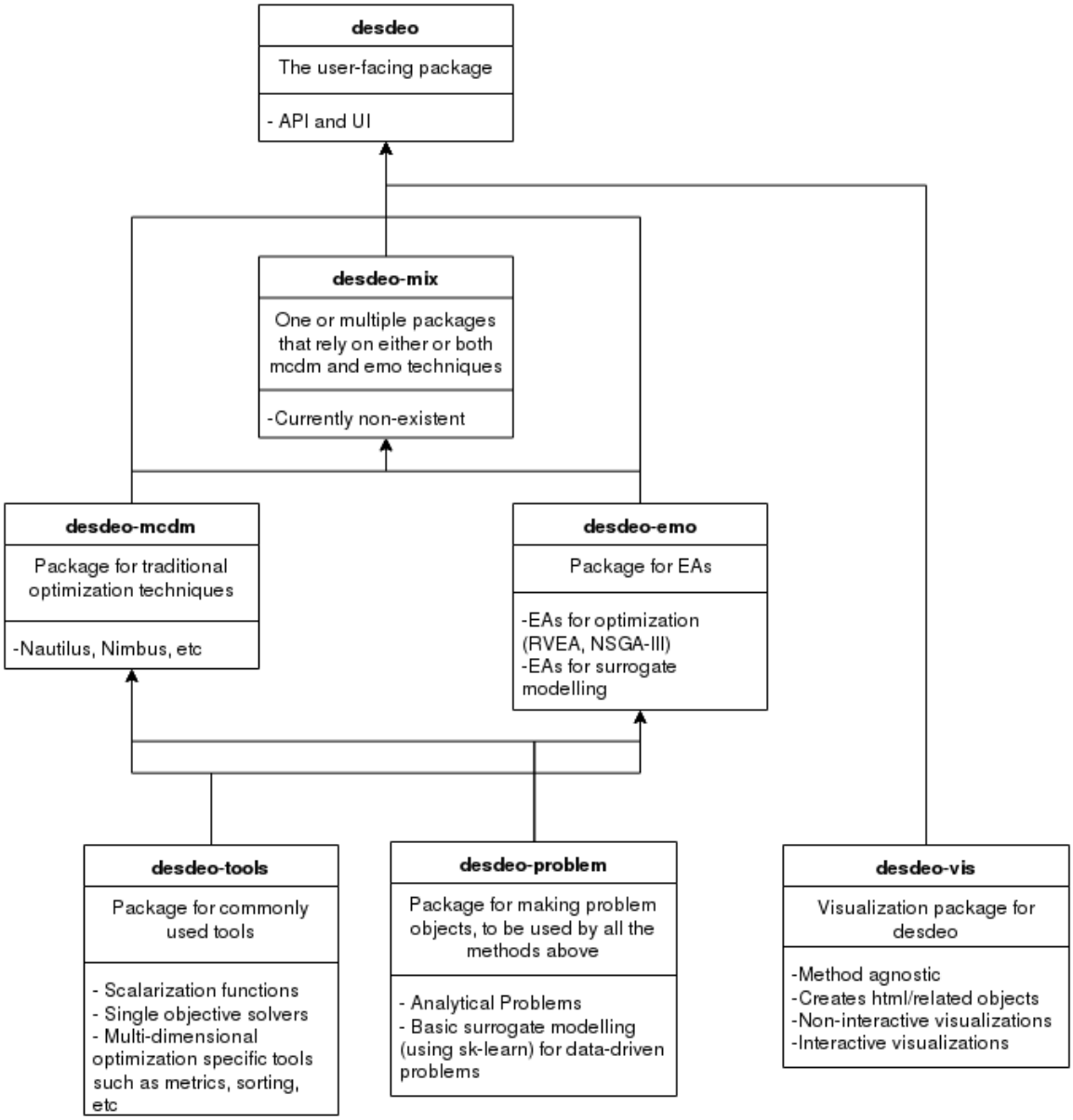
1.1.6 Publications Related to DESDEO

See publications.

1.2 Software

1.2.1 Modular Structure

The DESDEO framework consists of various modules defined as Python software packages. The modular structure can be seen in the image below, and a short description of the different packages follows. The descriptions contain links to the invdividual documentation pages for each of the package.


desdeo
The user-facing package
- API and UI
desdeo-mix
One or multiple packages that rely on either or both mcdm and emo techniques
-Currently non-existent
desdeo-mcdm
Package for traditional optimization techniques
-Nautilus, Nimbus, etc
desdeo-emo
Package for EAs
-EAs for optimization (RVEA, NSGA-III)
-EAs for surrogate modelling
desdeo-tools
Package for commonly used tools
- Scalarization functions
- Single objective solvers
- Multi-dimensional optimization specific tools such as metrics, sorting, etc
desdeo-problem
Package for making problem objects, to be used by all the methods above
- Analytical Problems
- Basic surrogate modelling (using sk-learn) for data-driven problems
desdeo-vis
Visualization package for desdeo
-Method agnostic
-Creates html/related objects
-Non-interactive visualizations
-Interactive visualizations

desdeo-problem

The desdeo_problem package contains tools and classes for defining and modelling multiobjective optimization problems. The defined problem classes can be used in the other packages in the DESDEO framework, such as desdeo_mcdm and *desdeo-emo*.

desdeo-tools

The desdeo_tools package contains tools to facilitate different tasks in the other packages in the DESDEO framework. These tools include, for example, scalarization routines and various solvers.

desdeo-emo

The desdeo_emo package contains evolutionary algorithms for solving multiobjective optimization problems. These algorithms include, for example, interactive RVEA.

desdeo-mcdm

The desdeo_mcdm package contains traditional methods for performing interactive multiobjective optimization. These methods include, but are not limited to, Synchronous NIMBUS and E-NAUTILUS, for example.

desdeo-vis

(*coming soon*) The desdeo_vis package contains tools for building visualizations.

desdeo-mix

(*coming soon*) The desdeo_mix package contains routines and algorithms which make use of the other packages in the DESDEO framework, and warrant their own implementation to be included as part of the framework.

1.2.2 Installation

To install a single package, see that package's documentation. To install the whole DESDEO framework, follow one of the alternatives given below according to the operating system on the machine you plan to install DESDEO on.

Linux and Windows

The recommended way of installing the DESDEO framework is to use pip by invoking the command:

```
pip install desdeo
```

For developing, usage of the poetry dependency management tool is recommended. However, no framework code should be contributed to the main DESDEO package. Instead, contributions should go to the relevant subpackage, which are listed here. See the documentation for the individual for further details. See *Modular Structure* for links to the individual documentations.

OSX

Instruction coming eventually.

1.3 Guides and Examples

1.3.1 Guides

Topical guides to be added eventually.

1.3.2 Examples

- How to use interactive methods: examples in desdeo-mcdm's documentation
- Examples on how to apply useful utilities: examples desdeo-tools's documentation
- How to define different problems: examples in desdeo-problem's documentation

1.4 Contributing

We welcome anybody interested to contribute to the packages found in the DESDEO framework. These contributions can be anything. Contributions do not have to necessarely be ground-breaking. From fixing typos in the documentation to implementing new multiobjective optimization methods, everything is welcome!

1.4.1 Guidelines and Conventions

The guidelines for code to be included in DESDEO should follow a few simple rules:

- Use spaces instead of tabs. Four spaces are used for indenting blocks when writing Python.
- Each line of code should not exceed 120 characters.
- Try to use type annotations for functions using Python's type hints.
- For naming classes use `CamelCase`, and for naming functions and methods use `snake_case`.
- Do not use global or file level declarations. Try to always encapsulate your declarations inside classes.
- Classes should be designed with duck typing conventions in mind.

Docstring style

The docstring style used throughout the DESDEO framework follows the style dictated by Google. Each function, class, and method should be documented.

1.4.2 Software development

This sections contains some basic tips to get started in contributing to DESDEO. We expect contributors to be proficient in at least the basics of Python.

Version control

DESDEO and all of its packages are under version control managed with `git`. It is highly suggested that anybody wanting to contribute to DESDEO should be first familiar with the basic principles of `git`. If terms such as *branching*, *committing*, *merging*, *staging*, and *cloning* are alien to the reader, a brief review of the basics of `git` is in place before contributing to DESDEO can start. At least if the reader wishes to make the experience as painless as possible...

For resources to learn git, git's official documentation is a very good place to start. Many other resources exists on the web as well. We will not be listing them all. However, if a gamified approach is desired instead of having to read documentation, an in-browser tutorial is available here.

Project management

The packages in DESDEO depend on many external Python packages. Sometimes a particular version of an external package is needed, which does not match the current version of the same external package on the computer's system DESDEO is going to be developed on. This can lead to many dependency problems. It is therefore highly recommended to use *virtual environments* when developing DESDEO to separate the packages needed by DESDEO from the packages present on the system's level.

Poetry is a modern and powerful tool to manage Python package dependencies and for building Python packages. `Poetry` is used throughout the DESDEO framework to manage and build its packages. While `Poetry` is *not* a tool for managing virtual environments, it nonetheless offers very simple commands to trivialize the task. Anybody contributing code to DESDEO should be familiar with `Poetry`.

Sometimes the Python version installed on a system is not compatible with the Python version required in DESDEO (3.7.x). In this case, it is recommended to use an external tool, such as pyenv, to facilitate the task of switching from one Python version to an other.

Example on how to get started

Once the reader is familiar with `git` and `Poetry`, starting to develop DESDEO should proceed relatively painlessly. Suppose we have a feature X which we wish to implement in DESDEO's `desdeo-mcdm` package. It is higly recommended to first `fork` the `desdeo-package` on GitHub on a separate repository and then cloning that repository. When doing so, use the forked repository's URL instead of the main repotiry's URL when cloning the project using `git`. Make sure to also setup your forked repository to be configured properly to be able to synchronize it later with the upstream repository.

We proceed by cloning the repository on our local machine:

```
$ git clone https://github.com/industrial-optimization-group/desdeo-mcdm
```

Next, we should switch to the newly created directory:

```
$ cd desdeo-mcdm
```

We can now easily use `Poetry` to first create a Python virtual environment by issuing Poetry's `shell`- command and then use Poetry's `install`-command to install the `desdeo-mcdm` package locally in our newly created virtual environment:

```
$ poetry shell
$ poetry install
```

The `install` command might take a while. Once that is done, `desdeo-mcdm` should now be installed in our virtual environment and be fully usable in it.

To start implemeting our new feature X, we should start by making a new `branch` and switching to it using `git`. This ensures that the changes we make are relative to `desdeo-mcdm`'s master branch, at least the version of it which was available at the time of cloning it. Let us create a new branch now for our featrue X named `feature-X`:

```
$ git branch feature-X
$ git checkout feature-X
```

We are now ready to start implementing our changes to the package. Frequent `committing` is encouraged.

Suppose that we are now done impelenting our feature X. We now wish it was included to the master branch of `desdeo-mcdm`. To do so, we need to first switch back to the master branch

```
$ git checkout master
```

Then we need to make sure the master branch is up-to-date with the upstream version of the branch by issuing a pull. (If a `forked` repository is being used, the repository must first be synchronized with the upstream repository.)

```
$ git pull
```

Lastly, we will have to merge our `feature-X` branch containing our changes with the master branch

```
$ git merge feature-X
```

If all went well, we should now be ready to issue a `pull request`. However, if any conflicts emerge during the merging of the branches, these conflicts should be addressed before making a `pull request`. When the merge is free of conflicts, a `pull request` can be issued on GitHub or from the terminal. A maintainer of the repository will then review your changes and either accept them into the upstream or request revisions to be made before the new code can be accepted.

1.4.3 Documentation

Introduction

To build the documentation for the DESDEO framework and its various modules, Sphinx is used. Sphinx offers excellent utilities for automatically generating API documentation based on existing documentation located in source code, and for adding custom content.

Automatically generated documentation and custom content is specified as reStructuredText. ReStructuredText is a markup language just like Markdown or html, but offers the possibility to extend the language for specific domains. The file extention `.rst` is used for files containing reStructuredText content. Sphinx can then be used to generate documentation in various formats, such as html and pdf, based on content provided as reStructuredText.

Resources to get started with Sphinx

The official documentation offers a good guide for getting started. It is advised to read through the guide before contributing to the documentation. After reading the guide, the reader is encouraged to check out the contents of the source file used to generate the current page. The source file can be accessed by going to the top of this page and following the *Edit on GitHub*-link. It is also advised to check out the content of the *docs* file found in the main repository of the DESDEO framework. After checking the source file used to generate this page, the user should be familiar with at least basic sectioning, hyperlinks, code blocks, note blocks, and lists.

Other useful resources include:

- Official documentation for reStructuredText.
- ReStructuredText syntax cheatsheet.
- A conference talk about Sphinx given during PyCon 2018. (YouTube has also many other videos on Sphinx as well)
- A more through tutorial written by the matplotlib developers on how to achieve a documentation similar to theirs.

Extensions

In the DESDEO framework, some Sphinx extensions are used to faciliate automatic documentation generation. At least the following extensions are used:

Included in Sphinx:

- Sphinx.ext.autodoc for automatically generating documentation based on docstrings.
- Sphinx.ext.napoleon for parsing the Google styled docstrings.
- Sphinx.ext.viewcode for accessing the documented source code from the documentation itself.

User provided extensions:

- Sphinx-autodoc-typehints for better type hints.
- automodapi for even better automatic API documentation generation.
- nbsphinx for converting Python notebooks into rst pages.

Building and testing the documentation

If Sphinx has been setup following the official quick guide, the documentation can be build by running the commnad

```
make html
```

in the root direcotry containing the documentation. This will produce documentation in an html format residing in the `_build` folder in the documentation's root directory. To view the documentation built, use any web browser. For example, with Firefox, this is achieved by issuing the command

```
firefox _build/html/index.html
```

Note: The directory `_build` generated by `sphinx-quickstart` should not be under version control.

Deployment

Note: Most of the content in this section is relevant only when setting up the documentation for the first time for a module.

The documentation for each of the DESDEO modules is hosted on readthedocs.org. For the documentation to be build correctly, a YAML congiguration file named `.readthedocs.yml` should be present in the root directory of the project (not the root directory of the documentation!) A minimal configarion file could look like this:

```
# Required
version: 2

# Build documentation in the docs/ directory with Sphinx
Sphinx:
configuration: docs/conf.py

# Optionally set the version of Python and requirements required to build your docs
python:
version: 3.7
install:
    - requirements: docs/requirements.txt
```

Especially the locations of the configuration files `docs/conf.py` and `docs/requirements.txt` are important to enable readthedocs to correctly build the documentation.

Note: The requirements file should contain the requirements for **building the documentation**. It does not necessarely need to contain all the requirements of the module the documentation is being build for. However, for building the documentation for some of the modules, like `desdeo-mcdm` for example, the whole module needs to be installed for Sphinx to be able to compile the documentation. In that case, having the project's whole requirements in the requirements file pointed at in `.readthedocs.yml` is justified.

If a `requirements.txt` if required, but *poetry* is used to manage dependencies, then the command

```
poetry export --dev -f requirements.txt > requirements.txt
```

can be used to generate a requirements file.

For more configuration options, go here. The whole documentation for readthedocs can be found here.

Caveats

Some common caveats with Sphinx:

- The intendation Sphinx expects in the reStructuredText files is **three spaces** to specify the scope of the *options* and *content* of a *directive*. Options should follow the directive immediately on the following line, one option per line, and the content should be separated by one blank line from the options (if no options are provided, the blank line should be between the directive and the contents). For example, the following is correct:

```
.. toctree::
   :maxdepht: 2

   content
   morecontent
```

The following, however, is **incorrect**:

```
.. toctree::
   :maxdepht: 2
   content
   morecontent
```

- If the contents of an item in a list span more than one line, the lines following the first line should have their indentation starting at the same level as the content on the first line. I.e.:

```
- This is the first line
  this is the second line
  this is the third line
  notice the indentation
```

1.5 Glossary

decision maker, DM A domain expert with adequate expertise related to an optimization problem able to provide preference information.

Pareto optimal solution A feasible solution to a multiobjective optimization problem which corresponds to an objective vector that cannot be exchanged for any other objective vector resulting from some other feasible solution wihtout having to make a trade-off in at least one objective value.

Pareto front The set of Pareto optimal solutions.

nadir (point) The worst possible objective values of a Pareto front.

ideal (point) The best possible objective values of a Pareto front.

reference point Desirable aspiration levels for each objective function.

INDEX

D

decision maker, DM, 13

I

ideal (*point*), 13

N

nadir (*point*), 13

P

Pareto front, 13
Pareto optimal solution, 13

R

reference point, 13