APPLICATION OF VARIATIONAL QUANTUM ALGORITHMS TO AUTONOMOUS GROUND VEHICLE MOBILITY

SAM COCHRAN[1], JEREMY MANGE[2], DAVID GORSICH[2], PARAMSOTHY JAYAKUMAR[2], JAMES STOKES[1], AND SHRAVAN VEERAPANENI[1]

Abstract. We examine one VQA in particular called the Quantum Approximate Optimization Algorithm (QAOA). We show how to map the problem of clustering a dataset onto a Max-Cut problem, and give an outline of how to solve Max-Cut using QAOA. We also introduce a method for improving the accuracy of QAOA by using the solution to a Max-Cut relaxation to warm-start the initial quantum state. We summarize several existing warm-starting approaches and compare their performance in simulated runs of QAOA. We also present some results for warm-started QAOA runs on existing quantum hardware.

1. Introduction

Variational quantum algorithms (VQAs) take advantage of the inherent parallelism in quantum devices to potentially accelerate these calculations, leading to faster and more efficient solutions. In VQAs, the core idea is to represent a computational problem as an optimization problem for an unknown quantum state. Since quantum states have a probabilistic nature, the solution to the computational problem is encoded within the output probabilities of the optimal quantum state. Monte Carlo sampling is employed to estimate these probabilities. The potential computational advantage of VQAs stems from the curse of dimensionality inherent in multipartite quantum states, which VQAs address by utilizing specialized Quantum Processing Units (QPUs) designed for state preparation. VQAs adopt a hybrid quantum-classical approach, where the central processing unit (CPU) performs gradient-based updates to learn the preparation of a quantum state tailored to the specific computational problem at hand. The interaction between the QPU and CPU in this manner holds the potential to deliver a quantum advantage, and variational algorithms have been developed with the objective of solving complex combinatorial optimization problems [10], preparing ground states quantum many-body Hamiltonians [23], solving linear systems of equations [2] and partial differential equations [1].

We focus our analysis on one VQA called the Quantum Approximate Optimization Algorithm (QAOA) [10]. QAOA can be used to find approximate solutions to a range of combinatorial optimization problems, with the Max-Cut problem being one example. We show the problem of clustering a set of datapoints can be mapped onto an equivalent Max-Cut problem, and we outline an implementation of QAOA for solving Max-Cut. For problems where QAOA struggles, its accuracy can be improved by using the solution to a Max-Cut relaxation to warm-start the initial quantum state. Several approaches for warm-starting QAOA have been proposed [9, 25] based on different Max-Cut relaxations. We will give a summary of these approaches, as well as a comparison of their performance on simulated QAOA runs. We will also present results from running warm-started QAOA on existing quantum computers. Our results indicate that warm-starting QAOA can significantly improve performance on difficult problems like Max-Cut. In most cases the warm-starting approach due to [25] seems to yield the best results.

The paper is organized as follows. First, we discuss hard computational problems in autonomous mobility that are expected to benefit from VQAs in Section 2, followed by a specific use case with available datasets in Section 3. In Section 4, we summarize the implementation of QAOA for solving Max-Cut problems. We introduce a general framework for warm-starting QAOA using Max-Cut relaxations, and then give an overview of a few existing methods based on different relaxations. In Section 5, we present QAOA results corresponding to runs on both simulators and existing quantum computers, and compare the performance of each warm-starting approach considered in Section 4.

[1]Department of Mathematics, University of Michigan, Ann Arbor, MI 48109 USA

[2]Ground Vehicle Systems Center, U.S. Army DEVCOM, Warren, MI

2. Hard computational problems in Autonomous mobility

We start by introducing several computationally difficult problems that arise in the area of autonomous mobility. We provide a description of the problems, a brief discussion of their relevance within the autonomous ground vehicle research context, and mathematical formulations highlighting their computational complexity.

2.1. **Path Planning with Static Obstacle Avoidance.**

A basic problem in autonomous mobility is that of path planning; specifically, computationally generating paths that avoid vehicle collisions with obstacles and are optimal according to some set of criteria. We begin with the simplest version of this problem wherein obstacles are static – instances of which can be solved by polynomial-time algorithms – before moving to the more computationally difficult version with moving obstacles.

2.1.1. *Autonomous Ground Vehicle Applications.* It is perhaps obvious how fundamental path planning with obstacle avoidance is for ground vehicle autonomy algorithms. One of the most basic high-level goals for unmanned ground vehicles (UGVs) is to move from one point to another safely and effectively, which inherently involves this type of path planning. Although a realistic autonomous vehicle context generally involves both static and moving obstacles, it is illustrative to first consider the simpler version of this problem, but also to note that the static-obstacle version also arises in real-world contexts. Numerous examples of this problem can be found both in ongoing Ground Vehicle Systems Center (GVSC) projects as well as in the literature; for example, [13], [17], [6].

2.1.2. *Discrete Variable Formulation.* Path planning with static obstacle avoidance is often formulated as a weighted graph shortest-path problem, with vertices representing valid vehicle locations, edges representing valid vehicle paths from one location to another, and edge weights representing the quantity to be minimized for the overall path (distance, fuel usage, etc.) Intuitively, the graph theory shortest path problem is fairly straight-forward: to find the shortest edge-weighted path in a given graph from a starting vertex to a target vertex.

The graph formulation of this problem is advantageous in that it is general enough to encompass both discretized Euclidean planar geometries (a common consideration within autonomous mobility, and the assumed context for the continuous variable formulation explored next) as well as other more complex types of situations such as terrains of varying heights, terrain characteristics, etc. which could be captured by the edge weights to accurately reflect the problem environment. The discrete variable problem, then, is to minimize:

$$\sum_{(u,v)\in E} w_{uv} \cdot x_{uv}$$

subject to:

$$\sum_{v\in V} x_{vs} - \sum_{w\in V} x_{sw} = 1, \sum_{v\in V} x_{vt} - \sum_{w\in V} x_{tw} = -1$$

$$\sum_{v\in V} x_{uv} - \sum_{w\in V} x_{vw} = 0, \forall v \neq s,t, \sum_{(u,v)\in E} x_{uv} \cdot o_u \cdot o_v = 0$$

In this formulation, $G = \{V, E\}$ is a directed, weighted graph with vertices V and edges E, s is the starting vertex, t is the target vertex, and $W = \{w_{uv}\} : (u, v) \in E$ are a set of edge weights. $O = \{o_v\} : v \in V$ is a set of binary variables indicating whether an obstacle occupies vertex v, and x_{uv} are binary variables for each edge $(u, v) \in E$ indicating whether the edge (u, v) is in the path. As mentioned previously, this is a well-known problem with several polynomial-time algorithms.

2.1.3. *Continuous Variable Formulation.* Alternatively, this problem can be formulated as an optimization problem using continuous variables with a plane. The underlying concept of the problem is the same as in the previous formulation, but with the source and targets defined as (x, y) points on the plane, and obstacles defined as polygons within that plane.

The continuous variable optimization version of this problem, then, is to minimize:

$$\frac{\sum_{i=1}^{n-1} \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}}{o \cdot \sqrt{n-1}},$$

subject to:

$$x_0 = s, x_n = t$$

In this formulation, $s = (s_x, s_y)$ is the starting point, $t = (t_x, t_y)$ is the target point, and n is a maximum number of points for the path. o is the set of polygonal obstacles which are defined by their exterior (x, y) points within the plane, and thus the optimization goal is to find a path with at most $n - 2$ intermediate points of minimum total length that does not intersect any obstacle. The definition of o is straightforward for the version of this problem involving static obstacles; however, this same continuous variable formulation can be used for the moving obstacle case discussed in the following section with a time-dependent definition of o.

2.2. **Path Planning with Moving Obstacles.** A natural extension of path planning with static obstacle avoidance is the introduction of moving obstacles. This is a more realistic type of problem for autonomous vehicle mobility, and the Department of Defense has identified this problem as a fundamental problem of interest for both naval unmanned surface vehicles (USVs) and unmanned ground vehicles (UGVs) [18].

Path planning with moving obstacles is a problem that has been widely studied, and a number of heuristic algorithms have proved successful for finding solutions to problem instances within certain contexts; however, the general problem has been proven to be NP-hard, by Canny and Reif [5] among others, through a reduction from the well-known Boolean satisfiability problem 3-SAT.

2.2.1. *Autonomous Ground Vehicle Applications.* As mentioned with reference to the previous problem, path planning is a fundamental task within ground vehicle autonomous mobility. The addition of moving obstacles makes the problem much more computationally difficult, but also maps more closely to the challenges of UGV mobility in real-world settings. Within GVSC, this problem has been studied and applied with several contexts, including dynamic trajectory planning with short time horizons [12] and path planning in the presence of moving obstacles focusing on vehicle dynamics on vehicle-terrain interaction on deformable terrains [7]. Applications of this same problem involving autonomous robots are also common in the literature; for example [27].

2.2.2. *Discrete Variable Formulation.* Extending the formulation from the static version of the problem, the moving-obstacle problem is defined as minimizing:

$$\sum_{(u,v)\in E} w_{uv} \cdot x_{uv}$$

subject to:

$$\sum_{v\in V} x_{vs} - \sum_{w\in V} x_{sw} = 1, \sum_{v\in V} x_{tv} = 0, \sum_{v\in V} x_{vt} = 1,$$

$$\forall v \neq s, t : \sum_{v\in V} x_{vu} - \sum_{w\in V} x_{wv} \in \{0, 1\}, \sum_{(u,v)\in E} x_{uv} \cdot o_{x_{uv}u} \cdot o_{x_{uv}v} = 0$$

The variables are defined as previously, with the following extensions: T represents the time horizon for the problem, $O = \{o_{vt}\}$ is the set of obstacles that now includes a time component, wherein o_{vt} is 1 if an obstacle occupies vertex v at time t, and 0 otherwise, and the indicator variable x_{uv} is no longer binary, but now has a value of n if the edge (u, v) is the nth edge on the path, and 0 if it is not in the path.

Finding a minimum-weight path that satisfies the given constraints is equivalent to finding a shortest-path for the underlying path planning problem with moving obstacles.

2.2.3. *Continuous Variable Optimization Formulation.* As mentioned in the continuous variable formulation for the static obstacle version of the problem, the same optimization formulation can be used with moving obstacles, since that formulation is generalized to specify obstacle intersection as a separate calculation. Thus the time-based obstacle definitions for the moving obstacles problem can be incorporated into that calculation based on the speed of the vehicle and the path under consideration, and the optimization formulation remains the same, although the problem itself is more computationally challenging, with the problem of finding an exact solution being NP-hard.

2.3. **Resource Assignment.** The general resource assignment (or "generalized assignment") problem is that of assigning a set of agents to a set of tasks in a way that is optimal according to some set of criteria. The equivalent problem in graph-theoretic terms is that of finding a minimum-weight matching of a given size for a weighted bipartite graph. This problem has a number of real-world applications and has been widely studied in various contexts, and has been proven to be NP-hard [20], [22].

2.3.1. *Autonomous Ground Vehicle Applications.* Within autonomous vehicle mobility, the resource assignment problem can be applied to assigning a set of vehicles to a set of tasks, targets, etc. This type of problem has a number of natural applications, such as assignment of a set of autonomous tasks to a fleet, assignment of an autonomous fleet to generate sensor readings over an area, assignment of an autonomous combat fleet to a set of enemy targets, assignment of a fleet of autonomous taxis to available passengers, and so forth. We describe two such examples briefly.

A fairly general resource assignment problem for autonomous ground vehicles is that of using a fleet of autonomous vehicles to transport material within an environment; that is, to accomplish a set of tasks consisting of picking up material at one location and dropping it off at a different location. Kulatunga et. al. [16] outline one such application, in which the environment is represented by a series of nodes with valid pathways defined such that vehicle congestion becomes a concern during the defined tasks. In this instance, the objective function to be minimized is the total schedule time, which is the total time required for the vehicles to carry out their assigned tasks.

A commonly proposed use for autonomous ground vehicles within the robotics domain, of particular interest for the Army and other military communities, is automated patrolling, in which an area is defined over which a group of robotic vehicles must be assigned in order to cover the important parts of the area within defined constraints. Farinelli et. al. [11] describe a series of simulated experiments utilizing this type of problem formulation, in which a set of maps is defined, and a group of robots must be assigned patrol tasks in order to cover the significant sections of the map area.

2.3.2. *Optimization Formulation.* For this problem formulation, we utilize the concept of a target, which itself is associated with one or more tasks to which vehicles can be assigned, similar to the formulation in [24]. The resource assignment problem, then, is that of assigning vehicles to tasks such that all tasks are performed for all targets, while minimizing an objective function of interest. For this formulation, we will use as our objective function the total distance traveled by all vehicles, so that the goal is to assign tasks in such a way as to minimize total vehicle travel distance. The problem is to minimize:

$$\sum_{i=1}^{N_V}\sum_{j=1}^{N_T}\sum_{k=1}^{N_T} d_{ijk}^{Z_{k-1}} \cdot z_{ijk}$$

subject to:

$$\forall k : \sum_{i=1}^{N_V}\sum_{j=1}^{N_T} z_{ijk} = 1, \forall j : \sum_{i=1}^{N_U}\sum_{k=1}^{N_A} z_{ijk} = N_S$$

In this formulation, N_V is the number of vehicles, N_T is the number of targets, N_S is the number of tasks per target, and N_A is the number of task assignments. z_{ijk} is a binary variable indicating whether vehicle i performs a task on target j in the kth assignment, $Z_k = \{z_{ij1}, z_{ij2}, ..., z_{ijk}\}$ is the set of assignments through assignment k, and Z_{N_A} is the set of all assignments. $d_{ijk}^{Z_{k-1}}$ is the "incremental" distance traveled by vehicle i in order to perform a task on target j as the kth assignment preceded by the assignments Z_{k-1}.

Finding an assignment vector Z_{N_A} that satisfies each constraint is then equivalent to finding an assignment of vehicles to tasks such that all tasks are accomplished with the lowest total distance traveled by all vehicles.

2.4. **Clustering.** The clustering problem is the problem of dividing a set of data points into "clusters" that are optimal according to some set of criteria. This is a very general problem, and thus the criteria defining a "good" cluster are highly context-dependent, with many different measures used within different applications. One such measure is the distance of data points in two clusters from one another, which translates to the well-known NP-complete Max-Cut problem. Thus this problem has two features of particular interest: it is general enough to be important for many real-world applications (including autonomous mobility), and

versions of it are computationally challenging enough to provide motivation for investigating the promise of performance gains from quantum computing approaches.

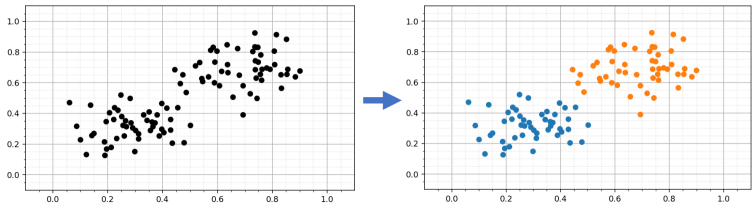

Figure 1. Simple Data Clustering.

2.4.1. *Autonomous Ground Vehicle Applications.* One of the fundamental tasks of autonomous vehicle subsystems that interpret sensor data is to classify measurements into meaningful identifiers; a classic example is object recognition using autonomous sensor systems. Many such cases involved unsupervised learning; that is, machine learning performed without access to labeled data. Often the key component of this type of learning is a form of clustering – partitioning the data into similar clusters which can then be used for classification and feature extraction. The clustering problem is a low-level, fundamental task which enables a number of higher-level autonomous behaviors [19], [8].

2.4.2. *Problem Formulation.* The clustering problem can be generalized in order to find an arbitrary number of clusters. However, since even the ideal 2 cluster problem and associated Max-Cut problem are NP-Hard, we define it here with two clusters. Given a set of k-dimensional points $P = \{x_1 = (x_{11}, x_{12}, ..., x_{1n}), x_2, ..., x_n\}$, and a distance function d, our goal is then to find clusters C_1 and C_2 that minimize:

$$\sum_{c_1, c_1' \in C_1} d(c_1, c_1') + \sum_{c_2, c_2' \in C_2} d(c_2, c_2'), \tag{1}$$

where c_1, c_1' are pairs of points in C_1, and c_2, c_2' are pairs of points in C_2. We can observe that minimizing the distance between points within each of two clusters is equivalent to maximizing the distance between points in opposite clusters, so minimizing equation (1) is equivalent to maximizing

$$\sum_{c_1 \in C_1, c_2 \in C_2} d(c_1, c_2). \tag{2}$$

The Quantum Approximate Optimization Algorithm (QAOA) is a quantum algorithm that can be used to approximately solve NP-Hard combinatorial optimization problems, with Max-Cut being one example. In order to use QAOA to solve the clustering problem, we will map clustering onto an equivalent Max-Cut problem. We start by representing each data point in our set P as a node in a weighted undirected graph $G = (V, E)$. The weight w_{jk} of the edge between any two nodes $j, k \in V$ is given by the Euclidean distance

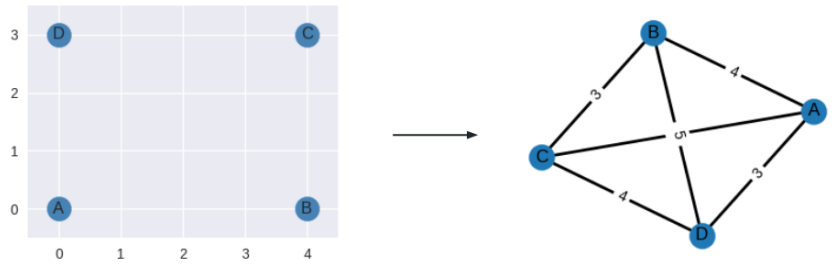

Figure 2. Mapping a simple four-point clustering problem onto a graph with an equivalent max-cut problem.

between the corresponding points in our dataset. Figure 2 shows an example of this mapping for a simple case with just four data points. Max-Cut is then defined as finding a bipartition x for the graph G that maximizes the total weight of edges between the two bipartite sets:

$$\begin{aligned} &\text{maximize} \quad \sum_{(j,k)\in E} \frac{1}{2} w_{jk}(1 - x_j x_k) \\ &\text{subject to} \\ &x_j \in \{-1, 1\} \text{ for } j = 1, ..., n. \end{aligned} \tag{3}$$

We note that (3) is equivalent to choosing clusters C_1 and C_2 to maximize (2), and so the two-cluster problem on our dataset is equivalent to this Max-Cut problem on G.

3. Clustering Problem for Autonomous Vehicle Sensor Data

The final problem outlined in the previous section, the clustering problem, is of particular interest within the autonomous vehicle research context. Additionally, since the clustering problem can be formulated as a Max-Cut problem, it is both computationally challenging (indeed NP-hard) in a way that motivates the use of quantum computing approaches, and also particularly amenable to the use of quantum algorithms, in that the Max-Cut problem has been widely studied in quantum computing literature.

In order to apply the quantum algorithm of interest, the Quantum Approximate Optimization Algorithm (QAOA), to instances of the clustering problem for autonomous vehicle sensor data, we obtained instances of this sensor data using publicly available research datasets. A number of sources for this type of data are available, several of which we describe here. We utilized the RELLIS-3D dataset for several of our QAOA experiments, but several of the datasets described would have been appropriate for the same purpose.

3.1. Data Sources.

Although the datasets in this section are distinct and targeted for different purposes, a brief summary table is provided below to provide an overview of their relative sizes and sensor types.

Dataset Name	Source	# Camera Images	# LIDAR scans	# Semantic Classes
KITTI	Karlsruhe Institute of Technology / Toyota	15,000	15,000	30
nuScenes	Motional	1,400,000	390,000	32
RUGD	Army Research Laboratory	37,000	-	24
RELLIS-3D	Army Research Laboratory / Texas A&M University	12,470	27,112	20
YCOR	Yamaha / Carnegie Mellon University	1,076	-	8

3.2. KITTI.

The KITTI dataset [14] was created by the Karlsruhe Institute of Technology in conjunction with Toyota, and consists of vehicle sensor data from urban and rural on-road driving. The data in this dataset was captured from a Volkswagon Passat B6 equipped with two color cameras, two grayscale cameras, a Velodyne LIDAR scanner, and a GPS system.

Part of the motivation for the creation of this dataset was to provide data for objection detection, and for this purpose 15,000 "frames" are provided, split into training and testing datasets, with ground truth object detection data provided. For each of these frames, a color camera image is provided, a LIDAR point cloud, and a bird's eye view image. This data is useful for 2D or 3D object detection tasks.

In addition, KITTI consists of data targeted at several other computer vision tasks, including semantic segmentation. This smaller dataset consists of 400 camera image frames, with semantic labels consisting of 30 object classes.

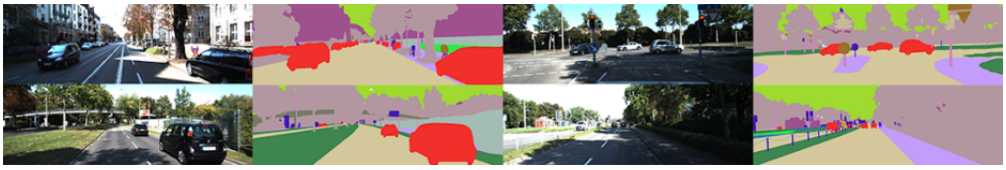

Figure 3. KITTI Semantic Segmentation Data.

The KITTI website (https://www.cvlibs.net/datasets/kitti/) also provides both evaluation criteria and a method to submit results from the several computer vision and data processing tasks for which the datasets are provided, and to compare evaluated results between submissions. This has proved valuable for pushing forward the development of algorithms and methods for these tasks, as well as providing a central location to see current results for standard benchmark data.

3.3. **nuScenes.** The nuScenes dataset [4] was created by Motional, and was inspired by the KITTI dataset. The data includes 1,000 "scenes", which consist of on-road sensor data collected in Boston and Singapore from Renault Zoe cars equipped with 6 color cameras, a Velodyne LIDAR scanner, 5 RADAR scanners, and a GPS system.

This dataset is focused on "challenging" driving situations, including unexpected behaviors from vehicles and pedestrians. The dataset is very large, including 1.4 million camera images, 390,000 LIDAR scans, 1.4 million RADAR scans, and several ground-truth bounding boxes and object annotations.

For 3D object detection, nuScenes includes bounding boxes in 40,000 keyframes (see figure 4). In addition, nuScenes released a separate dataset for 3D semantic segmentation using LIDAR. This dataset consists of 1.4 billion annotated points over 40,000 LIDAR point clouds. These annotations provide ground truth labels for 32 object classes, including 23 foreground classes and 9 background classes.

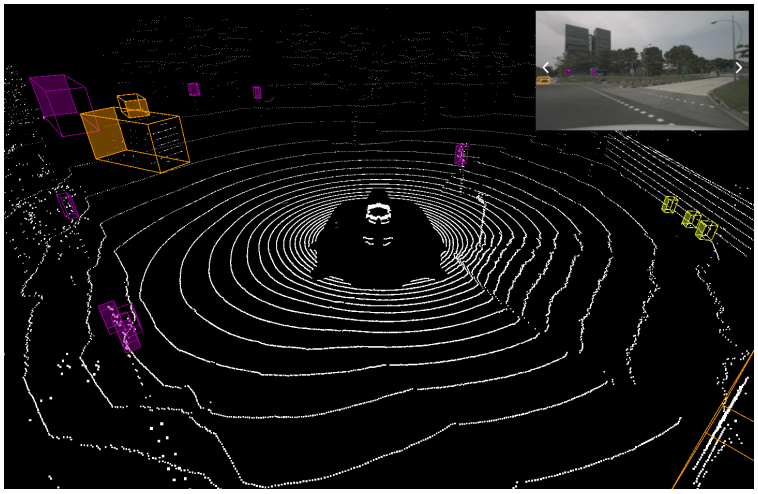

Figure 4. nuScenes LIDAR Scans with Bounding Boxes.

3.4. **RUGD.** The Robot Unstructured Ground Driving (RUGD) dataset was created at the Army Research Laboratory (ARL) for the express purpose of exploring off-road autonomous navigation tasks [26]. As the name indicates, the sensor data in this dataset was collected from a robot, in this case an unmanned ground robot performing exploration in off-road unstructured environments.

This dataset is composed of 18 scenes, with full video sequences for each scene along with semantic labeled ground-truth information for every fifth frame in each sequence, for a total of 37,000 images, with

7,453 labeled images (see figure 5). The labeling for these images consists of 24 categories, as seen in the example image.

The RUGD dataset was created in order to provide researchers a benchmark dataset with certain unique characteristics, which the authors summarize as follows [26]:

(1) The majority of scenes contain no discernible geometric edges or vanishing points, and semantic boundaries are highly irregular.
(2) Robot exploration traversal creates irregular routes that do not adhere to a single terrain type; eight distinct terrains are traversed.
(3) Unique frame viewpoints are encountered due to sloped terrains and vegetation occlusion.
(4) Off-road traversal of rough terrain results in challenging frame focus.

This dataset, then, provides a valuable source of robotic camera images which can be used for autonomous learning tasks, within both robotics and vehicle applications.

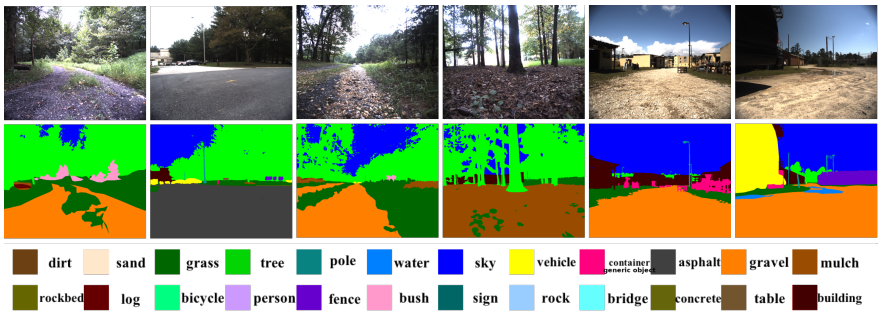

Figure 5. RUGD Images for Semantic Segmentation.

3.5. **RELLIS-3D.** Expanding on the work from the RUGD dataset, the RELLIS-3D dataset was created by the Army Research Laboratory in conjunction with Texas A&M University. RELLIS-3D is a "multi-modal" off-road dataset, which was collected from a Clearpath Robotics Warthog vehicle, equipped with mono and 3D stereo cameras, LIDAR, and a GPS system.

One of the main purposes of the RELLIS-3D dataset was to provide a synchronized multi-modal dataset with ground-truth semantic annotations, and as such this dataset provides more sensor types and additional terrain features not present in the RUGD dataset. The data consists of 5 off-road traversal scenes across a variety of terrain types, comprising a total of 12,470 images and 27,112 LIDAR scans, of which half are labelled with ground-truth semantic annotations. These annotations are made up of 20 classes, consistent with the RUGD semantic classes (see figure 6).

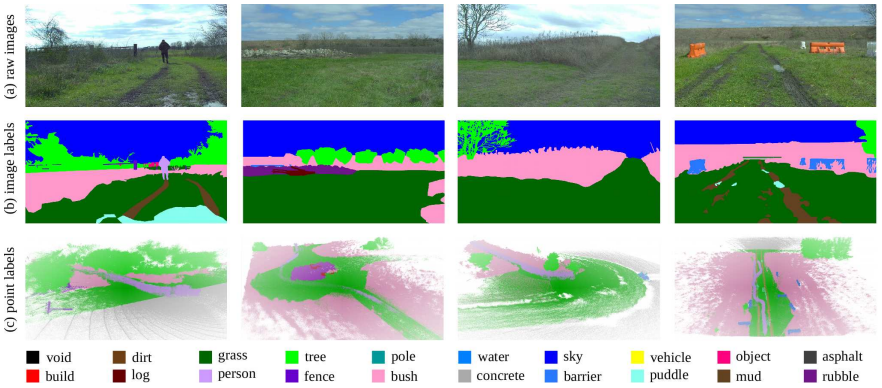

Figure 6. RELLIS-3D Images for Semantic Segmentation.

3.6. **YCOR.** The Yamaha-CMU-Off-Road (YCOR) dataset was created in collaboration between Yamaha and Carnegie Mellon University. The data for this dataset was collecting using an autonomous All-Terrain Vehicle (ATV), equipped with a 64-line Velodyne LIDAR scanner and a Carnegie Robotics stereo camera.

This is a smaller and more focused dataset than the others described here, and was primarily created in order to explore real-time semantic mapping using convolutional neural networks. The data consists of 1076 images, split into training and validation sets. These images are labeled with semantic data using 8 classes (see figure 7).

This dataset is of interest in demonstrating a real-time semantic segmentation approach using both LIDAR and camera data in order to generate semantic maps, as well as to test similar approaches using baseline data with ground-truth semantic labels.

Figure 7. YCOR Images for Semantic Segmentation.

4. QAOA

We now describe how to implement QAOA and summarize approaches for improving its performance using warm-starting.

4.1. **Standard QAOA.** A system of n qubits has 2^n standard basis states $|x\rangle$, each corresponding to a unique length-n bitstring $x \in \{-1, 1\}^n$. Each bitstring corresponds to a cut for our graph. An arbitrary state $|\psi\rangle$ of the system can be written as a superposition over these basis states:

$$|\psi\rangle = \sum_{x=1}^{2^n} \beta_x |x\rangle, \tag{4}$$

where $\sum_{x=1}^{2^n} |\beta_x|^2 = 1$. Upon measurement, we will observe the standard basis state $|x\rangle$ with probability $|\beta_x|^2$.

To initialize QAOA, we prepare a separable state by placing each individual qubit in the equal superposition state:

$$|\psi_0\rangle = \bigotimes_{i=1}^{n} |+\rangle. \tag{5}$$

In this initialization, every possible bitstring would be measured with equal probability. The goal of QAOA is to evolve the state such that upon measurement, bitstrings corresponding to high-value cuts are sampled with high probability.

In order to do this, we define a cost Hamiltonian H_C whose ground space is spanned by bitstrings corresponding to maximum cuts for our graph

$$H_C = \sum_{(j,k)\in E} \frac{1}{2} w_{jk}(1 - Z_j Z_k). \tag{6}$$

We also introduce a mixer Hamiltonian H_M such that the ground state of H_M is our initial state $|+\rangle$

$$H_M = -\sum_{i=1}^{n} X_i. \tag{7}$$

An important operation needed to implement QAOA is time evolution. For a given Hamiltonian H and time $t \in \mathbb{R}$, the time evolution operator is defined as

$$U(H, t) = \exp(-itH). \tag{8}$$

The quantum circuit for QAOA applies $p \geq 1$ layers of alternating time evolution under H_C and H_M. This circuit is parameterized by vectors $\alpha, \gamma \in \mathbb{R}^p$, which give the evolution times for each layer:

$$C_p(\alpha, \gamma) = U(H_M, \alpha_p)U(H_C, \gamma_p) \cdots U(H_M, \alpha_1)U(H_C, \gamma_1). \tag{9}$$

The objective function for QAOA is the expectation value of the cost Hamiltonian after applying the QAOA circuit:

$$f_p(\alpha, \gamma) = \langle \psi_0 | C_p^\dagger(\alpha, \gamma) H_C C_p(\alpha, \gamma) | \psi_0 \rangle. \tag{10}$$

For a system of n qubits, the full quantum state is described by a vector of 2^n complex-valued coefficients. The quantum circuit manipulates this entire statevector, but is limited in the operations it can perform because it is parameterized by just $2p$ variables. Approximately minimizing (10) is thus a reasonable task, provided we have a way to approximate the expectation value. This task is well suited to a quantum computer.

The objective (10) can be approximately minimized by finding circuit parameters α and γ such that $C_p(\alpha, \gamma)|\psi_0\rangle$ is near the ground state of H_C. In the limit as $p \to \infty$, it can be shown that the minimization becomes exact, as the circuit follows the adiabatic evolution of the state [10]. In practice, however, constraints such as noise and short coherence times limit available gate depths to modest values, so we construct a shallow circuit (choosing p to be relatively small) and use stochastic optimization techniques to choose the circuit parameters.

Measurement of a quantum circuit is inherently random, so information about the state at the end of the circuit must be extracted by repeatedly applying the circuit, measuring the output, and considering the distribution of measured values. Each of these evaluations of the circuit is called a shot. For a given choice of time parameters α and γ, running and measuring the circuit for m shots will give us m basis states $|x_1\rangle, \ldots, |x_m\rangle$. These measured states can be used to estimate (10):

$$f_p(\alpha, \gamma) \approx \hat{f}_p(\alpha, \gamma) = \frac{1}{m} \sum_{i=1}^{m} \langle x_i | H_C | x_i \rangle. \tag{11}$$

A variety of techniques have been developed for optimizing the parameters of a quantum circuit, including both gradient-based and gradient-free methods. We will use gradient-free methods here. A common choice for QAOA is constrained optimization by linear approximation (COBYLA). Using this optimizer, we can find circuit parameters that minimize (11). The circuit is then evaluated for these parameters over a final batch of shots, and the measured basis state corresponding to the highest value cut is taken as the solution.

The optimal circuit parameters can be understood mapping a path through parameter-space from our initial equal superposition state to a state close to the ground state of the cost Hamiltonian H_C. In situations where QAOA struggles (such as Max-Cut for large, fully-connected graphs), we can improve performance by shortening this path using a procedure called warm-starting.

4.2. **Warm-Starting QAOA.** The choice to initialize the state as an equal superposition means that QAOA begins with no information; each state is initially just as likely to be sampled as every other state. We now consider the case where we have some knowledge about what the solution might look like. Define a warm-started initial state by applying to each qubit i a Y-rotation by an amount $\theta_i \in [0, 2\pi)$, starting from the $|0\rangle$ state on the Bloch sphere:

$$|\psi_\theta\rangle = \bigotimes_{i=1}^{n} R_Y(\theta_i)|0\rangle. \tag{12}$$

Setting $\theta_i = \pi/2$ will initialize the i^{th} qubit in the $|+\rangle$ superposition state as before, while choosing θ_i to be 0 or π will initialize the i^{th} qubit in the deterministic $|0\rangle$ or $|1\rangle$ state, respectively. A visualization for a single qubit is shown in Figure 8. The idea of warm-starting is to carry out some inexpensive precomputations to inform the choice of initial state. Recall that the exact solution to the Max-Cut problem is a bitstring corresponding to a ground state of the cost Hamiltonian H_C. Given an approximate Max-Cut solution, the warm-starting angles $\theta_1, ..., \theta_n$ can be chosen to rotate the initial state of each qubit towards either $|0\rangle$ or $|1\rangle$, so that the quantum state is initialized closer to the desired ground state of H_C.

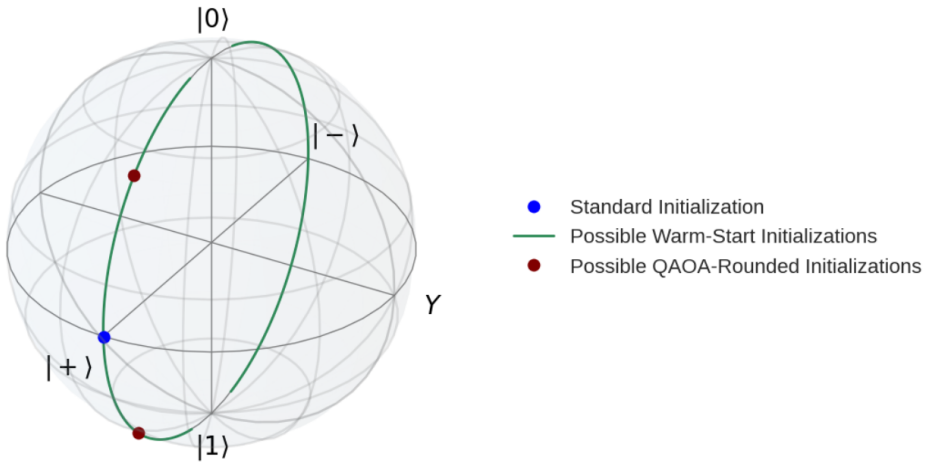

Figure 8. Bloch sphere for a single qubit. The position corresponding to the equal superposition state $|+\rangle$ is plotted in blue. All possible warm-start initial states $|\psi_\theta\rangle$, $\theta \in [0, 2\pi]$ lie on the green curve. This initialization is defined in (12). Note the gaps in the green curve at each pole corresponding to regularization as described in (17). The locations of the rounded warm-start initial states are plotted in red.

Once the warm-starting angles are chosen, we construct a warm-started mixer Hamiltonian H_M^θ. Recall that the initial state for standard QAOA given in (5) is a ground state of the original mixer (7). This property is necessary to argue that the QAOA solution converges to a maximum cut as the number of layers increases. The warm-started mixer H_M^θ is thus constructed so that the warm-started initial state (12) is a ground state

$$H_M^\theta = -\sum_{i=1}^{n} \big(\sin(\theta_i)X_i + \cos(\theta_i)Z_i\big). \tag{13}$$

Note that in the case where each warm-starting angle is chosen to be $\pi/2$, (13) is the same as the original QAOA mixer defined in (7).

Time evolution under the mixer (13) corresponds to rotating each qubit i about the axis $\vec{n} = -(\sin(\theta_i), 0, \cos(\theta_i))$ on the Bloch sphere, and it can be shown [9] that the time evolution operator can be implemented as

$$U(H_M^\theta, t) = \bigotimes_{i=1}^{n} R_Y(\theta_i)R_Z(-2t)R_Y(-\theta_i). \tag{14}$$

The warm-started QAOA circuit is then defined as

$$C_p^{\text{ws}}(\alpha, \gamma) = U(H_M^\theta, \alpha_p)U(H_C, \gamma_p)\cdots U(H_M^\theta, \alpha_1)U(H_C, \gamma_1), \tag{15}$$

and the warm-started QAOA objective function is given by

$$f_p^{ws}(\alpha, \gamma) = \langle\psi_\theta|(C_p^{ws})^\dagger(\alpha, \gamma)H_C C_p^{ws}(\alpha, \gamma)|\psi_\theta\rangle. \tag{16}$$

The objective (16) can be optimized using the methods detailed in the previous section.

There is a reachability issue that we must address in the case that any of the warm-starting angles in (12) are exactly 0 or π. In this case, the corresponding qubits are initialized exactly at the poles of the Bloch sphere, and the warm-started QAOA circuit (15) will never be able to move them away from this initial state. To remedy this, we introduce a regularization parameter $0 < \epsilon < \pi/2$, and set

$$\theta_i = \begin{cases} \epsilon, & |\theta_i| < \epsilon \\ \pi + \epsilon, & |\theta_i - \pi| < \epsilon \\ \theta_i & \text{otherwise} \end{cases} \tag{17}$$

for each $i \in \{1, ..., n\}$. This corresponds to forcing each qubit to be initialized outside of a spherical dome at either pole (see Figure 8).

4.2.1. *Goemans-Williamson.* We now examine a few approaches for choosing the warm-starting angles θ. An approach which has gained traction within the combinatorial optimization literature is based on the concept of relaxations. Consider the rank-d relaxation of Max-Cut, where each scalar variable $x_j \in \{-1, 1\}$ in the objective (3) is replaced with a vector $v_j \in \mathbb{R}^d$

$$\begin{aligned} &\underset{v_1, ..., v_n \in \mathbb{R}^d}{\text{maximize}} && \sum_{(j,k)\in E} \frac{1}{2} w_{jk}(1 - \langle v_j, v_k \rangle) \\ &\text{subject to} \\ &\|v_j\| = 1 \text{ for } j = 1, ..., n. \end{aligned} \tag{18}$$

Note that this optimization problem reduces to Max-Cut in the case where $d = 1$. In the case that the rank d is equal to n, the optimization problem (18) becomes convex [21]; this case is referred to as semidefinite programming (SDP). After obtaining a solution to the SDP problem, Goemans-Williams hyperplane rounding (GW) can be used [15] to generate a cut $x^* \in \{-1, 1\}^n$ that is at least 87.8% of the value of the maximum cut (assuming a graph with non-negative edge weights).

Ref. [9] proposed a warm-starting approach in which the GW cut is used to define the initial state $|\psi_\theta\rangle$ given in (12) according to the rule

$$\theta_i = \begin{cases} \pi/4, & x_i^* = -1 \\ 3\pi/4, & x_i^* = 1. \end{cases} \tag{19}$$

Each qubit is thus initialized its own Bloch sphere in one of the two positions plotted in red in Figure 8. We refer to this warm-starting approach as QAOA-Rounded (GW).

Because the GW cut is guaranteed to give a good approximate solution, it is desirable that the circuit be able to easily evolve the state from the warm-started initialization to the basis state corresponding to the GW cut. It is unclear how to do this using the circuit defined in (15). Ref. [9] propose modifying the mixer layer given in (14) by multiplying the warm-starting angles θ by -1. The resulting mixer layer is given by

$$U(H_M^{-\theta}, t) = \sum_{i=1}^{n} R_Y(-\theta_i) R_Z(-2t) R_Y(\theta_i). \tag{20}$$

Consider the application of the first layer of QAOA to the warm-started initial state (12). Choosing the parameters to be $\alpha_1 = \pi/2$ and $\gamma_1 = 0$, we obtain

$$U(H_C, \gamma_1 = 0) U(H_M^{-\theta}, \alpha_1 = \pi/2) |\psi_\theta\rangle = (-i)^n \bigotimes_{i=1}^{n} |-x_i^*\rangle. \tag{21}$$

Measurement of this state in the computational basis will yield a bitstring corresponding to the same cut as the GW solution. The mixer layer given in (20) therefore results in a QAOA circuit that is at least as good as GW after optimization. It is important to note, however, that the warm-started initial state (12) is not generally a ground state of the mixer $H_M^{-\theta}$, so the heuristic argument of convergence to a maximum cut as the number of circuit layers increases can no longer be applied.

4.2.2. *Rank-2 Relaxation.* GW is not the only relaxation of Max-Cut that has been considered. Ref. [3] proposed using the rank-2 relaxation given by setting the rank $d = 2$ in (18). Switching to polar coordinates and dropping constant terms, we can rewrite this relaxation as an unconstrained optimization problem

$$\underset{\phi \in [0, 2\pi)^n}{\text{maximize}} \quad \sum_{(j,k)\in E} -\frac{1}{2} w_{jk} \cos(\phi_j - \phi_k). \tag{22}$$

We henceforth refer to this optimization problem as BMZ. Similar to the way GW Rounding was used to generate an approximate cut from the SDP solution, Ref. [3] give a rounding algorithm (Procedure-CUT) for generating an approximate cut $x^* \in \{-1, 1\}^n$ from the optimized BMZ angles $\phi^* \in [0, 2\pi)^n$. The computational complexity of Procedure-CUT scales much better than that of GW, and it also achieves higher accuracy empirically. It is important to note, however, that unlike GW, Procedure-CUT does not have any theoretical accuracy guarantee.

We can use BMZ to define another version of rounded warm-starting by following the procedure from the previous section, replacing the GW approximate cut with one generated with Procedure-CUT. We refer to this approach as QAOA-Rounded (BMZ). There is, however, a more natural approach due to [25] that maps

the optimized BMZ angles directly onto a quantum state. We refer to this approach as QAOA-Warmest (BMZ). To implement QAOA-Warmest (BMZ), the optimal BMZ angles $\phi^* \in [0, 2\pi)^n$ are used to construct the warm-started initial state $|\psi_{\phi^*}\rangle$ using the definition in (12). The BMZ angles are also used to construct the warm-started mixer layer $U(H_M^{\phi^*}, t)$ from (14). Note that whereas each QAOA-Rounded approach only initializes each qubit to one of two states, QAOA-Warmest allows each qubit to be initialized almost anywhere in the XZ-plane of the Bloch sphere (see Figure 8).

Although applying a uniform rotation to each entry of θ will not change the BMZ objective (22), rounding after applying different uniform rotations can result in different quality cuts. In fact, the Procedure-CUT algorithm gives a way to find the best uniform rotation to apply before rounding to get an approximate maximum cut. For our implementation of QAOA-Warmest, we select an index i at random and subtract ϕ_i^* from each entry of ϕ^*. This corresponds to choosing a qubit at random to initialize in the $|0\rangle$ state, and in practice tends to initialize most qubits near the poles of the Bloch sphere. While we could use Procedure-CUT to carefully choose the best possible uniform rotation, we found that in the examples we tested on, doing so yielded no significant improvement over this randomized approach.

Algorithm 1 QAOA-Warmest (BMZ)

1: Solve (22) to find the optimal angles $\phi^* \in [0, 2\pi)^n$
2: Choose i uniformly from $\{1, 2, ..., n\}$
3: Update $\phi_j^* = |(\phi_j^* - \phi_i^*)|$ for all $j \in \{1, 2, ..., n\}$
4: Define the initial state $|\psi_{\phi^*}\rangle = \bigotimes_{i=1}^n R_Y(\phi_i^*)|0\rangle_n$
5: Set up the mixer layer $U(H_M^{\phi^*}, \alpha_m) = \bigotimes_{i=1}^n R_Y(\phi_i^*) R_Z(-2\alpha_m) R_Y(-\phi_i^*)$
6: Set up the cost layer $U(H_C, \gamma_m) = \exp\Big(-i\gamma_m \sum_{(j,k)\in E} \frac{1}{2} w_{jk}(1 - Z_j Z_k)\Big)$
7: Choose a number of layers p
8: Initialize the entries of $\alpha, \gamma \in \mathbb{R}^p$ to be small random numbers
9: Construct the QAOA circuit as $C_p^{ws}(\alpha, \gamma) = U(H_M^{\phi^*}, \alpha_p) U(H_C, \gamma_p) \cdots U(H_M^{\phi^*}, \alpha_1) U(H_C, \gamma_1)$
10: Define a function $f_p^{ws}(\alpha, \gamma) = \langle \psi_{\phi^*}| (C_p^{ws})^\dagger(\alpha, \gamma) H_C C_p^{ws}(\alpha, \gamma) |\psi_{\phi^*}\rangle$
11: Estimate $f_p^{ws}(\alpha, \gamma)$ using $\hat{f}_p^{ws}(\alpha, \gamma) = \frac{1}{m} \sum_{i=1}^m \langle x_i | H_C | x_i \rangle$, where each $|x_i\rangle$ is a measured basis state obtained after applying the circuit $C_p^{ws}(\alpha, \gamma)$ to the initial state $|\psi_{\phi^*}\rangle$
12: Use COBYLA to find parameters $\alpha^*, \gamma^* \in \mathbb{R}^p$ to approximately minimize $\hat{f}_p^{ws}$
13: Run the circuit with the optimal parameters $C_p^{ws}(\alpha^*, \gamma^*)$ for a final batch of shots
14: The measured bitstring that gives the highest cut value $|x^*\rangle$ is taken as the QAOA-Warmest solution

5. Results

5.1. **Simulated results.** We now discuss experimental results comparing simulated runs of standard QAOA, QAOA-Rounded (GW), QAOA-Rounded (BMZ), and QAOA-Warmest (BMZ). We expect that any of the warm-started methods will improve over standard QAOA, and note that any difference in performance between QAOA-Rounded (GW) and QAOA-Rounded (BMZ) can be chalked up to BMZ giving a more accurate result than GW. Any improvement resulting from using QAOA-Warmest (BMZ) instead of QAOA-Rounded (BMZ) would suggest that keeping the BMZ angles rather than rounding provides an algorithmic advantage.

5.1.1. *Measured distributions.* For our first experiment, we randomly generated 30 graphs, each fully-connected with 20 nodes. For each edge, we randomly drew a weight from $\{1, 2, ..., 10\}$. Then, for each of the 30 random graphs, we did one run of each version of warm-started QAOA with 512 shots and compared both the best measured cut (the actual output of each algorithm) and the estimated circuit expectation for each method. Figure 9 shows the results, sorted along the x-axis by cut size, with values on the right-hand side corresponding to better cuts.

Figure 9a, shows the distribution of best sampled cuts across the 30 random graphs. QAOA-Warmest (BMZ) (plotted in red) performs the best with more exact maximum cuts than any other method, and with distributions of best sampled cuts tightly clustered near the maximum cut. QAOA-Rounded (BMZ) (orange)

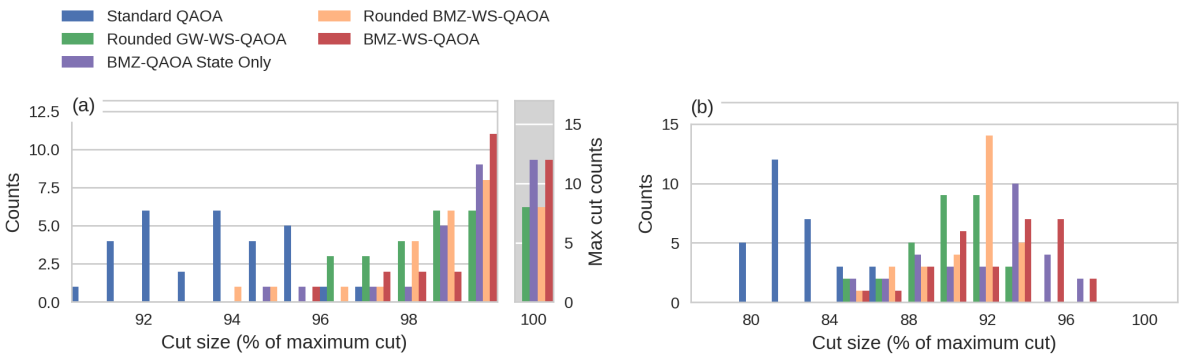

Figure 9. Each algorithm was run on 30 different randomly generated, 20 node, fully-connected graphs–each datapoint corresponds to the result for a single graph. (a) shows the distribution of best sampled cuts, with maximum cuts plotted separately on the right (note the different scale). (b) shows the distribution of circuit expectations. In each case, we used a 2 layer QAOA circuit with 100 shots.

also performs well, with only slightly fewer maximum cuts than the non-rounded approaches. QAOA-Rounded (GW) (green) has far fewer maximum cuts than any of the BMZ methods, with the distribution of best sampled cuts centered around slightly lower cut values. As expected, the warm-started approaches all improve over standard QAOA (blue), which achieves no maximum cuts, and has a distribution centered around significantly lower cut values than any of the warm-start algorithms.

It should be noted that if we measure the circuit for any of these methods enough times, we should eventually sample the maximum cut. It is of interest, then, to compare the underlying probability distribution prepared by each QAOA circuit. We do this by looking at the expected cut value; a good algorithm should prepare a distribution with a high expectation. It should be noted that in each warm-started QAOA algorithm, the classical optimizer is in fact attempting to maximize this expectation value, so this metric is a good indicator of how well each method is achieving its purpose. Figure 9b shows the distribution of circuit expectations for each method across the 30 random graphs. We see that QAOA-Warmest (BMZ) seems to have the best distribution of circuit expectations, with the highest median expectation of any method. The distribution for QAOA-Rounded (BMZ) centered around higher cut values than QAOA-Rounded (GW). Overall, the distributions for both rounded methods are centered lower than those of QAOA-Warmest (BMZ). As expected, the distribution of circuit expectations for standard QAOA is lower than all of the warm-started methods.

5.1.2. *Shots.* As we saw above, our warm-started QAOA algorithms tend to prepare distributions that are much more concentrated near high value cuts as compared to standard QAOA, indicating that a warm-started QAOA routine should need far fewer shots to achieve a given accuracy than standard QAOA. Figure 10 compares the performance of standard QAOA and QAOA-Warmest (BMZ) on 30 randomly generated complete 20-node graphs. standard QAOA achieves a steady increase in accuracy the more shots we take. The accuracy does not appear to be flattening out even all the way up to 4096, the largest shot value we tested. This is what we would expect, since more shots means more chances of sampling in the tails of the underlying distribution, and in particular more chances to sample cut values much higher than the standard QAOA expectation value. In contrast, QAOA-Warmest (BMZ) experiences only a limited accuracy improvement as we increase the number of shots. This indicates that BMZ warm-starting can be an effective method for achieving high accuracy with a limited shots budget.

5.1.3. *Single-layer parameter landscapes.* In the case of a single layer of QAOA, we can plot the parameter landscapes corresponding to the two variational circuit parameters (γ and β). While all the plots exhibit interesting symmetries, we focus on examining the significant structural differences in the landscapes corresponding to each approach. For example, standard QAOA has a saddle running horizontally at $\gamma = 0$, with peaks and troughs at γ values above and below. In contrast, QAOA-Rounded (BMZ) has its two optimal peaks at $\gamma = 0$, though the structure of the local mins is similar to standard QAOA (though mirrored across

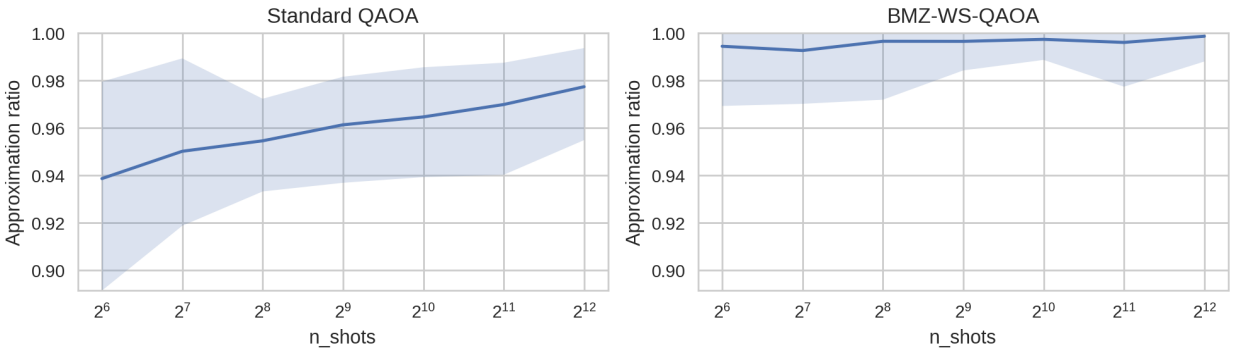

Figure 10. Comparison of standard QAOA and QAOA-Warmest (BMZ) on 30 randomly generated 20-node complete graphs for different numbers of shots. Both algorithms used 1 layer. The solid blue lines give the average results, while the shaded regions contain all the results over the 30 graphs. Note the log scale on the x-axis. QAOA-Warmest (BMZ) achieves better average accuracy with just 64 shots than standard QAOA with 4096 shots, and gains little if any advantage from increasing the number of shots past a certain point, indicating that BMZ warm-starting is an effective method for achieving high accuracy with a limited shots budget.

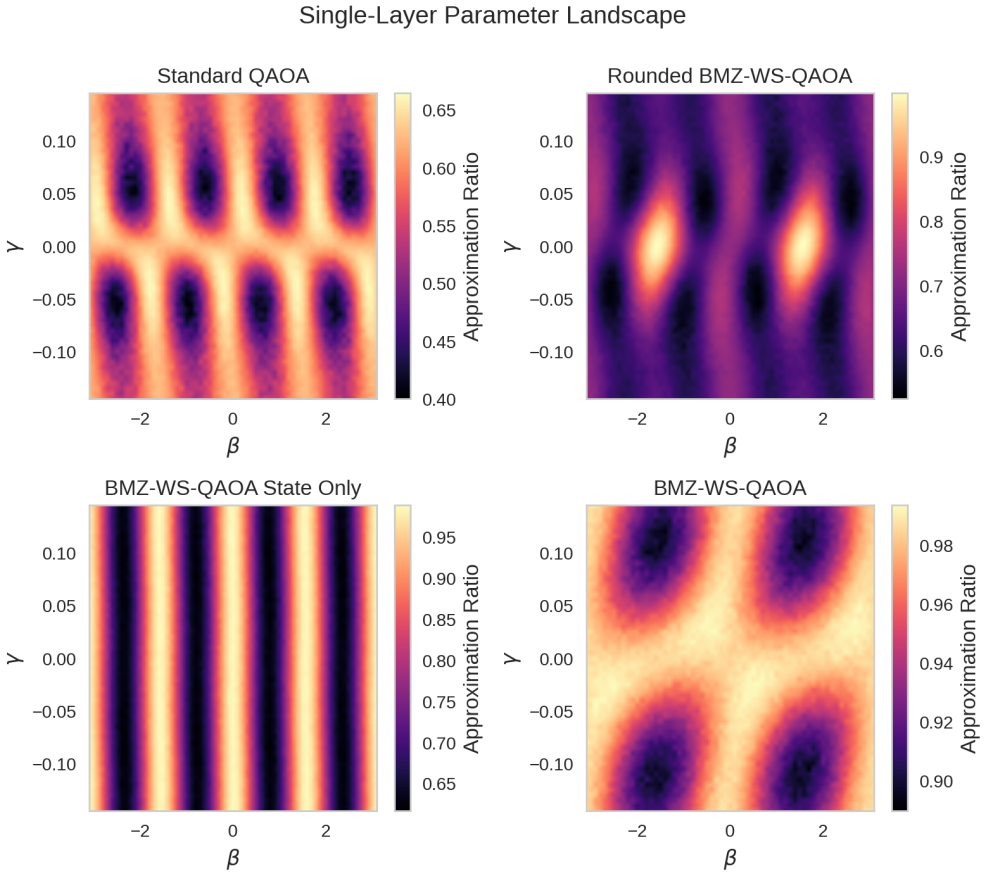

Figure 11. Parameter landscape for a single layer of QAOA using no warm-starting, rounded BMZ warm-starting, and BMZ warm-starting. β and γ are the parameters used for the mixer and cost propagators, respectively. Note the different scales for each colormap.

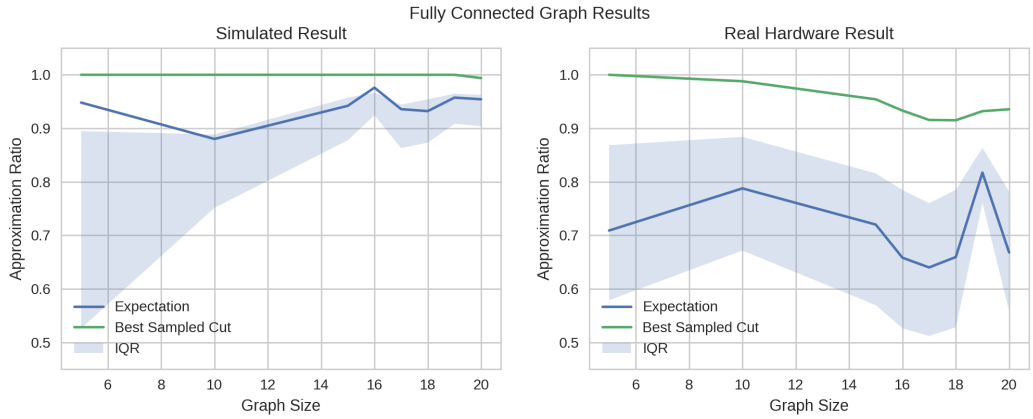

Figure 12. Results for fully-connected graphs with between 5 and 20 nodes. For each graph size, QAOA-Warmest (BMZ) was run on real hardware and on a simulator. Hardware results are plotted on the right and simulated results are plotted on the left. In each plot, the green line gives the best measured cut, the blue line gives the estimated circuit expectation, and the shaded blue region gives the middle 50% (IQR) of the measured cuts.

$\beta = 0$). The plot for QAOA-Rounded (GW) has been omitted because it is structurally identical to the plot for QAOA-Rounded (BMZ).

5.2. **Hardware results.** We now move to experimental results from runs on real quantum computers. We can consider two sources of difficulty when using currently available quantum hardware. First, we are limited by the number of qubits on a chip. To use QAOA to solve Max-Cut for a graph with n nodes, we will need at least n qubits, with more being needed if the graph connectivity exceeds the hardware connectivity. Second, we are limited by coherence times, meaning that only circuits up to a certain depth will be able to execute successfully.

We ran QAOA-Warmest (BMZ) on graphs of varying sizes using the Rigetti Aspen-M-3, a 79 qubit superconducting QPU. First, we consider fully-connected graphs with between 5 and 20 nodes. Because the Aspen-M-3 architecture is not fully-connected, running QAOA on a fully-connected graphs requires a large number of extra SWAP gates, and we found that 20 nodes was around the upper limit of problem size that can be run successfully. In order to get results for larger graph sizes, we also considered graphs with between 30 and 66 nodes whose connectivity matches that of the Aspen-M-3 qubits. We found that the QAOA circuits resulting from graphs larger than this are too deep, causing the runs to fail.

Figure 12 shows the results for the fully-connected graphs. For each graph size, we plot a single run of QAOA-Warmest (BMZ) on the Aspen-M-3 in red. Plotted in blue are the simulation results for QAOA-Warmest (BMZ) on 30 randomly generated graphs for each graph size. We include plots for both the best measured cut value (left) and the estimated circuit expectation (right). As we would expect, the accuracy on current hardware is lower and drops faster than the accuracy of our noiseless simulations. Still, our best measured cuts never drop below an approximation ratio of 0.92. Notably, the estimated circuit expectation for our hardware results seems to remain relatively constant as we increase the graph size, running more or less parallel to (and below) the average expectation of the simulated results.

Figure 13 shows the results for the larger graphs with lower connectivity matching the Aspen-M-3 qubits (note that for graphs of this size, we are unable to run simulations to compare against). We plot the best measured cut value (left) and the estimated circuit expectation (right). We again see the accuracy steadily dropping as the graph size increases. We attribute this to the increased impact of noise on quantum circuits with this level of width and depth.

In general, we expect the accuracy to decrease as the graph size increases, as larger graphs lead to harder Max-Cut problems. In addition, as the graphs get larger, the QAOA circuits get both wider and deeper, and we expect the impact of noise to be larger.

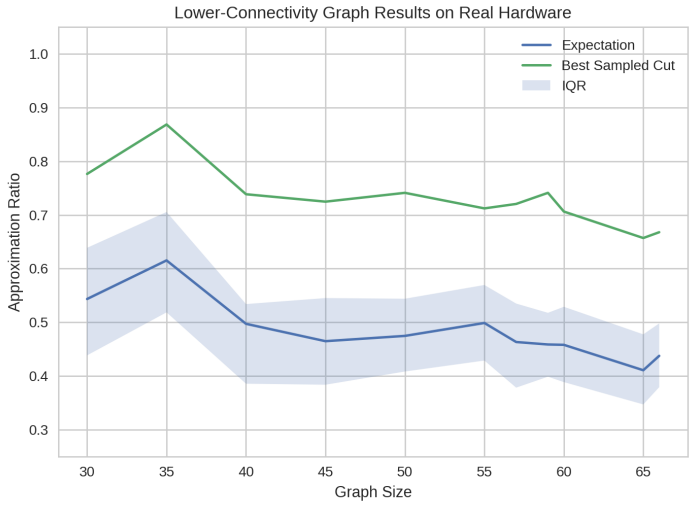

Figure 13. Results for lower-connectivity graphs with between 30 and 66 nodes. For each graph size, QAOA-Warmest (BMZ) was run on the quantum hardware. The green line gives the best measured cut, the blue line gives the estimated circuit expectation, and the shaded blue region gives the middle 50% (IQR) of the measured cuts.

6. Acknowledgements

Authors acknowledge support from the Automotive Research Center at the University of Michigan in accordance with Cooperative Agreement W56HZV-19-2-0001 with U.S. Army DEVCOM Ground Vehicle Systems Center.

References

[1] Hedayat Alghassi, Amol Deshmukh, Noelle Ibrahim, Nicolas Robles, Stefan Woerner, and Christa Zoufal. A variational quantum algorithm for the feynman-kac formula. *Quantum*, 6:730, 2022.

[2] Carlos Bravo-Prieto, Ryan LaRose, Yigit Subasi, Lukasz Cincio, and Patrick J Coles. Variational quantum linear solver. *arXiv preprint arXiv:1909.05820*, 2019.

[3] Samuel Burer, Renato Monteiro, and Yin Zhang. Rank-two relaxation heuristics for max-cut and other binary quadratic programs. *SIAM Journal on Optimization*, 12, 07 2001.

[4] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving, 2020.

[5] John Canny and John Reif. New lower bound techniques for robot motion planning problems. In *28th Annual Symposium on Foundations of Computer Science (sfcs 1987)*, pages 49–60, 1987.

[6] Venkata Sirimuvva Chirala, Saravanan Venkatachalam, Jonathon M. Smereka, and Sam Kassoumeh. A Multi-Objective Optimization Approach for Multi-Vehicle Path Planning Problems Considering Human–Robot Interactions. *Journal of Autonomous Vehicles and Systems*, 1(4):041002, 01 2022.

[7] James Dallas, Michael P. Cole, Paramsothy Jayakumar, and Tulga Ersal. Terrain adaptive trajectory planning and tracking on deformable terrains. *IEEE Transactions on Vehicular Technology*, 70:11255–11268, 2021.

[8] R. Domínguez, E. Onieva, J. Alonso, J. Villagra, and C. González. Lidar based perception solution for autonomous vehicles. In *2011 11th International Conference on Intelligent Systems Design and Applications*, pages 790–795, 2011.

[9] Daniel J. Egger, Jakub Mareček, and Stefan Woerner. Warm-starting quantum optimization. *Quantum*, 5:479, June 2021.

[10] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm. *arXiv preprint arXiv:1411.4028*, 2014.

[11] Alessandro Farinelli, Luca Iocchi, and Daniele Nardi. Distributed on-line dynamic task assignment for multi-robot patrolling. *Autonomous Robots*, 41:1321 – 1345, 2016.

[12] Huckleberry Febbo, Paramsothy Jayakumar, Jeffrey L. Stein, and Tulga Ersal. Real-Time Trajectory Planning for Automated Vehicle Safety and Performance in Dynamic Environments. *Journal of Autonomous Vehicles and Systems*, 1(4):041001, 12 2021.

[13] P. Frederick, R. Kania, M.D. Rose, D. Ward, U. Benz, A. Baylot, M.J. Willis, and H. Yamauchi. Spaceborne path planning for unmanned ground vehicles (ugvs). In *MILCOM 2005 - 2005 IEEE Military Communications Conference*, pages 3134–3141 Vol. 5, 2005.

[14] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[15] Michel X. Goemans and David P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*, 42(6):1115–1145, nov 1995.

[16] A. K. Kulatunga, D.K. Liu, Gamini Dissanayake, and Sarath. B. Siyambalapitiya. Ant colony optimization based simultaneous task allocation and path planning of autonomous vehicles. *2006 IEEE Conference on Cybernetics and Intelligent Systems*, pages 1–6, 2006.

[17] Sam Kysar, Jeremy Bos, Akhil Kurup, Zach Jeffries, Jake Carter, and Casey Majhor. Unstructured with a point: Validation and robustness evaluation of point-cloud based path planning. *SAE International Journal of Advances and Current Practices in Mobility*, 3, 04 2021.

[18] Jacoby Larson, Michael Bruch, and John Ebken. Autonomous navigation and obstacle avoidance for unmanned surface vehicles. In Grant R. Gerhart, Charles M. Shoemaker, and Douglas W. Gage, editors, *Unmanned Systems Technology VIII*, volume 6230, page 623007. International Society for Optics and Photonics, SPIE, 2006.

[19] Damien Matti, Hazım Kemal Ekenel, and Jean-Philippe Thiran. Combining lidar space clustering and convolutional neural networks for pedestrian detection. In *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 1–6, 2017.

[20] Mahmudun Nabi, Robert Benkoczi, Sherin Abdelhamid, and Hossam S. Hassanein. Resource assignment in vehicular clouds. In *2017 IEEE International Conference on Communications (ICC)*, pages 1–6, 2017.

[21] Yurii Nesterov and Arkadii Nemirovskii. *Interior-Point Polynomial Algorithms in Convex Programming*. Society for Industrial and Applied Mathematics, 1994.

[22] Lale Özbakır, Adil Baykasoğlu, and Pinar Tapkan. Bees algorithm for generalized assignment problem. *Appl. Math. Comput.*, 215:3782–3795, 2010.

[23] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J Love, Alán Aspuru-Guzik, and Jeremy L O'brien. A variational eigenvalue solver on a photonic quantum processor. *Nature communications*, 5:4213, 2014.

[24] Vadim N. Smelyanskiy, Eleanor Gilbert Rieffel, Sergey Knysh, Colin P. Williams, Mark W. Johnson, Murray C. Thom, William G. Macready, and Kristen L. Pudenz. A near-term quantum computing approach for hard computational problems in space exploration. *arXiv: Quantum Physics*, 2012.

[25] Reuben Tate, Jai Moondra, Bryan Gard, Greg Mohler, and Swati Gupta. Warm-Started QAOA with Custom Mixers Provably Converges and Computationally Beats Goemans-Williamson's Max-Cut at Low Circuit Depths. *Quantum*, 7:1121, September 2023.

[26] Maggie Wigness, Sungmin Eum, John G. Rogers, David Han, and Heesung Kwon. A rugd dataset for autonomous navigation and visual perception in unstructured outdoor environments. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5000–5007, 2019.

[27] Robert L. Williams and Jianhua Wu. Dynamic obstacle avoidance for an omnidirectional mobile robot. *Journal of Robotics*, 2010.