Towards graph classification with Gaussian Boson Sampling by embedding graphs on the X8 photonic chip

Edgard Pierre
Prevision.io
Paris, France
edgard.pierre@prevision.io

Michel Nowak
Prevision.io
Paris, France
michel.nowak@prevision.io

September 28, 2021

Abstract

Photonics chips on which one can perform Gaussian Boson Sampling have become accessible on the cloud, in particular the X8 chip of Xanadu. In this technical report, we study its potential use as a first step towards graph classification on quantum devices. In order to achieve this goal, we study the generated samples of the graph embedding method which leads to feature vectors. This is done on a restricted class of unweighted, undirected and loop-free graphs. Hardware constraints are matched to properties of graphs that can be encoded. We report experiments on the X8 chip as well as comparisons to numerical simulations on a classical computer and analytical solutions. We conclude this technical report by trying to take photon loss into account and explain the observed results accordingly.

1 Introduction

Gaussian Boson Sampling (GBS) is currently one of the most promising near term quantum computer. It recently achieved quantum supremacy [1]. This year, the company Xanadu released an 8 modes compact GBS photonic chip [2]. One of their work consisted in studying the possibility of embedding graphs into the chip [3] in order to perform various operations such as graph isomorphism [4] and graph similarity [5]. At the same time, improvements on graph neural networks [6] show that graph isomorphism plays a central role in the classification itself. The emergence of their quantum counterparts [7] with already proven applications [8] motivated us to investigate how photonics chips could help in this field. A preliminary part of our study was to determine to which extent graphs can be embedded on a photonic device and how one can derive isomorphism properties from it.

We will thus begin this technical report by describing the hypothesis that we make on the graphs we wish to embed on the chip and match them to the hardware constraints in section 2. We will then group these possible graphs into categories that are of interest for us, namely, isomorphic groups in section 3. At this point, we will be able to run numerical simulations and real calculations on the hardware (section 4). Finally, we will extract the feature vectors and compare them to expected analytical calculations in section 5.

In order to prepare for the core of the work, we quickly present the X8. Figure 1 shows the circuit that one can implement on this hardware. It is composed of eight modes, each starting in the vacuum state. Then, S2 gates are applied between the first 4 modes (called the signal modes) and the last 4 modes (the idler modes). Subsequently, a unitary transformation is applied on each of the two groups of modes. Finally, the photons are detected on the output channel. The measurement step can be done by counting the number of photons in each mode. We will further refer to this method as the event counting method. Detectors can also be triggered as soon as one photon reaches the output channel. This method will be referred to as the threshold method.
2 Possible graph embedding on the X8 chip.

We wish to encode graphs on the X8 hardware in order to subsequently derive informative features and solve Machine Learning tasks. The method used for the embedding can only implement graphs with a number of nodes equal to the number of modes of the device and needs to be fed with the associated adjacency matrix $A$. We are thus restricted to working with graphs containing eight nodes. We now present the initial hypothesis that we assume from the graphs we are interested in and map hardware constraints to the possible graphs that can be encoded. We are interested in graphs that:

- are unweighted (the matrix elements are 0 or 1)
- are undirected (the matrix is symmetric)
- do not have self loops (the diagonal of the matrix is 0).

As a result one can write the adjacency matrix with 28 independent parameters (which corresponds to $2^{28} \approx 2.68 \times 10^8$ possible graphs).

$$A = \begin{pmatrix}
0 & \delta_0 & \delta_1 & \delta_2 & \delta_3 & \delta_4 & \delta_5 & \delta_6 \\
\delta_0 & 0 & \delta_7 & \delta_8 & \delta_9 & \delta_{10} & \delta_{11} & \delta_{12} \\
\delta_1 & \delta_7 & 0 & \delta_{13} & \delta_{14} & \delta_{15} & \delta_{16} & \delta_{17} \\
\delta_2 & \delta_8 & \delta_{13} & 0 & \delta_{18} & \delta_{19} & \delta_{20} & \delta_{21} \\
\delta_3 & \delta_9 & \delta_{14} & \delta_{18} & 0 & \delta_{22} & \delta_{23} & \delta_{24} \\
\delta_4 & \delta_{10} & \delta_{15} & \delta_{19} & \delta_{22} & 0 & \delta_{25} & \delta_{26} \\
\delta_5 & \delta_{11} & \delta_{16} & \delta_{20} & \delta_{23} & \delta_{25} & 0 & \delta_{27} \\
\delta_6 & \delta_{12} & \delta_{17} & \delta_{21} & \delta_{24} & \delta_{26} & \delta_{27} & 0
\end{pmatrix}$$

with $\delta_i = 0$ or 1.

Bipartite graphs

The X8 chip can only embed bipartite graphs because of its structure. These graphs’ adjacency matrices are defined as the following:

$$A = \begin{pmatrix}
0 & M \\
M^T & 0
\end{pmatrix}$$

with $M$ a given sub-matrix and $M^T$ its transpose. With this configuration, one can see the number of independent parameters is reduced to 16 ($65536$ possible graphs).
Signal and idler mode constraint

The hardware constraints regarding the connections between the first four modes (the signal modes) of the X8 chip and the four last (the idler modes) also apply a new condition on the sub-matrix M: it must be symmetric. One can then reduce the number of independent parameters to 10 (1024 possible graphs).

$$A = \begin{pmatrix}
0 & 0 & 0 & 0 & \delta_0 & \delta_1 & \delta_2 & \delta_3 \\
0 & 0 & 0 & 0 & \delta_4 & \delta_5 & \delta_6 & \delta_7 \\
0 & 0 & 0 & 0 & \delta_8 & \delta_9 & \delta_{10} & \delta_{11} \\
0 & 0 & 0 & 0 & \delta_{12} & \delta_{13} & \delta_{14} & \delta_{15} \\
\delta_0 & \delta_4 & \delta_8 & 0 & 0 & 0 & 0 & 0 \\
\delta_1 & \delta_5 & \delta_9 & \delta_{13} & 0 & 0 & 0 & 0 \\
\delta_2 & \delta_6 & \delta_{10} & \delta_{14} & 0 & 0 & 0 & 0 \\
\delta_3 & \delta_7 & \delta_{11} & \delta_{15} & 0 & 0 & 0 & 0 \\
\end{pmatrix}$$

Until the end of this document, we will name the graphs as '$\delta_0 \delta_1 \delta_2 \delta_3 \delta_4 \delta_5 \delta_6 \delta_7 \delta_8 \delta_9$' (see Tab.1).

Normalization constraint

A last constraint on the adjacency matrix elements comes from the embedding method. Indeed, we use the Strawberry Fields (sf) library in order communicate with the X8 chip and embed the graphs. The BipartiteGraphEmbed class from the sf.ops module requires the use of a mean photon per mode normalization parameter in order to be sure that the singular values are scaled such that all Sgates have squeezing parameter $r=1$. Whatever the $m$ parameter is, the only configurations embeddable are the ones where the $r$ parameters for all the four signal modes are identical and or equals to zero. This gives the following constraint on $M$:

$$M = U \text{diag}(\text{tanh } r_i) U^T$$

where $r_i$ is the squeezing parameter on the $i$-th pair of modes and $U$ the interferometer used to prepare the state.

This leaves 75 graphs over the 1024 theoretically embeddable graphs fulfilling these conditions. We noticed that the value of $m$ depends on the number of substructures of the graph (called "subgraph" in the next section). For one substructure, we have $m = m_0 = 0.345274461385554870545$. For $n$ substructures (up to $n=4$), we have $m = n \times m_0$.

3 Graphs characterization and isomorphism.

Now that we have put constraints on the possible graphs that are of interest to us, we characterize them into groups. The terminology used in the coming listing is used assuming the reader has some basics in the graph theory.

Despite being able to embed 75 different graphs into the X8 chip, we noticed most of them were isomorphic. It is possible to group them into 10 categories:

- the graph contains a $K_2$ subgraph and 6 free vertices. Called "1K2". $m = m_0$
- the graph contains 2 $K_2$ subgraphs and 4 free vertices. Called "2K2". $m = 2 \times m_0$
- the graph contains a $C_4$ subgraph and 4 free vertices. Called "1C4". $m = m_0$
• the graph contains 2 \( P_3 \) subgraphs and 2 free vertices. Called "2P3". \( m = 2 \times m_0 \)
• the graph contains 3 \( K_2 \) subgraphs and 2 free vertices. Called "3K2". \( m = 3 \times m_0 \)
• the graph contains a Thomsen \( K_{3,3} \) subgraph and 2 free vertices. Called "1K33". \( m = m_0 \)
• the graph contains 2 \( S_3 \) subgraphs and no free vertex. Called "2S3". \( m = 2 \times m_0 \)
• the graph contains 4 \( K_2 \) subgraphs and no free vertex. Called "4K2". \( m = 4 \times m_0 \)
• the graph contains 2 \( C_4 \) subgraphs and no free vertex. Called "2C4". \( m = 2 \times m_0 \)
• the graph is a \( K_{4,4} \) graph. Called "1K44". \( m = m_0 \)

One can note that the only embeddable connected graph is 1111111111. We will use this graph as an example for the last part of our study (related to errors, see section 5).

We illustrate one simple and one complex graph for the reader. The first simple graph is an 8 node graph with only 4 nodes connected to each other in a square fashion as can be viewed in its planar representation in Figure 2a or in its bipartite representation in Figure 2b.

![Figure 2: 1C4 illustrations](image)

We also illustrate a more complex graph that will be used in later analysis: the 1K44 graph in its planar representation in Figure 3a and in its bipartite representation Figure 3b.

![Figure 3: 1K44 illustrations](image)

Other graphs representations can be found in Appendix A.

Table summarizes the different possible embeddable unweighted graphs on the X8 chip.
4 Generation of GBS samples and simulated samples

We shot 100,000 GBS samples for each of the 75 embeddable configurations. This was done on the same physical device (named X8_01). Although this statistics may be insufficient - as in the graph similarity study of [2], each configuration used 20 million samples - it was not possible to get such high precision statistics by the time of the study.

We used the following code to generate the GBS outputs. This code was adapted from the Strawberryfields 'X8 tutorial' demo available here: https://strawberryfields.ai/photonics/demos/tutorial_X8.html

```python
import strawberryfields as sf
nb_shot = 100000
for A, m in graph_parameters:
    eng = RemoteEngine("X8")
    prog = sf.Program(8)
    with prog.context as q:
        sf.ops.BipartiteGraphEmbed(A, mean_photon_per_mode=m) | q
        sf.ops.MeasureFock() | q
    spec = eng.device_spec
    prog.compile(device=spec)
    run_prog = eng.run(prog, shots=nb_shot)
    GBS_samples = run_prog.samples
```

with A the adjacency matrix of the embedded graph and m the mean photon per mode parameter described in section 2.

In addition, we also simulated for each configuration 1,000 GBS samples (and not 100,000 because of time constraints since each configuration was about 90 minutes computation time with the threshold option off on a Dell XPS15 laptop with a i7-9750H processor and 16 GB RAM) using the same A matrix and the same m parameter with the sf.sample.sample function. The minimalist used code is the following:

```python
nb_shot = 1000
for A, m in graph_parameters:
    n_mean = 8 * m # m was "per mode", n_mean is for the whole device
    simulated_samples = sf.sample.sample(A, n_mean, nb_shot, threshold=False)
```

These sets of GBS samples were saved and used as input for the calculation of feature vectors.
5 Feature vectors determination: comparison between experimental sampling, simulated sampling and analytical calculation

We are now able to convert the hardware and simulated samples into "feature vectors" (fv) as described in [5] and in the sf tutorial as well, available at this url: [https://strawberryfields.ai/photonics/apps/run_tutorial_similarity.html](https://strawberryfields.ai/photonics/apps/run_tutorial_similarity.html). Feature vectors can "be viewed as a mapping to a high-dimensional feature space".

They can be built considering 'events' or 'orbits'. A n-photons event corresponds to n photons detected at the end of the GBS device. Orbits are sub parts of events and are the set of all GBS samples that are equivalent under permutation of the modes. We calculated feature vectors for all the possible events and orbits up to 8 photons. Since most feature vectors components were almost zero for lots of orbits, we first focused on feature vectors from events. Also, since the photons are emitted by pairs, all the odd number of photon events are null under the assumption of a lossless device.

The following minimalist code shows how the feature vectors were calculated on the hardware.

```python
1 event = [2, 4, 6, 8]
2 mcpm = 8 # max count per mode
3 for A, m in graph_parameters:
4     n_mean = 8*m # m was " per mode", n_mean is for the whole device
5     fv_GBS = sf.similarity.feature_vector_events_sampling(GBS_samples, event, mcpm)
6     fv_simu = sf.similarity.feature_vector_events_sampling(simulated_samples, event, mcpm)
7     fv_adj = sf.similarity.feature_vector_events(nx.Graph(A), event, mcpm, n_mean)
```

Feature vectors comparison

Figure 4 shows the third component of the feature vector $f_k=(2,4,6,8), n_{max}=8$ for all the 75 embeddable graphs, calculated by three methods:

- GBS: using the experimental samples
- sampling: using the simulated samples
- theory: using the adjacency matrix (analytical calculation).

Configurations are sorted on the x axis by isomorphic category. We notice that each graph of a given isomorphic category has the same theoretical feature vector, in agreement with theory. Furthermore, each graph of a given isomorphic category has a value around an average with a given dispersion (up to 15% for the GBS hardware sampled fv components and 25% for the classically simulated sampling fv components). Dispersion is bigger for the simulated ones since the number of simulated shots was very small (1,000 per graph). This low statistics on the simulated sampling prevents us to get relevant conclusions on this comparison.

Nevertheless, we observe the GBS components always have smaller values for events of 6 photons. We expect this to be due to photon loss, and try to verify this in the following section.

Taking photon loss into account

These comparisons were made under the assumption the systems were lossless. This is not the case for the real GBS device. In order to check this aspect, we considered a configuration (graph 1111111111) where theory and simulated sampling were in agreement but not the samples from the device and added a loss parameter for the theory. Indeed, the sf.similarity.feature_vector_events function has a 'loss' argument (set to zero as default) simulating photon losses.
Figure 4: Feature vector for event 6

Figure 5: Feature vectors components relative deviation as a function of the loss factor for the graph
Figure 5 shows the relative deviation from theory of the components of the feature vector with respect to different loss factors. One can see that this loss factor must be set to about 45% in order to match the first component and about 60% for the three others components. A constant loss factor cannot make an agreement between the theoretical feature vector and the one calculated from the GBS samples. Nevertheless, the loss factor must be set to a pretty large value (close to 60% photon loss) in order to get one component matching with the theory, which is in agreement with the loss factor estimated in [2] (about 8 dB integrated loss).

Working with orbits instead of events

Features vectors made from events do not seem to be the most efficient way to estimate if the GBS device can separate isomorphic groups. Since a proof of graph similarity has already been made using the X8 chip [2], we decided to use the same strategy: considering three different orbits, including odd ones (which can happen since the device has loss. We thus use the loss as a feature!) and representing graphs in this 3D space. The chosen orbits were the same as the original paper: [1, 1, 1], [1, 1, 1, 1], and [2, 1, 1].

Figure 6: Feature vectors made from orbits. The components of the vectors are probabilities for the orbits [1, 1, 1], [1, 1, 1, 1], and [2, 1, 1].

Results are shown on Fig. 6. Each isomorphic category is clustered and only two (2P3 and 2S3) show overlaps. This is a promising result regarding the loss of the device. The result could certainly be improved with a better statistics.

6 Conclusion

This technical note summarises a study made about unweighted graphs. We first characterized which unweighted graphs can be embedded into the X8 chip. We noticed 75 unweighted bipartite graphs can be embedded, distributed into 10 different isomorphic categories. Each graph was embedded into the X8 chip and GBS samples were collected. Similar samples were also simulated with a lower statistics. For all of these
configurations, feature vectors from events and orbits were calculated. Selecting relevant orbits (including odd ones) allows the classification of the isomorphic categories to a certain extent despite the 8 dB integrated loss of the device.

Acknowledgements

The authors want to thanks Xanadu for letting them use their device. A special attention for Thomas R. Bromley and Juan Miguel Arrazola for their relevant comments.

References

[1] Han-Sen Zhong, Hui Wang, Yu-Hao Deng, Ming-Cheng Chen, Li-Chao Peng, Yi-Han Luo, Jian Qin, Dian Wu, Xing Ding, Yi Hu, et al. Quantum computational advantage using photons. *Science*, 370(6523):1460–1463, 2020.

[2] J. M. Arrazola, V. Bergholm, K. Brádler, T. R. Bromley, M. J. Collins, I. Dhand, A. Fumagalli, T. Gerrits, A. Goussev, L. G. Helt, J. Hundal, T. Isacsson, R. B. Israel, J. Izaac, S. Jahangiri, R. Janik, N. Killoran, S. P. Kumar, J. Lavoie, A. E. Lita, D. H. Mahler, M. Menotti, B. Morrison, S. W. Nam, L. Neuhaus, H. Y. Qi, N. Quesada, A. Repingon, K. K. Sabapathy, M. Schuld, D. Su, J. Swinarton, A. Száva, K. Tan, P. Tan, V. D. Vaidya, Z. Vernon, Z. Zabaneh, and Y. Zhang. Quantum circuits with many photons on a programmable nanophotonic chip. *Nature*, 591(7848):54–60, Mar 2021.

[3] Thomas R Bromley, Juan Miguel Arrazola, Soran Jahangiri, Josh Izaac, Nicolás Quesada, Alain Delgado Gran, Maria Schuld, Jeremy Swinarton, Zeid Zabaneh, and Nathan Killoran. Applications of near-term photonic quantum computers: software and algorithms. *Quantum Science and Technology*, 5(3):034010, may 2020.

[4] Kamil Bradler, Shmuel Friedland, Josh Izaac, Nathan Killoran, and Daiqin Su. Graph isomorphism and Gaussian boson sampling. *arXiv e-prints*, page arXiv:1810.10644, October 2018.

[5] Maria Schuld, Kamil Brádler, Robert Israel, Daiqin Su, and Brajesh Gupt. Measuring the similarity of graphs with a gaussian boson sampler. *Phys. Rev. A*, 101:032314, Mar 2020.

[6] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks?, 2019.

[7] Guillaume Verdon, Trevor McCourt, Enxhell Luzhnica, Vikash Singh, Stefan Leichenauer, and Jack Hidary. Quantum graph neural networks, 2019.

[8] Riccardo Mengoni, Massimiliano Incudini, and Alessandra Di Pierro. Facial expression recognition on a quantum computer. *Quantum Machine Intelligence*, 3(1):8, Mar 2021.
A Appendix: the 10 different graphs

The coming plots show the ten different graphs, depicted in two ways: left is by avoiding edge crossing and right is in a more usual bipartite way.

Figure 7: The '1K2' graph

Figure 8: The '2K2' graph
Figure 9: The '1C4' graph

Figure 10: The '2P3' graph

Figure 11: The '3K2' graph
Figure 12: The '1K33' graph

Figure 13: The '2S3' graph

Figure 14: The '4K2' graph
Figure 15: The "2C4" graph

Figure 16: The "1K44" graph