# Report on “Bounding the Response Time of DAG Tasks Using Long Paths”

John Doe[*]

May 20, 2024

## 1 Part 1

### 1.1 Description of Algorithm

DAG (directed acyclic graph) is used to model task scheduling. He et al. published “Bounding the Response Time of DAG Tasks Using Long Paths”(hereinafter referred to as *”the paper”*) in 2022 [2], which aims to provide a tighter bound of the Graham’s Bound [1]. Graham’s bound assumes all workloads other than the longest path delays the longest path. Thus, the bound is the sum of the longest path and the time taken to run other workloads ($len(G)$ is the length of longest path of $G$, $vol(G)$ is sum of all workloads of $G$, and $m$ is the number of processors):

$$R \leq len(G) + \frac{vol(G) - len(G)}{m} \tag{1.1}$$

This bound is “pessimistic”, because it considers all workloads not in the longest path to delay the longest path, while it is trivial to see that in many scenarios it is not true. The paper tighten the bound by identifying the workloads that can be executed parallel to the longest path.

To accomplish this, we need an algorithm to identify sets of vertices that satisfy:

- all elements of a set executes sequentially, and
- all sets does not share vertices.

The paper identifies that for a set of vertices to satisfy the aforementioned requirement, they need not form a legitimate path. A ”virtual path”, which is a set of elements with a specified execution sequence can fulfill the requirement. This approach tightens the bound but depends on an unknown beforehand execution sequence. The concept of a *Generalized Path* is proposed, which is a subset of vertices within a valid path in a DAG. It extends the notion of the longest path to this generalized path. By applying ”workload swapping,” which bases on the idea that certain swaps in the execution sequence do not affect the expected execution time, the paper proves the existence of a virtual path list that can be combined with a restricted critical path to form certain valid execution sequence, generalization of the special case. The paper further proves that a generalized path list containing the longest path is always bounded by some virtual path list, thus using such a generalized path list provides a practical bound on response time ($\lambda_i$

[*]Since it is required to be put in public. Using anonymous name.

denotes the generalized path list):

$$R \leq \min_{j \in [0,k]} \left\{ len(G) + \frac{vol(G) - \sum_{i=0}^{j} len(\lambda_i)}{m - j} \right\} \tag{1.2}$$

From the bound we can see that larger $len(\lambda_i)$ and smaller $k$ lead to tighter bounds, thus the list is generated by always finding the longest path remaining.

### 1.2 Generate random schedule

To add randomization to schedule with least overhead:

- As Eq. 1.2 shows, avoid the longest path, and add randomization to other workload, especially the other path in generalized path lists, since they are run parallel with the longest path while taking shorter time than it, leaving room for random waiting.
- Careful randomization might still be needed for longest path, as the longest path is likely to contain the beginning and ending of tasks' life circle, which is likely to be vulnerable, e.g. I/O. This can be done basing on the schedule of delaying.
- Develop real-time method of workload swapping. This is the method used for algorithm analysation in the paper, but it is not impossible to develop a practical swapping method.

## 2 Code

This section briefly explains the code and several assumptions.

Input (data.txt) is in following format:
<number of matrices>
<number of vertices>
<matrix 1>
<number of vertices>
<matrix 2>

Number of matrices and vertices is added for convenience, and it is the de facto standard input format for most algorithm competitions, e.g. ACM-ICPC.

DAGs are stored in adjacency list because input (and likely most DAG tasks) are sparse graph, adjacency list is significantly more space efficient than adjacency matrix ($O(|V| + |E|)$ vs. $O(|V|^2)$).

The input has already been topological sorted, thus the algorithm to find the longest path is based on this assumption. Since the graph is guarentined to be DAG, it is trivial to implement topological sorting algorithms, e.g. Kahn's algorithm.

## 3 Code Reflection

The output is as output.txt in main folder.

By uncommenting the prinf lines in the code, one can get a more verbose output as output-verbose.txt, and as Table 1.

One immediate side-reflection on data is that although WCET for each node is randomized, since the longest path is the sum of multiple nodes, they form normal distribution and end up in similar length.

Table 1: Output data of DAG G1 and G2.

| Longest path | Vol | Response Time |
|---|---|---|
| 30 | 39 | 34 |
| 27 | 38 | 32 |
| 25 | 30 | 27 |
| 19 | 31 | 25 |
| 21 | 35 | 28 |
| 21 | 28 | 24 |
| 29 | 42 | 35 |
| 23 | 36 | 29 |
| 21 | 34 | 27 |
| 22 | 35 | 28 |
| Average | | |
| 23.8 | 34.8 | 28.9 |

| Longest Path | Vol | Response Time |
|---|---|---|
| 22 | 45 | 33 |
| 28 | 50 | 39 |
| 20 | 48 | 34 |
| 35 | 62 | 48 |
| 20 | 32 | 26 |
| 28 | 54 | 41 |
| 31 | 52 | 41 |
| 20 | 44 | 32 |
| 31 | 54 | 42 |
| 17 | 43 | 30 |
| Average | | |
| 25.2 | 48.4 | 36.6 |

It is also noticed that the Volume/Response ratio in G2 is smaller than G1. The reason is G2 is more parallel than G1, which produces a smaller $len(G)/vol(G)$ ratio and a larger $\sum len(\lambda)$. The topological difference also causes that the response time of G1 is more sensitive to change of the length of longest path than G2.

## References

[1] R. L. Graham. “Bounds on Multiprocessing Timing Anomalies”. In: *SIAM Journal on Applied Mathematics* 17.2 (Mar. 1969), pp. 416–429. ISSN: 0036-1399. DOI: 10.1137/0117039. (Visited on 05/20/2024).

[2] Qingqiang He et al. “Bounding the Response Time of DAG Tasks Using Long Paths”. In: *2022 IEEE Real-Time Systems Symposium (RTSS)*. Dec. 2022, pp. 474–486. DOI: 10.1109/RTSS55097.2022.00047. (Visited on 05/01/2024).