Implementation of Lin *et al.* 2014 “Energy and Performance-Aware Task Scheduling in a Mobile Cloud Computing Environment” Algorithms

Shane Hussey | EECE 7205 V35 Fall 2023

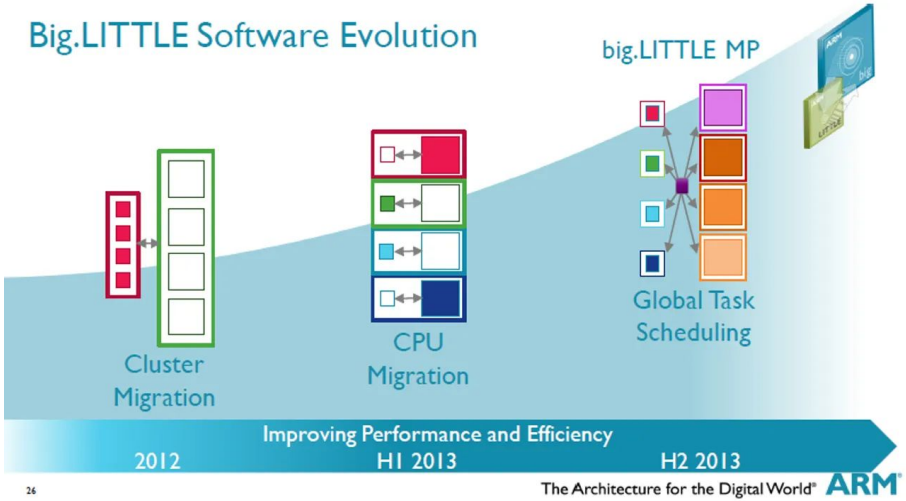
Background

The overall goal of big.LITTLE architecture was to provide a balance between performance and power efficiency in task scheduling for cell phones around 2014.

The original idea was to use the A15 custer for CPU intensive tasks and the A7 cluster for low power tasks.

- Gaming, Web page rendering - A-15
- Texting, Email - A7

In the earlier big.LITTLE software models, the software switched between cores and could not switch all cores on simultaneously.



https://www.zdnet.com/article/samsung-kicks-off-march-of-arms-big-little-architecture/

https://community.arm.com/arm-community-blogs/b/architectures-and-processors-blog/posts/ten-things-to-know-about-big-little

https://1000logos.net/samsung-logo/

MCC TASK SCHEDULING ALGORITHM (Lin et al. 2014)

1.) Step One: Initial Scheduling Algorithm
   a.) Primary assignment
   b.) Task prioritizing
   c.) Execution unit selection
2.) Step Two: Task Migration Algorithm
   a.) Outer loop
   b.) Kernel algorithm (i.e., rescheduling algorithm)

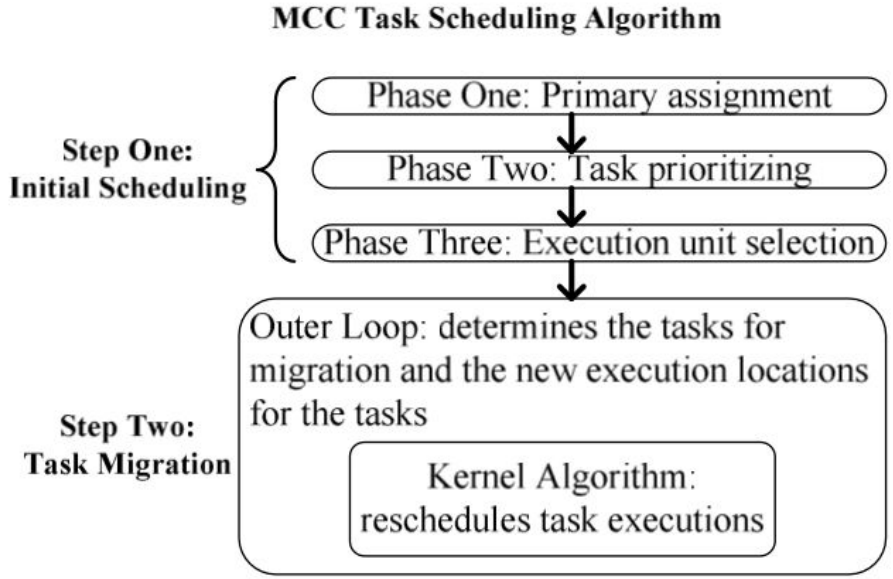


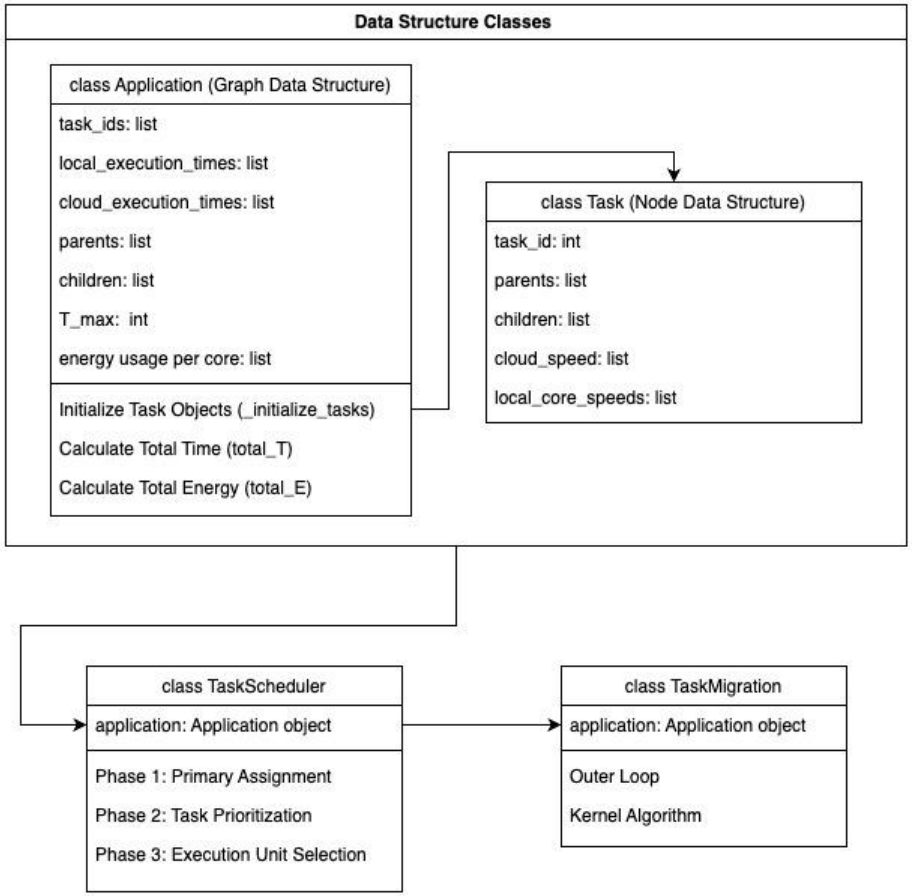
Figure 2. Flow chart of the MCC task scheduling algorithm.

Lin *et al.* 2014

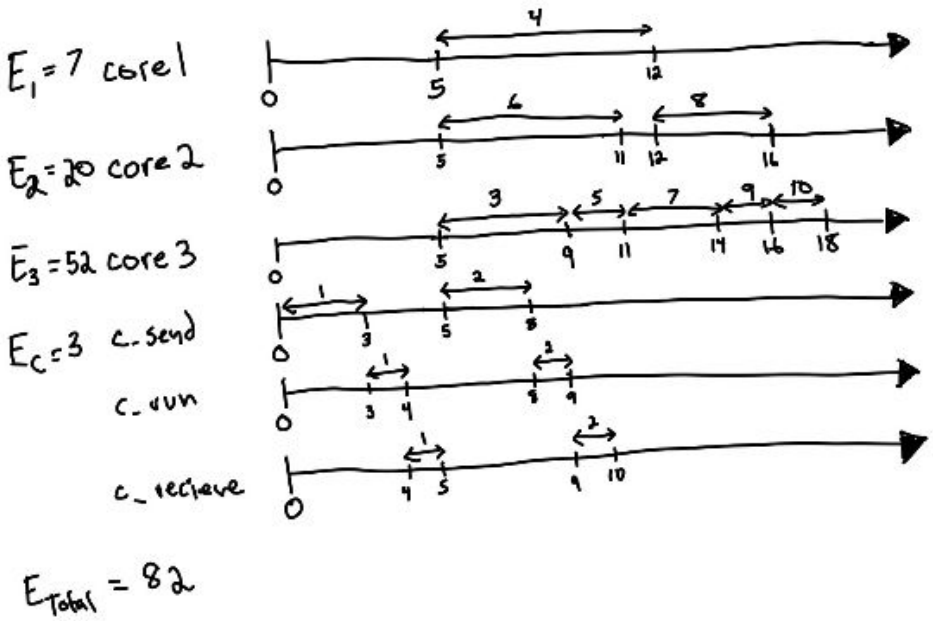
Data Structures and Implementation Overview

Example usage:

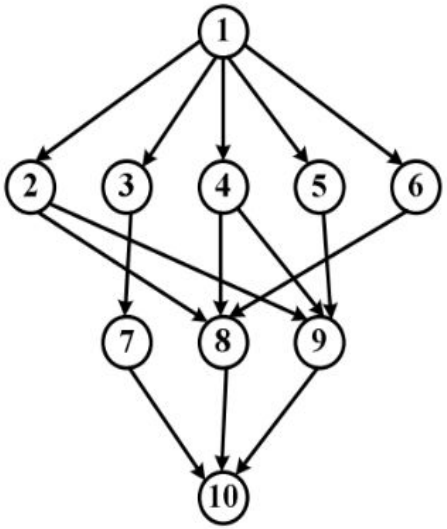
```
# initialize an application object
app = Application(task_ids=task_ids,
                  local_execution_times=local_execution_times,
                  cloud_execution_times=cloud_execution_times,
                  parents=parents,
                  children=children,
                  T_max=T_maximum,
                  energy_usage=e_usage)

app.set_entry_tasks_ready_time(0)

# run the task scheduler
TaskScheduler(app)

# run the task migration step
TaskMigration(app)
```



Test 1: Figure 1 inputs from Lin *et al.* 2014



Initial Assignment

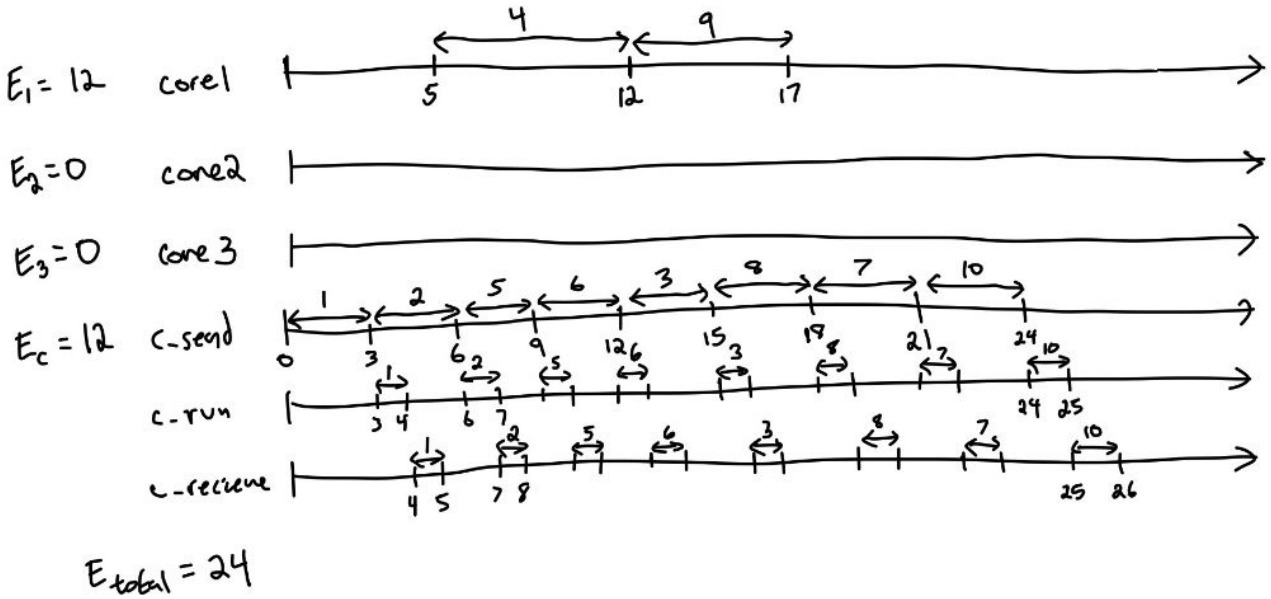


Task	Core1	Core2	Core3
1	9	7	5
2	8	6	5
3	6	5	4
4	7	5	3
5	5	4	2
6	7	6	4
7	8	5	3
8	6	4	2
9	5	3	2
10	7	4	2

$$1 \le i \le N, \begin{cases} T_i^s = 3 \\ T_i^c = 1 \\ T_i^r = 1 \end{cases}$$

Figure 1. An example task graph.

Test 1: (part 2)



T_total initial = 18

T_max = 27

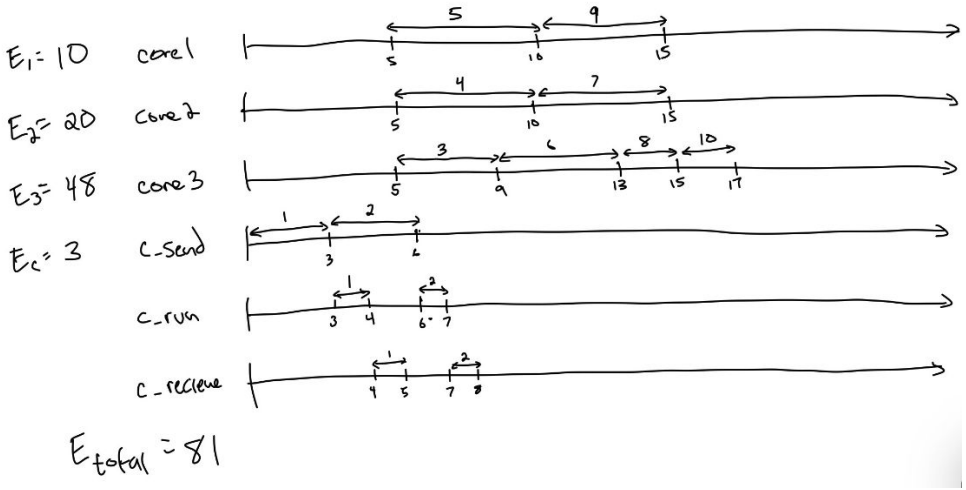
T_total final = 26

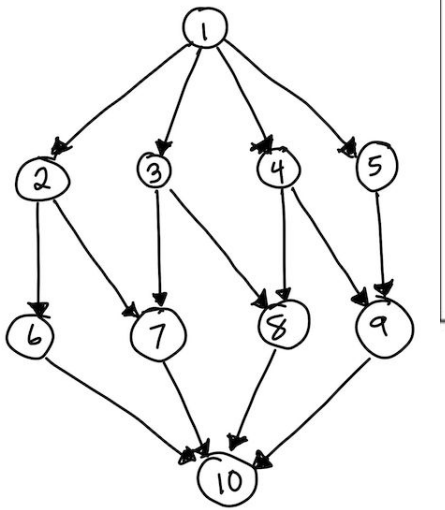
E_total of initial = 82

E_total final = 24

Test 2:

The same number of tasks as in Figure 1 of the paper, but change connections in the task graph (in this case, the execution time table in Figure 1 can be used directly)



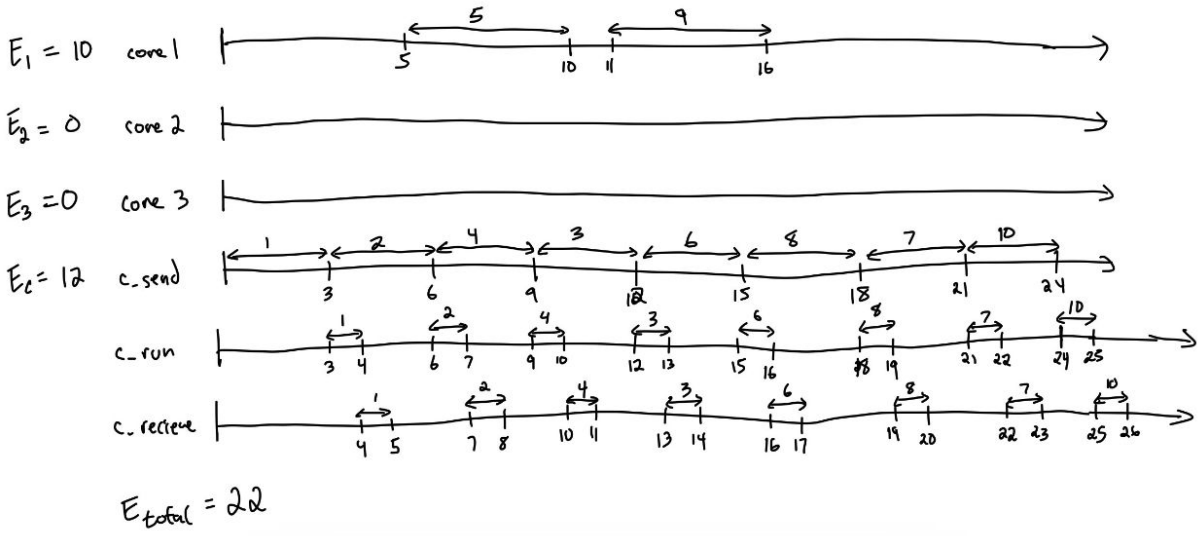


Task	Core1	Core2	Core3
1	9	7	5
2	8	6	5
3	6	5	4
4	7	5	3
5	5	4	2
6	7	6	4
7	8	5	3
8	6	4	2
9	5	3	2
10	7	4	2

$$1 \le i \le N, \begin{cases} T_i^s = 3 \\ T_i^c = 1 \\ T_i^r = 1 \end{cases}$$

Test 2:

The same number of tasks as in Figure 1 of the paper, but change connections in the task graph (in this case, the execution time table in Figure 1 can be used directly)



T_total initial = 17

T_max = 27

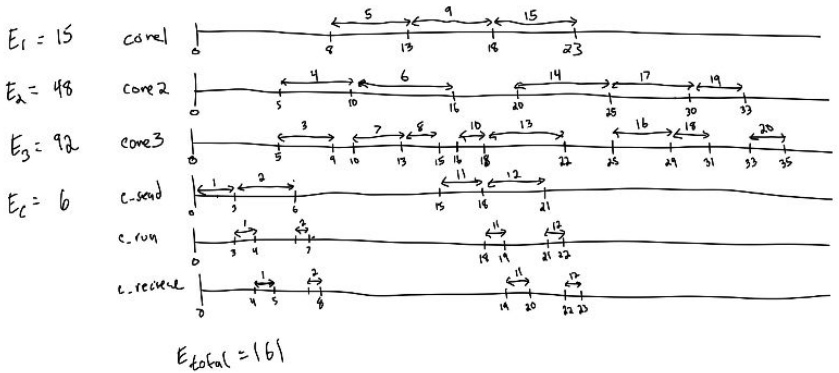
T_total final = 26

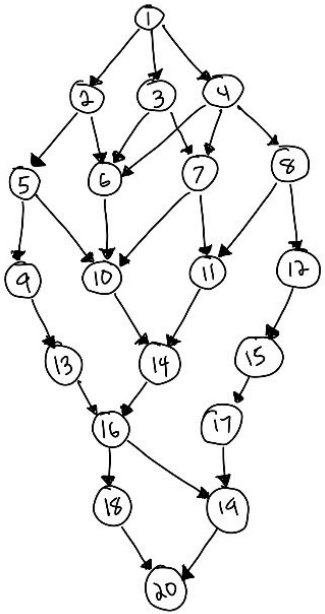
E_total of initial = 81

E_total final = 22

Test 3:

increase task number to 20, redesign the task graph (in this case, you need to add 10 more rows in the execution time table in Figure 1, and make sure that core 1 is the slowest and core 3 is the fastest in the added 10 rows)



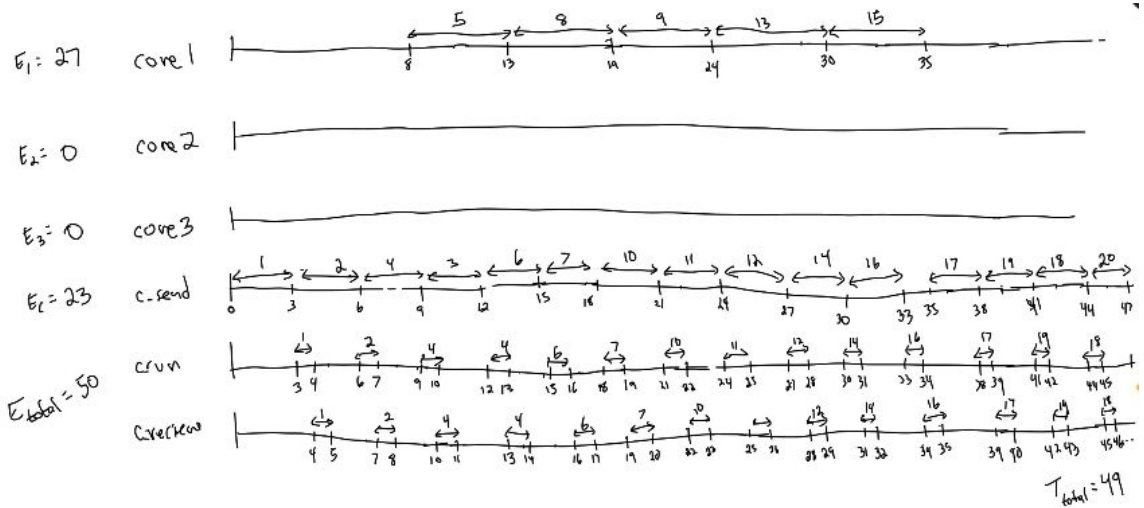


$$1 \le i \le N, \begin{cases} T_i^s = 3 \\ T_i^c = 1 \\ T_i^r = 1 \end{cases}$$

Task	Core1	Core2	Core3
1	9	7	5
2	8	6	5
3	6	5	4
4	7	5	3
5	5	4	2
6	7	6	4
7	8	5	3
8	6	4	2
9	5	3	2
10	7	4	2
11	9	7	5
12	8	6	5
13	6	5	4
14	7	5	3
15	5	4	2
16	7	6	4
17	8	5	3
18	6	4	2
19	5	3	2
20	7	4	2

Test 3:

increase task number to 20, redesign the task graph (in this case, you need to add 10 more rows in the execution time table in Figure 1, and make sure that core 1 is the slowest and core 3 is the fastest in the added 10 rows)



T_total initial = 35

T_max = 53

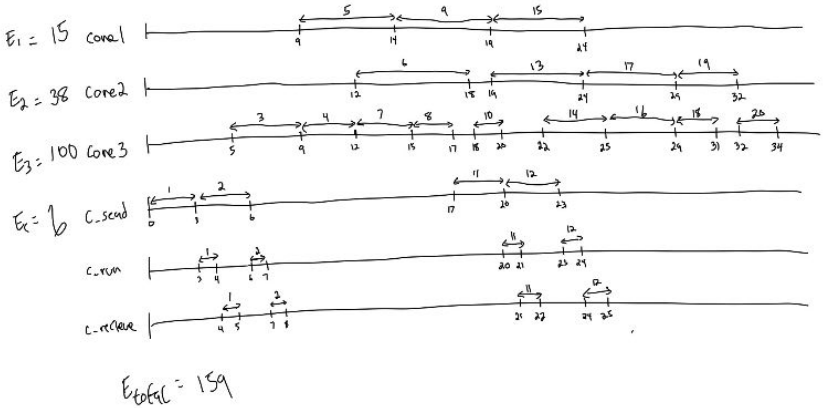
T_total final = 49

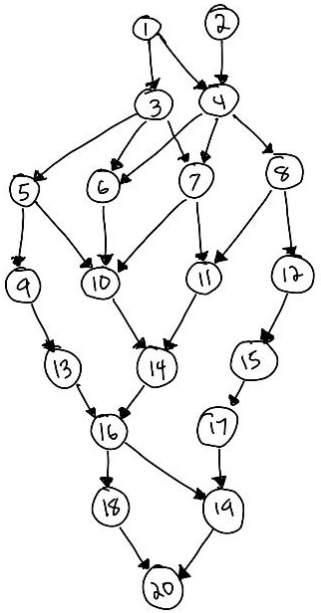
E_total of initial = 161

E_total final = 50

Test 4:

based on the previous input example (3), redesign its task graph by using multiple entry tasks (in this case, the execution time table in input example (3) can be used directly)



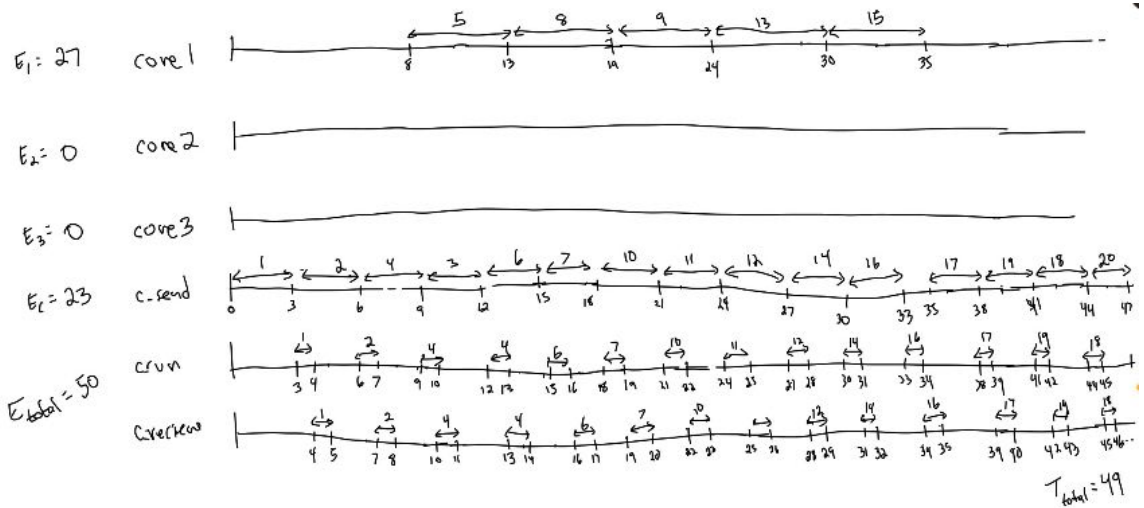


$$1 \le i \le N, \begin{cases} T_i^s = 3 \\ T_i^c = 1 \\ T_i^r = 1 \end{cases}$$

Task	Core1	Core2	Core3
1	9	7	5
2	8	6	5
3	6	5	4
4	7	5	3
5	5	4	2
6	7	6	4
7	8	5	3
8	6	4	2
9	5	3	2
10	7	4	2
11	9	7	5
12	8	6	5
13	6	5	4
14	7	5	3
15	5	4	2
16	7	6	4
17	8	5	3
18	6	4	2
19	5	3	2
20	7	4	2

Test 4:

based on the previous input example (3), redesign its task graph by using multiple entry tasks (in this case, the execution time table in input example (3) can be used directly)



T_total initial = 34

T_max = 53

T_total final = 49*

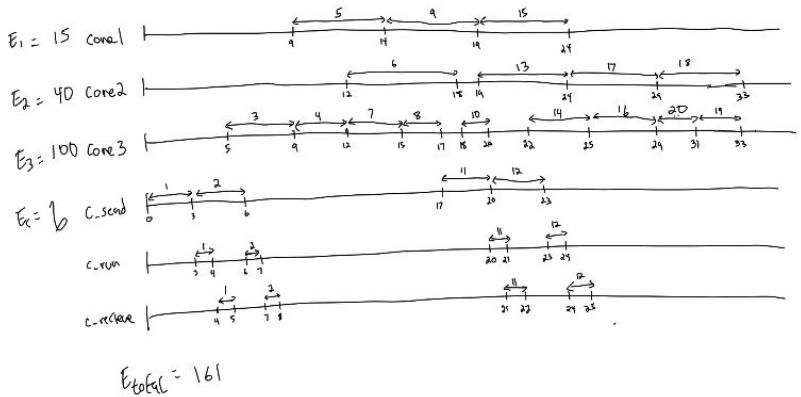
E_total of initial = 159

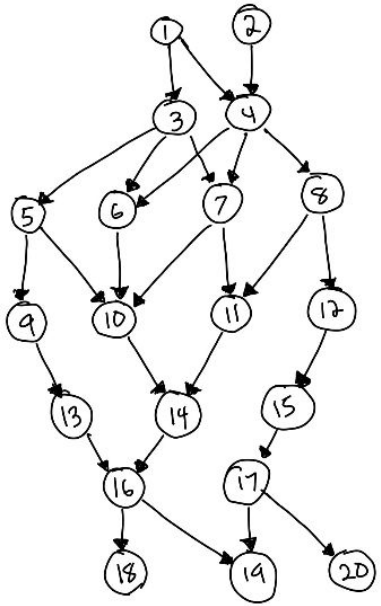
E_total final = 50

*same configuration as test3, probably because I didn't make significant changes to connections.

Test 5:

base on the previous input example (3), redesign its task graph by using multiple entry tasks and multiple exit tasks (in this case, the execution time table in input example (3) can be used directly)



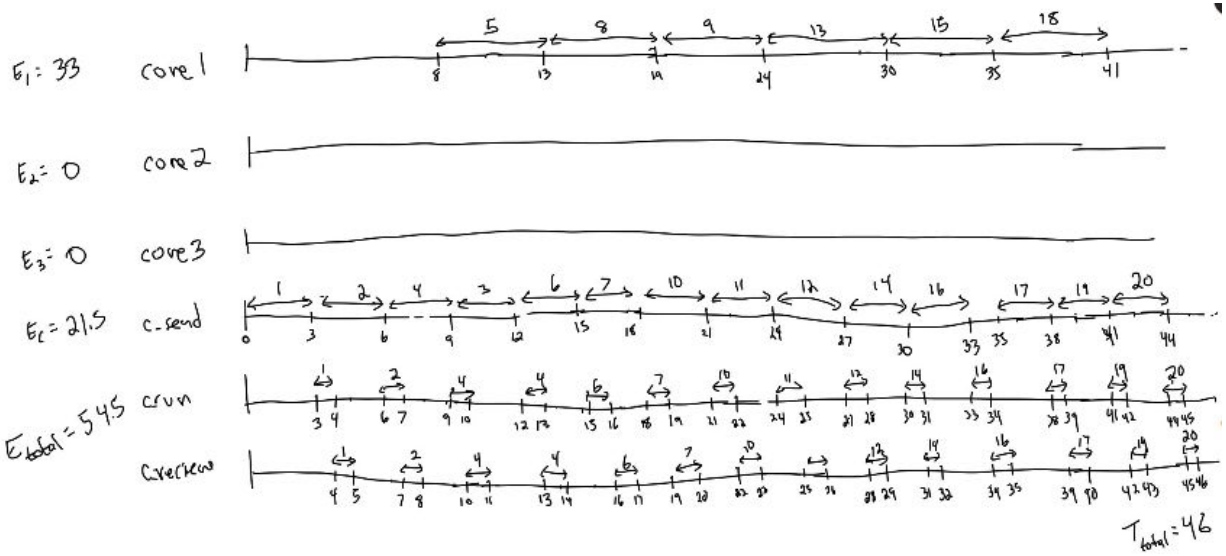


$$1 \le i \le N, \begin{cases} T_i^s = 3 \\ T_i^c = 1 \\ T_i^r = 1 \end{cases}$$

Task	Core1	Core2	Core3
1	9	7	5
2	8	6	5
3	6	5	4
4	7	5	3
5	5	4	2
6	7	6	4
7	8	5	3
8	6	4	2
9	5	3	2
10	7	4	2
11	9	7	5
12	8	6	5
13	6	5	4
14	7	5	3
15	5	4	2
16	7	6	4
17	8	5	3
18	6	4	2
19	5	3	2
20	7	4	2

Test 5:

base on the previous input example (3), redesign its task graph by using multiple entry tasks and multiple exit tasks (in this case, the execution time table in input example (3) can be used directly)



T_total initial = 33

T_max = 53

T_total final = 46

E_total of initial = 161

E_total final = 54.5