NExt ApplicationS of Quantum Computing

D5.11: Software package for mesh segmentation tasks

Document Properties

Contract Number	951821
Contractual Deadline	30-10-2024
Dissemination Level	Public
Nature	Software
Editors	Yagnik Chatterjee, TotalEnergies, Université de Montpellier
Authors	Yagnik Chatterjee, TotalEnergies, Université de Montpellier Henri Calandra, TotalEnergies
Reviewers	Vicente Moret Bonillo, Universidade da Coruna – CITIC Gonzalo Ferro Costas, CESGA
Date	21-11-2024
Keywords	Quantum variational algorithms, combinatorial optimization, mesh segmentation
Status	Submitted
Release	1.0

This project has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement No. 951821

History of Changes

Release	Date	Author, Organisation	Description of Changes
0.1	17/09/2024	Yagnik Chatterjee, TotalEnergies, Université de Montpellier Henri Calandra, TotalEnergies	Submitted for first review
1.0	21/11/2024	Yagnik Chatterjee, TotalEnergies, Université de Montpellier	Final Submission

Table of Contents

1 Executive Summary

This report constitutes the Deliverable 5.11 of Task 5.4 of the NEASQC Project.

In Deliverable 5.9, we demonstrated the LogQ encoding (Chatterjee, Bourreau, & Rančić, 2024) for Quadratic Unconstrained Binary Optimization (QUBO) problems. We used LogQ to tackle the mesh segmentation problem and also provided benchmarks against the classical KMeans clustering algorithm. In this deliverable, we present the software package to solve mesh segmentation using LogQ.

This report presents an overview of the files provided in the package and a short documentation on how to use them. There are two folders in the package: Log-Encoding and Mesh-Segmentation.

The folder Log-Encoding contains all the files required to run any QUBO problem using the LogQ encoding. On the other hand the Mesh-Segmentation folder contains a Jupyter notebook which solves the mesh segmentation problem. It imports files from the Log-Encoding.

Note that this deliverable does not contain any description of the problems themselves. This document is to be read in conjunction with the paper (Chatterjee, Bourreau, & Rančić, 2024) and the Deliverable 5.9 (Chatterjee, Bourreau, & Calandra, 2024).

2 Folder 1: Log-Encoding

The folder Log-Encoding contains the files required to solve the Maximum Cut problem and more generally a Quadratic Unconstrained Binary Optimization (QUBO) problem using the LogQ encoding.

Following is the description of the important files in the folder:

2.1 hamiltsolver_OWN_I_QSK_W.py

This is one of the most important files of the Log-Encoding folder. This file contains the function *hamiltsolver*.This function takes as input a Hamiltonian Matrix (which is the negative Laplacian Matrix in the case of the MaxCut problem) and returns the MaxCut partition of the problem. Following is the function description.

Input :

1. R : Hamiltonian Matrix
2. b : the backend - can be a quantum simulator or a quantum computer

Output :

1. partition : List of 0's and 1's depicting the 2 sets into which the nodes are divided
2. SS and SS1 : 2 subsets of nodes of the input graph which form the maximum cut.

While this code works directly for the MaxCut problem, it can be further edited to solve any QUBO problem. In order to solve a QUBO problem, input your QUBO as the Hamiltonian matrix and use your own interpretation of the partition vector to get the solution you require.

2.2 number_partitioning_OWN_I_QSK_W

This file solves the Number Partitioning problem by converting it to the MaxCut problem. The source code contains the variable *n* which can be changed to input the set of numbers given in the Number Partitioning problem.

The problem is then converted into the MaxCut problem before calling the file hamiltsolver_OWN_I_QSK_W.py to solve it. The code finally prints the best value of the objective function found.

2.3 indset_OWN_I_QSK_W.py

This file is actually a modified version of the file *hamiltsolver_OWN_I_QSK_W.py*. It solves the Maximum Weighted Independent Set Problem as a QUBO problem. This file contains the function *indsetsolver*. Following is its function description.

Input:

1. G: networkX node-weighted graph
2. b: the backend - can be a quantum simulator or a quantum computer
3. mult: parameter to regulate the strength of the penalty term. Higher values of mult reduce the penalty.
4. num_iter: maximum number of Genetic Algorithm generations.

Output:

1. objfn : the final value of the objective function
2. SS1: The Independent set (set of nodes) which give the MWIS
3. success: boolean to indicate whether the solution found is a feasible solution.

Firstly the problem is converted into a QUBO problem and then converted into an sQUBO matrix (Chatterjee, Bourreau, & Rančić, 2024). After that, we follow a procedure similar to the one used in *hamiltsolver_OWN_I_QSK_W.py*. Finally the partition vector is interpreted accordingly to extract the MWIS solution.

2.4 HermitiantoUnitary_OWN_I_QSK_H.py

This file converts the Hamiltonian into a sum of Pauli strings. This is a core file required for the running of all the previous files and therefore should be altered with caution.

3 Folder 2: Mesh-Segmentation

In this folder there is a Jupyter Notebook which carries out Mesh Segmentation using the LogQ algorithm. It imports the function *hamiltsolver* from the file *hamiltsolver_OWN_I_QSK_W.py*.

The notebook initially generates a 2D mesh of size *num_points* (cell 5). Then the mesh is converted into a graph (cell 6). The function *graphsolve* takes as input the number of segements required and the graph to be clustered.

Further the notebook also solves the clustering problem using the KMeans clustering algorithm in order to generate the benchmarks. Finally, several benchmarks such as cut discrepancy, consistency error and adjusted rand score are calculated by the respective functions.

List of Acronyms

Term	Definition
QAOA	Quantum Approximate Optimization Algorithm
QUBO	Quadratic Unconstrained Binary Optimization

***Table 1**: Acronyms and Abbreviations*

List of Figures

List of Tables

Bibliography

Chatterjee, Y., Bourreau, E., & Calandra, H. (2024). D5.9: Benchmarking of qaoa-based algorithms for mesh segmentation, against k-means, normalized and randomized cuts and core extraction methods. https://www.neasqc.eu/wp-content/uploads/2024/05/Deliverable_D5_9_V0.2_VF.pdf

Chatterjee, Y., Bourreau, E., & Rančić, M. J. (2024). Solving various np-hard problems using exponentially fewer qubits on a quantum computer. *Phys. Rev. A*, *109*, 052441. https://doi.org/10.1103/PhysRevA.109.052441