# 2025 AdvDS HW1

B12902036 吳述宇

March 9, 2025

## Paper Readings

Title : A Linear-Time Algorithm for a Special Case of Disjoint Set Union

### Summary

Most of the time, the DSU is considered a data structure which runs in near-linear time with an additional factor of $\alpha(n)$. In this paper, Gabow and Tarjan demonstrated that under certain conditions, it is possible to modify some operations with auxiliary structures and properties of the RAM model to make DSU run in $O(m+n)$(linear) time. They introduced a two-level structure of microsets and macrosets along with a lookup table for find operation acceleration. Also, it had extended the concept of static tree union, where all elements are known beforehand, to incremental tree union in more dynamic scenarios. Although pratically $\alpha(n)$ often is regarded as a constant, it is still a siginificant theoretical improvement. The technique can be applied to multiple problems like maximum matching and dynamic connectivity.

### Problems

In this section, I will provide a problem, and it will be answered through a survey and summary of research on the modified DSUs for current techniques, especially on parallel computing and distributing computing.

#### Discussion on the Modified DSU in Modern Computing Environment

In practice, the modified DSU proposed by Gabow and Tarjan can be seen as improving the complexity by a constant factor, due to the slow growth rate of inverse Ackermann function, when comparing to the classical DSU approach. However, when considering contemporary parallel and distributed computing paradigms, which can drastically accelerate computational tasks in large-scale applications, the improvement remains theoretical and may not yield significant performance benefits in practical scenarios. Hence, I want to give a survey about a research question :

**Can the modified DSU be effectively integrated with modern parallel or distributing computing techniques to achieve better performance or are they inherently incompatible due to constraints? If they are not compatible, which one is better for certain situations?**

Optimality in the sequential setting doesn't imply its optimality in the parallel environments. In [STTW16], it demonstrates that a parallel union-find algorithm can accelerate the performance 8 to 11 times in the multicore environment. Also, it uses bulk unions to enhance efficiency of union, and it has the ability to handle diverse problem sets such as community detection in large graphs or GNN training in ML. For both improvements, constant factor($\alpha(n) \leq 5$ in the real-world scenarios) and problem diversity, the parallel union-find method makes better performance. By contrast, the linear union-find algorithm excels in under constraint situation, specifically static, small-scale, less cores or offline computational problems. Its effectiveness is inherently limited due to its reliance on known and static merge sequences. Nevertheless, in environments with limited parallelism or scenarios that operations and elements are known in advance, [GT85] is more advantageous due to its simplicity, memory efficiency, and optimal sequential performance. Additionally, when the merge operations form narrow and tall tree structures, the effectiveness of parallel approaches diminishes significantly, making the sequential Gabow-Tarjan algorithm more suitable in comparison to parallel methods, which perform better with wide and shallow union-structures.

Another parallel union-find algorithm proposed by [MP09] addresses load balancing issues in large-scale computing environments. The approach effectively distributes large-scale storage across to memory nodes in different computers. Initially, it performs DSU operations locally to minimize communication overhead, followed by a global merging phase that updates information to build the connective components for the whole graph. It is as well particularly well-suited for large-scale and dynamic scenarios. However, in small-scale settings or single machine applications, [GT85] DSU is significantly more efficient for both communication and implementation complexity. The two mentioned current methods above both utilize batch processing and parallel computation techniques, which overall reduces the cost for multiple find operations.

Recently, a new distributed union find(DUF) algorithm [XGS+21] has introduced an approach that enables overlapping computation and communication by leveraging k-d tree decomposition to achieve more uniform distribution. This method is designed to operate in high-performance computing environments, utilizing up to 1024 processors to optimize the computation. Comparing to [GT85] focusing on minimizing theoretical time, modern DSU algorithms deal with overall throughput and scalability. For most contemporary applications, current methods tend to accept higher computational costs in exchange for reduced latency and better adaptability to parallel architectures. Given advancements in hardwares, [GT85] may seem not practical in real-world scenarios, but does it really imply its obsolescence?

Conceptually, it gives valuable heuristics even for practical researchers, suggesting that if the problems exhibit predefined structures, a specialized DSU can be optimized to push theoretical efficiency limit. Moreover, while large-scale parallel computing dominates many modern applications, certain constraint environments such as FPGA or embedded systems still needs minimzing memory overhead, [GT85]'s high performance without using parallel DSU, which costs large space), is effective resources-limited computing scenarios. Ultimately, the choice of DSU algorithm depends on the problem characteristics and computational environment. The classical but classic [GT85] still has a place in the modern world.

# References


[GT85] Harold N. Gabow and Robert Endre Tarjan. A linear-time algorithm for a special case of disjoint set union. In *The Annual ACM Symposium on Theory of Computing*, pages 1–13, 1985.

[MP09] Fredrik Manne and Md. Mostofa Ali Patwary. A scalable parallel union-find algorithm for distributed memory computers. In *Parallel Processing and Applied Mathematics*, pages 1–10, 2009.

[STTW16] Natcha Simsiri, Kanat Tangwongsan, Srikanta Tirthapura, and Kun-Lung Wu. Work-efficient parallel union-find with applications to incremental graph connectivity. In *Proceedings of the 22nd International Conference on Euro-Par*, pages 561–573, 2016.

[XGS+21] Jiayi Xu, Hanqi Guo, Han-Wei Shen, Mukund Raj, Xueyun Wang, and Xueqiao Xu. Asynchronous and load-balanced union-find for distributed and parallel scientific data visualization and analysis. *IEEE Transactions on Visualization and Computer Graphics*, 27(6):2808–2820, 2021.


# Discussion Partners


Thanks to **b12201033** and **ChatGPT**.