Maastricht University

Maastricht Science Programme
Faculty of Science and Engineering

Exploring Quantum Annealing Algorithms for Enhanced Particle Track Reconstruction for LHCb VELO Detector

Internal Advisor:
Dr. Suzan du Pree

Supervisors:
Dr. Jacco de Vries
PhD Candidate Xenofon Chiotopoulos

Author:
Vita Stefanija
i6295302

Academic Year:
2024-2025

Word Count:9128

Abstract

Real-time event reconstruction is essential for deriving meaningful insights from particle collisions but it presents substantial computational challenges. Quantum annealing offers theoretical promise in navigating complex energy landscapes, potentially benefiting particle track reconstruction tasks such as those required for the LHCb VELO detector. This study evaluates the scalability and Gaussian noise resilience of an annealing-based algorithm consisting of a Hamiltonian generator function and three solver paradigms: Quantum annealing (QA), Simulated Annealing (SA), and Hybrid Annealing (HA). Scalability tests demonstrated that while the Hamiltonian generator efficiently handles matrices up to 5 layers and 65 particles, further refinement is necessary to manage the complexities and noise levels associated with VELO data. QA exhibited a scalability threshold of 5 layers and 7 particles with a mean accuracy of $(98.67 \pm 1.75)\%$, highlighting current hardware limitations. On the other hand, HA achieved 100% efficiency for 5 layers and 65 particles and exhibited strong noise resilience, underscoring its potential for realistic applications to LHCb data but requiring additional problem-specific optimizations. These results highlight the limitations of QA and the promise of Hybrid methods, emphasizing the importance of advanced optimization techniques for solvers and refinements of Hamiltonian matrix construction to advance annealing-based algorithms for particle track reconstruction.

Table of Contents

Contents

1 Introduction

High-energy physics (HEP) is essential for uncovering the fundamental laws of nature by studying matter and energy under extreme conditions. Beyond its theoretical significance, HEP drives technological innovation across diverse fields through advancements in particle detection and computational systems. CERN, at the forefront of high-energy physics exploration, operates the world's most powerful Large Hadron Collider (LHC). Among its experiments, LHCb investigates heavy quarks to explore the matter-antimatter asymmetry that dominates our universe [1]. The Vertex Locator (VELO) detector is a high-precision tracking device that records precise spatial information about particles as they collide and pass through its layers [2].

Accurate particle track reconstruction is critical for extracting meaningful insights from HEP collisions. As experimental facilities advance, robust and scalable data storage and processing systems are essential to handle the growing data volumes [3]. Quantum computing, particularly quantum annealing, is emerging as a promising alternative, leveraging quantum superposition and tunneling to explore multiple solutions simultaneously [4].

This study aims to quantify the current status of Quantum Annealing (QA), Simulated Annealing (SA), and Hybrid Annealing (HA) for the particle track reconstruction in the LHCb VELO detector. Scalability, accuracy, convergence, and resilience to Gaussian noise were examined.

The algorithm consists of two key components: a Hamiltonian generator function and three solver functions utilizing QA, SA, and HA, respectively. The Hamiltonian generator function is responsible for encoding the problem into a QUBO (Quadratic Unconstrained Binary Optimization) formulation, whereas the three solver functions represent distinct annealing approaches that minimize the QUBO. QA leverages quantum tunneling to navigate complex energy landscapes, potentially offering an advantage in escaping local minima. SA, a classical stochastic optimization technique, provides a baseline due to its well-established reliability in small-scale and noise-free environments. HA combines classical pre-processing with quantum annealing optimization, aiming to balance the advantages of both worlds.

The algorithm was tested on a toy model detector and a generated event, however, an attempt to simulate LHCb conditions was made by adding Gaussian noise into the spatial data of the event model.

2 Background

2.1 LHCb VELO Detector and the particle track problem

The Large Hadron Collider Beauty (LHCb) experiment at CERN is designed to investigate the decays of subatomic particles, such as quarks. One of its key components, the Vertex Locator (VELO) detector, provides high-precision measurements critical for particle tracking [5]. It consists of 52 silicon microstrip detectors (26 on each side of the beam axis) placed along the z-axis near the collision point (see Figure 1 [6]). These sensors measure the positions of particles passing through a 5 cm interaction region around the beamline (x=y=0). The VELO's unique geometry and alignment with the beamline naturally frame particle track reconstruction as a Quadratic Unconstrained Binary Optimization (QUBO) problem.

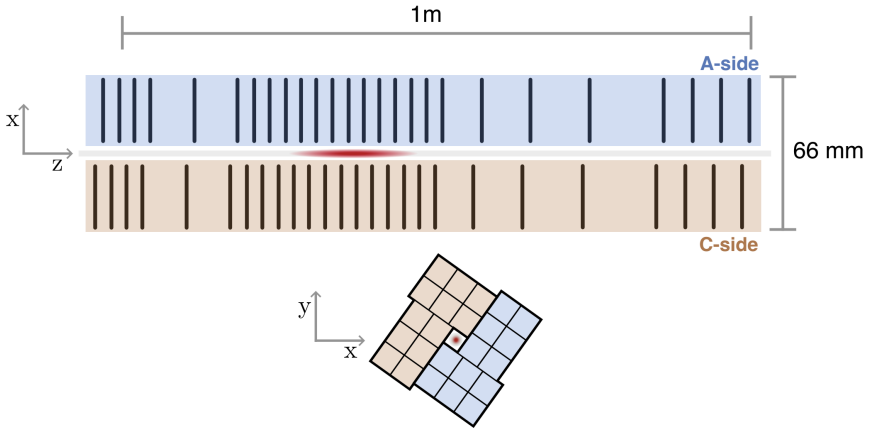


Figure 1: Schematic diagram of the VELO detector [6].

In QUBOs, the problem is encoded into an energy function where binary variables denote possible parts of a track. The goal is to minimize the energy function to identify the configuration of a system with the lowest energy. In this case, the lowest energy state corresponds to the optimal solution to the actual particle tracks [7].

Hits refer to the points of interaction where a particle crosses the detector layers, typically represented by x, y, and z coordinates. These points are generated as the particle moves through the detector, providing spatial data about its trajectory. Consecutive pairs of hits, referred to as doublets, indicate possible particle trajectories across adjacent layers. These connections, also termed segments, are mathematically represented as binary values: 1 if the segment belongs to the particle's true trajectory and 0 otherwise [8]. A collection of segments forms a possible path of an individual particle through the VELO. Track reconstruction aims to associate these hits with the correct collection of segments for each particle, reflecting its actual particle trajectory.

2.2 Annealing

"Annealing" is a term that originates from metallurgy, where it describes a thermal process that involves heating a material to a high temperature and cooling it down slowly, reducing internal stress and achieving a more stable structure, often improving strength, ductility, and resilience [9]. This concept was borrowed in computational optimization, where simulated annealing (SA) uses a similar gradual cooling metaphor (thermal fluctuations) to guide a system toward the optimal configuration. SA algorithms minimize an energy cost function by employing a wide-ranging and variable search approach, enabling them to bypass suboptimal points in the energy landscape and reach the global minimum [10]. As the "temperature" gradually lowers, the algorithm's exploration becomes more focused, centering around configurations that bring the system closer to the global or a nearly optimal solution [11].

In SA, thermal (classical) fluctuations are used to let the system transform from state to state, but it often struggles with local optima due to limited ability to escape energy barriers. Quantum Annealing (QA) overcomes this by utilizing quantum tunneling, allowing transitions through barriers that trap SA. Initially, strong quantum fluctuations explore the global phase space, analogous to high temperatures in SA, and are than gradually reduced to stabilize the system in the lowest-energy state [12].

Mathematically, quantum annealing can be described in terms of the Hamiltonian function, which encompasses the energy states of a system. For the D-wave quantum computer [7], the problem is formulated into a generalized Ising-like Hamiltonian [13]:

$$H(S) = -\frac{1}{2}\sum_{i,j} A_{ij} S_i S_j + \sum_i b_i S_i = -\frac{1}{2} S^T A S + b^T S \tag{1}$$

where, $S = \{S_1, S_2, \ldots, S_N\}$ is a state vector of doublets, A_{ij} is the interaction matrix that defines the coupling strengths and b is an external field vector (bias applied).

The variables S_i are directly mapped onto the qubits of D-wave's QA hardware. The probability of the qubit falling into the 0 or 1 state can be controlled by applying an external magnetic field to bias the qubit's final classical state. The links between qubits (coupling strengths) can also be adjusted based on the specific problem. Together, couplings and bias define means for formulating the track problem for the D-wave quantum computer [7].

Once the problem is set up for D-wave, quantum annealing process begins. The system starts in the lowest-energy eigenstate of the initial Hamiltonian, where all qubits are in the superposition state. As annealing progresses, the influence of the initial Hamiltonian decreases, while the problem Hamiltonian containing qubit biases and couplings, Equation 1, takes over [7]. Ideally, if all adiabatic conditions are satisfied [12], the system is guided into the minimum-energy state of the problem Hamiltonian. By the end of the process, all qubits transition to classical states, producing a binary-valued solution list that, in the context of track reconstruction, indicates the doublets included in the particle tracks [14]. While QA is a powerful technique for exploring global minima, combining it with classical optimization methods in a hybrid approach significantly enhances its scalability and efficiency [15]. QA D-wave computers have a limited number of qubits [7], and thus they face challenges in scaling up to handle high-dimensional problems, such as particle track reconstruction. The LeapHybridSampler along with other hybrid solvers, addresses this by partitioning the larger problem into smaller components that fit within the quantum annealer's capacity. Using the divide-and-conquer method [16], each sub-problem is individually solved with QA, while classical algorithms refine and stabilize the solutions. The hybrid solver thus combines the quantum processing unit for rapid, probabilistic exploration with classical processors for deterministic optimization, which manages the computational complexity.

2.3 Annealing for the Track Problem

The increasing complexity and data volume in modern colliders necessitate more efficient and scalable algorithms for track reconstruction [3]. Quantum annealing-based approaches have recently emerged as promising alternatives to the current state-of-the-art methods, such as Search by Triplets and Graph Neural Networks (GNNs) [17, 18].

The potential scaling benefits of QA compared to classical SA in the context of optimization problems relevant to HEP were examined [19]. Evidence shows that the D-Wave quantum annealer achieves a scaling advantage for certain problem instances exceeding 2000 qubits, providing the first validation of quantum annealing surpassing SA with 95 % confidence. However, compared to discrete-time simulated quantum annealing (SQA), the QA failed to achieve a quantum speedup, as SQA exhibited superior scaling by leveraging quantum tunneling-like mechanisms to overcome energy barriers [19]. The complex performance dynamics of QA were highlighted, showcasing its advantage over classical SA for certain problems while pointing to the need for further advancements to achieve broader quantum speedups in practical applications.

A. Zlokapa et al. address computational challenges posed by track reconstruction in current and future high-energy collider experiments with annealing [20]. Track reconstruction was reformulated as a QUBO problem, enabling the application of quantum annealing-inspired algorithms. The geometric Denby-Peterson (Hopfield) network method [21, 22] was adopted to fit the QA framework. The D-Wave 2X quantum annealer was tested against SA in dense track simulations, showing comparable or superior reconstruction efficiency and purity with significantly faster computation times. Heuristic pre-processing methods were also developed to embed large-scale tracking problems onto current QA hardware. These results demonstrate the potential of quantum annealing-inspired algorithms for scalable, high-speed particle tracking.

D. Magano et al. [4] also aimed to address computational challenges in current and future high-energy collider experiments. They explore the application of quantum search algorithms as means to reduce the computational complexity compared to classical approaches like SA. Findings suggest that integrating QA with existing methods could offer significant improvements in processing speed and resource utilization. A recent example is the successful application of simulated bifurcation algorithms to the tracking problem, showing reconstruction efficiency and purity comparable to or surpassing traditional SA methods, with computation times reduced by up to four orders of magnitude [23].

The application of annealing-based methods (classical, quantum, and hybrid) to particle track reconstruction has shown considerable promise; however, quantum speedup, particularly with QA alone, remains unproven and elusive.

2.4 The Toy model

Detector and Event Generator

The detector is modeled as a collection of modules, each defined by its z-position and physical dimensions (L_x, L_y). This modular representation captures the layered structure of detectors like LHCb's VELO, where hits are recorded as particles pass through. Hits are modeled with spatial coordinates (x, y, z) and are associated with specific modules, forming primary data for reconstruction. The units of a detector are arbitrary and can be adjusted based on the needs. For this study, the layer gap was set to 1 unit.

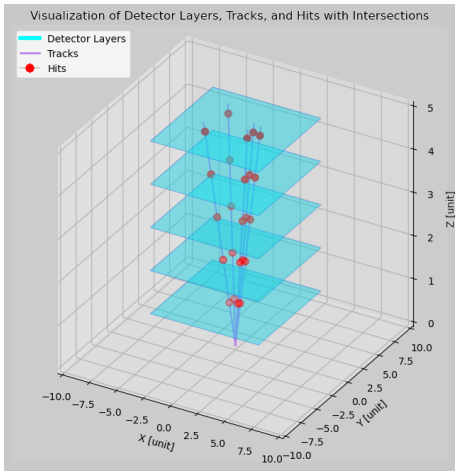


Figure 2: Toy Detector Layout and Generated Event with 5 particles and 5 layers.

Tracks represent particle trajectories, parameterized by their primary vertex and angular components ϕ and θ, which describe their orientation in 3D space:

$$\phi = \arctan\left(\frac{y}{x}\right), \quad \theta = \arccos\left(\frac{z}{\sqrt{x^2+y^2+z^2}}\right). \tag{2}$$

The trajectory of a particle is further described by its velocity vector $\vec{v} = (v_x, v_y, v_z)$, derived from azimuthal angle ϕ and polar angle θ chosen from ranges $[0, 2\pi]$ and $[0, \pi/10]$, respectively. These ranges model the spatial distribution of particles from collisions.

A hit is recorded if x_{hit} and y_{hit} are within the module bounds (L_x, L_y). The event generator then compiles all hits into an Event object, encapsulating the detector's modules, tracks, and global hits. This object serves as input to the Hamiltonian generator, enabling the encoding of the particle reconstruction problem into a QUBO matrix.
This design enables realistic simulations of detector events, ensuring geometrically consistent hits and tracks with the detector layout while modeling diverse particle trajectories.

Hamiltonian Generator

The Hamiltonian matrix is generated based on the adopted Denby-Peterson approach [6]. The matrix is built such that it favors the reconstruction of smooth, aligned, and long tracks while avoiding non-physical track bifurcations. Let S_{ab} be the oriented segment stemming from hit a to hit b. The angle θ_{abc} lies between S_{ab} and S_{bc}. Weights (α, β, γ) are adjustable parameters designed to reward favorable alignments or penalize unfavorable ones. The adapted Hamiltonian can then be expressed as follows:

$$H_{\text{DP}} = \alpha H_{\text{ang}}(S, \epsilon) + \beta H_{\text{bif}}(S) + \gamma H_{\text{inh}}(S) \tag{3}$$

where

$$H_{\text{ang}}(S, \epsilon) = -\frac{1}{2} \sum_{a,b,c} f(\theta_{abc}, \epsilon) S_{ab} S_{bc} \tag{4}$$

with

$$f(\theta_{abc}, \epsilon) = \begin{cases} 1 & \text{if } \cos\theta_{abc} \geq 1 - \epsilon \\ 0 & \text{otherwise} \end{cases}$$

The Angular term enforces alignment between consecutive segments by promoting near-parallel configurations. The alignment between segment pairs is encouraged if the cosine similarity between vectors is near 1.

$$H_{\text{bif}}(S) = \frac{1}{2}\left(\sum_{b \neq c} S_{ab} S_{ac} + \sum_{a \neq c} S_{ab} S_{cb}\right) \tag{5}$$

The bifurcation term penalizes bifurcations, ensuring that each hit belongs to a unique trajectory without branching. A penalty is introduced for segment pairs that share a starting or ending hit but differ in direction, which discourages configurations where a hit is shared by multiple segments.

The inhibitory term is specific to the toy model and penalizes interactions between incompatible or overlapping segments, weighted with an adjustable penalty γ.

3 Methodology

This project aimed to evaluate and enhance the performance of the toy algorithm, focusing on its two components: the Hamiltonian Generator, responsible for formulating and constructing the optimization problem, and three distinct QUBO solvers, tasked with minimizing the Hamiltonian to identify the system's lowest energy state, representing the optimal solution. Various classical optimization methods and Python-specific techniques were employed to enhance scalability and resilience in problem formulation. Furthermore, QA, SA, and HA solvers were examined for scalability, accuracy, convergence, and noise resilience. This section details the optimization of the Hamiltonian generator, the implementation of QUBO solvers, and the performance evaluation methods.

3.1 Optimization of Hamiltonian Generator

The original Hamiltonian generator (refer to Appendix) lacks scalability, with its complexity increasing disproportionately with input size, making it unsuitable for real LHCb data and underscoring the need for optimization to support realistic particle track reconstruction [5].

Data Structures and Sparse Representations

One of the key optimization techniques applied was the use of efficient data structures and algorithms [24]. The function employs appropriate sparse matrix formats instead of dense NumPy arrays, specifically the List of Lists (*lil_matrix*) for incremental construction and the Compressed Sparse Column (*csc_matrix*) format for efficient arithmetic operations once the matrix is assembled. The transition to sparse representations significantly enhanced scalability by reducing memory consumption and accelerating matrix operations through storing only non-zero elements and their positions [25].

Pre-computation and Data Access Patterns

Pre-computing segment vectors and norms enabled sequential access within optimized data structures. Cache utilization was further enhanced by processing pre-computed data in parallel, reducing redundant memory accesses and improving spatial locality [26]. Instead of iterating over every segment pair directly in nested loops, computations were modularized into angular and bifurcation checks, with results aggregated into sparse matrices. Efficient data aggregation using modularized computations minimized overhead and enhanced performance [27, 28].

Parallelization

Parallelization was implemented using the Joblib library [29]. The computational bottlenecks of the function, particularly the angular and bifurcation consistency checks, were restructured to enable concurrent execution of operations on disjoint segment pairs. Eliminating dependencies between computations allowed efficient parallel processing of large datasets, aligning with best practices in high-performance computing [30]. Parallelization also ensured scalability across a wide range of hardware configurations, which is essential for high-dimensional datasets common in particle physics [26].

Symmetry and Redundancy Reduction

The inherent symmetry of the Hamiltonian matrix was recognized and exploited, reducing redundant calculations by computing only the upper triangle of the matrix. This optimization was implemented in Python using efficient looping constructs and by altering the computation of Hamiltonian terms while maintaining the overall logic of the original code [31]. The computational complexity was reduced, leading to a direct improvement in execution time.

On-the-Fly Computations and Memory Optimization

On-the-fly computation methods were incorporated to optimize memory usage. Rather than storing large, sparse matrices, angular and bifurcation consistency checks were calculated on-the-fly during iteration, introducing storage burden only when necessary. This dynamic approach eliminated the need for excessive memory allocation, substantially reducing the memory footprint. By combining caching with on-the-fly computations, the optimized Hamiltonian generator can handle significantly larger datasets that would have been impractical with the original implementation [32].

Streamlined Conditional Checks

The efficiency of conditional checks was improved by simplifying the logic and reducing the number of branching instructions. Conditions were evaluated using integer arithmetic and logical operators, avoiding complex logic. This streamlined approach minimizes the number of branches the CPU needs to handle, enhancing branch prediction accuracy and reducing pipeline stalls. This implementation offers faster execution with fewer delays in instruction flow [33]

3.2 Construction of QUBO solvers

The second part of the annealing algorithm includes three functions— QA, SA, and HA-based, specifically developed to minimize the Hamiltonian matrix. Although all solvers were built to process large, sparse matrices, they employed different sampling mechanisms and exhibited varying computational trade-offs. Each function harnesses different computational paradigms, however, they have an equivalent overall structure. Matrix A is expected to be large and sparse [34], thus all solvers begin by ensuring the QUBO matrix is in a sparse format (csc_matrix), suitable for handling large datasets efficiently [25]. A Binary Quadratic Model (BQM) is then constructed across all solvers ensuring that the problem representation remains consistent. Additionally, retrieving and interpreting the best sample from the sampler's response is uniform, facilitating straightforward comparison of results. All functions return the solution as a NumPy array, standardizing the output for further processing or analysis. This consistent framework facilitated a reliable comparison across all solvers.

The QA solver utilized D-Wave's quantum annealing technology to solve QUBO matrices, leveraging quantum tunneling to escape local minima. The solver employed the `DWaveSampler`, encapsulated within an `EmbeddingComposite`, to manage the embedding of the problem onto D-Wave's quantum hardware. A total of 100 samples were collected to identify the optimal solution.

In contrast, the HA solver employed D-Wave's Leap Hybrid Sampler, which integrates quantum and classical computational resources to solve QUBO problems. The BQM was passed to the `LeapHybridSampler`, which dynamically allocated computational tasks between quantum and classical processors [35]. The HA solver combined quantum optimization with classical preprocessing, offering a hybrid approach that capitalized on the strengths of both paradigms.

The third function, SA solver, adopts a purely classical approach by utilizing the SA Sampler from the dimod library. SA heuristically approached the equilibrium distribution by performing updates at a sequence of decreasing temperatures [36]. The `SimulatedAnnealingSampler` was used with 100 reads, serving as a reliable baseline.

3.3 Evaluation of the algorithm

In order to evaluate the annealing algorithm in terms of its scalability, accuracy, robustness, and noise resilience, several tests were constructed. Comprehensive code for all tests can be found in the Appendix.

Scalability

The scalability of the Hamiltonian generator was tested by systematically comparing optimized and non-optimized implementations across varying matrix sizes. Exact equality checks were performed to ensure the accuracy of the optimized implementation. Additionally, the algorithm was safeguarded with try-except blocks, ensuring that scalability testing could continue independently if one function encountered a failure. Data collection was terminated either upon encountering a Memory Error or exceeding 3600 seconds. Notably, scalability results may vary depending on computational, hardware-specific resources.

The scalability of SA, QA, and HA QUBO solvers was tested using randomly generated binary sparse matrices. A similar approach was employed for testing solvers, with try-except blocks implemented to collect results until an embedding error, memory error, or time limit of 3600 seconds was encountered. The density of real LHCb VELO data is generally assumed to follow the following equation [6]:

$$s = \frac{N_0}{n^2}. \tag{6}$$

where n is the size of a matrix, N_0 are non-zero elements and s is the sparsity.

The initial scalability testing was conducted with an assumed density of 1%, while this equation was only applied during the solution testing phase. By starting with a generic density, scalability testing is simplified and not biased by specific assumptions or domain constraints. A uniform density allows solvers to be benchmarked for general performance before narrowing it down to the specific needs of a tracking problem. This approach ensures that solvers are robust and adaptable.

The scalability is expressed in terms of the matrix size(N), where the conversion to the number of particles (N_{particle}) and the number of layers in the detector (N_{layer}) is:

$$N = N_{\text{particle}}^2 \times (N_{\text{layer}} - 1) \tag{7}$$

Accuracy of QUBO solutions

Accuracy testing of QA, SA, and HA solvers on the toy model was performed. The Hamiltonian matrix is assumed to effectively represent the tracking problem, as it is based on a simplified toy model devoid of noise and real-world complexities. This approach isolates the testing of the QUBO solvers. The following standardized accuracy matrices were used to evaluate the performance of the algorithm [6].

Hit efficiency, which checks the completeness of the solver's solution by seeing how many hits (non-zero values) from the true solution are also present in the solver's solution. This metric is sensitive to missing hits in the solver's output and can be computed using equation 8.

$$e_{\text{eff}} = \frac{N_{\text{correct hit}}}{N_{\text{gen hit}}} \tag{8}$$

where $N_{\text{correct hit}}$ is the number of hits in the solver's solution that match the hits in the true solution, and $N_{\text{gen hit}}$ is the total number of hits in the true solution.

Fake Rate focuses on false positives in the solver's solution by determining how many hits in the solver's output don't align with the true solution. It evaluates the solver's ability to avoid false positives.

$$f_{\text{fake}} = \frac{N_{\text{fake hit}}}{N_{\text{all hit}}} \tag{9}$$

where $N_{\text{fake hit}} = N_{\text{all hit}} - N_{\text{correct hit}}$, with $N_{\text{fake hit}}$ = false hits , $N_{\text{all hit}}$ = total number of hits , and $N_{\text{correct hit}}$ = correctly identified hits

Hit purity is the fraction of hits in a reconstructed track that belong to a single true track, particularly sensitive to the quality of the hits within a track.

$$e_{\text{pure}} = \frac{N_{\text{correct hit}}}{N_{\text{all hit}}} \tag{10}$$

where $N_{\text{all hit}} = N_{\text{all track}}$ is the total number of hits in the solver's solution, and $N_{\text{correct hit}}$ refers to the correctly assigned hits in the solver's solution that match the hits in the true solution.

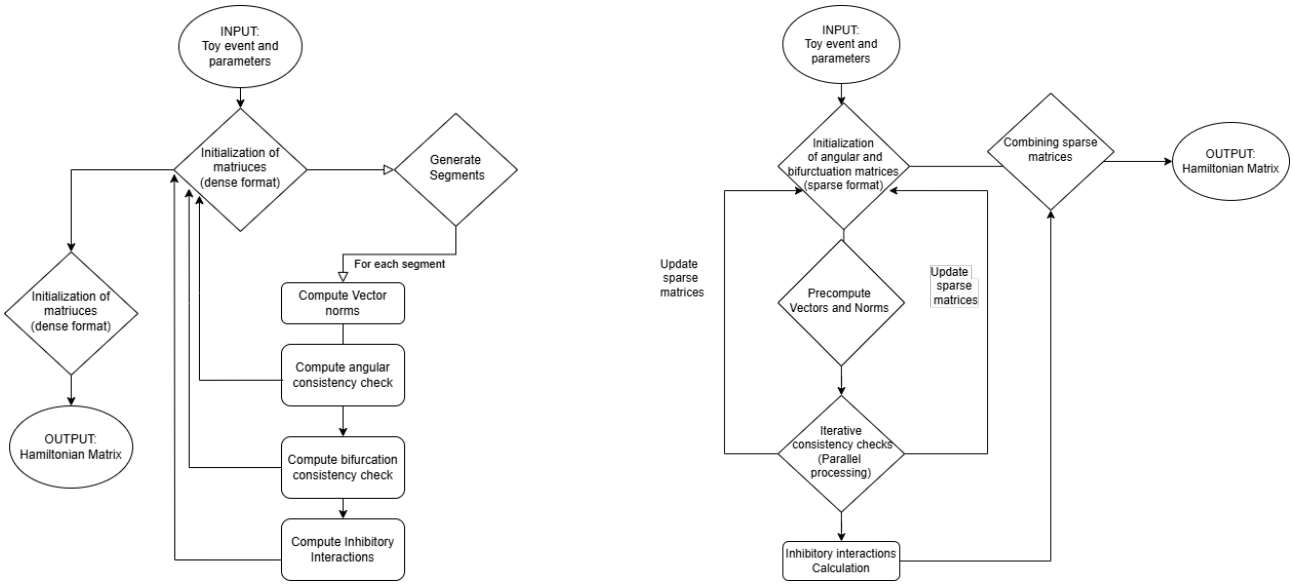
Convergence rates and Robustness testing

A systematic multi-run methodology was employed to evaluate the convergence of QA and SA solvers, focusing on their accuracy and stability across iterative steps. A Python function was developed to collect the accuracy of each solver's solution by varying the number of reads. Across a range of iterations, beginning from a defined minimum step size and progressing to the maximum iteration count, convergence was examined. This approach enabled a comparative analysis of QA and SA solver's performance over iterations, focusing on the convergence characteristics of the QA algorithm.

Furthermore, the complete algorithm was tested in noisy conditions to simulate real LHCb VELO detector conditions. Noise in real-world data typically originates from a combination of independent factors, including electronic interference, thermal fluctuations, and manufacturing imperfections, among others [34]. The Central Limit Theorem explains why these combined effects typically result in Gaussian noise [37]. Even though real data consists of various types of noise, it was found that for small-scale deviations, such as fine measurement errors, Gaussian noise provides a good approximation of the uncertainties in real hit positions [38].
Thus, Gaussian noise was introduced into the positions of hits generated with the toy event model, before the event was passed on to the Hamiltonian generator. The noise measurement unit was normalized in respect to the detector size, which for this testing was set to 10 units. Respective normalized noise values are thus given in arbitrary units of $\frac{Noisesize}{DetectorSize}$.

4 Results & Discussion

4.1 Scalability of Hamiltonian Generator



(a) Non-Optimized Flowchart (b) Optimized Flowchart

Figure 3: Comparison of Non-Optimized and Optimized Hamiltonian Generator Functions.

Logic flow: Optimized vs Non-Optimized

The flowcharts in Figure 3 illustrate the logic of the Optimized and non-optimized Hamiltonian generators. Despite their shared goal of encoding segment interactions through angular, bifurcation, and inhibitory consistency checks, the two implementations employ different efficiency and scalability approaches. The Non-Optimized function calculates vectors, norms, and consistency checks redundantly within a nested loop for each pair of segments, resulting in high computational overhead $O(N^2)$ and poor scalability. In contrast, the Optimized function eliminates redundancy by precomputing all vectors and norms, storing them in arrays for reuse, and utilizing parallel processing for pairwise checks. The inhibitory interaction is calculated after a parallel check for angular and bifurcation consistency. While both functions output Hamiltonian matrices combining angular, bifurcation, and inhibitory interactions, the computational demands that scaling brings vary for the two implementations.

The comparison between the scalability of the optimized and non-optimized Hamiltonian generator functions is displayed in Figure 4. The measured runtime is not fully reliable due to external variability, such as power source status, background processes, and resource allocation. Thus, the findings here serve as an indicative demonstration rather than deterministic results.

Overall, the plots highlight significant improvements in both computational time and memory usage, reflecting the efficiency of the optimized implementation.

Execution Time

Figure 4 illustrates the execution time scaling for the optimized and non-optimized implementations. Even though for smaller matrix sizes (< 200 NxN), the execution time difference is insignificant, for larger matrix sizes, such as 35×35, the non-optimized function fails due to time constraints, while the optimized implementation scales successfully up to 12000 NxN (5 layers and 65 particles).

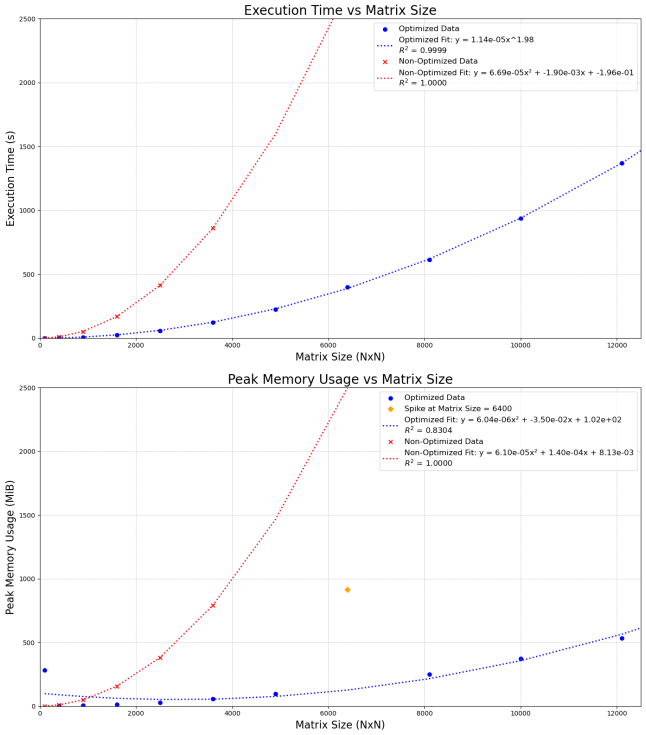


Figure 4: Optimized and Non-optimized Hamiltonian Time and Memory usage comparison with increased input size.

Polynomial fitting reveals a near-quadratic increase for optimized implementation; $y = 1.14 \times 10^{-5} x^{1.98}$ ($R^2 = 0.9999$). The non-optimized execution time follows $y = 6.69 \times 10^{-5} x^2 - 1.90 \times 10^{-3} x - 0.196$ ($R^2 = 1.0000$), exhibiting pure quadratic growth. For instance, at 4000 matrix size, the optimized execution time predicted by the fit is 135.8 seconds compared to 1069.2 seconds for the non-optimized fit, a performance improvement factor of 7.9×. The relative growth rate can be quantified as:

$$\text{Relative Growth Rate} = \frac{T_{\text{non-opt}}(n)}{T_{\text{opt}}(n)} = \frac{6.69 \times 10^{-5} n^2 - 1.90 \times 10^{-3} n - 0.196}{1.14 \times 10^{-5} n^{1.98}}$$

As n increases, the polynomial scaling difference between $O(n^{1.98})$ and $O(n^2)$ widens, with the non-optimized function growing increasingly faster due to the n^2 term dominating.

Among the optimization techniques applied to the Hamiltonian Generator, the improvement from pure to near quadratic growth can be attributed to two key optimization strategies, parallelization and sparse matrix utilization. The bottlenecks of Angular and Bifurcation condition checks within the Hamiltonian function were minimized by integrating parallelization strategy, which significantly reduces execution time [29]. Sequentially iterating through the upper triangle of the QUBO matrix is computationally demanding, thus distributing the workload across multiple CPU cores effectively streamlines matrix construction.

The use of sparse matrices furthermore reduced execution time. Representing the angular and bifurcation matrices as sparse matrices minimizes unnecessary memory allocation and accelerates matrix operations [39]. Sparse matrices are particularly effective for this problem due to the sparse nature of the QUBO track problem.

Memory usage

In assessing Peak Memory Usage, both functions exhibited quadratic growth with increasing matrix size. The Non-Optimized Function's Peak Memory Usage was best characterized by the quadratic fit $y = 6.10 \times 10^{-5}x^2 + 1.40 \times 10^{-4}x + 8.13 \times 10^{-3}$, with an excellent fit accuracy of $R^2 = 1.0000$. This reflects a direct proportionality to the matrix size squared, leading to aggressive scaling. In contrast, the Optimized Function followed the quadratic fit $y = 6.04 \times 10^{-6}x^2 - 3.50 \times 10^{-2}x + 1.02 \times 10^{2}$, with a weaker fit accuracy of $R^2 = 0.8304$. To ensure the accuracy of the fit, the anomaly observed at a matrix size of 6400 NxN was excluded from the fitting analysis. The exclusion of this data point is justified as it represents an anomaly that is discussed in detail later, with several potential causes identified.

Comparing the leading coefficients of x^2, the non-optimized implementation grows approximately 10.1× faster than the optimized implementation (6.10×10^{-5} vs. 6.04×10^{-6}). This difference in scaling is evident in the graph, where the non-optimized memory usage sharply increases with matrix size, exceeding 2000 MiB by 6400 NxN, while the optimized implementation maintains a much slower growth rate and lower memory usage. The much smaller leading coefficient for the optimized implementation demonstrates the potential for larger matrix sizes.

The spike at N = 6400

Although the optimized implementation exhibits strong scalability overall, the observed spike in memory usage for the optimized function portrays a temporary anomaly, likely associated with the behavior of sparse matrix operations and temporary dense matrices in the parallelization step.

The optimized implementation constructs matrices incrementally using the List of Lists format for insertion and after assembly, Compressed Sparse Column format for efficient arithmetic operations. Sparse matrix conversion processes, especially for large matrices require temporary duplication of data [25]. When converting to *csc* format, all the nonzero elements are reorganized into a new data structure optimized for computations. Temporarily duplication of the matrix's nonzero entries and index mappings takes place, which can cause a temporary surge in memory usage. Similar behavior has been reported in sparse matrix libraries such as SciPy, justifying temporarily increased memory consumption [40]. Furthermore, Joblib.Parallel, while significantly improving execution time, can also temporarily increase memory usage because each parallel thread requires its own copy of certain data/temporary arrays. Lastly, processes working on different subsets of the matrix create temporary dense arrays to store intermediate results. This could explain the memory spike, as large dense arrays consume substantially more memory than their sparse counterparts. Thus, the memory overhead can be also attributed to thread-local or process-local data copies [41].

To further investigate this anomaly, a higher-capacity computer should be used to test for its recurrence and accurately analyze its behavior.

Overall, the optimized function significantly improves computational performance while ensuring feasibility for medium-scale HEP applications, successfully handling matrices up to 12000 NxN, where scalability can be enhanced with the use of higher-performance hardware.

4.2 QUBO solvers

QUBO solving functions, QA, SA, and HA solvers, were analyzed in terms of their scalability and accuracy. The scalability of the functions was evaluated in terms of memory usage and execution times. Furthermore, the accuracy of solutions was tested using the toy model and true solutions of tracks that can be extracted from it.

4.2.1 Scalability

Figure 5 illustrates the scalability of QA solver in terms of execution time and memory usage against the increasing matrix size (NxN).

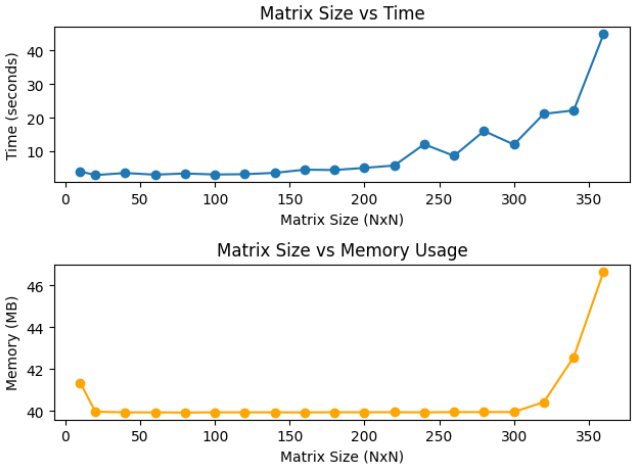


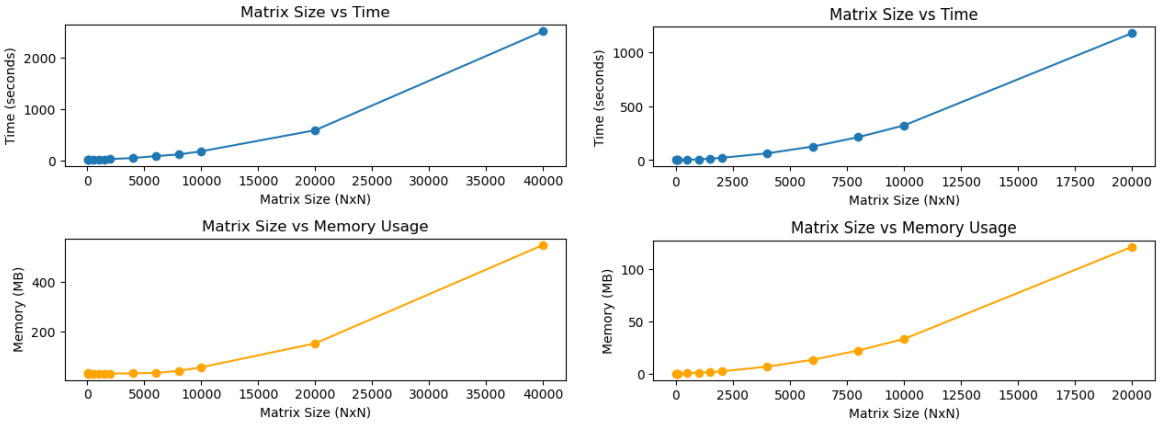
Figure 5: Scaling of Quantum Annealing QUBO solver.

QA solver Execution Time

The results highlight significant challenges associated with QA treating large problem sizes. The execution time remains relatively low, under 5 seconds for the majority of the tested sizes, however, as N approaches 300, execution time spikes beyond 40 seconds. This pattern can be attributed to factors inherent to QA solvers, such as embedding complexity, annealing time and connectivity constraints [42]. QA involves gradually transitioning the system from a high-energy state to a low-energy state that represents the optimal solution. Larger QUBO problems create a more rugged energy landscape with numerous local minima, thus longer annealing schedules are required for navigating such landscapes effectively. This contributes to the observed increase in execution time as the matrix size increases [43]. However, the spike with execution time from 300 NxN onwards is rather exponential. Besides the increased annealing time, the embedding process likely influenced the rapid increase. Larger matrices render more complex configurations, resulting in mapping complexity growing rapidly [7]. This spike highlights embedding as a bottleneck in QA scalability [7]. The sharp increase in execution time for large matrices along with the stepped pattern for $N > 240$ illustrates the interaction between problem structure, embedding complexity, and hardware restrictions for QA [7].

QA solver Memory usage

The memory usage graph reveals a relatively constant trend for small matrices but begins to rise sharply for matrices larger than $N = 300$. For smaller matrices the memory consumption reaches about 40MB whereas for $N = 350$, approximately 46 MB are used. QA solvers must store binary variables representing the problem, as well as interactions between variables. Sparse matrices require less memory as fewer interactions are encoded [25]. Furthermore, the embedding of variables onto the QPU requires additional data structures to manage logical-to-physical qubit mapping and coupler connections, leading to increased memory consumption. Regardless, QA remains memory-efficient compared to classical solvers, since QA directly encodes the problem onto the QPU where there is no need for memory-dense intermediate steps [7].



(a) HA solver Time and Memory usage

(b) Simulated Annealing solver Time and Memory usage

Figure 6: Scalability of HA and SA solvers.

HA and SA solvers Execution Time

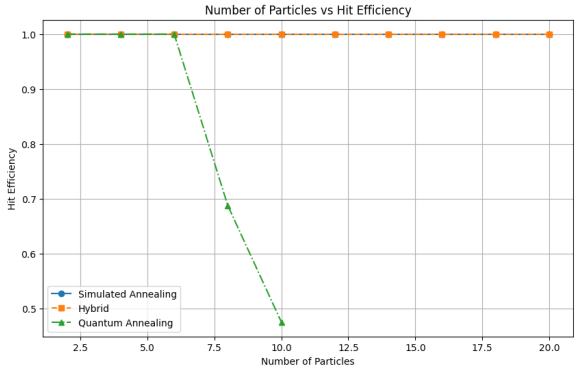
The runtime scaling of the HA Solver (Figure 6a) demonstrates superlinear growth, surpassing 500 seconds for a matrix of size 20,000 N×N. This trend aligns with known challenges of hybrid quantum-classical solvers, which involve significant classical overhead for embedding, pre-processing, and decoding of solutions [7, 44]. Latency and computational delays arise from embedding large QUBO problems onto quantum hardware [45]. Regardless, HA solver could tackle matrices up to 40, 000 NxN.

Simulated Annealing (Figure 6b) exhibits an even steeper runtime increase, exceeding 1,000 seconds for matrices of the same size. This exponential scaling reflects SA's theoretical runtime complexity, which heavily depends on the cooling schedule [46]. Slower decay schedules, essential for avoiding trapping the solution in local minima, entail a trade-off with computational time, as also observed in [45]. Consequently, SA is efficient for matrices with dimensions below 5,000 N×N, but becomes impractical for larger problems. The HA Solver demonstrates superior runtime performance for larger problems, leveraging quantum-classical mitigation of computational burdens. However, its advantage is offset by the classical overhead for pre- and post-processing tasks [47].

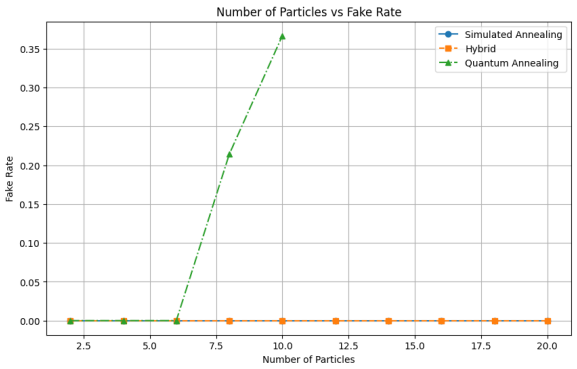
HA and SA solvers Memory Usage

The HA Solver's memory usage scales quadratically with matrix size, exceeding 150 MB for the largest matrices. This behavior reflects the construction of BQMs for quantum sampling, which store variables and interactions explicitly [7, 44]. While the sparse representation of A reduces initial memory usage, the embedding process and interaction encoding introduce quadratic growth. In contrast, SA demonstrates near-linear memory scaling, remaining below 100 MB even for the largest matrices. This efficiency arises from SA's lightweight computational demands, which primarily involve storing the problem matrix, solution vector, and intermediate steps during optimization [11]. The memory efficiency of SA makes it suitable for resource-constrained applications, though its runtime limitations remain a drawback.

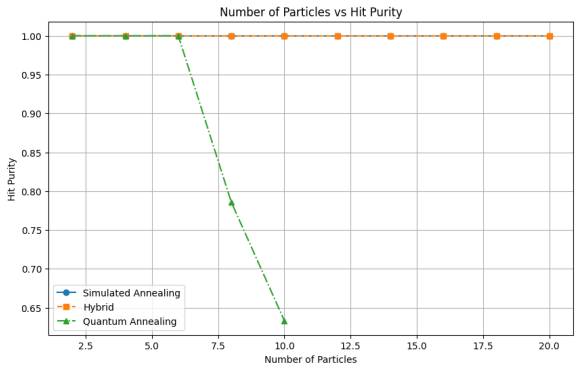
4.2.2 Accuracy of Solutions



(a) Hit Efficiency



(b) Fake Rates



(c) Hit Purity

Figure 6 provides a detailed comparison of the QUBO solver's accuracy of solutions.
Hit Efficiency: Measures the fraction of correctly identified hits.
Fake Rates: Represents incorrect hits identified by the solver.
Hit Purity: Indicates the proportion of correct hits among all identified hits.

Figure 7: Matrices of accuracy for QA,SA and HA solvers.

The accuracy tests were performed with the number of layers set to 5, while varying the number of particles and thus the matrix size, which can be computed using Equation 7. From Figure 7, it is evident that both SA and HA solvers achieve perfect hit efficiency, hit purity, and optimal fake rates for particle counts ranging from 1 to 20, demonstrating robustness and scalability for matrix sizes up to 1600 N×N [35, 46, 48].

In contrast, QA exhibits a dramatic decline in hit efficiency and hit purity, as visible from Figure 7(a) and (c), respectively. As the number of particles increases beyond 7 (196 N×N Matrix size), hit efficiency as well as hit purity drop by more than 50%. Initially maintaining an efficiency close to 1, the QA solver's performance deteriorates to $e_{\text{eff}} = 0.5$ and $e_{\text{pure}} = 0.65$ at 10 particles (400 N×N Matrix Size), while the fake rate rises sharply to 0.35, reflecting the generation of spurious hits.

These results align with known limitations of quantum annealers [45, 49, 50]. Besides the embedding complexity, QA is further constrained by the limited connectivity of current hardware architectures [7, 50]. Misalignment in QA-reconstructed tracks across standardized matrices reinforces previously identified scalability challenges already discussed in Section 4.2.1. Regardless, it is noteworthy that D-Wave Systems made significant advancements in qubit connectivity, increasing by 233 % between 2015 and 2024 [7]. Such progress highlights the potential for future applicability of QA to track reconstruction as hardware continues to evolve.

The HA solver effectively addresses many challenges by combining classical pre-processing with quantum optimization. Pre-processing reduces problem size and complexity, enabling more efficient embedding on quantum hardware. This allows HA solvers to maintain high hit efficiency and purity, significantly outperforming QA as a standalone solver, even at larger matrix sizes [35].

Table 1 demonstrates the HA solver's strong performance up to matrices of 19600 N×N. In noise-free conditions, both HA and SA solvers achieve optimal solutions with 100% hit efficiency and purity, underscoring the importance of accurate Hamiltonian construction. Beyond this matrix size, resource demands grow substantially, necessitating further testing to identify scalability limits. Advanced strategies, such as parallelized implementations, have shown potential to reduce runtime for larger problem sizes [24, 40]. Furthermore, optimizing classical pre-processing tailored to track reconstruction could additionally enhance HA solver scalability, providing a pathway for integrating QA into HEP.

Table 1: SA and HA Solver Accuracy Results Grouped by Increasing Matrix Size

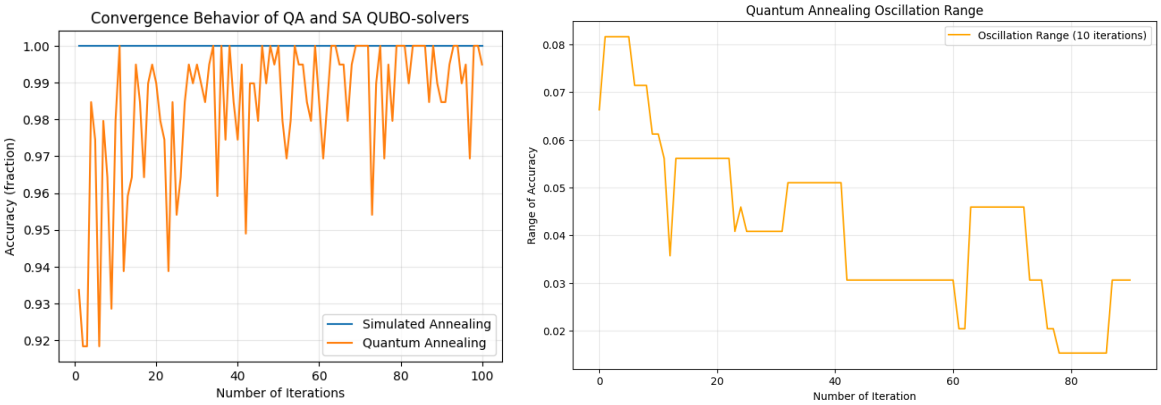
Matrix Size	Simulated Annealing			HA		
	Hit Purity	Hit Efficiency	Fake Rates	Hit Purity	Hit Efficiency	Fake Rates
100	1.00	1.00	0.00	1.00	1.00	0.00
400	1.00	1.00	0.00	1.00	1.00	0.00
900	1.00	1.00	0.00	1.00	1.00	0.00
1600	1.00	1.00	0.00	1.00	1.00	0.00
3600	1.00	1.00	0.00	1.00	1.00	0.00
6400	1.00	1.00	0.00	1.00	1.00	0.00
10000	1.00	1.00	0.00	1.00	1.00	0.00
14400	1.00	1.00	0.00	1.00	1.00	0.00
19600	1.00	1.00	0.00	1.00	1.00	0.00

4.2.3 Convergence behavior

Figure 8 displays the convergence behavior of QA and SA solvers. Accuracy, effectively expressed as the percentage of alignment with true solutions was used as means to analyze the convergence. A series of 100 iterations, with the number of reads increasing incrementally, was conducted to analyze this behavior. Matrix A was constructed using 7 particles and 5 layers, resulting in a size of 196 NxN matrix. While this represents a larger dataset in the context of QA [7], it poses no significant challenge for SA [11]. This setup provides a strong test for understanding the distinct behavior of the QA solver with problem sizes considered large relative to its computational capacity and operational framework. SA solver's results serve as a baseline.

SA solver's convergence

As Figure 8 illustrates, the SA solver's performance was characterized by complete stability, with a constant accuracy of 1 across all 100 iterations. This result demonstrates the algorithm's robustness in maintaining optimal solutions once convergence is achieved, even under well-defined conditions and smaller QUBO matrix sizes [48]. This result was expected given previous solution testing and the deterministic nature of SA, highlighting its strength in non-nosy environments.



(a) Convergence of SA and QA solvers (b) Oscillation range of QA solver

Figure 8: Convergence analysis with iterations in 1 step increase

QA solver's convergence

In contrast, the QA solver exhibited fluctuations in accuracy during the process but ultimately converged to a high mean accuracy of 0.9834. The fluctuations and eventual convergence reflect the inherently probabilistic nature of QA such as quantum tunneling [43]. The oscillatory behavior of the QA solver was statistically evaluated in 10-iteration windows. The accuracy fluctuated between a minimum of 91.84% and a maximum of 100%, with a mean of 0.9834, reflecting the solver's ability to achieve near-optimal solutions over 100 iterations. The standard deviation of 0.0198 and the largest observed oscillation range of 0.0816 underscore the variability in QA's accuracy, reflecting periodic deviations from optimal solutions before recovery.

Figure 8b examines the oscillation range closely. The range is largest at the start, peaking around 0.08. This indicates that during early iterations, QA undergoes significant variability, which can be attributed to the exploration phase. During this phase, the algorithm, via quantum tunneling, transitions between energy states, leading to larger fluctuations in accuracy [51]. As iterations progress, the oscillation range gradually decreases, dropping to around 0.04 and stabilizing. This trend reflects the annealing process settling into more stable regions of the solution space. Between iterations 40 and 60, oscillations drop significantly, with ranges as low as 0.02, indicating a near-convergence phase. However, a resurgence of oscillations occurs around iteration 60, where the range increases back to 0.05. Toward the end, the oscillation range stabilizes at very low values, fluctuating below 0.03, suggesting convergence has been nearly achieved. This trend reveals the balance between exploratory and refinement phases of the quantum annealing process.

Such fluctuations are consistent with intrinsic properties of QA, where solution quality is influenced by quantum tunneling and stochastic noise in the quantum systems [51]. Rather than just the nature of quantum mechanical effects, the oscillations are likely a result of hardware noise and decoherence [52]. Even though the oscillations can lead the annealer to sub-optimal solutions, they are not necessarily detrimental. The convergence of QA has been examined along with the optimal convergence conditions [53]. Results provide theoretical assurance of the strong ergodicity of QA, implying that the long-term behavior of the system under less restrictive conditions allows QA to visit all possible states within its phase space.

For QUBO problems, where the solution space can be highly non-linear, QA's oscillatory behavior offers an advantage by preventing premature convergence to suboptimal solutions. On the other hand, excessive oscillations near convergence (around iteration 60) may indicate inefficiencies, such as a non-optimized annealing schedule or problem-specific properties of the QUBO matrix, including rugged energy landscapes or highly degenerate solutions. Adjusting the annealing rate or introducing other problem-specific optimizations could further enhance convergence speed and stability [45].

The key distinction between QA and SA lies in their convergence stability and accuracy. SA consistently achieved perfect accuracy without variability, whereas QA's oscillatory behavior and slightly lower accuracy reflect the limitations of current state-of-the-art QA hardware [7]. Despite this, QA demonstrated comparable average performance to SA, particularly notable due to the relative differences in the size capacity of the solvers.

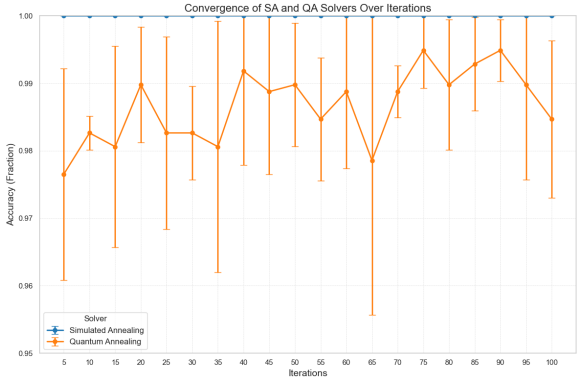


Figure 9: Avaraged convergence of QA and SA over 5 runs in steps of 5 iterations.

Run	Quantum Annealing		Range (Max - Min)
	Mean	Std Dev	
1	0.9862	0.0135	0.0357
2	0.9880	0.0116	0.0306
3	0.9829	0.0158	0.0561
4	0.9888	0.0115	0.0408
5	0.9875	0.0132	0.0255

Table 2: Performance Summary of Quantum Annealing.

Statistical analysis of QA convergence

Since QA is inherently probabilistic, the convergence of QA was also examined over multiple runs. The accuracy was reported every 5 iterations up to 100 iterations. This process was repeated 5 times, with the outcomes illustrated in Figure 9. The statistical analysis of 5 runs is displayed in Table 2.

In all 5 runs, the SA solver performed soundly, proofing itself as a reliable and efficient method for relatively smaller matrix sizes [48]. In contrast, QA achieves mean accuracies below 1.0, ranging from 0.9829 to 0.9888, with standard deviations between 0.0115 and 0.0158. This once again illustrates the variability in QA's convergence, suggesting that while QA is capable of approaching the optimal solution, it does not consistently achieve it in every run. This behavior further highlights the role of hardware noise and stochastic effects in limiting QA performance [54].

Fitting analysis

A fit was made to encapsulate this behavior. As Figure 10 illustrates, the exponential model provides a reasonably strong fit. Standard fitting matrices, R^2, RMSE, and Residuals were calculated and are displayed in Table 3.
The choice of an exponential model is motivated by the theoretical underpinnings of annealing processes, where the rate of improvement typically decays over time as the system transitions from exploration to refinement phase. The model's fit to the observed data validates its appropriateness in describing QA convergence.

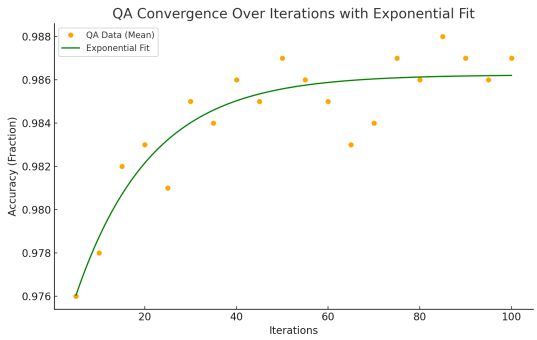


Figure 10: Exponential fit of QA mean accuracy data collected over 5 runs.

Metric	Value
R^2	0.828
RMSE	0.00126
Initial Rate of Change	6.2×10^{-4}
Final Rate of Change	1.89×10^{-6}
Mean Residual	8.28×10^{-13}
Std Dev of Residuals	0.00126

Table 3: Quantitative Analysis of QA Convergence Fit.

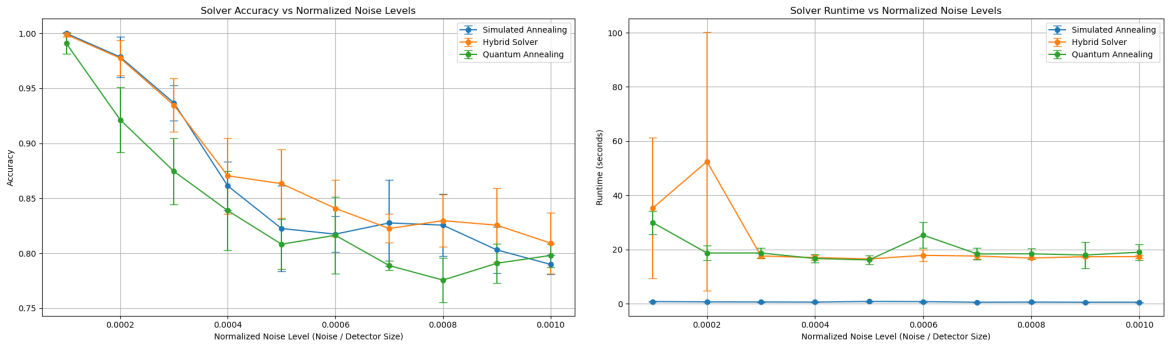
The exponential fit provides critical insights into the solver's iterative behavior. The model, with an R^2 value of 0.828, captures a strong correlation between the observed accuracy and the number of iterations. While the residual standard deviation of 0.00126 indicates a close approximation, the variability in residuals suggests minor discrepancies likely attributed to hardware noise and the inherent stochasticity of quantum tunneling [55]. Growth rate parameter of this fit, $b = 0.0610$ indicates a reasonably rapid the rate at which the accuracy approaches its asymptotic value. The initial rate of change, 6.2×10^{-4}, indicates rapid early convergence, with most accuracy improvements occurring within the first 30—50 iterations. Beyond this range, the accuracy stabilizes, as shown by the reduced final rate of change, 1.89×10^{-6}. This behaviour aligning with theoretical expectations of annealing dynamics [50].

The model predicts an asymptotic accuracy of 0.9862 under the given conditions, suggesting that while QA may not fully converge to a global minimum within 100 iterations, it achieves near-optimal solutions. The diminishing returns pattern observed in Figure 10 underscores the trade-offs inherent in QA, namely rapid initial convergence followed by a plateau where further iterations yield marginal improvements. Such behavior is consistent with the annealing schedule transitioning from exploration to refinement phases, a characteristic feature of QA[53]. While theoretical models predict QA's asymptotic convergence to the global minimum under ideal adiabatic conditions, practical implementations remain constrained by finite annealing times, noise, and hardware imperfections, which can prevent true asymptotic convergence [53, 56]. The exponential fit suggests that QA exhibits diminishing returns in accuracy improvements beyond 50 iterations. The exponential fit suggests that QA exhibits diminishing returns in accuracy improvements beyond 50 iterations. This finding has practical implications for determining stopping criteria in QA applications, allowing computational resources to be allocated more efficiently by terminating iterations once the accuracy stabilizes.

Although QA's asymptotic accuracy is slightly lower than SA's, its capacity to avoid premature convergence highlights its suitability for HEP problems with complex solution spaces. Furthermore, its diminishing error rate makes it ideal for scenarios demanding rapid approximations in challenging optimization landscapes.

4.3 Noice resiliance

Figure 11a compares the accuracy of three QUBO solvers under varying levels of normalized noise ($\frac{\text{Noise}}{\text{Detector Size}}$). The figure 11 highlights the inverse relationship between solver accuracy and increasing noise, a trend driven by inconsistencies introduced into the Hamiltonian matrix that hinder convergence to correct particle tracks.



(a) Accuracy with respect to increasing noise levels.

(b) Execution time with respect to Increasing noise levels.

Figure 11: Gaussian Noise resilience comparison between QA, SA and HA solver over 5 repetitions.

At the lowest noise level ($0.0001\frac{\text{Noise}}{\text{Detector Size}}$), all solvers perform near-perfectly, with HA and SA achieving accuracies above 97% and QA maintaining a mean accuracy of 98.5%. As noise increases, HA and SA retain high accuracy, while QA experiences a more pronounced decline. Beyond $0.0004\frac{\text{Noise}}{\text{Detector Size}}$, performance divergence becomes evident. HA sustains a mean accuracy of approximately 87%, outperforming SA and QA. Interestingly, at $0.0007\frac{\text{Noise}}{\text{Detector Size}}$, SA momentarily surpasses the other solvers with an accuracy of 83%. At the highest tested noise level, HA demonstrates the most resilience, achieving 81.5% accuracy compared to 80% and 79% for QA and SA, respectively.

QA solver noise resiliance

QA exhibits the steepest accuracy decline, dropping from 99% to 78% as noise increases to $0.001\frac{\text{Noise}}{\text{Detector Size}}$. This sensitivity aligns with known challenges of embedding noisy Hamiltonians onto quantum hardware, where limited qubit connectivity and the requirement for long qubit chains amplify noise effects [50]. The variability in QA's accuracy, particularly at lower noise levels, reflects its probabilistic nature, as quantum tunneling and hardware imperfections can occasionally lead to suboptimal solutions.

Interestingly, QA demonstrates a slight recovery between $0.0008\frac{\text{Noise}}{\text{Detector Size}}$ and $0.001\frac{\text{Noise}}{\text{Detector Size}}$. This suggests that its probabilistic exploration of rugged energy landscapes may sometimes counteract noise, a phenomenon observed in cases where tunneling enables global minima discovery despite noisy conditions [57, 58]. This resilience highlights QA's potential advantages, though further investigation into energy landscape dynamics is necessary to explain these behaviors comprehensively.

HA and SA solvers noise resiliance

SA performs reliably at lower noise levels, leveraging its deterministic algorithmic nature to converge effectively [48]. However, as noise increases, SA's accuracy gradually declines due to its susceptibility to rugged energy landscapes. This limitation is consistent with the algorithm's slower convergence to global minima in the presence of local minima [54]. HA solver demonstrates notable resilience to noise, consistently outperforming QA and SA in terms of accuracy at higher noise levels. By integrating classical pre-processing to mitigate noise before leveraging quantum optimization, HA effectively reduces the impact of inconsistencies in the Hamiltonian matrix [35]. Its ability to maintain high accuracy, despite some runtime variability, underscores its robustness in noisy environments.

All solvers exhibit stabilization between 0.0006 and 0.001, which can be attributed to a noise-driven phenomenon in certain scenarios, referred to as the flattening of the energy landscape. While this generally reduces accuracy, it can also prevent further dramatic drops in performance, leading to a plateau effect [58]. At high noise levels, this flattening allows QA to continue sampling solutions via tunneling [58], SA to stabilize through thermal fluctuations [59], and the HA solver to leverage classical pre-processing for noise mitigation [35].

Hamiltonian construction and noise resiliance

The performance of all solvers is inherently tied to the quality of the Hamiltonian matrix. The noise in the data impacts the angular and bifurcation checks, introducing inconsistencies in representing the actual track problem case effectively. This results in a Hamiltonian matrix with more erroneous terms. At lower noise levels, the Hamiltonian matrix retains its core structure, so even small distortions have a disproportionate impact on a solver's accuracy, as visible with the sharp decline between 0.0002 and 0.0004 $\frac{\text{Noise}}{\text{Detector Size}}$in figure 11a. This suggests that small perturbations in the Hamiltonian matrix significantly alter the energy landscape, generating spurious local minima that increase ruggedness and distort the global minimum. The quality of the QUBO formulation is central to solver efficiency, regardless of specific optimization mechanisms. The pronounced initial decline across all solvers points to Hamiltonian generation as the primary bottleneck of the algorithm. Strategies to mitigate noise during Hamiltonian construction should be implemented in order for the algorithm to tackle suboptimal and noisy conditions of real data application.

Resourse consumption and noise resiliance

Figure 11b shows the runtime of the solvers as noise increases. SA demonstrates the most consistent runtime, remaining near-zero across all noise levels due to its lightweight computational overhead [48]. In contrast, HA displays a significant spike in runtime at $0.0002\frac{\text{Noise}}{\text{Detector Size}}$, accompanied by high variability. This is likely due to increased computational demands during classical pre-processing to address noise before quantum optimization. At higher noise levels, HA's runtime stabilizes, reflecting its adaptability.
QA exhibits moderate runtime at low noise levels, which stabilizes as noise increases. This stability aligns with findings that quantum annealing runtime is more influenced by problem embedding complexity than by noise levels [50]. While QA's runtime variability is lower than HA's, its accuracy decline limits its utility under high-noise conditions.

Overall evaluation

The results in Table 4 highlight distinct solver's behavior under varying noise levels. The HA Solver achieves the highest mean accuracy of $(0.8979 \pm 0.0701)\%$) but has the largest runtime variability due to a spiking instance at 0.0002, leading to a higher mean runtime (24.1286s). In contrast, SA maintains the lowest mean runtime of $(0.6870 \pm 0.1118)s$ and consistent accuracy, though it struggles in higher-noise conditions. QA displays moderate runtime (20.0860s) with low variability, reflecting consistent computational demands. However, its accuracy $(0.8473 \pm 0.0664)\%$ is slightly lower, declining steeply with noise but occasionally surpassing SA, as seen at 0.001 $\frac{Nois}{DetectorSize}$.

This analysis indicates that the Hamiltonian generation process requires significant improvement, particularly by refining conditional checks to enhance its robustness. Additionally, the HA Solver's integration of classical pre-processing with quantum optimization proves effective in maintaining resilience under noisy conditions, while the QA solver shows notable capability in exploring intricate energy landscapes although further investigation is needed for definitive conclusions.

Table 4: Performance Summary of Solvers Across Noise Levels

Solver	Accuracy			Runtime (s)		
	Mean	Std Dev	Range	Mean	Std Dev	Range
Simulated Annealing	0.8634	0.0567	0.79-1.00	0.6870	0.1118	0.58-0.86
Hybrid Annealing	0.8979	0.0701	0.81-1.00	24.1286	17.6837	16.54-52.48
Quantum Annealing	0.8473	0.0664	0.77-0.99	20.0860	2.9277	16.19-29.92

The complexity of real LHCb collision events, illustrated in Figure 12 [60], underscores the significant challenges faced by solvers in real-world applications. Unlike the simplified toy model, real collisions involve dense networks of overlapping tracks, secondary interactions, and intricate detector responses [1]. Scalability, accuracy, and noise resilience tests suggest that HA solver has the greatest potential to handle small to medium-sized VELO events effectively. In contrast, QA struggles with scalability but shows promise for handling specific noise types. While advancements in quantum hardware [7] offer hope for QA's future applicability, hybrid approaches remain the most promising solution for addressing the full complexity of real LHCb collision events.

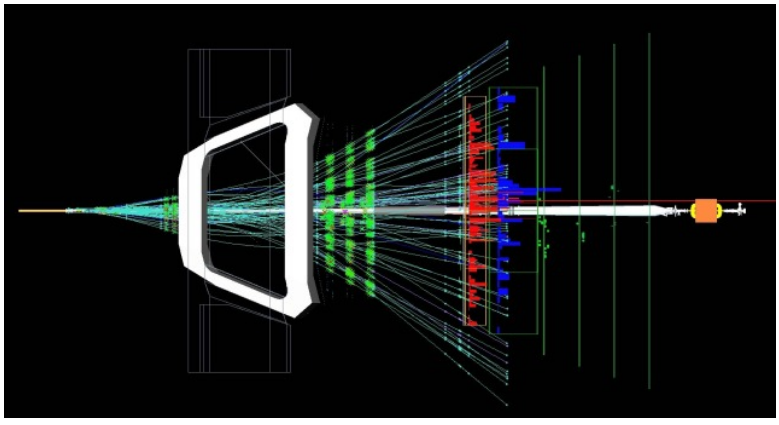


Figure 12: Event display of proton lead collisions from the LHCb VELO detector [60], showing dense particle tracks interacting with multiple layers.

5 Future Reaserch and Conclusion

This study systematically evaluates the performance of an annealing-based algorithm for particle track reconstruction, encompassing an optimized Hamiltonian generator and three distinct solvers: Quantum Annealing (QA), Simulated Annealing (SA), and Hybrid annealing (HA). By addressing key challenges in scalability, accuracy, convergence, and noise resilience, the research provides insights into the applicability and limitations of these paradigms for the track problem.

The optimized Hamiltonian generator exhibited substantial gains in computational efficiency, enabling it to scale to problem sizes of 5 layers and 65 particles (12000 NxN). Sparse matrix representations, on-the-fly computations, and parallelization collectively reduced both runtime and memory usage, demonstrating its suitability for encoding small to medium-sized reconstruction tasks into QUBO formulations. However, sensitivity to noise in data highlights the importance of further refining the Hamiltonian construction process to mitigate noisy conditions.

The scalability of the QA solver is insufficient to meet the demands of LHCb VELO track reconstruction. QA demonstrated asymptotic convergence to a mean accuracy of $(98.67 \pm 1.75)\%$ over 100 iterations for 5 layers and 7 particles (196 N×N), highlighting its current limitations for larger-scale tracking problems. Despite comparable performance to SA and Hybrid approaches under noisy conditions, QA achieved the lowest mean accuracy across all noise levels at $(84.73 \pm 6.64)\%$. SA consistently delivered robust performance in low-noise environments, achieving perfect accuracy (100%) and adequate runtime for matrix sizes up to 5 layers and 65 particles (19600 N×N). However, it struggled with rugged energy landscapes and noisy conditions, with an average accuracy drop of 21 %. The Hybrid solver emerged as the most versatile and scalable, achieving the highest mean accuracy of $(89.79 \pm 7.01)\%$ across all noise levels. Its stability in noisy conditions and superior scalability (up to 40000 N×N).

Collectively, these results establish the current status of QA, SA and HA in tackling challenges associated with the tracking problem. For further research, additional accuracy testing for HA and SA should be conducted, incorporating advanced optimization techniques. These techniques could include problem-specific annealing schedules and optimized pre- and post-processing methods tailored for hybrid annealing. Additionally, the convergence behavior of QA should be thoroughly investigated by examining the energy landscapes under varying conditions. This would help identify specific scenarios in which QA demonstrates promising performance. Furthermore, noise resilience studies could be expanded to include different noise models beyond Gaussian noise. Finally, the application of the annealing-based algorithm to real-world data should be explored to validate practical utility for LHCb VELO detector.

6 Critical Reflection

Working on this bachelor's thesis has given me a clearer understanding of both scientific research and my own abilities. Combining particle physics with quantum principles, my favorite topic, deepened my appreciation for how these fields connect and the incredible work they represent. I've always been drawn to puzzle-solving aspect of scientific research which initially brought me to MSP. I thoroughly enjoyed the puzzles this BTR project brought, but I also noticed importance of addressing practical limitations in research—an area where I recognize I have room to grow. Initially, the annealing-based algorithm was aimed to be applied to real LHCb VELO data, but given three months timeframe, this goal was too ambitious. Furthermore, although I've managed deadlines well, my organizational habits can be chaotic and need improvement. I now pay more attention to areas like file naming, code comments, reference management, and structuring research.
Finally, working alongside experienced and knowledgeable people showed me first-hand the dedication science demands. Though initially intimidating, it inspires me to strive for greater competence in the field.

7 Appendix

The GitHub repository containing all the code used in this study can be accessed at the following link:
https://github.com/vitahehe/quantum-ocean-sdk

The Quantum Annealing algorithm is available in the `Complete_annealing_algorithem` file.

The `trackhll` folder contains the event model and detector setup described in Section 2.4 (Background).

The original Hamiltonian generator can be found in the `playground_original_code` file.

Testing algorithms for the Hamiltonian generator and QUBO solvers are located in the `testingHAMILTONIAN` and `testingQUBO` files, respectively.

Solution testing, including code for noise resilience and convergence analysis, is available in `TESTINGSOLUTIONS.ipynb` and `Convergence_Noise.ipynb`.

References

[1] CERN. *The Large Hadron Collider: Facts and Achievements*. Accessed: 2024-09-22. 2024. URL: https://home.cern/.

[2] Rolf Paula Collins; Lindner. "LHCb VELO Upgrade Technical Design Report". In: (2013). CERN Document Server Record. URL: https://cds.cern.ch/record/1624070.

[3] G. Apollinari et al. "High-Luminosity Large Hadron Collider (HL-LHC)". In: *CERN Yellow Report: Monographs*. Vol. 4. 2017, pp. 1–516. DOI: 10.23731/CYRM-2017-004.

[4] D. Magano et al. "Quantum speedup for track reconstruction in particle accelerators". In: *Phys. Rev. D* 105 (7 Apr. 2022), p. 076012. DOI: 10.1103/PhysRevD.105.076012. URL: https://link.aps.org/doi/10.1103/PhysRevD.105.076012.

[5] Efrén Rodríguez Rodríguez. "The LHCb VELO detector: Design, operation and first results". In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 1065 (2024), p. 169469. ISSN: 0168-9002. DOI: https://doi.org/10.1016/j.nima.2024.169469. URL: https://www.sciencedirect.com/science/article/pii/S0168900224003954.

[6] Davide Nicotra et al. *A quantum algorithm for track reconstruction in the LHCb vertex detector*. 2023. arXiv: 2308.00619 [quant-ph]. URL: https://arxiv.org/abs/2308.00619.

[7] D-Wave Systems Inc. *D-Wave Systems*. Accessed: 2024-11-04. 2024. URL: https://www.dwavesys.com/.

[8] Kyungtaek Jun. "QUBO formulations for a system of linear equations". In: *Results in Control and Optimization* 14 (2024), p. 100380.

[9] D. C. Spink. *Principles of Physical Metallurgy*. McGraw-Hill, 1946.

[10] Darrall Henderson, Sheldon H Jacobson, and Alan W Johnson. "The theory and practice of simulated annealing". In: *Handbook of metaheuristics* (2003), pp. 287–319.

[11] Franco Busetti. "Simulated annealing overview". In: *World Wide Web URL www. geocities. com/francorbusetti/saweb. pdf* 4 (2003).

[12] Satoshi Morita and Hidetoshi Nishimori. "Mathematical foundation of quantum annealing". In: *Journal of Mathematical Physics* 49.12 (2008).

[13] Catherine Mcgeoch. "Theory versus practice in annealing-based quantum computing". In: *Theoretical Computer Science* 816 (May 2020). DOI: 10.1016/j.tcs.2020.01.024.

[14] Tadashi Kadowaki and Hidetoshi Nishimori. "Quantum annealing in the transverse Ising model". In: *Phys. Rev. E* 58 (5 Nov. 1998), pp. 5355–5363. DOI: 10.1103/PhysRevE.58.5355. URL: https://link.aps.org/doi/10.1103/PhysRevE.58.5355.

[15] Gian Giacomo Guerreschi and Mikhail Smelyanskiy. *Practical optimization for hybrid quantum-classical algorithms*. 2017. arXiv: 1701.01450 [quant-ph]. URL: https://arxiv.org/abs/1701.01450.

[16] Andrew M. Childs et al. *Quantum divide and conquer*. 2022. arXiv: 2210.06419 [quant-ph]. URL: https://arxiv.org/abs/2210.06419.

[17] Daniel Hugo Cámpora Pérez, Niko Neufeld, and Agustín Riscos Núñez. "Search by triplet: An efficient local track reconstruction algorithm for parallel architectures". In: *Journal of Computational Science* 54 (Sept. 2021), p. 101422. ISSN: 1877-7503. DOI: 10.1016/j.jocs.2021.101422. URL: http://dx.doi.org/10.1016/j.jocs.2021.101422.

[18] Anthony Correia et al. *Graph Neural Network-Based Pipeline for Track Finding in the Velo at LHCb*. 2024. arXiv: 2406.12869 [physics.ins-det]. URL: https://arxiv.org/abs/2406.12869.

[19] Tameem Albash and Daniel A. Lidar. "Demonstration of a Scaling Advantage for a Quantum Annealer over Simulated Annealing". In: *Phys. Rev. X* 8 (3 July 2018), p. 031016. DOI: 10.1103/PhysRevX.8.031016. URL: https://link.aps.org/doi/10.1103/PhysRevX.8.031016.

[20] Alexander Zlokapa et al. "Charged particle tracking with quantum annealing optimization". In: *Quantum Machine Intelligence* 3.2 (Nov. 2021). ISSN: 2524-4914. DOI: 10.1007/s42484-021-00054-w. URL: http://dx.doi.org/10.1007/s42484-021-00054-w.

[21] B. Denby. "Neural networks and cellular automata in experimental high energy physics". In: *Computer Physics Communications* 49 (1988), p. 429.

[22] C. Peterson. "Track finding with neural networks". In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 279 (1989), p. 537.

[23] Hideki Okawa et al. "Quantum-Annealing-Inspired Algorithms for Track Reconstruction at High-Energy Colliders". In: *Computing and Software for Big Science* 8.1 (Aug. 2024). ISSN: 2510-2044. DOI: 10.1007/s41781-024-00126-z. URL: http://dx.doi.org/10.1007/s41781-024-00126-z.

[24] William H Press et al. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge University Press, 2007.

[25] Brandon Yue. "Optimizing Out-Of-Memory Sparse-Dense Matrix Multiplication". MA thesis. Massachusetts Institute of Technology, 2020. URL: https://dspace.mit.edu/handle/1721.1/127021.

[26] I Arora. "Improving Performance of Data Science Applications in Python". In: *Indian Journal of Science and Technology* 17.24 (2024), pp. 2499–2507.

[27] Shoya Yasuda, Shunsuke Sotobayashi, and Yuichiro Minato. *HOBOTAN: Efficient Higher Order Binary Optimization Solver with Tensor Networks and PyTorch*. 2024. arXiv: 2407.19987 [cs.MS]. URL: https://arxiv.org/abs/2407.19987.

[28] Radhya Sahal. "Query Optimization for Big Data Batch Processing and Stream Processing". In: *Acta Scientific Computer Sciences* 4.1 (2022), pp. 4–7. URL: https://actascientific.com/ASCS/pdf/ASCS-04-0255.pdf.

[29] Joblib Development Team. *Joblib: running Python functions as pipeline jobs*. Version 1.4.2. 2023. URL: https://joblib.readthedocs.io/.

[30] James Demmel et al. "Avoiding communication in sparse matrix computations". In: *2008 IEEE International Symposium on Parallel and Distributed Processing*. IEEE. 2008, pp. 1–12.

[31] Gene H Golub and Charles F Van Loan. "Matrix computations". In: *JHU press* (2012).

[32] Philipp Rucker and Jörg Behrens. "Avoiding unnecessary computation and storage in high-performance computing applications". In: *Lecture Notes in Computer Science* 3241 (2004), pp. 725–732.

[33] Donald E Knuth. *The Art of Computer Programming, Volume 4A: Combinatorial Algorithms, Part 1*. Addison-Wesley Professional, 2011.

[34] R. Aaij et al. "Performance of the LHCb Vertex Locator". In: *arXiv preprint arXiv:1405.7808* (2014). URL: https://arxiv.org/abs/1405.7808.

[35] D-Wave Systems. *Leap Hybrid Solver Documentation*. https://docs.ocean.dwavesys.com/en/stable/hybrid/overview.html. Accessed: 2024-04-27.

[36] Dimod Development Team. *Simulated Annealing Sampler*. https://docs.ocean.dwavesys.com/en/stable/docs_neal/reference/sampler.html. Accessed: 2024-04-27.

[37] Gregory Gundersen. "Random Noise and the Central Limit Theorem". In: (2019). URL: https://gregorygundersen.com/blog/2019/02/01/clt/.

[38] Matthew John Needham. "Characterisation and commissioning of the LHCb VELO detector". PhD thesis. University of Edinburgh, 2009. URL: https://cds.cern.ch/record/1186697/files/CERN-THESIS-2009-044.pdf.

[39] SciPy Community. *Sparse Matrices (scipy.sparse)*. Version 1.14.1. 2023. URL: https://docs.scipy.org/doc/scipy/reference/sparse.html.

[40] Pauli Virtanen et al. "SciPy 1.0: Fundamental algorithms for scientific computing in Python". In: *Nature Methods* 17.3 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

[41] Jeffrey Dean and Sanjay Ghemawat. "MapReduce: Simplified data processing on large clusters". In: *Communications of the ACM* 51.1 (2008), pp. 107–113. DOI: 10.1145/1327452.1327492.

[42] Elijah Pelofske. *Comparing Three Generations of D-Wave Quantum Annealers for Minor Embedded Combinatorial Optimization Problems*. 2024. arXiv: 2301.03009 [quant-ph]. URL: https://arxiv.org/abs/2301.03009.

[43] Atanu Rajak et al. "Quantum annealing: an overview". In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 380.2214 (2022), p. 20210417. DOI: 10.1098/rsta.2021.0417. URL: https://doi.org/10.1098/rsta.2021.0417.

[44] Catherine McGeoch. *Adiabatic Quantum Computation and Quantum Annealing: Theory and Practice*. Vol. 5. Synthesis Lectures on Quantum Computing 2. Morgan Claypool Publishers, 2014, pp. 1–93. DOI: 10.2200/S00578ED1V01Y201405QMC007. URL: https://doi.org/10.2200/S00578ED1V01Y201405QMC007.

[45] T. F. Rønnow, Z. Wang, J. Job, et al. "Defining and Detecting Quantum Speedup". In: *Science* 345.6195 (2014), pp. 420–424. DOI: 10.1126/science.1252319. URL: https://doi.org/10.1126/science.1252319.

[46] Patrick Siarry. "Simulated Annealing: From Basics to Applications". In: *Metaheuristics: From Design to Implementation*. Springer, 2018, pp. 1–25. DOI: 10.1007/978-3-319-91086-4_1.

[47] R. Harris, T. Lanting, A. J. Berkley, et al. "Quantum Annealing with Manufactured Spins". In: *Nature* 421.6921 (2018), pp. 823–826. DOI: 10.1038/nature02192. URL: https://doi.org/10.1038/nature02192.

[48] Thomas Guilmeau, Emilie Chouzenoux, and Víctor Elvira. "Simulated Annealing: a Review and a New Scheme". In: *2021 IEEE Statistical Signal Processing Workshop (SSP)*. 2021, pp. 101–105. DOI: 10.1109/SSP49050.2021.9513782.

[49] V. Shiltsev. "Quantum Computing for High Energy Physics". In: *Reviews of Accelerator Science and Technology* 13.1 (2020), pp. 29–46. DOI: 10.1142/S1793626820300033. URL: https://doi.org/10.1142/S1793626820300033.

[50] Tameem Albash and Daniel A. Lidar. "Adiabatic quantum computation". In: *Reviews of Modern Physics* 90.1 (Jan. 2018). ISSN: 1539-0756. DOI: 10.1103/revmodphys.90.015002. URL: http://dx.doi.org/10.1103/RevModPhys.90.015002.

[51] Edward Farhi et al. *Quantum Computation by Adiabatic Evolution*. 2000. arXiv: quant-ph/0001106 [quant-ph]. URL: https://arxiv.org/abs/quant-ph/0001106.

[52] Benjamin Tasseff, Tameem Albash, Zachary Morrell, et al. "On the Emerging Potential of Quantum Annealing Hardware for Combinatorial Optimization". In: *Journal of Heuristics* 30 (2024), pp. 325–358. DOI: 10.1007/s10732-024-09530-5. URL: https://doi.org/10.1007/s10732-024-09530-5.

[53] Satoshi Morita and Hidetoshi Nishimori. "Convergence theorems for quantum annealing". In: *Journal of Physics A: Mathematical and General* 39.45 (Oct. 2006), pp. 13903–13920. ISSN: 1361-6447. DOI: 10.1088/0305-4470/39/45/004. URL: http://dx.doi.org/10.1088/0305-4470/39/45/004.

[54] Takahiro Nishimori and Kazuhiro Takeda. "Quantum and classical annealing in a continuous space with multiple local minima". In: *Physical Review A* 105 (6 2022), p. 062435. DOI: 10.1103/PhysRevA.105.062435.

[55] Yuta Shingu et al. *Quantum annealing with error mitigation*. 2022. arXiv: 2210.08862 [quant-ph]. URL: https://arxiv.org/abs/2210.08862.

[56] Mohammad H. Amin et al. "Quantum error mitigation in quantum annealing". In: *arXiv preprint arXiv:2311.01306* (2023). URL: https://arxiv.org/abs/2311.01306.

[57] Philipp Hauke et al. "Perspectives of quantum annealing: methods and implementations". In: *Reports on Progress in Physics* 83.5 (May 2020), p. 054401. ISSN: 1361-6633. DOI: 10.1088/1361-6633/ab85b8. URL: http://dx.doi.org/10.1088/1361-6633/ab85b8.

[58] David Roberts et al. "Noise amplification at spin-glass bottlenecks of quantum annealing: A solvable model". In: *Phys. Rev. A* 101 (4 Apr. 2020), p. 042317. DOI: 10.1103/PhysRevA.101.042317. URL: https://link.aps.org/doi/10.1103/PhysRevA.101.042317.

[59] Jürgen Branke, Stephan Meisel, and Christian Schmidt. "Simulated Annealing in the Presence of Noise". In: *Journal of Heuristics* 14.6 (2008), pp. 627–654. DOI: 10.1007/s10732-007-9058-7. URL: https://link.springer.com/article/10.1007/s10732-007-9058-7.

[60] LHCb Collaboration. *Proton-lead collisions as seen in LHCb*. CERN Document Server. Event Display for CERN Website Homepage. © 2012 CERN, for the benefit of the LHCb Collaboration. Accessing copyrighted material. 2012. URL: https://cds.cern.ch/record/1473466.