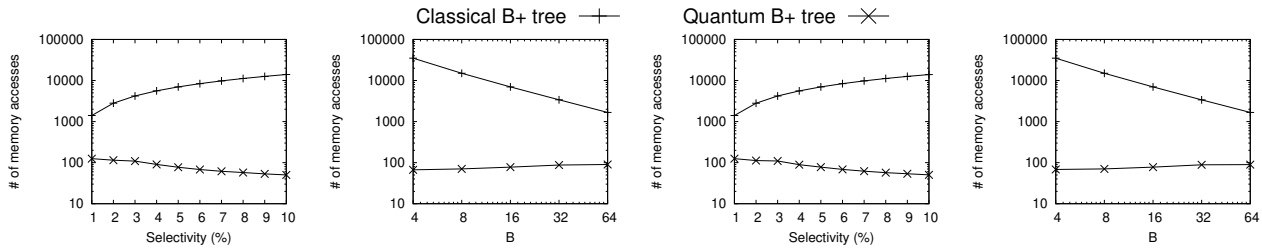
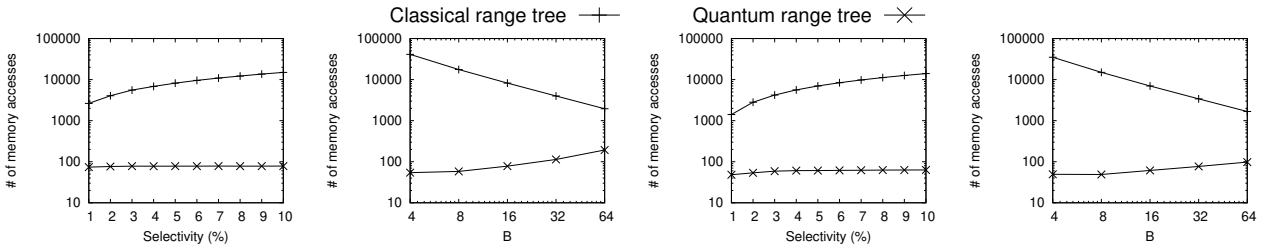


(a) Brightkite (b) Brightkite (c) Gowalla (d) Gowalla

Figure 1: 1-dimensional range queries



(a) Brightkite (b) Brightkite (c) Gowalla (d) Gowalla

Figure 2: 2-dimensional range queries

ADDITIONAL EXPERIMENT

In this report, we show experimental results to further discuss the advantages of the quantum B+ tree. The study of the real-world quantum supremacy [1, 2, 7], which is to confirm that a quantum computer can do tasks faster than classical computers, is still a popular topic in the quantum area. We are not going to verify quantum supremacy in this problem, but we believe it will be verified in a future quantum computer. In this paper, we choose to evaluate the *number of memory accesses* to make the comparisons, which corresponds to IOs in traditional searches. In the quantum data structures, a QRAM read operation is counted as 1 IO. In the classical data structures, a page access is counted as 1 IO. IOs cannot be regarded as the real execution time, since we have no any information about the real implementation of a QRAM, but they can reflect the potential of the data structures. In addition, the existing quantum simulators such as Qiskit [8] and Cirq [5] do not have a solution to simulate a efficient QRAM, so we choose to use C++ to perform the quantum simulations.

We conducted an experiment by simulating the quantum B+ tree on real-world datasets from SNAP [6] with C++. The two datasets named Brightkite (4m) and Gowalla (6m) are two lists of check-ins with timestamps and locations. To compare the quantum B+ tree and the classical B+ tree [4], we took the timestamps as the 1-dimensional data, which corresponds to the time-based range queries in real-world applications. Figure 1 shows the result. First, we studied the performance with different selectivity (i.e., $\frac{k}{N}$). We varied the selectivity between 1% and 10% and the results are shown in Figure 1(a) and Figure 1(c), where the x-axis denotes the selectivity and the y-axis denotes the number of memory accessed needed to answer a range query on average. We can see that the IOs needed by the classical B+ tree grows linearly with k. However, the IOs needed by the quantum B+ tree even decreases. It seems surprising but in fact it is reasonable. The reason is that a larger k shortened the process of the classical global search, such that we can turn to the local quantum search faster. Also, as mentioned in Section 5.1.2, post-selection costs $O(N/k)$ time, so a larger k also speedup the post-selection step. In addition, we studied how the block size affect the performance, and the results are shown in Figure 1(b) and Figure 1(d), where the x-axis denotes the block size and the y-axis denotes the number of memory accessed needed to answer a range query on average. IOs needed by the classical B+ tree decreases fast, since with a larger B, fewer pages were accessed by the B+ tree. IOs needed by the quantum B+ tree increases, and this observation corresponds to a middle result in the proof of Lemma 5.4 that B has an impact on the success rate of the post-selection step.

To show the advantage of the quantum range tree, we compared it with the classical range tree [3]. Note that Chazalle [3] has proved the lower bound of the orthogonal range searching problem, and the range tree is asymptotically optimal. We choose the state-of-the-art to make the comparison. We took the locations in the datasets as the 2-dimensional data, which corresponds to the location-based range queries in real-world applications. Figure 2 shows the result. We also varied the selectivity between 1% and 10%, and varied B from 4 to 64 to study their performance. The observations are almost the same as the 1-dimensional cases. The only difference is that a larger k did not lead to fewer IOs needed by the quantum range tree. The reason is that a larger k leads to more candidate nodes for local quantum search, which is different from the 1-dimensional case.

In conclusion, the quantum trees performs far better than the classical trees from the perspective of the number of memory accesses. In 1-dimensional case, a larger k and a smaller B leads to a better performance. In 2-dimensional case, a smaller B leads to a better performance.

REFERENCES

[1] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando GSL Brandao, David A Buell, et al. 2019. Quantum supremacy using a programmable superconducting processor. *Nature* 574, 7779 (2019), 505–510.

[2] Sergio Boixo, Sergei V Isakov, Vadim N Smelyanskiy, Ryan Babbush, Nan Ding, Zhang Jiang, Michael J Bremner, John M Martinis, and Hartmut Neven. 2018. Characterizing quantum supremacy in near-term devices. *Nature Physics* 14, 6 (2018), 595–600.

[3] Bernard Chazelle. 1990. Lower bounds for orthogonal range searching: I. the reporting case. *Journal of the ACM (JACM)* 37, 2 (1990), 200–212.

[4] Douglas Comer. 1979. Ubiquitous B-tree. *ACM Computing Surveys (CSUR)* 11, 2 (1979), 121–137.

[5] Bacon D Gidney C and contributors. 2018. Cirq: A python framework for creating, editing, and invoking noisy intermediate scale quantum (NISQ) circuits. https://github.com/quantumlib/Cirq.

[6] Jure Leskovec and Rok Sosič. 2016. Snap: A general-purpose network analysis and graph-mining library. *ACM Transactions on Intelligent Systems and Technology (TIST)* 8, 1 (2016), 1–20.

[7] Barbara M Terhal. 2018. Quantum supremacy, here we come. *Nature Physics* 14, 6 (2018), 530–531.

[8] Robert Wille, Rod Van Meter, and Yehuda Naveh. 2019. IBM's Qiskit tool chain: Working with and developing for real quantum computers. In *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 1234–1240.