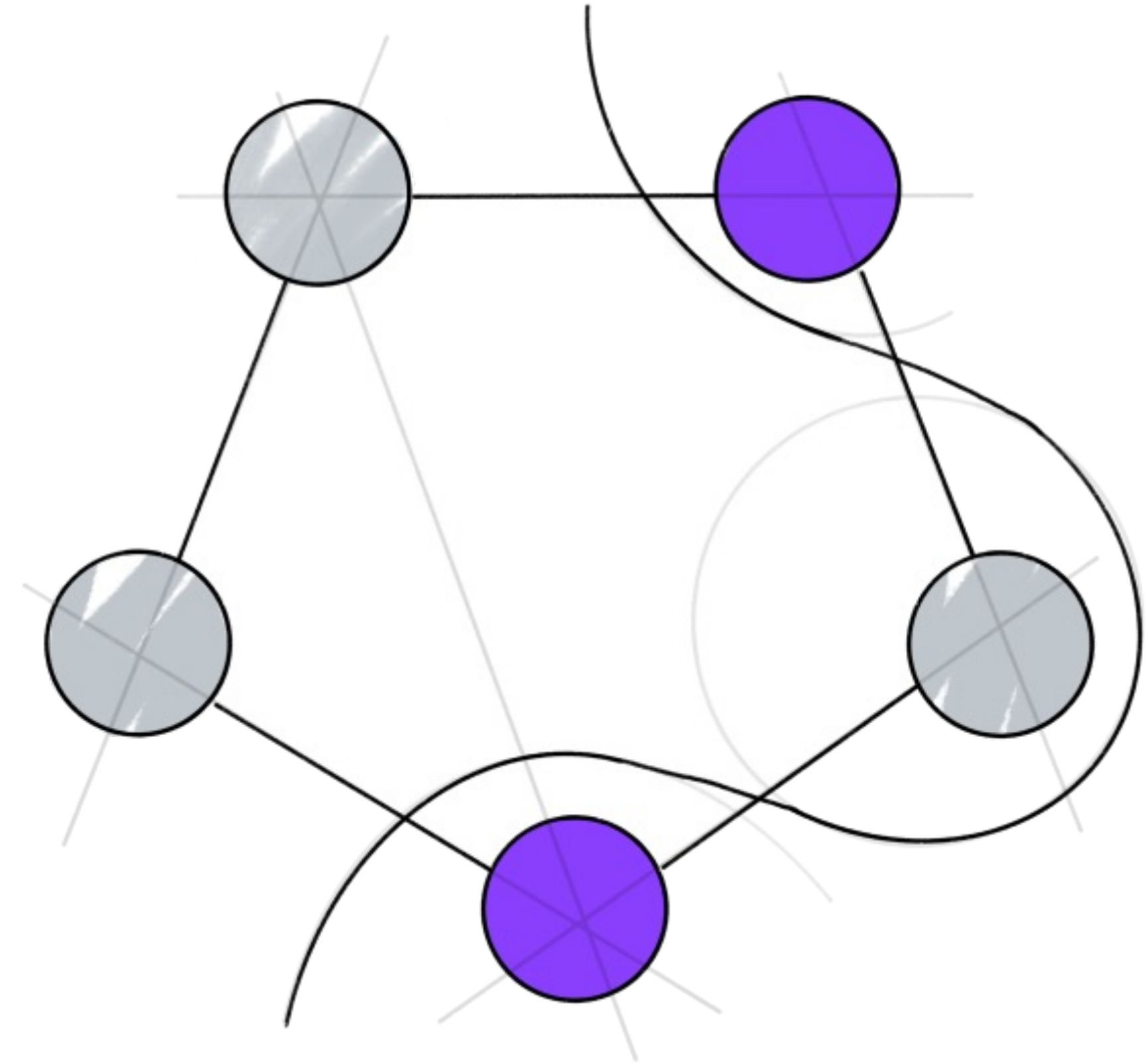


# 最適化問題

小林 有里  
Yuri Kobayashi  
IBM Quantum



# 海外から来た友達と遊園地へ





# アトラクションを回るルートを考える

午後6時に入園して、閉園時間午後10時まで、どの順番でアトラクションを回ると最も無駄なく移動できるか？



考慮（こうりょ）すべきことは何か

- ・ 訪問する順番
- ・ 移動距離



# アトラクションを回るルートを考える

午後6時に入園して、閉園時間午後10時まで、どの順番でアトラクションを回ると最も無駄なく移動できるか？



9個だけ好きなアトラクションを選んだとして、巡回路は何通りあるか？

## 9の階乗

$$\begin{aligned} 9! &= 9 \times 8 \times 7 \times 6 \times 5 \times 4 \times 3 \times 2 \times 1 \\ &= 362880 \text{通り} \end{aligned}$$



# 現実的にはもっと多くの変数を考慮すべき

午後6時に入園して、閉園時間午後10時まで、どの順番でアトラクションを回ると最も**満足度**を高められるか？



考慮すべきことは何か？

- ・ どのアトラクションを選択するか
- ・ アトラクションの数
- ・ 移動距離
- ・ 待ち時間
- ・ 乗車時間
- ・ 閉園までの時間
- ・ アトラクションの好み（満足度）



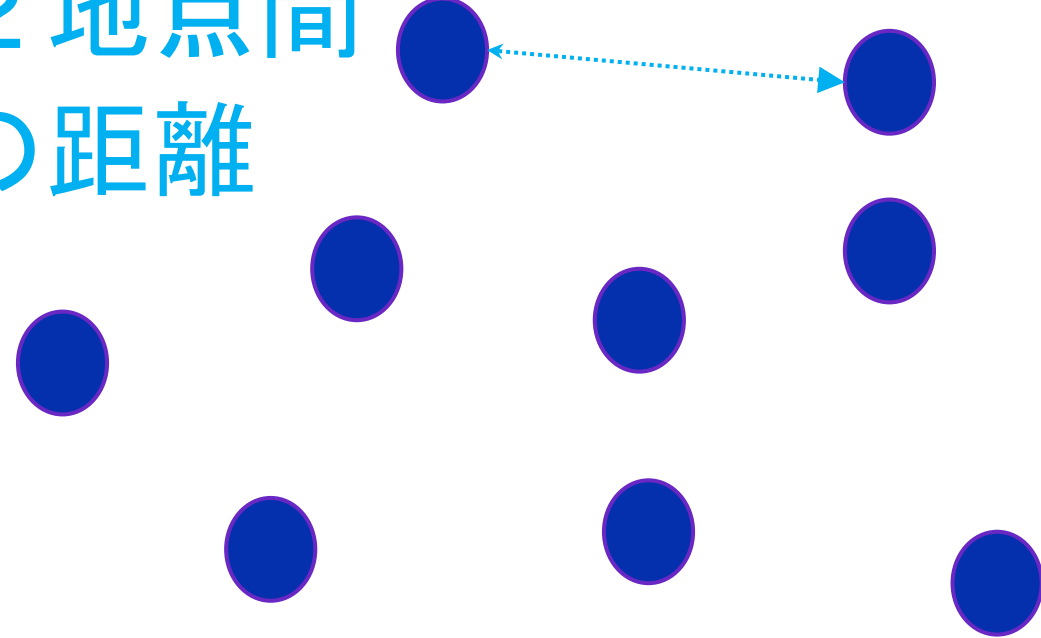
# 巡回セールスマン問題

与えられるデータ：n個の地点と2地点間の距離

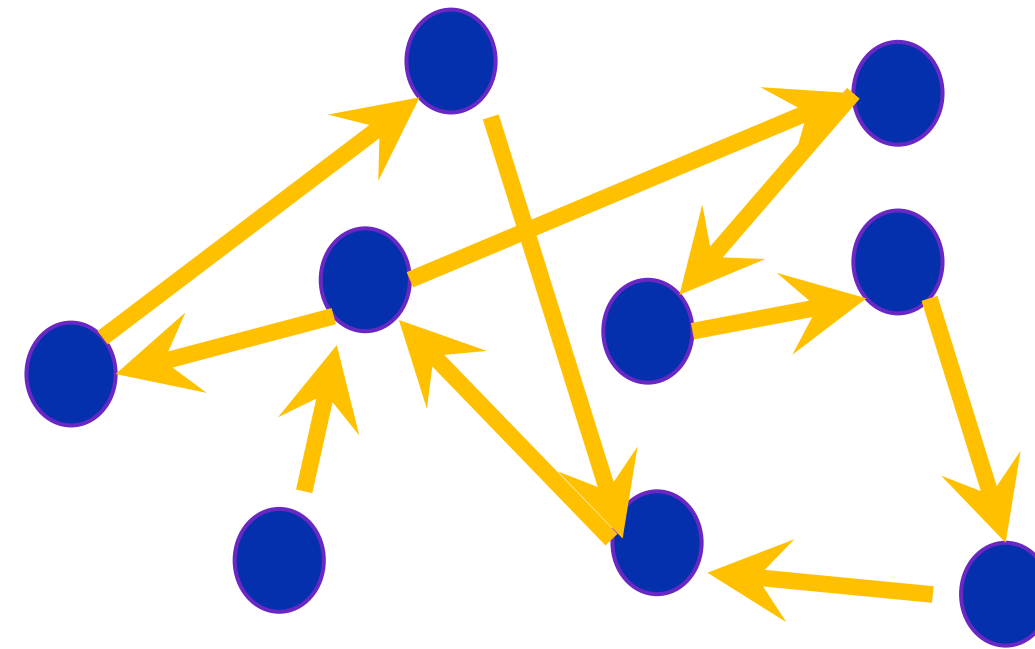
条件：すべての地点を1度ずつ通り元の地点に戻る

目標：総距離をできるだけ短くしたい

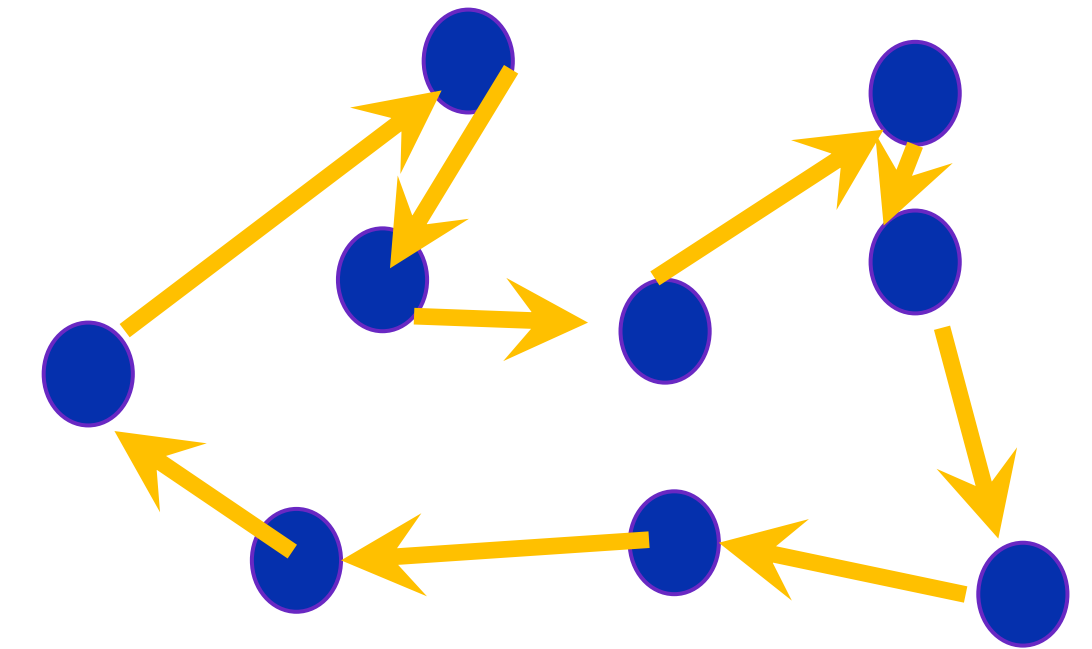
2地点間の距離



N個の地点



ある巡回路

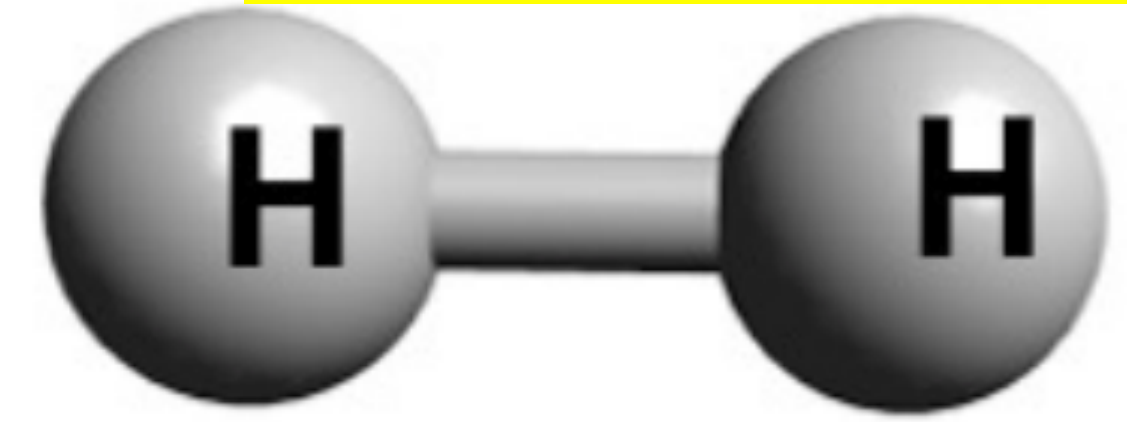


別の巡回路

巡回路の選び方は  $n! = n(n-1)(n-2) \cdots 2 \cdot 1$  通り  $n \rightarrow \infty$   **$n!$**   $\rightarrow$  組合せ爆発!

ふつうのコンピューターでは手に負えないこの問題を  
量子コンピューターはどうアプローチして解くのでしょうか

# 水素分子のエネルギーを求める



エネルギーを求めるためには、シュレディンガー方程式を解く必要があります。

$$\hat{H} |\psi\rangle = E |\psi\rangle$$

ハミルトニアン：波動関数からエネルギーを取り出す**演算子**。中身は分子によって変わる。

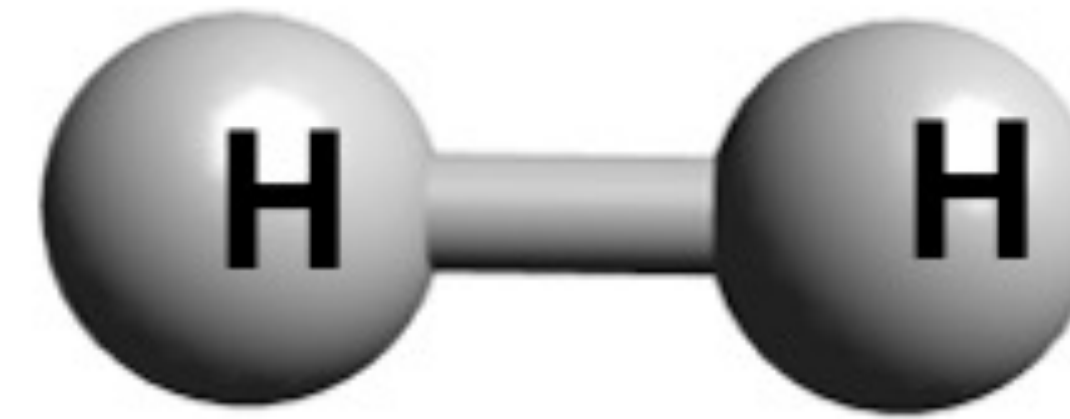
エネルギー：波動関数に固有のエネルギー

波動関数：粒子の位置情報が保存されている関数/ベクトル

参考) 演算子：関数に対して行う演算を記号で表現したもの。例えば、 $d/dx$ は関数を $x$ で微分する演算子とみなせる。



# 水素分子のエネルギーを求める

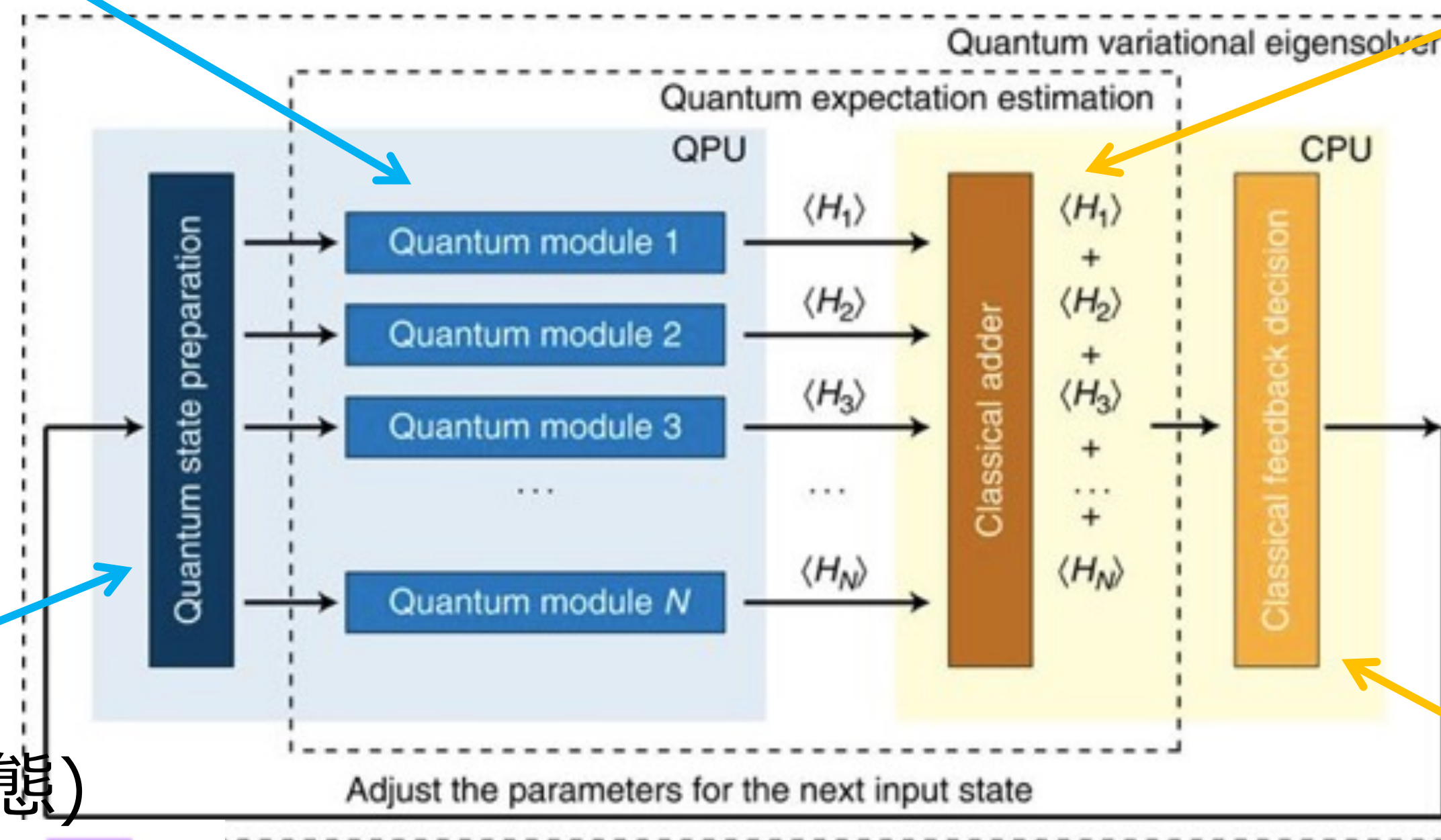


(1) 水素のエネルギー式 (ハミルトニアン)

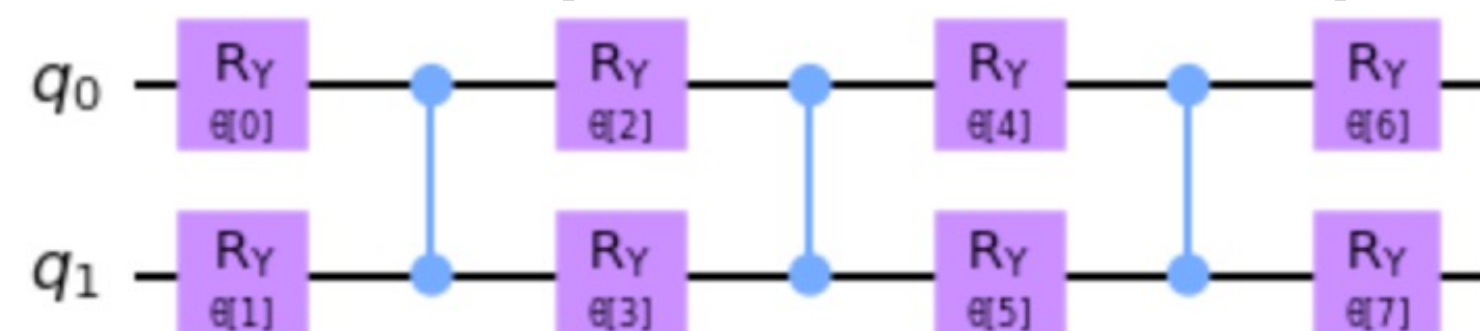
$$\begin{aligned} H2_{op} = & (-1.052373245772859 * I \wedge I) + \backslash \\ & (0.39793742484318045 * I \wedge Z) + \backslash \\ & (-0.39793742484318045 * Z \wedge I) + \backslash \\ & (-0.01128010425623538 * Z \wedge Z) + \backslash \\ & (0.18093119978423156 * X \wedge X) \end{aligned}$$

よく使う量子ゲートであるX演算やZ演算で表現されている。

(3) エネルギーの計算  
(測定と合算)



(2) パラメーター付き  
量子回路 (トライアル状態)



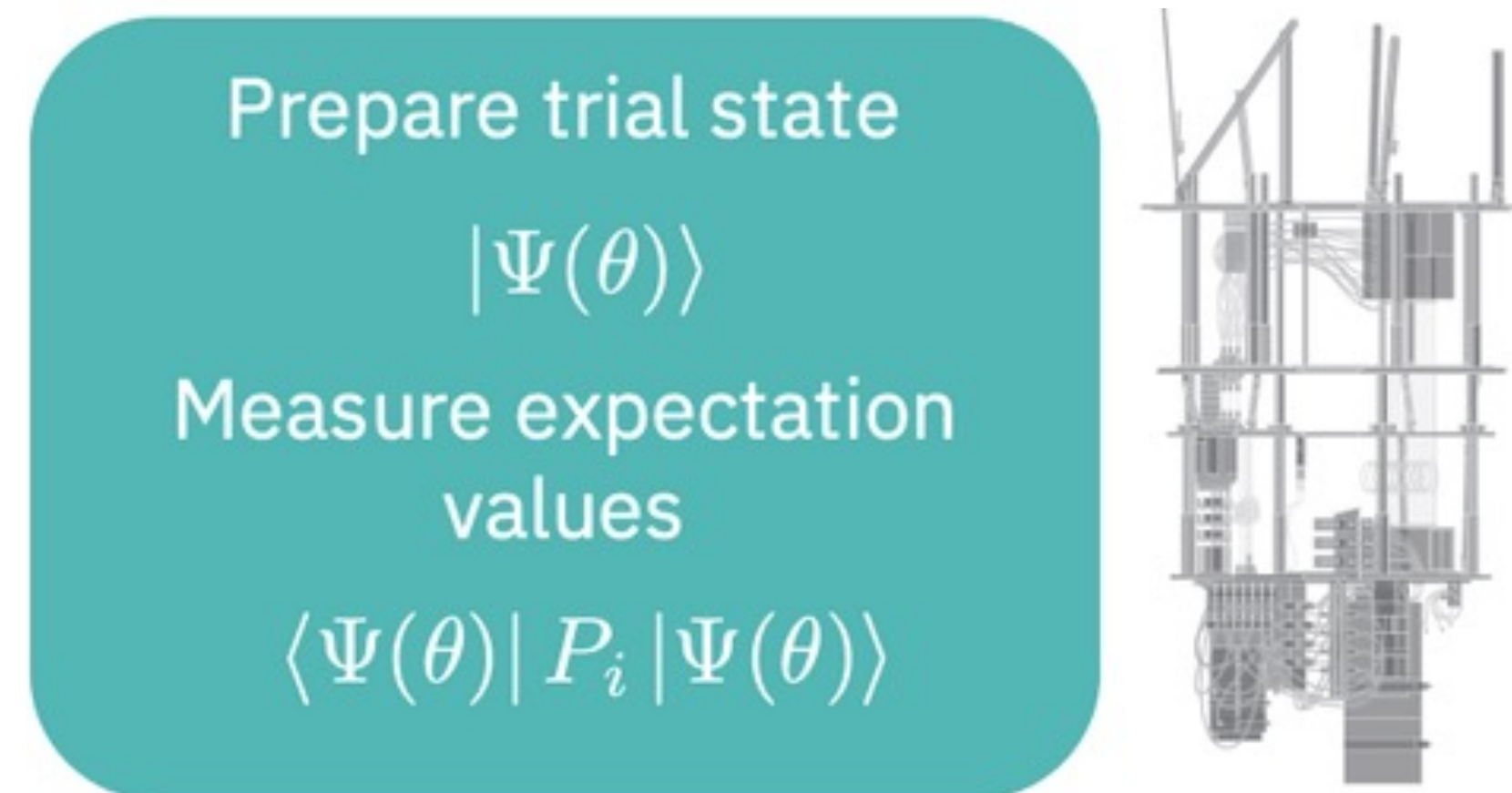
(4) 古典コンピュータで  
最適化ルーチンを使って  
パラメーター調整

パラメーターを変化させ、最も低いエネルギーを求める



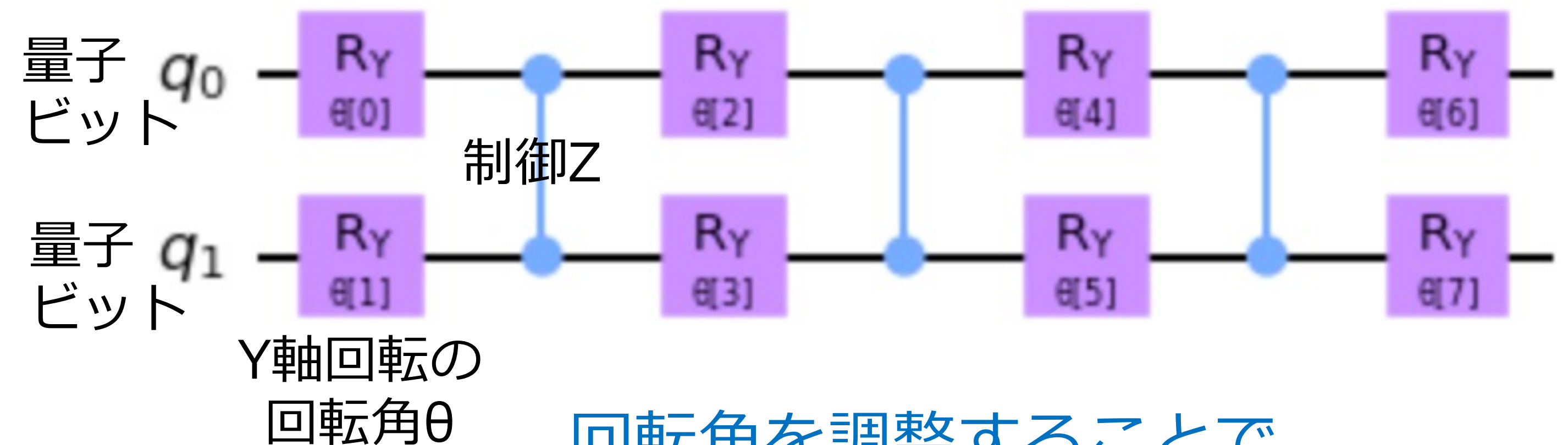
# パラメーター付き量子計算

## 量子コンピューター

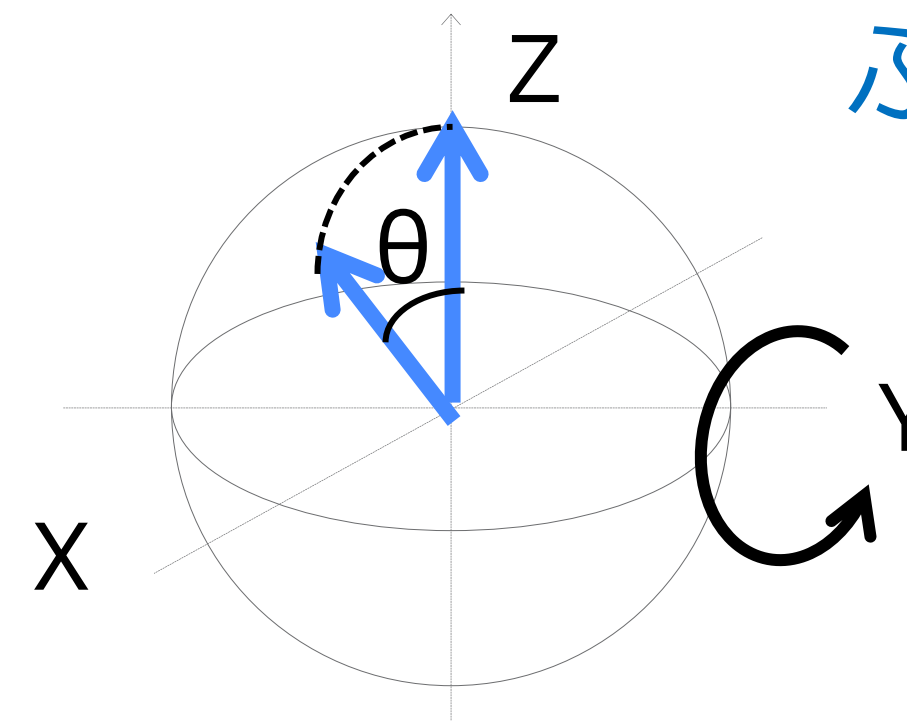


パラメーター付き量子回路で  
コスト関数を表現

## 2量子ビットのパラメーター付き回路の例



回転角を調整することで  
ふさわしい量子回路に調整していく



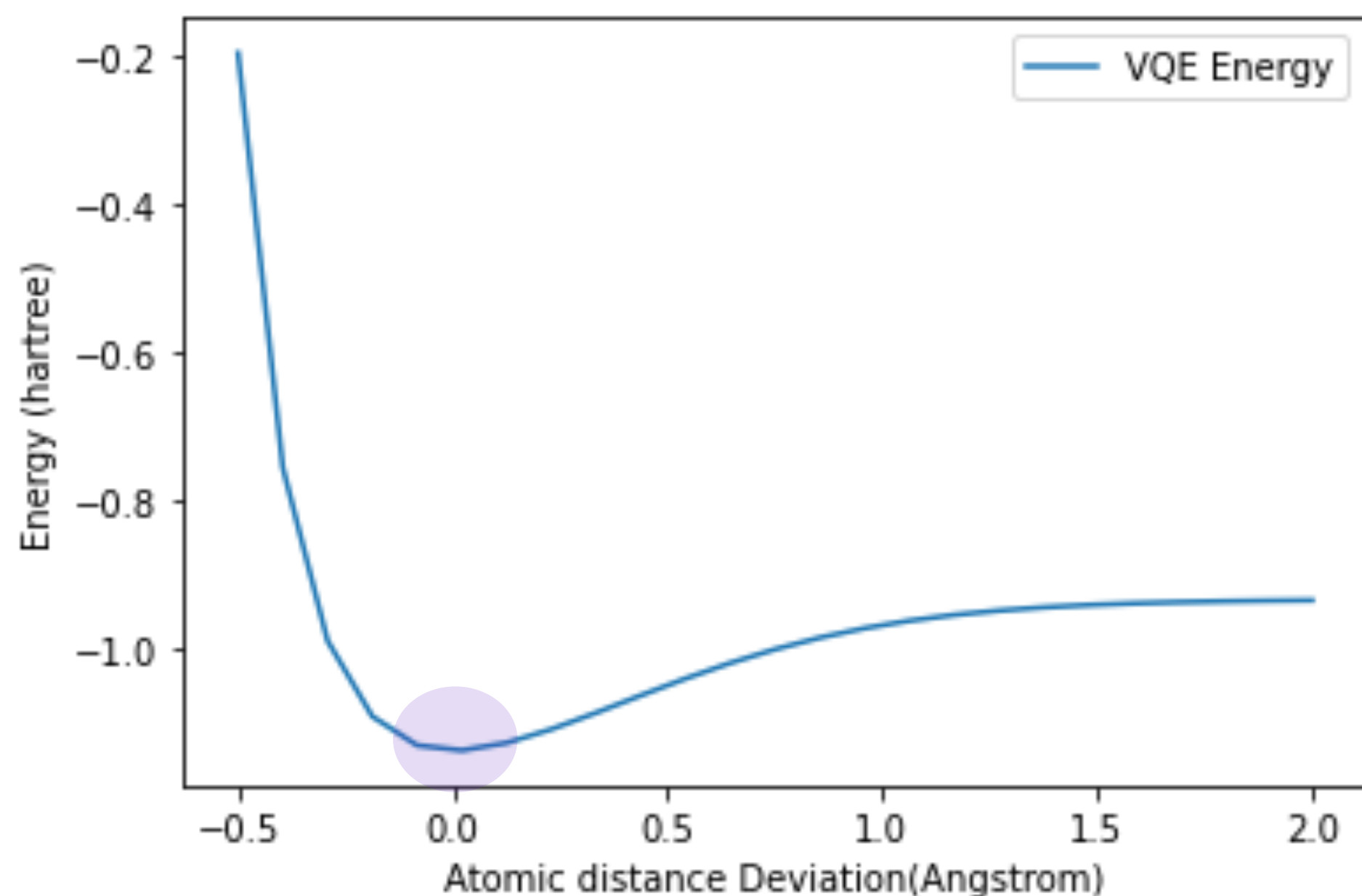
パラメーター付き量子回路で表すもの：  
分子のエネルギー、最適化問題の目的関数（コスト関数）、など



# パラメーター付き量子回路で解く

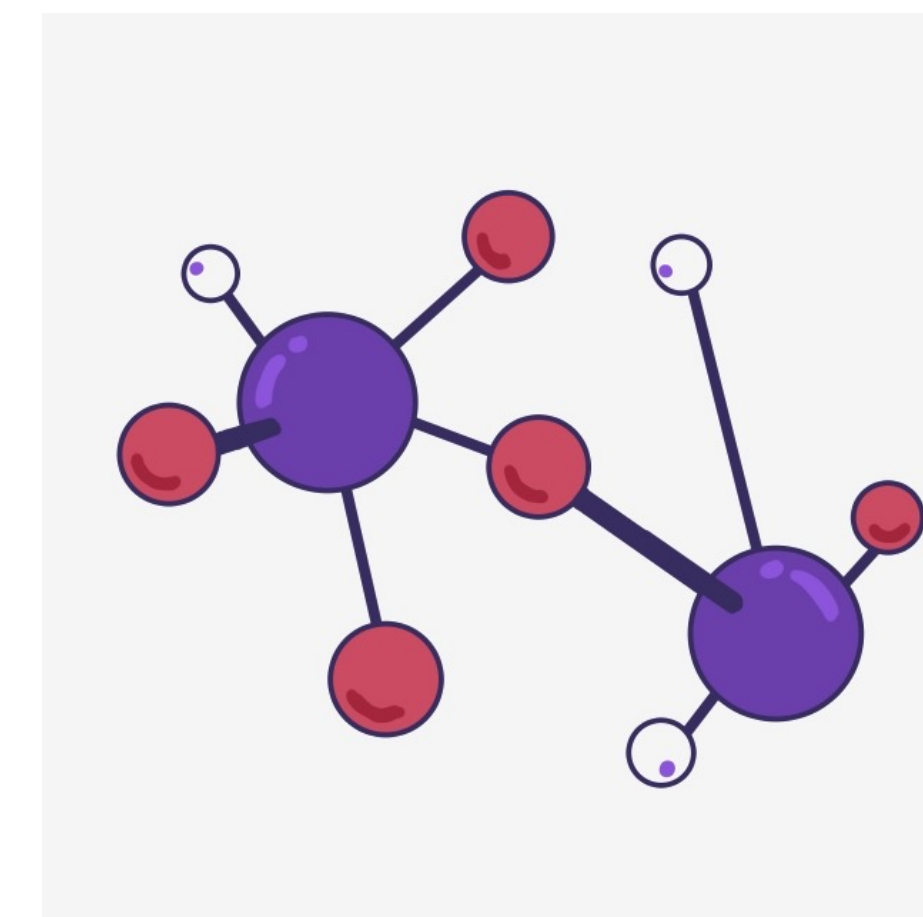
従来のコンピューターだと指数関数的な計算時間がかかる巡回セールスマンのような問題を量子コンピューターはどのようなアプローチで解くのでしょうか。

分子の基底エネルギーを求めるとき同様、都市のまわり方における最短経路の分布を目的関数として定式化して、一番低いエネルギー状態（基底状態）をみつけると、それが探している解となるようなアプローチです。



最適解

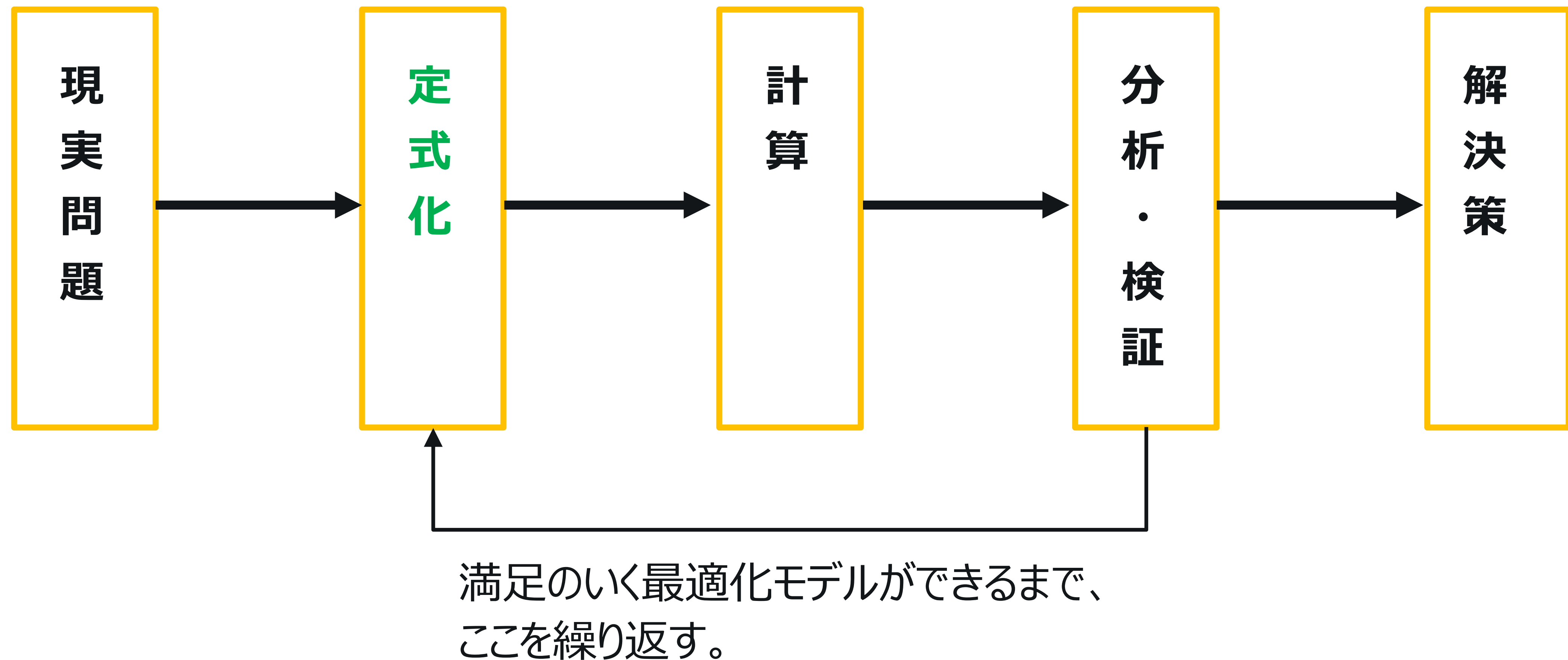
分子の基底エネルギーを求めるアルゴリズム (VQE, QAOA等) を、より一般的な最適化問題を解くことに応用できる。





# 最適化問題の流れ

解こうとしている問題を数式でかきあらわすことを「**定式化**」といいます。





# 定式化とは？

解きたい問題を数式であらわすこと

目的関数・・・最小化または最大化したいもの

- 最小化：コスト、距離、重量、処理時間
  - 例）AからBへ最短時間で行く経路を求める
- 最大化：利益、価値、生産高、効率、満足度
  - 例）300円の予算で最も満足度の高い遠足に  
持っていくお菓子を決める





# 定式化のお作法

目的関数：最小化または最大化したいもの  
(最小化：園内の移動距離、最大化：満足度、報酬)

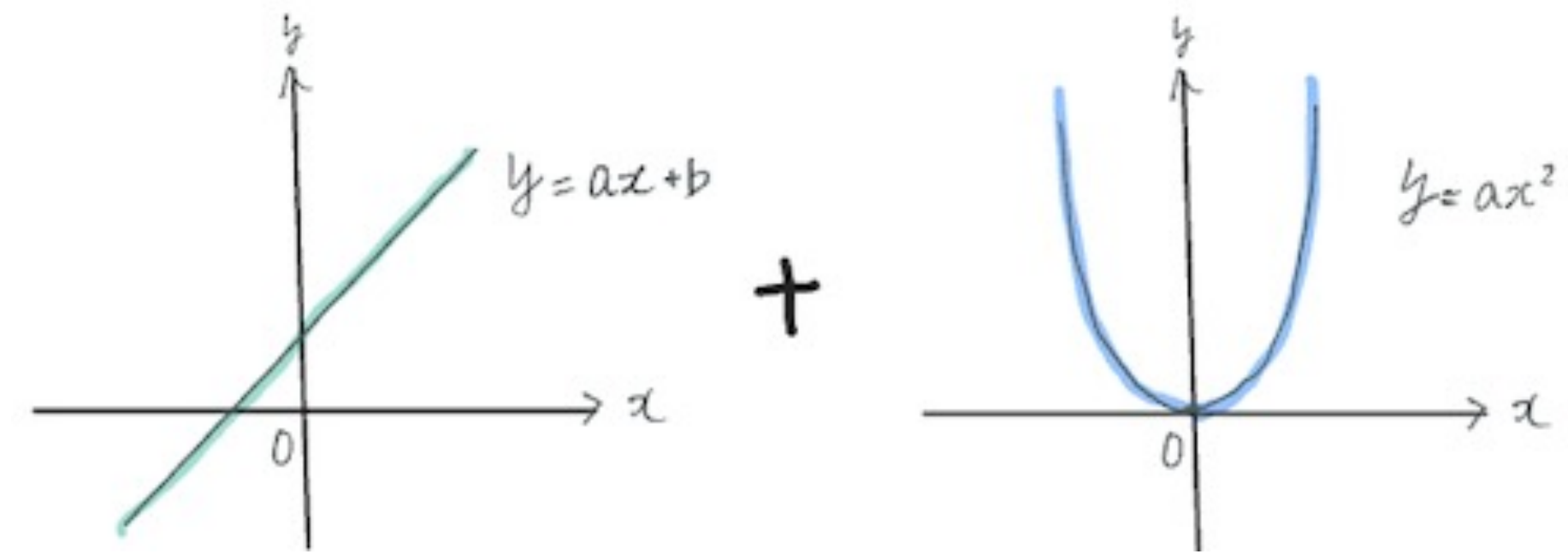
(最小化) Minimize ~ (最大化) Maximize ~

制約条件：探すべき変数がある集合に含まれることを指す  
(閉園時間まで  $t=4$  時間、アトラクション数、2回連続で  
同じアトラクションには乗車しないルール, etc.)

Subject to ~ 具体的な制約条件の記述

# 目的関数

目的関数： **1 次式** + **2 次式**  
(線形式)



例)

変数が1つの場合： **2x** + **4** + **x<sup>2</sup>**

変数が2つの場合： **2x<sub>1</sub>** + **x<sub>2</sub>** + **4 x<sub>1</sub><sup>2</sup>** + **x<sub>1</sub>x<sub>2</sub>**

変数が3つの場合： **2x<sub>1</sub>** + **1x<sub>2</sub>** + **4x<sub>3</sub>** + **2x<sub>1</sub>x<sub>2</sub>** + **4x<sub>1</sub>x<sub>3</sub>** + **1x<sub>2</sub>x<sub>3</sub>**

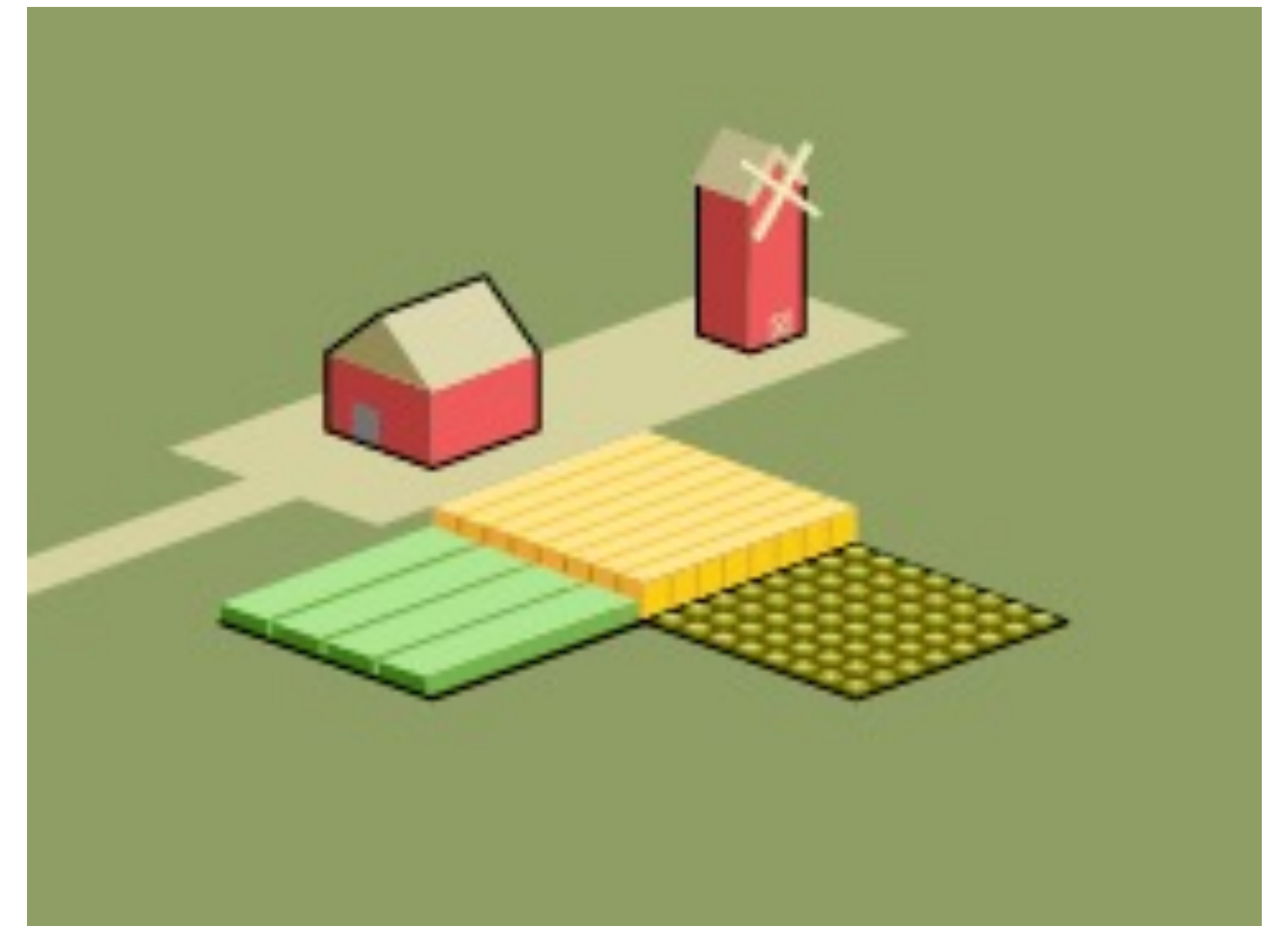
変数の前についている係数は、各変数の特徴量として、過去データから予め提供されることが多い

ある制約条件のもとで 2 次式の最小値または最大値を求める問題を

「**2 次計画問題** (quadratic program)」と呼びます。



# 今日の演習課題： 「農地の収穫量最適化問題」





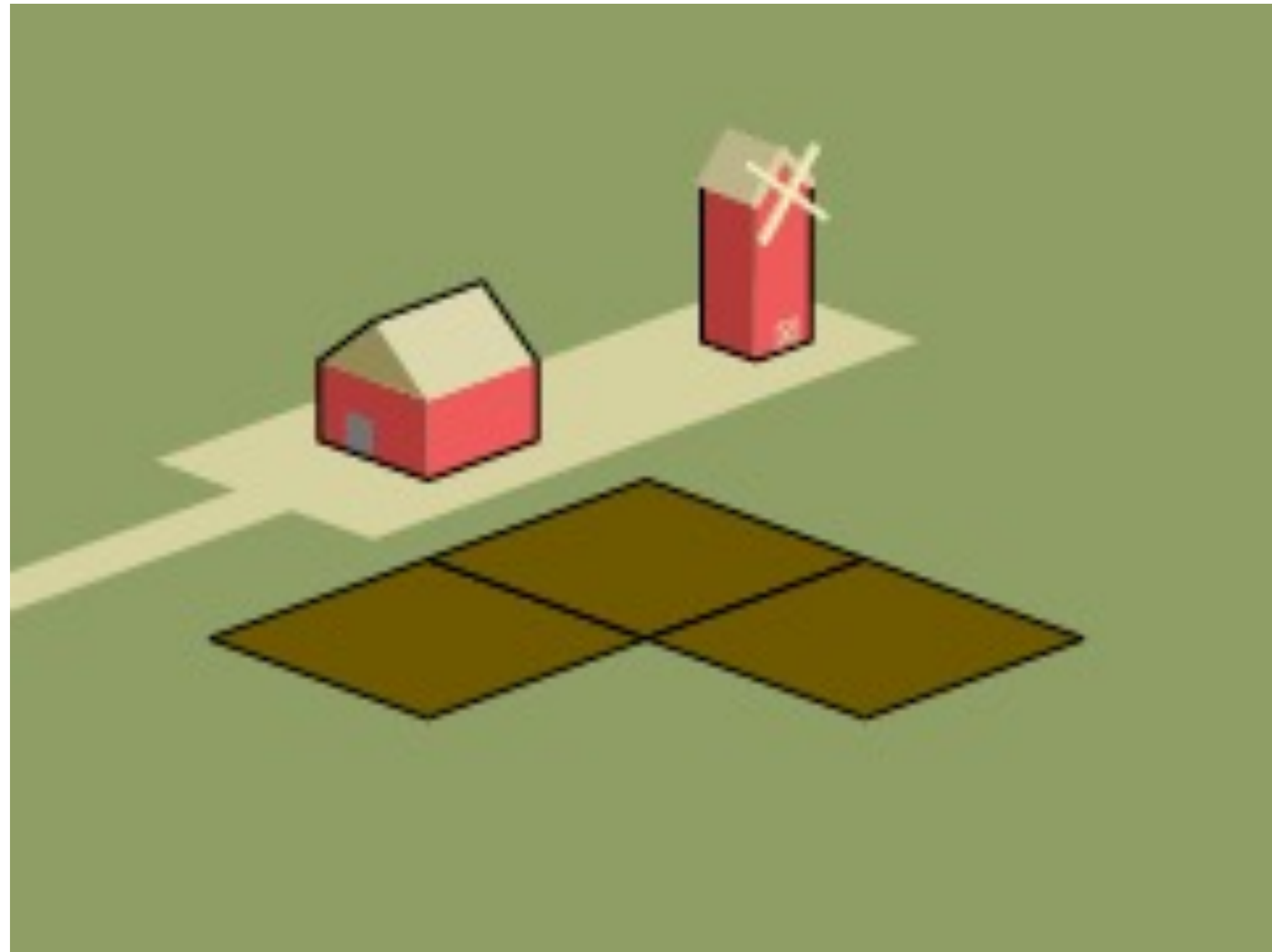
# IBM Quantum Challenge Africa 2021

2021年の IBM Quantum Challengeでは、アフリカのQiskit Communityメンバーが、現在のアフリカが抱える課題をテーマに問題設定を行いました。そのうちの 하나가「**農業の生産性**」というテーマでした。アフリカの農業は生産性の観点から、人口増加による食料需要に供給が追いつかないなどの課題があることが指摘されています。

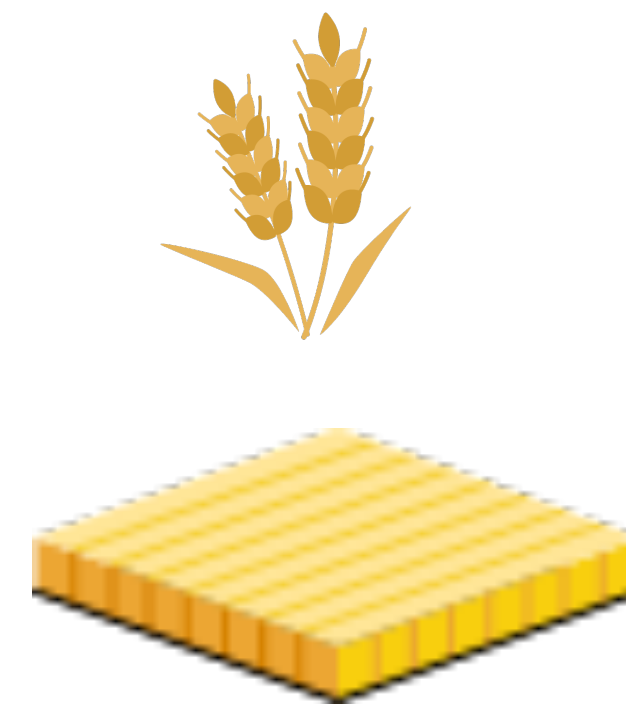




# 農地の収穫量を最大化するには？



3 ヘクタール



Wheat

麦



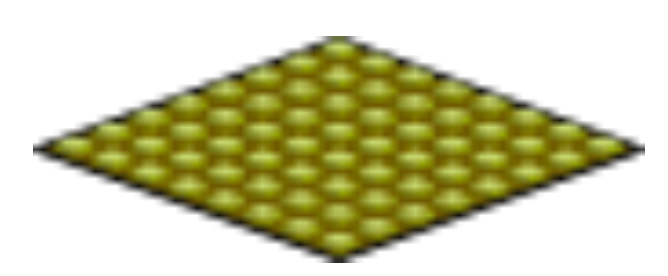
Soybeans

大豆



Maize

トウモロコシ



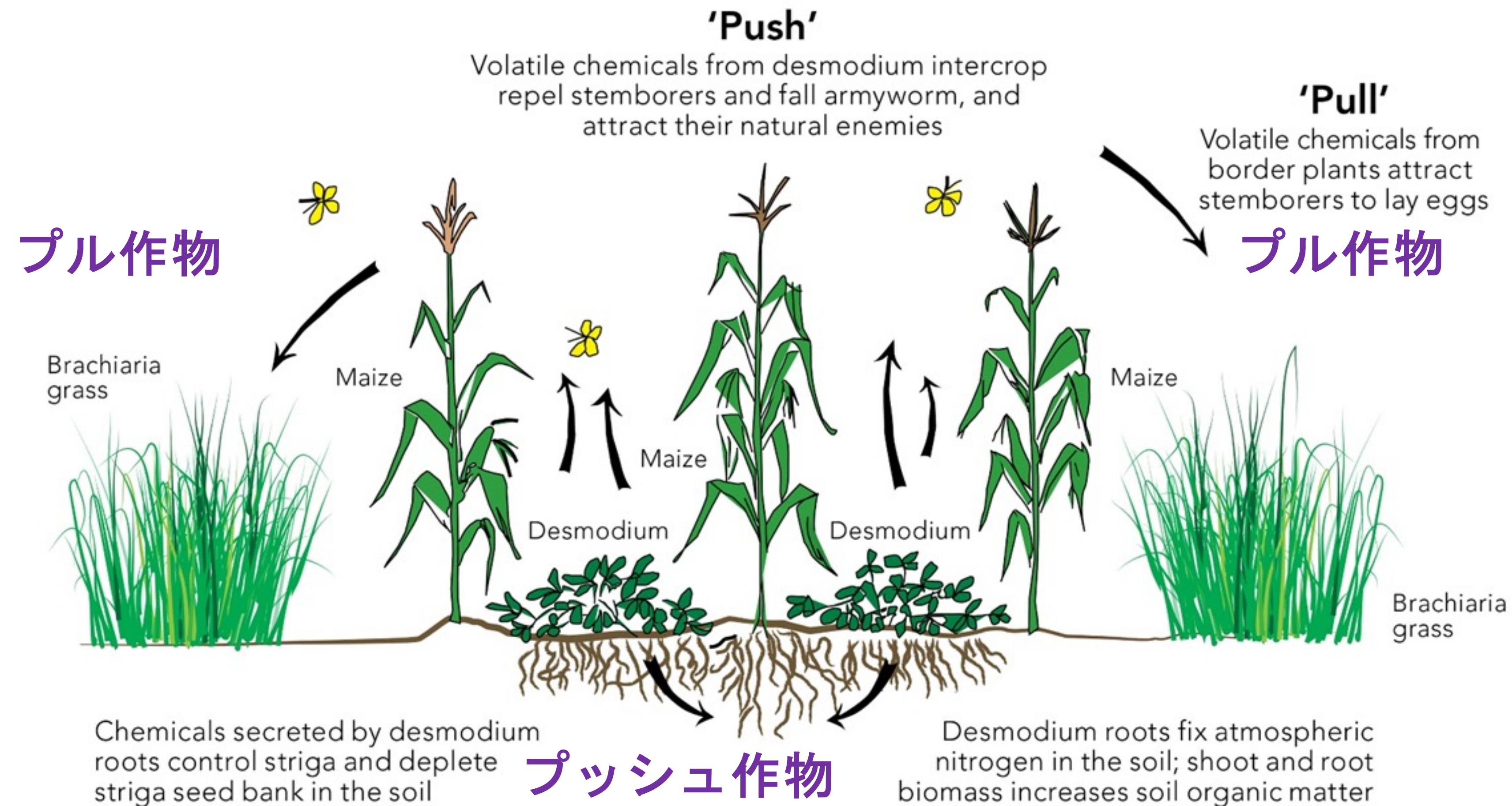
PushPull

プッシュプル



# プッシュプル農法

プッシュプル農法とは、害虫を寄せ付けないプッシュ作物と、害虫を引き寄せるプル作物をペアで栽培することです。化学薬品をつかわずに害虫被害を抑制する農法。













# 農法別の収穫量係数

農法には「**単作**」「**間作**」「**プッシュプル農法**」の3種類があります。単作とは、一つの作物だけを栽培する方法です。単作では、病気や害虫の影響を受けやすく、収穫量全体に影響を及ぼします。間作とは、収穫量を 増やすために2つの異なる作物を選択することです。組合せによって、両方の収穫量が増えたり、逆に収穫量が減ったりする場合があります。プッシュプル農法とは、害虫を寄せ付けないプッシュ作物と、害虫を引き寄せるプル作物をペアで栽培することです。

注) 以下は**大豆を単作したときの収穫量を「1」としたときの比較係数**であり、**単位はヘクタールではありません**。

	 小麦	 大豆	 トウモロコシ	 プッシュプル
小麦 	2	2.4	4	4
大豆 	2.4	1	2	1
トウモロコシ 	4	2	4	5
プッシュプル 	4	1	5	--

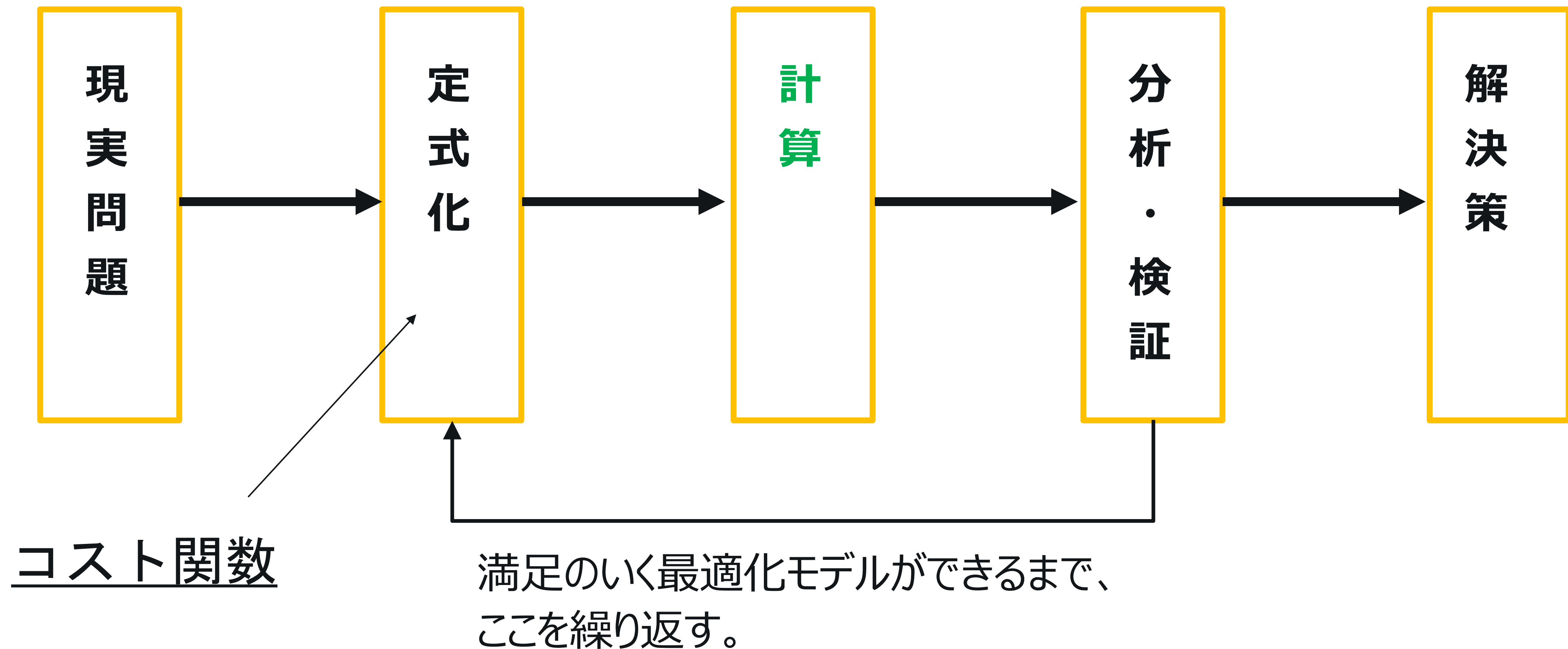
単作

間作

プッシュプル

# 計算（2次計画問題を特く）

定式化後の計算は「**2次計画問題を量子コンピュータで解くこと**」を意味します。





# コスト関数の特徴

コスト関数： **1 次式** + **2 次式**  
(線形式)

例)

変数が1つの場合：**2** $x$  + **4** + **1** $x^2$

変数が2つの場合：**2** $x_1$  + **1** $x_2$  + **4** $x_1^2$  + **1** $x_1x_2$

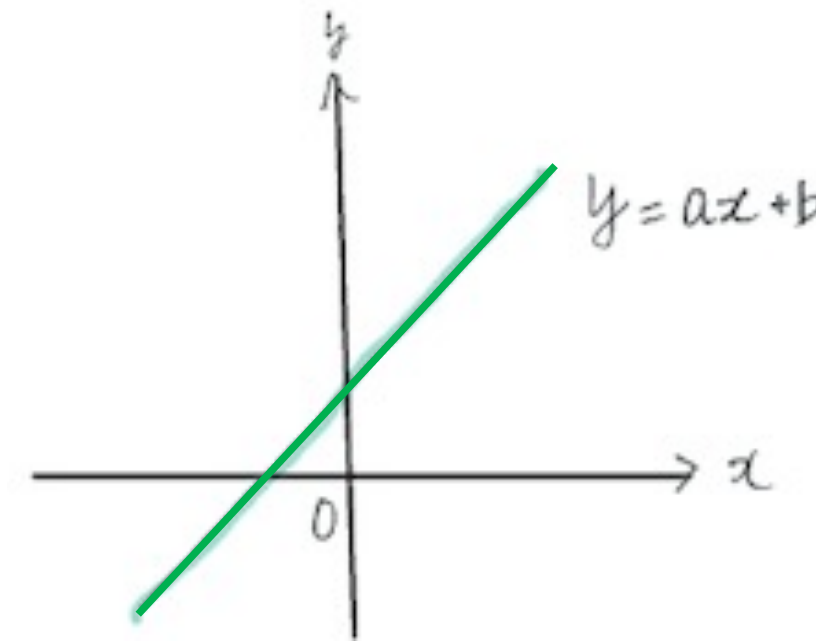
変数が3つの場合：**2** $x_1$  + **1** $x_2$  + **4** $x_3$  + **2** $x_1x_2$  + **4** $x_1x_3$  + **1** $x_2x_3$

変数の前についている係数は、各変数の特徴量として、過去データから予め提供されることが多い

ある制約条件のもとで 2 次式の最小値または最大値を求める問題を

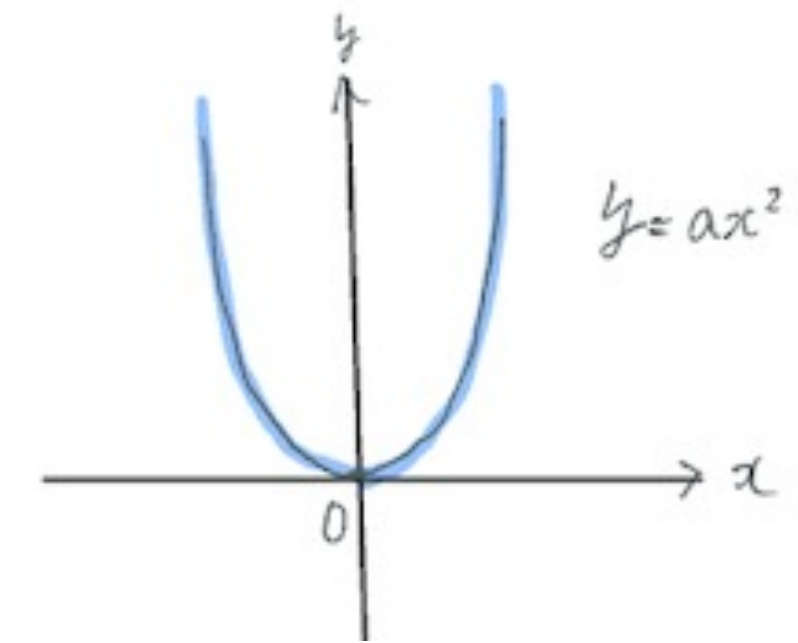
「**2 次計画問題** (quadratic program)」と呼びます。

$$y = ax + b$$



+

$$y = ax^2$$



**Qiskit Optimization**という最適化問題専用のアプリケーションモジュールは、2次計画問題作成に特化したデータ型をつくる**QuadraticProgram**クラスを提供します

- 関数宣言
- **QuadraticProgram**クラスで**収穫量 (cy)**インスタンスを生成
- 変数の追加
- 目的関数の定義
- 制約の追加

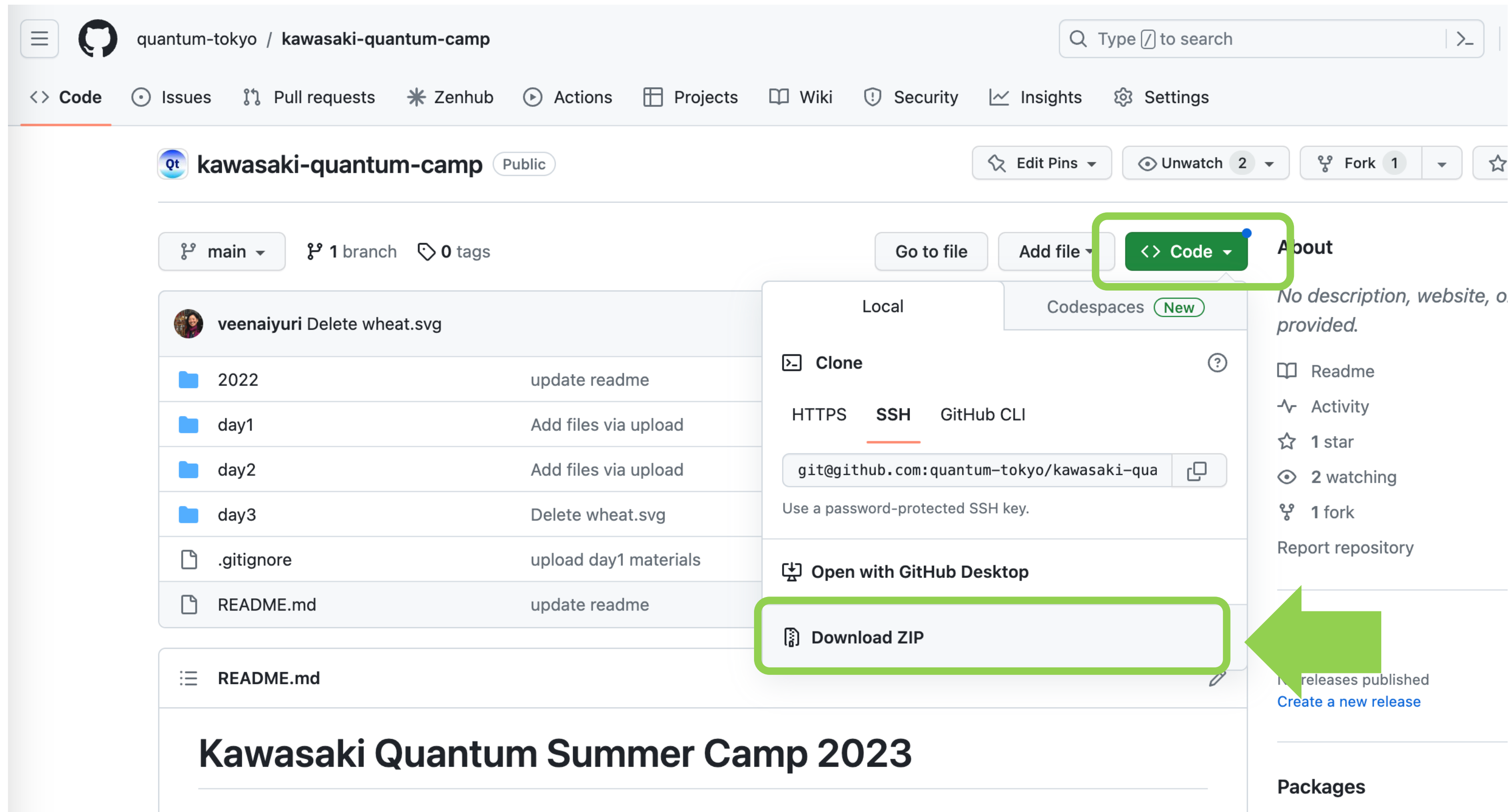
Crop Yield



<https://github.com/quantum-tokyo/kawasaki-quantum-camp>

‘optimization’フォルダの中身全部をダウンロードください

## ibm.biz/kwskgit



The screenshot shows the GitHub repository page for `quantum-tokyo / kawasaki-quantum-camp`. The repository is public and has 1 branch and 0 tags. The file list includes folders for 2022, day1, day2, and day3, along with `.gitignore` and `README.md`. The `Code` button is highlighted with a green box. The `Download ZIP` button is also highlighted with a green box, and a large green arrow points to it from the right.

**Day3フォルダのなかの  
Optimizationフォルダ下の  
ファイル식을ダウンロード**



# IBM Quantum Labの入り口



# quantum-computing.ibm.com

IBM Quantum Platform

YURI KOBAYASHI

API Token

IBM Quantum has a new navigation and application updates! [Learn more](#) →

**Recent jobs** [View all](#)

0 Pending 2333 Completed jobs

Job ID	Status	Created	Completed	Compute resource
cji1vfsul2rlcpo44gbg	Completed	About 1 hour ago	About 1 hour ago	ibmq_guadalupe
cji1v9b3smr2evnaoba0	Completed	About 1 hour ago	About 1 hour ago	ibmq_guadalupe
cji1v33iept3ghtblqkg	Completed	About 1 hour ago	About 1 hour ago	ibmq_guadalupe
cji1usjsogidtk783mk0	Completed	About 1 hour ago	About 1 hour ago	ibmq_guadalupe
cji1um27fkv7bhth1frg	Completed	About 1 hour ago	About 1 hour ago	ibmq_guadalupe

**What's new** →

- Service Alert: Upcoming updates to Open Plan access and terms (27 days ago)
- Service Alert: Update on recent job errors (4 months ago)
- Service Alert: Planned retirement ibmq\_oslo and ibmq\_geneva (4 months ago)
- Service Alert: Temporary service deterioration (5 months ago)
- Service Alert: All 7Q systems have now been made broadly accessible (6 months ago)
- Quantum News: IBM Quantum Awards: Open Science Prize

**Your systems** → 20

**Documentation** [Open app](#) →

Search docs

Qiskit Runtime

**Learning** [Open app](#) →

Course **New**

Fundamentals of quantum algorithms

**Learning** [Open app](#) →

Course **New**

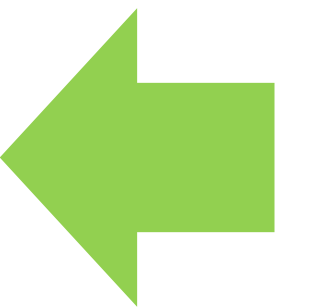
Fundamentals of quantum algorithms

IBM Quantum Composer

Graphically build circuits

**IBM Quantum Lab**

Develop quantum experiments



# 定式化（農地の収穫量）

目的関数（最大化）

maximize

$$2 \times \text{orange block} + \text{green block} + 4 \times \text{yellow block}$$

$$+ 2.4 \times (\text{orange block} \times \text{green block})$$

$$+ 4 \times (\text{orange block} \times \text{yellow block})$$

$$+ 4 \times (\text{orange block} \times \text{light green block})$$

$$+ 2 \times (\text{green block} \times \text{yellow block})$$

$$+ 1 \times (\text{green block} \times \text{light green block})$$

$$+ 5 \times (\text{yellow block} \times \text{light green block})$$

1 次式のパート：  
単作の係数をあてはめていく

2 次式のパート：  
間作とプッシュアップルの係数をあてはめていく

制約条件

subject to

$$\text{orange block} + \text{green block} + \text{yellow block} + \text{light green block} \leq 3$$

$$0 \leq \text{orange block} \leq 1$$

$$0 \leq \text{green block} \leq 1$$

$$0 \leq \text{yellow block} \leq 1$$

$$0 \leq \text{light green block} \leq 1$$

(1) 選べる作物の品種は3種類以下

(2) 各作物は0 haまたは1 ha作付け可能



# 演習：定式化

## #目的変数の定義

全角が入らないよう注意しましょう

```
cy.maximize(  
    linear={"麦":2 , "大豆":1 , "トウモロコシ":4 },  
    quadratic={"(麦, 大豆)":2.4 , ("麦, " トウモロコシ "):4 , ("麦, "プッ  
シユプル)":4 ,("大豆, "トウモロコシ)":2 , ("大豆, "プッシュユプル)":1 , ("トウモロ  
コシ, "プッシュユプル)":5 },
```

## #制約の追加

```
cy.linear_constraint(linear={"麦": 1, "大豆": 1,"トウモロコシ": 1, "プッシュ  
プル": 1}, sense="<=", rhs=3)
```

# まとめ

最適化問題は、**意志決定の手段**である

多くの組合せの中から最適な解を見つける問題は、データ量が爆発的に膨れ上がり  
(**組み合わせ爆発**)、ふつうのコンピュータでは**計算が困難**

量子コンピュータは**分子の基底エネルギーを求めるときと同じアルゴリズム**(VQE, QAOA)を活用して最適化問題を解くことができる

計算するためには、**現実問題を定式化**することが必要で、与えられた制約条件の下、ある**コスト関数**（目的関数）を**最大**または**最小**にすることが求められる

目的関数と制約条件さえ定義できれば、量子コンピュータでの計算が可能な変換をしてくれるライブラリを活用して問題を解くことができる



# Thank you

Qiskit Optimization ドキュメンテーション（日本語訳）

[https://qiskit.org/documentation/optimization/locale/ja\\_JP/index.html](https://qiskit.org/documentation/optimization/locale/ja_JP/index.html)

Qiskit Optimization (Qiskit YouTube チャンネル)

<https://youtu.be/claoY57eVIc>

Qiskit Tutorials Optimization編（Quantum Tokyo YouTube チャンネル）

<https://youtu.be/bhA86ayQpPk>

# 遊園地のアトラクションを巡る 最適なルートに関する論文

## 東京ディズニーランドの 最適巡回路

<http://www.st.nanzan-u.ac.jp/info/gr-thesis/2014/11se134.pdf>

### 東京ディズニーランドの最適巡回路

2011SE134 小石愛子 2011SE288 山辺有紗

指導教員：佐々木美裕

#### 1 はじめに

東京ディズニーランドは日本で1番年間来場者数が多いテーマパークである[1]。夏休みなど長期休暇のある時期は特に混雑し、待ち時間が400分以上となるアトラクションもある。このように混雑する場合、乗りたいアトラクションやショーを効率的に回らなければ、一日では広いディズニーランド内を回りきることはできない。そこで、東京ディズニーランドにあるアトラクションを効率的にめぐする方法がないか研究したいと考えた。

#### 2 本研究の目的

本研究の目的を2つ挙げる。第1の研究目的は個人の満足度が最大となるように東京ディズニーランドを開園から閉園まで1日かけて巡回する最適巡回路を求めることである。

第2の研究目的は、東京ディズニーランドの最適巡回路を表示させるiPhoneアプリケーション作成を行い、本研究が実際に東京ディズニーランドで利用できるような実用的なものとするところである。

#### 3 モデルの説明

このモデルを考える上で、次の仮定を設ける。

- 開園 8:30 から閉園 21:30 までの 780 分間、東京ディズニーランドのパーク内を巡回する。
- パーク内で過ごす時間はアトラクション間の移動時間、各アトラクションでの待ち時間・所要時間の合計で表す。
- 対象人数は1人とする。
- 園内を歩く速度は分速 80 メートルとする。
- 開園と同時に入場ゲートを出発し、巡回後の満足度の合計が最大となるように、いくつかのアトラクションを巡り、閉園時間までに入場ゲートに戻る巡回路を求める。
- このモデルにおいて、昼食、夕食の時間を考慮しない。
- 入口、出口はアトラクションでないので満足度、待ち時間、乗車時間はともに0とする。
- 東京ディズニーランドおすすめアトラクションランキング[8]をもとに人気順位19位までのアトラクションを研究対象とする。  
19のアトラクション、入口、出口に以下のように番号をつける。

- ①入口
- ②シンデレラ城
- ③カリブの海賊

- ④ジャングルクルーズ
- ⑤ウェスタンリバー鉄道
- ⑥ビックサンダーマウンテン
- ⑦蒸気船
- ⑧ビーバーブラザーズ
- ⑨スプラッシュマウンテン
- ⑩ホーンテッドマンション
- ⑪アリスのティーパーティー
- ⑫イツスモールワールド
- ⑬フィルハーマジック
- ⑭プーさんのハニーハント
- ⑮ロジャーラビット
- ⑯ミッキーの家
- ⑰ミニーの家
- ⑱バズライトイヤー
- ⑲スペースマウンテン
- ⑳モンスターズインク
- ㉑出口

19のアトラクションと入口、出口、それぞれの番号を地図上に示すと図1のようになる。



図1 各アトラクション番号と位置

- 東京ディズニーランド公式ホームページ[6]載されている東京ディズニーランドの各アトラクションの乗車時間と図1のアトラクション番号との対照を表1にまとめた。

#### 4 定式化

以下に記号を定義する。

$I$ :アトラクションの集合。