Technical Perspective: Optimal Algorithms for Multiway Search on Partial Orders

Rajesh Jayaram
Google Research
rkjayaram@google.com

Given a list of comparable items $\mathcal{A} = \{a_1, \ldots, a_n\}$ sorted so that $a_1 < a_2 < \cdots < a_n$, a canonical problem is locating a target item q within A if it exists. The canonical algorithm for this problem, of course, is binary search, which locates q using at most $O(\log n)$ comparisons between q and elements of A. Binary search is an indispensable tool for totally ordered datasets. However, many naturally occurring datasets are only partially ordered (posets), meaning that not all pairs of elements are comparable. Every such poset can be expressed as a *directed acyclic graph* (DAG), with edges (x, y) representing the relation $x < y$.

Consider, for instance, a family $\mathcal{A}$ of image labels, consisting of objects (such as cat, kitten, baby), verbs (such as jumping), and activities (such as sport). Taking various combinations of these descriptors induces the DAG shown in Figure 1.

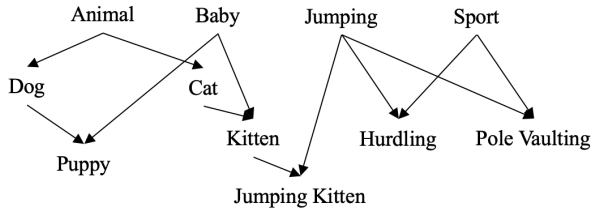

Figure 1: A partially ordered set of image labels.

Such DAGs arise in the problem of image classification with crowd-sourcing, where the objective is to assign the most specific label possible to an image with the help of a human user, who can be asked questions of the form "is this image an **x**", where **x** is a label in the family $\mathcal{A}$. This motivates the *partially ordered multiway search* problem (POMS): given a DAG G and an oracle who can answer *reachability queries*, namely, is vertex x reachable from vertex y in G, find the target q in as few queries as possible. This generalizes the case of binary search when G is a path. Since adaptive interactions with an oracle (such as a human) can be expensive, one is allowed to batch together k queries at a time to ask the oracle.

The case where G is a tree and $k = 1$ has been studied for quite some time, and tight bounds of $\Theta(d \log_{1+d}(n))$-queries are known [1, 2], where d is the maximum degree of G. However, for the more general case of searching in DAGs and $k > 1$, the best known upper and lower bounds were $O(\log n \log_{1+k} n + \frac{d}{k} \log_{1+d} n)$ and $\Omega(\frac{d}{k} \log_{1+d} n)$ respectively [4]. In *Optimal Algorithms for Multiway Search on Partial Orders*, which appeared in PODS '22 [3], the authors Lu, Martens, Niewerth, and Tao, fully resolve the complexity of POMS, by giving tight upper and lower bounds of $\Theta(\log_{1+k} n + \frac{d}{k} \log_{1+d} n)$. This matches the complexity of binary search when $d = k = 1$, giving a strict generalization to the case of DAGs. Notably, these improvements in query complexity save expensive human-computer interactions in real-world applications such as the prior image-recognition task.

The main challenge of POMS is to efficiently identify a small subgraph which contains the query q to recurse on. In binary search, this is easily handled by comparing q to the middle vertex, and recursing on the left or right paths. The case of trees can be posed as a nice competitive programming problem: for constants d and k, the solution is to find a separator vertex in T whose removal splits T into at most d subtrees of size at most $n/2$, and determine with d queries which of them contains the target q. However, the case of DAGs is not nearly as straightforward. Namely, decomposition is not as simple as in trees since there can be many paths to the query from a given vertex. The authors develop a new recursion scheme which maintains an important path-preserving invariant, and requires a deeper understanding of reachability in DAGs.

The authors also consider POMS in the external memory (EM) model, where vertices are written to disk in blocks, which the algorithm must read into memory in order to query the oracle. This model is important for database applications, since many standard data structures are not I/O efficient. The authors design a black-box reduction from a query-efficient to an I/O efficient algorithm for a large class of data structures, which marks an important step towards making theoretically optimal data-structures practical.

REFERENCES

[1] D. Dereniowski and M. Kubale. Efficient parallel query processing by graph ranking. *Fundamenta Informaticae*, 69(3):273–285, 2006.
[2] E. Emamjomeh-Zadeh, D. Kempe, and V. Singhal. Deterministic and probabilistic binary search in graphs. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 519–532, 2016.
[3] S. Lu, W. Martens, M. Niewerth, and Y. Tao. Optimal algorithms for multiway search on partial orders. In *Proceedings of the 41st ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 175–187, 2022.
[4] Y. Tao, Y. Li, and G. Li. Interactive graph search. In *Proceedings of the 2019 International Conference on Management of Data*, pages 1393–1410, 2019.

SIGMOD Technical Perspective, 2023

https://doi.org/XXXXXXX.XXXXXXX