Title: AI methods for quantum circuit transpiling and optimization

Speakers: David Kremer Garcia

Collection/Series: Special Seminars

Subject: Other

Date: October 11, 2024 - 9:30 AM

URL: https://pirsa.org/24100065

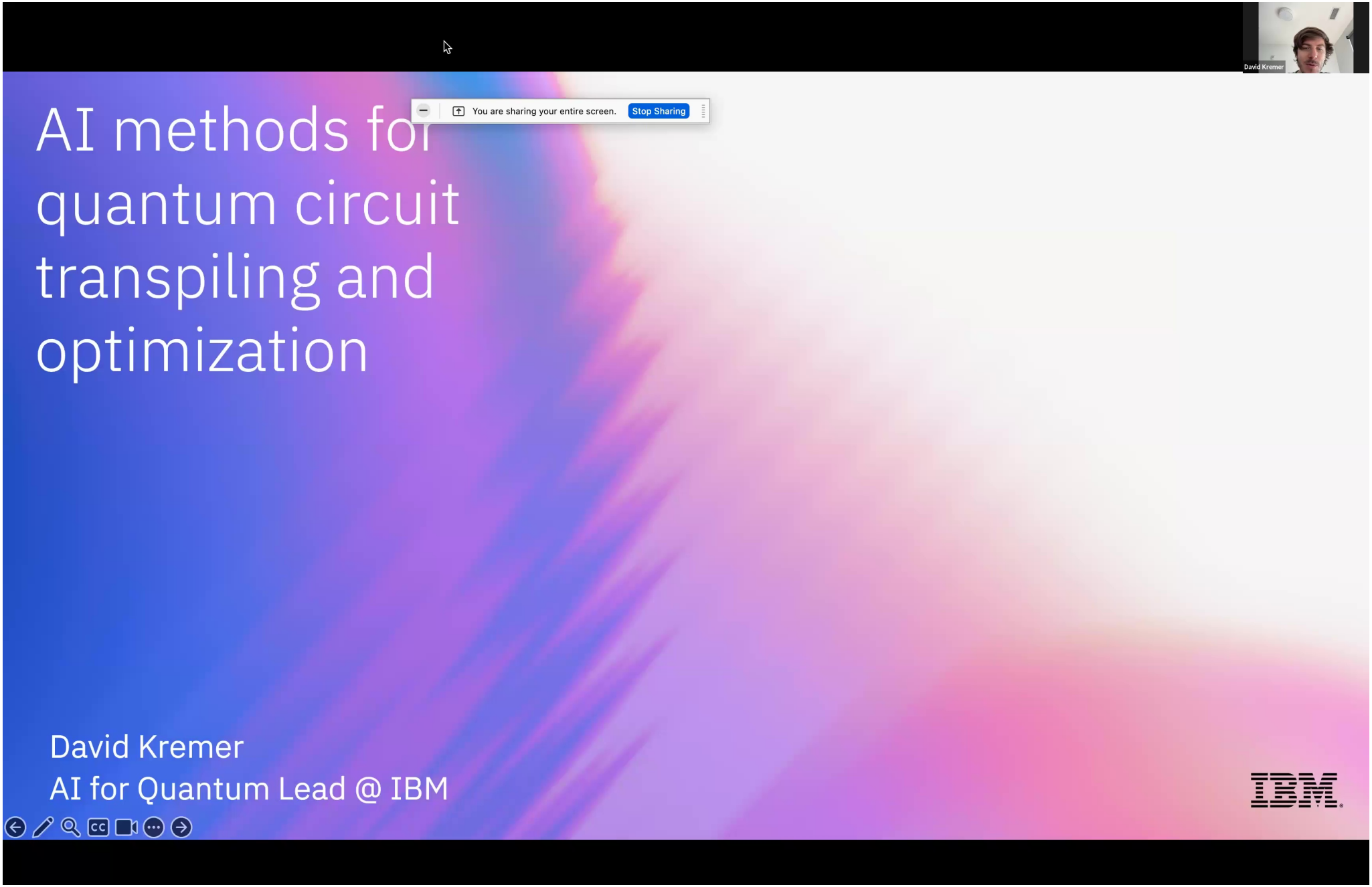
Abstract:

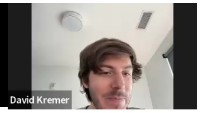
The 2nd talk of a monthly webinar series jointly hosted by Perimeter, IVADO, and Institut Courtois.

Speaker: David Kremer Garcia, AI Engineer & Lead Data Scientist, IBM Quantum, Yorktown Heights, NY, USA.

In this session, I will talk about how we are using AI to improve quantum circuit transpiling and optimization. I will show some of our recent work, where we apply AI methods such as Reinforcement Learning to different transpiling tasks and achieve a remarkable balance between speed and quality of the results. I will also talk about how we integrate these methods with other heuristics to provide "ai-enhanced transpiling" through our Qiskit Transpiler Service.


Emily Petroff
SLIDE_ZoomWaitingRoom.pptx.pdf
1 page
Show the Inspector
Search
Bienvenue à notre série de séminaires IA & le Quantique
Ces conférences font partie d'une collaboration entre IVADO, l'Institut Perimeter et l'Institut Courtois, visant à accélérer les découvertes de nouveaux matériaux et molécules en intégrant l'IA avec la physique quantique à travers la recherche, les activités scientifiques et le transfert de connaissances.
Welcome to our Quantum & AI Lecture Series
These lectures are part of a collaboration between IVADO, Perimeter Institute, and Institut Courtois, aimed at accelerating discoveries in new materials and molecules by integrating AI with quantum physics through research, scientific activities, and knowledge transfer.
IVADO
Institut Courtois
PERIMETER INSTITUTE


David Kremer
You are sharing your entire screen.
Stop Sharing
AI methods for quantum circuit transpiling and optimization
David Kremer
AI for Quantum Lead @ IBM
IBM



Transpiling with AI

IBM **Quantum**

I will talk about our recent results on transpiling with AI:

- Transpiling recap
- Why use AI?
- What is Reinforcement Learning?
- Circuit synthesis with AI
- Circuit routing with AI
- Qiskit Transpiler Service

Practical and efficient quantum circuit synthesis and transpiling with Reinforcement Learning

David Kremer,[1,*] Victor Villar,[1] Hanhee Paik,[1] Ivan Duran,[1] Ismael Faro,[1] and Juan Cruz-Benito[1,†]
[1]*IBM Quantum, IBM T.J. Watson Research Center, Yorktown Heights, NY 10598*

This paper demonstrates the integration of Reinforcement Learning (RL) into quantum transpiling workflows, significantly enhancing the synthesis and routing of quantum circuits. By employing RL, we achieve near-optimal synthesis of Linear Function, Clifford, and Permutation circuits, up to 9, 11 and 65 qubits respectively, while being compatible with native device instruction sets and connectivity constraints, and orders of magnitude faster than optimization methods such as SAT solvers. We also achieve significant reductions in two-qubit gate depth and count for circuit routing up to 133 qubits with respect to other routing heuristics such as SABRE. We find the method to be efficient enough to be useful in practice in typical quantum transpiling pipelines. Our results set the stage for further AI-powered enhancements of quantum computing workflows.

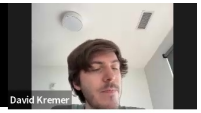
I. INTRODUCTION

The field of quantum computing has grown into a community that includes domain experts in a wide range of various disciplines such as chemistry and Artificial Intelligence (AI) from industry, government and academia. With the rapidly developing technologies, quantum computing is now accessible by users via cloud where users to industry scale problems on the order of tens of thousands of qubits [4].

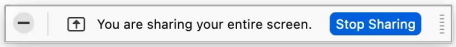
The integration of AI tools is considered a critical path to advance in several scientific fields [5–11]. In particular, we observe AI has the potential to help leverage the full potential of quantum computing [12]. As the sizes of quantum circuits input to quantum computing systems continue to grow in the era of quantum utility, there is a need for smarter and more effective methods to enhance the quantum computation workflow. The integration of classical computing resources, such as GPUs and CPUs, with QPUs will pave the way for new frontiers in computing, and the collaboration between AI and quantum computing will be critical in realizing this potential.

Quantum circuit transpilation and optimization are central components of quantum computing workflows. Similar to compilers in classical computing, transpilers map logical quantum circuits to the instructions present physical quantum devices, and allow quantum circuit developers to focus on the quantum algorithms rather than specific details of the hardware. High-quality transpilation of quantum circuits (minimizing the overhead introduced in this mapping) is an important goal for the field of quantum computing in general, but especially relevant for near-term quantum computing hardware, where even small improvements on the transpiled circuit sizes can lead to important reductions in the noise present in the results.

https://arxiv.org/abs/2405.13196



Transpiling with AI



IBM **Quantum**

What

We want to provide “high quality” circuits in “reasonable” time in the form of Qiskit SDK Transpiler passes.

How

We apply Reinforcement Learning to different problems in Transpiling.

Why

RL can help us find a good balance between optimality, speed and storage.

Practical and efficient quantum circuit synthesis and transpiling with Reinforcement Learning

David Kremer,[1,*] Victor Villar,[1] Hanhee Paik,[1] Ivan Duran,[1] Ismael Faro,[1] and Juan Cruz-Benito[1,†]
[1]*IBM Quantum, IBM T.J. Watson Research Center, Yorktown Heights, NY 10598*

This paper demonstrates the integration of Reinforcement Learning (RL) into quantum transpiling workflows, significantly enhancing the synthesis and routing of quantum circuits. By employing RL, we achieve near-optimal synthesis of Linear Function, Clifford, and Permutation circuits, up to 9, 11 and 65 qubits respectively, while being compatible with native device instruction sets and connectivity constraints, and orders of magnitude faster than optimization methods such as SAT solvers. We also achieve significant reductions in two-qubit gate depth and count for circuit routing up to 133 qubits with respect to other routing heuristics such as SABRE. We find the method to be efficient enough to be useful in practice in typical quantum transpiling pipelines. Our results set the stage for further AI-powered enhancements of quantum computing workflows.

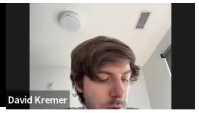
I. INTRODUCTION

The field of quantum computing has grown into a community that includes domain experts in a wide range of various disciplines such as chemistry and Artificial Intelligence (AI) from industry, government and academia. With the rapidly developing technologies, quantum computing is now accessible by users via cloud where users to industry scale problems on the order of tens of thousands of qubits [4].

The integration of AI tools is considered a critical path to advance in several scientific fields [5–11]. In particular, we observe AI has the potential to help leverage the full potential of quantum computing [12]. As the sizes of quantum circuits input to quantum computing systems continue to grow in the era of quantum utility, there is a need for smarter and more effective methods to enhance the quantum computation workflow. The integration of classical computing resources, such as GPUs and CPUs, with QPUs will pave the way for new frontiers in computing, and the collaboration between AI and quantum computing will be critical in realizing this potential.

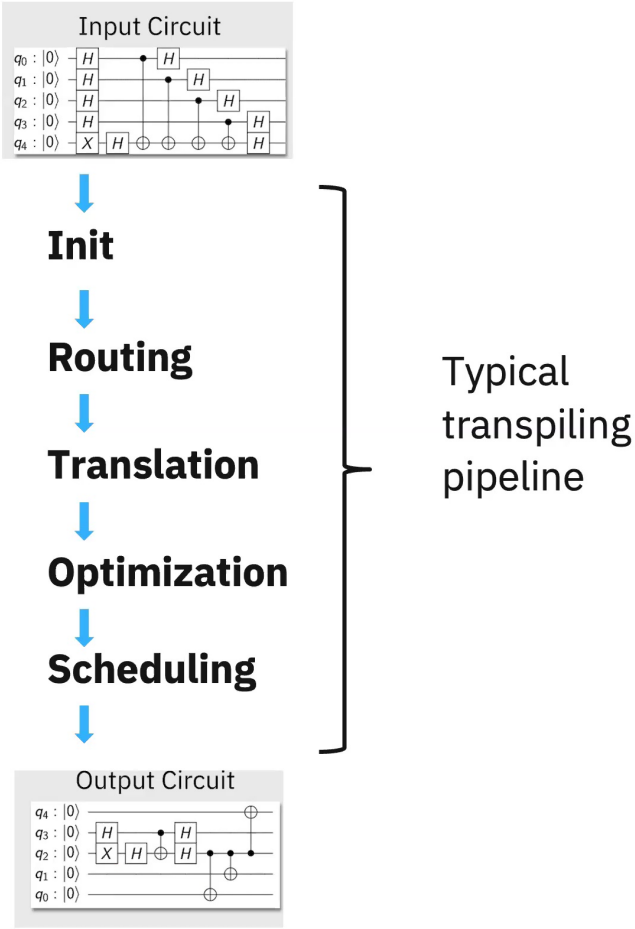
Quantum circuit transpilation and optimization are central components of quantum computing workflows. Similar to compilers in classical computing, transpilers map logical quantum circuits to the instructions present physical quantum devices, and allow quantum circuit developers to focus on the quantum algorithms rather than specific details of the hardware. High-quality transpilation of quantum circuits (minimizing the overhead introduced in this mapping) is an important goal for the field of quantum computing in general, but especially relevant for near-term quantum computing hardware, where even small improvements on the transpiled circuit sizes can lead to important reductions in the noise present in the results.

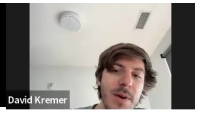
https://arxiv.org/abs/2405.13196



Recap: what is *Transpilation*?

"Transpilation is the process of rewriting a given input circuit to match the topology of a specific quantum device, and/or to optimize the circuit for execution on present day noisy quantum systems."

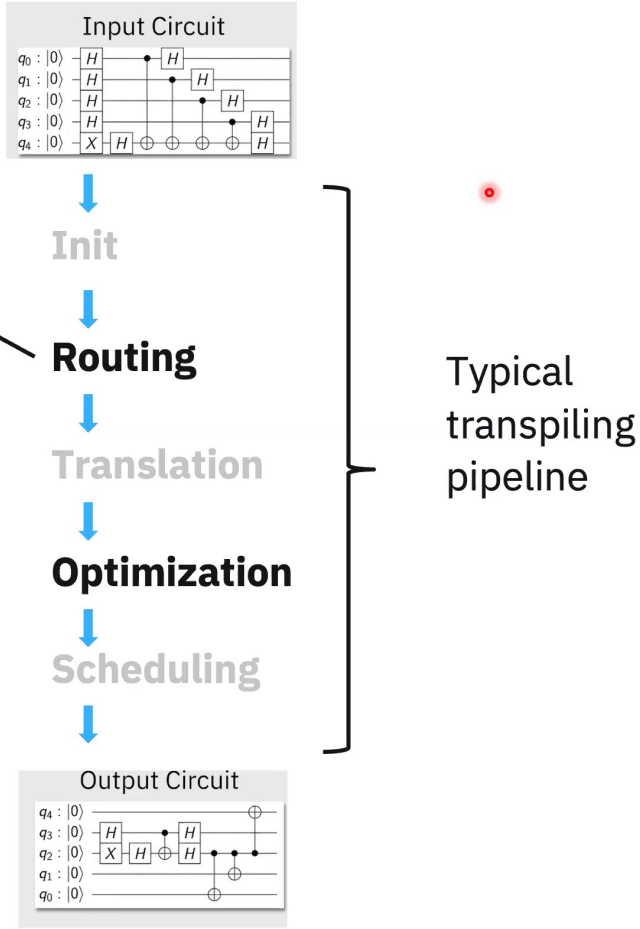


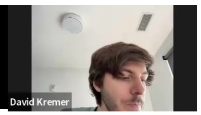


We have AI Passes for Routing and Synthesis

IBM **Quantum**

Routing: *Insert SWAPs to make a circuit compatible with a given coupling map.*



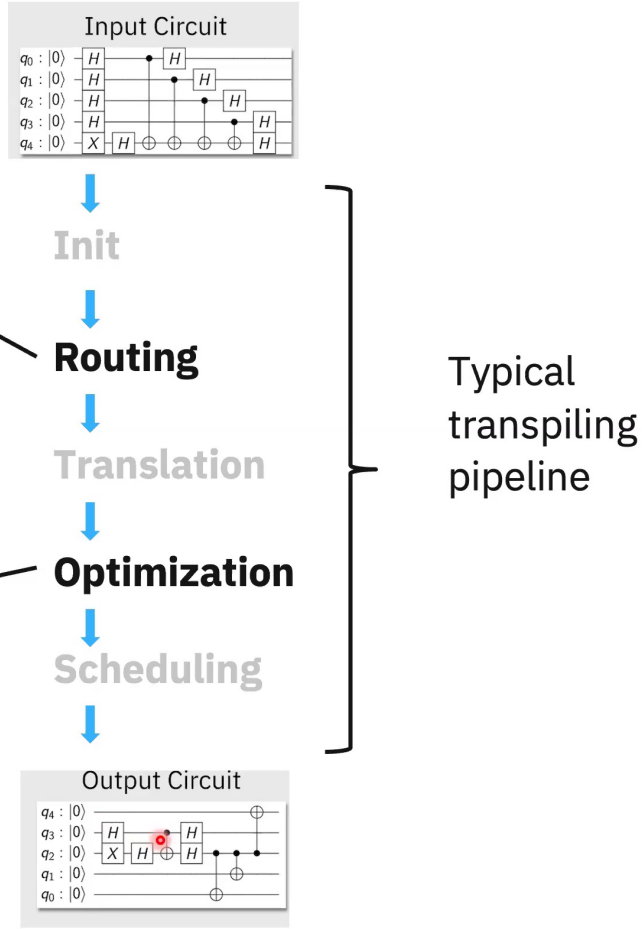


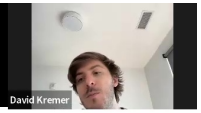
We have AI Passes for Routing and Synthesis

IBM **Quantum**

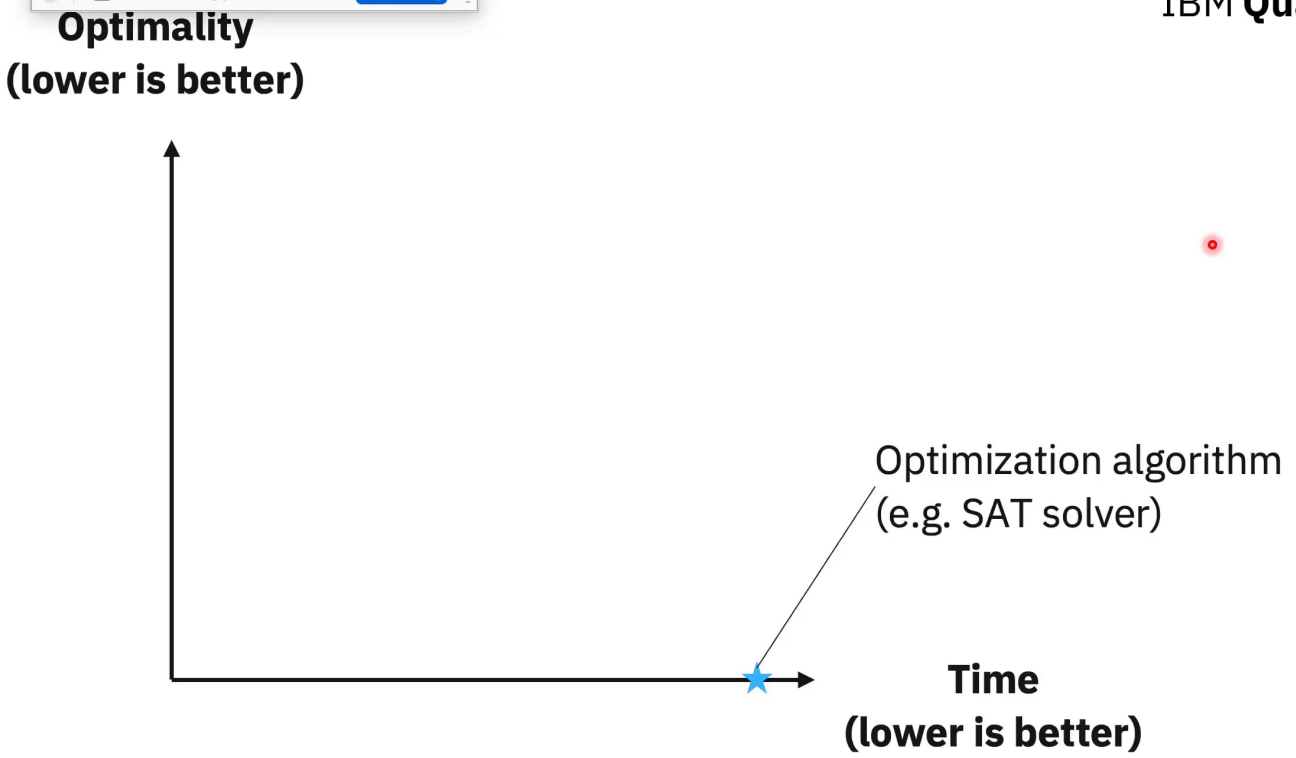
Routing: *Insert SWAPs to make a circuit compatible with a given coupling map.*

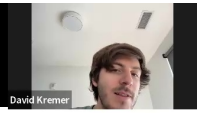
Circuit Synthesis: *Generate a circuit that implements a given operator. Typically used in optimization by re-synthesizing parts of a circuit to achieve better CNOT count and/or depth.*



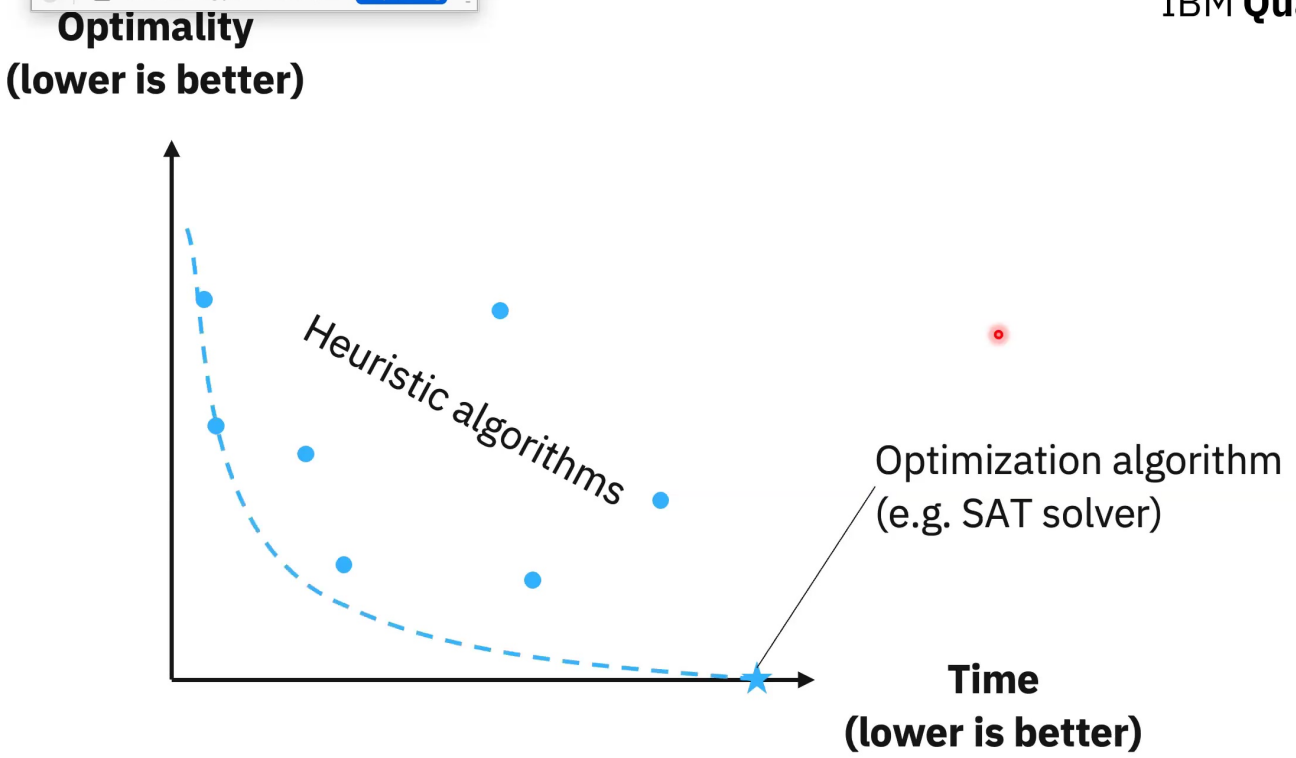


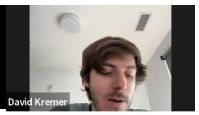
Why with AI?





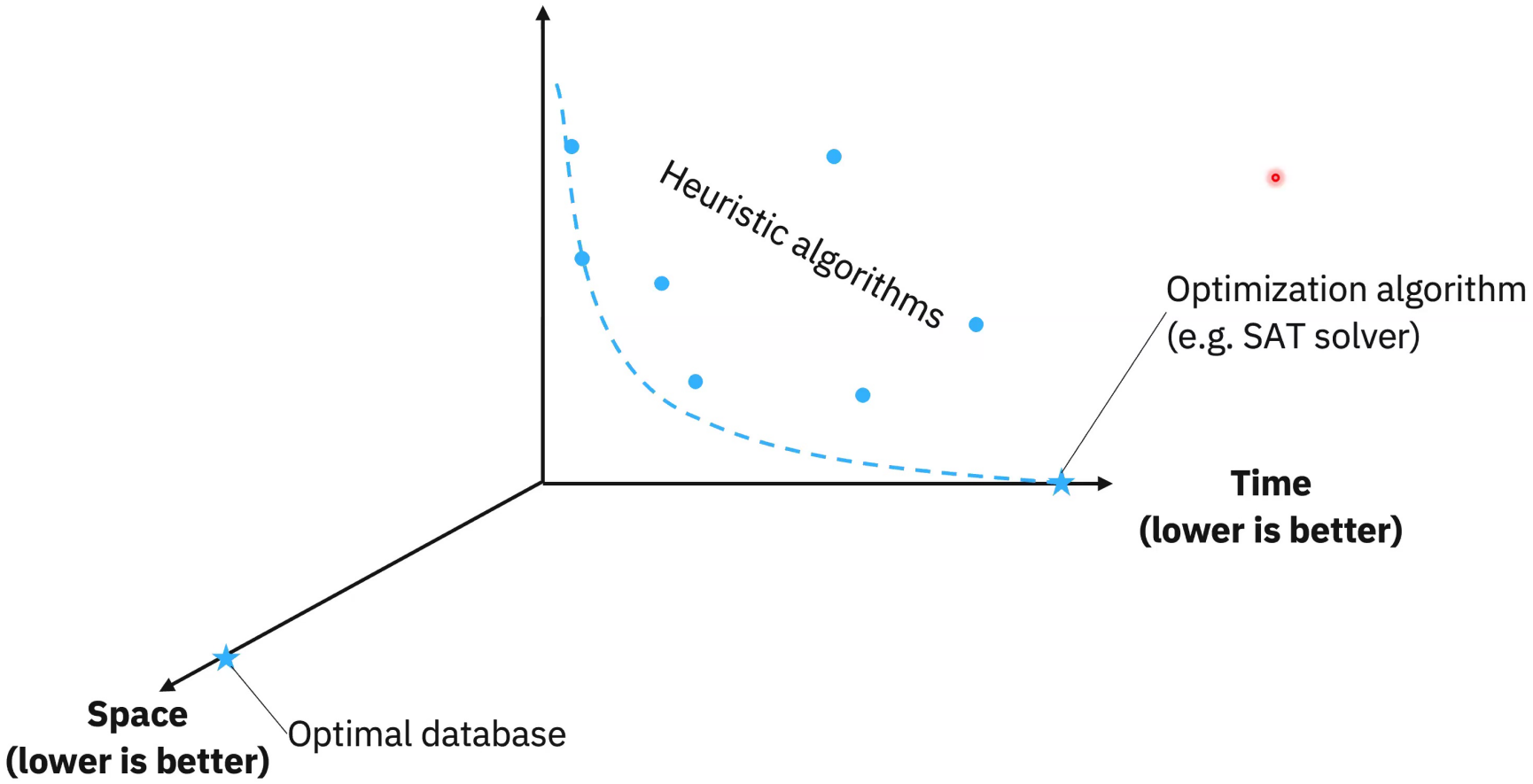
Why with AI?

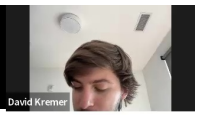




Why with AI?

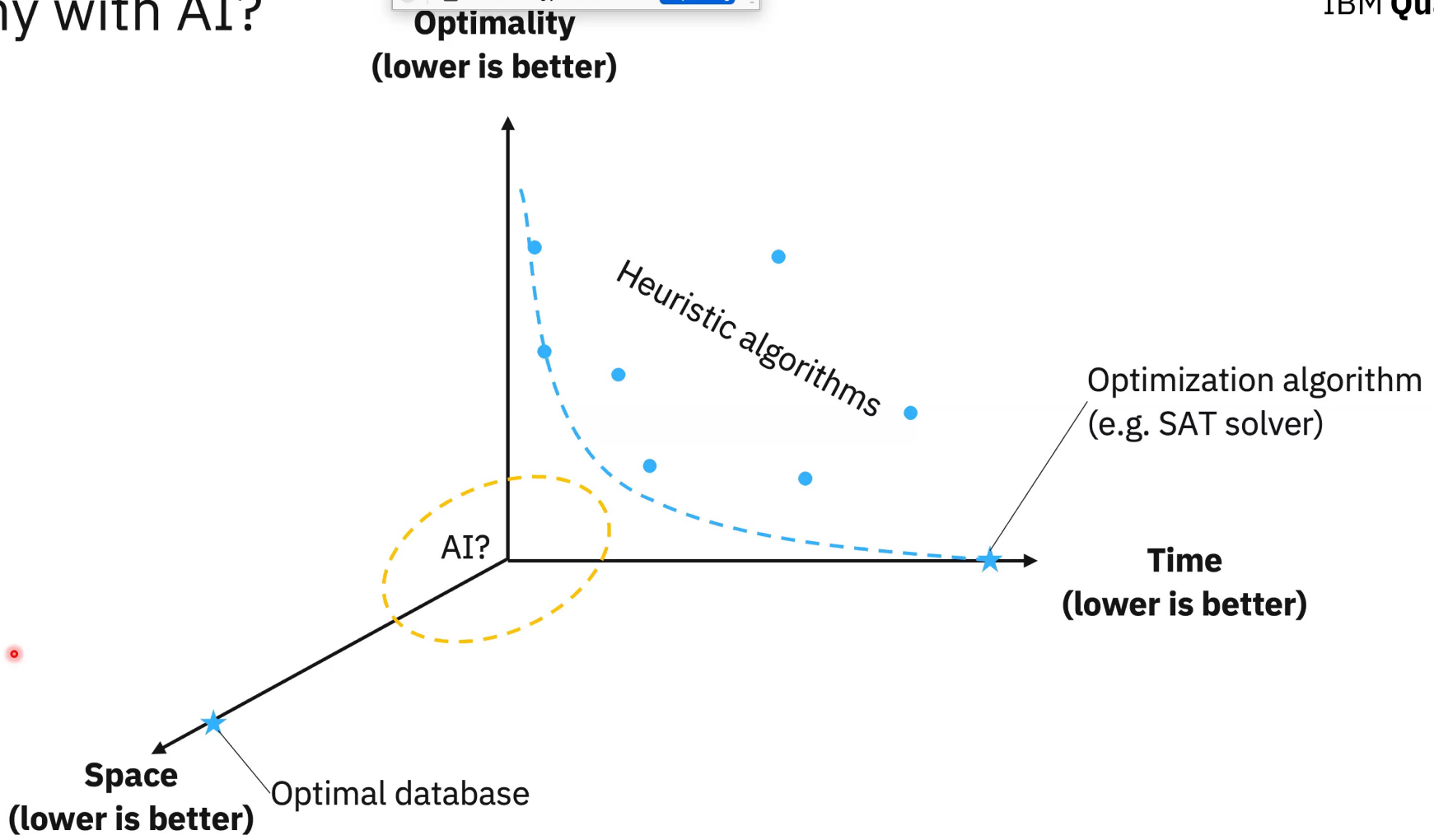
IBM **Quantum**





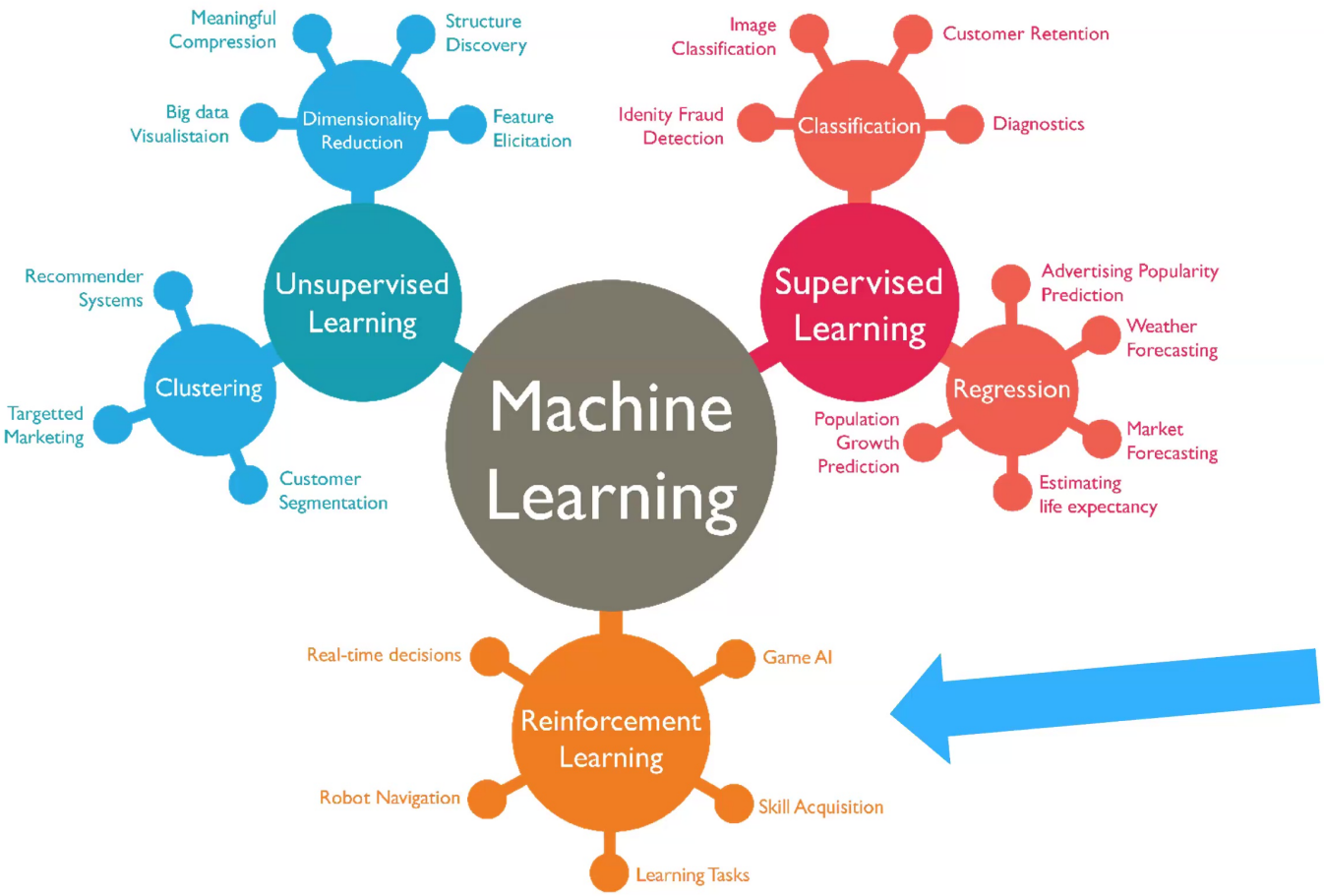
Why with AI?

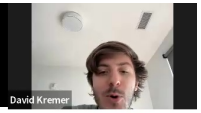
IBM **Quantum**



What is *Reinforcement Learning*?

IBM **Quantum**



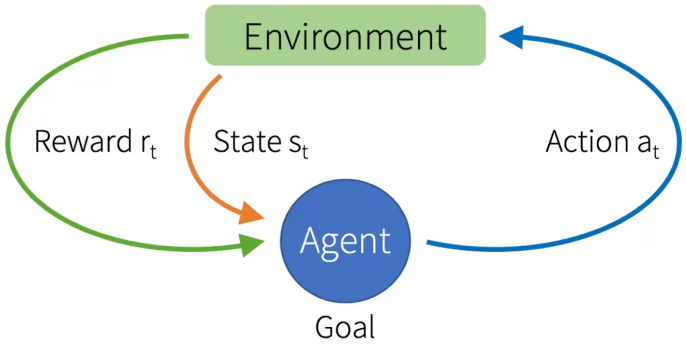


What is *Reinforcement Learning*?

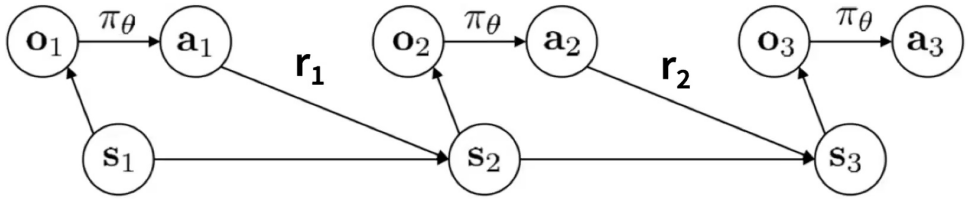
Reinforcement Learning is a family of algorithms that has the objective of learning an optimal *policy* by interacting with an environment.

RL algorithms choose the best actions to apply in a given situation based on observations from the environment.

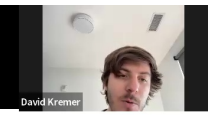
The way these algorithms are guided through the learning is by providing a reward function that rewards or penalizes actions based on their outcomes.



s: state
o: observation
π: policy
a: action
r: reward



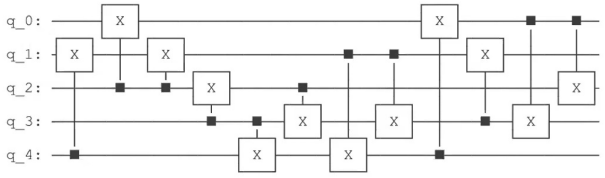
Partially Observed Markov Decision Process (POMDP)



Recap: Circuit Synthesis

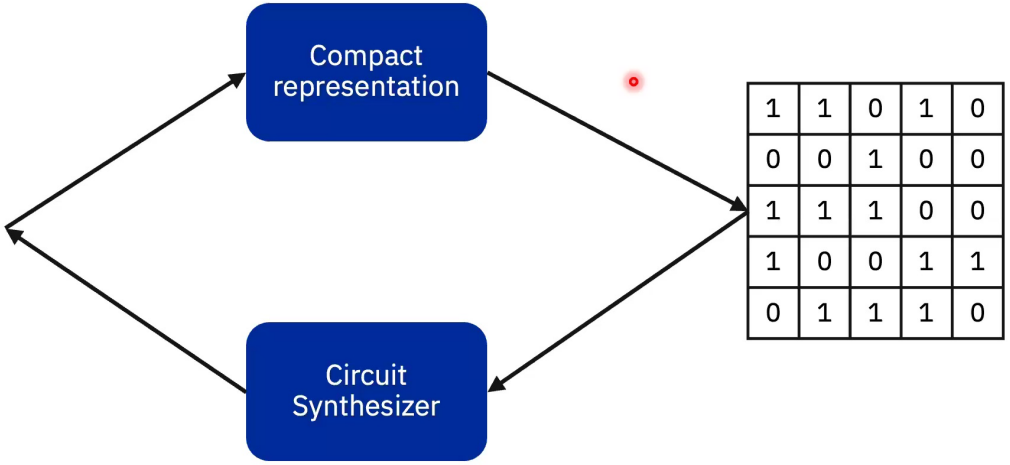
IBM **Quantum**

Quantum circuit



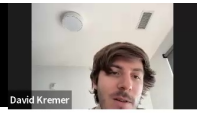
Operator

N to 1, usually easy



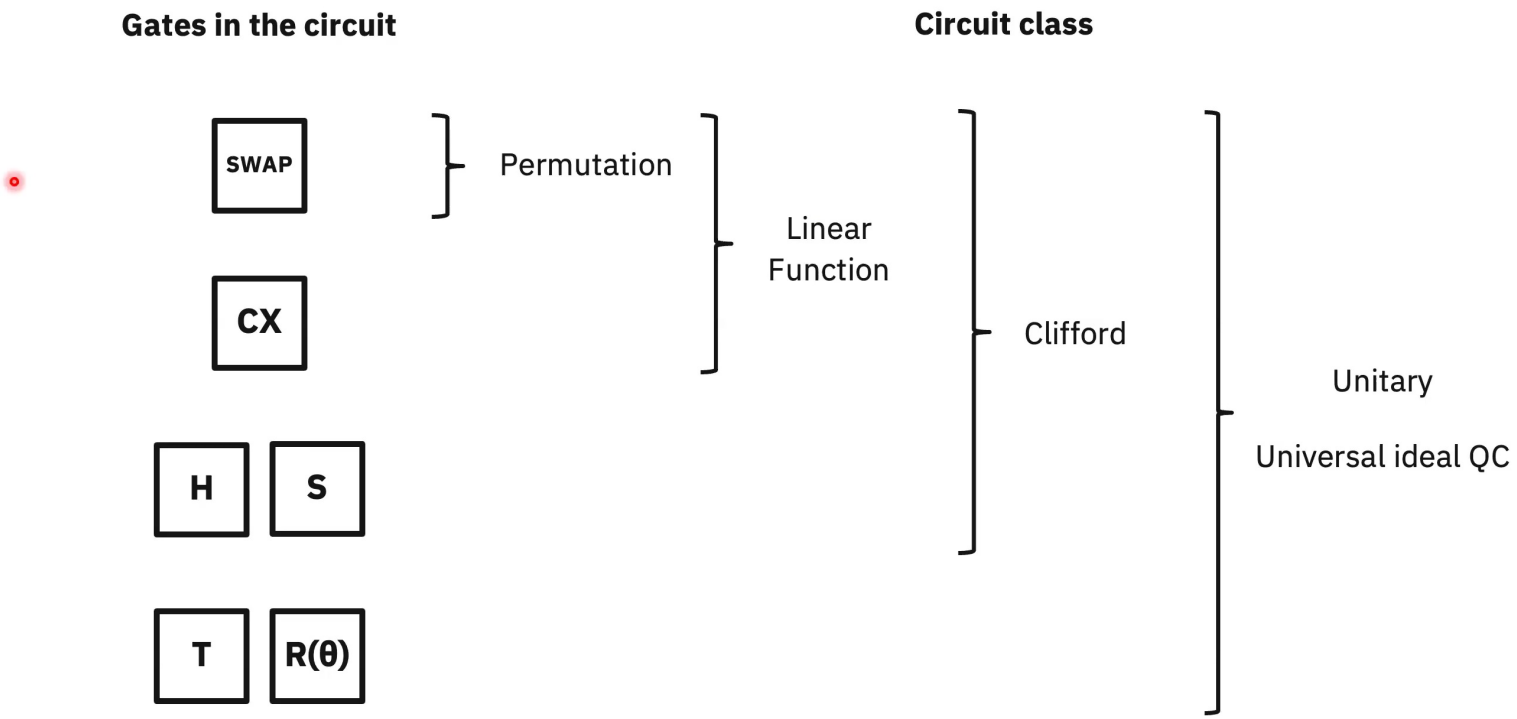
1 to N, usually hard

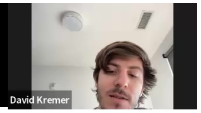
Especially trying to find small circuits given a representation



Recap: Circuit Synthesis

IBM **Quantum**

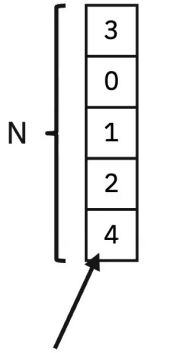




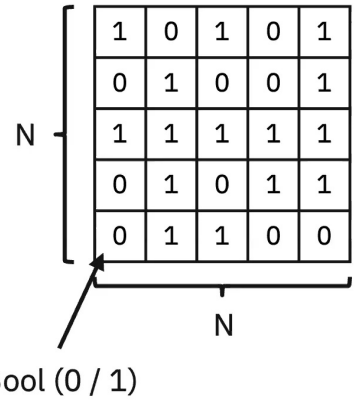
Recap: Circuit Synthesis

IBM **Quantum**

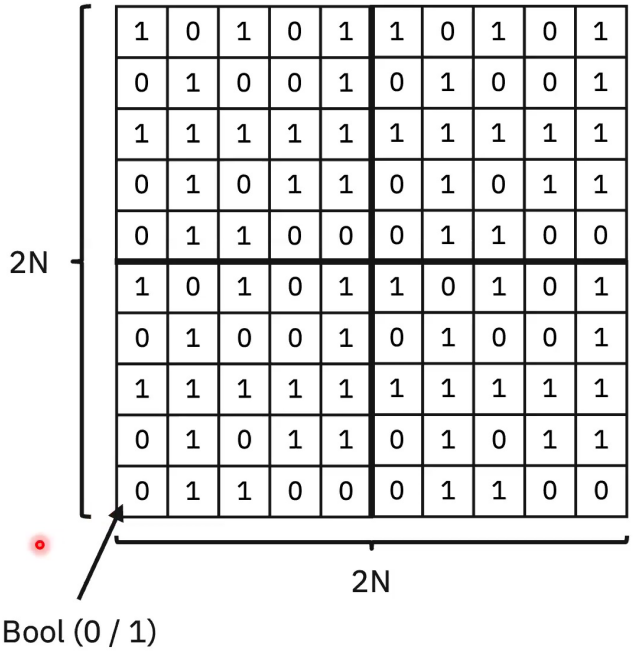
Permutation

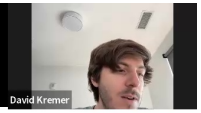


Linear Function



Clifford

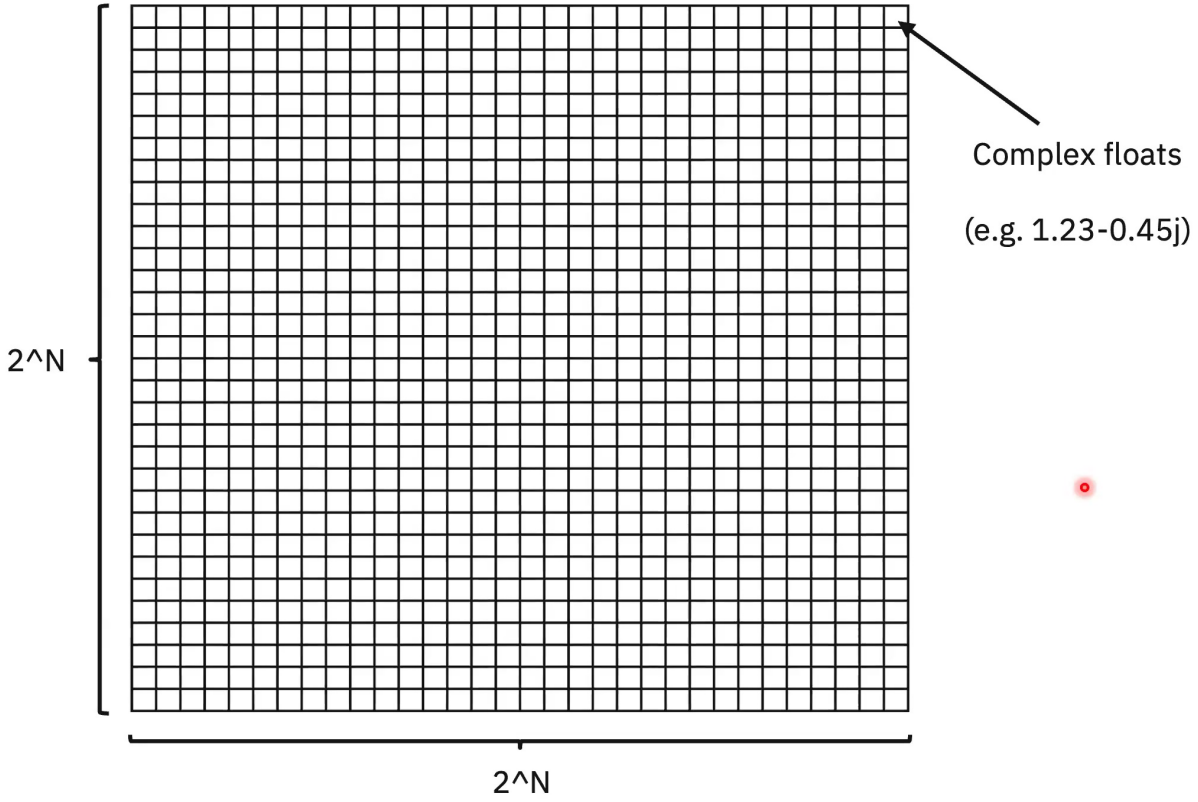


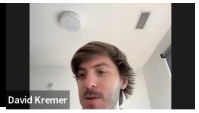


Recap: Circuit Synthesis

IBM **Quantum**

Unitary





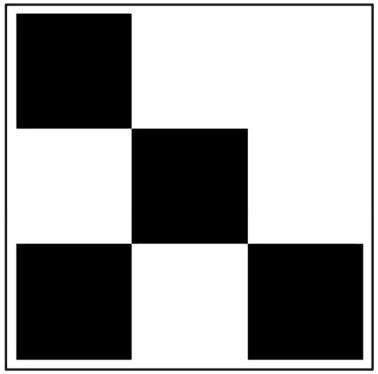
The Circuit Synthesis Game

IBM **Quantum**

Rules:

- The game starts at a given operator
- It proceeds by steps, where you choose a gate at each step
- You can only place CNOTs and SWAPs at given locations (e.g. linear connectivity)
- The goal is to arrive to the identity operator (a diagonal matrix)
- You get -1 points for each CNOT, -3 for each SWAP, and 100 if you reach the goal.

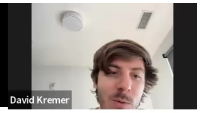
Operator



Allowed gates

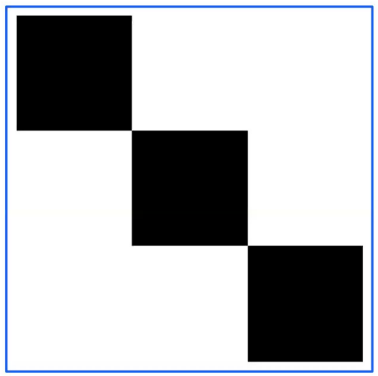
- CX (0,1)/(1,0)
- CX (1,2)/(2,1)

- SWAP (0,1)
- SWAP (1,2)



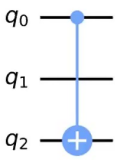
The Circuit Synthesis Game

Identity

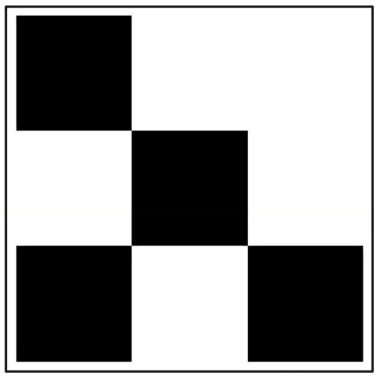


Scramble!

CX (0,2)



Operator



Allowed gates

- CX (0,1)/(1,0)
- CX (1,2)/(2,1)

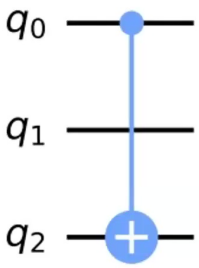
- SWAP (0,1)
- SWAP (1,2)

It is just like trying to solve a fancy Rubik's cube!

The Circuit Synthesis Game

IBM **Quantum**

Original Circuit

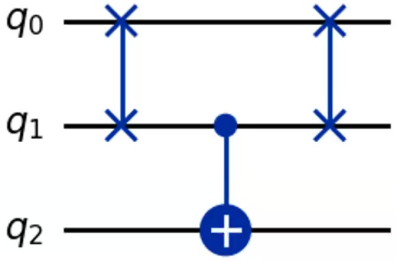


Just 1 CNOT!

Does not respect the coupling map

=

Synthesized Circuit V1

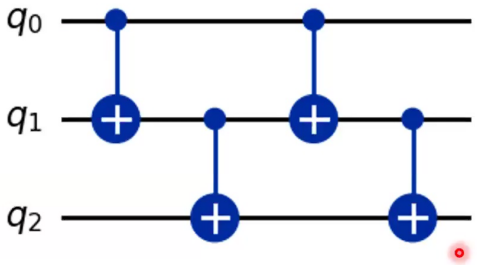


Score: 93
(7 CNOTs)

Respects the coupling map

=

Synthesized Circuit V2



Score: 96
(4 CNOTs)

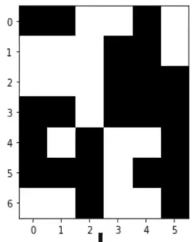
Respects the coupling map

Optimal CNOT count!

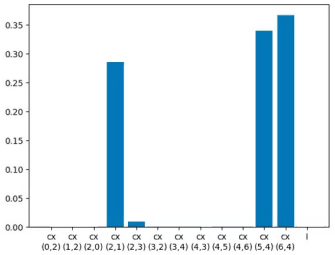
Circuit Synthesis with RL

IBM **Quantum**

Operator

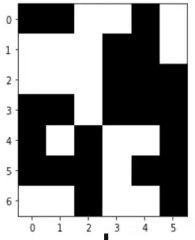


RL Neural Network



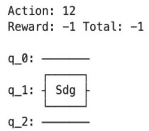
Distribution over possible actions (gates)

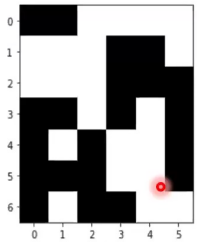
Operator **Best next gate** **Current Circuit**



RL Neural Network

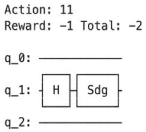
Sdg(1)

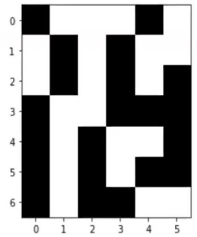




RL Neural Network

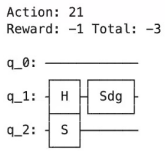
H(1)



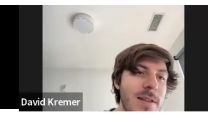


RL Neural Network

S(2)

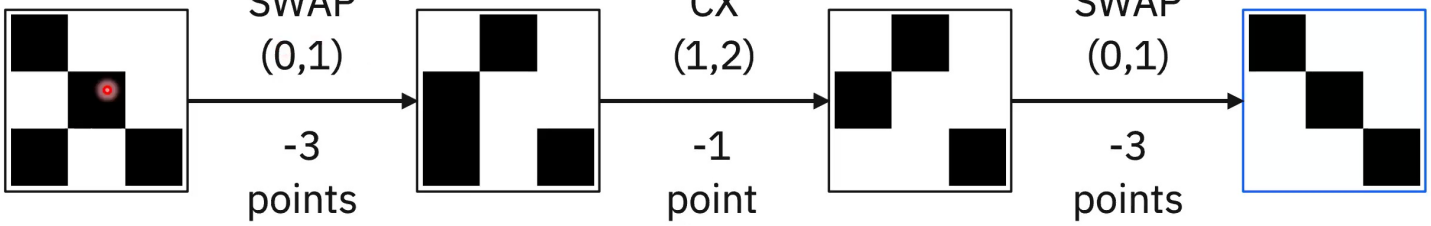


...

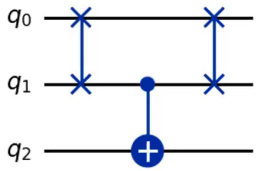


The Circuit Synthesis Game

IBM **Quantum**

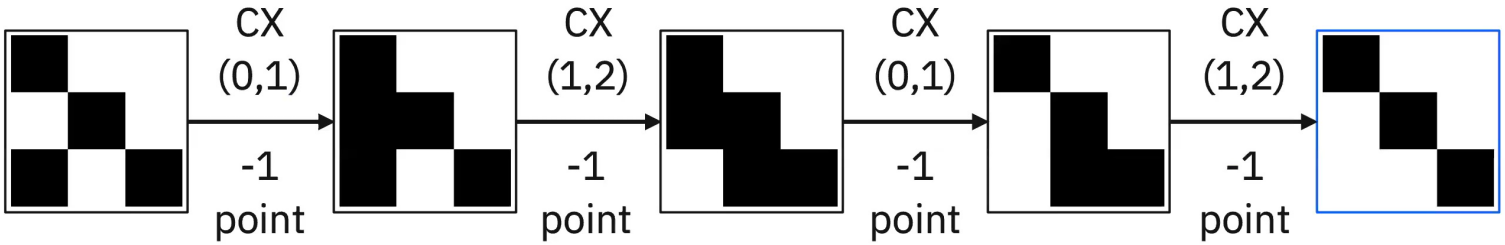


You win!!

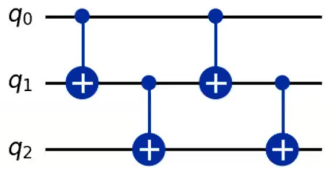


Score: 93

Can you do better?

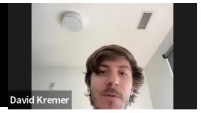


You win!!



Score: 96

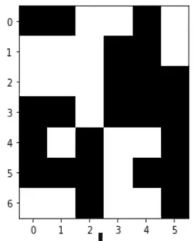
New high score!!



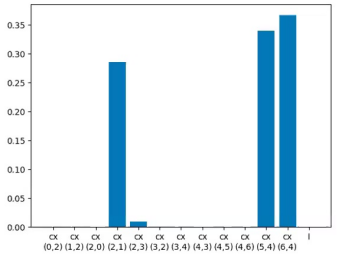
Circuit Synthesis with RL

IBM **Quantum**

Operator

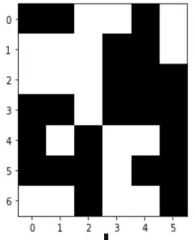


RL Neural Network

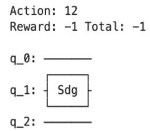


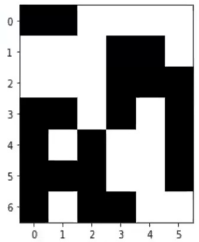
Distribution over possible actions (gates)

Operator | **Best next gate** | **Current Circuit**

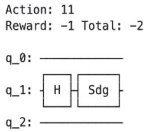


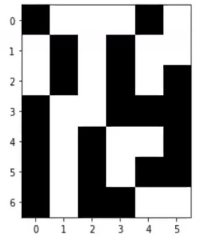
RL Neural Network → Sdg(1)



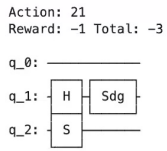


RL Neural Network → H(1)

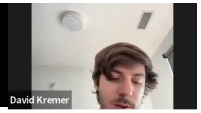




RL Neural Network → S(2)



...



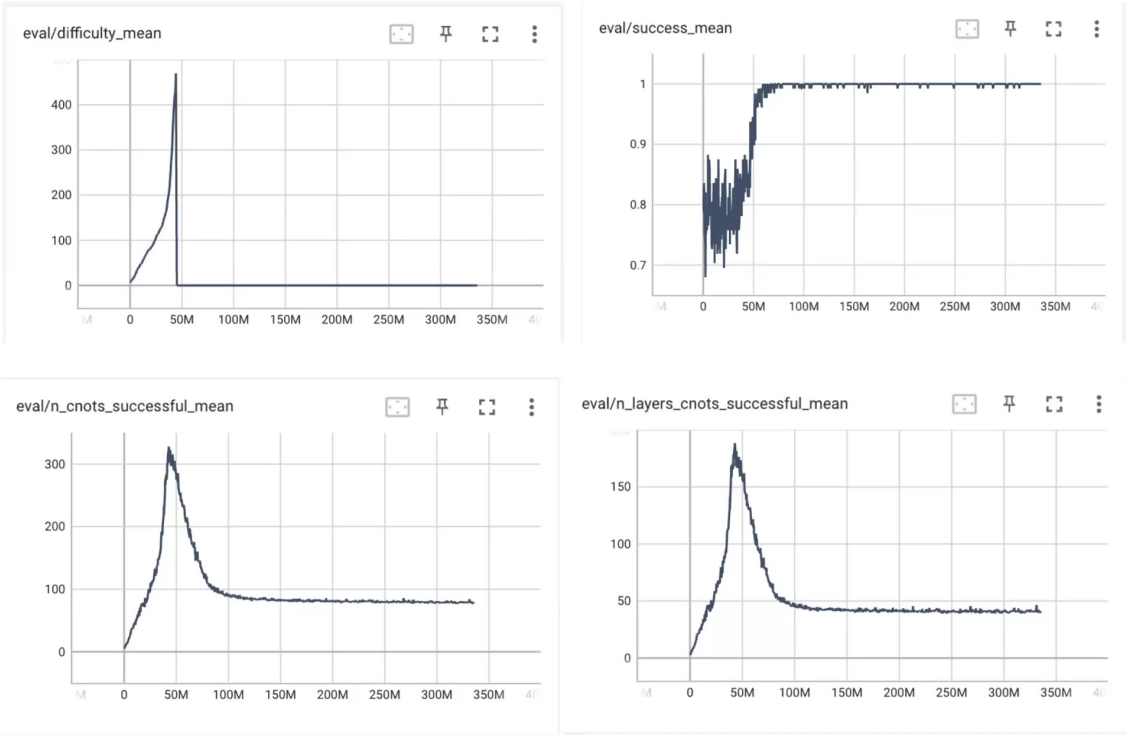
But how do we train these networks?

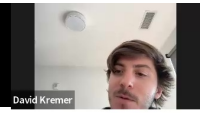
IBM **Quantum**

- We start from a random init network (no previous knowledge or dataset)
- We let the network synthesize circuits of increased difficulty.
- We provide rewards:
  - Positive when a target is reached.
  - Negative for each gate / layer.
- We update the weights according to the RL algorithm (PPO in this case) based on rewards.

Example training for linear functions

(9 qubits linear connectivity)

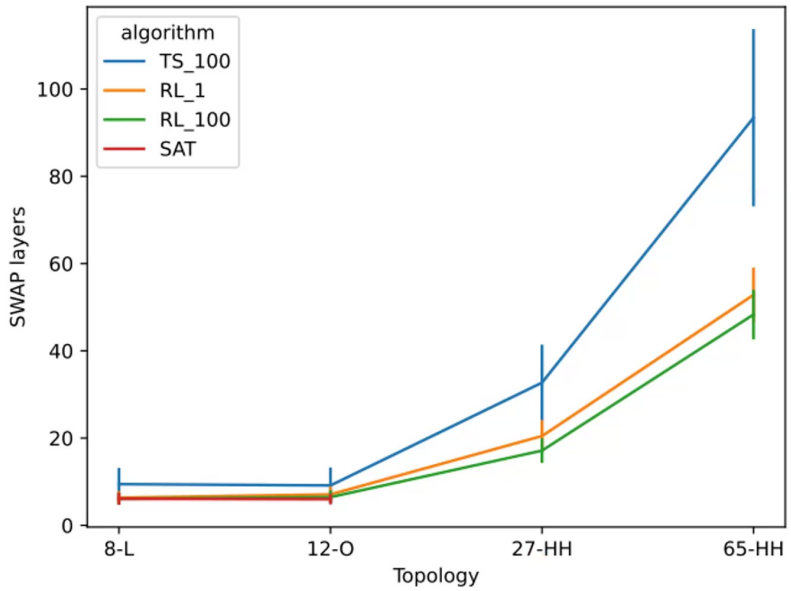




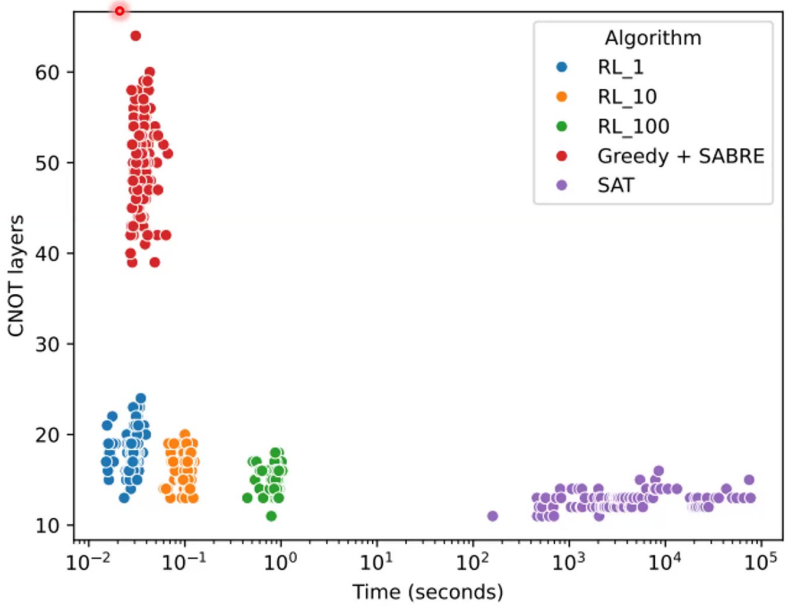
Circuit Synthesis with RL – Some results

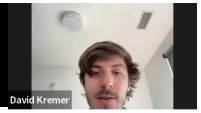
IBM **Quantum**

Synthesis of Permutations



Synthesis 6 qubit Cliffords on linear connectivity



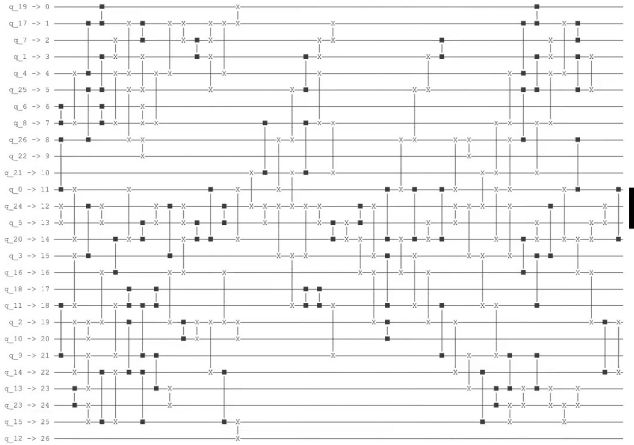


Optimizing Circuits with Re-Synthesis

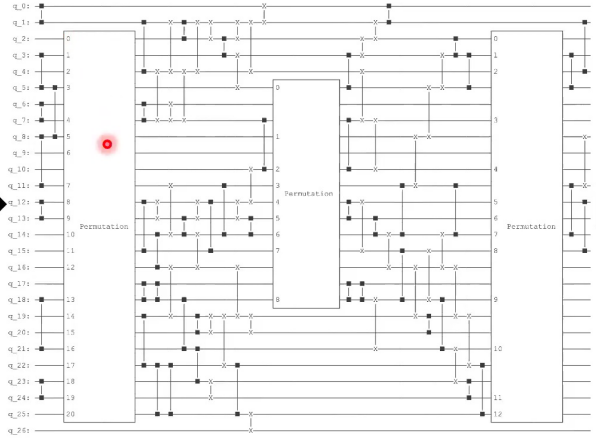
IBM **Quantum**

Example of a SWAP optimization pipeline

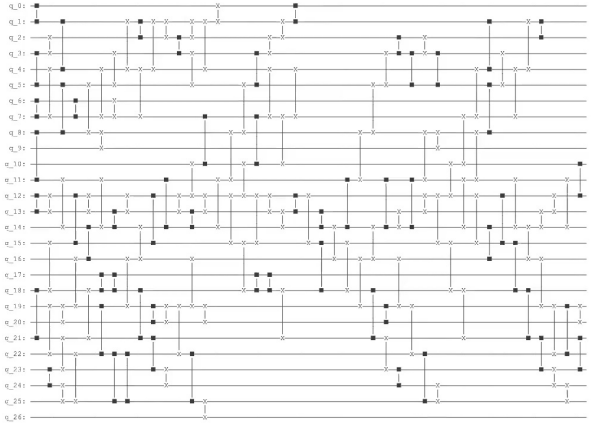
Transpiled (QiskitSDK)
254 swaps, 238 cnot layers

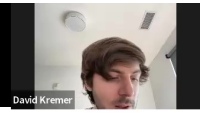


Collect large permutations
~15 permutations in the circuit



Re-synthesize permutations with RL
254 swaps, 222 cnot layers

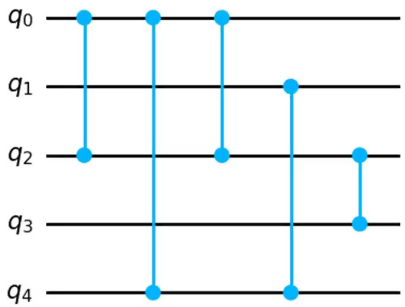




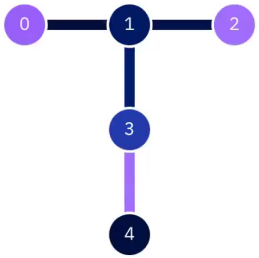
Recap: Circuit Routing

IBM **Quantum**

Circuit

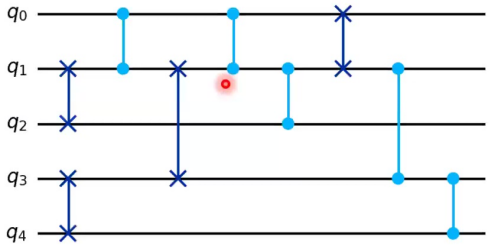


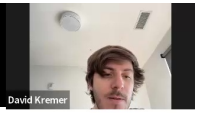
Coupling map



Routing: Make the circuit compatible with the coupling map by inserting SWAPs

Routed Circuit





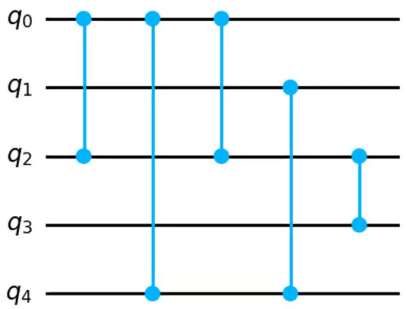
The Routing Game

IBM **Quantum**

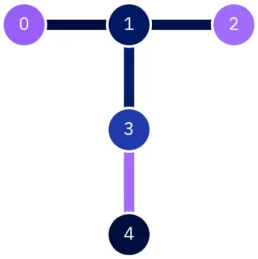
Rules:

- The game starts with an input circuit and an empty output circuit
- It proceeds by steps, where you choose a SWAP to apply at each step (with -1 point)
- SWAPs are allowed only on the pairs of qubits connected in the coupling map
- If an operation on the input circuit is allowed by the coupling map, it passes from the input to the output circuit
- The game finishes when the input circuit gets empty (with +100 points)

Circuit



Coupling map



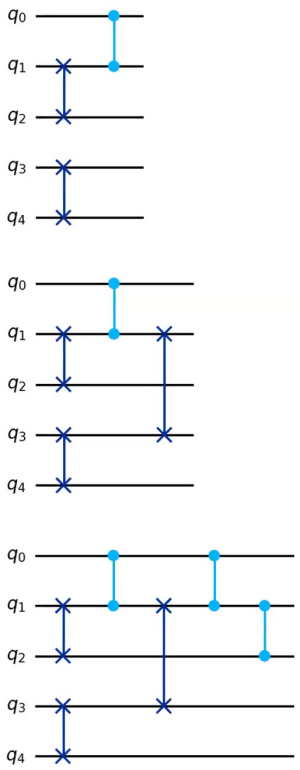
The Routing Game

IBM **Quantum**

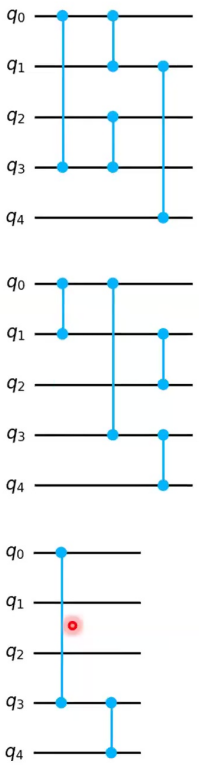
Step 3

Score: -3

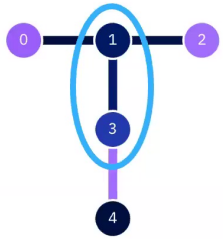
Output Circuit



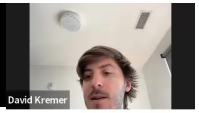
Input Circuit



SWAP (1,3)



Clear valid gates

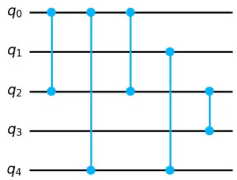


Circuit Routing with RL

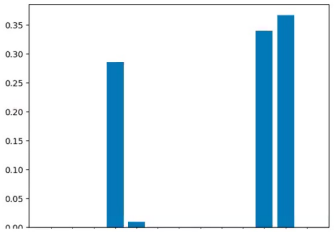
You are sharing your entire screen. Stop Sharing

IBM **Quantum**

Input Circuit

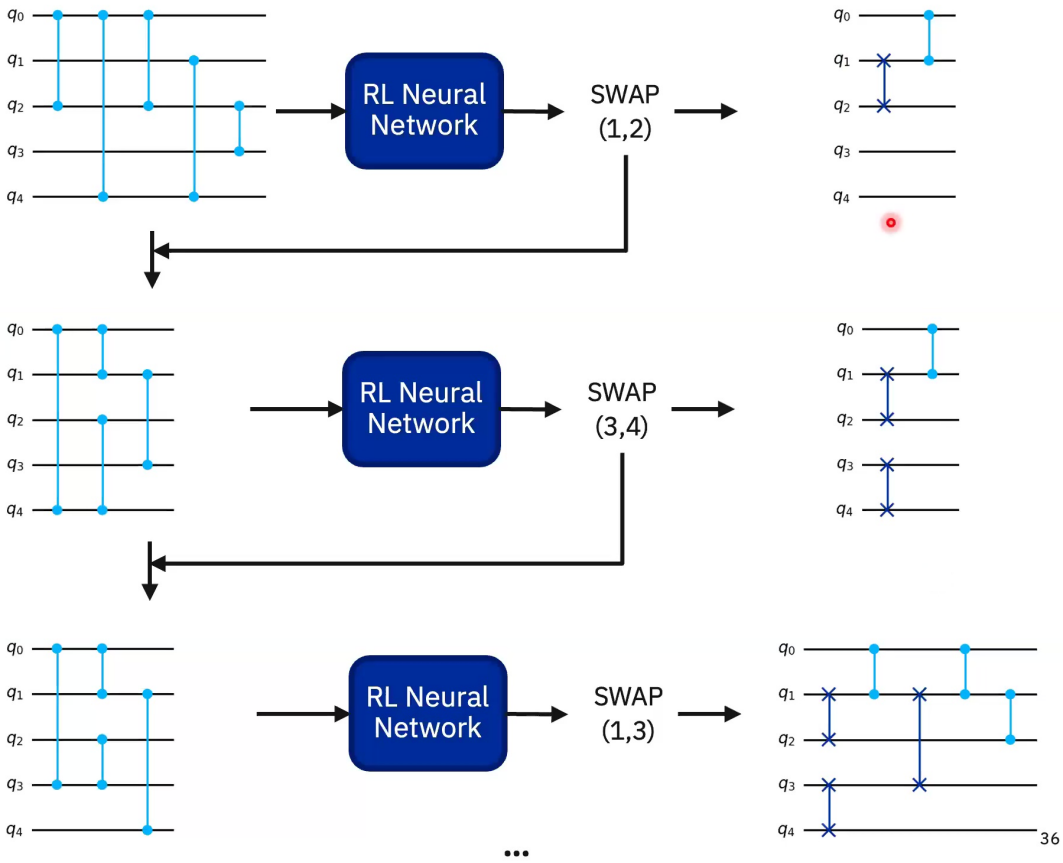


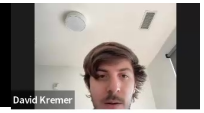
RL Neural Network



Distribution over possible actions (SWAPs)

Input Circuit **Best next SWAP** **Output Circuit**

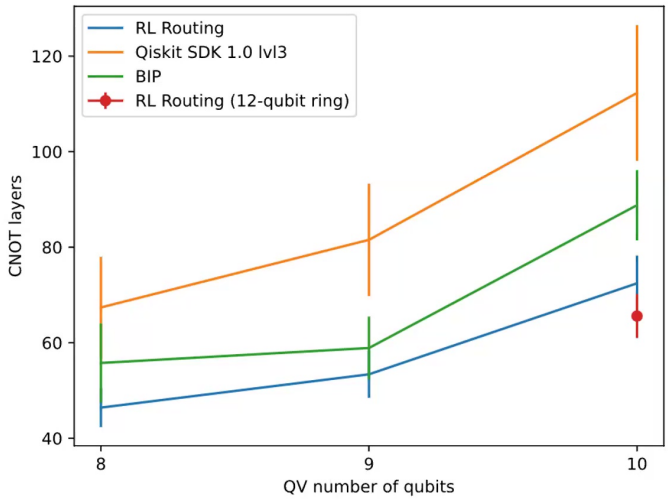




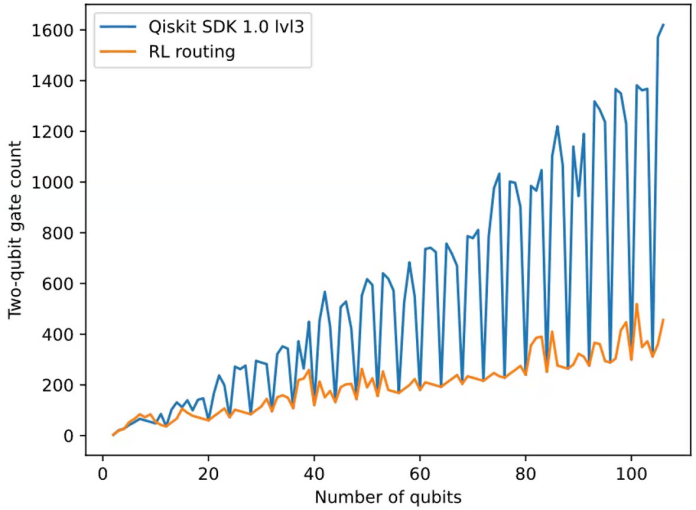
Circuit Routing with RL – Some results

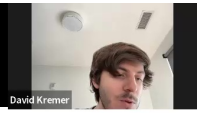
IBM **Quantum**

Routing of QV circuits



Routing of EfficientSU2 circuits (circular entanglement)





Circuit Routing with RL – Benchpress results

IBM **Quantum**

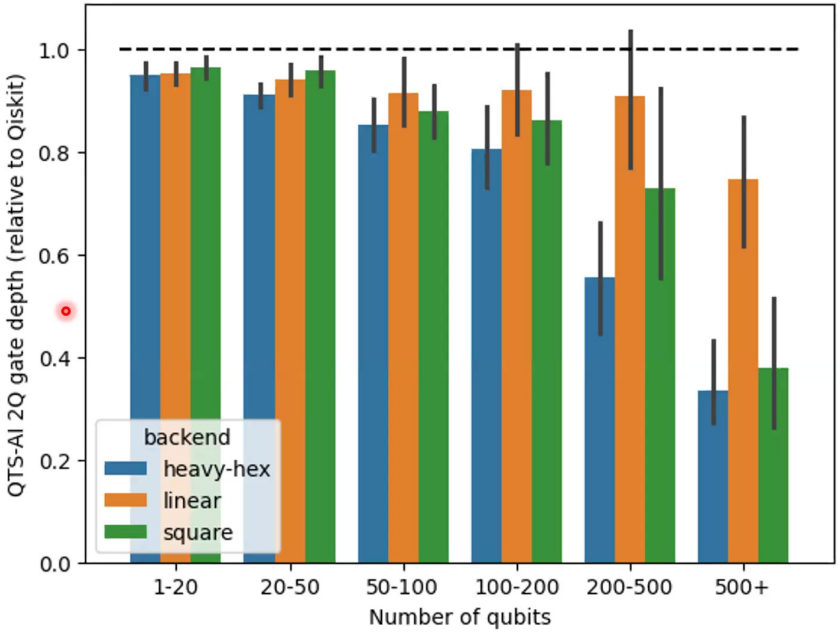
We ran the Qiskit SDK transpiler enhanced with the **AI passes** on the Benchpress circuits.

The results show an **improved transpiling** quality overall, especially **for larger circuits**.

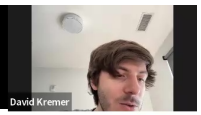
Compared to standard Qiskit SDK transpilation, Qiskit+AI transpiler delivered:

- a further **24% reduction** on average in 2-qubit gates, and
- a further **36% reduction** on average in circuit depth

for utility-scale circuits (i.e. 100+ qubits) when transpiling to the heavy hex topology of IBM Quantum hardware.



Circuit depth for circuits transpiled with Qiskit+AI, relative to standard Qiskit SDK, for the Benchpress circuits (lower is better)



Qiskit Transpiler Service – Tech Preview!

IBM **Quantum**

Open-source version coming soon!!!

Available to IBM Quantum Premium Users

https://www.ibm.com/quantum/blog/ai-transpiler-passes

IBM Quantum **Documentation** | Overview | Start | Build | Transpile | Verify | Run | API reference

Transpile

- Introduction
- Transpiler stages
- Transpile with pass managers
- Configure preset pass managers
- Create a pass manager for dynamical decoupling
- Write a custom transpiler pass
- Transpile against custom backends
- Transpiler plugins
- Qiskit transpiler service
  - Transpile circuits remotely with the Qiskit transpiler service
  - **AI transpiler passes**

AI transpiler passes

The AI-powered transpiler passes are experimental passes that work as a drop-in replacement of "traditional" Qiskit passes for some transpiling tasks. They often produce better results than existing heuristic algorithms (such as lower depth and CNOT count), but are also much faster than optimization algorithms such as Boolean satisfiability solvers. The AI transpiler passes run on the cloud and are available to IBM Quantum™ Premium Plan users.

Note

This is an experimental feature available only to the IBM Quantum Premium Plan. The AI-powered transpiler passes are in beta release status, subject to change. If you have feedback or want to contact the developer team, please use this Qiskit Slack Workspace channel ↗.

The following passes are currently available:

Routing passes

- `AIRouting` : Layout selection and circuit routing

Circuit synthesis passes

- `AICliffordSynthesis` : Clifford circuit synthesis
- `AILinearFunctionSynthesis` : Linear function circuit synthesis
- `AIPermutationSynthesis` : Permutation circuit synthesis

To use the AI transpiler passes through our cloud services, install the `qiskit-transpiler-service` package (see instructions here).

40