COURSEWORK

IMPERIAL COLLEGE LONDON

DEPARTMENT OF COMPUTING

Scheduling and Resource Allocation

Author:
Arnav Kohli (CID: 02018546)
Ratul Shek (CID: 02015627)

Date: November 27, 2024

1 Question 1

1.1 Part 1

1.1.1 Explanation of Proof

The Least Cost Last (LCL) algorithm schedules jobs based on the following parameters:

- Input: A set of n jobs $N = \{1, 2, \ldots, n\}$ with precedence constraints (if job i must precede job j, $i \prec j$).
- A cost function $g_j(C_j)$ for each job j, where g_j is non-decreasing.

The goal is to minimize $g_{max}* = max_{j \in N} g_j(C_j)$, where $g_j(C_j)$ is a non-decreasing cost function of the job completion time C_j, subject to precedence constraints. The proof shows that, at each step, there exists an optimal schedule where the job in the set L (jobs without successors) with the least cost increase is scheduled last.

Definitions: Begin by extracting a subset $L \subseteq N$, which represents a set of jobs without successors, these jobs can be scheduled at the end without violating precedence constraints. $p(S) = \sum_{j \in S} p_j$ represents the sum of the total processing time of the jobs in a subset $S \subseteq N$. Therefore $g_{max} * (S)$ will be the cost of an optimal schedule for the jobs in S.

Lower bound on optimal cost: Based on the definitions, the processing times of all jobs on a single machine will be $p(N)$ and the set L contains the job that is executed last, based on this information it is possible to construct the following inequality: $g_{max} * (N) \geq min_{j \in L} g_j(p(N))$. This inequality denotes that since the subset L will contain at least one of the jobs that will be scheduled last, it will mean that the optimal cost of completing all the jobs in N is at least the minimum of scheduling a job $j \in L$ last.

Effect of Removing a job: If one were to remove a job from the set N, it would net the following inequality: $g_{max} * (N) \geq g_{max} * (N - \{j\})$. This makes sense, as the optimal total cost of scheduling N jobs cannot increase if one were to remove a job, as there are now fewer jobs to schedule.
Optimal Cost with $J_{l \in L}$ Scheduled Last: Select a job J_l which is in L, and select it so that it results in the least cost if it were to be selected last, i.e. $g_l(p(N)) = min_{j \in L} g_j(p(N)$. It can then be inferred that:

$$g_{max} * (N) \geq max\{g_l(p(N), g_{max} * (N - \{l\}\}$$

Thus confirming that the optimal schedule cost will be at least the maximum between the cost of scheduling J_l last and the cost of scheduling all remaining jobs. Repeatedly applying this rule leads to the optimal solution, as finding J_l can be achieved in linear time, $O(n)$, it follows that the optimal solution can be completed proportional to $O(n^2)$ time.

1.1.2 Illustration of proof for maximum tardiness

To illustrate with a smaller Directed Acyclic Graph (DAG) where the cost function being minimized is tardiness, consider 5 jobs, $J_n = \{J_1, J_2, J_3, J_4, J_5\}$, with the following due dates and completion times:

Due dates: $[30, 73, 55, 52, 54]$
Completion times: $[17, 26, 20, 14, 12]$

L is defined to be the set of jobs, $L \in J_n$, that can be scheduled in the current iteration, in other words, it contains the set of jobs that can be scheduled last, i.e. jobs that have no successors. Tardiness is defined to be the measure of how late a job is in respect to its completion time:

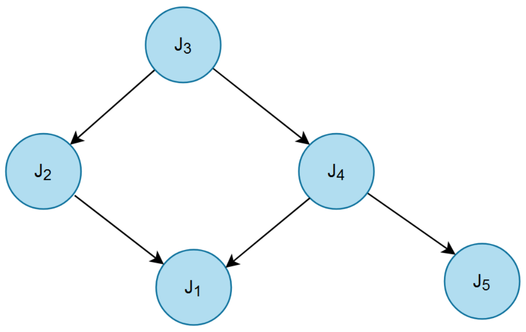
```
tardiness(j)=max{0, sum(completion_times)-due_date(j)}
```

Steps:

- Since scheduling starts from the last job, begin by calculating the sum of all completion times, 89. `L` contains jobs 1 and 5 because they have no successor. The tardiness for these jobs is `{59, 35}`. Since job 5 has the lowest tardiness, pick this job and add it to the schedule. Since this job has now been scheduled, the completion time reduces by its processing time.
  `schedule: {5}`
  `total_completion_time = 77`
- In the next iteration, there is no new job that can be scheduled (no new job without successors). Pick 1, add it to the schedule and reduce the completion time.
  `schedule: {1, 5}`
  `total_completion_time = 60`
- Jobs 2 and 4 can now be scheduled. Repeating the steps above, we have tardiness: `{0, 8}`. Which means 2 will be scheduled.
  `schedule: {2, 1, 5}`
  `total_completion_time = 34`
- Job 4 has to be scheduled before 3.
  `schedule: {4, 2, 1, 5}`
  `total_completion_time = 20`

Since only one job is remaining, the final schedule is `schedule: {3, 4, 2, 1, 5}` with a maximum tardiness of 47 when scheduling job 2.

Figure 1: Diagram of the Directed Acyclic Graph



The algorithm outlined above is implemented in the code for the provided DAG. Further explanation of the structure can be found below.

1.2 Part 2

A general graph with edges (precedences), weights and jobs as its nodes can be found at `src/graph.py`. Precedences are defined as an adjacency matrix. The least cost rule (lcl) algorithm is defined in `src/lcl.py`. The `LCLGraph` data structure inherits from the base `Graph`, overrriding the `schedule_jobs()` with an lcl based schedule. This minimizes the maximum tardiness (based on the formula presented above) resulting from scheduling all of the jobs, while respecting the precedence constraints defined.

Iteration	Job Scheduled	Minimum Tardiness	Schedule
1	31	0	[31]
2	1	0	[1, 31]
5	2	65	[2, 15, 11, 1, 31]
6	5	44	[5, 2, 15, 11, 1, 31]
16	16	46	[16, 14, 29, 13, 28, 27, 26, 25, 24, 12, 5, 2, 15, 11, 1, 31]
25	7	0	[7, 23, 22, 6, 21, 20, 19, 18, 17, 16, 14, 29, 13, 28, 27, 26, 25, 24, 12, 5, 2, 15, 11, 1, 31]
Final (30)	30	0	[30, 10, 9, 4, 3, 8, 7, 23, 22, 6, 21, 20, 19, 18, 17, 16, 14, 29, 13, 28, 27, 26, 25, 24, 12, 5, 2, 15, 11, 1, 31]

Table 1: Least Cost Rule Schedule

The overall maximum tardiness achieved from this schedule is 65, from scheduling job 2.

2 Question 2

The cost function being considered for this question is ΣT_j which is the sum of the tardiness of each job in the schedule. The lcl algorithm is greedy, and does not take intermediate deviations from a local optimum that may lead to a global optimum in account. The tabu search algorithm is a heuristic which provides some tolerance to solutions that are worse than the current local optimum with the hope that they lead to a global optimum.

2.1 Part 1

The algorithm can be found in `src/tabu.py`. It is structured in a similar way to the lcl algorithm, where the `TabuGraph` algorithm inherits from `Graph`, and implements its own `schedule_jobs()`.
Table 2 shows the first 4 iterations. Although rather large, this is in line with the detail presented in the notes, and the requirements to illustrate steps where the cost function (ΣT_j) decreases. The ΣT_j for the initial schedule

$[30, 29, 23, 10, 9, 14, 13, 12, 4, 20, 22, 3, 27, 28, 8, 7, 19, 21, 26, 18, 25, 17, 15, 6, 24, 16, 5, 11, 2, 1, 31]$

is 341. Note that the code expects the initial schedule to be 0-indexed, but it is presented as 1-indexed in the output.

- For $L = 20$, $K = 10$ and $\gamma = 10$, the only improvement is found at iteration 10, where ΣT_j becomes 335.
  $x_{10} =$
  $[30, 23, 10, 9, 14, 13, 12, 4, 20, 22, 3, 29, 27, 28, 8, 7, 19, 21, 26, 18, 25, 17, 15, 6, 24, 16, 5, 11, 2, 1, 31]$
- For $L = 20$, $K = 100$ and $\gamma = 10$, there are several improvements detailed in table 2. The final lowest ΣT_j is 226 at iteration 99.
  $x_{100} =$
  $[30, 14, 4, 20, 23, 3, 13, 29, 28, 10, 9, 19, 8, 22, 21, 18, 17, 15, 7, 6, 16, 27, 11, 26, 25, 24, 12, 5, 2, 1, 31]$
- Similarly, for $L = 20$, $K = 1000$ and $\gamma = 10$, the lowest ΣT_j obtained is 196 at iteration 451.
  $x_{1000} =$
  $[30, 4, 23, 3, 22, 10, 14, 13, 9, 8, 7, 29, 6, 21, 27, 20, 19, 18, 17, 16, 28, 26, 25, 15, 11, 24, 12, 5, 2, 1, 31]$

There are instructions in the accompanying code and README.md to reproduce these results for each case.

2.2 Part 2

In the subsequent experiements, when γ is being varied, L is set to 20, and when L is being varied, γ is set to 10. As shown in figure 2, on increasing L it was found that the ΣT_j trended downwards, with occasional spikes in either direction. A local optimum solution of 163 was achieved for L equal to 120, 125, 130.
The reason for the decrease in ΣT_j is likely because with longer tabu lists, the search has a lower chance of getting stuck in a cycle (and if it does get stuck, the cycle's length is L). Short tabu lists allow quick returns to recently explored solutions This can create short cycling patterns, trapping the search in a suboptimal region. There is a sweet spot beyond which there are no returns from subsequent increases in the tabu list size. This is likely because the search becomes more prohibitive as the size of the tabu list increases. The main job of the tabu list in this algorithm is balancing cycle prevention and providing some leeway to re-explore avenues that may lead to a better solution after some iterations have passed.
In the case of γ, ΣT_j showed a general steep increase (with occasional deviations downwards), and after a point ΣT_j stagnated at 210. This is illustrated in figure 3.
Tolerance is responsible for ensuring solutions that may look sub-optimal at this stage are explored in hopes of reaching the global optimum. This helps the algorithm escape local optima by occasionally accepting worse solutions. For the given constraints, it is apparent that being stricter about accepting better solutions leads to the global optimum, while being more lenient tends to lead to a haphazard exploration of the search space which takes longer to converge (or never converges) on the global optimum. The best schedule found for this part is:

$x_{TS} = [30, 10, 4, 3, 23, 20, 19, 14, 9, 8, 7, 6, 22, 21, 18, 17, 16, 29, 13, 12, 27, 28, 26, 25, 24, 5, 2, 15, 11, 1, 31]$
with a total tardiness $\Sigma T_j = 158$. This was achieved with $L = 20$ and $\gamma = 1$.

3 List of Figures

Solution	**Lowest ΣT_j (g_{best})**	**Current Iteration Schedule**	**Tabu List (T)**
0	341	[30, 29, 23, 10, 9, 14, 13, 12, 4, 20, 22, 3, 27, 28, 8, 7, 19, 21, 26, 18, 25, 17, 15, 6, 24, 16, 5, 11, 2, 1, 31]	[]
1	341	[30, 29, 23, 10, 9, 14, 13, 12, 4, 20, 22, 3, 27, 28, 8, 7, 19, 21, 26, 18, 25, 17, 15, 6, 24, 16, 5, 11, 2, 1, 31]	[(23, 29)]
2	341	[30, 29, 23, 10, 9, 14, 13, 12, 4, 20, 22, 3, 27, 28, 8, 7, 19, 21, 26, 18, 25, 17, 15, 6, 24, 16, 5, 11, 2, 1, 31]	[(23, 29), (10, 29)]
3	341	[30, 29, 23, 10, 9, 14, 13, 12, 4, 20, 22, 3, 27, 28, 8, 7, 19, 21, 26, 18, 25, 17, 15, 6, 24, 16, 5, 11, 2, 1, 31]	[(23, 29), (10, 29), (9, 29)]
4	341	[30, 29, 23, 10, 9, 14, 13, 12, 4, 20, 22, 3, 27, 28, 8, 7, 19, 21, 26, 18, 25, 17, 15, 6, 24, 16, 5, 11, 2, 1, 31]	[(23, 29), (10, 29), (9, 29), (14, 29)]

10	335	[30, 23, 10, 9, 14, 13, 12, 4, 20, 22, 3, 29, 27, 28, 8, 7, 19, 21, 26, 18, 25, 17, 15, 6, 24, 16, 5, 11, 2, 1, 31]	[(23, 29), (10, 29), (9, 29), (14, 29), (13, 29), (12, 29), (4, 29), (20, 29), (22, 29), (3, 29)]
15	333	[30, 23, 10, 9, 14, 13, 12, 4, 20, 22, 3, 29, 28, 8, 7, 19, 21, 27, 26, 18, 25, 17, 15, 6, 24, 16, 5, 11, 2, 1, 31]	[(23, 29), (10, 29), (9, 29), (14, 29), (13, 29), (12, 29), (4, 29), (20, 29), (22, 29), (3, 29), (27, 28), (8, 27), (7, 27), (19, 27), (21, 27)]
16	323	[30, 23, 10, 9, 14, 13, 12, 4, 20, 22, 3, 29, 28, 8, 7, 19, 21, 27, 18, 26, 25, 17, 15, 6, 24, 16, 5, 11, 2, 1, 31]	[(23, 29), (10, 29), (9, 29), (14, 29), (13, 29), (12, 29), (4, 29), (20, 29), (22, 29), (3, 29), (27, 28), (8, 27), (7, 27), (19, 27), (21, 27), (18, 26)]
19	305	[30, 23, 10, 9, 14, 13, 12, 4, 20, 22, 3, 29, 28, 8, 7, 19, 21, 27, 18, 26, 17, 15, 6, 25, 24, 16, 5, 11, 2, 1, 31]	[(23, 29), (10, 29), (9, 29), (14, 29), (13, 29), (12, 29), (4, 29), (20, 29), (22, 29), (3, 29), (27, 28), (8, 27), (7, 27), (19, 27), (21, 27), (18, 26), (17, 25), (15, 25), (6, 25)]
20	300	[30, 23, 10, 9, 14, 13, 12, 4, 20, 22, 3, 29, 28, 8, 7, 19, 21, 27, 18, 26, 17, 15, 6, 25, 16, 24, 5, 11, 2, 1, 31]	[(23, 29), (10, 29), (9, 29), (14, 29), (13, 29), (12, 29), (4, 29), (20, 29), (22, 29), (3, 29), (27, 28), (8, 27), (7, 27), (19, 27), (21, 27), (18, 26), (17, 25), (15, 25), (6, 25), (16, 24)]
38	290	[30, 10, 9, 14, 13, 12, 4, 20, 23, 3, 29, 28, 8, 7, 19, 22, 21, 18, 27, 17, 15, 6, 26, 25, 16, 24, 11, 5, 2, 1, 31]	[(6, 25), (16, 24), (5, 11), (10, 23), (9, 23), (14, 23), (13, 23), (12, 23), (4, 23), (20, 23), (3, 22), (22, 29), (22, 28), (8, 22), (7, 22), (19, 22), (18, 27), (17, 26), (15, 26), (6, 26)]
39	272	[30, 10, 9, 14, 13, 12, 4, 20, 23, 3, 29, 28, 8, 7, 19, 22, 21, 18, 27, 17, 15, 6, 26, 16, 25, 24, 11, 5, 2, 1, 31]	[(16, 24), (5, 11), (10, 23), (9, 23), (14, 23), (13, 23), (12, 23), (4, 23), (20, 23), (3, 22), (22, 29), (22, 28), (8, 22), (7, 22), (19, 22), (18, 27), (17, 26), (15, 26), (6, 26), (16, 25)]
47	267	[30, 10, 14, 13, 12, 4, 20, 23, 3, 9, 29, 28, 8, 7, 19, 22, 21, 18, 27, 17, 15, 6, 26, 16, 25, 11, 24, 5, 2, 1, 31]	[(20, 23), (3, 22), (22, 29), (22, 28), (8, 22), (7, 22), (19, 22), (18, 27), (17, 26), (15, 26), (6, 26), (16, 25), (11, 24), (9, 14), (9, 13), (9, 12), (4, 9), (9, 20), (9, 23), (3, 9)]
53	265	[30, 10, 14, 13, 12, 4, 20, 23, 3, 29, 28, 9, 8, 19, 7, 22, 21, 18, 17, 15, 6, 27, 26, 16, 25, 11, 24, 5, 2, 1, 31]	[(19, 22), (18, 27), (17, 26), (15, 26), (6, 26), (16, 25), (11, 24), (9, 14), (9, 13), (9, 12), (4, 9), (9, 20), (9, 23), (3, 9), (9, 29), (9, 28), (7, 19), (17, 27), (15, 27), (6, 27)]
54	255	[30, 10, 14, 13, 12, 4, 20, 23, 3, 29, 28, 9, 8, 19, 7, 22, 21, 18, 17, 15, 6, 27, 16, 26, 25, 11, 24, 5, 2, 1, 31]	[(18, 27), (17, 26), (15, 26), (6, 26), (16, 25), (11, 24), (9, 14), (9, 13), (9, 12), (4, 9), (9, 20), (9, 23), (3, 9), (9, 29), (9, 28), (7, 19), (17, 27), (15, 27), (6, 27), (16, 26)]
62	250	[30, 14, 13, 12, 4, 20, 23, 3, 10, 29, 28, 9, 8, 19, 7, 22, 21, 18, 17, 15, 6, 27, 16, 26, 11, 25, 24, 5, 2, 1, 31]	[(9, 12), (4, 9), (9, 20), (9, 23), (3, 9), (9, 29), (9, 28), (7, 19), (17, 27), (15, 27), (6, 27), (16, 26), (11, 25), (10, 14), (10, 13), (10, 12), (4, 10), (10, 20), (10, 23), (3, 10)]
67	244	[30, 14, 13, 12, 4, 20, 23, 3, 29, 28, 10, 9, 19, 8, 22, 21, 7, 18, 17, 15, 6, 27, 16, 26, 11, 25, 24, 5, 2, 1, 31]	[(9, 29), (9, 28), (7, 19), (17, 27), (15, 27), (6, 27), (16, 26), (11, 25), (10, 14), (10, 13), (10, 12), (4, 10), (10, 20), (10, 23), (3, 10), (10, 29), (10, 28), (8, 19), (7, 22), (7, 21)]
68	238	[30, 14, 13, 12, 4, 20, 23, 3, 29, 28, 10, 9, 19, 8, 22, 21, 18, 7, 17, 15, 6, 27, 16, 26, 11, 25, 24, 5, 2, 1, 31]	[(9, 28), (7, 19), (17, 27), (15, 27), (6, 27), (16, 26), (11, 25), (10, 14), (10, 13), (10, 12), (4, 10), (10, 20), (10, 23), (3, 10), (10, 29), (10, 28), (8, 19), (7, 22), (7, 21), (7, 18)]
71	236	[30, 14, 13, 12, 4, 20, 23, 3, 29, 28, 10, 9, 19, 8, 22, 21, 18, 17, 15, 7, 6, 16, 27, 26, 11, 25, 24, 5, 2, 1, 31]	[(15, 27), (6, 27), (16, 26), (11, 25), (10, 14), (10, 13), (10, 12), (4, 10), (10, 20), (10, 23), (3, 10), (10, 29), (10, 28), (8, 19), (7, 22), (7, 21), (7, 18), (7, 17), (7, 15), (16, 27)]
76	234	[30, 14, 13, 4, 20, 23, 3, 12, 29, 28, 10, 9, 19, 8, 22, 21, 18, 17, 15, 7, 6, 16, 27, 11, 26, 25, 24, 5, 2, 1, 31]	[(10, 13), (10, 12), (4, 10), (10, 20), (10, 23), (3, 10), (10, 29), (10, 28), (8, 19), (7, 22), (7, 21), (7, 18), (7, 17), (7, 15), (16, 27), (11, 26), (4, 12), (12, 20), (12, 23), (3, 12)]
89	232	[30, 14, 13, 4, 20, 23, 3, 29, 28, 10, 9, 19, 8, 22, 21, 18, 17, 15, 7, 6, 12, 16, 27, 11, 26, 25, 24, 5, 2, 1, 31]	[(7, 15), (16, 27), (11, 26), (4, 12), (12, 20), (12, 23), (3, 12), (12, 29), (12, 28), (10, 12), (9, 12), (12, 19), (8, 12), (12, 22), (12, 21), (12, 18), (12, 17), (12, 15), (7, 12), (6, 12)]

90	230	[30, 14, 13, 4, 20, 23, 3, 29, 28, 10, 9, 19, 8, 22, 21, 18, 17, 15, 7, 6, 16, 12, 27, 11, 26, 25, 24, 5, 2, 1, 31]	[(16, 27), (11, 26), (4, 12), (12, 20), (12, 23), (3, 12), (12, 29), (12, 28), (10, 12), (9, 12), (12, 19), (8, 12), (12, 22), (12, 21), (12, 18), (12, 17), (12, 15), (7, 12), (6, 12), (12, 16)]
95	229	[30, 14, 13, 4, 20, 23, 3, 29, 28, 10, 9, 19, 8, 22, 21, 18, 17, 15, 7, 6, 16, 27, 11, 26, 25, 24, 12, 5, 2, 1, 31]	[(3, 12), (12, 29), (12, 28), (10, 12), (9, 12), (12, 19), (8, 12), (12, 22), (12, 21), (12, 18), (12, 17), (12, 15), (7, 12), (6, 12), (12, 16), (12, 27), (11, 12), (12, 26), (12, 25), (12, 24)]
99	226	[30, 14, 4, 20, 23, 3, 13, 29, 28, 10, 9, 19, 8, 22, 21, 18, 17, 15, 7, 6, 16, 27, 11, 26, 25, 24, 12, 5, 2, 1, 31]	[(9, 12), (12, 19), (8, 12), (12, 22), (12, 21), (12, 18), (12, 17), (12, 15), (7, 12), (6, 12), (12, 16), (12, 27), (11, 12), (12, 26), (12, 25), (12, 24), (4, 13), (13, 20), (13, 23), (3, 13)]
112	223	[30, 14, 4, 20, 23, 3, 29, 28, 10, 9, 19, 8, 22, 21, 18, 17, 15, 7, 6, 13, 16, 27, 11, 26, 25, 24, 12, 5, 2, 1, 31]	[(12, 26), (12, 25), (12, 24), (4, 13), (13, 20), (13, 23), (3, 13), (13, 29), (13, 28), (10, 13), (9, 13), (13, 19), (8, 13), (13, 22), (13, 21), (13, 18), (13, 17), (13, 15), (7, 13), (6, 13)]
113	220	[30, 14, 4, 20, 23, 3, 29, 28, 10, 9, 19, 8, 22, 21, 18, 17, 15, 7, 6, 16, 13, 27, 11, 26, 25, 24, 12, 5, 2, 1, 31]	[(12, 25), (12, 24), (4, 13), (13, 20), (13, 23), (3, 13), (13, 29), (13, 28), (10, 13), (9, 13), (13, 19), (8, 13), (13, 22), (13, 21), (13, 18), (13, 17), (13, 15), (7, 13), (6, 13), (13, 16)]
122	217	[30, 4, 20, 23, 3, 14, 29, 28, 10, 9, 19, 8, 22, 21, 18, 17, 15, 7, 6, 16, 27, 11, 26, 25, 24, 13, 12, 5, 2, 1, 31]	[(9, 13), (13, 19), (8, 13), (13, 22), (13, 21), (13, 18), (13, 17), (13, 15), (7, 13), (6, 13), (13, 16), (13, 27), (11, 13), (13, 26), (13, 25), (13, 24), (4, 14), (14, 20), (14, 23), (3, 14)]
344	212	[30, 23, 20, 19, 14, 4, 29, 22, 21, 3, 18, 10, 13, 12, 9, 8, 7, 28, 6, 17, 16, 27, 26, 25, 24, 5, 15, 2, 11, 1, 31]	[(15, 27), (15, 26), (15, 25), (15, 24), (5, 15), (2, 11), (20, 22), (19, 22), (14, 22), (4, 22), (22, 29), (3, 18), (10, 17), (13, 17), (12, 17), (9, 17), (8, 17), (7, 17), (17, 28), (6, 17)]
346	209	[30, 23, 20, 19, 14, 4, 29, 22, 21, 3, 18, 10, 13, 12, 9, 8, 7, 28, 6, 17, 27, 16, 26, 25, 24, 5, 2, 15, 11, 1, 31]	[(15, 25), (15, 24), (5, 15), (2, 11), (20, 22), (19, 22), (14, 22), (4, 22), (22, 29), (3, 18), (10, 17), (13, 17), (12, 17), (9, 17), (8, 17), (7, 17), (17, 28), (6, 17), (16, 27), (2, 15)]
352	207	[30, 20, 19, 14, 4, 29, 23, 22, 3, 21, 18, 10, 13, 12, 9, 8, 7, 28, 6, 17, 27, 16, 26, 25, 24, 5, 2, 15, 11, 1, 31]	[(14, 22), (4, 22), (22, 29), (3, 18), (10, 17), (13, 17), (12, 17), (9, 17), (8, 17), (7, 17), (17, 28), (6, 17), (16, 27), (2, 15), (20, 23), (19, 23), (14, 23), (4, 23), (23, 29), (3, 21)]
365	206	[30, 20, 14, 4, 29, 19, 23, 3, 22, 21, 10, 13, 12, 9, 8, 7, 28, 6, 18, 27, 17, 16, 26, 25, 24, 5, 2, 15, 11, 1, 31]	[(2, 15), (20, 23), (19, 23), (14, 23), (4, 23), (23, 29), (3, 21), (10, 18), (13, 18), (12, 18), (9, 18), (8, 18), (7, 18), (18, 28), (6, 18), (17, 27), (14, 19), (4, 19), (19, 29), (3, 22)]
436	203	[30, 4, 23, 3, 29, 22, 10, 14, 13, 9, 8, 7, 28, 6, 21, 27, 20, 19, 18, 17, 16, 26, 25, 15, 11, 24, 12, 5, 2, 1, 31]	[(9, 12), (8, 12), (7, 12), (12, 28), (6, 12), (12, 21), (12, 27), (12, 20), (12, 19), (12, 18), (12, 17), (12, 16), (12, 26), (12, 25), (12, 15), (11, 12), (12, 24), (4, 29), (23, 29), (3, 29)]
445	202	[30, 4, 23, 3, 22, 10, 14, 13, 9, 8, 7, 29, 6, 21, 28, 27, 20, 19, 18, 17, 16, 26, 25, 15, 11, 24, 12, 5, 2, 1, 31]	[(12, 18), (12, 17), (12, 16), (12, 26), (12, 25), (12, 15), (11, 12), (12, 24), (4, 29), (23, 29), (3, 29), (22, 29), (10, 29), (14, 29), (13, 29), (9, 29), (8, 29), (7, 29), (6, 28), (21, 28)]
449	199	[30, 4, 23, 3, 22, 10, 14, 13, 9, 8, 7, 29, 6, 21, 27, 20, 19, 18, 28, 17, 16, 26, 25, 15, 11, 24, 12, 5, 2, 1, 31]	[(12, 25), (12, 15), (11, 12), (12, 24), (4, 29), (23, 29), (3, 29), (22, 29), (10, 29), (14, 29), (13, 29), (9, 29), (8, 29), (7, 29), (6, 28), (21, 28), (27, 28), (20, 28), (19, 28), (18, 28)]
451	196	[30, 4, 23, 3, 22, 10, 14, 13, 9, 8, 7, 29, 6, 21, 27, 20, 19, 18, 17, 16, 28, 26, 25, 15, 11, 24, 12, 5, 2, 1, 31]	[(11, 12), (12, 24), (4, 29), (23, 29), (3, 29), (22, 29), (10, 29), (14, 29), (13, 29), (9, 29), (8, 29), (7, 29), (6, 28), (21, 28), (27, 28), (20, 28), (19, 28), (18, 28), (17, 28), (16, 28)]
1000	196	[30, 4, 23, 3, 22, 10, 14, 13, 9, 8, 7, 29, 6, 21, 27, 20, 19, 18, 17, 16, 28, 26, 25, 15, 11, 24, 12, 5, 2, 1, 31]	[(9, 17), (9, 15), (9, 11), (7, 27), (7, 28), (7, 12), (10, 14), (10, 23), (10, 22), (10, 21), (4, 10), (10, 13), (10, 20), (10, 19), (3, 10), (10, 18), (10, 29), (10, 17), (10, 15), (10, 11)]

Table 2: Tabu search for 3.2.1 with tolerance 10, list length 20 up to a 1000 iterations

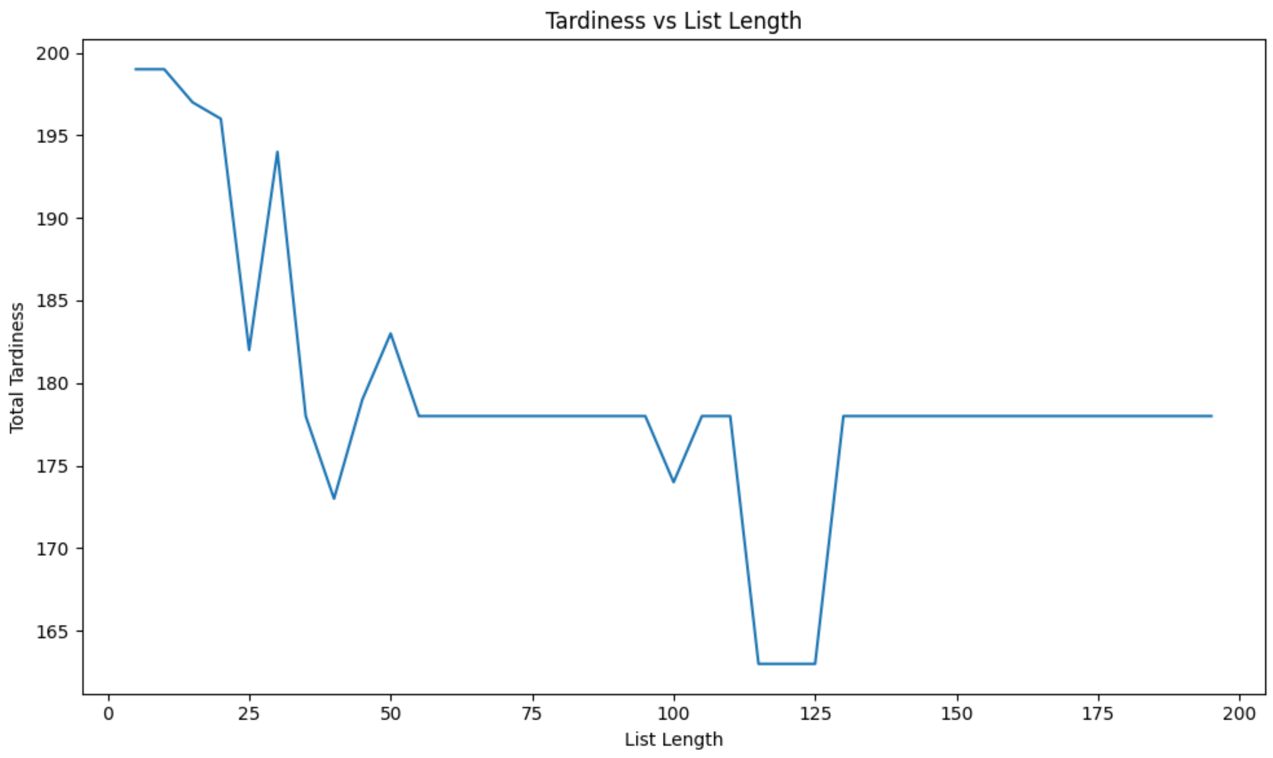


Figure 2: Effect of L on ΣT_j

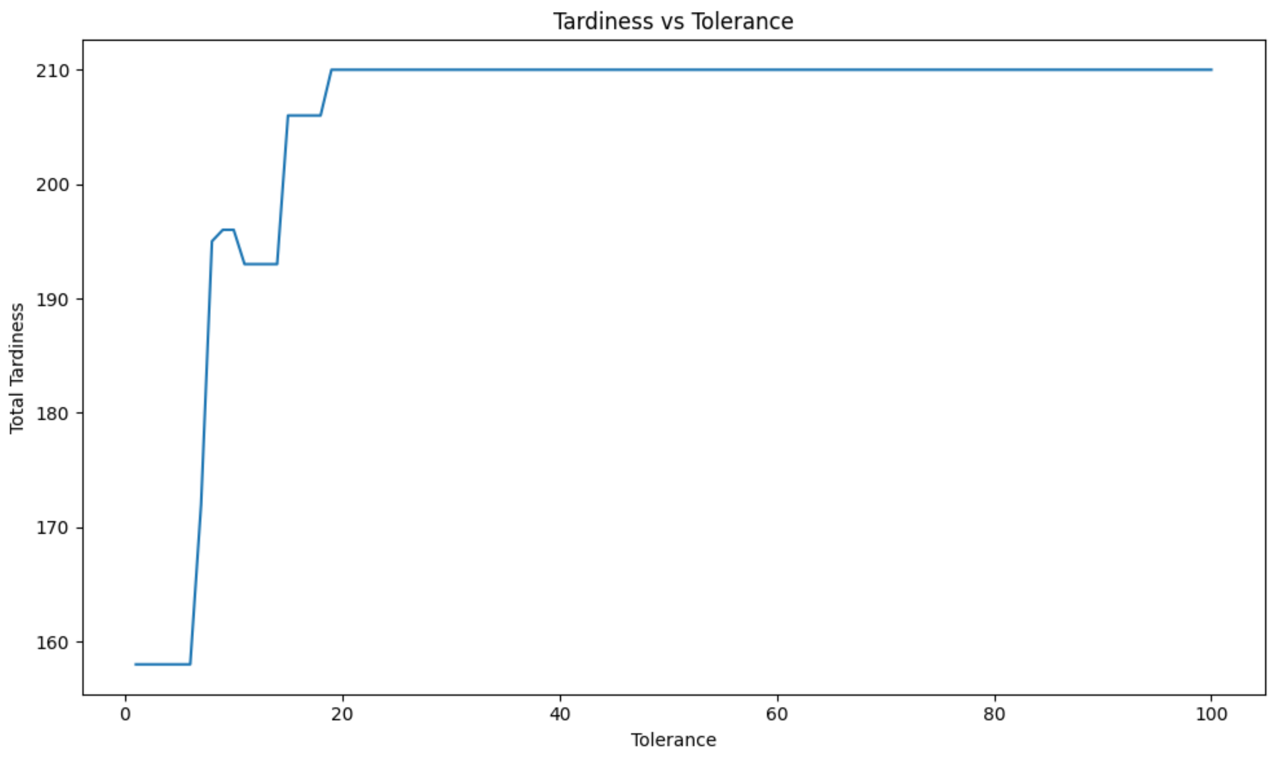


Figure 3: Effect of γ on ΣT_j