Data and code for the article:

A probabilistic framework based on the gradient descent algorithm for multi-objective land use optimization

Data

Here, we release two datasets of land use optimization:

- ✓ SimExp
- ✓ GMCase

1. **SimExp**
   Include:
   LU.tif: Land use raster (refer to Figure 1(a))
   Ws: folder includes the parameters described in Appendix C.
2. **GMCase**
   Include:
   LU.tif: Land use raster (refer to Figure 2)
   Ws: folder includes the parameters described in Appendix C.

Code

1. **Run environment**
   Require module:
   - ✓ gdal 3.3.1
   - ✓ numpy 1.20.3
   - ✓ pandas 1.5.0
   - ✓ tensorflow-gpu 2.9.1 (or tensorflow with gpu)
   - ✓ geatpy 2.7.0 (optional, for GA experiment)

   We run the code on the platform of Windows 10, using Python 3.9.5. For GPU acceleration, we use an NVIDIA GTX 2080Ti graphic card with CUDA 11.3. The versions of the modules are for reference, different versions could also support the code, but we do not have any test results for other environments.
2. **File description**
   - ➢ loader.py: A script to load Landuse(LU) Data.
   - ➢ model.py: Implementation of MOLU model for GA and pMOLU model for GDA.
   - ➢ objs_sim.py: Objective functions for the simulation data experiment.
   - ➢ objs.py: Objective functions for Guangming case study.
   - ➢ Opt_GMcaseGDA.py: The main code for optimization using GDA in Guangming case study.
   - ➢ Opt_SimExpGA.py: The main code for optimization using GA in simulation data experiment.
   - ➢ Opt_SimExpGDA.py The main code for optimization using GDA in simulation data experiment.

3. The simulation data experiment

The objective for the experiment is implemented in the script `objs_sim.py`, the model is implemented in `model.py` with the support of the script `loader.py`. Please ensure the 3 scripts and the data folder `SimExp` are in the root directory of the environment when running the following script:

3.1. run the script `Opt_SimExpGA.py` to conduct GA optimization, you can change the parameter `rsam` in line 60 to run the experiment of different image sizes.

```
problem = MyProblem(objs, LUname='LU', z_LU=4, rsam=8, path='SimExp')
```

3.2. run the script `Opt_SimExpGDA.py` to conduct GDA optimization, you can change the parameter `rsam` in line 17 to run the experiment of different image sizes.

```
Model = GDAmodel(objs, LUname='LU', thC=None, z_LU=4, rsam=8, mask=N
one, path='SimExp', GPU=0, cmap=cmap)
```

3.3 The result will be written in the folder `GAResult{image size}` at SimExp. For the experiment in the article, we run these two scripts with 4 different `rsam` parameters (1, 2, 4, 8) to obtain the results of GA/GDA:

rsam=1 => image size=256*256;	rsam=2 => image size=128*128
rsam=4 => image size=64*64;	rsam=8 => image size=32*32

We also summarize the running result in `./SimExp/RAtime.xlsx`

4. The case study

The objective for the experiment is implemented in the script `objs.py`, the model is implemented in `model.py` with the support of the script `loader.py`. Please ensure the 3 scripts and data folder `GMCase` are in the root directory of the environment when running the case:

Run the notebook `Opt_GMcaseGDA.ipynb` to obtain the optimized scenarios (Figure 4).

Run the script `Opt_GMcaseGDA.py` to conduct GDA optimization for the case study in Guangming. The default code will generate the equal weight result, you can change the parameter of preference in line 18 to obtain the scenarios with different preferences.

```
Model.perference()  # default
Model.perference(scale=[1.05, 1, 1, 1, 1, 1, 1])
```

The preference vector represents the weight of the MOUF for the 7 objectives correspondingly: Economic benefit; Ecological benefit; Housing capacity; Employment capacity; Livability; Employment–housing divergence; Compatibility

In the article, we set the weight of 1.05 for the preferred objective.

You can also change the following parameter:

- `thC` in line 10: The threshold of CC to stop iteration ($\tau^{(CN)}$):

```
Model = GDAmodel(objs, LUname='LU', thC=.1, z_LU=12, rsam=4, mask=15
, path='GMCase')
```

- Parameters of `init_train` line 13-15: evaluate the utopia point $f^{(o)*}$:

```
Model.init_train(500, lr=0.1)
```

- Iteration parameter in line 19: first parameter `n_iter` is the max iteration time, `thStop` is the switch for threshold, `nprint` controls the iteration interval of logs. Each log will plot the scenario and print objective values, and the scenario will be saved in `./GMCase/trainLU`.

```
Model.train(n_iter=10000, thStop=1, nprint=50)
```

In line 20, you can use `save` function to save the results. The normalized weight will be saved in a folder `./GMCase/inp`, the tensorflow model will save in `./GMCase`, the optimizing log will save as `./GMCase/train.csv`,, the optimized scenario will be saved as `./GMCase.LU_opt.tif`.

```
Model.save()
```

5. **Apply to other case studies**

   You can apply our method to other studies, what you need to do is prepare the data and write the objective functions (or just use our objective functions).

   1. Prepared the data and weight, and put them into a folder just like the sample data.
   2. Write a python script, `objs.py` is a script for your reference.
   3. Write the main script to conduct optimization, refer to `Opt_GMcaseGDA.py`.