conv$_{o}ptdocumentation$

Release 0.0.12

Karr Lab

Oct 04, 2020

Contents

`conv_opt` is a high-level Python package for solving linear and quadratic optimization problems using multiple open-source and commercials solvers including Cbc, CVXOPT, FICO XPRESS Optimizer, GLPK, Gurobi, IBM CPLEX, MINOS, MOSEK Optimizer, quadprog, SciPy, and SoPlex.

CHAPTER 1

Contents

1.1 Installation

1.1.1 Requirements

First, install Python and pip. The following command illustrates how to install Python and pip on Ubuntu Linux:

```
apt-get install python python-pip
```

1.1.2 Optional requirements

Second, optionally install additional solvers. Please see our detailed instructions.

- Cbc and CyLP
- FICO XPRESS
- Gurobi
- IBM CPLEX
- MINOS
- SoPlex
- MOSEK Optimizer

1.1.3 Installing this package

Third, install this package. The latest release of this package can be installed from PyPI using this command:

```
pip install conv_opt
```

Alternatively, the latest version of this package can be installed from GitHub using this command:

```
pip install git+https://github.com/KarrLab/conv_opt.git#egg=conv_opt
```

Support for the optional solvers can be installed using the following options:

```
pip install conv_opt[cbc,cplex,gurobi,minos,mosek,soplex,xpress]
```

1.2 Features

1.2.1 Problems

This package supports two types of optimization problems:

- Linear problems

$$\begin{aligned} &\text{Maximize } c'x \\ &\text{Subject to} \\ &\quad Ax = b \\ &\quad x_l \leq x \leq x_u \end{aligned}$$

- Quadratic problems

$$\begin{aligned} &\text{Maximize } x'Qx + c'x \\ &\text{Subject to} \\ &\quad Ax = b \\ &\quad x_l \leq x \leq x_u \end{aligned}$$

1.2.2 Objectives

This package supports two types of objectives:

- linear
- quadratic

1.2.3 Variable types

This package supports five types of variables:

- binary
- integer
- continuous
- semi-integer
- semi-continuous
- partially-integer

1.2.4 Constraints

This package supports one type of constraint:

- linear

1.2.5 Solvers

This package supports several solvers:

- Open-source
  - Cbc via CyLP
  - CVXOPT via CVXPY
  - GLPK via optlang
  - quadprog
  - SciPy
- Non-commercial with free academic licenses
  - MINOS via solveME
  - SoPlex via soplex_cython
- Commercial with free academic licenses
  - FICO XPRESS
  - IBM CPLEX
  - Gurobi
  - MOSEL Optimizer

However, as described below, some of the solvers only support some of these features.

Objective types

- Cbc: only supports linear objectives
- GLPK: only supports linear objectives
- quadprog: only supports quadratic objectives

Variable types

- Cbc: only supports binary, integer, and continuous variables
- CVXOPT: only supports continuous variables
- FICO XPRESS: supports all variable types
- GLPK: only supports binary, integer, and continuous variables
- Gurobi: doesn't support partially integer variables
- IBM CPLEX: doesn't support partially integer variables
- MINOS: only supports continuous variables

- MOSEK Optimizer: only supports binary, integer, and continuous variables
- quadprog: only supports continuous variables
- Scipy: only supports continuous variables

Constraint types

Cbc, GLPK, and quadprog only support linear constraints. Only SciPy's COBLYA and SLSQP method support constraints.

Python versions

- CPLEX supports Python 2.7, 3.5, and 3.6

Licensing

- FICO XPRESS: licenses are tied to machines, or a license server must be used
- Gurobi: licenses are tied to machines, or a license server must be used
- IBM CPLEX: No license file or activation is needed
- MINOS: free academic licenses can be obtained from Michael Saunders at Stanford
- MOSEL Optimizer: license files can be used on multiple machines
- SoPlex: free academic licenses can be automatically obtained from the SoPlex website

1.3 Tutorial

1. Import the `conv_opt` package:

```
import conv_opt
```

2. Create a model and, optionally, name the model:

```
model = conv_opt.Model(name='model')
```

   The default name is `None`.

3. Create variables; optionally set their names, types, and upper and lower bounds; and add them to models:

```
var_1 = conv_opt.Variable(name='var_1', type=conv_opt.VariableType.
↪continuous, lower_bound=0, upper_bound=1)
model.variables.append(var_1)

var_2 = conv_opt.Variable(name='var_2', type=conv_opt.VariableType.
↪continuous, lower_bound=0, upper_bound=1)
model.variables.append(var_2)
```

   The default name is `None`.

   The `type` argument must be one of the following values. The default type is continuous.

   - `conv_opt.VariableType.binary`

- `conv_opt.VariableType.integer`
- `conv_opt.VariableType.continuous`
- `conv_opt.VariableType.semi_integer`
- `conv_opt.VariableType.semi_continuous`
- `conv_opt.VariableType.partially_integer`

The default upper and lower bounds are `None`.

4. Set the objective expression and direction:

```
model.objective_terms = [
    conv_opt.LinearTerm(var_1, 1.),
    conv_opt.QuadraticTerm(var_2, var_2, 1.),
]
model.objective_direction = conv_opt.ObjectiveDirection.maximize
```

`objective_terms` should be a list of linear (`conv_opt.LinearTerm`) and quadratic terms (`conv_opt.QuadraticTerm`). The arguments to the constructors of these class are the variables involved in the term and coefficient for the term.

`objective_direction` can be either of following values. The default direction is minimize.

- `conv_opt.ObjectiveDirection.maximize`
- `conv_opt.ObjectiveDirection.minimize`

5. Create constraints; optionally set their names and upper and lower bounds; and add them to models:

```
contraint_1 = conv_opt.Constraint([
    conv_opt.LinearTerm(var_1, 1),
    conv_opt.LinearTerm(var_2, -1),
], name='contraint_1', upper_bound=0, lower_bound=0)
model.constraints.append(contraint_1)
```

The first argument should be a list of linear terms.

The default name and upper and lower bounds are `None`.

6. Configure the options for solving the model:

```
options = conv_opt.SolveOptions(solver=conv_opt.Solver.cplex,␣
↪presolve=Presolve.off, verbosity=False)
```

The `solver` argument allows you to select a specific solver. The argument must be one of the following values. The default solver is GLPK.

- `conv_opt.Solver.cbc`
- `conv_opt.Solver.cplex`
- `conv_opt.Solver.cvxopt`
- `conv_opt.Solver.glpk`
- `conv_opt.Solver.gurobi`
- `conv_opt.Solver.minos`
- `conv_opt.Solver.mosek`
- `conv_opt.Solver.quadprog`

- conv_opt.Solver.scipy
- conv_opt.Solver.soplex
- conv_opt.Solver.xpress

The presolve aregument must be one of the following values. The default value is off.

- conv_opt.Presolve.auto
- conv_opt.Presolve.on
- conv_opt.Presolve.off

Please see the API documentation for information about additional options.

7. Solve the model:

```
result = model.solve(options)
if result.status_code != conv_opt.StatusCode.optimal:
    raise Exception(result.status_message)
value = result.value
primals = result.primals
```

The result will be an instance of conv_opt.Result. The attributes of this class include:

- status_code
- status_message
- value
- primals
- reduced_costs
- duals

status_code will be an instance of the conv_opt.StatusCode enumerated type.

8. Get diagnostic information about the model:

```
stats = model.get_stats()
```

9. Convert the model to the lower level API of one of the solvers:

```
options = conv_opt.SolveOptions(solver=conv_opt.Solver.cplex)
cplex_model = model.convert(options)
```

10. Export the model to a file:

```
filename='/path/to/file.ext'
model.export(filename)
```

conv_opt supports the following extensions:

- alp
- cbf
- dpe: dual perturbed model
- dua: dual
- jtask: Jtask format
- lp

- mps
- opf
- ppe: perturbed model
- rew: model with generic names in mps format
- rlp: model with generic names in lp format
- sav
- task: Task format
- xml: OSiL

1.4 Testing

The package can be tested by running these commands:

```
pip install pytest
python -m pytest tests
```

We tested the package with the following package versions:

- OS: Mint Linux 18.1 (based on Ubuntu 16.04)
- Python: 2.7.14, 3.6.3
- NumPy: 1.13.3
- SymPy: 1.1.1
- Solvers and solver interfaces:
  - Cbc: 2.8.5, CyLP: 0.7.4
  - CVXOPT: 1.1.9, CVXPY: 0.4.11
  - GLPK: 4.57, swiglpk: 1.4.4, optlang: 1.2.5
  - FICO XPRESS: 8.4.0
  - Gurobi: 7.5.1
  - IBM CPLEX: 12.7.1.0
  - MINOS: 5.6
  - MOSEK Optimizer: 8.1.33
  - quadprog: 0.1.6
  - SciPy: 1.0.0
  - SoPlex: 3.1.1

1.5 About

1.5.1 License

The software is released under the MIT license

```
The MIT License (MIT)

Copyright (c) 2017 Karr Lab

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

1.5.2 Development team

This package was developed by the Karr Lab at the Icahn School of Medicine at Mount Sinai in New York, USA.

1.5.3 Acknowledgements

1.5.4 Questions and comments

Please contact the Karr Lab with any questions or comments.

1.6 References

- FICO XPRESS Optimizer Python interface: see `/path/to/xpress/docs/solver/optimizer/python/HTML/python-interface.html`
- Gurobi Python API Overview
- IBM CPLEX Python API Reference Manual
- MOSEK Optimizer API for Python