Sending Images Via Superdense Coding

Stephen Diadamo

Objective

The objective of the submission is to simulate a protocol that allows two parties to share PNG images over a quantum network making use of a generalized superdense coding scheme.

Summary

In this submission, I was able to design a protocol that sends PNG images over the quantum network.
The protocol works as follows:

1. **INITIALIZATION**: Assume there are enough EPR pairs to complete the protocol.
2. **SENDER**: Encodes a PNG image into an array of data points. The first two data points are the dimensions of the image and the following data points represent the colour of the pixels and the alpha values (i.e. the transparency). The information is encoded in a binary representation.
3. **SENDER**: An array which represents how many qubits are needed to decode each data point is generated and transmitted to the receiver classically (e.g. four qubits are needed to transmit eight bits of information, so the number four is sent to the receiver).
4. **SENDER**: Encodes with respect to each data point their share of the EPR pairs in chunks of two bits (e.g. '0111' is encoded in two steps, first '01' with one qubit and then '11' on a second) and transmits the qubit via the quantum network, essentially using multiple applications of superdense coding.
5. **RECEIVER**: Receives an array of how many data points should be received and how many qubits make up each data point, and receives all qubits (e.g. if the data to receive is ['11', '1101'], then the receiver receives the array [1, 2], that is, a length two array, the number of data points to receive, excepting 1 qubit for the first data point and two qubits for the second).
6. **RECEIVER**: Measures all qubits and using the classical data reconstructs the binary information representing the image. The binary information can be decoded and reformatted such that the PNG image can be reconstructed.

In my implementation, I was able to simulate sending PNG images with transparency (i.e. the alpha value for each pixel is included). A problem arose because of the number of qubits needed to send larger images. I found that in order for the simulation to run on my computer, I had to restrict the scenario to PNG images of size two pixels by two pixels and no transparency, using only the colours red, green, black, and white. Even in this restricted case, since I have used a superdense coding scheme, I am able to encode four different colours instead of just two, which would be the maximum classically. With access to more qubits, the scheme works in general.

Design Overview

I expand on each point above.

1: In the simulation, I generate the necessary EPR pairs as needed, but more realistically, these pairs would already exist for each party.

2: The sender provides a path to a PNG image. Using the Python library PyPNG, it is easy to decode the image into its components, extracting the dimensions of the image and pixel data which I call data points. The data points are then converted into an array of binary strings and will be the data transmitted to the receiver. Here, a decision had to be made regarding how to handle odd-lengthed binary numbers in order to make use of the superdense coding idea. My decision was to simply pad each odd-lengthed binary string with a leading 0, but alternative, more efficient methods could be used to transmit the bit of data, possibly even classically.

For my restricted implementation, I use just four colours, and encode them as '00' for black, '01' for green, '10' for red, and '11' for white. For example, the image in Figure 1 will be converted into the array: ['10', '10', '11', '01','00', '10'], two entries for the dimensions and four entries for the pixel colours. To further reduce the number of qubits needed, a second restriction was made which excludes the first two entries of the array and I assume that the images sent are always two by two.

Figure 1: 2x2 PNG image

3: Because the protocol makes multiple uses of the quantum network, the receiver is sent meta-data about how much quantum data they will receive. This meta-data describes how many qubits will be transmitted for each data point. Continuing with the above example, the meta-data would be ['1', '1', '1', '1', '1', '1'], that is, one qubit will be sent to reconstruct each data point. The meta-data is sent classically.

4: The sender will encode their share of the EPR pairs to represent each data point of the image and then send their qubits over the quantum network to complete their portion of the superdense coding protocol.

5: The receiver will receive all classical and quantum data. For each element in the meta-data array, the receiver will wait for that many number of qubits to arrive before reconstructing the data points. During this process, they complete their steps of superdense coding protocol which is measuring both qubits in the EPR pair and reconstructing the binary data. Once the number of qubits specified are received and measured for a certain data point, the process repeats until all binary data of the data points is reconstructed.

6: With the binary data decoded, the receiver can reconstruct the image data array and then use the PyPNG library to reconstruct the PNG file. With the restricted case, a simple compression/decompression scheme is implemented so the receiver can decompress and format the data in order to reconstruct the image.

Overall the protocol is quite simple. Many additional features could be added regarding security and compression, but for this simple example, I exclude these parts and focus on the main task of sending an image. Since I was only able to make this work in the restricted case, I have included in the code a flag to remove such restrictions in the case of access to more qubits.