Team 2 Opportunity Identification Project

Lingnan Gao, Andrew Wright

October 2017

Abstract

In this report, we identify the problem of Makespan Minimized Scheduling with Dependencies as a problem ripe for a Fixed Parameter Tractable (FPT) approach. While little work has been done in the area of FPT algorithms in operations research, recent work on the related problem of Makespan Minimized Scheduling without dependencies [1] demonstrates that there is promise to make progress on the topic.

1 Introduction

Scheduling on Directed Acyclic Graphs (DAG) is a crucial problem with many applications such as in the cases of task scheduling in parallel systems and workflow distributions. Typically, an optimal scheduling on a DAG would lead to a minimized completion time, reducing costs, etc.. Due to the significance of this problem, it has been extensively studied; challenges arise when it comes to find an optimal solution to this problem, as this problem is NP-hard. Finding an optimal solution using current methods will more likely than not be computationally prohibitive to find.

Although much work has been carried out to find an solution to this problem, most of this work relies on heuristic methods (e.g. HEFT [2]) or meta-heuristics (e.g. generic algorithms) to solve this problem. While these problems may work well under certain conditions, they do not provide an efficient, general solution.

One possible alternative solution to this problem is to develop a Fixed parameter Tractable (FPT) algorithm to solve this problem. However, to the best of our knowledge, limited work has been done in the area of FPT algorithms in scheduling research, and no work has been done on the problem of developing FPT algorithms for dependency scheduling problems.

2 Open Problem Description

The open problem we have identified is makespan scheduling on a DAG. Given a weighted Directed Acyclic Graph, $G = (V, E)$, the vertex $v \in V$ denotes a task to be executed, while the weight on v represents the execution time. A directed edge on G reflects dependencies among tasks, an edge from vertex u to v means task v cannot be started unless u has been completed. Our objective is, given M identical machines to process tasks on, to find out the execution orders for those tasks in order minimize the time to completion of all tasks.

Definition 1. *The* makespan *of a schedule of tasks S is the total time to completion from the start of the first task to the end of the last task.*

Makespan scheduling on Directed Acyclic Graph

Input: A directed graph $G = \{V, E\}$ with a function $W_v : V \to \mathbb{R}$, where W_v is vertex weights, and a set of identical machines, M

Question: Is there a Fixed Parameter Tractable algorithm parameterized by tree width which minimizes the makespan while respecting dependencies?

A possible parameter for development of an FPT algorithm might be the number of execution machines M. Another parameter which makes some intuitive sense may be the tree width of the graph.

3 Related Questions

To the best our knowledge, the FPT hardness result and solutions to the makespan scheduling for DAG still remains open. However, we observe that this problems is closely related to the problem of makespan scheduling without dependencies.

The makespan scheduling problem states that for n weighted tasks with no dependencies among them, is there a way to find a scheduling of n tasks so that total time to completion is minimized. The work of [1] shows that there is a FPT-algorithm to solve this problem when the problem is parameterized by p_{max}, the task with longest execution time.

Due to the similarities between these two problems, we can show that makespan scheduling can trivially be reduced to DAG makespan scheduling, meaning that DAG makespan is more difficult to solve. To show this, we create two terminal vertices *src* and *dst* with 0 execution time. For each task, we create a vertex u with weight equal to its execution time. For each vertex u, we add an edge between (src, u) and one edge between (u, dst). When this is done, we have a DAG. This implies that if we have a solver to solve any makespan minimized scheduling on DAG, we could also apply it to the makespan scheduling problem without dependencies.

4 Special Cases

There are four special cases of this problem that show promise to be solved:

- $M = 1$: A trivial case, where we can do a topological sort and use this ordering as the order for task execution.
- $M \geq |treewidth|$ of G: The treewidth of$G$ is the maximum number of tasks can be run at the same time. With the number of machines greater than the treewidth, we can schedule any task to run when it gets ready. The total running time would be the longest path in G.
- Each task except for *src* and *dst* is dependent on another task, while at the same time, while at the same time, depend by another task. In another word, those task would form a line. This is similar to $M = 1$ case, and we can solve this problem by scheduling each task one by one.
- There are no dependencies at all, which is same to the makespan scheduling, and the method in [1] would apply to this problem.

5 Conclusion

In this report, we identified Makespan Scheduling with Dependent Tasks as an open problem in the area Fixed Parameter Algorithms. The related work in makespan scheduling is reviewed, and special cases where we can obtain an answer for are analysed.

References

[1] Matthias Mnich and Andreas Wiese. Scheduling and fixed-parameter tractability. *Mathematical Programming*, 154(1-2):533–562, 2015.

[2] Haluk Topcuoglu, Salim Hariri, and Min-you Wu. Performance-effective and low-complexity task scheduling for heterogeneous computing. *IEEE transactions on parallel and distributed systems*, 13(3):260–274, 2002.