Assignment 2: State Vector Simulations

Big Data and Intelligent System Lab @ ECNU

Due date: July 29, 2024, 10:00 AM

1 Introduction

In the previous assignment, we have implemented operation matrix simulations. You may have noticed that constructing operation matrices is very costly. In this assignment, we will explore a more efficient simulation method called the **state vector simulation**. It can save both computation and memory overheads compared with the operation matrix simulation. However, the overheads still grow exponentially with the number of qubits increasing. The state vector simulation is also a full-state quantum circuit simulation method.

2 State Vector Simulations

Instead of constructing the entire operation matrix, the *state vector simulation* (SVSim) directly applies the transformation specified by a quantum gate to the state vector to reduce the overhead.

2.1 Single-qubit Gates

For a single-qubit gate G_a, applying it to the kth qubit is equivalent to applying its gate matrix to every pair of amplitudes whose binary indices differ **only** in the kth bits ($k = 0, \ldots, n-1$):

$$\begin{bmatrix} \alpha'_{*\ldots*0_k*\ldots*} \\ \alpha'_{*\ldots*1_k*\ldots*} \end{bmatrix} = G_a \cdot \begin{bmatrix} \alpha_{*\ldots*0_k*\ldots*} \\ \alpha_{*\ldots*1_k*\ldots*} \end{bmatrix}.$$

The stride between $\alpha_{*\ldots*0_k*\ldots*}$ and $\alpha_{*\ldots*1_k*\ldots*}$ is 2^k, i.e., $(*\ldots*1_k*\ldots*)_2 - (*\ldots*0_k*\ldots*)_2 = 2^k$.

For example, Fig. 1(b) shows how to do SVSim for G_a on q_0. Four pairs of amplitudes are 00$\underline{0}$-00$\underline{1}$, 01$\underline{0}$-01$\underline{1}$, 10$\underline{0}$-10$\underline{1}$, and 11$\underline{0}$-11$\underline{1}$. Their binary indices differ only in the 0th position. The stride within each pair is $2^0 = 1$.

More specifically, suppose $G_a = X$, which flips the state of q_0, i.e., $|0\rangle$ to $|1\rangle$ and $|1\rangle$ to $|0\rangle$. Since the circuit in Fig. 1(a) has three input qubits, the X gate on q_0 should turns $|000\rangle$ into $|001\rangle$, $|001\rangle$ into $|000\rangle$, $|010\rangle$ into $|011\rangle$, $|011\rangle$ into $|010\rangle$, $|100\rangle$ into $|101\rangle$, $|101\rangle$ into $|100\rangle$, $|110\rangle$ into $|111\rangle$, and $|111\rangle$ into $|110\rangle$. This is equivalent to exchanging the amplitudes corresponding to these quantum states in the state vector:

$$X\begin{bmatrix}\alpha_{000}\\ \alpha_{001}\end{bmatrix} = \begin{bmatrix}\alpha_{001}\\ \alpha_{000}\end{bmatrix}, \quad X\begin{bmatrix}\alpha_{010}\\ \alpha_{011}\end{bmatrix} = \begin{bmatrix}\alpha_{011}\\ \alpha_{010}\end{bmatrix}, \quad X\begin{bmatrix}\alpha_{100}\\ \alpha_{101}\end{bmatrix} = \begin{bmatrix}\alpha_{101}\\ \alpha_{100}\end{bmatrix}, \quad X\begin{bmatrix}\alpha_{110}\\ \alpha_{111}\end{bmatrix} = \begin{bmatrix}\alpha_{111}\\ \alpha_{110}\end{bmatrix}.$$

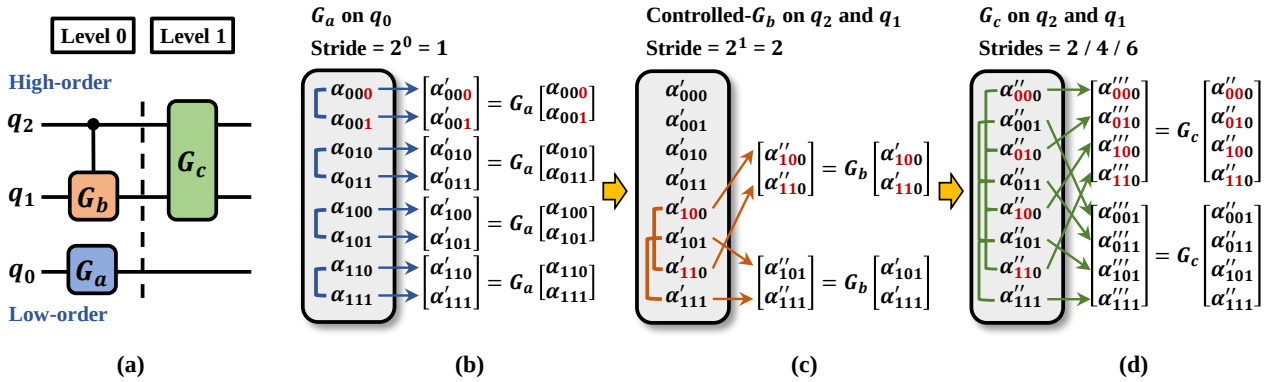

Figure 1: (a) An example of a 3-qubit 2-level quantum circuit. The state vector simulation (SVSim) for (b) G_a, (c) controlled-G_b, and (d) G_c.

The state vector becomes $[\alpha_{001}, \alpha_{000}, \alpha_{011}, \alpha_{010}, \alpha_{101}, \alpha_{100}, \alpha_{111}, \alpha_{110}]^T$, as **amplitudes are updated in place**.

2.2 Two-qubit Controlled Gates

For a 2-qubit controlled gate controlled-G_b on a control qubit q_c and a target qubit q_t, assuming $c > t$, G_b of size 2×2 is applied to q_t if $q_c = |1\rangle$:

$$\begin{bmatrix} \alpha'_{*\ldots*1_c*\ldots*0_t*\ldots*} \\ \alpha'_{*\ldots*1_c*\ldots*1_t*\ldots*} \end{bmatrix} = G_B \cdot \begin{bmatrix} \alpha_{*\ldots*1_c*\ldots*0_t*\ldots*} \\ \alpha_{*\ldots*1_c*\ldots*1_t*\ldots*} \end{bmatrix}.$$

In other words, we only need to find the amplitude pairs whose binary indices meet the following conditions:

1. The binary bits corresponding to the control qubit are 1's.
2. The binary bits corresponding to the target qubit differ.
3. The remaining binary bits are the same.

For example, as shown in Fig. 1(c), the amplitude pairs are 100-110, and 101-111 for controlled-G_b with q_2 controlling q_1. Suppose $G_b = X$ and controlled-G_b is CX. The state vector will be updated to $[\alpha_{001}, \alpha_{000}, \alpha_{011}, \alpha_{010}, \alpha_{111}, \alpha_{110}, \alpha_{101}, \alpha_{100}]^T$.

It should be noted that SVSim for a controlled gate only involves the gate matrix **applied to the target qubit(s)**. For example, the controlled-G_b has two input qubits, so its gate matrix should be at least 4×4 according to the previous assignment. However, SVSim only uses the gate matrix of G_b instead of controlled-G_b. There is no need to consider the case of nonadjacent input qubits as the stride between amplitudes is considered.

2.3 Two-qubit Non-controlled Gates

For a 2-qubit non-controlled gate G_c of size 4×4 on q_c and q_t ($c > t$), SVSim can be described as:

$$\begin{bmatrix} \alpha'_{*\ldots*0_c*\ldots*0_t*\ldots*} \\ \alpha'_{*\ldots*0_c*\ldots*1_t*\ldots*} \\ \alpha'_{*\ldots*1_c*\ldots*0_t*\ldots*} \\ \alpha'_{*\ldots*1_c*\ldots*1_t*\ldots*} \end{bmatrix} = G_C \cdot \begin{bmatrix} \alpha_{*\ldots*0_c*\ldots*0_t*\ldots*} \\ \alpha_{*\ldots*0_c*\ldots*1_t*\ldots*} \\ \alpha_{*\ldots*1_c*\ldots*0_t*\ldots*} \\ \alpha_{*\ldots*1_c*\ldots*1_t*\ldots*} \end{bmatrix},$$

where the strides between $\alpha_{*\ldots*0_c*\ldots*0_t*\ldots*}$ and $\alpha_{*\ldots*0_c*\ldots*1_t*\ldots*}$ are 2^t, $\alpha_{*\ldots*0_c*\ldots*0_t*\ldots*}$ and $\alpha_{*\ldots*1_c*\ldots*0_t*\ldots*}$ are 2^c, $\alpha_{*\ldots*0_c*\ldots*0_t*\ldots*}$ and $\alpha_{*\ldots*1_c*\ldots*1_t*\ldots*}$ are $2^c + 2^t$.

For example, as shown in Fig. 1(d), the strides between α''_{000}-α''_{010}, α''_{000}-α''_{100}, and α''_{000}-α''_{110} are 2, 4, and 6, respectively. Suppose $G_c = SWAP$. The state vector will be updated to $[\alpha_{001}, \alpha_{000}, \alpha_{111}, \alpha_{110}, \alpha_{011}, \alpha_{010}, \alpha_{101}, \alpha_{100}]^T$.

All quantum gates can be similarly simulated by SVSim, i.e., calculating the strides to find the amplitudes involved and applying the gate matrix to update these amplitudes. Since each gate on the circuit updates the entire state vector once, for an n-qubit quantum circuit with T levels, the time complexity of SVSim is $O(n \cdot T \cdot 2^n)$. The space complexity is $O(2^n)$ for storing the state vector and $O(n \cdot T)$ for storing all the small gate matrices.

3 Getting Started

Please clone the branch `a2-svsim` of the repository QSimLab from GitHub. It contains the necessary files to start this assignment, including the following contents.

qsim/svsim.cpp This is the only file that needs to be modified and is where you write your code. It is unnecessary to make changes to other files.

qsim/ The other files in this folder provide basic data structures and functions necessary for quantum circuit simulation. Descriptions can be seen in `qsim/README.md`.

main/ This contains the main function of this project.

py/ This folder includes a Python file that uses a Qiskit simulator. It can be used to verify whether the simulator that you implement works correctly.

Makefile This is the Makefile indicating how to compile this project on Windows.

4 Implementation

This section outlines the functions required to implement in your solution (in `svsim.cpp` only). We will explain them in a top-down approach.

4.1 `void svsimForGate(Matrix<DTYPE>& sv, QGate& gate)`

This function is called by `void SVSim(Matrix<DTYPE>& sv, QCircuit& qc)` to conduct the SVSim for a quantum gate. The initial state vector is recorded in `sv`, which should be updated during the simulation.

4.2 `bool isLegalControlPattern(ll ampidx, QGate& gate)`

This function is called by `svsimForGate` to check if the binary index of an amplitude `ampidx` is a legal control pattern of `gate`. Suppose that `gate` is controlled by $|1\rangle$'s. If there exists a control qubit q_i of `gate` such that the ith binary bit of `ampidx` is 0, `ampidx` is an illegal control pattern of `gate`. Otherwise, `ampidx` is a legal control pattern of `gate`.

5 Assignment Submission

Your assignment should be submitted via https://yunbiz.wps.cn/c/collect/cBtAOXFPmT9 before the due date. For your convenience, a feasible solution will be pushed to the GitHub branch `a2-svsim-ans` on July 26 at 10:00 AM. Feel free to refer to it if needed. The submission format is as follows.

1. If you only modify `svsim.cpp`, please rename it as "`a2-`*`your_name`*`.cpp`" and submit it.
2. If other files have been modified for some reason, please compress the entire project into a zip file, name it as "`a2-`*`your_name`*`.zip`", and upload it. It is strongly recommended to add a README briefly explaining the modifications.