Internship Project Report: Exploration of Quantum-Enhanced Initialization for Jaccard-Weighted Densest Subgraphs

Yashaswini Mathur[1]

[1] TCS Research & Innovation, Bangalore
[1] Indian Institute of Science Education and Research, Bhopal

Abstract :

This report details an exploratory study into enhancing the initialization phase of the Iterative (ITR) algorithm for the Jaccard-Weighted Densest Subgraph (JWDS) problem, leveraging Gaussian Boson Sampling (GBS). The JWDS problem, which combines subgraph density maximization with Jaccard similarity constraints across multiple graphs, is a computationally challenging NP-hard variant of the Densest Subgraph Problem (DSP). The classical ITR algorithm offers a 2-approximation for JWDS, with its performance significantly influenced by the quality of its initial sets (the densest common subgraph and individual snapshot densest subgraphs).

We investigate the feasibility and potential benefits of utilizing GBS as a subroutine to generate these critical initial sets. GBS, a photonic quantum computing paradigm, is explored for its inherent bias towards sampling dense subgraphs and its capability to generate graph feature vectors for similarity analysis. This report outlines the proposed hybrid approach, detailing the GBS-assisted initialization and the subsequent classical execution of the ITR algorithm. We present preliminary classical implementations and discuss the planned methodology for simulating the GBS components and empirically comparing the performance of GBS-initialized ITR against its traditional classical initialization. This work aims to lay the groundwork for understanding how quantum technologies can offer practical enhancements to established classical approximation algorithms for complex graph optimization problems.

Index Terms: Gaussian Boson Sampling, Densest Subgraph Problem, Hybrid Quantum-Classical Algorithms, Jaccard Similarity, Graph Optimization, Internship Project.

1 Introduction

The Densest Subgraph Problem (DSP) is a fundamental and extensively studied problem in graph mining, focused on identifying a subset of vertices within a given graph whose induced subgraph maximizes a specific measure of density (Jethava et al. (2015), Rozenshtein et al. (2020)). Though there are other definitions of density such as the proportion of edges, density is most commonly defined as the ratio of the number of edges within the subgraph to the number of its vertices. Note that the densest subgraph can be found in polynomial time (Goldberg (1984)) and approximated efficiently (Charikar (2000)).

The enduring significance of DSP stems from its wide-ranging applications across diverse fields, including the detection of communities in social networks (Semertzidis et al. (2019)), identification of molecular complexes in bioinformatics (Fratkin et al. (2006)), and uncovering fraudulent activities in financial markets (Du et al. (2009)).

While the basic DSP is polynomial-time solvable, many of its practically relevant variants are computationally intractable, falling into the NP-hard complexity class. These complex variants often involve constraints on the size of the subgraph (e.g., Densest k-Subgraph (DkS)) (Khot (2006), Andersen et al. (2009), Khuller et al. (2009)), or analysis across multiple graphs with specific similarity requirements (Jethava et al. (2015)). A particularly challenging variant is the Jaccard-Weighted Densest Subgraph (JWDS) problem, which seeks to maximize a combined objective of subgraph density and Jaccard similarity across a sequence of graphs.

For the JWDS problem, the classical Iterative (ITR) algorithm (Arachchi et al. (2024)) offers a notable solution. This algorithm is a greedy heuristic that iteratively improves a solution, and importantly, it has been shown to be a 2-approximation algorithm for JWDS. We suspect the performance of ITR is significantly influenced by the quality of its initial sets, which typically include the densest common subgraph and the densest subgraphs for each individual snapshot.

The inherent computational hardness of these real-world DSP variants, especially those with size constraints, motivates the exploration of alternative computational paradigms, including quantum computing approaches. Gaussian Boson Sampling (GBS), a non-universal photonic quantum computing paradigm, is particularly promising due to its unique mathematical association with graph properties, enabling it to sample dense subgraphs with high probability (Arrazola et al. (2018)). This capability positions GBS as a valuable tool for generating high-quality initial solutions for classical optimization algorithms.

This report presents an exploratory study, conducted as part of an internship project at TCS Research, Bangalore, into a hybrid quantum-classical approach for the JWDS problem. Our objective is to explore and outline a strategy for enhancing the ITR algorithm's initialization phase by leveraging GBS. We will discuss preliminary findings from classical implementations and initial GBS code, and detail a comprehensive plan for future simulations.

1.1 Report Organisation

This report is structured to provide a comprehensive overview of the Densest Subgraph Problem and the proposed quantum-enhanced approach. Section 1 introduces the problem, its importance, and the motivation for exploring quantum solutions. Section 2 (Background) provides a detailed review of the Jaccard-Weighted Densest Subgraph problem and the ITR algorithm, and an introduction to Gaussian Boson Sampling and hybrid quantum-classical algorithms. Section 3 (Proposed Approach) will detail the hybrid initialization strategy for ITR using GBS, along with preliminary classical and GBS code implementations. Section 4 (Future Work) will outline the comprehensive plan for future simulations and comparative studies. Finally, Section 5 (Conclusion) will summarize the key findings and discuss the implications for future research.

2 Background

This section provides a comprehensive review of relevant classical and quantum concepts essential for understanding the proposed hybrid approach to the Jaccard-Weighted Densest Subgraph problem.

2.1 Classical Densest Subgraph Problems

The Densest Subgraph Problem (DSP) is a fundamental problem in graph mining, aiming to find a subset of vertices whose induced subgraph maximizes a measure of density. The most common definition of density is the ratio of the number of edges within the subgraph to the number of its vertices, often referred to as average degree, $d(S) = |E(S)|/|S|$. This objective function is equivalent to half of the average degree of the vertices within the subgraph. While this basic DSP is polynomial-time solvable using techniques like max-flow algorithms, many of its practically relevant variants are computationally intractable, falling into the NP-hard class.

Key NP-hard DSP variants include:

- **Densest k-Subgraph Problem (DkS):** This variant seeks a subgraph containing exactly k vertices that maximizes density. DkS is NP-hard, generalizing the maximum clique problem.
- **Densest at-least-k Subgraph Problem (DalkS):** This problem aims to maximize density subject to the subgraph containing at least k vertices. It is NP-hard.
- **Densest at-most-k Subgraph Problem (DamkS):** This variant maximizes density under the constraint that the subgraph contains at most k vertices. It is NP-hard and considered as hard to approximate as DkS.

The NP-hardness of these size-constrained problems implies that finding exact optimal solutions for large-scale graphs is generally infeasible within practical timeframes. This necessitates a shift towards approximation algorithms. For DkS, the current best-known polynomial-time approximation ratio is $O(n^{1/4+\epsilon})$. For DalkS, a linear-time 3-approximation algorithm exists.

Classical approximation algorithms for DSP include Charikar's Greedy Peeling Algorithm, which provides a 1/2-approximation guarantee for the classic DSP in linear time. Iterative variants like Greedy++ and FISTA aim to converge closer to the optimal solution. For finding multiple dense subgraphs, naive iterative removal is often impractical. Specialized heuristics like Iterative Weighted Dense Subgraphs (IWDS) are used for overlapping subgraphs.

2.2 The Jaccard-Weighted Densest Subgraph (JWDS) Problem

The Jaccard-Weighted Densest Subgraph (JWDS) problem is a complex variant of DSP that arises in multi-graph analysis. It aims to find a collection of vertex subsets $S = (S_1, \dots, S_k)$ from a sequence of graphs $G = (G_1, \dots, G_k)$ that maximizes a combined objective function $q(S)$, which incorporates both subgraph density and Jaccard similarity. The objective function is typically defined as $q(S) = d(S) + f(S)$, where $d(S)$ is the sum of individual subgraph densities and $f(S)$ is a term related to the pairwise Jaccard similarity between the selected subgraphs S_i and S_j. The Jaccard index, $J(S_i, S_j) = |S_i \cap S_j|/|S_i \cup S_j|$, quantifies the similarity between two sets.

The classical Iterative (ITR) algorithm is a greedy heuristic designed to solve the JWDS problem. It operates by iteratively improving a solution, and its performance is significantly influenced by the quality of its initial sets. The algorithm typically starts by initializing the subgraph sequence with the densest common subgraph (JCDS with $\epsilon = 1$) and the sequence of densest subgraphs for each individual snapshot. ITR is a 2-approximation algorithm for JWDS.

2.3 Gaussian Boson Sampling (GBS)

Gaussian Boson Sampling (GBS) is a non-universal photonic quantum computing paradigm that has emerged as a leading candidate for demonstrating quantum computational advantage. It operates by preparing squeezed states of light, entangling them within a linear interferometer, and then measuring the photon-counting output distribution. The probability of observing a specific photon-counting outcome in GBS is proportional to the Hafnian of a submatrix derived from the encoded graph's adjacency matrix. The Hafnian directly counts the number of perfect matchings in a graph, which is strongly correlated with graph density.

This fundamental link gives GBS its inherent ability to sample dense subgraphs with high probability. This property can be leveraged to identify dense subgraphs by generating numerous samples and applying classical post-processing. For instance, Strawberry Fields provides functions to search for collections of dense subgraphs across a range of sizes from GBS samples. GBS has been shown to enhance classical stochastic algorithms for NP-hard problems like DkS.

Beyond dense subgraph identification, GBS is also capable of generating graph feature vectors and kernels for graph similarity analysis. This involves associating features with the relative frequencies of specific photon-counting measurements, which form a high-dimensional feature vector representing the graph. GBS-derived kernels can capture global properties of graphs, such as the girth or chromatic number, a key advantage over many traditional graph kernels that have an inherently local perspective.

Despite its potential, GBS-based approaches face limitations in the Noisy Intermediate-Scale Quantum (NISQ) era. These include inevitable noise (photon loss, spectral impurity, thermal noise), which can lead to classical simulability beyond a certain threshold. GBS is primarily a sampling method, not a direct optimization algorithm, and its samples may exhibit biases (e.g., towards even-sized subgraphs). Therefore, classical post-processing is crucial to refine GBS outputs and integrate them into a broader workflow.

2.4 Hybrid Quantum-Classical Algorithms

Hybrid quantum-classical (HQC) algorithms represent a powerful paradigm for leveraging the strengths of both quantum and classical computing resources. This approach is particularly relevant for current Noisy Intermediate-Scale Quantum (NISQ) devices, which are prone to noise and have limited qubit counts, making full fault-tolerant quantum computation impractical.

In a hybrid framework:

- **Quantum Processors** are utilized for specific, computationally intensive sub-tasks where quantum mechanics can offer

an advantage. This includes tasks like sampling from complex probability distributions (as in GBS) or evaluating expectation values of cost functions (as in variational quantum algorithms like QAOA).
- **Classical Computers** handle the complementary tasks, which include problem formulation, parameter optimization, error mitigation, and crucial pre- and post-processing steps. For instance, classical post-processing is essential for filtering and refining the probabilistic outputs from quantum devices to extract high-quality solutions that meet specific constraints.

3 Proposed Approach

This section details the proposed hybrid quantum-classical approach for the Jaccard-Weighted Densest Subgraph (JWDS) problem and the preliminary implementations completed.

Jaccard-Weighted Densest Subgraph (JWDS) Problem: Given a graph sequence $\mathcal{G} = (G_1, \ldots, G_k)$, where each $G_i = (V, E_i)$ is an undirected graph, and a real number λ, find a collection of vertex subsets $S = (S_1, \ldots, S_k)$, such that $q(S)$ is maximized.

The objective $q(S)$, incorporates both subgraph density and Jaccard similarity. It is formally defined as

$$q(S) = d(S) + \lambda \sum_{i=1}^{k} \sum_{j=i+1}^{k} J(S_i, S_j)$$

where $d(S)$ is the sum of individual subgraph densities ($d(S_i) = |E(S_i)|/|S_i|$) and the Jaccard index, $J(S_i, S_j) = |S_i \cap S_j|/|S_i \cup S_j|$, quantifies the similarity between two sets.

3.1 The ITR Algorithm

Algorithm 1 ITR Algorithm (Algorithm 2 in [])

```
Require: Graph sequence $G = (G_1, \ldots, G_k)$, parameter $\lambda$
Ensure: Subgraph sequence $S = (S_1, \ldots, S_k)$ with good $q(S)$
1:  Initialize $S$ to be the densest common subgraph.
2:  Initialize $S'$ to be the sequence of densest subgraphs for each snapshot.
3:  Current best $S_{best} \leftarrow S$
4:  repeat
5:      $S_{prev} \leftarrow S$
6:      for each snapshot $G_i$ do
7:          for each vertex $v \in V$ do
8:              Consider adding $v$ to $S_i$ or removing $v$ from $S_i$.
9:              Calculate density gain $d(S_{new}) - d(S_{old})$.
10:             Check if Jaccard constraint $J(S_{new}, S_j) \geq \epsilon$ is satisfied for all $j \neq i$.
11:             If improvement and constraints satisfied, update $S_i$.
12:             Update sizes of intersections and unions for $S_i$ and $S_j$.
13:     end for
14:     If $q(S) > q(S_{best})$, then $S_{best} \leftarrow S$.
15: until $S$ converges (i.e., $S = S_{prev}$)
16: return $S_{best}$
```

The Iterative (ITR) algorithm is a classical greedy heuristic designed to solve the JWDS problem. Its performance is significantly influenced by the quality of its initial sets. The algorithm typically starts by initializing the subgraph sequence with the densest common subgraph (JCDS with $\alpha = 1$) and the sequence of densest subgraphs for each individual snapshot (JCDS with $\alpha = 0$). The algorithm then iteratively refines these sets by adding or removing vertices to improve the objective function $q(S)$ while satisfying the pairwise Jaccard similarity constraints. The process continues until convergence, ensuring the objective value does not decrease in each iteration.

The authors of Arachchi et al. (2024) state that due to the choice of these initial sets, ITR is a **2-approximation algorithm** for JWDS. This guarantee means that the solution found by ITR is at most twice the optimal solution value, provided the initial sets meet the theoretical requirements for maximizing the density and Jaccard terms, respectively.

3.2 Proposed GBS-Enhanced Initialization Strategy

We propose a novel hybrid quantum-classical approach to enhance the initialization phase of the ITR algorithm by leveraging Gaussian Boson Sampling (GBS). This strategy aims to provide higher-quality initial sets (S and S') for ITR, potentially leading to improved final solutions or faster convergence.

Using GBS for S' (Densest Subgraphs for Each Snapshot): For each individual graph G_i in the sequence, GBS is utilized to generate a pool of dense subgraph candidates. GBS is inherently biased towards sampling dense subgraphs with high probability when a graph's adjacency matrix is encoded into the device. This makes GBS a powerful heuristic for efficiently exploring the space of dense subgraphs, particularly for NP-hard variants like Densest k-Subgraph (DkS). The process involves:

1. Encoding the adjacency matrix of each G_i into a GBS device (or simulator).
2. Generating a large number of photon samples from the GBS device.
3. Post-processing these GBS samples to extract candidate subgraphs. Tools like 'strawberryfields.apps.subgraph.search()' can be used to identify collections of dense subgraphs for a range of sizes from these samples.
4. From the generated candidates for each G_i, the densest subgraph (or the densest subgraph within a specified size range) is selected to form the initial S'_i.

It is important to note that GBS provides a heuristic initialization. While it aims to provide high-quality starting points, the probabilistic nature of GBS sampling means that the 2-approximation guarantee of the original ITR algorithm does not strictly transfer to this hybrid variant. The empirical performance of this GBS-enhanced initialization will be the focus of our study.

3.3 Preliminary Implementations

As part of this internship project, preliminary implementations have been completed to lay the groundwork for the full comparative study:

- **Classical Implementations:** Core classical components of the ITR algorithm have been implemented, including functions for calculating subgraph density, computing Jaccard

similarity between vertex sets, and the iterative improvement loop of the ITR algorithm. These form the baseline for comparison.
- **GBS Code for Dense Subgraph:** Initial GBS code has been developed to demonstrate the capability of GBS to identify dense subgraphs of a specific size. This preliminary code utilizes GBS simulation to generate samples from a given graph and post-processes these samples to extract the densest subgraph within a predefined size constraint. This serves as a proof-of-concept for GBS's role in candidate generation.

3.4 *Experimental Setup*

- **Data:** We used custom randomly generated input for the initial analysis for the initial analysis. Further larger datasets would be required for better evaluation.
- **Simulation Environment:** Strawberry Fields was utilized for GBS simulation. Standard libraries such as numpy were used to support the code in Python.
- **Noise:** Though it was shown by [], that dense sbugraph sampling works well even with noise, these simulation are yet to be completed and not a part of this report. This would include simulating photon loss and thermal noise to assess the robustness of the GBS-generated initial sets.
- **Classical Baselines:** The performance of the GBS-enhanced ITR is compared against a classical baseline in which the ITR algorithm is initialized using one of two classical strategies:
  - **Densest Common Subgraph (S):** A greedy peeling-based approach is used to compute a common subset of vertices shared across all snapshots that maximizes the total sum of subgraph densities. This subset is then replicated across all snapshots to initialize the ITR algorithm.
  - **Separate Densest Subgraphs (S'):** For each snapshot G_i, a peeling-based densest subgraph algorithm is applied independently to obtain the densest subset S_i.

The code evaluates both initializations, computes their respective q-scores, and proceeds with the one that yields the higher initial score. This ensures a fair and adaptive classical baseline for comparison with the GBS-based variant. *Both strategies are inspired by Charikar's peeling algorithm for Densest Subgraph (Charikar (2000)), a known 2-approximation method, extended here to handle either shared or per-snapshot subgraph selection.*

3.5 *Evaluation of the Results*

We evaluate the performance of the JWDS ITR (Iterative Refinement) algorithm under two initialization strategies:

- **GBS-based Initialization**: Uses simulated Gaussian Boson Sampling to construct high-density, temporally aligned subgraphs with refinement.
- **Classical Initialization**: Identifies densest subgraphs in each graph independently using a greedy peeling algorithm.

Both methods are followed by the same iterative algorithm to optimize the objective:

$$q(S_1, \ldots, S_k; \lambda) = \sum_{i=1}^{k} d(S_i) + \lambda \sum_{1 \le i < j \le k} J(S_i, S_j)$$

where $d(S_i)$ is the density of subgraph S_i, and $J(S_i, S_j)$ is the Jaccard index between subgraphs S_i and S_j. We fix $\lambda = 0.3$.

Datasets and Graph Structures To analyze performance across different temporal structures, we construct and evaluate on two synthetic graph sequences.

Graph	Edges
G1	(a,b), (b,c), (c,d), (d,e), (e,f), (f,g), (g,h), (h,i), (i,j), (j,k), (k,l), (l,a)
G2	(a,c), (b,d), (c,e), (d,f), (e,g), (f,h), (g,i), (h,j), (i,k), (j,l)
G3	(a,b), (b,d), (c,e), (e,f), (f,g), (g,i), (h,j), (j,k), (k,l), (l,a)
G4	(a,e), (b,f), (c,g), (d,h), (e,i), (f,j), (g,k), (h,l), (i,a), (j,b)
G5	(a,d), (b,e), (c,f), (d,g), (e,h), (f,i), (g,j), (h,k), (i,l), (j,a)

Table 1
Edge sets for snapshots in Dataset A

Graph	Edges
G1	(a,b), (a,c), (a,d), (b,c), (b,d), (c,d), (a,e)
G2	(e,f), (f,g), (g,e), (e,h)
G3	(h,i), (i,j), (j,h), (i,a)
G4	(c,d), (c,f), (f,g), (d,g)
G5	(a,j), (j,e), (e,a)

Table 2
Edge sets for snapshots in Dataset B

We use the following metrics to compare the two approaches:

- **Initial q-score**: Objective value before ITR begins
- **Final q-score**: Objective after convergence
- **Runtime**: Time taken for initialization
- **Convergence Speed**: Number of ITR iterations until convergence
- **Robustness**: Output consistency across runs
- **Subgraph Quality**: Final subgraph densities and Jaccard overlaps

Table 3
Comparison of GBS-based vs Classical initialization on Dataset 1

Metric	**GBS-based**	**Classical**
Q_{init}	5.4000	**7.3333**
Q_{final}	6.1500	**7.3333**
Time (s)	14.71	**0.00**
Iterations	1	**1**
Avg. Density	0.734	**0.864**
Avg. Jaccard	0.88	**1.000**

3.5.1 Key Observations

- **Classical wins decisively in this case**: The classical initialization (using the common densest subgraph across all snapshots) achieves the maximum possible Jaccard overlap and higher densities, leading to superior initial and final q-scores.
- **GBS struggles on highly uniform data**: In this example, the graphs share a strong common core with near-complete overlap. GBS, being stochastic and diversity-oriented, fails to fully exploit this uniformity, selecting smaller subgraphs and yielding weaker q-scores.
- **Runtime differences are stark**: The GBS-based initialization is significantly slower than the classical greedy peeling approach.
- **Convergence is fast in both cases**: Both approaches converge within a single ITR iteration due to the stability of the subgraphs.

This example illustrates a key limitation of GBS-based initialization: when the optimal solution is a large, well-aligned common subgraph, classical heuristics can outperform.

Table 4
Comparison of GBS-based vs Classical initialization on Input 2

Metric	GBS-based	Classical
Q_{init}	4.2000	**6.0057**
Q_{final}	6.0057	6.0057
Time (s)	31.20	**0.01**
Iterations	1	1
Avg. Density	1.10	1.10
Avg. Jaccard	0.19	0.19

3.5.2 Key Observations

- **Classical initialization starts stronger**: The higher initial q-score suggests that in modular or clearly clustered graphs, classical peeling methods are better at immediately capturing high-density subgraphs.
- **Final results are identical**: Both methods converge to the same final q-score, densities, and overlaps—likely due to the strong convergence guarantees of the greedy ITR refinement stage.
- **GBS is significantly slower here**: The GBS-based initialization is computationally more intensive, yet offers no performance gain in this structured setting.
- **ITR dominates outcome quality**: The fact that both methods yield identical final results shows that the iterative refinement plays a decisive role, especially when initialized with reasonable subsets.

3.5.3 Overall Takeaways Across Examples

- **Classical methods thrive in structured data**: In both examples, classical initialization either matched or outperformed GBS in terms of initial q-score, particularly when clear clusterings or common structures exist.
- **GBS remains competitive**: Despite a weaker start in Input 2, GBS catches up fully post-refinement. Its value may lie more in irregular, noisy, or alignment-challenging instances.
- **Refinement is the great equalizer**: The greedy ITR procedure effectively compensates for weaker initialization. In simpler instances, this limits GBS's advantage—but also makes it a safe addition.
- **Use GBS adaptively**: GBS is most useful when classical heuristics struggle to produce coherent starting sets. For simple graphs, the overhead may not be justified.

4 Future Work

We outline the next stages of this research, including simulation plans, evaluation strategies, and anticipated outcomes of the proposed GBS-enhanced JWDS framework.

4.1 Detailed Simulation Plan

- **Graph Datasets:** Experiments will be conducted on both synthetic and real-world graph sequences. Synthetic datasets will include graphs with planted dense subgraphs or temporal drift, allowing controlled testing of algorithmic behavior. Real-world datasets (e.g., social, biological, or citation networks) will be sourced from standard repositories like SNAP or Network Repository. Each sequence will vary in number of graphs (k), size (up to a few hundred nodes), and edge density.
- **GBS Simulation Environment:** Gaussian Boson Sampling will be simulated using platforms such as `Strawberry Fields` or `PennyLane`. Parameters will include the number of modes (equal to the number of nodes), mean photon number (to tune sampling probability), and the number of samples per graph. These simulations aim to approximate the output of a realistic GBS device.
- **Noise Models:** To reflect practical constraints of near-term quantum hardware (NISQ), noise models such as photon loss, thermal noise, and detection inefficiencies will be incorporated into GBS simulations. The impact of noise on initialization quality and downstream performance will be studied.
- **Classical Baselines:** The classical ITR initialization will use two strategies: (1) greedy peeling-based algorithms for individual densest subgraphs, and (2) a shared subset approach for densest common subgraph across snapshots. Where applicable, classical approximation algorithms or flow-based methods for DCS will also be tested.

4.2 Expected Outcomes and Potential Impact

We hypothesize that GBS-based initialization, by leveraging quantum sampling to capture globally biased dense regions, will outperform classical methods on irregular or temporally misaligned graphs. Even when the final results converge, GBS may enable faster convergence or provide more diverse and robust subgraph candidates.

This hybrid quantum-classical approach has potential implications for broader classes of graph mining tasks. As GBS devices scale, their ability to assist in initializing combinatorial optimization algorithms could enhance current techniques in social network analysis, bioinformatics, and graph-based machine learning—fields of significant relevance to theoretical computer science and industry research.

4.3 Potential for Alternative Similarity Measures

Currently, the objective function uses the pairwise Jaccard index to encourage temporal consistency among subgraphs. While intuitive and computationally efficient, Jaccard may not fully capture nuanced structural similarities, especially in graphs with high vertex overlap but differing edge patterns.

We hypothesize that incorporating alternative or hybrid similarity measures—such as graph edit distance, spectral similarity, cosine similarity over embedding vectors, or soft neighborhood overlap—could significantly improve performance, particularly in datasets where structural consistency is more relevant than strict vertex identity.

Future work will explore these alternatives and evaluate their effect on:

- The expressiveness of the objective function.
- The quality and diversity of the final subgraph solutions.
- The comparative advantage of GBS-based vs. classical initialization under richer similarity models.

Preliminary intuition suggests that GBS-based initialization may show greater benefit when the similarity metric captures higher-order or fuzzy structural alignment, as it naturally samples dense regions with non-obvious correlations.

5 Conclusion

We tackled the Joint Weighted Densest Subgraph (JWDS) problem by introducing a hybrid quantum-classical algorithm that uses Gaussian Boson Sampling (GBS) for initialization, followed by a classical Iterative Refinement (ITR) phase. Our goal was to evaluate whether quantum-generated samples could improve the quality of subgraph initializations, potentially leading to better or faster convergence in optimizing a joint density-similarity objective.

In both tested scenarios, classical initialization strategies outperformed GBS-based initialization in terms of initial and final q-scores. This suggests that the classical peeling heuristics currently offer more effective and efficient initializations in the JWDS setting—especially when followed by a strong greedy ITR phase capable of rapidly improving suboptimal inputs. Nevertheless, GBS initialization did produce meaningful subgraph candidates with diverse structures and comparable densities, indicating that it captures valuable global structure. With further refinements—such as better feature encoding, adaptive sample filtering, or integration with modified similarity metrics—GBS could yet serve as a useful module in hybrid algorithms for complex graph problems.

While current results do not demonstrate a practical advantage, they underline the potential of GBS as a structurally meaningful biasing mechanism. The hybrid approach remains promising in theory, particularly for problems where classical initialization is difficult or inconsistent. Future work should explore GBS-informed similarity functions, real-device evaluations, and hybrid frameworks for related problems in graph mining and temporal networks.

Acknowledgments

This project was conducted as part of an internship at TCS Research & Innovation. I would like to thank for the opportunity to explore this research area. I also thank my supervisor, Mr. M Girish, and Mr. Prabhat Anand, for their guidance throughout the project.

References

Andersen, Reid and Kumar Chellapilla (2009). "Finding Dense Subgraphs with Size Bounds". In: *Proceedings of the 6th Workshop on Algorithms and Models for the Web-Graph (WAW)*. Vol. 5427. Lecture Notes in Computer Science. Springer, pp. 25–37. DOI: 10.1007/978-3-642-03647-3_4.

Arachchi, Chamalee Wickrama and Nikolaj Tatti (Sept. 2024). "Jaccard-constrained dense subgraph discovery". In: *Machine Learning* 113. Published online: 23 July 2024, pp. 7103–7125. DOI: 10.1007/s10994-024-06595-y. URL: https://doi.org/10.1007/s10994-024-06595-y.

Arrazola, Juan Miguel and Thomas R. Bromley (2018). "Using Gaussian Boson Sampling to Find Dense Subgraphs". In: *Physical Review Letters* 121.3, p. 030503. DOI: 10.1103/PhysRevLett.121.030503. URL: https://doi.org/10.1103/PhysRevLett.121.030503.

Charikar, Moses (2000). "Greedy Approximation Algorithms for Finding Dense Components in a Graph". In: *Proceedings of the 3rd International Workshop on Approximation Algorithms for Combinatorial Optimization (APPROX)*. Vol. 1913. Lecture Notes in Computer Science. Springer, pp. 84–95. DOI: 10.1007/3-540-44436-X_9.

Du, Xiaowei et al. (2009). "Migration Motif: A Spatial-Temporal Pattern Mining Approach for Financial Markets". In: *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. ACM, pp. 1135–1144. DOI: 10.1145/1557019.1557142.

Fratkin, Eliot et al. (2006). "MotifCut: Regulatory Motifs Finding with Maximum Density Subgraphs". In: *Bioinformatics* 22.14, e150–e157. DOI: 10.1093/bioinformatics/btl223.

Goldberg, Andrew V. (1984). *Finding a Maximum Density Subgraph*. Technical Report UCB/CSD-84-171. University of California, Berkeley. URL: https://people.eecs.berkeley.edu/~avg/papers/maxdens.pdf.

Jethava, Vinay and Niko Beerenwinkel (2015). "Finding Dense Subgraphs in Relational Graphs". In: *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD)*. Vol. 9284. Lecture Notes in Computer Science. Springer, pp. 641–654. DOI: 10.1007/978-3-319-23525-7_39.

Khot, Subhash (2006). "Ruling Out PTAS for Graph Min-Bisection, Dense k-Subgraph, and Bipartite Clique". In: *SIAM Journal on Computing* 36.4, pp. 1025–1071. DOI: 10.1137/S0097539705447279.

Khuller, Samir and Barna Saha (2009). "On Finding Dense Subgraphs". In: *Proceedings of the 36th International Colloquium on Automata, Languages and Programming (ICALP)*. Vol. 5555. Lecture Notes in Computer Science. Springer, pp. 597–608. DOI: 10.1007/978-3-642-02930-1_50.

Rozenshtein, Pavel et al. (2020). "Finding Events in Temporal Networks: Segmentation Meets Densest Subgraph Discovery". In: *Knowledge and Information Systems (KAIS)* 62.4, pp. 1611–1639. DOI: 10.1007/s10115-019-01383-3.

Semertzidis, Konstantinos et al. (2019). "Finding Lasting Dense Subgraphs". In: *Data Mining and Knowledge Discovery* 33.5, pp. 1417–1445. DOI: 10.1007/s10618-019-00650-0.

Code and Walkthrough Repository

All code files, GBS circuit notebooks, and walkthrough appendices for this project are available at: https://github.com/Picozen2718/Summer_Internship25_TCS_Research

This includes:

- Classical and quantum-enhanced JWDS algorithms
- Strawberry Fields simulation notebooks
- Graph similarity and metric evaluation code
- Appendix walkthroughs and supporting scripts

Study Materials and Supporting Notes

This appendix contains references to additional material studied during the course of the project. These documents supported the design and implementation of the hybrid quantum-classical framework.

A Building a Basic GBS Circuit and Sampling Subgraphs

A.1 Background

Boson and fermions, are the two elementary classes of particles. The most prevalent bosonic system in our everyday lives is light. The distinguishing characteristic of bosons is that they follow "Bose-Einstein statistics" - the particles like to bunch together (contrast this to fermionic matter like electrons, which must follow the Pauli Exclusion Principle and keep apart).

Boson Sampling: Aaronson and Arkhipov's original proposal was to inject many single photons into distinct input ports of a large interferometer, then measure which output ports they appear at. The natural interference properties of bosons means that photons will appear at the output ports in very unique and specific ways. Physical implementations of boson sampling make use of a process known as spontaneous parametric down-conversion to generate the single-photon source inputs.

However, this method is non-deterministic — as the number of modes in the apparatus increases, the average time required until every photon source emits a simultaneous photon increases exponentially. Hence we now use states of light that are experimentally less demanding (though still challenging!). These states of light are called Gaussian states, because they bear strong connections to the Gaussian (or Normal) distribution from statistics. In practice, we use a particular Gaussian state called a squeezed state for the inputs. In this case, single mode squeezed states — can produce problems in the same computational complexity class as boson sampling. Even more significantly, this negates the scalability problem with single photon sources, as single mode squeezed states can be deterministically generated simultaneously.

A.2 Squeezed Gaussian States

Our starting point is the vacuum state. Other states can be created by evolving the vacuum state according to:

$$|\psi\rangle = exp(-itH)|0\rangle$$

where H is a bosonic Hamiltonian and t is the evolution of time.

State where the Hamiltonian is at most quadratic in $\hat{x}$ *and* $\hat{p}$ are called Gaussian. For a single qumode, Gaussian states are parameterized by 2 continuous complex variables. The displacement parameter, $\alpha \in C$. And the squeezing parameter, $z \in C$ ($z = re^{i\phi}$ *with* $r \geq 0$). Given a Gaussian distribution, we can identify each state, through it's displacement (centre of Gaussian) and squeezing (variation and rotation) parameters. Squeezed states are a family of Gaussian states where the displacement is 0 and squeezing parameter is free.

A.2.1 Some useful points

- Unitary operations can always be associated with a generating Hamiltonian. Gaussian are those with degree one or two of the generating Hamiltonians.

$$U = exp(-itH)$$

- Number states (Fock states) are eigenstate of the number operator $\hat{n} = \hat{a}^\dagger \hat{a}$. They form a discrete countable basis for a single qumode. And all Gaussian states can be written in this basis.
- Note that a single-mode squeezed vacuum state is a superposition of only even photon number states. Therefore, if you measure the photon number in a single squeezed vacuum mode, you will only ever detect an even number of photons (0, 2, 4, ...).
- A squeezed state is a minimum uncertainty state with unequal quadrature variances, and satisfies the following eigenstate equation:

$$\left(\hat{a}\cosh(r) + \hat{a}^\dagger e^{i\phi}\sinh(r)\right)|z\rangle = 0$$

In the Fock basis, it has the decomposition

$$|z\rangle = \frac{1}{\sqrt{\cosh(r)}}\sum_{n=0}^{\infty}\frac{\sqrt{(2n)!}}{2^n n!}\left(-e^{i\phi}\tanh(r)\right)^n|2n\rangle$$

whilst in the Gaussian formulation, $\mathbf{r} = (0,0)$,

$$\mathbf{V} = \frac{\hbar}{2}R(\phi/2)\begin{bmatrix} e^{-2r} & 0 \\ 0 & e^{2r}\end{bmatrix}R(\phi/2)^T.$$

We can use the squeezed vacuum state to approximate the zero position and zero momentum eigenstates:

$$|0_x\rangle \approx S(r)|0\rangle, \quad |0_p\rangle \approx S(-r)|0\rangle$$

where $z = r$ is sufficiently large.

- If you have just one qumode (a single quantum mode) prepared in a squeezed vacuum state, and you increase your simulation cutoff high enough, you will see a non-zero probability of detecting 8 photons (or any other even number of photons) from that single mode. That is the probability generally decreases as the photon number increases, but it's never zero for any even number (theoretically, it extends infinitely).

A.3 Hafnian: Source of Hardness

Finding the permanent is $\#P - Hard$ and by extension so is Boson Sampling.

Using phase space methods, we can show that the probability of measuring a Fock state containing only 0 or 1 photons per mode is given by

$$|\langle n_1, n_2, \ldots, n_N \mid \psi\rangle|^2 = \frac{\left|\mathrm{Haf}\left((U(\bigoplus_i \tanh(r_i))U^T)_{S,T}\right)\right|^2}{\prod_{i=1}^{N}\cosh(r_i)}$$

i.e., the sampled single photon probability distribution is proportional to the **hafnian** of a submatrix of $U\left(\bigoplus_i \tanh(r_i)\right)U^T$, dependent upon the output covariance matrix.

The hafnian of a matrix is defined by

$$\mathrm{Haf}(A) = \frac{1}{n!2^n}\sum_{\sigma\in S_{2N}}\prod_{i=1}^{N}A_{\sigma(2i-1)\sigma(2i)}$$

where S_{2N} is the set of all permutations of $2N$ elements. In graph theory, the hafnian calculates the number of perfect matchings in an arbitrary graph with adjacency matrix A.

Compare this to the permanent, which calculates the number of perfect matchings on a *bipartite* graph — the hafnian turns out to be a generalization of the permanent, with the relationship

$$\text{Per}(A) = \text{Haf}\left(\begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix}\right)$$

As any algorithm that could calculate (or even approximate) the hafnian could also calculate the permanent, a #P-hard problem, it follows that calculating or approximating the hafnian must also be a classically hard problem.

A.4 Sampling Subgraphs Using GBS

Strawberry Fields provides a convenient interface to sample dense subgraphs via the `sample` module. Given an adjacency matrix A, a GBS device can sample from its encoded Gaussian state.

The density of a k-node subgraph is equal to the number of its edges divided by the maximum possible number of edges. Identifying the densest graph of a given size, known as the densest-k subgraph problem, is NP-Hard.

It was found that a defining feature of GBS is that when we encode a graph into a GBS device, it samples dense subgraphs with high probability. This property can be used to find dense subgraphs by sampling from a GBS device and postprocessing the outputs.

The challenge is to find dense subgraphs of different sizes; it is often useful to identify many high-density subgraphs, not just the densest ones.

This is the purpose of the `search()` function in the `subgraph` module: to identify collections of dense subgraphs for a range of sizes.

The output of this function is a dictionary whose keys correspond to subgraph sizes within the specified range. The values in the dictionary are the top subgraphs of that size and their corresponding density.

A.5 Resources Consulted

- Strawberry Fields documentation: `https://strawberryfields.ai/photonics/`
- PennyLane GBS demo: `https://pennylane.ai/qml/demos/gbs`
- GBS tutorial: `https://strawberryfields.ai/photonics/demos/run_gaussian_boson_sampling.html`
- Walrus library for hafnian: `https://the-walrus.readthedocs.io/en/latest/index.html`
- Pizzimenti term paper: `https://wp.optics.arizona.edu/opti646/wp-content/uploads/sites/55/2022/12/Pizzimenti_Term_Paper.pdf`
- Relevant arXiv papers: `https://arxiv.org/pdf/2402.16341`, `https://arxiv.org/pdf/1801.07488`

B Appendix B: Variational GBS and Training Tasks

B.1 Introduction and Background

Many quantum algorithms rely on the ability to train the parameters of quantum circuits.

Training is often performed by evaluating gradients of a cost function with respect to circuit parameters, then employing gradient-based optimization methods.

The probability of observing the output $S = (s_1, s_2, ..., s_m)$ $, where\ s_i$ is the number of photons detected at the i-th mode, is given by:

$$Pr(S) = \frac{1}{\mathcal{N}} \frac{|Haf(A_s)|^2}{s_1!...s_m!}$$

A is an arbitrary symmetric matrix with eigenvalues bounded between -1 and 1. The matrix can also be rescaled by a constant factor, which is equivalent to fixing a total mean photon number in the distribution.

We want to train this distribution to perform a specific task. Example: Reproduce the statistical properties of a dataset, or optimize a circuit to specific pattern with high probability.

So, we want to find a parametrization (assignment of parameters) of the distribution (probability of GBS samples) and optimize using gradient based techniques. - Called Variational GBS (VGBS)

But, gradients can be challenging to compute. So we do the following to A. (WAW parameterization) - $A \rightarrow A_W = WAW$ (W is a diagonal matrix) This is easier because we have the following factorization. $Haf(A_W) = Haf(A)det(W)$ Also we can embed trainable parameter $\theta = (\theta_1, ..., \theta_d)$ into the diagonal elements of W. (Ex: an exponential embedding)

We look at the following 2 specific training tasks.

B.2 Stochastic Optimization Task

Different from deterministic optimization, stochastic optimization algorithms employ processes with random factors to do so.

Due to these processes with random factors, stochastic optimization does not guarantee finding the optimal result for a given problem. But, there is always a probability of finding the globally optimal result.

Specifically, the probability of finding the globally optimal result in stochastic optimization relates to the available computing time. The probability of finding the globally optimal result increases as the execution time increases.

Stochastic cost function given by averaging over samples from a trainable GBS distribution.

A stochastic optimization problem is defined with respect to a function $h(\bar{n})$ that assigns a cost to an input sample $\bar{n}$. The cost function is the average of $h(\bar{n})$ over samples generated from a parametrized distribution $P_\theta(\bar{n})$:

$$C(\theta) = \sum_{\bar{n}} h(\bar{n}) P_\theta(\bar{n})$$

The cost function $C(\theta)$ can then be optimized by varying $P_\theta(\bar{n})$.

In this setting, $P_\theta(\bar{n})$ is the variational GBS distribution and is specified in `Stochastic` by an instance of `VGBS`.

The gradient of the cost function $C(\theta)$ is given by

$$\partial_\theta C(\theta) = \sum_S h(S) P_\theta(S) \sum_{k=1}^{m} (s_k - \langle s_k \rangle)\, \partial_\theta \log w_k,$$

where $\langle s_k \rangle$ denotes the average photon number in mode k. This gradient is an expectation value with respect to the GBS distribution, so it can be estimated by generating samples from the device.

Note: The transformed WAW matrix needs to have eigenvalues bounded between -1 and 1, so continuing training indefinitely can lead to unphysical distributions when the weights become too large. It's important to monitor this behavior.

We can see that the parameters now work well.

B.3 Unsupervised Learning Task

Data are assumed to be sampled from an unknown distribution and a common goal is to learn that distribution.

That is the goal is to use the data to train a model that can sample from a similar distribution, thus being able to generate new data.

Training can be performed by minimizing the Kullback-Leibler (KL) divergence.

$$KL(\theta) = -\frac{1}{T}\sum_{S}\log[P_\theta(S)]$$

In this case S is an element of the data, $P(S)$ is the probability of observing that element when sampling from the GBS distribution, and T is the total number of elements in the data. For the GBS distribution in the WAW parametrization, the gradient of the KL divergence can be written as

$$\partial_\theta KL(\theta) = -\sum_{k=1}^{m}\frac{1}{w_k}\left(\langle s_k\rangle_{\text{data}} - \langle s_k\rangle_{\text{GBS}}\right)\partial_\theta w_k$$

For unsupervised learning with VGBS (using KL divergence and WAW parametrization), we can efficiently calculate the gradient of the cost function using a classical computer. This makes the training process very fast and practical, even though the final goal is to leverage the quantum computer's unique ability to generate samples from a distribution that is otherwise classically intractable.

B.4 References

- Strawberry Fields VGBS tutorial: `https://strawberryfields.ai/photonics/apps/run_tutorial_training.html`
- Naomi Solomons' Thesis and talk on GBS enhanced dense subgraph finding: `https://youtu.be/ZVtjY7vSLhE`
- API Reference: `https://strawberryfields.readthedocs.io/en/stable/code/api/strawberryfields.apps.train.VGBS.html`
- Unsupervised GBS demo: `https://strawberryfields.readthedocs.io/en/stable/code/api/strawberryfields.apps.data.Planted.html`
- Pennylane glossary: `https://pennylane.ai/qml/glossary/variational_circuit`
- Optimization comparison: `https://www.baeldung.com/cs/deterministic-stochastic-optimization`

C Appendix C: GBS Graph Similarity and Kernel Methods

Gaussian Boson Sampling can be used to construct a similarity measure between graphs, known as a *graph kernel.* Such kernels are useful in machine learning tasks involving graph-structured data, such as classification using support vector machines (SVMs).

The graph data used is from a dataset of 188 graphs each corresponding to the structure of a chemical compound. (MUTAG) - There are 20000 samples for 4 graphs. - 1 and 4 are isomorphic (similar) and 2 and 3 are similar. The data module has pre-calculated GBS samples for MUTAG dataset.

You get each of these samples by encoding the graph into a GBS device and collecting photon click events.

We want to create a feature vector to describe each graph. It can be composed in many ways. Our approach is to associate features of with the relative frequencies of certain types of measurement being recorded from a GBS device configured to sample from the graph.

- **Orbits:** Collections of samples that are identical up to permutation of modes. Example: $[1,1,2,0]$ and $[2,1,0,1]$ both belong to orbit $[2,1,1]$.
- **Events:** Coarse-grained groupings of orbits, denoted $E_{k,n_{\max}}$, containing all k-photon orbits with mode occupancy bounded by $n_{\max}$.

The similarity module provides the following tools for dealing with coarse-grained orbits and events.

C.1 Evaluating Feature Vectors

Three methods to compute the probability of orbits or events:

- **Sampling-based:** Estimate probabilities from observed samples.
- **Monte Carlo estimation:** Uniformly sample events and calculate exact probabilities.
- **Exact computation:** Compute all hafnians exactly (computationally intensive).

C.2 Applications

Once feature vectors are obtained, they can be used in downstream machine learning tasks, such as training a support vector machine (SVM) for graph classification.

C.3 References

- GBS similarity tutorial: `https://strawberryfields.ai/photonics/apps/run_tutorial_similarity.html`
- MUTAG dataset: `https://strawberryfields.readthedocs.io/en/stable/code/api/strawberryfields.apps.data.Mutag0.html`
- GBS similarity API: `https://strawberryfields.readthedocs.io/en/stable/code/api/strawberryfields.apps.similarity.html`
- Support vector machines: `https://en.wikipedia.org/wiki/Support_vector_machine`