Quantum-assisted finite-element design optimization

Vreumingen, D. van; Neukart, F.; Dollen, D. von der; Othmer, C.; Hartmann, M.; Voigt, A.C.; Bäck, T.H.W.

Quantum-assisted Finite-element Design Optimization

Dyon van Vreumingen[1, 3], Florian Neukart[1, 3, *], David Von Dollen[1], Carsten Othmer[2], Michael Hartmann[2], Arne-Christian Voigt[2], and Thomas Bäck[3]

[1] *Volkswagen Group Region Americas, USA*
[2] *Volkswagen Group Research, Germany*
[3] *Leiden Institute for Advanced Computer Science, Leiden University, The Netherlands*
[*] Corresponding author: Florian Neukart (florian.neukart@vw.com)

August 13, 2019

Abstract

Quantum annealing devices such as the ones produced by D-Wave systems are typically used for solving optimization and sampling tasks [1–15], and in both academia and industry the characterization of their usefulness is subject to active research. Any problem that can naturally be described as a weighted, undirected graph may be a particularly interesting candidate [16,17], since such a problem may be formulated a as quadratic unconstrained binary optimization (QUBO) instance, which is solvable on D-Wave's Chimera graph architecture. In this paper, we introduce a quantum-assisted finite-element method for design optimization. We show that we can minimize a shape-specific quantity, in our case a ray approximation of sound pressure at a specific position around an object, by manipulating the shape of this object. Our algorithm belongs to the class of quantum-assisted algorithms, as the optimization task runs iteratively on a D-Wave 2000Q quantum processing unit (QPU), whereby the evaluation and interpretation of the results happens classically. Our first and foremost aim is to explain how to represent and solve parts of these problems with the help of a QPU, and not to prove supremacy over existing classical finite-element algorithms for design optimization.

Keywords: quantum computing, quantum physics, finite-element, design optimization, QUBO

1 Introduction

According to the laws of quantum mechanics, a quantum mechanical system, which is in the ground state (state of minimal energy) of a time-independent system, also remains in the ground state if a change to it happens only slowly, i.e. adiabatically. This is known as the adiabatic theorem. The idea of adiabatic quantum computing is to construct a system having a ground state that is still unknown at that time, which corresponds to solving a particular problem, and another one whose ground state is easy to prepare experimentally. Subsequently, the easy-to-prepare system is adiabatically transferred to the system whose ground state one is interested in, and then measured. If the transition is slow enough, one can obtain a minimum-energy solution to the problem. D-Wave's QPUs deploy a system described by the two-dimensional Ising spin hamiltonian [16, 17]:

$$H_{\mathbf{h},\mathrm{J}}(\mathbf{s}) = \sum_{i=1}^{n} h_i s_i + \sum_{\langle i,j \rangle} J_{ij} s_i s_j. \tag{1}$$

Here, $\mathbf{s}$ is a vector of n spins, $s_i \in \{-1, 1\}$, which carry an individual energy weight h_i and are interconnected through 2-local couplings J_{ij}. The sum in the second term of the hamiltonian runs over only those spin pairs which are connected, as $J_{ij} = 0$ for uncoupled spin pairs. As such, the hamiltonian is characterised by the linear weight vector $\mathbf{h}$ and the coupling matrix J. The search for the minimum spin configuration $\mathbf{s}_{\min}$ for the Ising hamiltonian is known to be NP-hard [15, 18].

It is generally preferable for a computational application to work with $\{0, 1\}$-valued bits of information as opposed to spins, which can be achieved through the transformation $x_i = \frac{1}{2}(s_i + 1)$. This formulation of the Ising spin problem, which is polynomially reducible to the original form and vice versa, is known as a *quadratic unconstrained binary optimization* problem, or QUBO for short, and can be solved by the QPU in the same fashion as a conventional Ising model. The equivalence between the two problem classes implies that any problem to be solved with the D-Wave QPU may be formulated either as a QUBO instance or directly as an Ising model. The objective quantity that the QPU minimizes in the QUBO case is given by the quadratic form [16, 17]

$$\mathrm{Obj}_{\mathrm{Q}}(\mathbf{x}) = \mathbf{x}^{\top} \mathrm{Q}\, \mathbf{x}, \tag{2}$$

where $\mathbf{x}$ is an N-sized vector of $\{0, 1\}$-valued variables, and Q is an $N \times N$ real-valued upper triangular matrix containing the (adjusted) linear weights in the diagonal, and the couplings in the off-diagonal entries.

D-Wave's QPU physically implements an undirected graph in which the qubits,

representing the binary variables, are the nodes, and the couplings the edges. The initial configuration is set up such that all qubits are in uniform superposition, $|x_i\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. During the annealing cycle, the state is evolved according to the energy landscape described by Q. Eventually, when the system reaches the ground state, a minimum solution to the QUBO problem is found.

In this work, we demonstrate a method for using the QPU as an optimizer for a finite-element design problem. That is, we seek to optimize the shape of a 3D body defined by a finite number of elements against certain physical circumstances, by expressing the physical interaction of the elements in a QUBO form, and having the QPU find the minimum-energy configuration corresponding to a (sub)optimal shape. The next sections describe and discuss the problem in the framework of finite-element methods, as well as our approach to the problem and the observed results.

This paper is structured as follows. Sections 2 and 3 briefly discuss the research field of finite-element methods, the problem we address and its context in vehicle engineering, and how the two relate in our work. Section 4 outlines our method for solving the problem, including a detailed description of the QUBO formulation and the procedure that our proposed algorithm follows. Section 5 showcases the results in terms of shape optimization that we obtained by executing the algorithm, and examines a number of features and limitations of the algorithm that appear from these results. Lastly, we present our conclusions in section 6 and give an outlook on possible future work in section 7.

2 Finite-element methods

Finite-element methods (FEM) are a general group of numerical methods used in various physical tasks. Most well-known is the application of FEM in the investigation of the strength and deformation of solids with a geometrically complex shape, because here the use of classical methods, e.g. beam theory, proves to be too time-consuming or impossible. Logically, the FEM is based on the numerical solution of a complex system of differential equations. The computation domain, e.g. the solid, is divided into finitely many subdivisions of a simple form, creating a mesh of the original solid using a finite number of corner points ('vertices') and faces ('elements' or 'simplices'). The physical behaviour of this meshed solid can be more easily calculated with well-known elementary functions due to its simplified geometry. The physical behaviour of the whole body is modelled in the transition from one element to the next, through very specific problem-dependent continuity conditions that must be fulfilled by the elementary functions. These functions contain parameters that usually have a physical meaning, such as the shift of a

certain point in the component at a given time. The search for the motion function is thus returned to the search for the values functions' parameters. By using more and more parameters such as more and/or smaller elements and higher order functions, the accuracy of the approximate solution can be improved. The development of the FEM was possible in essential stages only by the development of powerful computers, since it requires considerable computing power. Therefore, this method was formulated from the outset to be processed on computers. Further information can be found in the work by Pepper et al. [19].

3 Quantum-assisted finite-element method for design optimization

With the algorithm we introduce in the following sections, we are able to find designs based on a quantity that we minimize. One practical example concerns minimizing the wind noises on an external mirror of a vehicle, and another one is minimizing the noises through vibrations caused by the engine or different road conditions in a vehicle. The areas to optimize are commonly obtained with a complex finite-element simulation, and evolutionary algorithms have proven to be very valuable for searching the design space [20–23]. As one part of the wind noise prediction simulation chain, we can compute acoustic sources on the mirror surface. This is an instance of a so-called acoustic scattering problem, which has to be solved in order to extract only those sources which are most relevant (noise-causing) at the position of the passengers. Solving the scattering problem is very time-consuming, especially in real vehicle applications, where the number of elements can be in the order

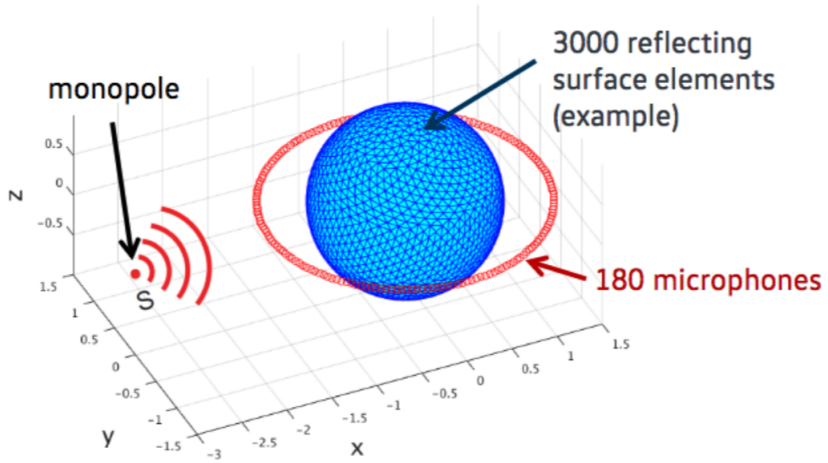


Figure 1. Acoustic monopole emitting a spherical wave scattered by a rigid sphere.

of millions. Even for relatively few, a direct solver implementing straightforward matrix inversion quickly runs into memory and computation time limits. Thus, we are after finding an algorithm that scales better with an increasing number of elements. The present state of quantum computing does not allow us to compete with classical algorithms in terms of number of elements or speed, as the currently newest version of the QPU, containing approximately 2048 qubits, can only reliably find minor embeddings for shapes with up to 50 elements. Of course, we can split a QUBO instance with more than 50 elements and process problems of arbitrary size, but this significantly increases the required computation time.

In the introduced algorithm, we start with an initial shape and intend to find a shape that deflects sound waves emitted by an acoustic monopole source such that the sound pressure within an area at another position around the shape is minimized. In the same instance, our algorithm must be form-preserving, as in the end the shape should still resemble the initial design. In the scenario we describe, the initial shape is a spherical mesh with finitely many triangular surface elements (simplices), which is hit by sound waves emitted from an acoustic monopole (see fig. 1). Fig. 1 shows microphones positioned around the shape, and the objective is to minimize the sound pressure at any position of choice, by altering the sphere's shape. As the size of the current D-Wave QPU is limited to 2048 qubits and each qubit bears only 6 connections to neighboring qubits, we make a number of assumptions and approximations in order to make this problem feasible for submission to the QPU with a reasonable number of elements. More complex formulations however are possible, but adding more interactions would require more qubits, allowing processing of fewer elements within reasonable execution time.

4 Approach

In the definition of the sound scattering problem, we make one major simplification to ensure that the resulting formulation is a finite-element method that is well-suited for the QPU, which is to approximate sound waves as rays (i.e. straight lines in 3D space). That is, propagation of sound waves is treated similarly to the propagation of light as done in graphical raytracing [24, 25], ignoring wave effects and interference altogether. The most important reason for this is that it allows us to consider each element separately in terms of its contribution to the measured sound pressure, as it avoids the necessity to construct a wave-based model harbouring high degrees of interaction between elements through sound wave interference. After all, the multitude of incident and scattered sound waves creates

a highly complex situation that cannot be described without distant (i.e. non-neighbouring) element-element coupling. Since we seek to devise a QPU-assisted finite-element method for optimising a shape, finding a way to describe a 'first-order approximation' with only neighbour couplings is more important than figuring out a very accurate scattering solution. Since we know that sound waves in reality reflect linearly off a surface identically to light rays, we use this as the approximation to base our quantum-assisted algorithm on.

The algorithm is a 3D search routine, which iteratively considers different candidate positions for each vertex in the shape (not to be confused with a qubit node in the QPU graph), and then lets the QPU decide which vertex arrangement causes the least number of rays to be reflected towards a microphone. This microphone is represented by a rectangularly bounded plane positioned next to the shape (see fig. 2). In each iteration, the routine assigns to each vertex K 'mutations', which are small random deviations from the original vertex position; that is, for each vertex $\mathbf{v}^i$ in the set V of vertices, it considers $\mathbf{v}^i + \mathrm{d}\mathbf{v}^i_1, \cdots, \mathbf{v}^i + \mathrm{d}\mathbf{v}^i_K$ with $\mathrm{d}\mathbf{v}^i_j$ small. Each vertex is encoded by K qubits, and the $|1\rangle$ state of the j-th corresponding qubit indicates that mutation j was assigned to this vertex (if the state is $|0\rangle$, this particular mutation was not assigned). For each simplex, the partial loss ℓ, being the amount of pressure received from this simplex, is computed separately for each of the K^3 simplex configurations created from the vertex mutations (i.e. three vertices per simplex, and K mutations for each vertex). The QUBO matrix Q is then constructed so that it contains, for each vertex, the loss information associated with the simplices neighbouring the vertex. Based on this information, the QPU will choose the minimal loss vertex configuration among the ones supplied, and use these as the input for the next iteration. This continues for a given number of iterations, or until convergence is observed.

A more detailed description of the QUBO formulation is provided in the next section.

4.1 QUBO problem formulation

Define S as the set of all simplices s determining the shape, $N = |V|$ and C as the set of all configurations c over the entire shape, where c is a list of vertex mutation assignments $\{(i, j)\}$, with $i \in \{1, \ldots, N\}$ and $j \in \{1, \ldots, K\}$, indicating assignment of mutation j to vertex i (i.e. $\mathbf{v}^i \mapsto \mathbf{v}^i + \mathrm{d}\mathbf{v}^i_j$). Each configuration is a complete list, in that every vertex is assigned a mutation. Define a loss function $\mathcal{L}(S, c)$, which maps a configuration c to a loss value, to be the total of the partial losses $\ell(s, c)$ of

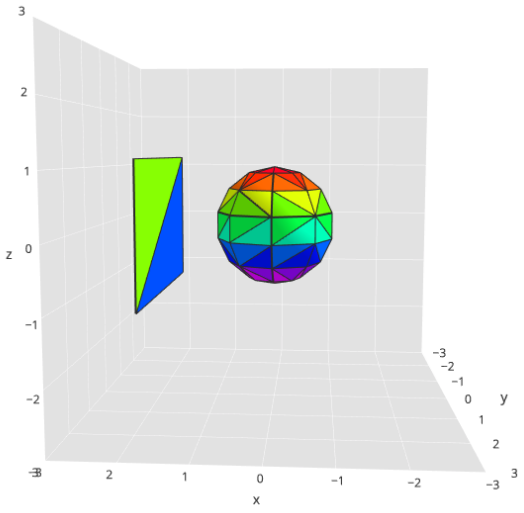


Figure 2. A rigid sphere, which serves as the initial shape, and a rectangular area at which the sound pressure must be minimized. The mere purpose of the colour scheme is visual aid.

simplices $s \in S$ for configuration c,

$$\mathcal{L}(S, c) = \sum_{s \in S} \ell(s, c), \tag{3}$$

and a loss partition function $\mathcal{Z}(S, C)$, the sum of the loss function over all configurations:

$$\mathcal{Z}(S, C) = \sum_{c \in C} \mathcal{L}(S, c) = \sum_{c \in C} \sum_{s \in S} \ell(s, c). \tag{4}$$

In this form, $\mathcal{Z}$ is a function of K^N configurations. Now, we observe that this sum can be rewritten by visiting all edges $(\mathbf{v}, \mathbf{w})$ in the edge set E, and considering for each edge the two simplices adjacent to that edge. Since each simplex has three edges, this means each simplex is counted thrice, so we divide this new total by 3, to obtain:

$$\mathcal{Z}(S, C) = \frac{1}{3} \sum_{c \in C} \sum_{(\mathbf{v}, \mathbf{w}) \in E} \sum_{s \in S_{(\mathbf{v}, \mathbf{w})}} \ell(s, c), \tag{5}$$

where $S_{(\mathbf{v}, \mathbf{w})}$ is the set of the two simplices adjacent to edge $(\mathbf{v}, \mathbf{w})$.

Now notice that there are K^{N-3} configurations which fix a triple of mutations for three vertices of a simplex s, and are thus equivalent for this particular simplex. As such, instead of counting each configuration separately, we consider only K^3

configurations that are nonequivalent with respect to this simplex to sum over (represented by the set C_s), and multiply the result by K^{N-3}:

$$\mathcal{Z}(S, C) = \frac{K^{N-3}}{3} \sum_{(\mathbf{v},\mathbf{w})\in E} \sum_{s\in S_{(\mathbf{v},\mathbf{w})}} \sum_{c\in C_s} \ell(s, c). \tag{6}$$

This representation of the partition function now gives us an intuitive way to define a QUBO matrix Q for this problem. This instance is to be minimized by a $\{0, 1\}$-valued vector $\mathbf{x}$ representing a configuration $c(\mathbf{x})$, whose entry corresponding to the mutation assignment (i, j) is 1 if $\mathbf{v}^i$ is assigned mutation j and 0 otherwise, as stated before. That is, we view each entry x_{ij} as representing whether mutation (i, j) is included in the configuration list of $c(\mathbf{x})$ (in which case $x_{ij} = 1$) or not (implying $x_{ij} = 0$). The edge pairs naturally correspond to the off-diagonal terms of this matrix: for any edge pair $(\mathbf{v}^{i_1}, \mathbf{v}^{i_2})$ with mutations (i_1, j_1) and (i_2, j_2) respectively, we only need to sum over the partial loss values for all possible configurations regarding the two neighbouring simplices. If we define $\hat{\ell}(s, j_1, j_2, k)$ to be the partial loss from a simplex s adjacent to edge $(\mathbf{v}^{i_1}, \mathbf{v}^{i_2})$ (that is, $s \in S_{(\mathbf{v}^{i_1},\mathbf{v}^{i_2})}$) when its third, off-edge vertex is assigned mutation k (while $\mathbf{v}^{i_1}$ is assigned mutation j_1 and $\mathbf{v}^{i_2}$ is assigned mutation j_2), we thus obtain the following matrix form:

$$Q^{i_1 j_1}_{i_2 j_2} = \alpha \sum_{s\in S_{(\mathbf{v}^{i_1},\mathbf{v}^{i_2})}} \sum_{k=1}^{K} \hat{\ell}(s, j_1, j_2, k). \tag{7}$$

Here, α is an energy scaling factor that absorbs the $K^{N-3}/3$ in front of the sum in eq. 7 (in practice, this K^{N-3} will turn out to be huge, so adjustment is necessary). In this form of Q, each entry fixes an edge, and a configuration for both vertices of this edge. Since Q contains K rows and K columns for each vertex, it is an $NK \times NK$ matrix.

Lastly, it is important to make sure the QPU returns a result vector $\mathbf{x}$ such that each vertex is being assigned only one mutation in the corresponding configuration $c(\mathbf{x})$. Since $\mathbf{x}$ is binary, this is equivalent to requiring

$$\forall i:\ 0 = \left(\sum_{j=1}^{K} x_{ij} - 1 \right)^2 = -\sum_{j=1}^{K} x_{ij} + 2 \sum_{j=1}^{K} \sum_{j'>j}^{K} x_{ij} x_{ij'} + 1. \tag{8}$$

A straightforward way to enforce this requirement is by adding it as a penalty term to the loss function with some large constant penalty coefficient λ, as proposed in a

recent paper on QPU traffic flow optimization [15]:

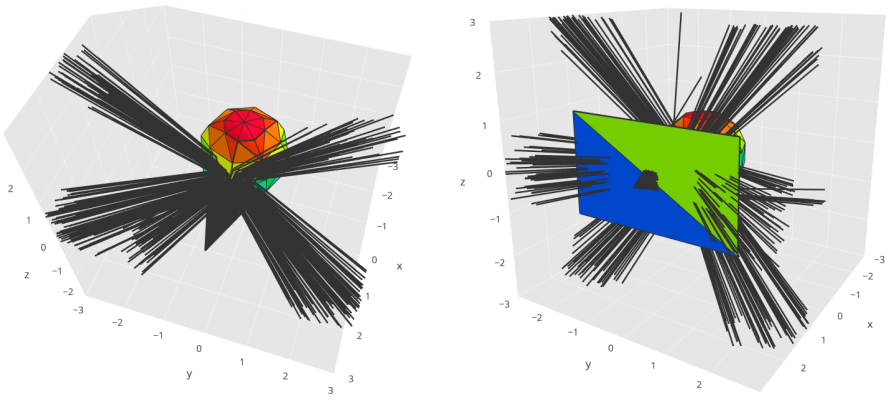
$$\tilde{\mathcal{L}}(S, \mathbf{x}) = \mathcal{L}(S, c(\mathbf{x})) + \lambda \sum_i \left(\sum_{j=1}^{K} x_{ij} - 1 \right)^2. \tag{9}$$

In the QUBO matrix, this directly translates to adding $-\lambda$ to the diagonal elements Q_{ij}^{ij} and adding 2λ to the off-diagonal elements $Q_{ij'}^{ij}$ $(j' > j)$ corresponding to vertex $\mathbf{v}^i$. Provided λ is large enough, this measure guarantees the QPU sets exactly one of the bits $x_{i1}, \ldots, x_{iK}$ to 1, as any infeasible assignment would cause an increase in loss that would be higher than any possible gain from selecting a different configuration.

4.2 Algorithm

With an overview of the procedure in our approach, and an explanation of the QUBO formulation, we can now turn to the algorithm itself. This iterative algorithm executes the following steps.

1. First, we generate the low-resolution mesh of an initial spherical shape. The vertices of this shape are conveniently represented as rectangular lattice points in the (θ, ϕ) space of spherical coordinates (the radius r may be chosen equal to unity without loss of generality). The edges of the mesh can then by found by Delaunay tessellation of this lattice. With the method of Delaunay triangulation, points in the $\mathbb{R}^2$ plane are transformed into triangles so that there are no other points within the circumscribed circle of each triangle. The



A **B**

Figure 3. Initial setup for the first experiment with the monopole at $(2.5, 0, 0)$. (**A**) Scattering of 300 incoming rays, casted from the monopole, off the spherical surface. (**B**) A significant number of rays is reflected backwards, intersecting the microphone. The rays crossing the microphone in the centre are considered incoming rays and are not counted towards the recorded sound pressure.

method is used, for example, to optimize calculation networks for many finite-element methods. As a result, the triangles of the edge set have the largest possible internal angles; mathematically speaking, the smallest interior angle over all triangles is maximized. This feature is very desirable in computer graphics because it minimizes rounding errors. The algorithm responsible for computing Delaunay tessellations is explained in detail by Dobkin et al. [26]. Given the vertices and edges in spherical coordinate space, a 3D spherical shape is constructed by the coordinate map $x_i = \sin\theta_i \cos\phi_i$, $y_i = \sin\theta_i \sin\phi_i$, $z_i = \cos\theta_i$. The convex hull of this shape is created around these 3D dots by drawing a face for all triangles, and the outward normal for each triangle is calculated. After this initial setup, the sequence of iterations starts.

2. As the first step in each iteration, K vertex mutations are computed for each vertex. The mutations are chosen probabilistically such that $\mathrm{d}\mathbf{v}_j^i$ is within a sphere of decreasing radius $R_i = \beta\rho_i t^{-\mu}$, with t the current iteration and μ a constant. That is, each $\mathrm{d}\mathbf{v}_j^i$ is picked with (uniformly) random tangential and azimuthal angles, and uniformly random radius in the interval $[0, R_i)$. Here, ρ_i is a shape-dependent bound for each vertex, whose purpose is to prevent the shape from becoming chaotic[1]. The factor β acts as a control parameter setting the step size of the algorithm. Furthermore, in addition to this $(1, K)$-like search method (in analogy to $(1, \lambda)$ search in evolutionary strategies, with selection occuring in step 5), we implement an option for $(1 + [K - 1])$ search by allowing $\mathrm{d}\mathbf{v}_1^i = \mathbf{0}$ for all vertices i.

3. For each simplex s, we compute the K^3 partial loss values $\hat{\ell}(s, i, j, k)$. These are determined by casting a set number of rays towards that simplex when its first vertex is in mutation i, its second in mutation j and its third in mutation k, and counting the fraction of rays that hits the rectangular microphone plane.

4. From these partial loss values, the $NK \times NK$-size QUBO matrix Q is computed as defined in the previous section. This matrix is then submitted to the QPU.

5. The QPU returns an NK-size bitstring $\mathbf{x}$ containing the preferred mutations of each vertex that yield minimal loss among all configurations. As mentioned in section 4.1, this bitstring is of the form $[x_{11}, x_{12}, \ldots, x_{1K};\ x_{21}, \ldots x_{2K};\ \ldots\ ; x_{N1}, \ldots, x_{NK}]$, where for each vertex i, only one of the bits $x_{i1}, \ldots, x_{iK}$ is 1,

[1] By chaotic we mean the shape having too sharp corners, vertices extruding too far from the shape, edges intersecting other simplices etc., as well as the shape generally containing too many or too deep concavities. In practice, ρ_i is determined by a soft convexity constraint which ensures that, as long as $\beta \leq 1$, moving a vertex $\mathbf{v}_i$ by a distance R_i in any direction will approximately retain the convexity of the shape. Since preserval of the convexity from the viewpoint of one vertex depends only on its neighbour vertices (and itself), ρ_i is defined precisely by the position of $\mathbf{v}_i$ and the position of its neighbours.

indicating the chosen preferred mutation for this vertex, and the others are 0. The shape is subsequently adapted according to this bitstring.

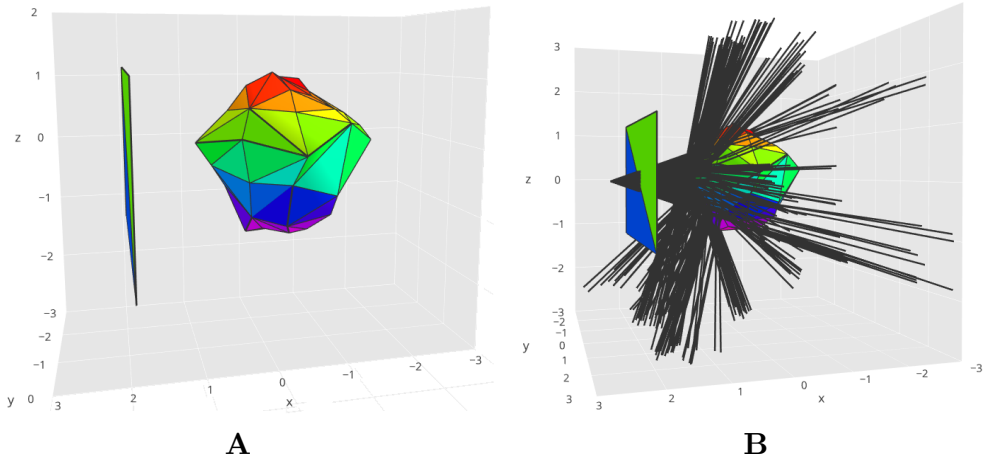
6. Steps 2–5 are repeated as often as necessary.

In the following, we show and discuss some of the resulting shapes that we have obtained from running this algorithm.

5 Experimental results and discussion

In our first experiment, we consider the situation with the monopole source sitting at $(2.5, 0, 0)$. The microphone is at $x \approx 2$ and is approximately bounded by $y \in [-2, 2]$, $z \in [-1.15, 1.15]$. See figure 3. We run the algorithm with $K = 3$, $\beta = 0.7$ and $\mu = 0.18$. At this point, we conduct $(1, K)$ search by having the routine choose $\mathrm{d}\mathbf{v}_0^i$ randomly, as described in section 4.2. For the computation of the partial loss values associated with the triangles, we sample 50 rays casted toward each triangle. It must be noted that often either all or none of the rays end up intersecting the microphone plane; however sampling more rays reduces potential variance in the partial loss calculations, making the algorithm more robust.

The resulting shape as determined by the algorithm is shown in figure 4. As one can see, the algorithm is successful in achieving its goal of minimizing the sound pressure, expressed in the amount of sound rays, at the microphone. It has found a way to adjust the front triangles such that each ray will either scatter in the



A **B**

Figure 4. Result after running the algorithm with the monopole at $(2.5, 0, 0)$ (see figure 3). (**A**) The resulting shape. Again, rainbow colours are used for visual support. The shape displays a sharp edge at the front, giving it a streamlined structure. (**B**) Rays scattering off the new shape. All rays are directed around the microphone plane in one way or another, as can be seen from the fact that no outgoing ray intersects the microphone plane.

negative x direction or, if scattered backwards in the positive x direction, travels around the microphone plane. This is clearly a consequence of the sharp tip the shape has obtained, which was absent in the case of the sphere.

It is worth noting that the rear of the sphere, at the far away end from the microphone, was deformed into a seemingly random structure. This is caused by the fact that no rays would hit this side in the first place; as such the quantum algorithm has no information about it (meaning the quadratic QUBO entries corresponding to those triangles are zero) and will choose a random vertex in each iteration. As such, it would make sense, in a further version of this algorithm, to prune these

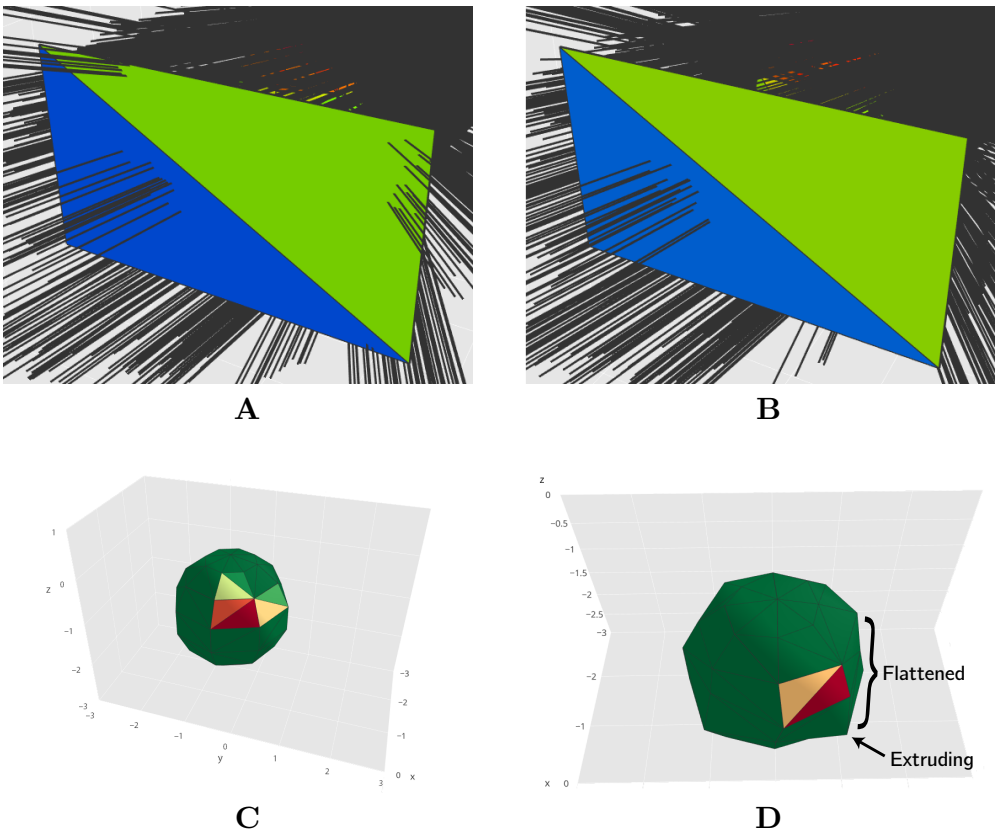


Figure 5. Results and comparison after executing the algorithm with the monopole at $(0, 3, 2)$ with a lower step size. (**A**) At first, before shape adjustment, rays intersect the microphone at three positions: at the upper left, at the upper right and slightly to the left of the centre. (**B**) The shape returned by the optimisation algorithm now reflects the rays, which would initially hit the corners of the microphone, away from it. However, the centre rays seem to remain in place. (**C**) Partial loss shading of the initial sphere. Dark green triangles reflect no rays towards the microphone, while a darker shade of red indicates higher partial loss. Shade is normalised to the maximum partial loss of any triangle. (**D**) The final shape. One can see that the right side of the shape has been flattened, having an extruded point, which contributes to reflection away from the microphone.

triangles in order to allow processing of more detailed shapes (containing more elements) on the QPU. In this work, however, we chose not to do this as our wish was to investigate the effect of the algorithm on the entire shape. After all, our problem was inspired by external vehicle mirror design, which does not allow for cut shapes. The values for β and μ were chosen by trial-and-error search, by testing a small set of combinations covering $\beta \in [0.3, 1.0]$ and $\mu \in [0.15, 0.20]$. We noticed that a too low step size renders the algorithm incapable of sufficiently adapting the shape within the given number of iterations, as it usually gets stuck in a local, suboptimal point, which cannot be optimized any further. This seems to occur in particular with $(1 + [K - 1])$ search. On the other hand, a too high step size usually (especially in the case of $(1, K)$ search) produces a too irregular shape. A good example showing the consequence of choosing a too low step size can be seen in figure 5. Here, we moved the monopole to $(0, 3, 2)$ and chose a step size control $\beta = 0.3$. We observe that although two sources of loss have been eliminated, one seems to be persistent. The result in figure 5(d) with only two triangles having nonzero partial loss (which, even though not shown in the figure, is lower than that of the sphere in fig. 5(c)) is most likely considered as a local optimum by the algorithm, meaning it chooses not to depart from there.

6 Conclusions

In this work, we have presented a finite-element method for optimizing a three-dimensional shape under given physical criteria. By formulating an approximation of this finite-element problem in a QUBO form, and by embedding the corresponding matrix on the QPU as specified, we have been able to show that it is possible to successfully carry out finite-element design optimization on a D-Wave QPU. We have shown that by supplying an initial shape we can optimize the geometry to minimize a specified quantity, such as sound pressure, at a target area around the shape or the vibration of single elements, and in the same instance partially preserve the geometry. This is important, as if we supply the design of an outside mirror of a vehicle and intend to minimize the noise at the passenger's positions, we still want to end up with a design that captures all the properties a mirror must have.

Furthermore, we have demonstrated how to usefully combine the computing power of a classical computer with that of a quantum computer. That is, we calculate the sound pressure on an initial geometry classically and have the QPU solve the problem prepared on the classical computer. It is this combination of CPU and QPU efforts that in the end yields the desired solution.

7 Future work

For the next version of the algorithm, we intend to find a formulation that will incorporate additional constraints on the final shape. In addition, we would like to add wave behaviour corrections to increase the degree of realism in the model, or alternatively, discard the ray-casting approximation and find a way to model sound waves directly. Additionally, we wish to explore scalability of the algorithm, as we should be able to process shapes with more elements by splitting the QUBO matrix with the `qbsolv` decomposing solver tool [27], instead of having the D-Wave software find minor embeddings for shapes with few elements. This will be of use in the future, when we expect new D-Wave QPU generations. With these new chips having more couplers between the qubits, we will be able to embed shapes with more elements and hopefully determine smoother geometries. We will continue to focus on laying the foundation for solving practically relevant problems by means of quantum computing, quantum simulation, and quantum optimization [16, 17, 28–33].

Acknowledgments

Thanks go to VW Group CIO Martin Hofmann and VW Group Region Americas CIO Abdallah Shanti, who enable our research.

References

[1] Marcello Benedetti, John Realpe-Gómez, Rupak Biswas, and Alejandro Perdomo-Ortiz. Estimation of effective temperatures in quantum annealers for sampling applications: A case study with possible applications in deep learning. *Physical Review A*, 94(2), 2016.

[2] Vadim N. Smelyanskiy, Davide Venturelli, Alejandro Perdomo-Ortiz, Sergey Knysh, and Mark I. Dykman. Quantum Annealing via Environment-Mediated Quantum Diffusion. *Physical Review Letters*, 118(6), 2017.

[3] Davide Venturelli, Dominic J. J. Marchand, and Galo Rojo. Quantum Annealing Implementation of Job-Shop Scheduling. jun 2015.

[4] Zhang Jiang and Eleanor G. Rieffel. Non-commuting two-local Hamiltonians for quantum error suppression. *Quantum Information Processing*, 16(4), 2017.

[5] Sergei V. Isakov, Guglielmo Mazzola, Vadim N. Smelyanskiy, Zhang Jiang, Sergio Boixo, Hartmut Neven, and Matthias Troyer. Understanding quantum

tunneling through quantum Monte Carlo simulations. *Physical Review Letters*, 117(18), 2016.

[6] B. O'Gorman, R. Babbush, A. Perdomo-Ortiz, A. Aspuru-Guzik, and V. Smelyanskiy. Bayesian network structure learning using quantum annealing. *The European Physical Journal Special Topics*, 224(1):163–188, 2015.

[7] Eleanor G. Rieffel, Davide Venturelli, Bryan O'Gorman, Minh B. Do, Elicia M. Prystay, and Vadim N. Smelyanskiy. A case study in programming a quantum annealer for hard operational planning problems. *Quantum Information Processing*, 14(1), 2014.

[8] Davide Venturelli, Salvatore Mandrà, Sergey Knysh, Bryan O'Gorman, Rupak Biswas, and Vadim Smelyanskiy. Quantum optimization of fully connected spin glasses. *Physical Review X*, 5(3), 2015.

[9] A. Perdomo-Ortiz, J. Fluegemann, S. Narasimhan, R. Biswas, and V.N. Smelyanskiy. A quantum annealing approach for fault detection and diagnosis of graph-based systems. *The European Physical Journal Special Topics*, 224(1):131–148, 2015.

[10] Sergio Boixo, Troels F. Rønnow, Sergei V. Isakov, Zhihui Wang, David Wecker, Daniel A. Lidar, John M. Martinis, and Matthias Troyer. Evidence for quantum annealing with more than one hundred qubits. *Nature Physics*, 10(3):218–224, 2014.

[11] Ryan Babbush, Alejandro Perdomo-Ortiz, Bryan O'Gorman, William Macready, and Alan Aspuru-Guzik. Construction of Energy Functions for Lattice Heteropolymer Models: Efficient Encodings for Constraint Satisfaction Programming and Quantum Annealing. In *Advances in Chemical Physics*, volume 155, pages 201–244. 2014.

[12] J.A. Smolin and G. Smith. Classical signature of quantum annealing. *Frontiers in physics*, 2(52), 2014.

[13] Alejandro Perdomo-Ortiz, Neil Dickson, Marshall Drew-Brook, Geordie Rose, and Alan Aspuru-Guzik. Finding low-energy conformations of lattice protein models by quantum annealing. *Scientific Reports*, 2, 2012.

[14] Los Alamos national laboratory. D-Wave 2X quantum computer, 2016.

[15] Florian Neukart, Gabriele Compostella, Christian Seidel, David von Dollen, Sheir Yarkoni, and Bob Parney. Traffic flow optimization using a quantum annealer. *Frontiers in ICT*, 4(29), 2017.

[16] Florian Neukart, David Von Dollen, and Christian Seidel. Quantum-assisted cluster analysis. mar 2018.

[17] F. Neukart, C. Seidel, G. Compostella, and D. Von Dollen. Quantum-enhanced reinforcement learning for finite-episode games with discrete state spaces. *Frontiers in physics*, 5(71), 2017.

[18] A. Lucas. Ising formulations of many NP problems. *Frontiers in physics*, 2(5), 2014.

[19] D.W. Pepper and J.C. Heinrich. *The finite element method: basic concepts and applications with MATLAB, MAPLE and COMSOL*. CRC press, 3 edition, 2017.

[20] A. Sanz-García, A.V. Pernía-Espinoza, R. Fernández-Martínez, and F.J. Martínez-de Pisón-Ascacíbar. Combining genetic algorithms and the finite element method to improve steel industrial processes. *Journal of Applied Logic*, 10(4):298–308, 2012.

[21] T Bäck, D B Fogel, and Z Michalewicz. Handbook of Evolutionary Computation. *Evolutionary Computation*, 2:1–11, 1997.

[22] Thomas Bäck, Peter Krause, and Christophe Foussette. Automatic Metamodelling of CAE Simulation Models. *ATZ worldwide*, 117(5):36–41, 2015.

[23] Fabian Duddeck. Multidisciplinary optimization of car bodies. *Structural and Multidisciplinary Optimization*, 35(4):375–389, 2008.

[24] Arthur Appel. Some techniques for shading machine renderings of solids. In *Proceedings of the April 30–May 2, 1968, spring joint computer conference on - AFIPS '68 (Spring)*, page 37, 1968.

[25] Turner Whitted. An improved illumination model for shaded display. *Communications of the ACM*, 23(6):343–349, 1980.

[26] David P. Dobkin, C. Bradford Barber, and Hannu Huhdanpaa. The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software*, 1996.

[27] D-Wave systems. Qbsolv, a decomposing solver, 2018.

[28] Florian Neukart and Sorin-Aurel Moraru. On Quantum Computers and Artificial Neural Networks. *Signal Processing Research*, 2(1), 2013.

[29] Florian Neukart and Sorin Aurel Morar. Operations on quantum physical artificial neural structures. In *Procedia Engineering*, volume 69, pages 1509–1517, 2014.

[30] Sven Eisenkrämer. Volkswagen trials quantum computers, 2017.

[31] Florian Neukart. Quantum physics and the biological brain. In *Reverse Engineering the Mind*, pages 221–229. 2017.

[32] Anna Levit, Daniel Crawford, Navid Ghadermarzy, Jaspreet S. Oberoi, Ehsan Zahedinejad, and Pooya Ronagh. Free energy-based reinforcement learning using a quantum processor. may 2017.

[33] Daniel Crawford, Anna Levit, Navid Ghadermarzy, Jaspreet S. Oberoi, and Pooya Ronagh. Reinforcement learning using quantum boltzmann machines. *arXiv preprint arXiv:1612.05695v2*, pages 1–17, 2016.

[34] T. Bäck and S. Khuri. An evolutionary heuristic for the maximum independent set problem. In *Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence*, pages 531–535. IEEE.

[35] D-Wave systems. Quantum computing: how D-Wave systems work, 2017.

[36] D. Korenkevych, Y. Xue, Z. Bian, F. Chudak, W. G. Macready, J. Rolfe, and E. Andriyash. Benchmarking quantum hardware for training of fully visible Boltzmann machines.

[37] Dmytro Korenkevych, Yanbo Xue, Zhengbing Bian, Fabian Chudak, William G. Macready, Jason Rolfe, and Evgeny Andriyash. Benchmarking Quantum Hardware for Training of Fully Visible Boltzmann Machines. *Frontiers in physics*, 2(5), nov 2016.

[38] T. Lanting, A. J. Przybysz, A. Yu Smirnov, F. M. Spedalieri, M. H. Amin, A. J. Berkley, R. Harris, F. Altomare, S. Boixo, P. Bunyk, N. Dickson, C. Enderud, J. P. Hilton, E. Hoskinson, M. W. Johnson, E. Ladizinsky, N. Ladizinsky, R. Neufeld, T. Oh, I. Perminov, C. Rich, M. C. Thom, E. Tolkacheva, S. Uchaikin, A. B. Wilson, and G. Rose. Entanglement in a quantum annealing processor. *Physical Review X*, 4(2), 2014.

[39] Martijn Van Otterlo and Marco Wiering. Reinforcement learning and Markov decision processes. *Reinforcement Learning*, pages 3–42, 2012.

[40] R.S. Sutton and A.G. Barto. *Reinforcement learning: an introduction*. MIT press, Cambridge, 1998.