

# 量子コンピュータと機械学習



人類の解けない問題を解く

# 企業概要

企業名	MDR株式会社
所在地	東京都文京区本郷2-40-14(本郷三丁目駅徒歩1分)
設立	2008年
資本金	1億3,000万円(資本準備金94,986,050円)
事業内容	量子コンピュータフルスタック開発
従業員	17名程度(パートタイム・アドバイザー含む)
組織体制	研究開発事業部:自社サービス、受託、コンサル 財務管理部:財務、経理、契約、社内庶務

# 自己紹介

湊雄一郎(みなとゆういちろう)  
MDR株式会社 代表取締役



1978年 東京都世田谷区生まれ  
2004年 東京大学工学部建築学科卒業(構造計算力学)  
2005年 株式会社隈研吾建築都市設計事務所勤務  
2008年 MDR株式会社設立～現在に至る

2008年 環境省エコジャパンカップ・エコデザイン部門グランプリ  
2015年 総務省異能vation最終採択  
2017年 内閣府ImPACT山本プロジェクト、プログラムマネージャー補佐

建物の設計やインテリアデザイン、図面が引けます！



# 受賞・採択・支援



NVIDIA Inception Partner



Microsoft for Startups  
Microsoft Innovation Award 2018 finalist



NASA Ames QuAil D-Wave Application

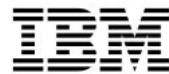


日経ビジネス2018年7月16日号

SNS載せるの勘弁してください><



aws activate



IBM for startup

Tokyo  
Financial Information  
& Technology Summit 2018

Start-up Showcase Finalist

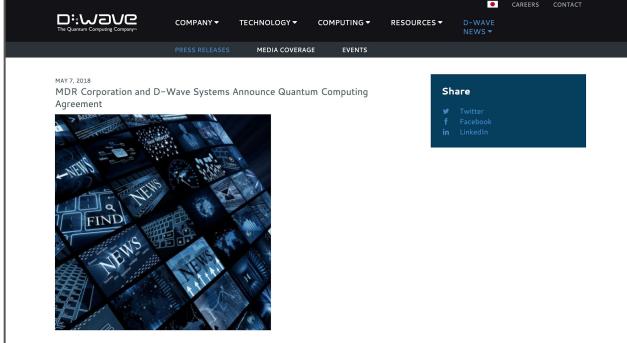


MUFGデジタルアクセラレータ  
準グランプリ

戦略的情報通信研究開発推進事業(SCOPE)  
独創的な人向け特別枠



総務省異能vationプログラム



D-Wave Systems Inc. Release

IPAS  
スタートアップ x 知財戦略

Q-LEAP  
【Flagshipプロジェクト】



資金調達

AQC2016量子コンピュータ国際会議ポスター発表@googleLA

AQC2017量子コンピュータ国際会議ポスター発表@Tokyo

AQC2018量子コンピュータ国際会議量子機械学習ポスター発表@NasaAmes

**アプリケーションからハードウェアまで一貫して開発するチーム**  
東京大学出身者を中心とした開発体制と、金融業界出身者による財務体制。

ソフト・ミドル

ハード

CEO

**湊雄一郎**

東京大学(工学部)建築学科卒  
総務省異能vation  
内閣府ImPACTプロジェクトPM補佐



調達

CFO

**竹林陽一**

ゴールドマン・サックス  
モルガン・スタンレー  
コロンビア大学院博士課程中退(修士号取得)  
東京大学(工学部)化学生命工学科卒



**才田大輔**

東芝 研究開発センター  
東京大学工学系研究科電子工学専攻  
(工学博士)



財務・経理

**石原眞二**

三菱銀行(現三菱UFJ銀行)勤務後、  
(株)エービーシーファイナンス取締役  
中央大学(法学部)卒



営業・管理

**中村人哉**

東京工業大学イノベーション専攻(博士後期課程修了)  
東芝 研究開発センター  
PwCC(現IBM)戦略部門IT戦略部門日本統括  
ソニーグローバルソリューションズ経営企画部門長



# 重点取り組み分野

- ・量子シミュレーション
- ・組合せ最適化
- ・量子機械学習

## 重点取り組み分野

- ・金融(セキュリティ・リスク管理)
- ・自動車(自動運転)
- ・創薬材料(第一原理計算・シミュレーション)

# MDRの競合はGoogleや世界の大手企業

MDRは現在量子コンピュータの世界的な知名度が高まっている。

Duke University		X		X	
D-Wave	X				
Google	X	X			
Griffith Univ./Univ. Of Queensland				X	
Honeywell			X		
IBM		X			
ID Quantique				X	
Institut d'Optique			X		
Intel		X		X	
IonQ			X		
IQM Finland		X			
MDR	X	X			
Microsoft					X
MIT Lincoln Lab	X	X	X		X
MIT/Univ. of Innsbruck			X		
Niels Bohr Institute					X
Nokia Bell Labs					X
Northrop Grumman	X				

独自設計の磁束量子ビット型の超電導量子ビットを開発成功

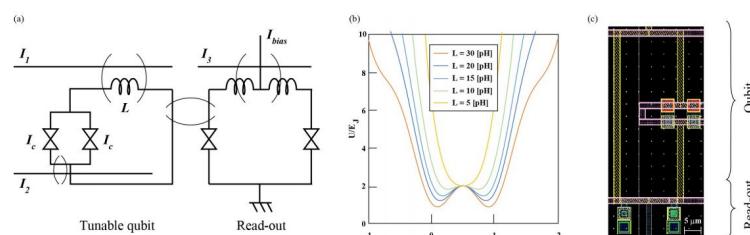


Fig. 1(a) 作成した基本的な電子ビットの構造 (b) 設計した電子ビットにおけるエネルギーボテンシャル (c) 基本的な電子ビット構造のレイアウト図面

	D-Wave q	Google/Cir t	IBM/Qiski t	Microsof t	Rigetti
1QBit	X		X	X	X
Bohr Technology				X	
Cambridge Quantum Computing		X	X	X	
Entropica Labs				X	X
GTN				X	
Heisenberg Quantum Simulation		X			X
Horizon Quantum Computing					X
MDR	X		X		
OTI Lumionics	X			X	X
ProteinQure	X		X	X	X

# 直近のデバイスの活動

## Principle Verification of the Superconducting Flux Qubit Cell Toward the Quantum Sampling Approach for Training of Deep Neural Networks

\*Daisuke Saida<sup>1</sup>, Hayato Ariyoshi<sup>2</sup>, Yuki Yamanashi<sup>2</sup>

MDR Inc.<sup>1</sup> Yokohama National University<sup>2</sup>

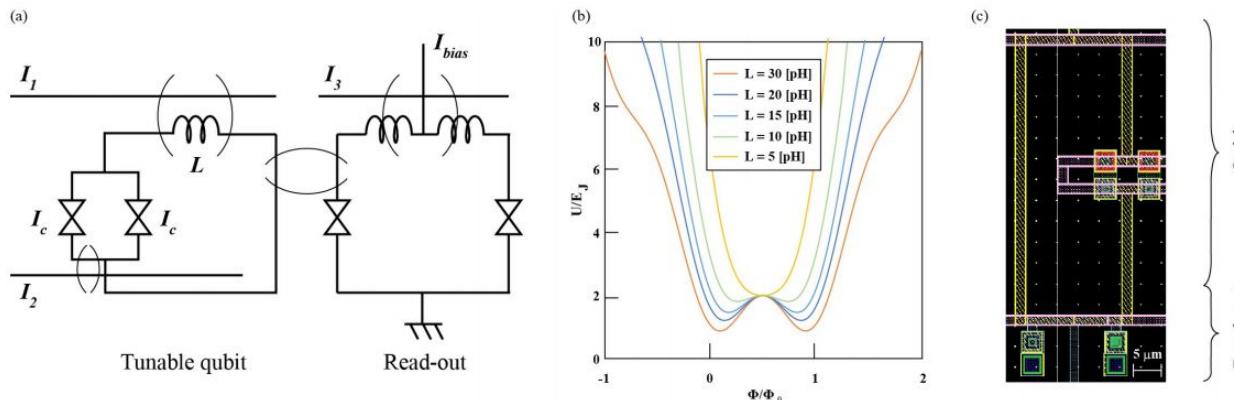


Fig. 1(a) 作製した基本的な量子ビットの構造 (b) 設計した量子ビットにおけるエネルギーポテンシャル (c) 基本的な量子ビット構造のレイアウト図面

# 世界のベンチャー

世界のトップ量子コンピュータベンチャー企業一覧 (2019年2月現在)							
#	ベンチャー/大手IT	所在	Google	IBM	Microsoft	Rigetti	D-Wave
1	1Qbit	カナダ		◎	◎	◎	◎
2	Bohr Technology	ポーランド			◎		
3	Cambridge Quantum Computing	イギリス	◎	◎	◎		
4	Entropica Labs	シンガポール			◎	◎	
5	GTN	イギリス			◎		
6	Heisenberg Quantum Simulation	ドイツ	◎			◎	
7	Horizon Quantum Computing	シンガポール				◎	
8	MDR	日本		◎			◎
9	OTI Lumionics	カナダ			◎	◎	◎
10	ProteinQure	カナダ		◎	◎	◎	◎
11	QC Ware	米国	◎	◎	◎	◎	◎
12	Q-Ctrl	オーストラリア		◎			
13	Qu & Co	オランダ		◎			
14	Quantum Benchmark	カナダ	◎	◎			
15	Qulab	米国			◎	◎	
16	QxBranch	米国		◎	◎	◎	◎
17	Riverlane Research	イギリス			◎	◎	
18	Solid State AI	カナダ		◎	◎		
19	Strangeworks	米国		◎	◎	◎	
20	Zapata Computing	米国	◎	◎	◎	◎	
カナダ			5				
米国			5				
イギリス			3				
シンガポール			2				
日本			1				



# 2000名を超える量子コンピュータアプリコミュニティ



一度のイベントで300-500名の集客

<https://qnn.connpass.com/>

オフライン2000名  
オンライン1200名

The screenshot shows a Connpass event page for "Quantum". The page features a large yellow "Quantum" title. Below it is a logo consisting of a blue square with a white stylized letter "Q". The event details are as follows:

- Event Name: 量子コンピュータ
- Model: ゲートモデル & アニーリングモデル
- Organizer: MDR株式会社

At the bottom right of the page, there is a red rectangular box highlighting the member count: "メンバー (2001人)".

## ⑦量子アルゴリズムの権威 米ハーバード大学教授が創業したベンチャー

Zapata Computing概要



### 企業概要

We develop quantum computing software and algorithms to solve industry-critical problems. Our offices are based out of The Engine, MIT's startup incubator in Cambridge, MA. Zapata Computing spun-out of Harvard University in 2017.

創業

2017年

本拠地

USA・Boston

チーム

- CEO: Christopher J Savoie 氏  
・連続起業家。九州大学薬学博士&法学修士
- CSO: Alan Aspuru-Guzik 氏  
・ハーバード大学 化学科教授
- 量子アルゴリズム開発では学術界でトップの評価



ファイナンス

- 累計調達額: \$5.4M
- ステージ: Seed
- 投資家: Engine, Pillar Computing

量子コンピューティングでは、コンピューター自体の製造以外にも難しい課題がある。高度な量子アルゴリズム、すなわち量子コンピューターの性能を最大限に引き出すために特別に作られたソフトウェアが必要になることだ。

アラン・アスブル＝グジックは量子アルゴリズムの開発ですでに学会で高い評価を得ているが、このほどその市場を拡大しようとしている。ハーバード大学で教授を務め(7月にトロント大学に移籍予定)、MITテクノロジーレビューの2010年版「35歳未満のイノベーター」の一覧に名を連ねたアスブル＝グジック教授は5月17日、発表によると540万ドルを調達して、ザパタ・コンピューティング(Zapata Computing)を共同創業した。ザパタ・コンピューティングの最終目標は、いわば「量子アルゴリズムのスーパー・マーケット」となることだ。既成ソフトウェアの多種多様な品ぞろえを多くの企業に提供し、量子コンピューターの巨大な処理能力を多くの企業が活用できるようにすることにある。



(参考)直近の記事(MIT Technology Review)

# 現在日本で話題になる2方式

汎用計算のできる量子ゲートと、組合せ最適化問題に特化した量子アニーリング・イジングがある。

## 量子コンピュータ

「量子力学」の原理を応用して計算

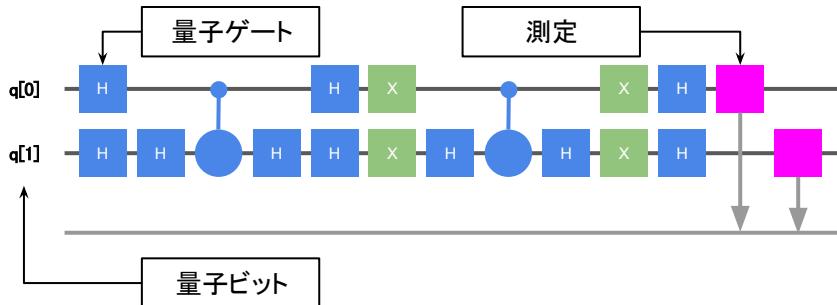
### 量子ゲート

時間ごとに量子ゲートを変えて計算する汎用マシン

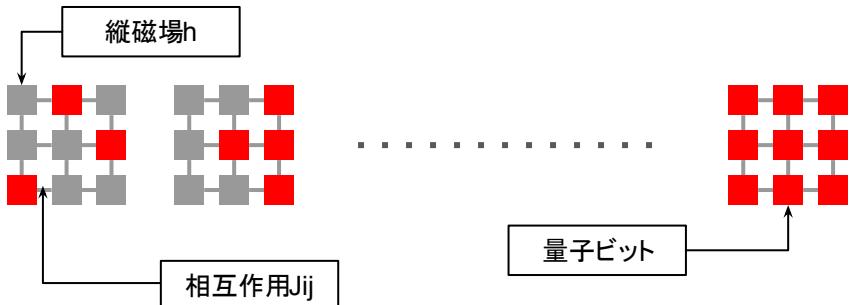
### 量子アニーリング・イジング

最初に値を設定して解く組合せ最適化専用マシン

#### 米国や中国中心の取り組み



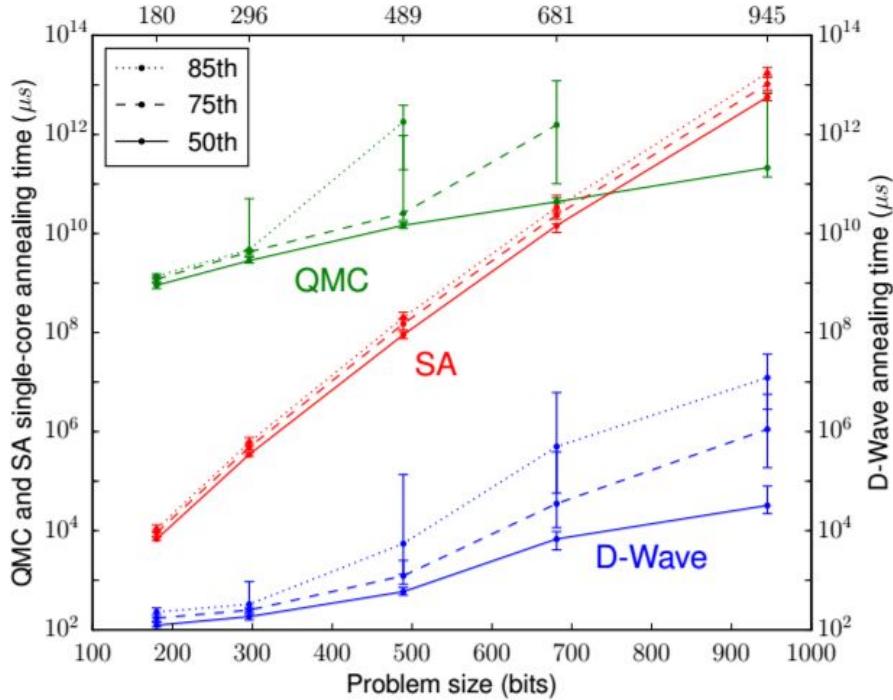
計算時間



計算時間

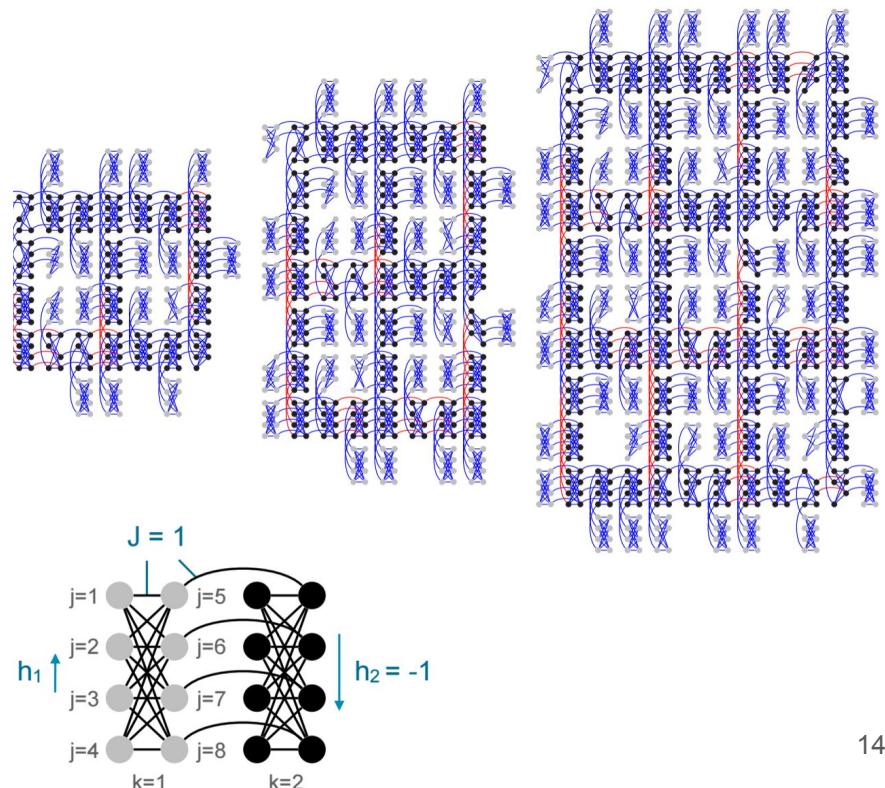
# 量子コンピュータの流行ったきっかけ

2015年にD-Waveの量子コンピュータは「1億倍高速」とNASAとGoogleが会見で発表



arXiv:1512.02206v4 [quant-ph] 22 Jan 2016

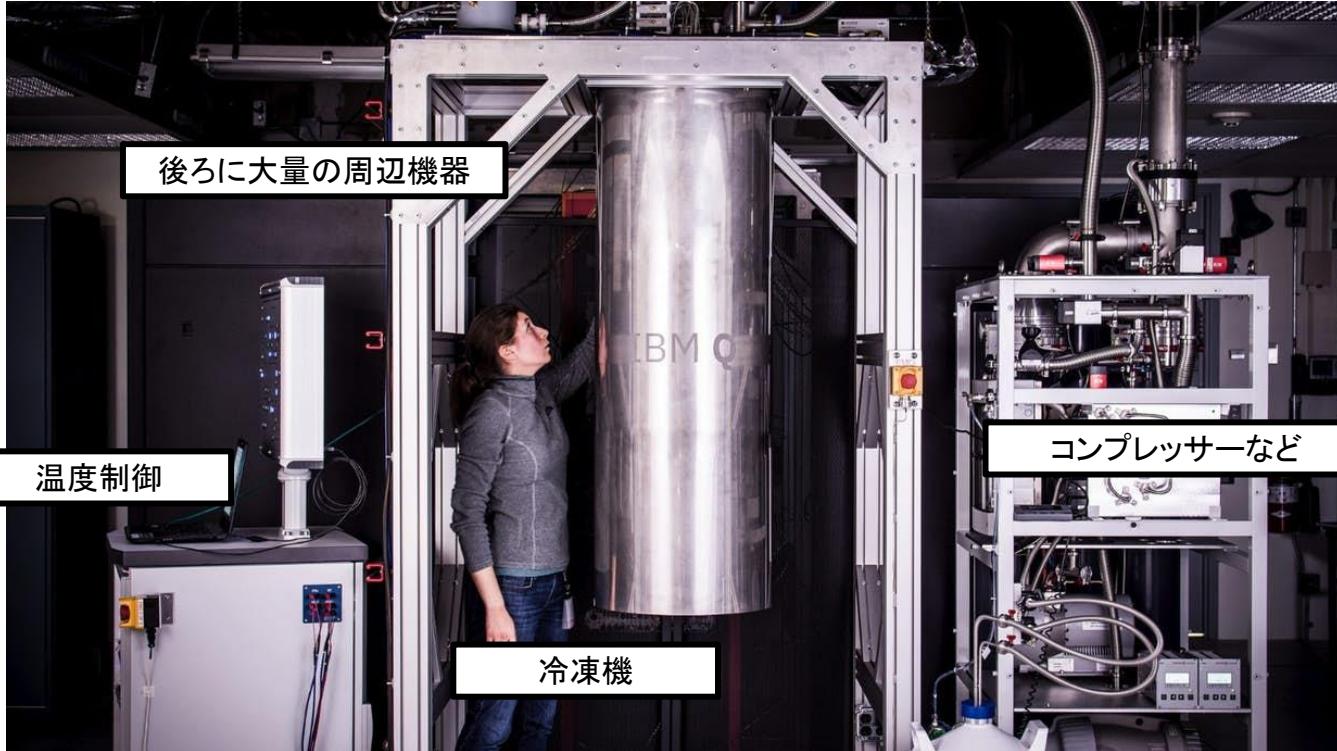
What is the Computational Value of Finite Range Tunneling?



# System One

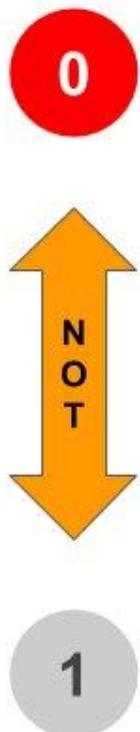


# 超電導量子コンピュータハードウェア

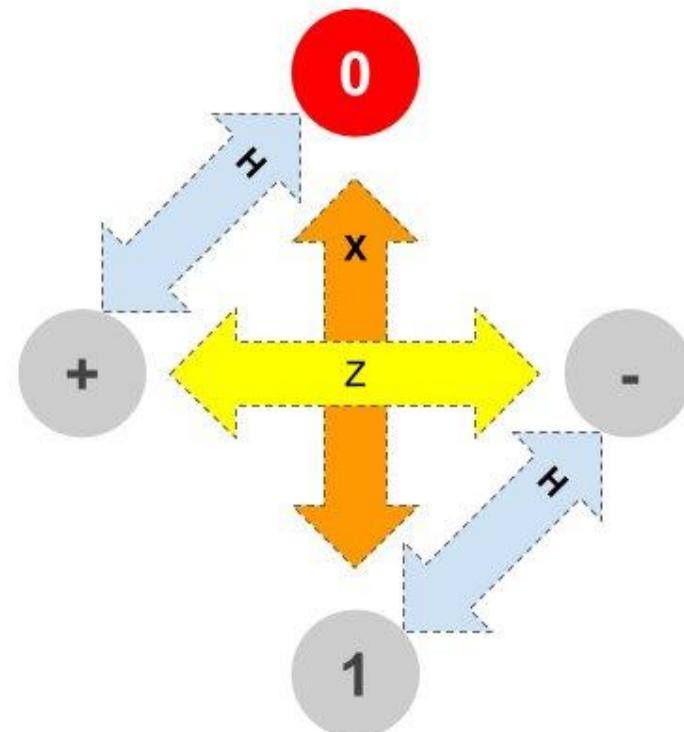


<https://newatlas.com/ibm-next-quantum-processors/49590/>

## 今のコンピュータ

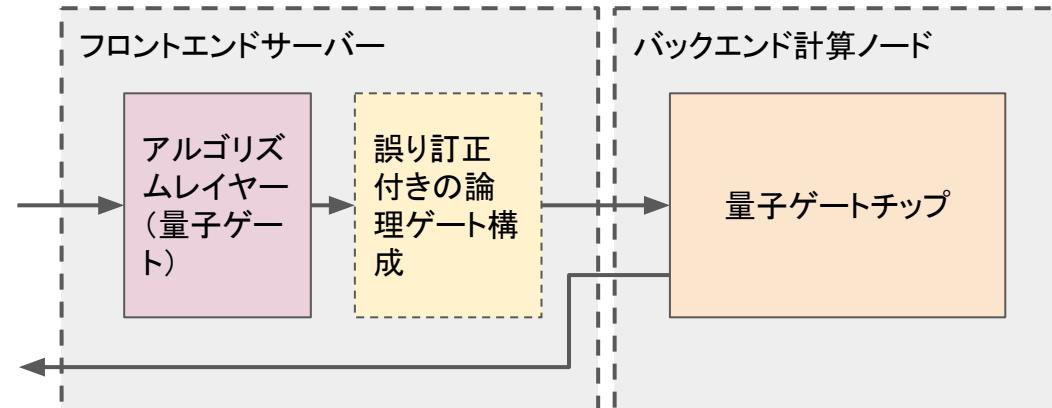
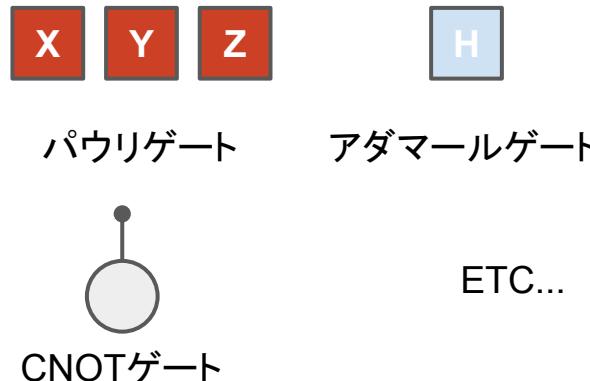


## 量子コンピュータ



# ゲートの基本

量子ビットの初期化、ゲート演算、測定の基本ステップ。





Pythonで簡単量子コンピュータプログラミング

```
-----  
pip install blueqat
```

```
-----  
from blueqat import Circuit  
Circuit().h[0].cx[0,1].m[:].run(shots=100)
```

```
Counter({'00': 48, '11': 52})
```

# Blueqat

Blueqat

```
from blueqat.opt import Opt
c = Opt().add([[1,1],[1,1]]).add("(q0+q1)^2", N=2)

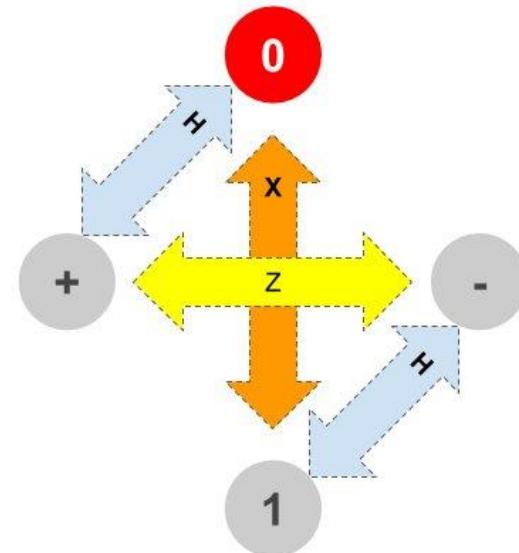
#qaoa
print(c.qaoa().most_common(5))
#=>(((0, 0), 0.7639901896866), ((1, 0), 0.10321404014639714), ((0, 1),
0.10321404014639707), ((1, 1), 0.029581730020605202))

#annealing
print(c.run())
[0, 0]
```

# 例題1：量子重ね合わせ

0と1の重ね合わせによって測定するたびに答えが50%ずつに。

```
from blueqat import Circuit  
Circuit().h[0].m[:].run(shots=100)  
  
Counter({'0': 49, '1': 51})
```

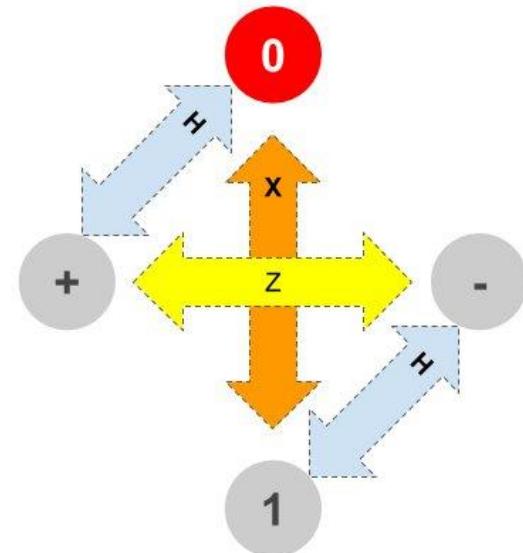
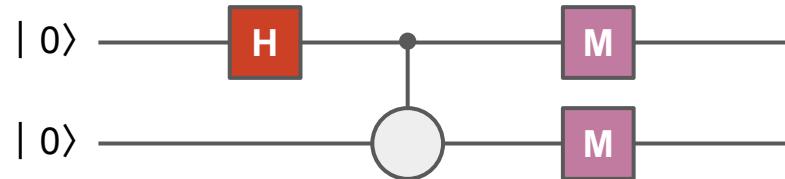


## 例題2：量子もつれ

重ね合わせただけでは答えがたくさん出てきてしまいます。重ね合わせからデータを絞り込むために量子もつれを使います。

```
from blueqat import Circuit
Circuit().h[0].cx[0,1].m[:].run(shots=100)

Counter({'00': 55, '11': 45})
```



# (参考)行列固有値

```
from blueqat import vqe
from blueqat.pauli import qubo_bit as q

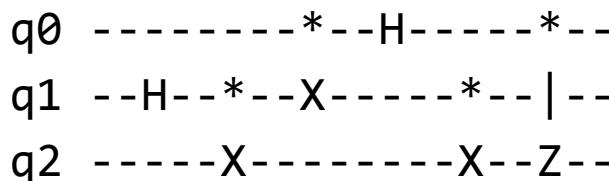
hamiltonian =
-3*q(0)-3*q(1)-3*q(2)-3*q(3)-3*q(4)+2*q(0)*q(1)+2*q(0)*q(2)+2*q(0)
*q(3)+2*q(0)*q(4)+2*q(1)*q(2)+2*q(1)*q(3)+2*q(1)*q(4)+2*q(2)*q(3)+
2*q(2)*q(4)+2*q(3)*q(4)
step = 2

result = vqe.Vqe(vqe.QaoaAnsatz(hamiltonian, step)).run()
print(result.most_common(12))
```

## (参考)量子テレポートーション

q0の状態をq2にテレポート。下の回路は0がうつる。

```
from blueqat import Circuit
Circuit().h[1].cx[1,2].cx[0,1].h[0].cx[1,2].cz[0,2].m[:].run(shots
=100)
Counter({'110': 27, '000': 16, '100': 23, '010': 34})
```

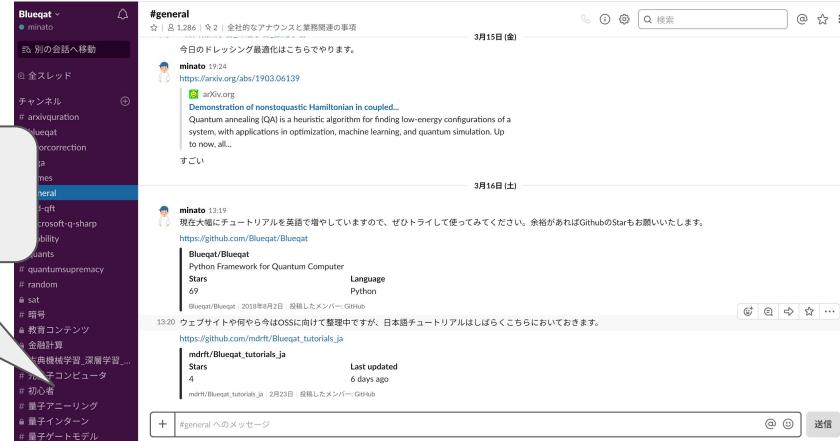


# 質問したくなったら、、、

下記のリンクからslackコミュニティへご参加ください。(多分)誰かしら教えてくれます。

[https://join.slack.com/t/blueqat/shared\\_invite/enQtNDA3NjU0MjM2NjEyLWU1MjJI NTM5NGNkNjk1NmYzYjU5NDIiOTc5M2QwNDc1YmMyNzYzYjY3YmE2NGUxMTI1OTEyYTUwOWEyNmY4MWY](https://join.slack.com/t/blueqat/shared_invite/enQtNDA3NjU0MjM2NjEyLWU1MjJI NTM5NGNkNjk1NmYzYjU5NDIiOTc5M2QwNDc1YmMyNzYzYjY3YmE2NGUxMTI1OTEyYTUwOWEyNmY4MWY)

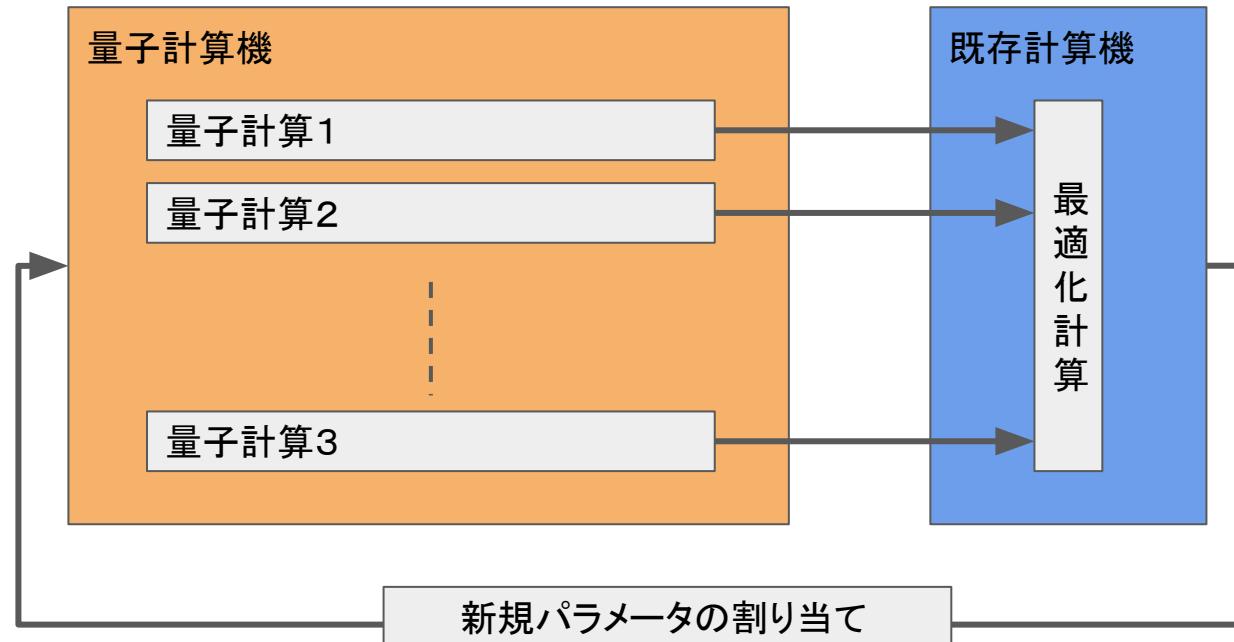
1250名 !



# 量子ゲートとVariational method

主流の量子コンピュータと既存計算機のハイブリッド計算アルゴリズム

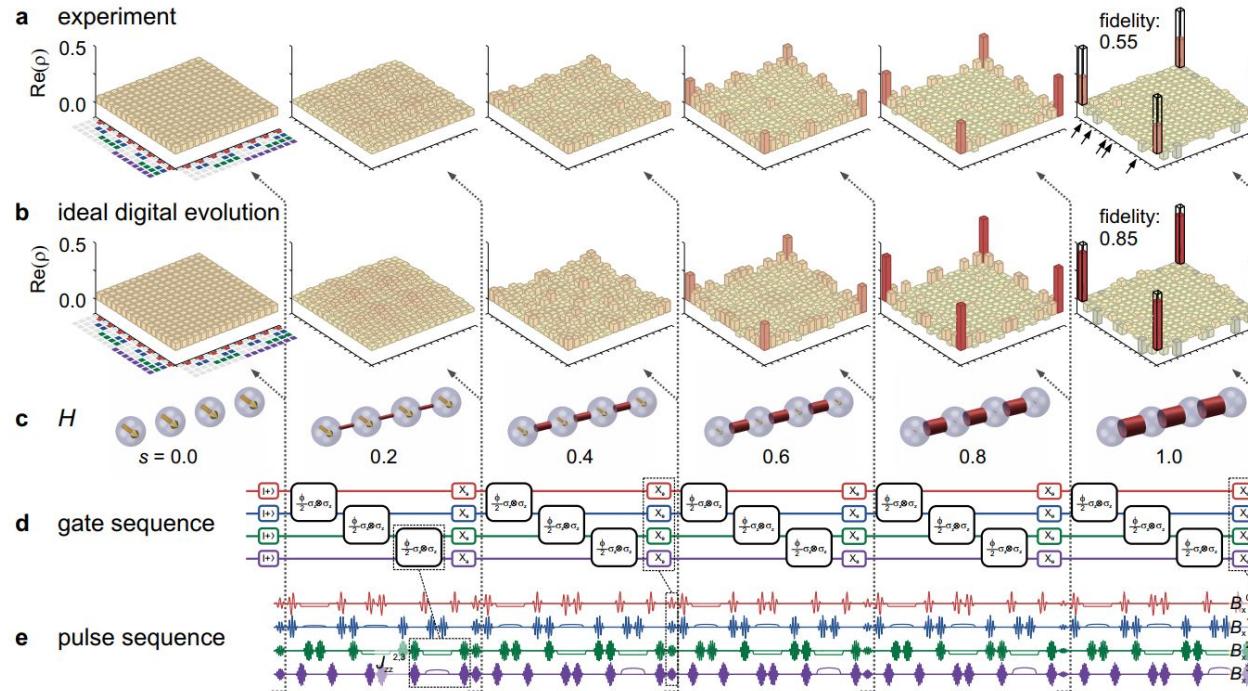
$$E(\psi(\theta)) = \langle \psi(\theta) | H | \psi(\theta) \rangle \geq E_0$$



## シュレーディンガー方程式

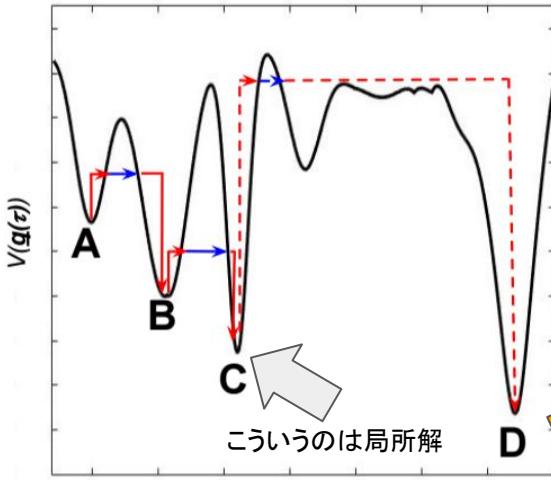
## 量子シミュレーション

$$|\psi(t)\rangle = \exp(-i/\hbar H(t - t_0)) |\psi(t_0)\rangle$$

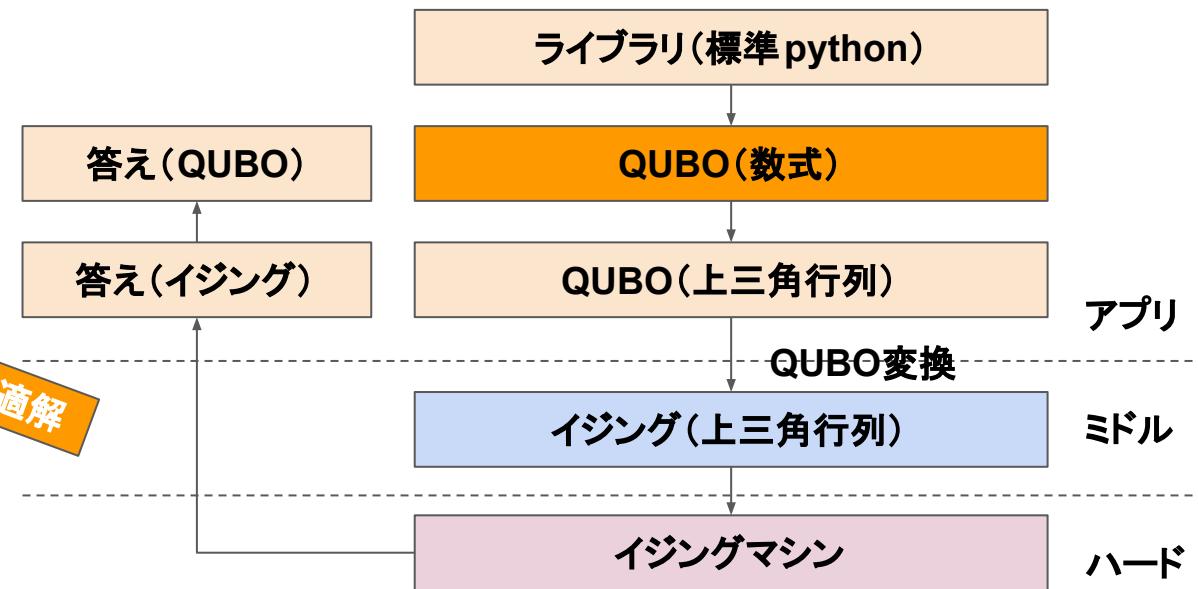


# 量子アニーリング・イジングの演算方式

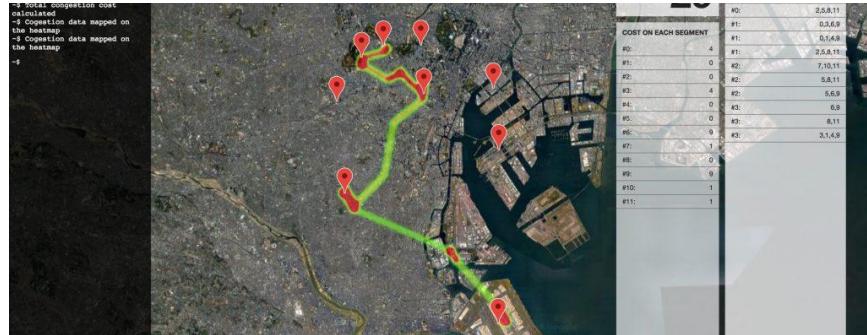
QUBOもしくはイジングで数式を作り、それから最適解を計算機で探索する。



arXiv:1512.02206v4 [quant-ph] 22 Jan 2016  
What is the Computational Value of Finite Range Tunneling?



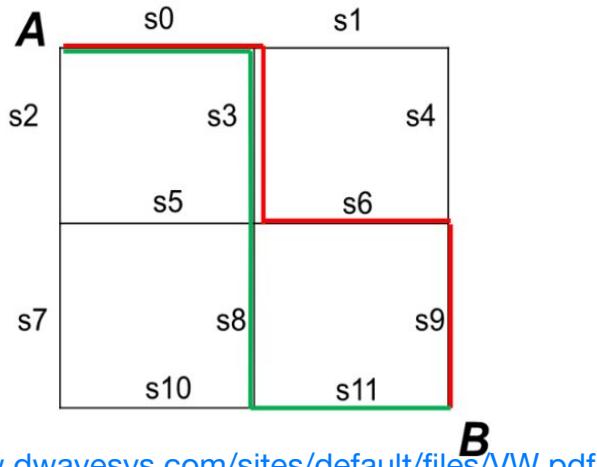
# 例題)自動車2台の経路最適化を行う



## Additional constraints

$$K * (Q_{11} + Q_{12} + Q_{13} - 1)^2$$

$$K * (Q_{21} + Q_{22} + Q_{23} - 1)^2$$



Car	Route	Binary variable
Car 1	#1: s0,s3,s6,s9	$Q_{11}$
Car 1	#2: s0,s3,s8,s11	$Q_{12}$
Car 1	#3: s2,s7,s10,s11	$Q_{13}$
Car 2	#1: s0,s3,s6,s9	$Q_{21}$
Car 2	#2: s0,s3,s8,s11	$Q_{22}$
Car 2	#3: s2,s7,s10,s11	$Q_{23}$

引用: <https://www.dwavesys.com/sites/default/files/VW.pdf>

# コスト関数と制約条件を解く

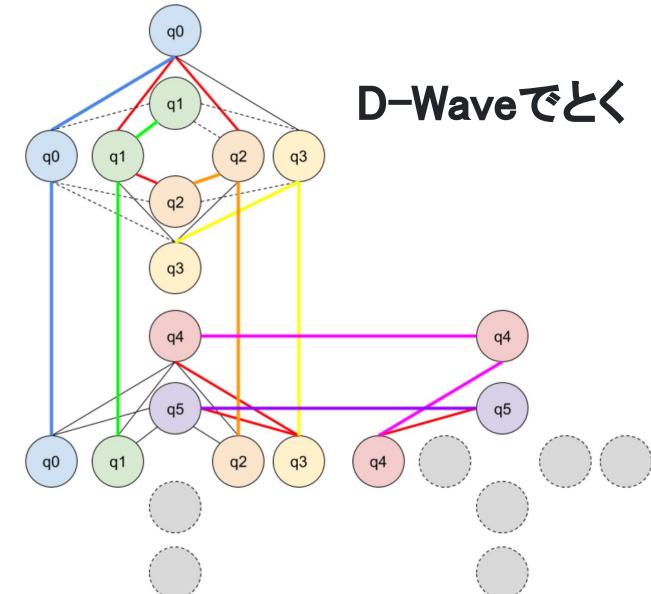
$$H = 4q_0 + 4q_0q_1 + 8q_0q_3 + 4q_0q_4 + 4q_1 + 2q_1q_2 + 4q_1q_3 + 8q_1q_4 + 2q_1q_5 + 4q_2 + 2q_2q_4 + 8q_2q_5 + 4q_3 + 4q_3q_4 + 4q_4 + 2q_4q_5 + 4q_5$$

これを係数を見やすくQUBOMatrixの形に直すと、

	$q_0$	$q_1$	$q_2$	$q_3$	$q_4$	$q_5$
$q_0$	4	4	0	8	4	0
$q_1$		4	2	4	8	2
$q_2$			4	0	2	8
$q_3$				4	4	0
$q_4$					4	2
$q_5$						4

$$K(q_0 + q_1 + q_2 - 1)^2 + K(q_3 + q_4 + q_5 - 1)^2$$

	$q_0$	$q_1$	$q_2$	$q_3$	$q_4$	$q_5$
$q_0$	$4 - K$	$4 + 2K$	$2K$	8	4	0
$q_1$		$4 - K$	$2 + 2K$	4	8	2
$q_2$			$4 - K$	0	2	8
$q_3$				$4 - K$	$4 + 2K$	$2K$
$q_4$					$4 - K$	$2 + 2K$
$q_5$						$4 - K$



上記の太線は量子ビットのコピーです。

この回路に上記を代入します。

D-Waveでとく

## Quantum Approximate Optimization Algorithm (量子ゲート)

QAOAアルゴリズムは組合せ最適化問題を解くためのアルゴリズムです。ハミルトニアンというコスト関数を作り、それをアルゴリズムに入れ込んで計算を行います。

計算原理は量子断熱計算に近く、初期状態をすべて $|+\rangle$ のsuperpositionからスタートし、量子断熱計算のステップ数を活用して離散的にシミュレーションを行います。最終的に求めるハミルトニアンを最終状態として、 $|+\rangle$ 状態から断続的に入れ替えを行います。

ステップ数と入れ替える2つのハミルトニアンについて、ステップするものが特徴で、このパラメータに関しては古典のアルゴリズムで

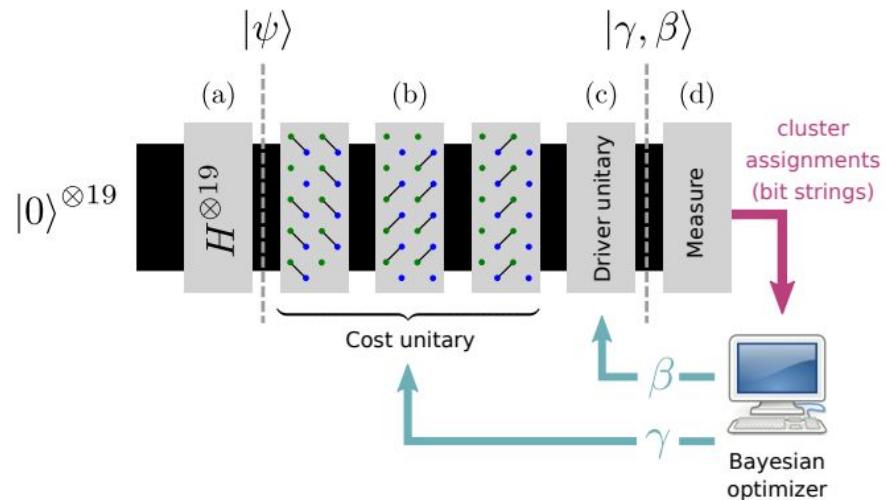
Rigettiが実機で実装をして話題となりました。

Unsupervised Machine Learning on a Hybrid Quantum Computer  
<https://arxiv.org/pdf/1712.05771.pdf>

参考：量子ゲートで組合せ最適化問題を解くQAOAの実装

<http://blog.mdrft.com/post/229>

また、RigettiのGroveという量子ゲートマシン向けのライブラリが



## Qboost (量子アニーリング)

2017年NASAが衛星写真の二値分類に利用したアルゴリズムで元々はGoogleとD-Waveが2009年に開発したアルゴリズムをリバイバルしています。

Deploying a quantum annealing processor to detect tree cover in aerial imagery of California

<https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0172505>

NIPS 2009 Demonstration: Binary Classification using  
Hardware Implementation of Quantum Annealing  
<https://static.googleusercontent.com/media/www.google.com/ja/googleresearch.pdf>

こちらは基本的にはboostingという弱学習機を組み合わせて1つの学習エンジンの組合せ最適化問題の機能をうまく使って衛星写真からカリフォルニア州の木の伐採を行なっています。

単純に多数決で済ませてしまうと面白くないので、相互作用項を利用したりが量子アニーリングを活用したアルゴリズムぼくてとても面白い。この制約で理想的な形式が取れませんでしたが、2017年の最新のD-Waveが採用され、実際に分類問題で特徴量の数を調整しながらの分類ができる



## 量子ボルツマンマシン（量子アニーリング）

量子ボルツマンマシンはD-Waveなどの量子アニーラを使ってサンプリングを行い学習を行うモデルでもっとも有名なのがRBM（制限付きボルツマンマシン）です。D-Waveマシンのようにイジングモデルを扱うマシンは01の二値で扱う必要がある、ネットワーク構成がundirected graphであるなどの理由で、RBMはD-Waveのような量子アニーリングと相性がいいです。

初期の応用は、シミュレーションでD-WaveマシンにRBMを搭載できるかどうかというYoshua Bengio先生が行った2013年の論文から始まります。

On the Challenges of Physical Implementations of RBMs

Vincent Dumoulin, Ian J. Goodfellow, Aaron Courville, Yoshua Bengio(Submitted on 18 Dec 2013 (v1), last revised 24 Oct 2014 (this version, v2))

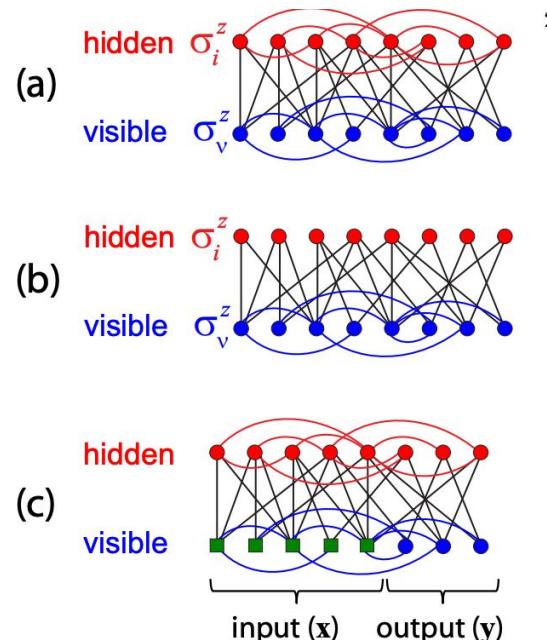
<https://arxiv.org/abs/1312.5258>

この論文の中では、実際にはシミュレーションで量子アニーリングマシンを使ってイジングにRBMを実装できるかどうかの評価が行われました。実際には接続数をきちんと確保すればある程度のノイズがあっても学習はできるという結論において実装は可能であるという結論に達しました。

その後、2015年にロッキードマーチン社のSteven Adachi氏による評価によって、ハミルトニアン的にどのように具体的に量子アニーリングマシンがRBMに使えるのかという評価が進められました。

Application of Quantum Annealing to Training of Deep Neural Networks

<https://arxiv.org/abs/1510.06356>



## Quantum Neuron (量子ゲート)

量子アニーラは基本的に無向グラフで、 $J_{ij}=J_{ji}$ のため、バックプロパゲーション、フォワードプロパゲーションの区別をつけるのが難しいです。デジタル式の量子ゲートでは、初期状態 $|0\rangle$ からスタートし、ゲートを使って入力データを量子状態として作成し、フォワードプロパゲーションとしてデジタルで推論ができます。その回路を提案したのが、Quantum Neuronです。

基本的に量子ゲート回路は線形の回路ですので、一番問題になるのはト回路で表現するかということです。

Quantum Neuron: an elementary building block for machine learning  
<https://arxiv.org/abs/1711.11240>

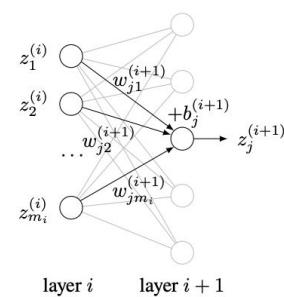
こちらの記事がとても詳しいです。

量子コンピュータでニューラルネットワークな論文紹介～量子二：

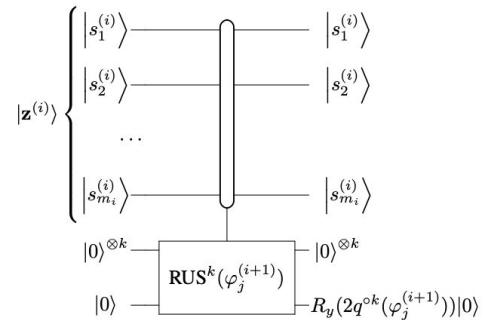
<https://qiita.com/piyo7/items/2104fe7084c95ed4b97b>

まず、ニューロンの結合は回転角を使って、コントロールユニタリと後、回転角をうまく使って非線形な関数を導き出し活性化関数を実現

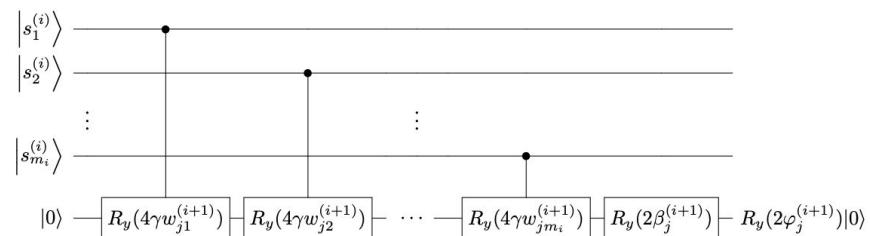
バックプロパゲーションについては課題だと思いますが、最適化アルゴリズム



(a)



(b)



(c)

# <https://quadrant.ai/whatisit>

QUADRANT

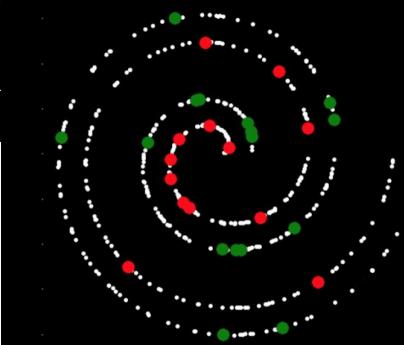
Home What Is Generative? Blog CONTACT US

## What Is Generative Machine Learning?

DEEP LEARNING HAS DRIVEN THE AI/MACHINE LEARNING REVOLUTION BECAUSE IT WORKS WELL. BUT IT ONLY WORKS WHEN POWERED BY ENOUGH DATA.

*...the rocket engine*

As Andrew Ng, Founder of Coursera, Google Brain and Head of AI at Baidu, has said:



Does this help? Generative learning models ingest unlabeled data (white) and output structure, making predicting class labels much easier without having to

Not only can you use generative machine learning to build accurate discriminative models, but generative models also discover useful insights into the structure and categories hidden within your data.

*It is precisely these insights that allow for training with less data.*

In many application areas the goal is not simply to predict specific outcomes, but to understand the underlying structure. Models which uncover structure may be much more valuable than predictions alone.

# Finding Hadamard Matrices by a Quantum Annealing Machine

Andriyan B. Suksmono,

School of Electrical Eng. and Informatics, ITB, Bandung, Indonesia

Yuichiro Minato

MDR Inc., Tokyo, Japan

# 1. Introduction

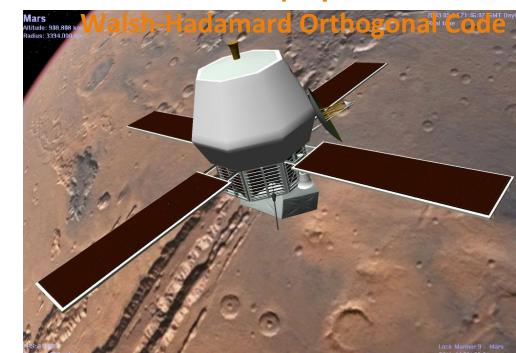
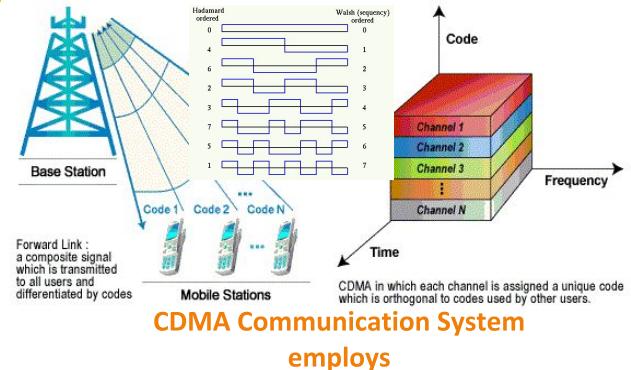
- **Hadamard matrix (H-matrix)**

- **Definition:** an orthogonal binary {-1,1} matrix
- **Applications:** orthogonal codes used in CDMA, ECC (Error Correction Code) with maximal error correction capability, employed in Mariner-9
- **Scientific/Math:** H-matrix conjecture is a ~100 years old unsolved problem

- **Why finding a H-matrix is hard?**

- For an M-order matrix, there are  $[2^{(M^2)}] \sim \exp(M^2)$  binary matrices
- H-matrix **conjecture** predicts, there is a H-matrix for every  $M=4k$ , k positive integer. How to find it?
  - Brute force, worst-case condition: one should check all binary matrices, an  $O[\exp(M^2)]$  problem --> a hard problem.

- **Proposed Solution: USE A QUANTUM COMPUTER !**



Mariner-9 employed Hadamard's ECC to protect Mars's images sent to Earth

## PROBLEM

Find  $H \approx \min E\{s_i\}$

## 2. Methods: (a) Use symbolic computing to formulate many-terms 2-body Hamiltonian

s-domain k-body  
Energy function  
 $E_k(s_i)$

Alg.1

q-domain k-body  
Energy function  
 $E_k(q_i)$

Alg.2

q-domain 2-body  
Energy function  
 $E_2(q_i)$

Alg.3

s-domain 2-body  
Energy function  
 $E_2(s_i)$

Alg.4

Hamiltonian  
Formulation  
 $H_2(\{\sigma_i^z\})$

---

### Algorithm 1 Construction of $E_k(s_i)$ by symbolic computation

---

- 1: Construct array of variables  $\{s_i, s_j\}$  according to the order  $M$  of the H-matrix
  - 2: Calculate inner product  $d_{ij} = \langle s_i, s_j \rangle$  of symbols between every pairs of columns of the array
  - 3: Calculate  $E_k(s_i) = \sum d_{i,j}^2$
  - 4: Cleanup  $s_i^2$  terms by substitution:  $E_k(s_i) = E_k(s_i)|_{s_i^2 \leftarrow 1}$
- 

### Algorithm 2 Transform $E_k(s_i) \rightarrow E_k(q_i)$

---

- 1: For all binary variables  $s_i$ :
  - 2:  $E_k(s_i)|_{s_i \leftarrow q_i}$
- 

---

### Algorithm 3 Transform $E_k(q_i) \rightarrow E_2(q_i)$

---

- 1: Construct a list of substitution pair  $sPair[col_i, col_j, col_k]$
  - 2:  $E_2(q_i) \leftarrow E_k(q_i)$
  - 3: For all high order terms in  $E_2(q_i)$  and based on  $sPair$ :
  - 4:  $E_2(q_i) \leftarrow E_2(q_i)|_{q_i \cdot q_j \leftarrow q_k} + H \wedge (q_i, q_j, q_k, \delta)$
  - 5: Simplify  $E_2(q_i)$
- 

---

### Algorithm 4 Transform $E_2(q_i) \rightarrow E_2(s_i)$

---

- 1: For all variables  $\{q_i\}$ :
  - 2:  $E_2(s_i) \leftarrow E_2(q_i)|_{q_i \leftarrow (\frac{1-s_i}{2})}$
-

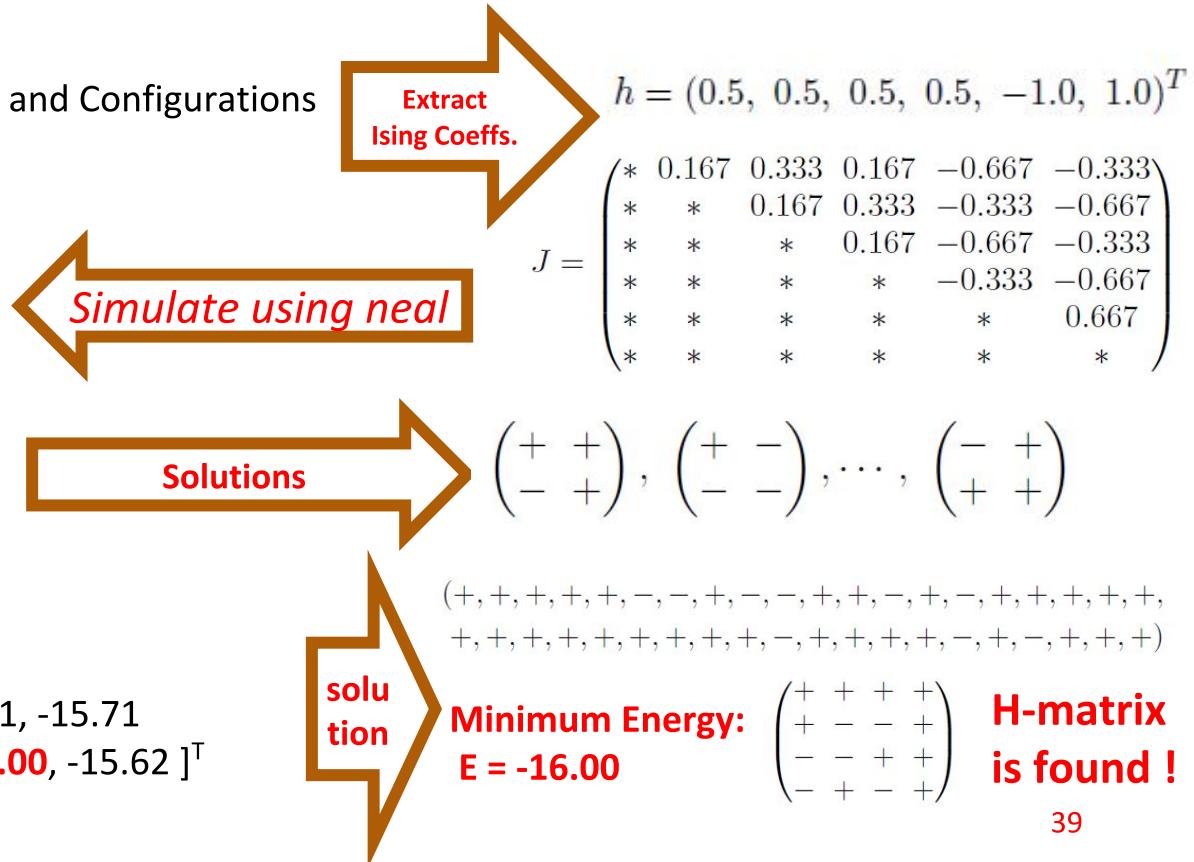
### 3. Simulations: (a) Finding H-matrices

- Finding 2-order H-matrix:
  - NSWEEP=1000, Results: Energy and Configurations

No	Configuration	Energy	Ground-State
1	(+,-,+,-,+,-)	-2.33	Y
2	(+,-,-,-,+,-)	-2.33	Y
3	(+,-,+,-,+,-)	-2.33	Y
4	(-,-,+,-,+,-)	-2.33	Y
5	(+,-,+,-,+,-)	-2.33	Y
6	(+,-,-,-,+,-)	-2.33	Y
7	(+,-,+,-,-,+)	-2.00	N
8	(+,-,-,+,-,+,-)	-2.33	Y
9	(+,-,+,-,-,+,-)	-2.00	N
10	(-,-,+,-,+,-)	-2.33	Y

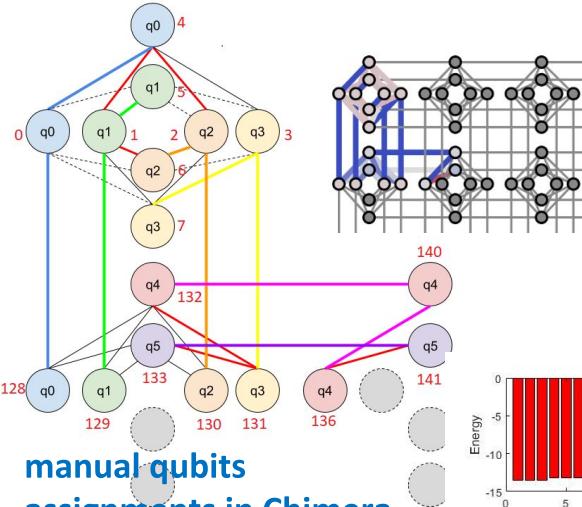
- Finding 4-order H-matrix:

$$E = [-16.00, -15.62, -15.62, -15.71, -15.71, -15.71, -15.71, -15.71, -16.00, -15.62]^T$$



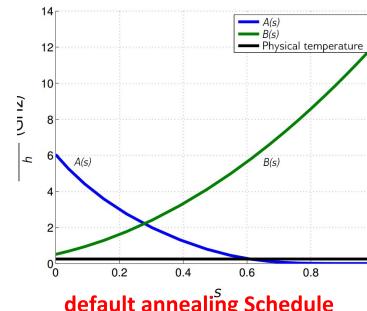
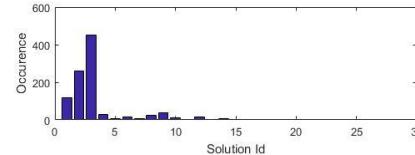
# 4. DW2000Q:(a) Finding H-matrices

- Finding 2-order H-matrix



Minimum energy  
E = -13.52

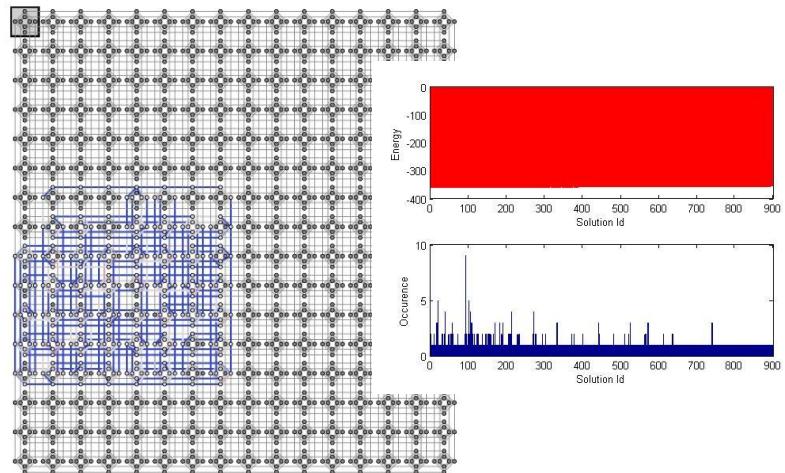
$$\begin{pmatrix} + & - \\ - & - \end{pmatrix}$$



NREADS=100

0

- Finding 4-order H-matrix



Minimum energy  
E= -322.91

$$\begin{pmatrix} - & - & + & + \\ + & - & + & - \\ - & - & - & - \\ - & + & + & - \end{pmatrix}$$

# Conclusions and Further Directions

1. Finding H-matrices (H-SEARCH) is a **hard problem** which potentially can be solved by a **quantum computer**.
2. Construction of the Hamiltonians of H-SEARCH and its related problems needs manipulation of equation with **large number of terms**. We have managed this problem by **symbolic computing**.
3. We have **implement** H-SEARCH in a quantum annealer **DW2000Q**. Although only up to **4-order H-matrix**, future generation of quantum annealers capable to implement higher orders due to increasing number of qubits and connectivity beyond Chimera.
4. In the forthcoming research, we will address the issue of scaling/**speed-up** and implementation of H-SEARCH **in quantum gate model**.

# Solving tiling puzzles with quantum annealing

## 1 Introduction

A tetromino is a geometric shape composed of four squares. Tetrominoes come in five distinct shapes, as shown in figure 1. Each tetromino has a name; from left to right, they are “I,” “O,” “L,” “T,” and “S.” The tetromino puzzle requires one to cover a rectangular board with tetrominoes. Using a specific number of tetrominoes, the board must be covered and have no overlapping pieces or empty spaces, as demonstrated in figure 2. The tetrominoes can be rotated or flipped.

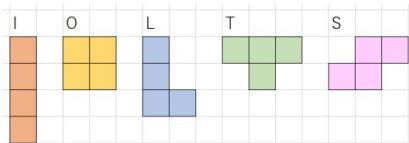


Figure 1: Five tetromino shapes

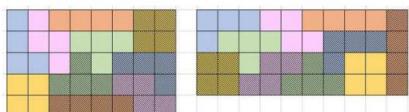


Figure 2: Example of a board tiled with tetrominoes

tion, we actually solve the formulas using both a simulator and an actual quantum annealing machine, DW2000Q.

## 2 Methods

### 2.1 Details of the tetromino tiling puzzle

In this article, we use a board that is  $5 \times 8$ , creating 40 squares (figure 3), and tile this board with two of each of the tetrominoes shown in figure 1. The tetrominoes can be rotated or flipped. Using these 10 tetrominoes, we must cover the board with no overlap or gaps, as in figure 4. There are 783 possible solutions for this puzzle[6].

When thinking about putting tetromino “I” on this board, there are 41 possible placements, as shown in figure 5. We named these placements “q0” - “q40”. The numbers on the tetrominoes in figure 5 are the grid numbers for the board in figure 3.

For example, possible placement “q0” indicates the placement of tetromino “I” in the top left corner of the board without rotation, and “q16” indicates the placement of tetromino “T” in the top left corner of the board with a 90 degree rotation.

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40

Figure 3: Board to tile with tetrominoes

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40

Figure 4: Board tiled with tetrominoes

# Solving tiling puzzles with quantum annealing

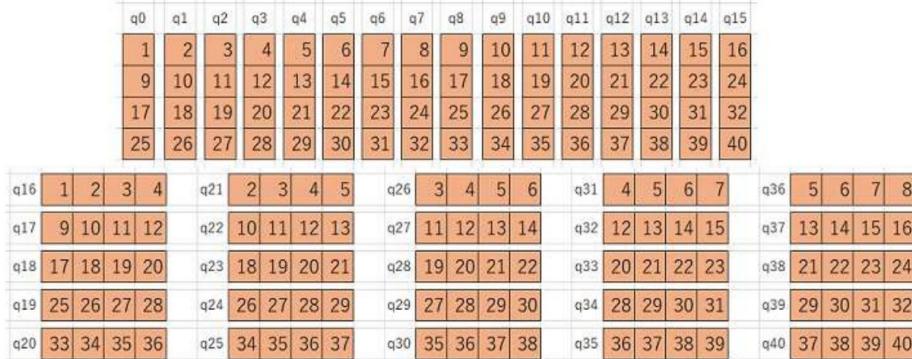


Figure 5: 41 possible placements of tetromino “I”

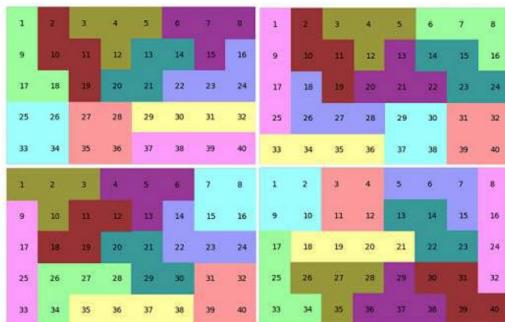


Figure 8: Example of correct solutions

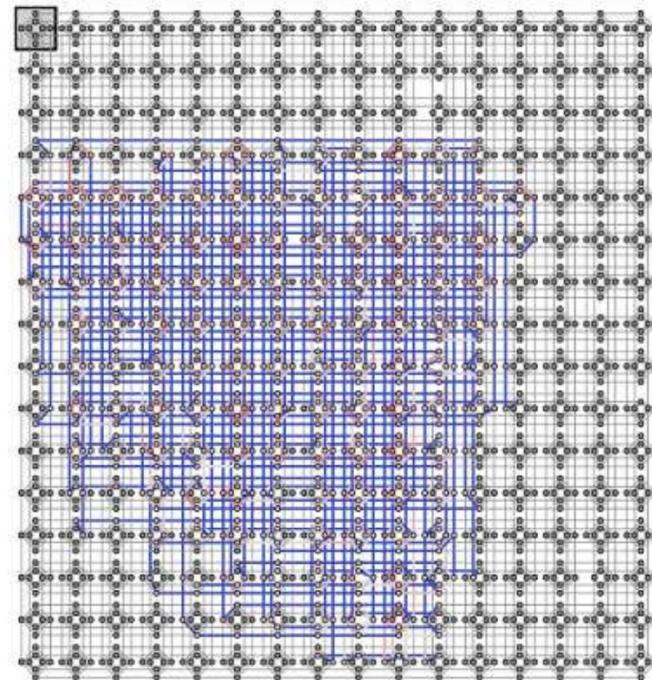


Figure 10: subQUBO implementation on chimera graph

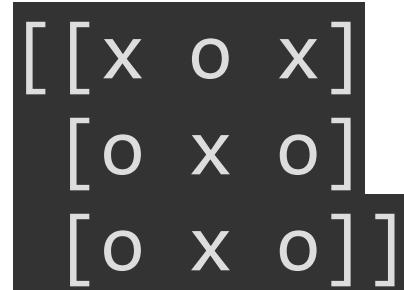
# Gaming

## SUDOKU

1	4	5	3	2	7	6	9	8
8	3	9	6	5	4	1	2	7
6	7	2	9	1	8	5	4	3
4	9	6	1	8	5	3	7	2
2	1	8	4	7	3	9	5	6
7	5	3	2	9	6	4	8	1
3	6	7	5	4	2	8	1	9
9	8	4	7	6	1	2	3	5
5	2	1	8	3	9	7	6	4

SUDOKU

## TIC TAC TOE

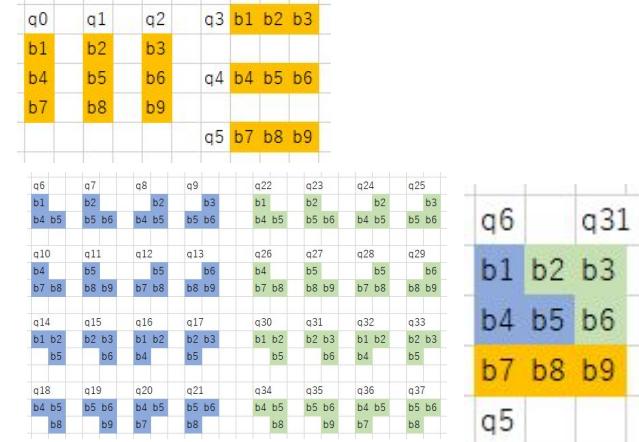


a =

```
Opt().add("10*(q0+q1+q2+q3+q4+q5+
q6+q7+q8-4)^2-(q0+q1+q2)^2-(q3+q4
+q5)^2-(q6+q7+q8)^2-(q0+q3+q6)^2-
(q1+q4+q7)^2-(q2+q5+q8)^2", N=9).a
dd(np.diag([-100, 0, 0, 100, -100, 0, 1
00, -100, 100])).run()
```

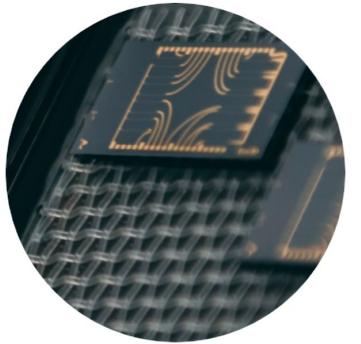
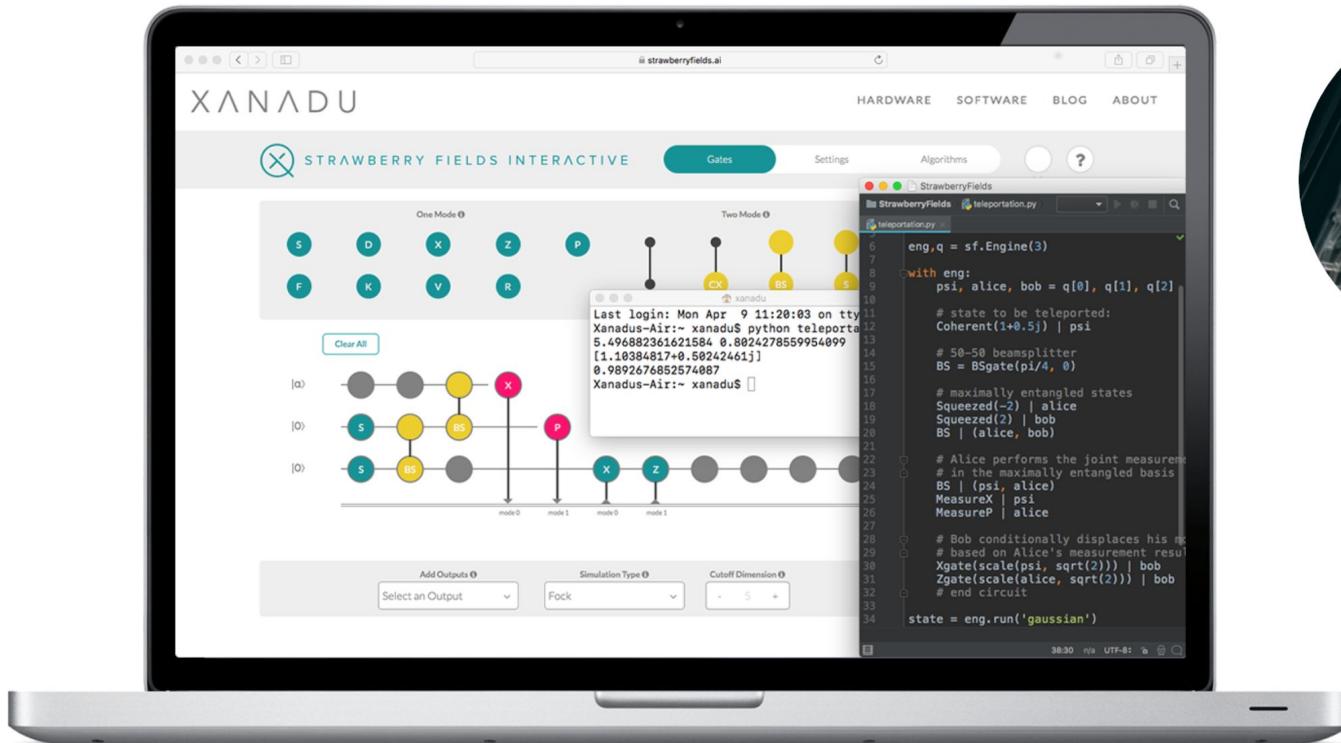
```
print(np.reshape(a,(3,3)))
```

## TILING



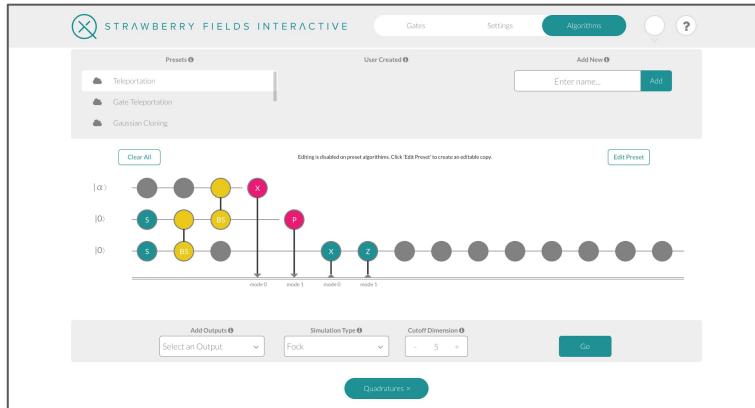
```
a = opt.opt()
a.qubo =
=opt.optm("(1-(q0+q1+q2+q3+q4+q5))^2+(1-(q6+q7+q8+q9+q10+q11+q12+q1
+q14+q15+q16+q17+q18+q19+q20+q21))^2
+(1-(q22+q23+q24+q25+q26+q27+q28+q29+q30+q31+q32+q33+q34+q35+q36+q3
+q37))^2+(1-(q0+q3+q6+q14+q16+q22+q30+q32))^2+(1-(q1+q3+q7+q8+q14+q15+
q16+q17+q23+q24+q30+q31+q32+q33))^2+(1-(q2+q3+q9+q15+q17+q25+q31+q3
+q33))^2+(1-(q0+q4+q6+q8+q10+q16+q18+q20+q22+q24+q26+q32+q34+q36))^2+(1-
(q1+q4+q6+q7+q8+q9+q11+q12+q14+q17+q18+q19+q20+q21+q22+q23+q24+q2
+q25+q28+q30+q33+q34+q35+q36+q37))^2+(1-(q2+q4+q7+q9+q13+q15+q19+q
21+q23+q25+q29+q31+q35+q37))^2+(1-(q0+q5+q10+q11+q12+q13+q18+q21+q26+q27+q28+q29+q34+q37))^2+(1-
(q2+q5+q11+q13+q19+q27+q29+q35))^2", 38)
res = a.sa()
print(res)
print(np.where(np.array(res)==1)[0])
```

# 光量子・連續量・XANADU



# 光連続量プログラミング実例: Teleportation

量子テレポーテーション(りょうしテレポーテーション、英:Quantum teleportation)とは、古典的な情報伝達手段と量子もつれ(Quantum entanglement)の効果を利用して離れた場所に量子状態を転送することである。



The screenshot shows the Strawberry Fields Interactive software interface with the Algorithms tab selected. It displays the generated Python code for the teleportation circuit:

```
#!/usr/bin/env python3
import numpy as np
import strawberryfields as sf
from strawberryfields.ops import *
from strawberryfields.utils import scale

# initialise backend, engine and register
eng, q = sf.Engine(3, hbar=0.5)

with eng:
    Coherent((1+0.5j) | q[0])
    Spade(-2, 0) | q[1]
    Sgate(2, 0) | q[2]
    Bsgate(0.7854, 0) | (q[1], q[2])
    Bsgate(0.7854, 0) | (q[0], q[1])
    MeasureX | q[1]
    MeasureX | q[0]
    Xgate(scale(q[0], 1.41421356237000)) | q[2]
    Zgate(scale(q[1], 1.41421356237000)) | q[2]

state = eng.run('fock', cutoff_dim=5)
```

# 光連続量実例: Teleportation : Blackbird code

The screenshot shows the Strawberry Fields Interactive web application interface. At the top, there is a navigation bar with the logo "STRAWBERRY FIELDS INTERACTIVE", a "Gates" button, a "Settings" button, a "Algorithms" button (which is highlighted in teal), and a help icon. Below the navigation bar, there are four tabs: "Quadratures x", "Wigner Function x", "Blackbird Code x" (which is also highlighted in teal), and "Fock States x". The main content area displays a quantum circuit code in a dark-themed code editor. The code is written in Python using the strawberryfields library. It initializes a backend engine, creates three qubits, and performs a sequence of gates including Coherent, Sgate, BSgate, MeasureP, MeasureX, Xgate, and Zgate. Finally, it runs the engine with a cutoff dimension of 5.

```
#!/usr/bin/env python3
import numpy as np
import strawberryfields as sf
from strawberryfields.ops import *
from strawberryfields.utils import scale

# initialise backend, engine and register
eng, q = sf.Engine(3, hbar=0.5)

with eng:
    Coherent(1+0.5j) | q[0]
    Sgate(-2, 0) | q[1]
    Sgate(2, 0) | q[2]
    BSgate(0.7854, 0) | (q[1], q[2])
    BSgate(0.7854, 0) | (q[0], q[1])
    MeasureP | q[1]
    MeasureX | q[0]
    Xgate(scale(q[0], 1.41421356237000)) | q[2]
    Zgate(scale(q[1], 1.41421356237000)) | q[2]

state = eng.run('fock', cutoff_dim=5)
```

# 注目の技術



Company Resources

Request Access

NEW IonQ publishes new benchmarks for quantum computation ▶

## A true quantum leap.

Introducing the first commercial trapped ion quantum computer. By manipulating individual atoms, it has the potential to one day solve problems beyond the capabilities of even the largest supercomputers.

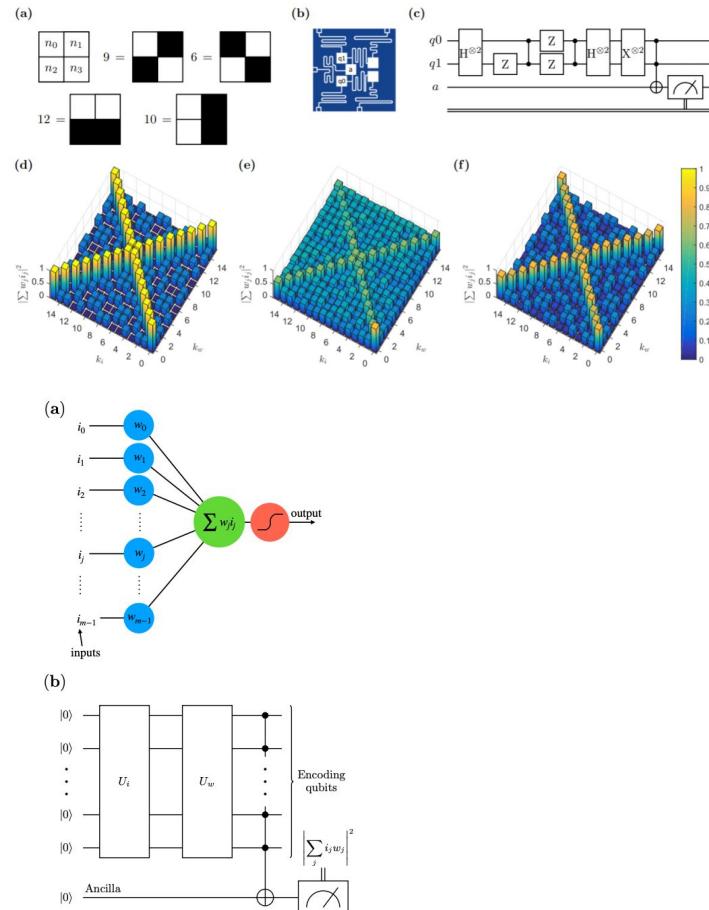
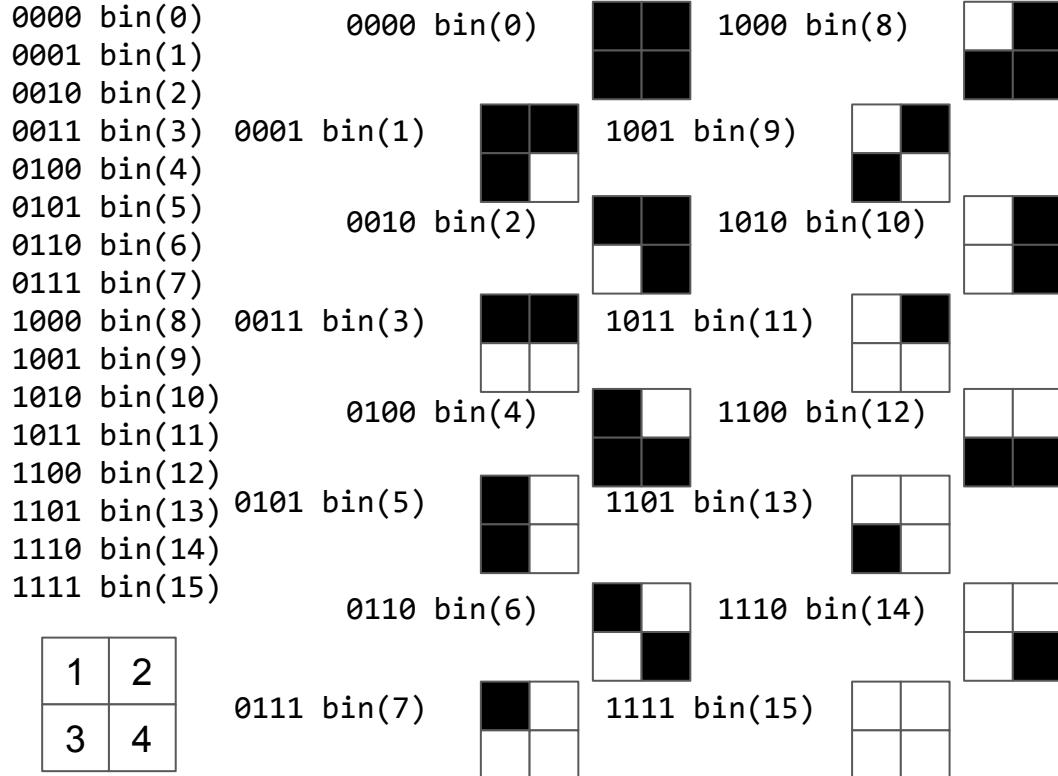
[Request Access](#)



# An Artificial Neuron Implemented on an Actual Quantum Processor



<https://arxiv.org/abs/1811.02266>



# 2量子ビットと状態ベクトルを使い分ける

2量子ビットで表現できる重ね合わせ(組み合わせ)状態は、 $2^2 = 4$ 通り。ただ、これでパターンを作るわけではなく、4ビットをつくる。

```
|0> ---H---  
|0> ---H---
```

# $[1,0]$ ベクトルの初期化でテンソル積を取る。

```
qq = np.kron([1,0],[1,0]) = [1,0,0,0]
```

#アダマールゲートを準備。テンソル積をとって準備。

```
H = [[1,1],[1,-1]]
```

```
HH = np.kron(H,H)
```

#準備

```
HH@qq = [1,1,1,1]
```

準備完了！

1	2
3	4

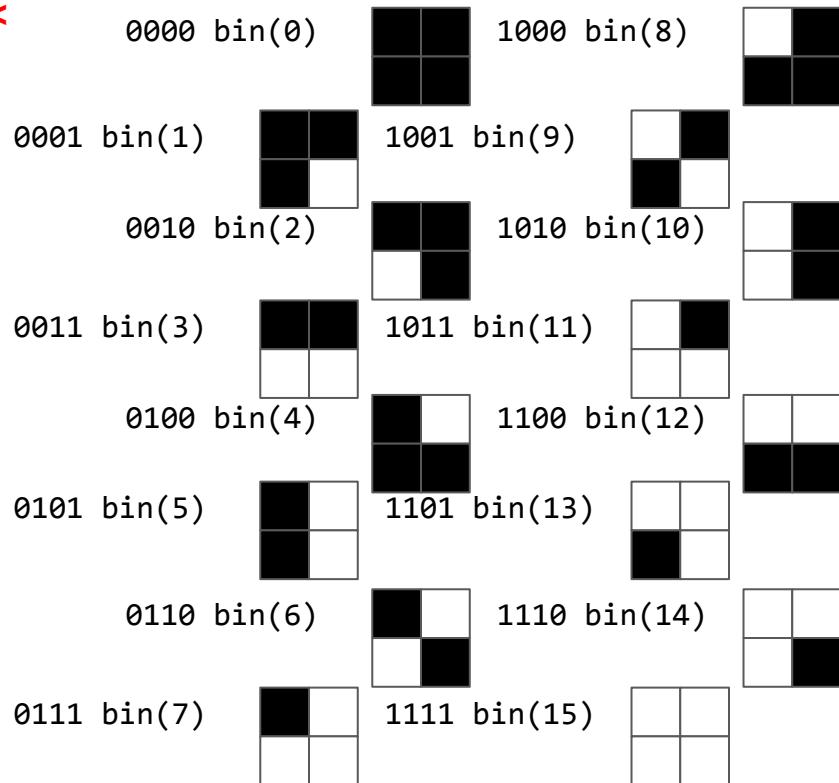
# 状態ベクトルの位相を使ってパターンを生成

位相を使って+1を-1に。-1を0に読み替えてください><

1	2
3	4

[1,1,1,1]

0000 bin(0)  
0001 bin(1)  
0010 bin(2)  
0011 bin(3)  
0100 bin(4)  
0101 bin(5)  
0110 bin(6)  
0111 bin(7)  
1000 bin(8)  
1001 bin(9)  
1010 bin(10)  
1011 bin(11)  
1100 bin(12)  
1101 bin(13)  
1110 bin(14)  
1111 bin(15)



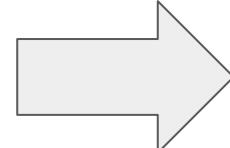
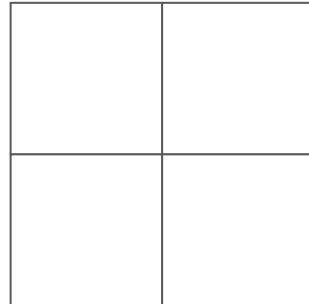
# 基本のゲートはCZやZゲート

ここではCZやZを使って位相を操作。 $+1$ は $|+\rangle$ 状態を、 $-1$ の $|-\rangle$ 状態へ任意の状態ベクトルを操作。

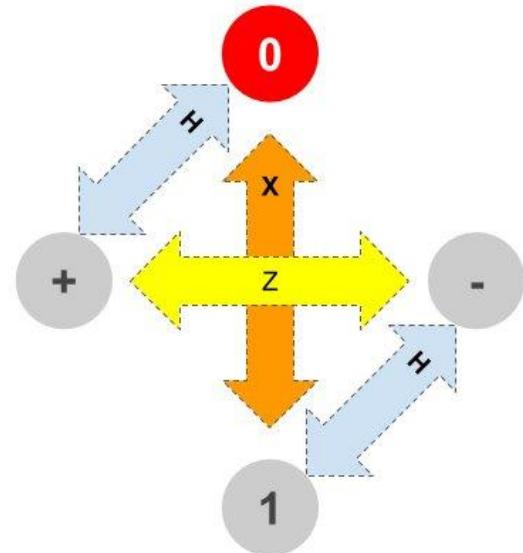
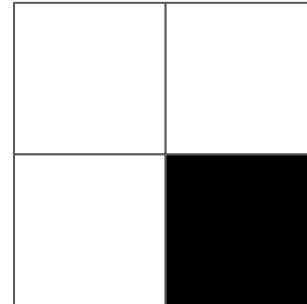
$CZ = [[1, 0, 0, 0], [0, 1, 0, 0], [0, 0, 1, 0], [0, 0, 0, -1]]$

$CZ@[1, 1, 1, 1] = [1, 1, 1, -1]$

1111 bin(15)



1110 bin(14)



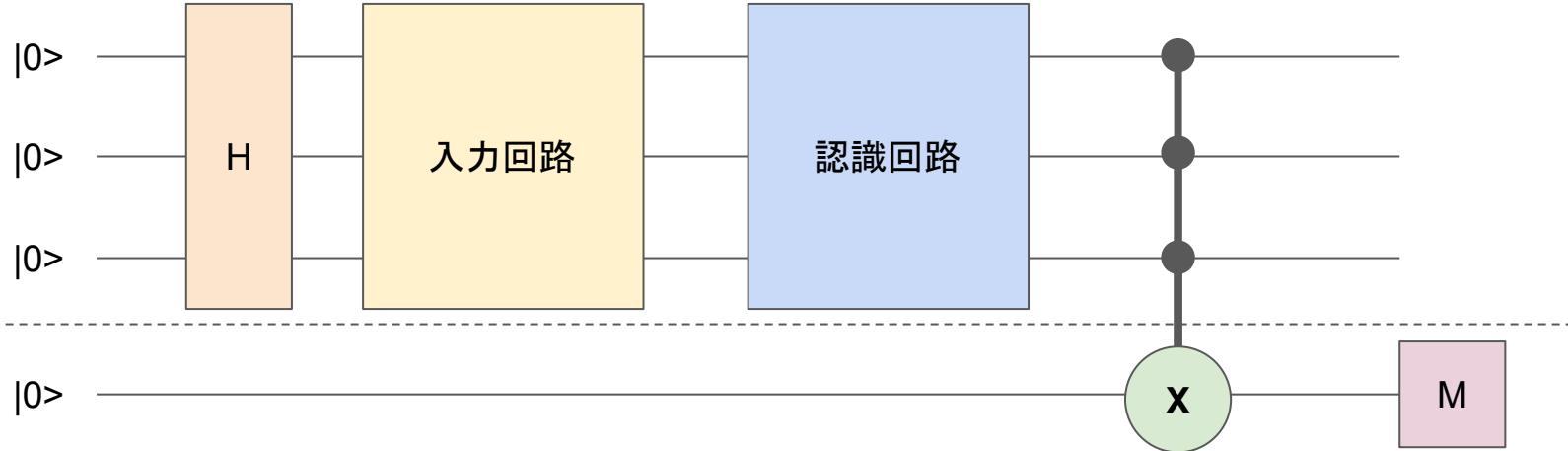
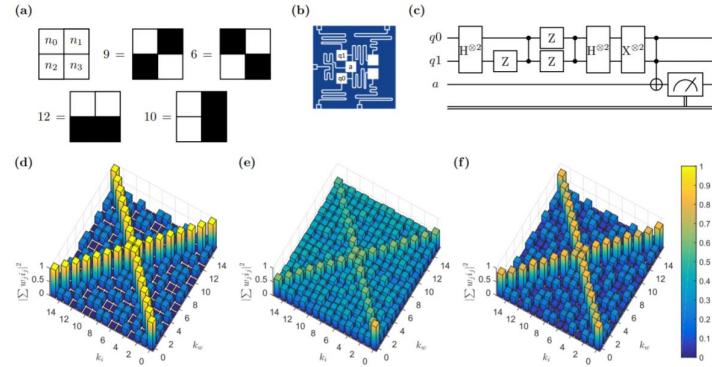
ちなみに

$Z = [[1, 0], [0, -1]]$

# 回路の基本ステップは3ステップ

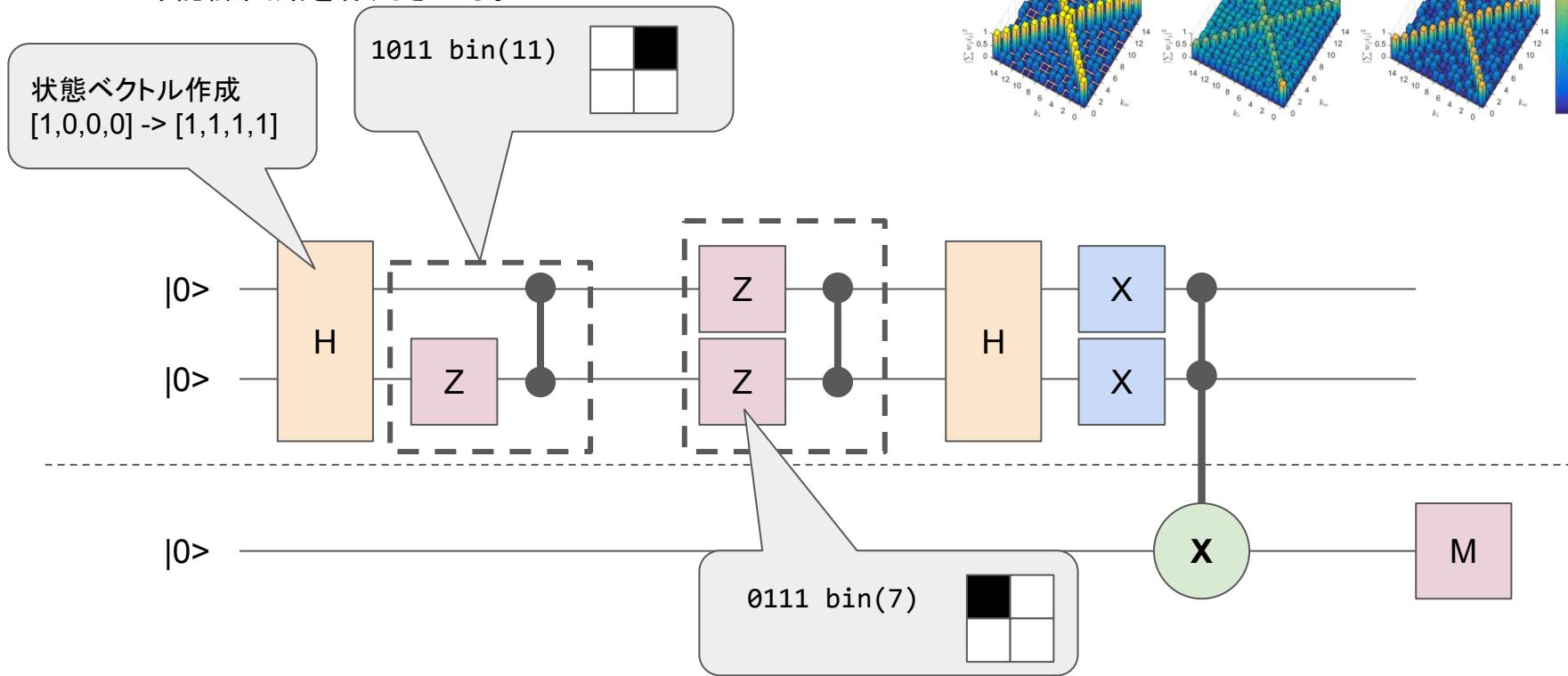
- ・状態ベクトルの準備
- ・入力回路
- ・認識回路

活性化関数の代わりに制御ゲートを使います。



# 例題

論文の通り、 $k=11$ というパターン入力に対して、 $k=7$ という認識回路を作用させる。



# 入力inputと結合wの表現

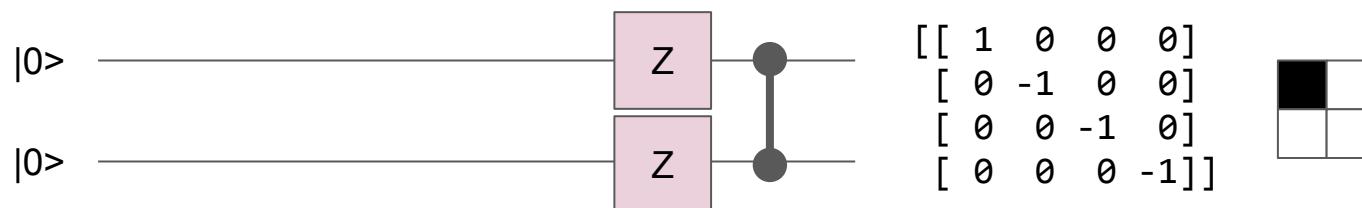
$Z = [[1, 0], [0, -1]]$

$I = [[1, 0], [0, 1]]$

$CZ = [[1, 0, 0, 0], [0, 1, 0, 0], [0, 0, 1, 0], [0, 0, 0, -1]]$



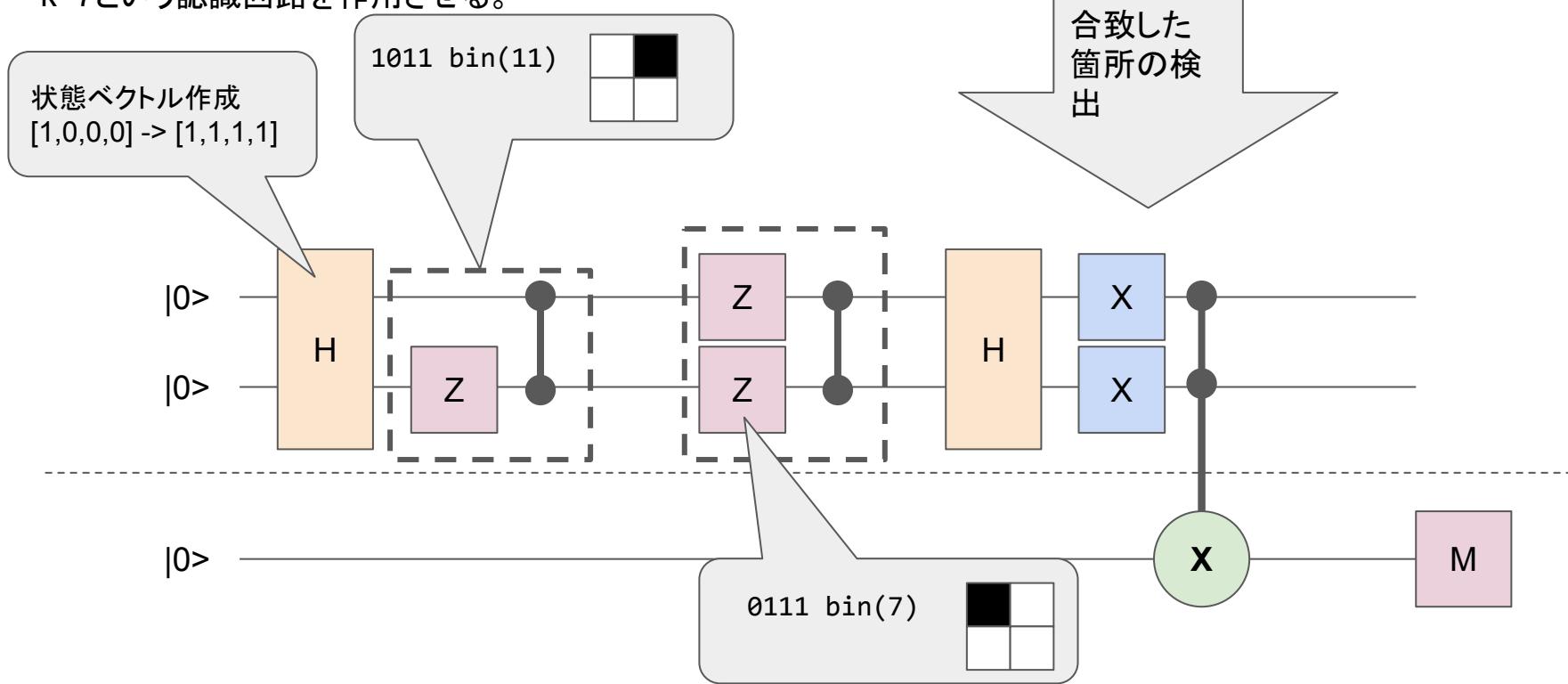
$CZ @ np.kron(I, Z)$



$CZ @ np.kron(Z, Z)$

# 認識

論文の通り、 $k=11$ というパターン入力に対して、 $k=7$ という認識回路を作用させる。



# まとめ

今年は変分型と並行して、状態ベクトルを活用した汎用量子計算型の機械学習がひととおり試される年。

以上。