Running Illustrative Data Using the TiLearn Library

September 29, 2024

1 Introduction to the TiLearn Library

TiLearn /ta l rn/ is a combination of "Time" and "Machine Learning." The library was built with the initial goal of automating the process of evaluating job weights (w_j) and improving the branching process (Learning to Branch) through machine learning methods.

The source code has been uploaded to GitHub, and the link is provided below.

GitHub link: https://github.com/Bancie/TiLearn

The library has also been published on PyPI and can be easily downloaded and used.

Link: https://pypi.org/project/TiLearn/

Additionally, the document site is designed to provide usage guides, instructions on how to run the library, and how it works.

Documentation link: https://bancie.github.io/TiLearn/

The library currently supports the following features:

- A function for running the EDD algorithm.
- A function for running the WSPT algorithm.
- A function for solving the problem $1||\sum C_j$.
- A function for solving the problem $1||\sum w_j C_j$.
- A function for solving the problem $1|prec|\sum C_j$.
- A function for solving the problem $1|prec|\sum w_j C_j$.
- A function for handling a mixture of both $1|prec|\sum w_j C_j$ and $1||\sum w_j C_j$ problems.

Detailed guides on how to use these functions, including parameters and the meaning of each parameter, will be updated on the documentation site.

To use the library, it can be installed with the following command

[ ]:
```
%pip install tilearn
```

Some supporting libraries for analysis

[ ]:
```
%pip install pandas
%pip install ipython
```

```
%pip install jinja2
```

Declare the main library

[4]:
```
import tilearn as tl
from tilearn import _plat as pl
```

Some supporting libraries for data analysis

[5]:
```
import csv
import pandas as pd
from IPython.display import display
import matplotlib.pyplot as plt
```

2 Running Illustrative Data Using TiLearn

Suppose we need to process the following three lists

- List 1:

[6]:
```
list1 = '/Users/chibangnguyen/Documents/thuhoach.nckh/tilearn/data/list1.csv'
display(pd.read_csv(list1))
```

	Name	p	r	d	w
0	Job 1	4	0	100	0.65
1	Job 2	1	0	100	0.84
2	Job 3	3	0	100	0.46
3	Job 4	3	0	100	0.79
4	Job 5	1	0	100	0.17
5	Job 6	3	0	100	0.50
6	Job 7	4	0	100	0.95
7	Job 8	2	0	100	0.14
8	Job 9	5	0	100	0.52
9	Job 10	2	0	100	0.40
10	Job 11	4	0	100	0.55
11	Job 12	1	0	100	0.39
12	Job 13	2	0	100	0.57
13	Job 14	1	0	100	0.90
14	Job 15	1	0	100	0.22

- List 2:

[7]:
```
list2 = '/Users/chibangnguyen/Documents/thuhoach.nckh/tilearn/data/list2.csv'
display(pd.read_csv(list2))
```

	Name	p	r	d	w
0	Job 16	4	0	100	0.70
1	Job 17	3	0	100	0.95
2	Job 18	4	0	100	0.49
3	Job 19	1	0	100	0.13

```
4  Job 20  5  0  100  0.94
5  Job 21  1  0  100  0.57
6  Job 22  4  0  100  0.47
```

- List 3:

[8]:
```
list3 = '/Users/chibangnguyen/Documents/thuhoach.nckh/tilearn/data/list3.csv'
display(pd.read_csv(list3))
```

```
    Name  p  r    d     w
0  Job 23  4  0  100  0.24
1  Job 24  1  0  100  0.54
2  Job 25  2  0  100  0.81
3  Job 26  2  0  100  0.41
4  Job 27  5  0  100  0.22
5  Job 28  2  0  100  0.29
6  Job 29  5  0  100  0.65
7  Job 30  4  0  100  0.69
```

In List 1, the jobs are independent and have no precedence, meaning they follow the structure of $1||\sum w_j C_j$. However, Lists 2 and 3 follow a structure that includes precedence constraints, $1|prec|\sum w_j C_j$.

The folder structure is set up as follows

```
tilearn/
  main.py
  data/
      backup/
      list1.csv
      list2.csv
      list3.csv
```

Define the data paths

[9]:
```
original = '/Users/chibangnguyen/Documents/thuhoach.nckh/tilearn/data'
backup = '/Users/chibangnguyen/Documents/thuhoach.nckh/tilearn/data/backup'
```

Select the problem type for each list by setting `prec` as follows:

- `prec=0` for problems that do not require job precedence.
- `prec=1` for problems that require job precedence.

[10]:
```
lists = [
    pl.List(file_path=list1, prec=0),
    pl.List(file_path=list2, prec=1),
    pl.List(file_path=list3, prec=1),
]
```

Run the program with the following command

[11]:
```
schedule = tl.optimal_list(lists, original, backup)
print(schedule)
```

```
[['Job 14', 1.0, 0, 100, 0.9, 0.9], ['Job 2', 1.0, 0, 100, 0.84, 0.84], ['Job
12', 1.0, 0, 100, 0.39, 0.39], ['Job 13', 2.0, 0, 100, 0.57, 0.285], ['Job 4',
3.0, 0, 100, 0.79, 0.26333333333333336], ['Job 7', 4.0, 0, 100, 0.95, 0.2375],
['Job 16', 4.0, 0, 100, 0.7, 0.175], ['Job 17', 3.0, 0, 100, 0.95,
0.2357142857142857], ['Job 23', 4.0, 0, 100, 0.24, 0.06], ['Job 24', 1.0, 0,
100, 0.54, 0.156], ['Job 25', 2.0, 0, 100, 0.81, 0.22714285714285715], ['Job
15', 1.0, 0, 100, 0.22, 0.22], ['Job 26', 2.0, 0, 100, 0.41, 0.205], ['Job 10',
2.0, 0, 100, 0.4, 0.2], ['Job 18', 4.0, 0, 100, 0.49, 0.1225], ['Job 19', 1.0,
0, 100, 0.13, 0.124], ['Job 20', 5.0, 0, 100, 0.94, 0.156], ['Job 21', 1.0, 0,
100, 0.57, 0.19363636363636363], ['Job 5', 1.0, 0, 100, 0.17, 0.17], ['Job 6',
3.0, 0, 100, 0.5, 0.16666666666666666], ['Job 1', 4.0, 0, 100, 0.65, 0.1625],
['Job 3', 3.0, 0, 100, 0.46, 0.15333333333333335], ['Job 11', 4.0, 0, 100, 0.55,
0.1375], ['Job 22', 4.0, 0, 100, 0.47, 0.1175], ['Job 27', 5.0, 0, 100, 0.22,
0.044], ['Job 28', 2.0, 0, 100, 0.29, 0.07285714285714286], ['Job 29', 5.0, 0,
100, 0.65, 0.09666666666666668], ['Job 30', 4.0, 0, 100, 0.69, 0.115625], ['Job
9', 5.0, 0, 100, 0.52, 0.10400000000000001], ['Job 8', 2.0, 0, 100, 0.14, 0.07]]
```

To make the list more readable, you can use the following command

[12]:
```
header = ['Name','p','r','d','w','p-factor']
output = '/Users/chibangnguyen/Documents/thuhoach.nckh/tilearn/ouput.csv'
schedule = tl.optimal_list(lists, original, backup)
with open(output, 'w', newline='') as file:
    writer = csv.writer(file)
    writer.writerow(header)
    writer.writerows(schedule)
```

View the output data from the `output.csv` file:

[13]:
```
display(pd.read_csv('/Users/chibangnguyen/Documents/thuhoach.nckh/tilearn/ouput.
↪csv'))
```

	Name	p	r	d	w	p-factor
0	Job 14	1.0	0	100	0.90	0.900000
1	Job 2	1.0	0	100	0.84	0.840000
2	Job 12	1.0	0	100	0.39	0.390000
3	Job 13	2.0	0	100	0.57	0.285000
4	Job 4	3.0	0	100	0.79	0.263333
5	Job 7	4.0	0	100	0.95	0.237500
6	Job 16	4.0	0	100	0.70	0.175000
7	Job 17	3.0	0	100	0.95	0.235714
8	Job 23	4.0	0	100	0.24	0.060000
9	Job 24	1.0	0	100	0.54	0.156000
10	Job 25	2.0	0	100	0.81	0.227143
11	Job 15	1.0	0	100	0.22	0.220000
12	Job 26	2.0	0	100	0.41	0.205000

```
13   Job 10   2.0   0   100   0.40   0.200000
14   Job 18   4.0   0   100   0.49   0.122500
15   Job 19   1.0   0   100   0.13   0.124000
16   Job 20   5.0   0   100   0.94   0.156000
17   Job 21   1.0   0   100   0.57   0.193636
18    Job 5   1.0   0   100   0.17   0.170000
19    Job 6   3.0   0   100   0.50   0.166667
20    Job 1   4.0   0   100   0.65   0.162500
21    Job 3   3.0   0   100   0.46   0.153333
22   Job 11   4.0   0   100   0.55   0.137500
23   Job 22   4.0   0   100   0.47   0.117500
24   Job 27   5.0   0   100   0.22   0.044000
25   Job 28   2.0   0   100   0.29   0.072857
26   Job 29   5.0   0   100   0.65   0.096667
27   Job 30   4.0   0   100   0.69   0.115625
28    Job 9   5.0   0   100   0.52   0.104000
29    Job 8   2.0   0   100   0.14   0.070000
```

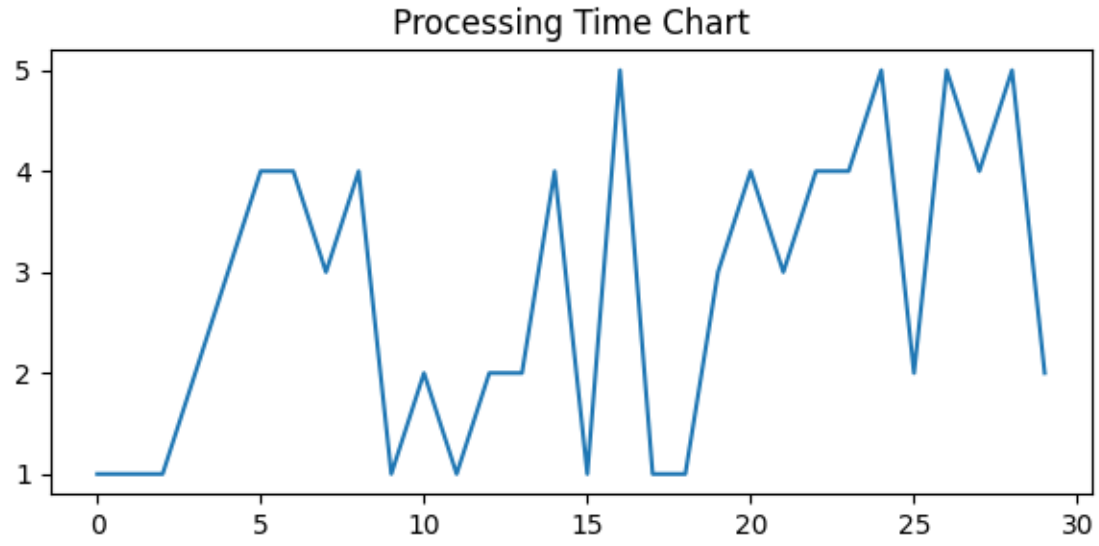
2.1 Data Visualization

We use the `pandas` library to help visualize the data.

[14]:
```
df = pd.read_csv(r'/Users/chibangnguyen/Documents/thuhoach.nckh/tilearn/ouput.
↪csv', engine='pyarrow')
```

Let's first look at the processing time chart

[15]:
```
plt.figure(figsize=(7, 3))
df['p'].plot(title='Processing Time Chart')
plt.show()
```



Based on the chart above, we can see that after optimizing the job list, the processing times for each job tend to increase from the first job to the last job.

Next, let's look at the job weight chart

[16]:
```
plt.figure(figsize=(7, 3))
df['w'].plot(title='Job Weight Chart')
plt.show()
```

From this chart, we can see that after optimizing the job list, the weights of each job tend to decrease from the first job to the last job.

From these insights, it is easy to observe that jobs with shorter processing times and higher priority weights are pushed to the top, while jobs with longer processing times and lower priorities are placed at the bottom.

3 Scheduling for Design Production Using TiLearn

3.1 Introduction to the Design Process

In the processing stage, studios often face the challenge of allocating resources and adjusting designs for multiple projects simultaneously. This is based on client priority, the complexity of each project, and external factors such as construction time. Applying scheduling techniques helps manage data more efficiently, plan better, and track progress while adjusting when necessary.

Typically, an interior design project goes through the following main stages:

- Concept development
- Space planning

- Design development
- Material selection
- Cost estimation
- Construction
- Installation
- Decoration
- Handover

Each phase needs to be completed before moving to the next. Therefore, the problem can be modeled as a scheduling problem with precedence constraints: $1|prec|\sum w_j C_j$, where tasks must follow priority and precedence rules.

Additionally, the studio may need to perform condition surveys for various projects, which do not require sequential completion. In this case, the problem can be modeled as $1||\sum w_j C_j$, where jobs can be completed in any order.

3.2 Running Illustrative Data

Suppose we have the following three projects: A, B, and C

Project A:

[17]:
```
project_A = '/Users/chibangnguyen/Documents/thuhoach.nckh/tilearn/data_project/
  ↪Project-A.csv'
display(pd.read_csv(project_A))
```

	name	p	r	d	w
0	Project A - Concept development	4	0	200	0.85
1	Project A - Space planning	3	0	200	0.80
2	Project A - Design development	5	0	200	0.75
3	Project A - Material selection	2	0	200	0.70
4	Project A - Cost estimation	2	0	200	0.60
5	Project A - Construction	10	0	200	0.95
6	Project A - Installation	4	0	200	0.70
7	Project A - Decoration	3	0	200	0.85
8	Project A - Handover	1	0	200	1.00

Project B:

[18]:
```
project_B = '/Users/chibangnguyen/Documents/thuhoach.nckh/tilearn/data_project/
  ↪Project-B.csv'
display(pd.read_csv(project_B))
```

	name	p	r	d	w
0	Project B - Concept development	8	0	200	0.85
1	Project B - Space planning	7	0	200	0.80
2	Project B - Design development	20	0	200	0.95
3	Project B - Material selection	3	0	200	0.70

```
4      Project B - Cost estimation   2  0  200  0.60
5         Project B - Construction  40  0  200  0.95
6         Project B - Installation  15  0  200  0.70
7           Project B - Decoration   9  0  200  0.85
8             Project B - Handover   1  0  200  1.00
```

Project C:

[19]:
```
project_C = '/Users/chibangnguyen/Documents/thuhoach.nckh/tilearn/data_project/
  ↪Project-C.csv'
display(pd.read_csv(project_C))
```

```
                              name   p  r    d     w
0  Project C - Concept development   5  0  200  0.85
1       Project C - Space planning   2  0  200  0.80
2   Project C - Design development   4  0  200  0.75
3   Project C - Material selection   6  0  200  0.70
4      Project C - Cost estimation   1  0  200  0.60
5         Project C - Construction  15  0  200  0.95
6         Project C - Installation   4  0  200  0.70
7           Project C - Decoration   4  0  200  0.85
8             Project C - Handover   1  0  200  1.00
```

The list of locations requiring a survey:

[20]:
```
survey = '/Users/chibangnguyen/Documents/thuhoach.nckh/tilearn/data_project/
  ↪Survey.csv'
display(pd.read_csv(survey))
```

```
                name   p  r    d     w
0  Survey Location 1   2  0  200  0.33
1  Survey Location 2   6  0  200  0.67
2  Survey Location 3   5  0  200  0.45
3  Survey Location 4  15  0  200  0.85
4  Survey Location 5   4  0  200  0.10
5  Survey Location 6   4  0  200  0.60
6  Survey Location 7  20  0  200  0.70
```

The folder structure is set up as follows:

```
tilearn/
  main.py
  data_project/
      backup/
      Project-A.csv
      Project-B.csv
      Project-C.csv
      Survey.csv
```

Define the data paths

[21]:
```
original_project = '/Users/chibangnguyen/Documents/thuhoach.nckh/tilearn/
 ↪data_project'
backup_project = '/Users/chibangnguyen/Documents/thuhoach.nckh/tilearn/
 ↪data_project/backup'
```

The survey list does not require sequential completion, so `prec=0`. However, for projects A, B, and C, jobs require sequential completion, so `prec=1` for all of them.

[22]:
```
lists_project = [
    pl.List(file_path=survey, prec=0),
    pl.List(file_path=project_A, prec=1),
    pl.List(file_path=project_B, prec=1),
    pl.List(file_path=project_C, prec=1),
]
```

Run the program with the following command

[23]:
```
header = ['Name','p','r','d','w','p-factor']
output = '/Users/chibangnguyen/Documents/thuhoach.nckh/tilearn/ouput_project.
 ↪csv'
schedule = tl.optimal_list(lists_project, original_project, backup_project)
with open(output, 'w', newline='') as file:
    writer = csv.writer(file)
    writer.writerow(header)
    writer.writerows(schedule)
```

You can view the output data from the `output_project.csv` file

[24]:
```
display(pd.read_csv('/Users/chibangnguyen/Documents/thuhoach.nckh/tilearn/
 ↪ouput_project.csv'))
```

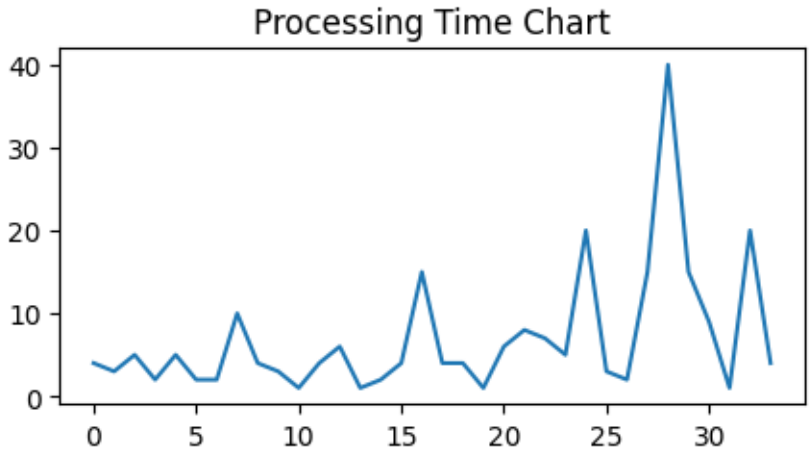
	Name	p	r	d	w	p-factor
0	Project A - Concept development	4.0	0	200	0.85	0.212500
1	Project A - Space planning	3.0	0	200	0.80	0.235714
2	Project C - Concept development	5.0	0	200	0.85	0.170000
3	Project C - Space planning	2.0	0	200	0.80	0.235714
4	Project A - Design development	5.0	0	200	0.75	0.150000
5	Project A - Material selection	2.0	0	200	0.70	0.207143
6	Project A - Cost estimation	2.0	0	200	0.60	0.227778
7	Project A - Construction	10.0	0	200	0.95	0.095000
8	Project A - Installation	4.0	0	200	0.70	0.117857
9	Project A - Decoration	3.0	0	200	0.85	0.147059
10	Project A - Handover	1.0	0	200	1.00	0.194444
11	Project C - Design development	4.0	0	200	0.75	0.187500
12	Project C - Material selection	6.0	0	200	0.70	0.116667
13	Project C - Cost estimation	1.0	0	200	0.60	0.185714
14	Survey Location 1	2.0	0	200	0.33	0.165000
15	Survey Location 6	4.0	0	200	0.60	0.150000
16	Project C - Construction	15.0	0	200	0.95	0.063333

17	Project C - Installation	4.0	0	200	0.70	0.086842
18	Project C - Decoration	4.0	0	200	0.85	0.108696
19	Project C - Handover	1.0	0	200	1.00	0.145833
20	Survey Location 2	6.0	0	200	0.67	0.111667
21	Project B - Concept development	8.0	0	200	0.85	0.106250
22	Project B - Space planning	7.0	0	200	0.80	0.110000
23	Survey Location 3	5.0	0	200	0.45	0.090000
24	Project B - Design development	20.0	0	200	0.95	0.047500
25	Project B - Material selection	3.0	0	200	0.70	0.071739
26	Project B - Cost estimation	2.0	0	200	0.60	0.090000
27	Survey Location 4	15.0	0	200	0.85	0.056667
28	Project B - Construction	40.0	0	200	0.95	0.023750
29	Project B - Installation	15.0	0	200	0.70	0.030000
30	Project B - Decoration	9.0	0	200	0.85	0.039062
31	Project B - Handover	1.0	0	200	1.00	0.053846
32	Survey Location 7	20.0	0	200	0.70	0.035000
33	Survey Location 5	4.0	0	200	0.10	0.025000

3.3 Data Visualization

We use the pandas library to help visualize the data for the design processing task

[34]:
```
df_project = pd.read_csv(r'/Users/chibangnguyen/Documents/thuhoach.nckh/tilearn/
  ↪ouput_project.csv', engine='pyarrow')
plt.figure(figsize=(5, 2.4))
df_project['p'].plot(title='Processing Time Chart')
plt.show()
```



[35]:
```
plt.figure(figsize=(5, 2.4))
df_project['w'].plot(title='Job Weight Chart')
plt.show()
```

From the two charts above, we can see that jobs with shorter processing times and higher weights (priority levels) are pushed to the top, while jobs with longer processing times and lower priorities are placed at the bottom, thereby optimizing the project process efficiently.