Minimod: A Mixed Integer Solver for Spatio-Temporal Optimal Nutrition Intervention

Aleksandr Michuda

`minimod` is a solver written in python that solves the optimal nutrition intervention over space and time. It uses `mip` a mixed integer solver that uses the `CBC` solver, a fast, extensible and open-source linear solver. `minimod` comes with some constraints baked in and the underlying `mip` model is exposed and can be used for adding custom constraints if needed.

The problem that `minimod` solves initially can be written like so:

$$\begin{aligned}
&\text{Max} \sum_k \sum_t Y_{k,t} \sum_j \frac{EfCvg_{kj,t}}{(1+r)^t} \\
&+ \sum_k \sum_j \sum_t X_{k,j,t} \frac{EfCvg_{k,j,t}}{(1+r)^t} \\
&s.t. \\
&\sum_k \sum_t Y_{k,t} \sum_j \frac{\text{TC}_{k,j,t}}{(1+i)^t} \\
&+ \sum_k \sum_j \sum_t X_{k,j,t} \frac{\text{TC}_{k,j,t}}{(1+i)^t} \leq \text{TF}
\end{aligned}$$

which essentially maximizes discounted coverage given a budget constraint. This problem can give solutions that are both time and space specific. The dual problem minimizes discounted costs given a minimum coverage constraint.

Quickstart

In order to run the model, we need to first import `minimod` and `pandas` in order to later load in our data:

```
1 import minimod as mm
2 import pandas as pd
```

```
Using Python-MIP package version 1.6.8
```

Now we load in data and instantiate the model we want (`BenefitSolver` or `CostSolver`).

The solvers take several arguments:

- data -> a pandas dataframe of benefits and cost data. This data needs to be of a certain form, mainly a long form of data.
  - Default: None

k	j	t	benefits	costs
maize	north	0	100	10
maize	south	0	50	20
maize	east	0	30	30
maize	west	0	20	40

- `intervention_col` -> the name of the intervention variable
  - Default: 'intervention'
- `space_col` -> the name of the spatial variable
  - Default: 'space'

- `time_col` -> the name of the time variable
  - Default: 'time'
- `benefit_col` -> the name of the benefit variable
  - Default: 'benefit'
- `cost_col` -> the name of the cost variable
  - Default: 'costs'
- `interest_rate_cost` -> the interest rate on costs
  - Default: 0.0 -`interest_rate_benefit` -> the interest rate on benefits
  - Default: 0.03
- `va_weight` -> The weight to give the benefits during intervention
  - Default: 1.0

For `BenefitSolver`, we also have:

- total_funds -> Maximum Budget
  - Default: 35821703

And for `CostSolver`:

- minimum_benefit -> The Minimum benefit constraint
  - Default: 15958220

Now we load the data and instantiate the solver:

```
4 df = pd.read_csv('../../examples/data/processed/example1.csv')
5
6 c = mm.CostSolver(data = df)
```

We then fit the model:

```
c.fit()
```

```
Loading MIP Model with:
               Solver = CBC
               Method = MIN,


[Note]: Optimal Solution Found
```

The optimal interventions are stored in an attribute available after fitting:

```
opt_df = c.opt_df
```

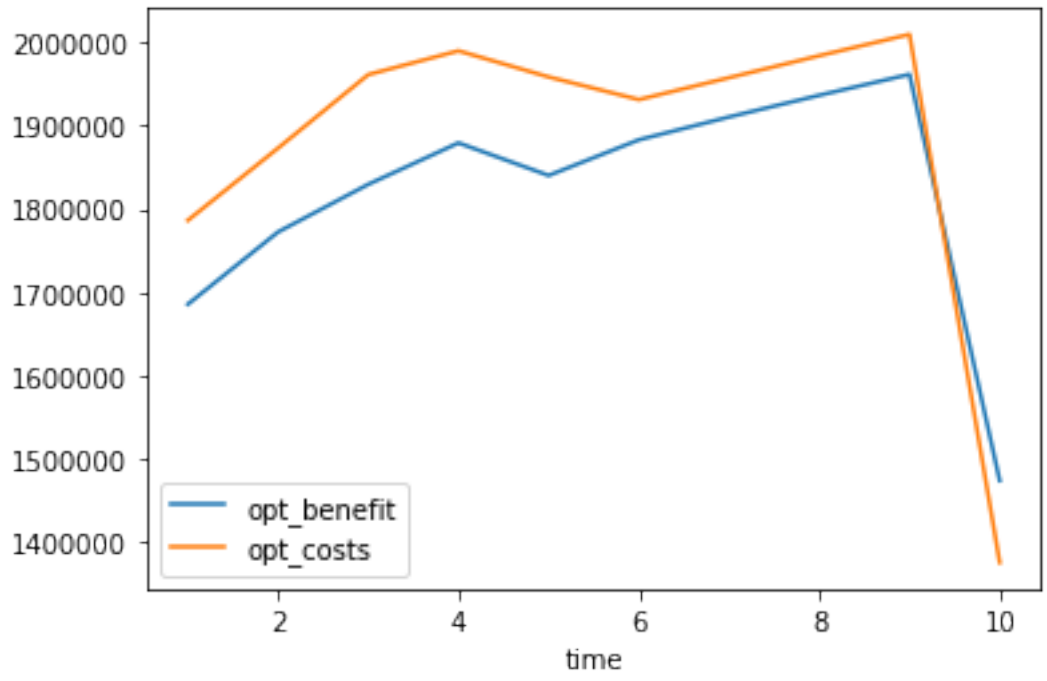
We can graph the costs and benefits through time from this:

```
%matplotlib inline

import matplotlib.pyplot as plt

opt_df.groupby('time').sum()[['opt_benefit', 'opt_costs']].plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f77bf965070>
```



Then we can generate a report

```
18 c.report()
```

```
                Optimized Scenario with:


                Method: MIN

                Discount Factor on Costs: 1.0

                Discount Factor on Benefits: 0.970873786407767


+------------------------------+

+------------------------------+
```

Total Costs and Coverage by Year

+------------------------------+

time	opt_vals	opt_benefit	opt_costs
1	3	1.68539e+06	1.78616e+06
2	3	1.77221e+06	1.87296e+06
3	3	1.82944e+06	1.96123e+06
4	3	1.87922e+06	1.98977e+06
5	3	1.8401e+06	1.95842e+06
6	3	1.88294e+06	1.93123e+06
7	3	1.91036e+06	1.95754e+06
8	3	1.93643e+06	1.98395e+06
9	3	1.96151e+06	2.00954e+06
10	3	1.47362e+06	1.37522e+06

Total Cost

+------------------------------+

18826018.698786113

Total Coverage

+------------------------------+

18171231.83213076

Cost per Coverage

+------------------------------+

1.0360342585854607