AWS Quantum Technologies Blog

Cách Strangeworks sử dụng Amazon Braket để khám phá vấn đề xếp hàng hóa lên máy bay

bởi Stuart Flannigan, Andrew J. Ochoa, Michael Brett và Charunethran Panchalam Govindarajan vào **02 THÁNG 9, 2025** trong Amazon Braket, Quantum Technologies

Máy tính lượng tử hứa hẹn đem lại bước ngoặt cho việc tính toán các bài toán ở nhiều ngành công nghiệp, mặc dù vẫn còn là câu hỏi mở là công nghệ này sẽ hữu ích ra sao trong thực tế. Trong bài blog này, nhóm từ Strangeworks, một đối tác AWS, đánh giá các triển khai khác nhau của thuật toán **QAOA (Quantum Approximate Optimization Algorithm)** đối với vấn đề xếp hàng hóa lên máy bay do Airbus đặt ra như một phần của Quantum Mobility Challenge năm trước.

Bài viết xem xét nhiều biến thể của QAOA và một loạt benchmark sử dụng QPU **Rigetti Ankaa-3** có sẵn qua Amazon Braket.

Vấn đề xếp hàng hóa máy bay mà Airbus đưa ra là dạng bài toán tối ưu kiểu **bin packing** (đóng hộp) xảy ra trong nhiều ngành — du lịch, sản xuất, logistics. Những bài toán kiểu này rất khó giải vì số lượng các khả năng tăng theo cấp số nhân khi biến đổi nhiều hơn, dẫn đến không gian bài toán rất lớn ngay cả với các trường hợp đơn giản.

Máy tính lượng tử có thể phù hợp để giải các bài toán tối ưu này, có tiềm năng tăng tốc so với giải pháp cổ điển (classical). Tuy nhiên, khi phần cứng lượng tử hiện tại vẫn chưa vượt trội so với các công nghệ cổ điển, các phương pháp heuristic kết hợp xử lý lượng tử và cổ điển (hybrid) đang được xem xét nhiều.

Hiện tại phần cứng lượng tử kiểu "gate-based" chưa vượt qua được công nghệ cổ điển tốt nhất, nhưng phương pháp benchmark trong bài này đạt được kết quả chính xác cho các bài toán lên đến **80 qubit / biến**.

Thuật toán QAOA thuộc loại **hybrid quantum-classical**, nghĩa là nó sử dụng cả phần tính toán lượng tử và phần tính toán cổ điển, được tin là phù hợp trong thời đại NISQ (noisy intermediate-scale quantum). Nhóm Strangeworks dựa trên hai thuật toán QAOA có sẵn:

- QAOA tiêu chuẩn (Standard QAOA), có trong nhiều thư viện mã nguồn mở bao gồm Braket algorithm library .
- **Relax-and-Round QAOA**, một biến thể được phát triển bởi đội của Rigetti Computing.

Nhóm cũng phát triển biến thể riêng **StrangeworksQAOA**, với cải tiến trong phần xử lý cổ điển, và biến thể này vượt trội so với QAOA chuẩn trong bài toán này, và cũng có cải thiện khi áp dụng cho biến thể QRR QAOA.

Sử dụng Braket Hybrid Jobs qua nền tảng Strangeworks

Kết hợp tài nguyên lượng tử và cổ điển thông qua các thuật toán hybrid như QAOA là con đường để sử dụng máy tính lượng tử hiện có cùng với luồng công việc cổ điển. Nhóm dùng Amazon Braket Hybrid Jobs, cho phép toàn bộ workflow QAOA được gửi như một job duy nhất. Điều này cải thiện thời gian chạy vì thuật toán chỉ phải chờ trong hàng đợi Braket một lần, thay vì chờ mỗi lần gửi mạch lượng tử riêng lẻ.

Một tính năng đáng chú ý khác trong Braket Hybrid Jobs là cached circuit compilation — cho phép các mạch kế tiếp nếu có cấu trúc giống mạch trước đó dùng lại phần biên dịch trước đó, tiết kiệm thời gian và chi phí.

Người dùng có thể truy cập Braket và các tính năng này qua platform **Strangeworks Compute**. Nền tảng này có hệ thống quản lý công việc trực tuyến và SDK để tải xuống, cho phép người dùng truy cập StrangeworksQAOA, Braket Hybrid Jobs cũng như các thiết bị lượng tử, lượng tử cảm hứng (“quantum-inspired”) và các solver cổ điển.

Khi kích thước hệ thống tăng, kết quả cho thấy nếu dùng thuật toán phù hợp thì phần cứng lượng tử “noisy” có thể giải các bài toán lớn hơn một cách đáng tin cậy. Sự kết hợp giữa nền tảng Strangeworks, StrangeworksQAOA, Braket Hybrid Jobs và thiết bị Rigetti Ankaa-3 thể hiện một con đường thực tế để giải các bài toán tối ưu bằng cách kết hợp tài nguyên cổ điển và lượng tử.

Phương pháp benchmark

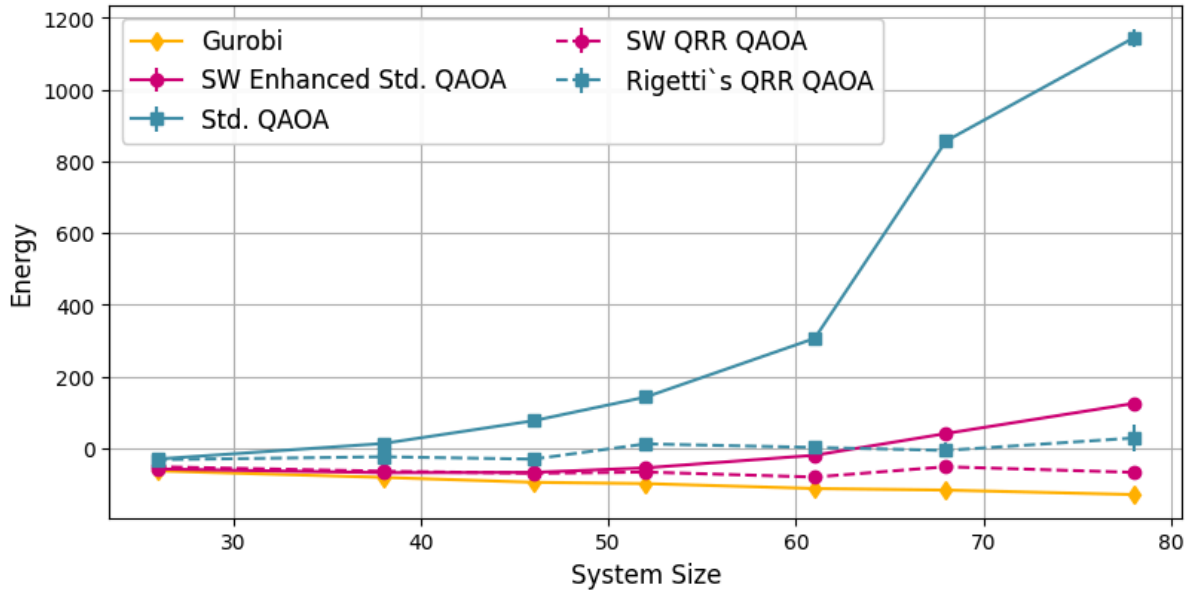
Để benchmark StrangeworksQAOA, chúng tôi xét bài toán tối ưu xếp hàng hóa lên máy bay, được hình thành ban đầu như một phần của thử thách tính toán lượng tử của Airbus [4]. Bài toán này cố gắng tối đa hóa khối lượng hàng được xếp lên máy bay, đồng thời đảm bảo rằng số lượng container, (n), có thể vừa vào số chỗ có sẵn, (N). Do đó chúng tôi bao gồm các ràng buộc: không chồng lấn container và không có container trùng lặp trên khoang. Theo Ref. [5], chúng tôi biểu diễn bài toán này dưới dạng **QUBO (Quadratic Unconstrained Binary Optimization)** mà ta có thể giải trên máy tính lượng tử bằng thuật toán QAOA . Trong kết quả benchmark, chúng tôi xét các giá trị cho số container (n) và số chỗ (N) được thể hiện trong **Bảng 1**.

Bảng 1: Các giá trị xét cho số container (n) và số chỗ có sẵn trên máy bay (N). Ta có thể tạo bài toán QUBO với số biến tương ứng, và những biến này phải được mã hóa lên máy tính lượng tử chứa cùng số qubit.

Containers, *n*	**Spaces, *N***	**No. of Variables/qubits**
5	3	26
6	4	38
6	5	46
7	5	52
7	6	61
8	6	68
8	7	78

Chúng tôi áp dụng **StrangeworksQAOA** thông qua nền tảng Strangeworks để giải bài toán tối ưu xếp hàng hóa đã mô tả ở phần trước. Trong **Hình 1**, chúng tôi so sánh **Standard QAOA** [2] (ô vuông liền) và thuật toán **StrangeworksQAOA** (vòng tròn liền), trong đó bước tối ưu hóa cổ điển đã được cải thiện.

Trong cách tiếp cận của Strangeworks, chúng tôi phân tích tất cả các bitstring cổ điển được dùng để tính hàm chi phí một cách riêng biệt và đơn giản lưu giữ (store) lời giải có chi phí tốt nhất tìm được (xem phần kỹ thuật tiếp theo). Ngoài ra, chúng tôi cũng đưa vào kết quả sử dụng biến thể **Quantum Relax-and-Round (QRR)** của QAOA từ đội Rigetti [3], nó cải thiện Standard QAOA bằng cách tối ưu hoá cách tính hàm chi phí (trái ngược với StrangeworksQAOA không sửa hàm chi phí), dẫn tới quy trình tối thiểu hóa tốt hơn (ô vuông gạch). Trong quá trình tối thiểu hóa QRR cải tiến này, chúng tôi cũng có thể áp dụng thủ tục Strangeworks để theo dõi và giữ lại bitstring tốt nhất duy nhất (đường nét gạch màu hồng) mà không tốn thêm chi phí tính toán hay chi phí tài chính so với QRR chuẩn.



Hình 1: Chúng tôi so sánh máy lượng tử **Rigetti Ankaa-3** chạy thuật toán **StrangeworksQAOA** (vòng tròn liền) với **Standard QAOA** (ô vuông liền), cả hai dùng framework **Braket Hybrid Job**. Ngoài ra, còn có kết quả từ QRR của Rigetti (ô vuông gạch) và **StrangeworksQRR** (vòng tròn gạch). Chúng tôi dùng mạch **RealAmplitude** làm ansatz QAOA. Lưu ý: chúng tôi lấy trung bình kết quả trên 8 lần chạy và biểu diễn sai số thống kê của giá trị trung bình bằng thanh lỗi. Để so sánh, cũng bao gồm lời giải tối ưu tìm bởi bộ giải cổ điển **Gurobi** . Chúng tôi so sánh các giá trị hàm chi phí trả về bởi mỗi thuật toán.

Trong **Bảng 2**, chúng tôi dùng hồi quy tuyến tính (linear regression) để khớp một đường thẳng cho mỗi tập điểm dữ liệu của các thuật toán khác nhau trong Hình 1. Việc này cho ta chỉ báo về cách tỉ lệ (scaling) của mỗi đường khi kích thước hệ thống được tăng lên. Sự khác biệt trong gradient của các đường này gợi ý rằng nếu ngoại suy ra kích thước lớn hơn thì các đường sẽ bắt đầu lệch nhiều hơn; sai khác lớn hơn ở gradient có nghĩa là kết quả sẽ phân kỳ nhanh hơn. Do đó, vì phiên bản Strangeworks của QRR có gradient gần nhất với kết quả chính xác (Gurobi), ta có thể suy ra rằng phiên bản này sẽ tiếp tục có sai số nhỏ nhất trong các thuật toán được thử khi kích thước hệ thống tăng. Trong khi các kết quả ở Hình 1 cho hệ kích thước xét ở phân tích này cho thấy StrangeworksQAOA có sai số nhỏ (~10%), phân tích gradient cho thấy sai số sẽ tiếp tục tồi hơn nếu kích thước hệ thống tăng thêm. Điều này là điều có thể dự đoán được và cho thấy cần cải tiến thêm cho toàn bộ workflow, tức là cả phần lượng tử lẫn phần cổ điển của thuật toán.

Algorithm	**Gradient**
Gurobi	-1.25
StrangeworksQAOA	3.47
Standard QAOA	n.a
Strangeworks QRR QAOA	-0.16

Rigetti QRR QAOA	1.06

Bảng 2: Gradient của đường khớp cho mỗi thuật toán (sau khi fit bằng hồi quy tuyến tính). Lưu ý rằng Standard QAOA tăng theo cấp số (exponential) (xem Hình 1) nên không thể có một fit tuyến tính hợp lý cho trường hợp đó.

Chúng tôi nhận thấy rằng có thể đi từ các kết quả nhiễu (noisy) không có ý nghĩa (ô vuông liền) tới các kết quả khớp chặt với lời giải tối ưu (vòng tròn gạch). Sự tương phản mạnh mẽ giữa thuật toán kém nhất (ô vuông liền) và tốt nhất (vòng tròn gạch) nhấn mạnh nhu cầu phải tối ưu hoá hạ tầng cổ điển bao quanh phần tính toán lượng tử. Các thuật toán này đều chạy trên cùng thiết bị, vì vậy tất cả đều chịu ảnh hưởng từ cùng một mô hình/spectrum nhiễu. Kết quả chỉ ra rằng **StrangeworksQAOA** bền vững hơn trước các bất cập xoay quanh các thuật toán lai lượng-cổ điển này — một tính chất có thể hữu ích nếu chúng ta muốn mở rộng kích thước bài toán trong tương lai và tiến hành thí nghiệm QAOA trên các phần cứng lớn hơn nơi nhiễu có thể đóng vai trò nổi bật hơn.

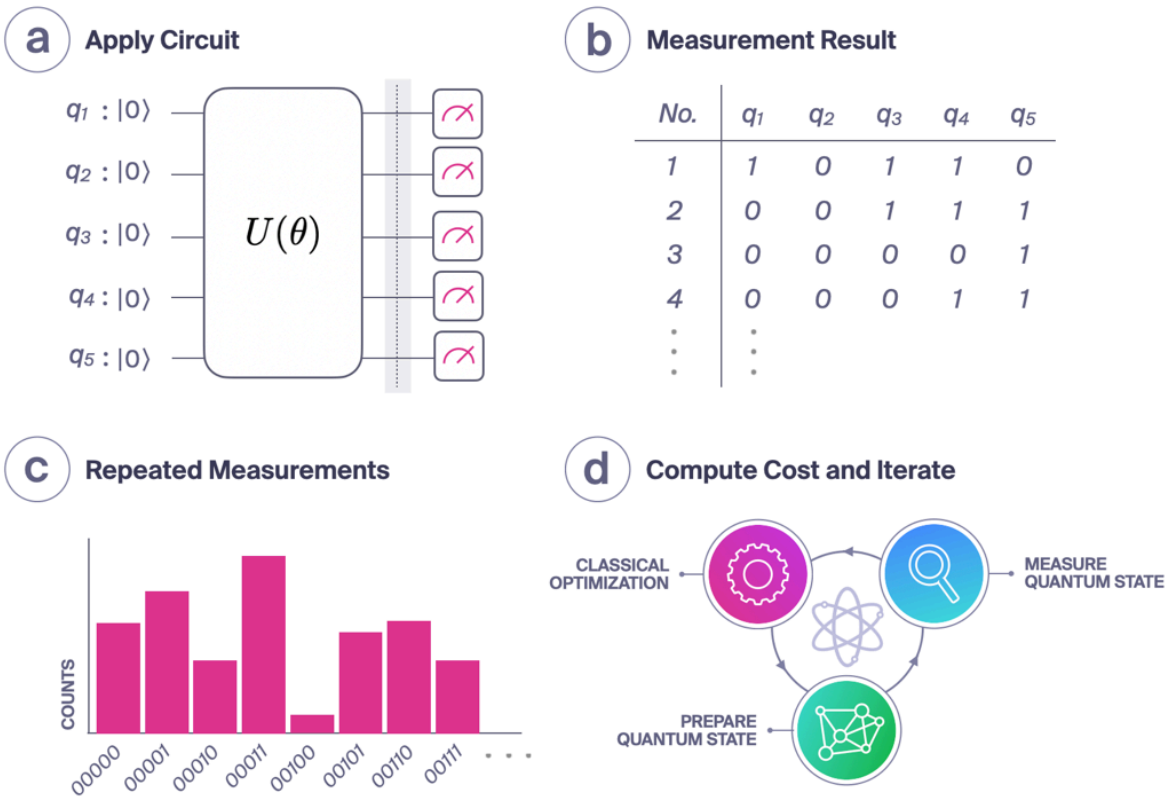
Chi tiết thuật toán StrangeworksQAOA

Thuật toán QAOA là một thuật toán hybrid cổ điển-lượng tử lặp trong một vòng giữa phần tính toán lượng tử và xử lý cổ điển (xem Hình 2). Chúng tôi bắt đầu bằng cách tham số hoá một mạch lượng tử (U(\theta)) với một tập tham số cổ điển (\theta). Trong trường hợp này, chúng tôi không dùng ansatz QAOA tiêu chuẩn do hạn chế độ sâu mạch (circuit depth) của phần cứng đối với các bài toán lớn hơn mà chúng tôi xét. Thay vào đó, chúng tôi dùng một ansatz thường dùng trong thí nghiệm VQE là **Real Amplitude Quantum Circuit** và chạy mạch tương ứng trên QPU Rigetti Ankaa-3. Lưu ý rằng chúng tôi dùng ansatz này cho tất cả các phiên bản thuật toán trong phân tích, kể cả “Standard QAOA” của chúng tôi. Sau đó chúng tôi đo trạng thái qubit (0 hoặc 1), lặp nhiều lần để thu được phân bố xác suất.

Tiếp theo, chúng tôi đưa phân bố xác suất vào một hàm cổ điển để tính hàm chi phí. Phép tính hàm chi phí là phần duy nhất của thuật toán khác nhau giữa **Standard QAOA** và **StrangeworksQAOA**, như sẽ thấy trong phần tiếp theo. Sau khi hàm chi phí được tính cho mạch lượng tử với một bộ tham số biến phân cho trước, kết quả được đưa vào hàm tối thiểu hoá cổ điển — ở đây dùng phương pháp **COBYLA** từ gói SciPy của Python — và các tham số biến phân được cập nhật. Quy trình này lặp nhiều lần, với phần lượng tử hỗ trợ tính hàm chi phí và phần cổ điển tinh chỉnh tham số biến phân cho đến khi hàm chi phí được tối thiểu hoá, dẫn tới lời giải cuối cùng.

Trong **Standard QAOA**, thường thì đáp án cuối được chọn từ kết quả đo của mạch biến phân cuối cùng, trong đó bitstring có xác suất cao nhất được lấy làm đáp án. Tuy nhiên, khi xét kích thước hệ lớn (N) với số cơ sở (2^N) tăng theo cấp số, chúng tôi thấy rằng với số lần đo thực tế và khả thi, phân bố xác suất trở nên phẳng (flat). Điều này có nghĩa mỗi trạng thái cơ sở chỉ được đo một lần, nên không thể lấy trạng thái có xác suất lớn nhất. Vì vậy, đối với

Standard QAOA trong Hình 1 chúng tôi lấy **giá trị chi phí trung bình** trên tất cả các trạng thái cơ sở đo được làm kết quả.



(Chú thích Hình 2: QAOA workflow — a) thiết lập mạch với tham số biến phân cổ điển và áp dụng lên qubit; b) đo ra bitstring; c) lặp nhiều lần để tạo phân bố xác suất; d) tính chi phí (xem công thức) và dùng thuật toán tối ưu hoá cổ điển để sinh mạch mới với tham số cập nhật; lặp lại giữa phần lượng tử và phần cổ điển để tìm mạch tối ưu cho giá trị chi phí nhỏ nhất.)

Trong QAOA nói chung, để tính hàm chi phí cổ điển ta lấy phân bố xác suất (Hình 2(c)) và bài toán QUBO/đồ thị, rồi tính chi phí cho từng trạng thái cơ sở đo được. Với trạng thái cơ sở thứ (q), chi phí cho trạng thái đó là).

$$C_q = \sum_{i,j} J_{i,j} q_i q_j$$

Ở đây (q_i {0,1}) là giá trị của bit thứ (i) trong bitstring thứ (q), và (J{i,j}) là các hệ số coupling của QUBO/đồ thị vấn đề. Tổng chi phí được tính bằng cách trọng số (C_q) theo số

lần trạng thái cơ sở đó được đo, là các count (N_q), và tổng lại để có chi phí trung bình trên tất cả kết quả đo:

$$C_T = \frac{1}{S}\sum_q C_q N_q$$

Ở **Standard QAOA**, các giá trị trung gian (C_q) thường bị bỏ qua và không được sử dụng — tuy nhiên xem Ref. [10] để thảo luận cách cải thiện.

Trong **StrangeworksQAOA**, chúng tôi theo dõi giá trị nhỏ nhất (C_q) và trạng thái cơ sở có chi phí tối thiểu trong suốt các vòng lặp của thuật toán. Trạng thái cơ sở có chi phí nhỏ nhất này sau đó được báo cáo là lời giải bởi StrangeworksQAOA trong Hình 1. Việc chọn trạng thái cơ sở như vậy là hợp lý vì lời giải cho bài toán xếp hàng hóa là một đáp án cổ điển — tức là một trạng thái cơ sở đơn. Trái ngược với các phép tính lượng tử phổ thông, nơi ta thường tính trung bình trên phân bố đo, ở đây chúng ta chỉ quan tâm mức độ tìm được lời giải tốt nhất. Trong Hình 1, mỗi phiên bản QAOA được chạy 5 lần để lấy trung bình; thanh lỗi thống kê nhỏ so với bề rộng đường biểu diễn cho thấy kết quả ổn định với bài toán được xét.

Kết luận và triển vọng

Phân tích của chúng tôi cho thấy giá trị của việc tối ưu hoá các thuật toán lượng tử cho ứng dụng cụ thể. Trong khi các triển khai QAOA mã nguồn mở tiêu chuẩn là điểm bắt đầu hữu ích, các phương pháp tinh chỉnh kỹ càng như **StrangeworksQAOA** có thể đạt kết quả tốt hơn trong một số trường hợp mà không tốn thêm chi phí tính toán hay tài chính.

Độ tin cậy và khả năng tái lập của kết quả trên bài toán xếp hàng hóa gợi mở một hướng đi khả quan. Khi tính toán lượng tử tiếp tục trưởng thành, các tối ưu hoá như vậy sẽ ngày càng quan trọng để đạt **lợi thế lượng tử thực tiễn** (practical quantum advantage) cho các ứng dụng thực tế.

Để khám phá các thuật toán này cho thách thức tối ưu của bạn, hãy truy cập nền tảng Strangeworks platform và tài liệu về QAOA để bắt đầu với Amazon Braket.

Tài liệu tham khảo

1. Romero, S., Osaba, E., Villar-Rodriguez, E., Oregi, I. & Ban, Y. *Hybrid approach for solving real-world bin packing problem instances using quantum annealers.* Sci Rep 13, 11777 (2023).

2. Farhi, E., Goldstone, J. & Gutmann, S. *A Quantum Approximate Optimization Algorithm.* arXiv:1411.4028 (2014).

3. Dupont, M. & Sundar, B. *Extending relax-and-round combinatorial optimization solvers with quantum correlations.* PhysRevA.109.012429 (2024).

4. Airbus Quantum Computing Challenge — https://www.airbus.com/sites/g/files/jlcbta136/files/2021-10/Airbus-QuantumComputing-Challenge-PS5.pdf. (truy cập 2023-07-24).

5. Pilon, G., Gugole, N. & Massarenti, N. *Aircraft Loading Optimization – QUBO models under multiple constraints.* arXiv:2102.09621 (2021).

6. Guerreschi, G. G. *Solving Quadratic Unconstrained Binary Optimization with divide-and-conquer and quantum algorithms.* arXiv:2101.07813v1 (2021).

7. Gurobi Optimization — https://www.gurobi.com/. (truy cập 2023-07-19).

8. QAOA Circuit Ansatz — https://qiskit.org/documentation/stubs/qiskit.circuit.library.RealAmplitudes.html (truy cập 2023-07-19).

9. Real Amplitude Quantum Circuit Ansatz — https://qiskit.org/documentation/stubs/qiskit.circuit.library.RealAmplitudes.html (truy cập 2023-08-08).

10. Braket QAOA Example — https://github.com/amazon-braket/amazon-braket-examples/blob/main/examples/pennylane/2_Graph_optimization_with_QAOA/2_Graph_optimization_with_QAOA.ipynb (truy cập 2023-10-23).

Stuart Flannigan

Stuart Flannigan is a Senior Quantum Software Engineer at Strangeworks, but his background is in analyzing quantum simulation experiments. Since joining Strangeworks, 3 years ago, he has been working on taking the theoretical ideas of quantum systems and combining them into classical and ai workflows, to make them more accessible to a wider industry audience.

Andrew J. Ochoa

Andrew J. Ochoa is Chief Scientist at Strangeworks, where he leads research and development initiatives and manages technical customer engagements in quantum computing and optimization. He holds a Ph.D. in Physics from Texas A&M University and an MBA from UT McCombs. Andrew has authored numerous publications in quantum computing and has been with Strangeworks since 2018 aligning scientific innovation with practical applications.

Michael Brett

Michael is a Principal Specialist for Quantum Computing in the High Performance Computing group at Amazon Web Services (AWS). In this role, he leads global business development and go to market efforts for Amazon Braket, a fully-managed quantum computing service in the cloud at AWS. He was previously SVP for Applications at Rigetti Computing, a quantum computing hardware company based in Berkeley, and CEO of QxBranch, a quantum computing applications start-up company acquired by Rigetti in 2019. Michael has a background in probabilistic risk modelling for aerospace applications and studied a Bachelor of Engineering in Aerospace Avionics and an Executive Master of Business in Complex Project Management, both from the Queensland University of Technology in Brisbane, Australia.

Charunethran Panchalam Govindarajan

Charunethran Panchalam Govindarajan is a Sr. Product Marketing Manager at AWS, focused on High-Performance Computing and Quantum Technologies. He has worked across a broad range of technology domains, with a core interest in intersection of R&D and product development. Charunethran holds a Master's degree in Electrical Engineering from Stanford University. Outside of work, he enjoys sketching and philosophical conversations.