Machine Learning on Optimizing Algorithms on Tree Decomposition

Zhiyuan Shen, Yifei Wang, Nuo Zheng

Abstract—NP problems are one of the most famous problem in the theory computer science aspect. The exponential time complexity makes it unsolvable if the data is large. However, many methods are found to reduce the problem scale of the origin problem, including tree-decomposition. Different tree-decomposition cost different run times. However, we can fastly generate tree-decompositions [2], that means, if choose the best tree-decomposition, the run time of program can be reduce a lot. Our goal is to do the further optimization based on the tree-decompositions.

1 Introduction

This paper aims to use the machine learning to predict the run time of 3-colorability on a graph and its tree-decomposition. Thus we can generate multiple tree-decomposition, and find the fastest one then solve it.

This project chooses the 3-colorability as an example because it is a relatively easy NP-complete problem. This paper may first introducing some background information and give a fully view of how the project built.

This project conducts 3 parts, including data generating part, machine learning part and final test part. All 3 parts will be fully explained below analyses.

In conclusion, the paper will show the findings of optimization through machine learning on the tree-decompositions and try to explain the reasons behind such result.

2 Related Works

From the basic algorithm, the most traditional algorithm is to enumerate the colors of all nodes. After that, graph-coloring problem is determined to be the NP-complete problem. Rather than solve the problem straightforward, academic circle tries to reduce the scale of the problem. Tree-decomposition is an effective method[5] to reduce the scale of origin problem.

Then, the focus is shifted to the problems on the tree-decompositions. Improved tree-decomposition methods are found [2][3][4] to form a more standard and solvable structures.

Next, machine learning was introduced into this problem [1]. Machine learning is used to predict the run time of tree-decomposition structure and help problem-solver to dicide to solve which structure.

3 Background Information

This section will introduce 3 basci concept of the project, 3-colorability, Tree-decomposition, and machine-learning.

3.1 3-colorability

A coloring of a graph refers to a proper vertex coloring, namely a labeling of the graph's vertices with colors such that no two vertices sharing the same edge have the same color.

A coloring using at most k colors is called a proper k-coloring. Thus if $k \leq 3$, we call the graph is 3-colorable.

The same, if a graph $G = (V, E)$ is 3 colorable. There should exist one example that $\forall v \in V, color(v) \in \{R, B, G\}$, and $\forall e = (u, v) \in E, color(u)! = color(v)$.

3.2 Tree Decomposition

A *tree decomposition* of a graph $G = (V, E)$ is a tree T, where

1) Each vertex i of T is labeled by a subset $B_i \subset V$ of vertices G, referred to as a "bag".
2) Each edge of G is in a subgraph induced by at least one of the B_i
3) $\forall u \in V$, The subtree of T consisting of all "bags"containing u is connected.

A rooted *nice tree decomposition* (T, B) of a graph $G = (V, E)$ is *nice* if each node x of T is of one of the following types:

1) leaf node: x has no children and $|B_x| = 1$.
2) introduce node: x has a unique child y and $B_x = B_y \cup \{v\}$ for some $v \in V \backslash B_y$.
3) forget node: x has a unique child y and $B_x = B_y \backslash \{v\}$ for some $v \in B_y$.
4) join node: x has exactly 2 children y, z and $B_x = B_y = B_z$

We can convert a given tree decomposition of width t and $O(n)$ nodes into a nice tree decomposition of width t and $O(tn)$ nodes in time $t^{O(1)}n$[4].

3.3 Machine Learning and Predictions

4 Conclusion

In conclusion, the integration of multithreading algorithms marks a significant advancement in optimization beyond traditional algorithms. By harnessing the power of parallel processing, these algorithms offer fresh perspectives and avenues for optimization. Notably, the observed lower bound of time complexity, surpassing that of the original time complexity divided by the number of processors, underscores the tangible benefits in time efficiency that multithreading brings to program execution. However, this represents only the beginning of the potential unlocked by parallel algorithms. Further exploration and research are imperative to unearth even more efficient solutions and to fully realize the capabilities of parallel computing in algorithmic optimization. As technology continues to evolve, the synergy between multithreading algorithms and parallel computing architectures holds promise for pushing the boundaries of computational efficiency and problem-solving capabilities.

References

[1] M. Abseher, N. Musliu, and S. Woltran. Improving the efficiency of dynamic programming on tree decompositions via machine learning. *Journal of Artificial Intelligence Research*, 58:829–858, 04 2017.

[2] H. L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal on Computing*, 25(6):1305–1317, 1996.

[3] H. L. Bodlaender and T. Kloks. Better algorithms for the pathwidth and treewidth of graphs. In J. L. Albert, B. Monien, and M. R. Artalejo, editors, *Automata, Languages and Programming*, pages 544–555, Berlin, Heidelberg, 1991. Springer Berlin Heidelberg.

[4] K. Kangas, M. Koivisto, and S. Salonen. A Faster Tree-Decomposition Based Algorithm for Counting Linear Extensions. In C. Paul and M. Pilipczuk, editors, *13th International Symposium on Parameterized and Exact Computation (IPEC 2018)*, volume 115 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 5:1–5:13, Dagstuhl, Germany, 2019. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[5] J. Matoušek and R. Thomas. Algorithms finding tree-decompositions of graphs. *Journal of Algorithms*, 12(1):1–22, 1991.