Moscow Metro Optimization

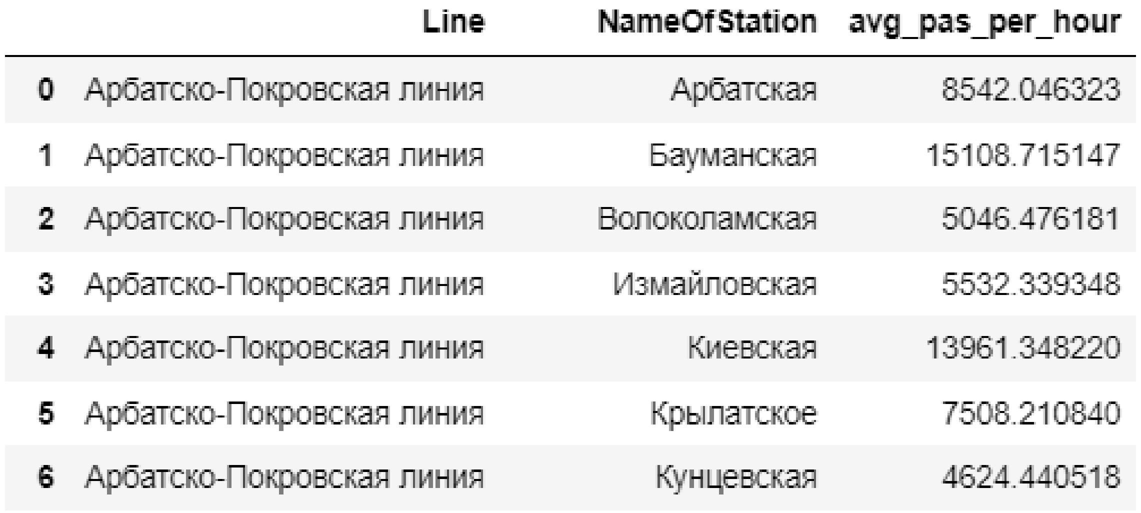
by BAE’24 students:
Sergej Ishmuratov and Andrey Strukov

The main ideas of the project:

- To get and clear the data of the current distribution of trains by lines and of the passenger traffic;
- Look for opportunities to minimize the overall average waiting time by moving a train from one line to another;
- Compare the results with the current distribution

The data overview

After some work with data, we got a dataset with average number of passengers on each station per hour:



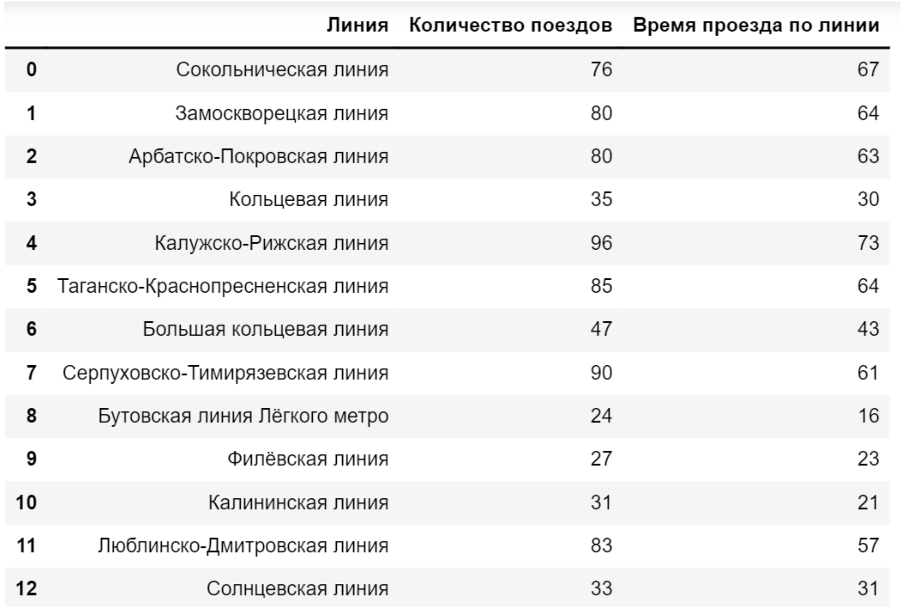
	Line	NameOfStation	avg_pas_per_hour
0	Арбатско-Покровская линия	Арбатская	8542.046323
1	Арбатско-Покровская линия	Бауманская	15108.715147
2	Арбатско-Покровская линия	Волоколамская	5046.476181
3	Арбатско-Покровская линия	Измайловская	5532.339348
4	Арбатско-Покровская линия	Киевская	13961.348220
5	Арбатско-Покровская линия	Крылатское	7508.210840
6	Арбатско-Покровская линия	Кунцевская	4624.440518

Source:

ПОРТАЛ ОТКРЫТЫХ ДАННЫХ
Правительства Москвы

The data overview

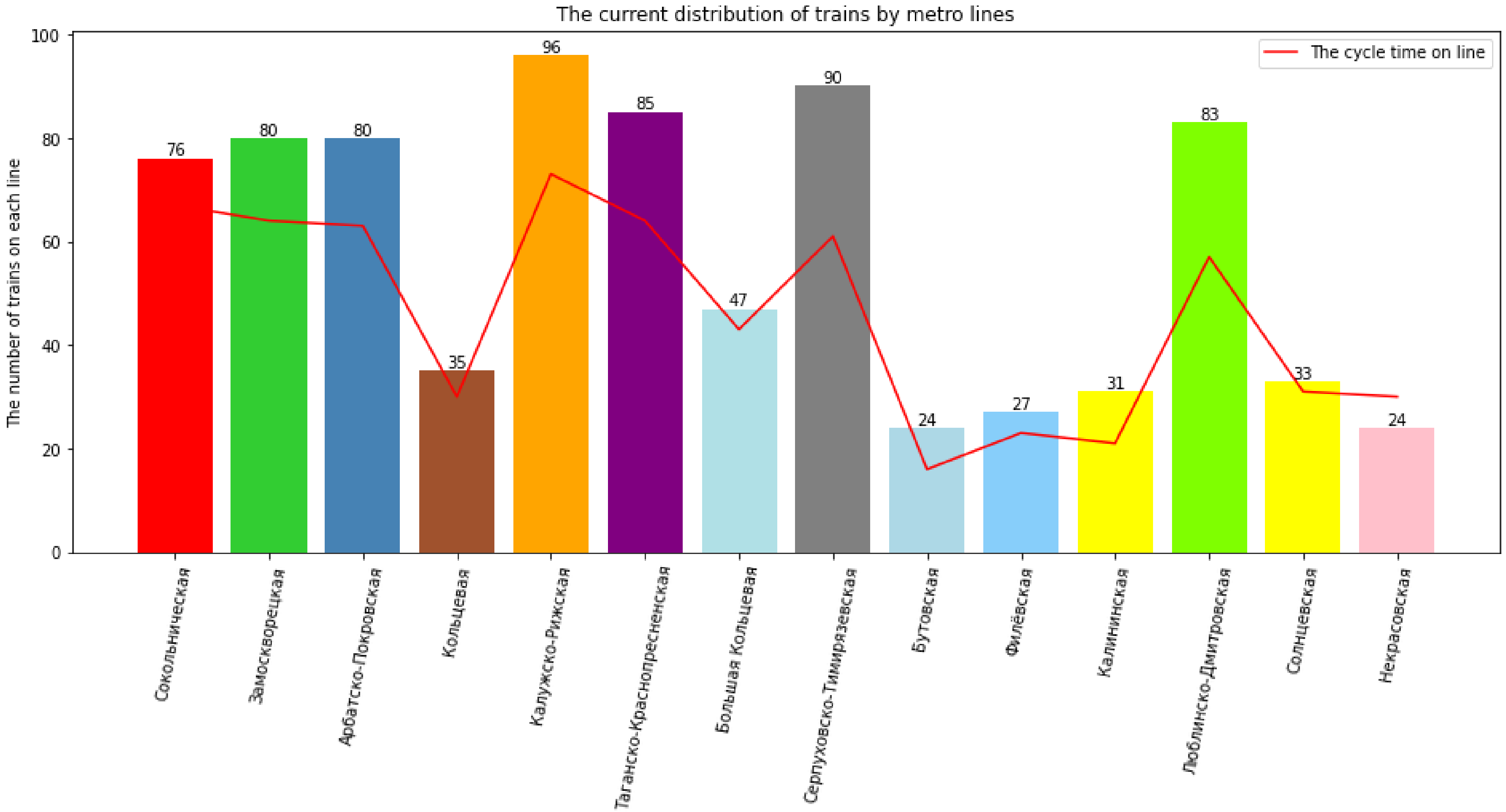
After some work with data, we got this dataset with the current distribution of trains.



	Линия	Количество поездов	Время проезда по линии
0	Сокольническая линия	76	67
1	Замоскворецкая линия	80	64
2	Арбатско-Покровская линия	80	63
3	Кольцевая линия	35	30
4	Калужско-Рижская линия	96	73
5	Таганско-Краснопресненская линия	85	64
6	Большая кольцевая линия	47	43
7	Серпуховско-Тимирязевская линия	90	61
8	Бутовская линия Лёгкого метро	24	16
9	Филёвская линия	27	23
10	Калининская линия	31	21
11	Люблинско-Дмитровская линия	83	57
12	Солнцевская линия	33	31

Source:

The current distribution



The model

The passenger comes to the station according to some arrival process.

The passenger waits for the next train and gets on this train.

The train stays on the station at the moment t=0.

The next train will come to the station after some fixed interval of time.

The intervals

$$\text{The interval} = \frac{\text{The time of full cycle for this line}}{\text{The number of trains on the line}} = \frac{\text{🔁}}{\text{🚇🚇🚇}}$$

Step 1

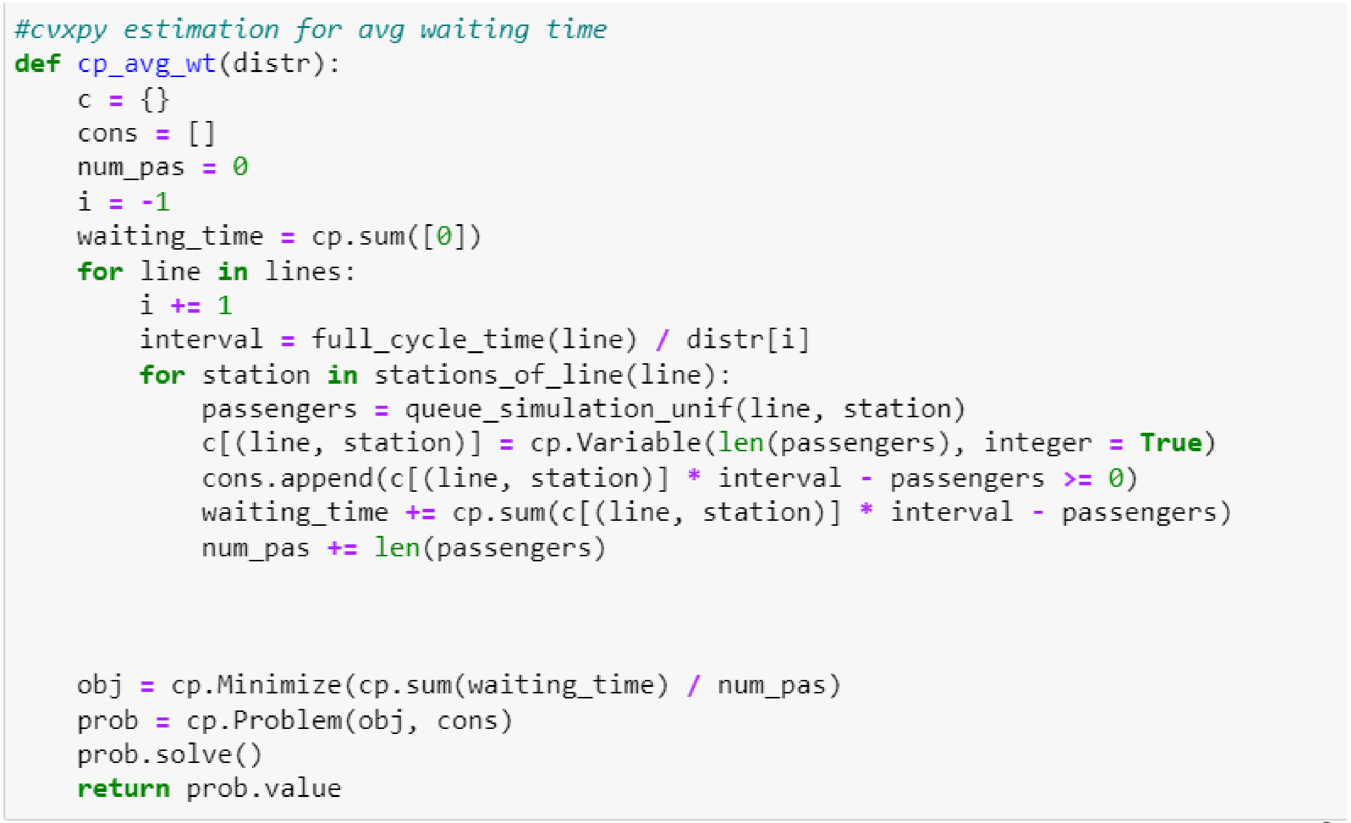
Compute the average waiting time for the fixed distribution of trains

Compare cvxpy and numpy-only realization

Compare the uniform distribution for the arrival process and Poisson process.

Estimation of waiting time: numpy vs cvxpy

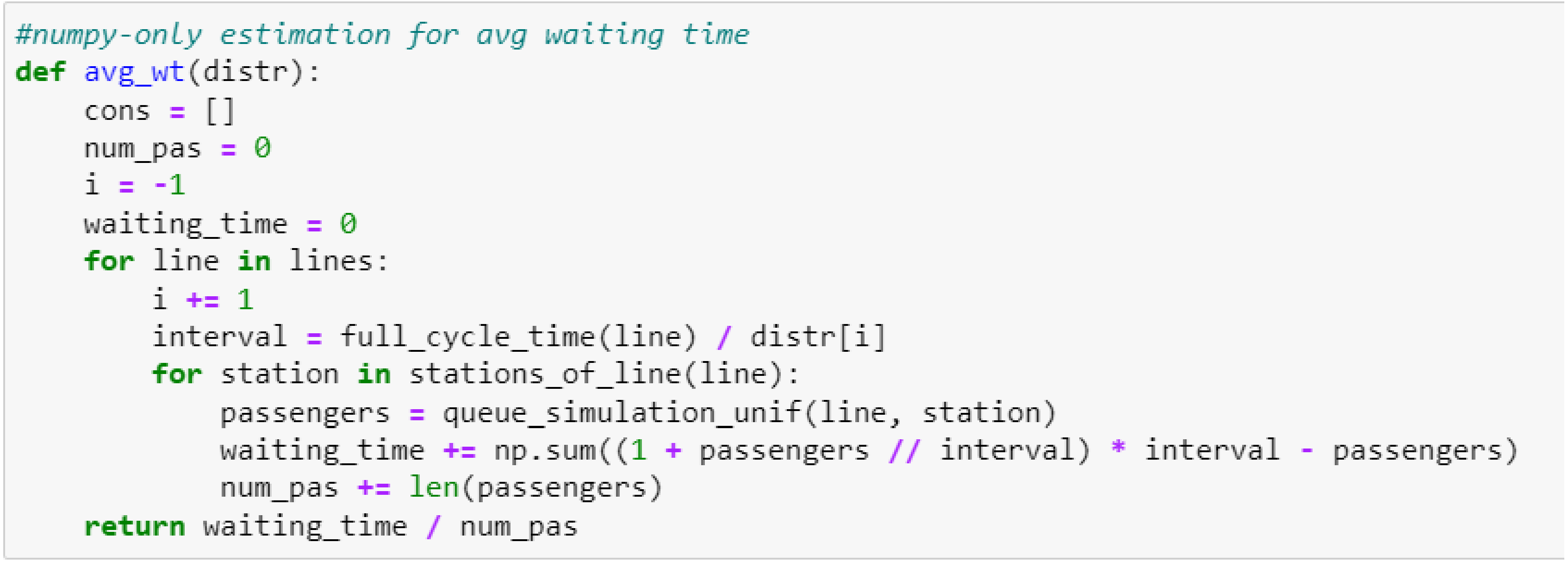
Cvxpy based estimation:



```
#cvxpy estimation for avg waiting time
def cp_avg_wt(distr):
    c = {}
    cons = []
    num_pas = 0
    i = -1
    waiting_time = cp.sum([0])
    for line in lines:
        i += 1
        interval = full_cycle_time(line) / distr[i]
        for station in stations_of_line(line):
            passengers = queue_simulation_unif(line, station)
            c[(line, station)] = cp.Variable(len(passengers), integer = True)
            cons.append(c[(line, station)] * interval - passengers >= 0)
            waiting_time += cp.sum(c[(line, station)] * interval - passengers)
            num_pas += len(passengers)



    obj = cp.Minimize(cp.sum(waiting_time) / num_pas)
    prob = cp.Problem(obj, cons)
    prob.solve()
    return prob.value
```

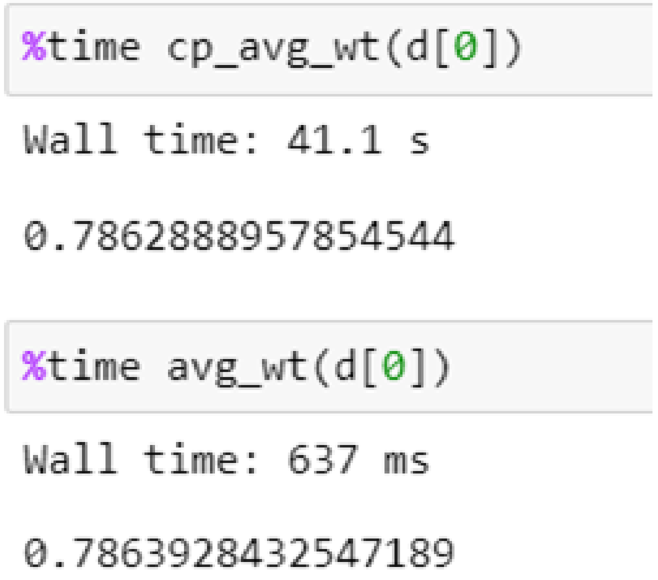
Estimation of waiting time: numpy vs cvxpy

Numpy based estimation:



```python
#numpy-only estimation for avg waiting time
def avg_wt(distr):
    cons = []
    num_pas = 0
    i = -1
    waiting_time = 0
    for line in lines:
        i += 1
        interval = full_cycle_time(line) / distr[i]
        for station in stations_of_line(line):
            passengers = queue_simulation_unif(line, station)
            waiting_time += np.sum((1 + passengers // interval) * interval - passengers)
            num_pas += len(passengers)
    return waiting_time / num_pas
```

Estimation of waiting time: numpy vs cvxpy



```
%time cp_avg_wt(d[0])
```

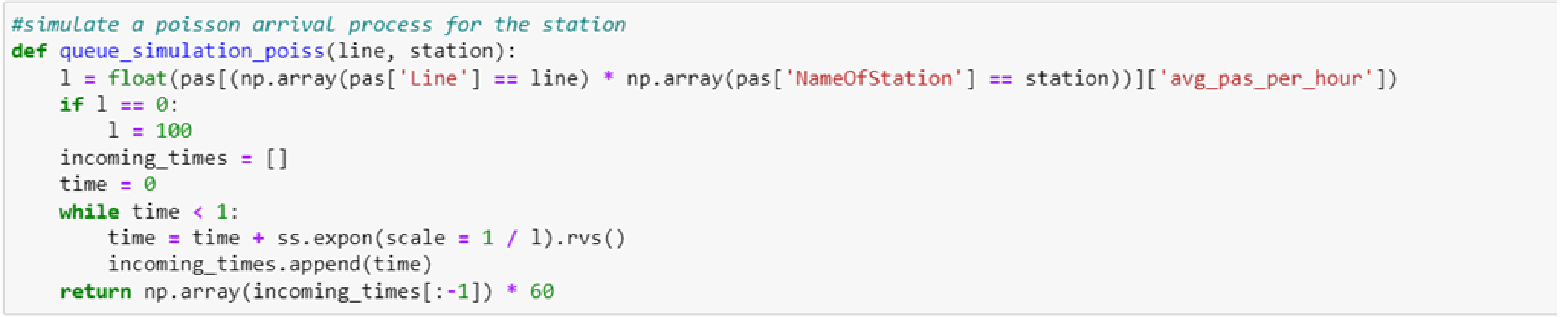
```
Wall time: 41.1 s

0.7862888957854544
```

```
%time avg_wt(d[0])
```

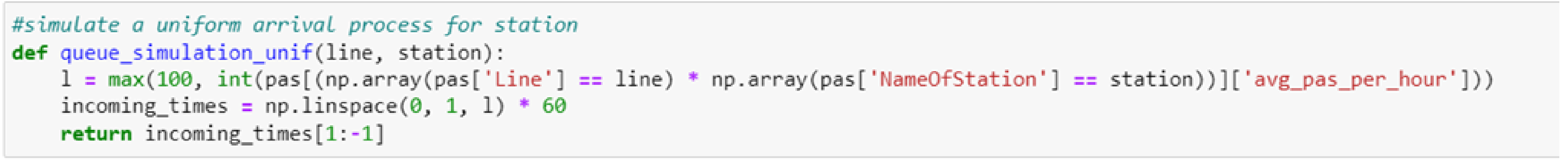
```
Wall time: 637 ms

0.7863928432547189
```

Numpy is better!

Poisson & Uniform arrival process, simulations

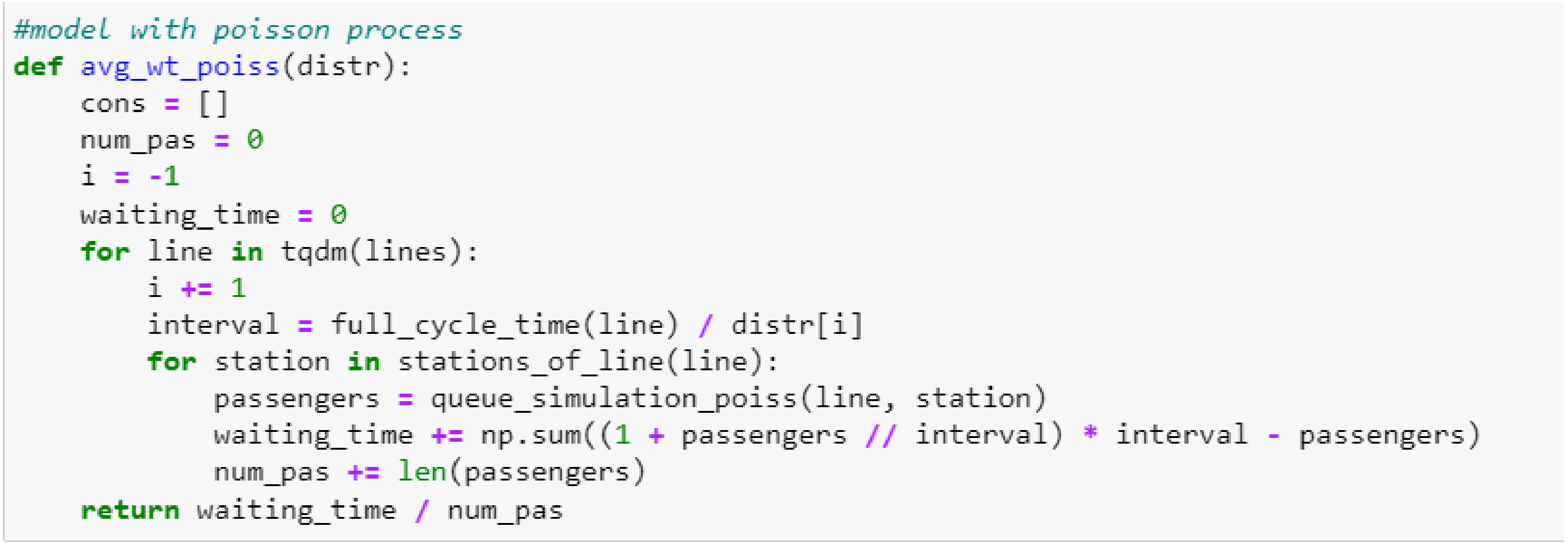


```python
#simulate a poisson arrival process for the station
def queue_simulation_poiss(line, station):
    l = float(pas[(np.array(pas['Line'] == line) * np.array(pas['NameOfStation'] == station))]['avg_pas_per_hour'])
    if l == 0:
        l = 100
    incoming_times = []
    time = 0
    while time < 1:
        time = time + ss.expon(scale = 1 / l).rvs()
        incoming_times.append(time)
    return np.array(incoming_times[:-1]) * 60
```

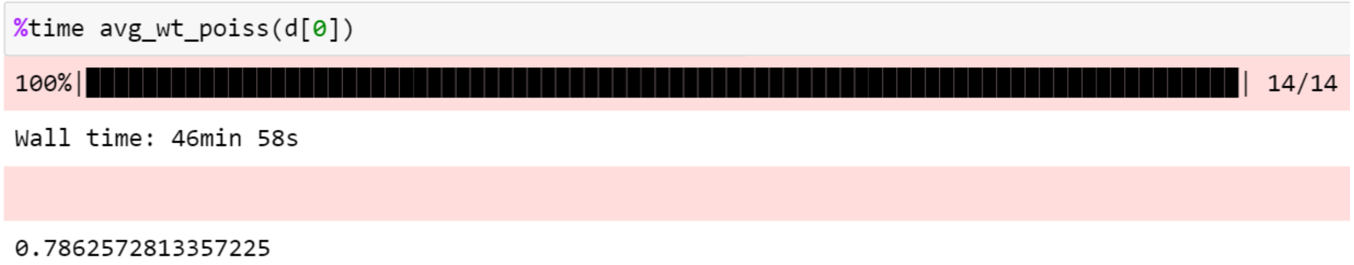


```python
#simulate a uniform arrival process for station
def queue_simulation_unif(line, station):
    l = max(100, int(pas[(np.array(pas['Line'] == line) * np.array(pas['NameOfStation'] == station))]['avg_pas_per_hour']))
    incoming_times = np.linspace(0, 1, l) * 60
    return incoming_times[1:-1]
```

Poisson process or uniform distribution for the arrival?



```python
#model with poisson process
def avg_wt_poiss(distr):
    cons = []
    num_pas = 0
    i = -1
    waiting_time = 0
    for line in tqdm(lines):
        i += 1
        interval = full_cycle_time(line) / distr[i]
        for station in stations_of_line(line):
            passengers = queue_simulation_poiss(line, station)
            waiting_time += np.sum((1 + passengers // interval) * interval - passengers)
            num_pas += len(passengers)
    return waiting_time / num_pas
```



```
%time avg_wt_poiss(d[0])
```

```
100%|██████████████████████████████████████████████████| 14/14

Wall time: 46min 58s

0.7862572813357225
```

Poisson process or uniform distribution for the arrival?

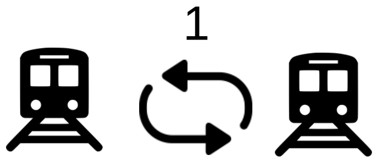
Poisson process and uniform distribution have almost the same effects, but Poisson process works much more slowly.

The final model: uniform distribution for arrivals + numpy – only realization.

Step 2

Find the distributions which has one train difference compared with the current distribution.

Find the best distribution among them.

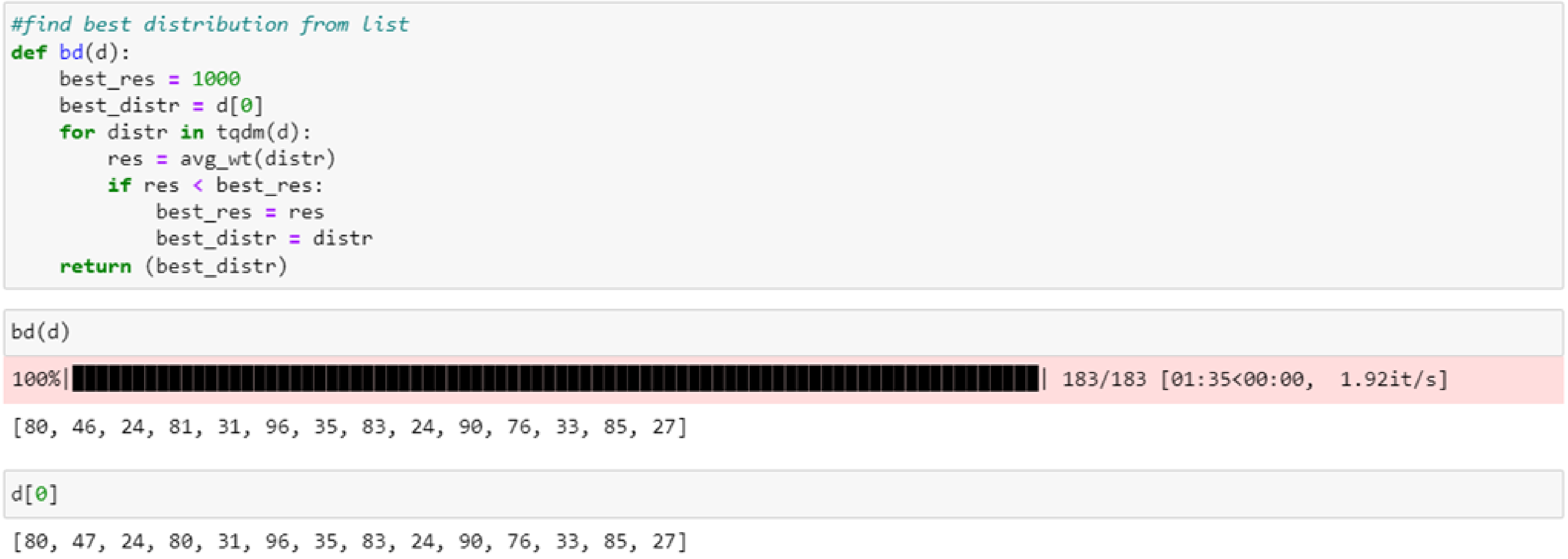


```
cur_trains = [int(trains[trains['Линия'] == i]['Количество поездов']) for i in lines]
```

```
#find the distribution with difference in one train
def neighboring_distr(distr):
    ans = []
    ans.append(distr)
    for i in range(len(distr)):
        for j in range(len(distr)):
            if i != j:
                c = distr.copy()
                c[i], c[j] = c[i] - 1, c[j] + 1
                if c[i] >= 0:
                    ans.append(c)
    return ans
```

```
d = neighboring_distr(cur_trains)
```

Moving the first train



```
#find best distribution from list
def bd(d):
    best_res = 1000
    best_distr = d[0]
    for distr in tqdm(d):
        res = avg_wt(distr)
        if res < best_res:
            best_res = res
            best_distr = distr
    return (best_distr)
```

```
bd(d)
```

```
100%|████████████████████████████████████████████████████████████████████████████████| 183/183 [01:35<00:00,  1.92it/s]

[80, 46, 24, 81, 31, 96, 35, 83, 24, 90, 76, 33, 85, 27]
```

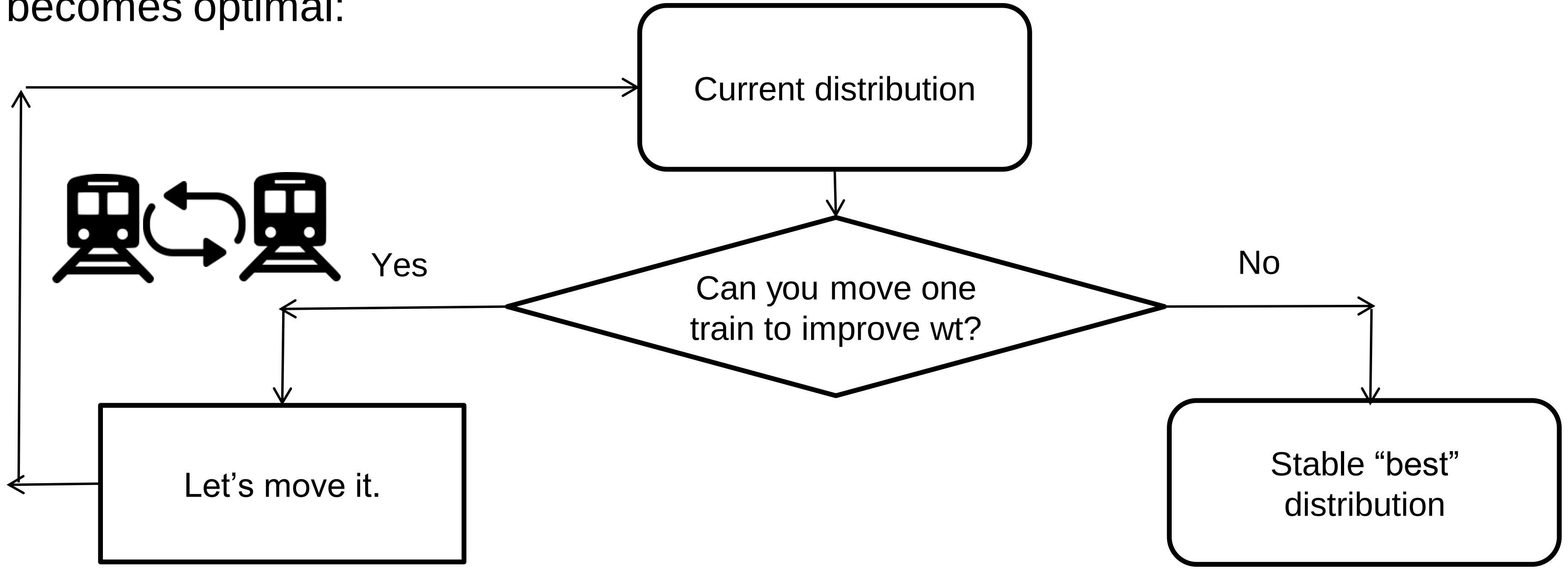
```
d[0]
```

```
[80, 47, 24, 80, 31, 96, 35, 83, 24, 90, 76, 33, 85, 27]
```

The first suggestion is to move from Большая кольцевая линия to Замоскворецкая линия.

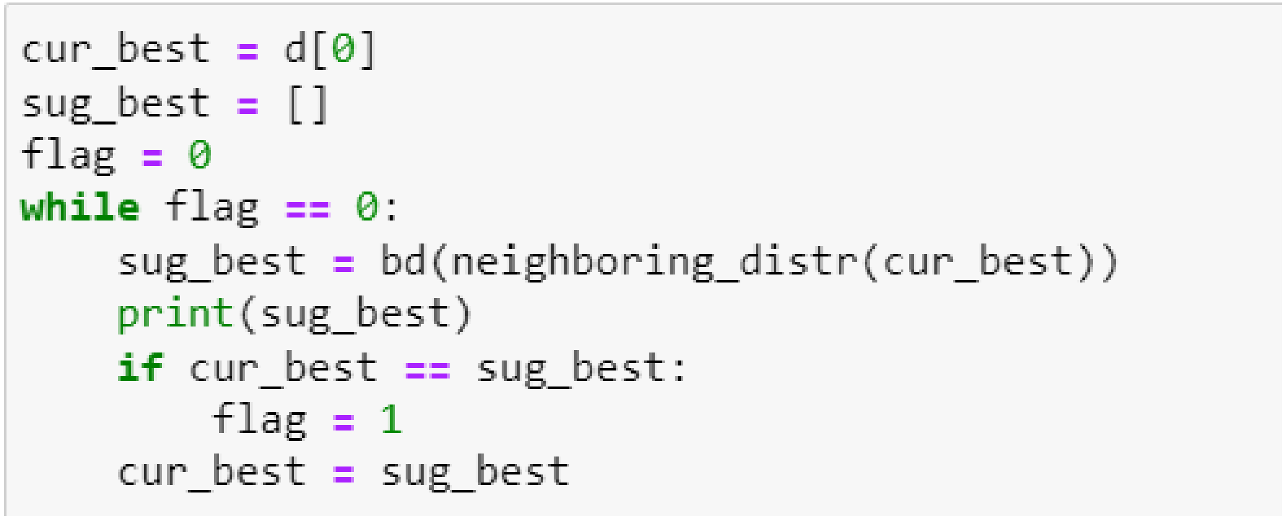
Step 3: Optimizing

Now let's create an algorithm to move trains from line to line until the distribution becomes optimal:


Current distribution
Yes
Can you move one train to improve wt?
No
Let's move it.
Stable "best" distribution

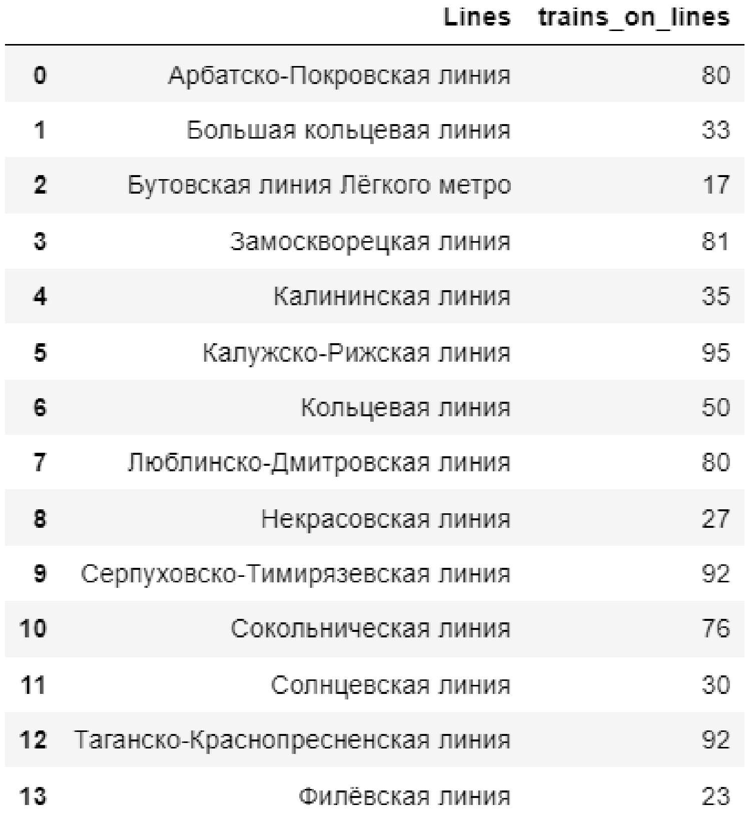
Optimizing

Now let’s create an algorithm to move trains from line to line until the distribution becomes optimal:

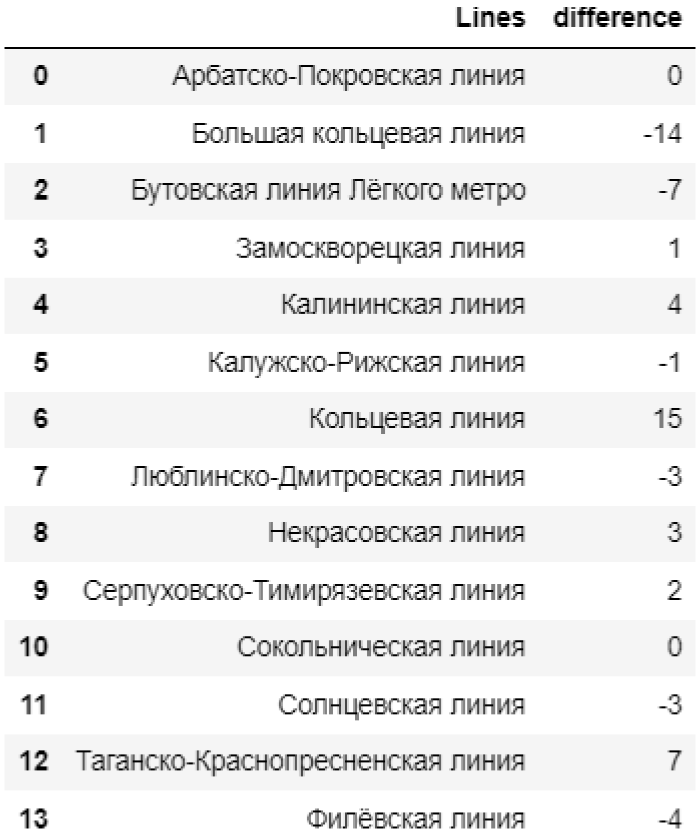


```
cur_best = d[0]
sug_best = []
flag = 0
while flag == 0:
    sug_best = bd(neighboring_distr(cur_best))
    print(sug_best)
    if cur_best == sug_best:
        flag = 1
    cur_best = sug_best
```

The results: optimal distribution

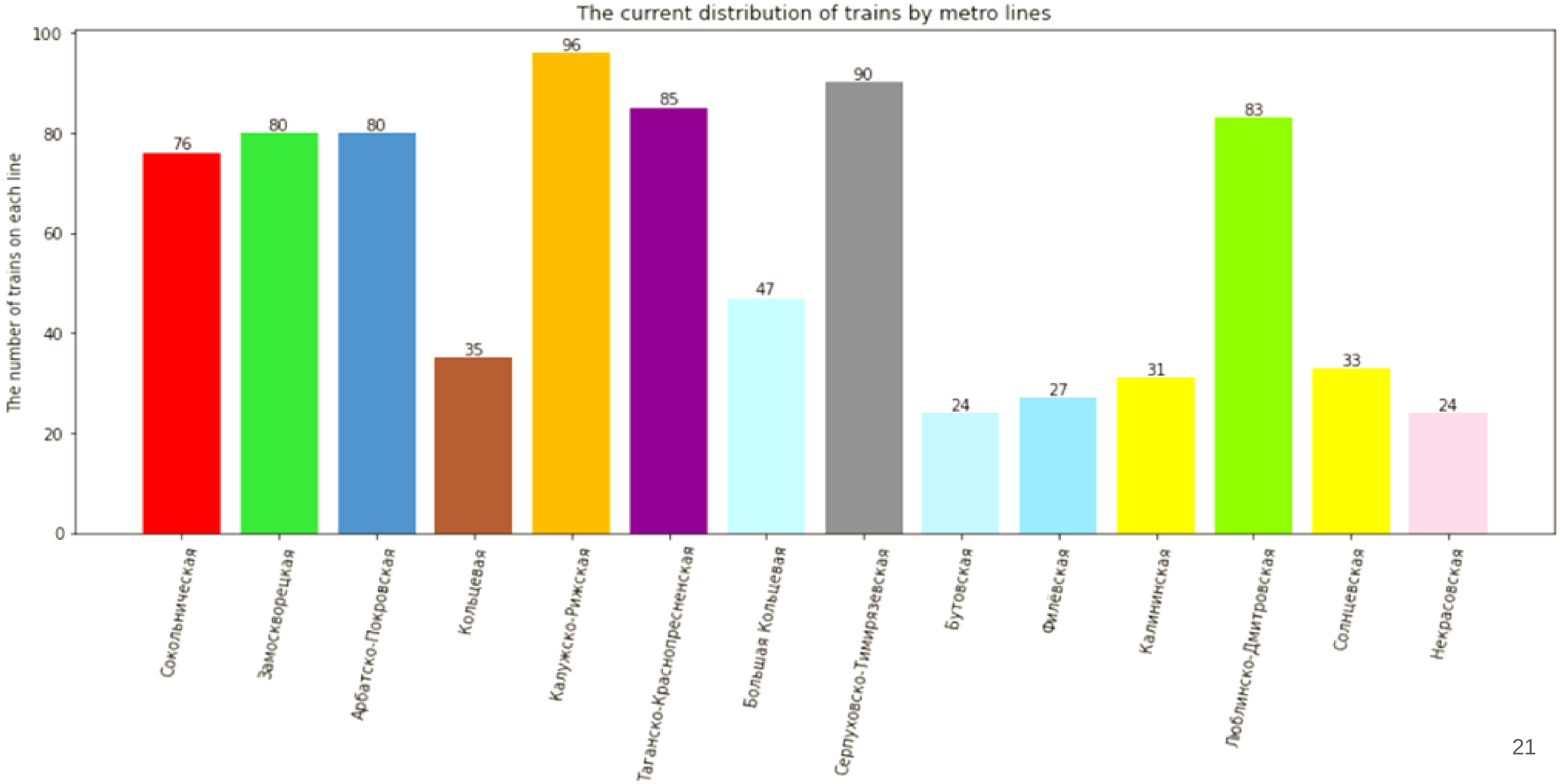


	Lines	trains_on_lines
0	Арбатско-Покровская линия	80
1	Большая кольцевая линия	33
2	Бутовская линия Лёгкого метро	17
3	Замоскворецкая линия	81
4	Калининская линия	35
5	Калужско-Рижская линия	95
6	Кольцевая линия	50
7	Люблинско-Дмитровская линия	80
8	Некрасовская линия	27
9	Серпуховско-Тимирязевская линия	92
10	Сокольническая линия	76
11	Солнцевская линия	30
12	Таганско-Краснопресненская линия	92
13	Филёвская линия	23

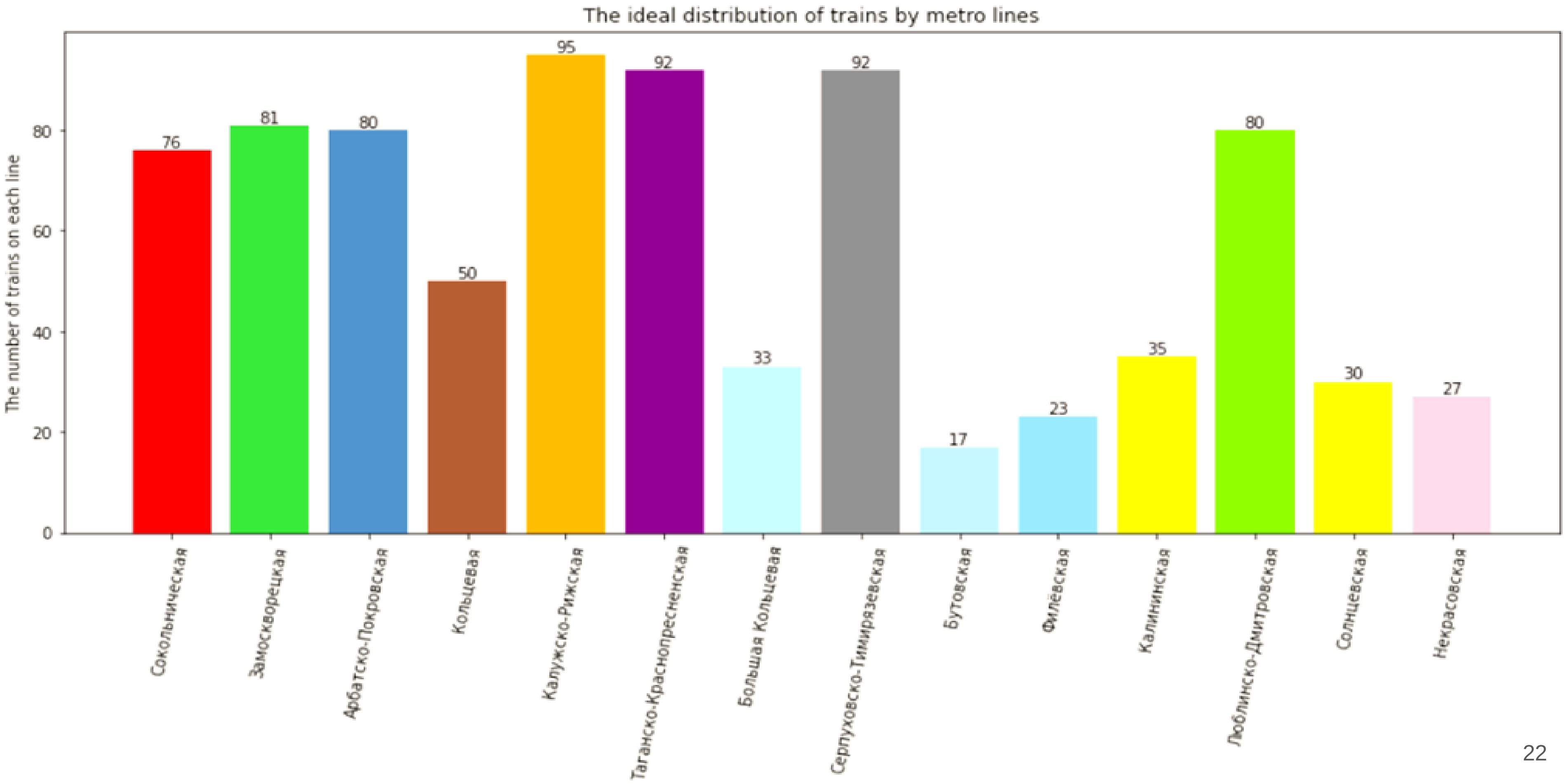


	Lines	difference
0	Арбатско-Покровская линия	0
1	Большая кольцевая линия	-14
2	Бутовская линия Лёгкого метро	-7
3	Замоскворецкая линия	1
4	Калининская линия	4
5	Калужско-Рижская линия	-1
6	Кольцевая линия	15
7	Люблинско-Дмитровская линия	-3
8	Некрасовская линия	3
9	Серпуховско-Тимирязевская линия	2
10	Сокольническая линия	0
11	Солнцевская линия	-3
12	Таганско-Краснопресненская линия	7
13	Филёвская линия	-4

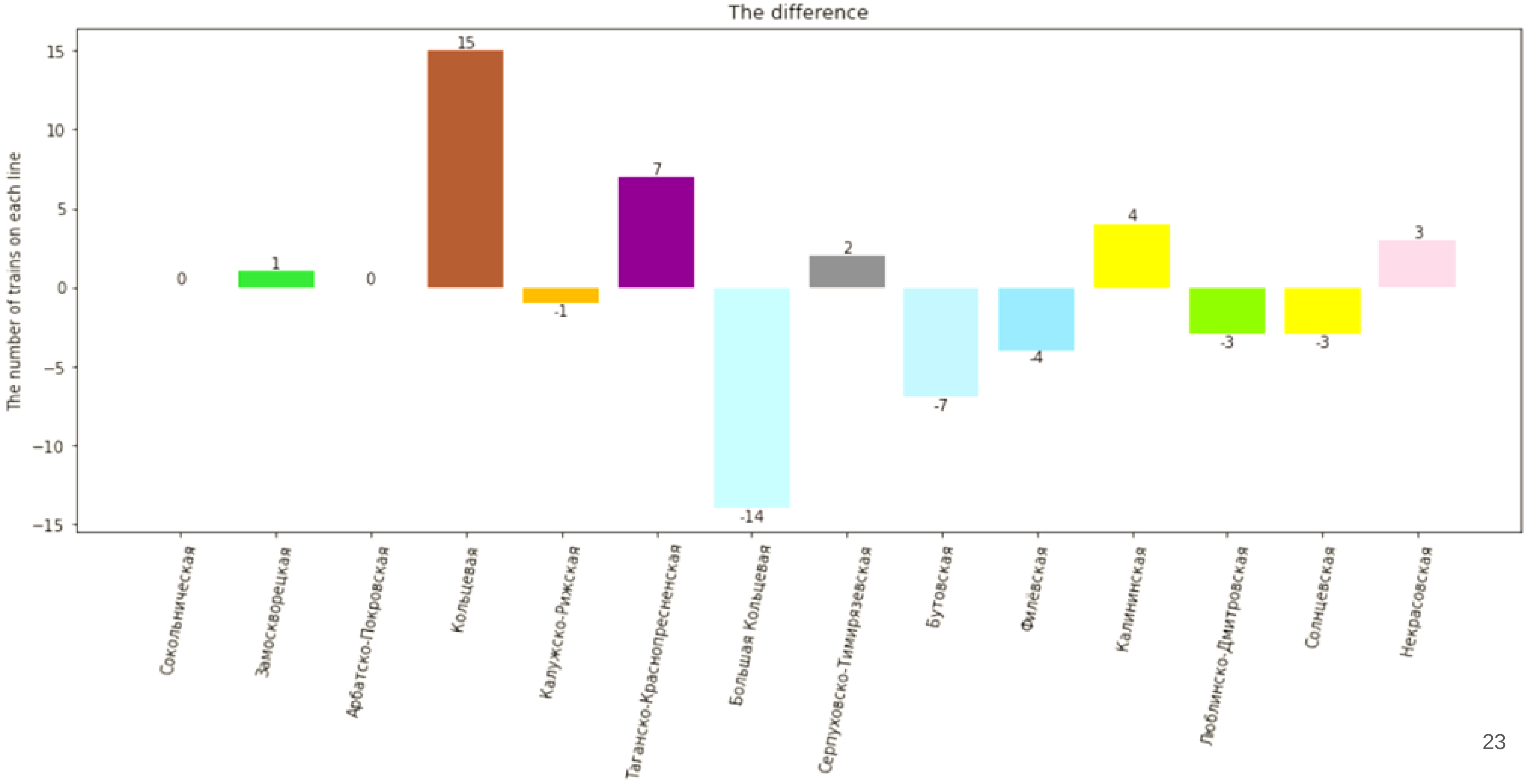
Current distribution



Optimal distribution



The difference in distributions



Discussion of the results

- Overall, the current distribution is close to the ideal one on most lines.
- However, it seems optimal to relocate more trains to the Кольцевая линия and reduce the number of trains on the Большая Кольцевая линия.
- Why hasn’t that happened?

Possible explanations:

- The technical issues regarding Кольцевая линия;
- The government’s desire to popularize Большая Кольцевая линия, possibly as an alternative to more busy lines.

Questions?