Application of Graphene Scheduling Method on Public Datasets

James Zhang
Columbia University
New York, U.S.
tz2642@columbia.edu

Haiyu Wei
Columbia University
New York, U.S.
hw3036@columbia.edu

Zhanfei Qiu
Columbia University
New York, U.S.
zq2238@columbia.edu

Mike Yang
Columbia University
New York, U.S.
ty2467@columbia.edu

ABSTRACT

Cluster scheduling is a critical technique for resource management in modern data centers. This study addresses the scheduling of tasks in complex directed acyclic graphs (DAGs) by analyzing public datasets from Google, Alibaba, and Azure. We examine an improved scheduling method, Graphene[1], and a new criteria, New Lower Bound (NewLB), which combines critical path, total work, and modified critical path metrics to optimize task execution by packing necessary tasks together. Experiments show that NewLB performs well on complex DAGs (levels 10–30) in the Alibaba dataset but has limited applicability for simple or sparsely dependent DAGs, such as those in Google datasets. Additionally, while Graphene is effective for multi-branch DAGs, it struggles with complex DAGs lacking clear bottlenecks. This research provides insights into current scheduling methods with the combination of real public datasets to evaluate its effectiveness on data center performance.

1 INTRODUCTION

1.1 Background Information

As modern data centers grow in size and complexity, efficient cluster management has become a critical challenge. Scheduling tasks with complex dependencies, particularly in the form of directed acyclic graphs (DAGs), poses significant obstacles to optimizing resource utilization and execution efficiency. Google's Borg system [2] achieved remarkable resource utilization through efficient task packing, resource sharing, and isolation mechanisms. However, it has limitations when handling DAGs with complex dependency structures.

To address these challenges, the Graphene framework [1] introduced a dependency-aware scheduling strategy that focuses on optimizing resource allocation and task dependencies. By constructing offline priority orders for DAG tasks and enforcing them online, Graphene significantly improves median job completion time and cluster throughput, making it the stat-of-the-art scheduler method. However, this framework was only tested and evaluated on its internal clusters, instead of the large public clusters. Therefore, building on these foundations, this study applies existing scheduling frameworks and evaluates the effectiveness of the method combining metrics such as Critical Path, Total Work, Modified Critical Path, and New Lower Bound to optimize the execution of complex DAG tasks from public datasets.

1.2 Motivation

1.2.1 Sustainability. Modern data centers are significant consumers of energy, contributing to rising carbon emissions globally. As computational workloads grow in complexity, there is an increasing need to prioritize sustainability in data center operations. Many organizations aim to minimize their environmental impact by optimizing task scheduling to align with low-carbon intensity periods, reducing energy waste, and improving overall energy efficiency. Inspired by prior work such as carbon-aware scheduling strategies [3], this project incorporates sustainability considerations by proposing task scheduling methods that balance performance with reduced carbon footprints. By leveraging NewLB metrics and resource-aware strategies, our methods enable data centers to maintain operational efficiency while minimizing their environmental impact.

1.2.2 Energy Efficiency. Energy efficiency is a critical challenge in data centers, where resource fragmentation, underutilization, and inefficient task allocation often lead to substantial energy waste. Previous studies, such as Graphene [1], have demonstrated the importance of addressing resource

Conference'17, July 2017, Washington, DC, USA
2024. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

heterogeneity and dependency management to optimize energy use. Building on these insights, this project emphasizes efficient resource allocation by combining Critical Path analysis, Total Work, and Modified Critical Path into the Graphene model on public datasets. This allows for better packing of tasks, reducing idle resource states, and improving the energy-to-task ratio. By aligning scheduling strategies with energy-efficient objectives, our approach contributes to reduced operational costs and enhanced sustainability in data center operations.

1.3 Problem Definition

This project addresses the following challenges:

(1) **Optimization of Complex DAG Scheduling:** Scheduling tasks with intricate dependency structures remains a significant challenge. Borg and Graphene offer valuable insights into dependency-aware and resource-efficient scheduling but leave room for improvement [1, 2].
(2) **Resource Fragmentation in Task Scheduling:** Current scheduling algorithms often fall short in resource packing and parallelization, leading to underutilized cluster resources.
(3) **Trade-off Between Fairness and Performance:** In multi-task environments, fairness and task completion time often conflict, requiring careful balancing.

To tackle these issues, we applied the Graphene Method[1], integrating Critical Path analysis, Total Work, and Modified Critical Path to predict the NewLB, which helps to better predict the optimal runtime of a job. To apply this to public datasets, we first identify the bottlenecks of the DAG and evaluate its effectiveness based on the comparison of this NewLb and the other three metrics.

1.4 Target Audience

The efficient scheduling of tasks in modern data centers requires collaboration across different fields, including academia, industry, and algorithm development. This project aims to address the challenges of resource utilization, task dependency optimization, and scheduling fairness, making it relevant to a diverse group of stakeholders. The primary target audience for this project includes:

- **Academic Researchers:** Experts studying data center scheduling, resource management, and task dependency optimization.
- **Data Center Operators:** Professionals seeking to enhance resource utilization and reduce task latency.
- **Algorithm Developers:** Engineers specializing in developing scheduling algorithms for complex DAGs and performance optimization.

2 PROJECT DESIGN

2.1 Algorithm Development

2.1.1 Graphene Model. The Graphene model is a comprehensive framework for optimizing the scheduling of Directed Acyclic Graphs (DAGs). It addresses bottlenecks in task execution and improves overall efficiency by integrating three core metrics:

- **Critical Path (CPLength)**:
  The length of the longest path in a DAG, representing the lower bound of task execution time. CPLength neglects tasks outside the critical path, which makes it insufficient for DAGs with high complexity and branching.
- **Total Work (TWork)**: The total amount of resources required to complete all tasks, including compute, memory, and bandwidth. While TWork assumes unlimited resources and serves as an ideal lower bound, it does not account for scheduling overheads caused by dependencies.
- **Modified Critical Path (ModCP)**: Selecting a node on the critical path and calculating:

$$\text{ModCP} = \max(\text{TWork}, \text{CPLength}) + \text{Duration of remaining nodes.}$$

  ModCP combines the characteristics of TWork and CPLength, providing a more accurate representation of scheduling bottlenecks.
- **Graphene Model (Graphene)**: The DAG is split into two parts, and the maximum value among CPLength, TWork, and ModCP is chosen as the optimization basis. Graphene considers not only tasks on the critical path but also other distributed tasks and their resource demands, addressing the limitations of CPLength and TWork.

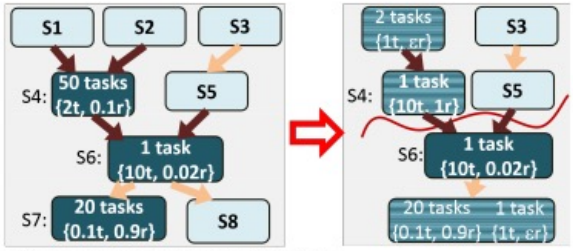

Figure 18: For the DAG on left, the modified DAG used by our lower bound is on the right. Lower bound improves from $14t$ to $22.8t$. Edge types are as per the legend in Figure 4.

Figure 1: Example on NewLb[2]

2.1.2 Computation Method. The computation process of the Graphene model is designed to optimize the scheduling of complex DAGs through the following steps:

(1) **Identify the Bottleneck (Neck)**: Locate the bottleneck in the DAG, which significantly impacts execution efficiency. The DAG is split at this neck, dividing it into two independent parts to reduce dependency constraints. This bottleneck stage, s, is defined from previous research[1] such that $U(s)$ is an empty set as:

$$U(s) = V - A(s) - D(s) - \{s\} = \emptyset \quad (1)$$

Here:
- V: Represents the vertex set of the DAG G,
- $A(s)$: Represents the set of ancestor nodes to stage s,
- $D(s)$: Represents the set of descendant nodes for stage s,
- $\{s\}$: Refers to the stage being considered.

(2) **Prioritize Critical Tasks**: Use resource-aware and task-priority strategies to ensure that critical tasks are executed before non-critical tasks. Optimize task packing to reduce resource fragmentation and increase utilization.
(3) **Dynamic Scheduling Adjustments**: In complex DAG scenarios, dynamically adjust task execution order to respect dependencies. By extending the Graphene framework, the Graphene model is applied in multi-task environments to improve robustness and efficiency.

2.1.3 Analysis Method. To analyze these datasets, the following steps were undertaken:

(1) **Dependency Modeling:** Parent-child relationships and start-after relationships were used to construct DAGs. Bottlenecks were identified by analyzing the shortest time intervals between task completions and subsequent task starts [4, 5].
(2) **Resource Utilization Analysis:** CPU and memory usage were calculated for each task. Total Work (TWork) was computed by summing up resource usage across DAGs [6]. Critical Path (CPLength) was calculated as the maximum execution time of any path in a DAG [4, 7].
(3) **Validation and Testing:** The Graphene method[1] was applied to optimize task scheduling. Results were validated across multiple levels of DAG complexity, revealing its strengths in Alibaba datasets compared to Google data [5, 6].

3 RESULTS&ANALYSIS

3.1 Dataset Overview

Efficient cluster scheduling requires a thorough understanding of the datasets used to model and analyze directed acyclic graphs (DAGs). In this project, we analyze datasets from Google, Alibaba, and Azure, each of which presents unique characteristics and challenges.

3.2 Google Cluster Dataset

The Google Cluster dataset (2019) serves as an initial benchmark for evaluating scheduling strategies. This dataset includes task metadata such as CPU usage, task duration, and parent-child relationships. However, several limitations were identified:

- **Limited Dependency Information:** Only 54 users in `clusterdata_2019_a` and 41 users in `clusterdata_2019_b` have tasks with dependencies out of thousands users in total. As shown in Figure 2 Similar percentage was also found in the rest of the clusters. Most tasks appear as single nodes in the DAG structure, making it challenging to apply dependency-aware methods like NewLB.

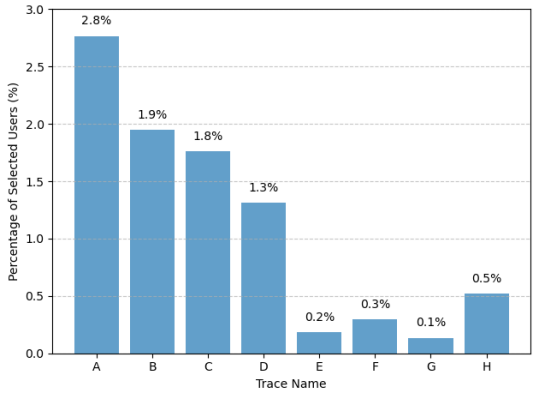

Figure 2: Percentage of users with potential complex structure of DAGs

- **Simplistic DAGs:** Many DAGs in this dataset are too simple, with minimal branching or hierarchy. NewLB provided a 5.60% improvement in scheduling efficiency but failed to significantly impact non-cuttable DAGs (e.g., single-node DAGs) [4, 7].

Despite these limitations, the Google cluster dataset provides valuable insights into resource allocation patterns, such as CPU and memory usage. Analyzing this data has laid the groundwork for more complex DAG models.

Therefore, we took a sample user with the most number of tasks processed in `clusterdata_2019_a` since this cluster has the most users with potential complex DAGs. Then, we grouped DAG based on its `parent_collection_id` and `start_after_collection_ids`[2, 4, 8]. Despite the large number of DAGs and the significant amount of information they contain, the maximum hierarchical depth of any DAG is surprisingly just 3 levels.

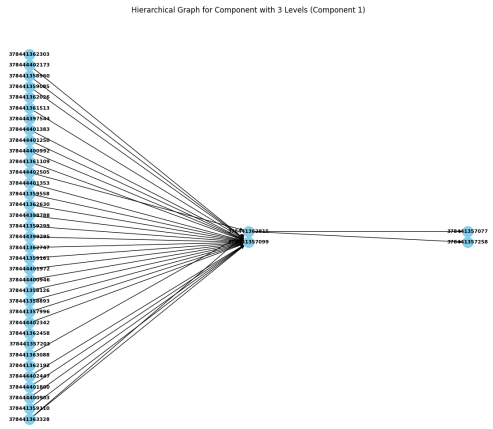

Figure 3: Hierarchical Graph of the DAG with most information

Although the number of nodes is large, most nodes are individually connected to two specific nodes, which makes this DAG a simple structure. Next, we analyzed all of the DAGs the sample user processed, and we obtained the CPLength and their NewLB.

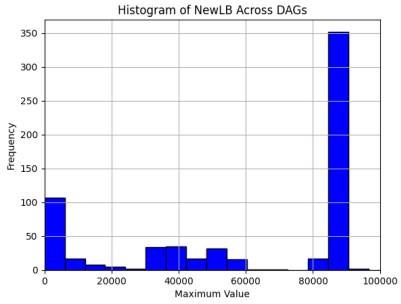

Figure 4: Histogram of NewLB of the DAGs within the sample user

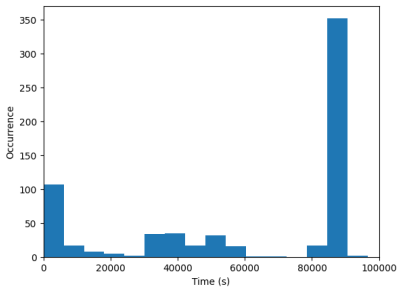

Figure 5: Histogram of CPLength of the DAGs within the sample user

Compared between Figure 5 and Figure4, we can find no change between its CPLength and NewLB, indicating that the Graphene Method doesn't improve these types of DAG structures, implying its ineffectiveness in simple DAG. Previous research also reported its DAG information is too simple for sophisticated analysis, along with their only accessible tool, BigQuery, which is not very friendly for researchers outside Google.[9–11]. Since the user with most complex DAG in its largest cluster doesn't have much useful information, along with inconvenience of the access to the whole dataset, we put more emphasis on the other two public datasets.

3.3 Azure Cluster Dataset

Lower Bound is attempted to calculate on the Azure Cluster Dataset.

- **DAGs form at Tenant Level**: the dependency within the data set is not made clear. Therefore estimate method is used leveraging two factors: 1. starttime and end time, and 2. tenant id. We assume that data within a tenant can form a dag. And those with start time closer to within 2 seconds of the end time of other intra-tenant tasks are dependent on them [5].
- **Methodology and comment:**: CPlen, Twork, and ModCP is calculated for each DAG, and the maximum of them is designed to calculate the new lower bound. The challenges here is mostly again with certain missing data. Specifically, data missing on the resources requested by each of the tasks. As such, a heuristic is employed again. Tasks are mapped to vms by their vmId, and thus the resources requested on the vms substitute in for the tasks. With this, we provisionally constructed resource information for all tasks. And unfortunately, the missing data caused the metrics of most DAGs to be uncomputable.
- **Summary of data:** As to provide helpful insight for future research, summary information about the azure traces are provided in the next bullet points. [5].

- **Dag Dependency** The average depth of dag in azure is 2. Below is an example of the DAG with id "1439322". The shallow nature of the dags is reflected by these dependency graphs. For only one outbound/inbound edge exist between tasks who we presume to be connected to one another. [5].

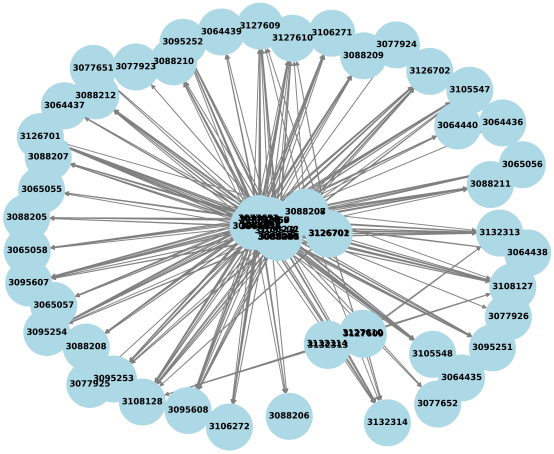

Figure 6: Example dependency graphs of DAGs with ID "1439322". This graph demonstrates the overall shallow trend of DAG dependencies in Azure.

- **Dag Size** The size of most DAGs is also rather modest. With only one. This is reflected in the graph below. [5].

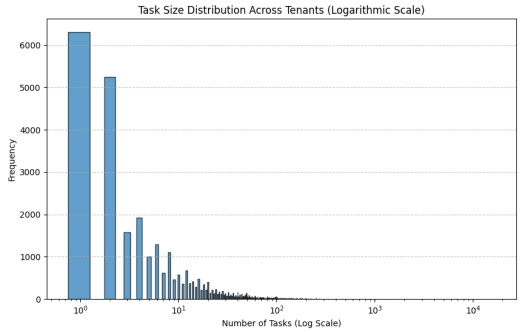

Figure 7: Distribution of size of DAGs in the Azure Cluster Data Set. As seen, most DAGs have single or double digit size (amount of tasks)

3.4 Alibaba Cluster Dataset

The Alibaba dataset offers a more structured and detailed view of DAGs, making it suitable for testing advanced scheduling algorithms. Key observations include:

- **Complex DAG Structures:** This dataset contains multi-level DAGs with well-defined dependencies. For instance, DAGs with up to 53 levels were identified, showcasing a wide range of complexities [6].
- **Effective Use of NewLB:** In levels between 10 and 30, NewLB outperforms Critical Path (CPLength), significantly optimizing scheduling efficiency for 15% of DAGs. The runtime improvement for these DAGs averages around 20% [6].
- **Hierarchical Analysis:** By grouping DAGs by levels and identifying bottlenecks, the dataset enables the application of cutting-edge algorithms like Graphene [6].

Alibaba cluster trace v2018 was used for the analysis(cite Alibaba cluster) because it had ample details, including jobs, tasks, and their relationships, to form a DAG. Specifically, we chose batch instances and batch task tables for analysis.

Metric	Counts
Total Nodes	12004350
Total DAGs	3175025
Maximum DAG Depth	53
Median DAG Depth	14.50
Average Task Time(s)	66.74
Maximum Task Time(s)	583886
Average CPU Requested	83.87
Maximum CPU Requested	1000.00
Average Memory Requested	0.34
Maximum Memory Requested	17.17

Table 1: Statistics of the Batch Information

Table 1 summarizes the basic statistics of the DAG information after we filtered out the tasks that are not considered DAGs and invalid values due to confidential reasons(cite Alibaba cluster). After we grouped all of the tasks into DAGs, we then classified them based on their hierarchy depth.

Based on Figure8, we discovered that 92.18% of the DAGs have levels less than 5, and 99.63% of the DAGs have levels less than 10. The DAG levels show a continuous sequence from 0 to 23, followed by a set of isolated levels: 25, 28, 31, 33, and 53. Particularly at a higher end of the range, the number of DAGs at each level is very few, generally less than 100, while DAGs at the same levels share a similar shape.

Next, we identified the bottleneck point at each DAG to prepare for the application of Graphene. As described in Section 2.1.2, the identification of a bottleneck point within a DAG is crucial. However, not all DAGs in the Alibaba Cluster Dataset contain such a point. Our analysis revealed that only 452,445 DAGs included a bottleneck point, accounting

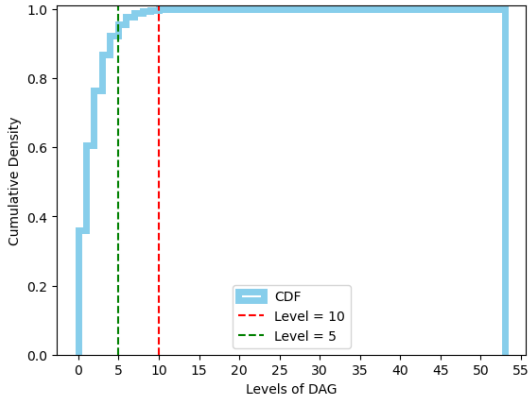

Figure 8: Cumulative Density Function (CDF) of the DAG based on their hierarchial levels

for 14.25% of total DAGs. Then, we calculated four metrics mentioned in the Graphene Model.

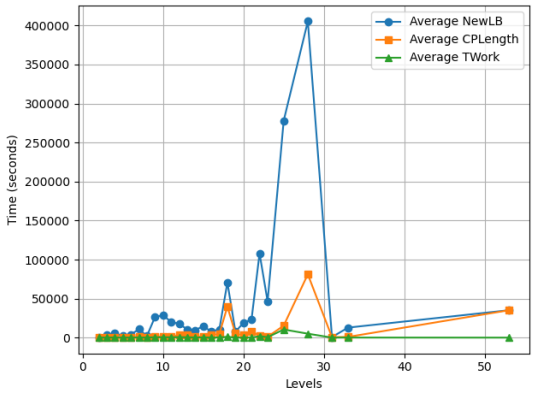

Figure 9: Average Time of NewLB, Twork, CPLength by levels of DAGs

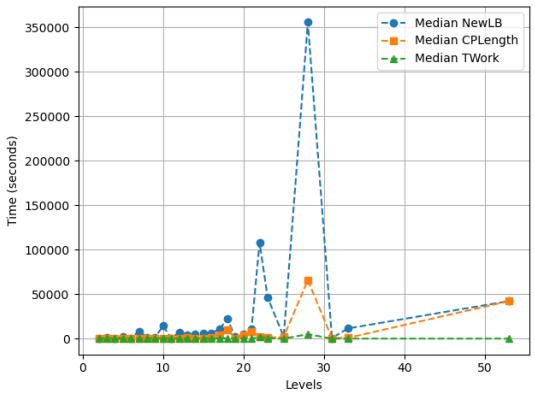

Figure 10: Median Time of NewLB, Twork, CPLength by levels of DAGs

Figure 9 and Figure 10 reveal that the NewLB of DAGs increases greatly with the level of DAGs between 20 and 30, whereas the increase in NewLB is subtle for levels between 0 and 10. Also, the discrepancy between Figure 9 and Figure 10 at levels between 10 and 20 shows that NewLB does increase some sort of DAGs, but not all of them. Also, if we focused on the Twork for both figures, we found that the Twork metric is far smaller than the other two metrics, which means the resource-weighted runtime is not significant.

It is evident that NewLB increases the runtime of a DAG when its CPLength exceeds approximately 40 seconds based on Figure 11. Additionally, the longer the CPLength, the larger the NewLB becomes. While 80% of DAGs have a CPLength shorter than 1000 seconds, this percentage drops to 40% when considering NewLB. Furthermore, more DAGs have a NewLB between 1000 and 10,000 seconds compared to their corresponding CPLength in this range.

Next, we show the normalized resource utilization of the DAGs based on their levels, and there are two resources available: CPU and Memory[12].

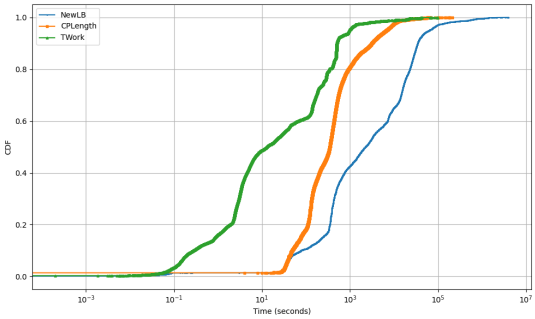

Figure 11: Cumulative Density Function of NewLB, Twork, CPLength of DAGs in time space

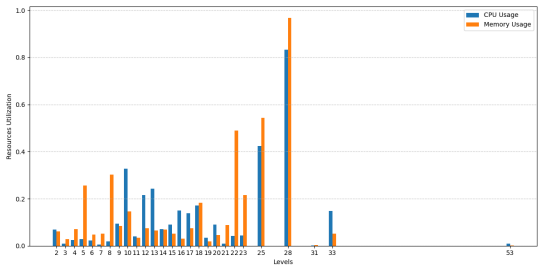

Figure 12: Normalized Average Resources Utilizations based on levels of DAGs

Figure12 shows a similar trend that the resource is consumed most with levels of DAGs between 20 and 30, with level of 28 reaches the maximum. This trend agrees with Figure 9 and Figure 10 that the NewLB integrated both the length and resource consumption of a DAG. Three levels of DAG, 5, 8, and 10 also show a large resource utilization while its NewLB is relatively low, which may be due to the simple structure of the DAG. This agrees with our observation that graphene yields a better result in complex structures.

Now, we know that DAGs with 28 levels are the most resource-consuming and run-time-consuming DAGs, so we take a sample job 567380.

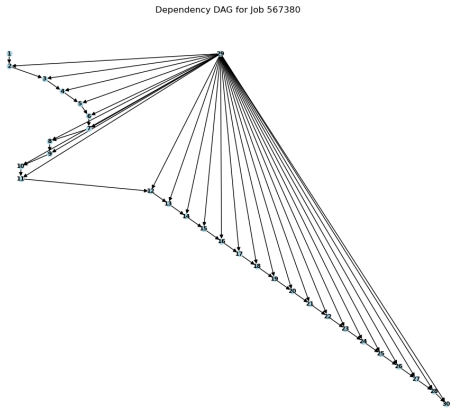

Figure 13: Sample DAG 567380 with the level of 28

This DAG is the most interesting one by far as it has fairly complex structures, and the majority of the nodes have a common ancestor node, which is also the root node as shown in Figure 13. At the same time, this DAG consumes lots of both CPU and memory as reflected by its instance information. Therefore, the Graphene method[1] scheduled this type of DAG with the longest NewLB. However, based on the bottleneck method shown in Sec 2.1.2, all of the nodes except for two root nodes are eligible, and it may be impossible to cut such a DAG with this special structure. Since this method only provides sketchy guidance and applications, the effectiveness of application on this type of DAG remains to be seen.

4 CONCLUSION

The dataset analysis highlights the varying applicability of Graphene[1] on different types of public datasets based on available information:

- The Google dataset is limited by its simplistic task structures and inconvenient access, necessitating more robust data for a more effective application of the Graphene method.
- Alibaba data proves ideal for evaluating advanced algorithms due to its detailed and hierarchical DAGs. However, this dataset suffers from a short collection period, spanning only 8 days[6, 11], compared to other datasets spanning one-month duration[2, 4, 8].
- Azure data gives us partial information to potentially form the DAG, and it offered resource-intensive scenarios in terms of cluster machines itself but requires additional information, such as resource usage and detailed task runtime, to help us apply with Graphene model[1].

On the Graphene side, our analysis revealed that it was only suitable for DAGs with complex structures, which also require a bottleneck point, which accounts for less than 1% in Alibaba Cluster Dataset. This character limits most of the DAGs from public datasets, and thus makes it less effective in real application. Eligible DAGs with other different structures, such as Figure 13, may also not be able to be applied. However, for some DAGs meeting these two requirements, it provides a more accurate runtime prediction by considering both length and resource usage.

Future work will focus on extending NewLB to handle less hierarchical DAGs, refining algorithms for simple task structures, and exploring more datasets from other providers like Meta [13].

This comprehensive dataset analysis provides the foundation for improving cluster scheduling efficiency in diverse environments.

REFERENCES

[1] R. Grandl, S. Kandula, S. Rao, A. Akella, and J. Kulkarni, "GRAPHENE: Packing and Dependency-Aware scheduling for Data-Parallel clusters," in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, (Savannah, GA), pp. 81–97, USENIX Association, Nov. 2016.

[2] A. Verma, L. Pedrosa, M. R. Korupolu, D. Oppenheimer, E. Tune, and J. Wilkes, "Large-scale cluster management at Google with Borg," in *Proceedings of the European Conference on Computer Systems (EuroSys)*, (Bordeaux, France), 2015.

[3] M. Sahebdel, A. Zeynali, N. Bashir, M. H. Hajiesmaili, and J. Oke, "Data-driven algorithms for reducing the carbon footprint of ride-sharing ecosystems," in *Companion Proceedings of the 14th ACM International Conference on Future Energy Systems*, e-Energy '23 Companion, (New York, NY, USA), Association for Computing Machinery, 2023.

[4] G. Inc., "Google cluster data." https://github.com/google/cluster-data, 2019. Accessed: [Insert Access Date].

[5] M. Corporation, "Azure public dataset." https://github.com/Azure/AzurePublicDataset/tree/master, 2023. Accessed: [Insert Access Date].

[6] A. Group, "Alibaba cluster data." https://github.com/alibaba/clusterdata, 2023. Accessed: [Insert Access Date].

[7] J. Wilkes, "Google cluster-usage traces v3," technical report, Google Inc., Mountain View, CA, USA, Apr. 2020. Posted at https://github.com/google/cluster-data/blob/master/ClusterData2019.md.

[8] M. Tirmazi, A. Barker, N. Deng, M. E. Haque, Z. G. Qin, S. Hand, M. Harchol-Balter, and J. Wilkes, "Borg: the next generation," in *EuroSys'20*, (Heraklion, Crete), 2020.

[9] D. Shahmirzadi, N. Khaledian, and A. M. Rahmani, "Analyzing the impact of various parameters on job scheduling in the google cluster dataset," *Cluster Computing*, vol. 27, pp. 7673–7687, Sep 2024.

[10] A.-I. Tuns and A. Spătaru, "Cloud service failure prediction on google's borg cluster traces using traditional machine learning," in *2023 25th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, pp. 162–169, 2023.

[11] M. Xu, Z. Fu, X. Ma, L. Zhang, Y. Li, F. Qian, S. Wang, K. Li, J. Yang, and X. Liu, "From cloud to edge: a first look at public edge platforms," in *Proceedings of the 21st ACM Internet Measurement Conference*, IMC '21, (New York, NY, USA), p. 37–53, Association for Computing Machinery, 2021.

[12] J. Guo, Z. Chang, S. Wang, H. Ding, Y. Feng, L. Mao, and Y. Bao, "Who limits the resource efficiency of my datacenter: an analysis of alibaba datacenter traces," in *Proceedings of the International Symposium on Quality of Service*, IWQoS '19, (New York, NY, USA), Association for Computing Machinery, 2019.

[13] D. Huye, Y. Shkuro, and R. R. Sambasivan, "Lifting the veil on Meta's microservice architecture: Analyses of topology and request workflows," in *2023 USENIX Annual Technical Conference (USENIX ATC 23)*, (Boston, MA), pp. 419–432, USENIX Association, July 2023.