Qiskit Document Tutorials 勉強会
アルゴリズム編 - QAOA

2021年7月13日

IBM Quantum インターン
東京大学工学部物理工学科
中筋 渉太 (Nakasuji, Shota)

IBM Certified Associate Developer - Quantum...
IBM Professional Certification

IBM Quantum Challenge 2021 Achievemen...
IBM

Qiskit Localization Contributor - Platinum...
IBM

IBM Quantum Challenge - Fall 2020 - Advanced
IBM

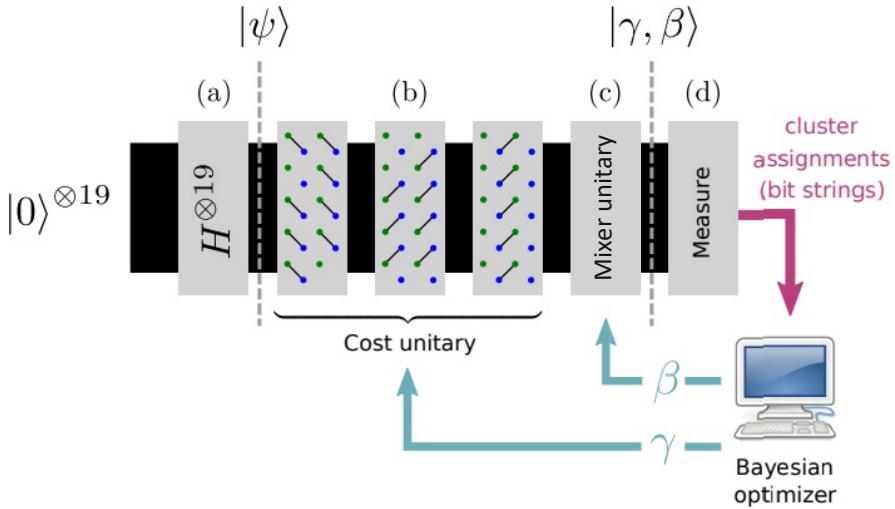
はじめに

本日は Qiskit Tutorials アルゴリズム編より QAOA についてご紹介します。

- ただし現行の日本語版では Deprecated となった Aquaでコードが記述されています。
  - https://qiskit.org/documentation/stable/0.26/locale/ja_JP/tutorials/algorithms/06_qaoa.html
- そのため本発表において紹介するコードは，最新の元版 (英語) を参照しています。
  - https://qiskit.org/documentation/tutorials/algorithms/05_qaoa.html
- やや踏み込んで Qiskit Textbook 4.1.3 でカバーされている内容も一部ご紹介します。
  - https://qiskit.org/textbook/ja/ch-applications/qaoa.html

QAOAとは

- Quantum Approximate Optimization Algorithm
- 2014 年に Farhi らによって提案された NISQ アルゴリズム
  - Farhi, Edward, Jeffrey Goldstone, and Sam Gutmann. "A quantum approximate optimization algorithm." *arXiv preprint arXiv:1411.4028* (2014).
- VQE の枠組みにおいて組合せ最適化問題を近似的に解く

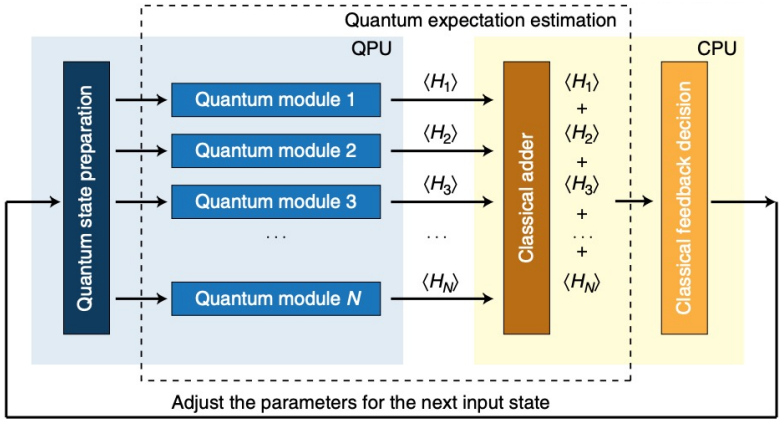


QAOAの概略図
- 左上：量子パート
- 右下：古典パート

Otterbach, J. S., et al., *arXiv preprint arXiv:1712.05771* (2017). より引用し一部改変

VQE の復習

- 量子古典ハイブリッドアルゴリズムの代表格
- 変分原理に基づき，以下を繰り返してハミルトニアンの固有値を求める
  - 量子パート：パラメータ回路 (ansatz) に対してハミルトニアンの期待値を求める
  - 古典パート：期待値がより小さくなるようにパラメータを更新する (最適化計算)
- QAOAでは，この ansatz を組合せ最適化問題用に特化した形で構築する



VQEの概略図
- 青領域：量子パート
- 黄領域：古典パート

Peruzzo, Alberto, et al., Nature communications 5.1 (2014): 1-7.

組合せ最適化問題

- 解が離散的である最適化問題
  - 例：巡回セールスマン問題
- 様々な実用例が存在している
  - 交通流最適化
  - スケジューリング最適化
- 非常に難しい問題が含まれる
  - 厳密解を現実的な時間で得られない
  - 近似解を得る方法が研究されている
  - QAOA もそうした手法の一つである

巡回セールスマン問題のイメージ

出典：Wikipedia

QAOA

問題設定：コスト関数 C を最小化する最適化問題

事前準備：コスト関数 C を以下の形に落とし込む

$$C \to H_{cost} = -\sum_i h_i Z_i - \sum_{i<j} J_{ij} Z_i Z_j$$

手順：

a. $|0\rangle^{\otimes n}$に $H^{\otimes n}$ をかけ，初期状態を用意する

$$|\psi\rangle = H^{\otimes n}|0\rangle^{\otimes n} = |+\rangle^{\otimes n}$$

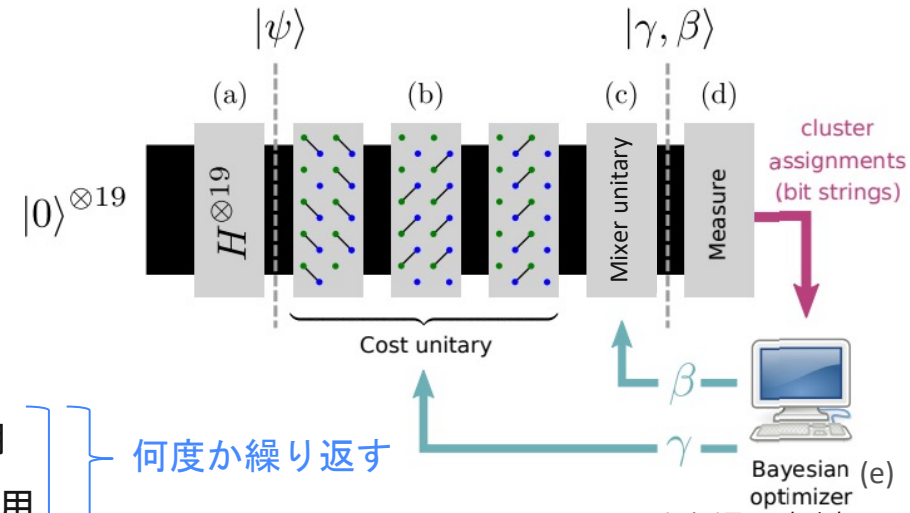
b. γ でパラメタ付けされた “cost unitary” を作用

c. β でパラメタ付けされた “mixer unitary” を作用

$$|\psi\rangle \to |\gamma, \beta\rangle$$

d. 測定を行って，H_{cost} の期待値を評価する

e. 期待値が小さくなるように γ, β を更新する



何度か繰り返す

収束するまで繰り返す

Otterbach, J. S., et al., *arXiv preprint arXiv:1712.05771* (2017). より引用し一部改変

QAOA

疑問

1. なぜこの方法で上手くいくのか？
2. “cost unitary”, “mixer unitary”とは？

実際

- 問題の定式化さえできれば OK
- 中身が分からなくても実行は可能

```
optimizer = COBYLA()
qaoa = QAOA(optimizer, quantum_instance=Aer.get_backend('statevector_simulator'))

result = qaoa.compute_minimum_eigenvalue(qubit_op)

x = sample_most_likely(result.eigenstate)
```

理論的背景についても簡単にご紹介します

断熱量子計算

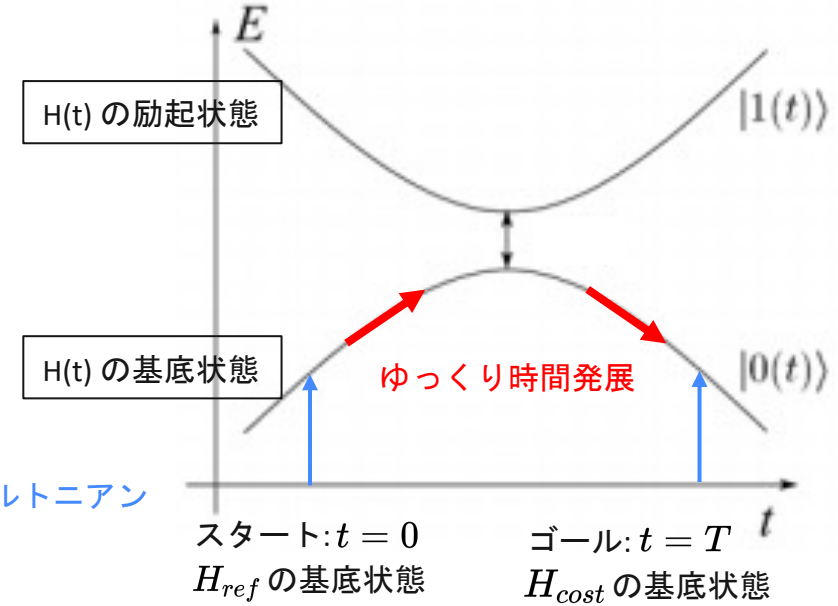
- QAOAの背景となっている計算モデル
- 量子断熱定理に基づいて計算を行う
  - 以下のようなハミルトニアンを考える

自明な基底状態をもつハミルトニアン

$$H(t) = \left(1 - \frac{t}{T}\right) H_{ref} + \frac{t}{T} H_{cost}$$

求めたい解を基底状態にもつハミルトニアン

  - $t = 0$ から $t = T$ までの時間発展を考える



大関真之, 西森秀稔. 本物理学会誌 66.4 (2011).
より引用し一部改変

- H_{ref} の自明な基底状態から十分ゆっくり時間発展させると，基底状態を辿り，H_{cost} の基底状態に達する (量子断熱定理)

断熱量子計算からQAOAへ

- 断熱量子計算は，原理的に量子回路で実行可能
  - $H(t)$ による時間発展をゲートで実装すればよい
  - “十分ゆっくり” の実現には大量のゲートが必要
    → NISQデバイスでは，実現が困難

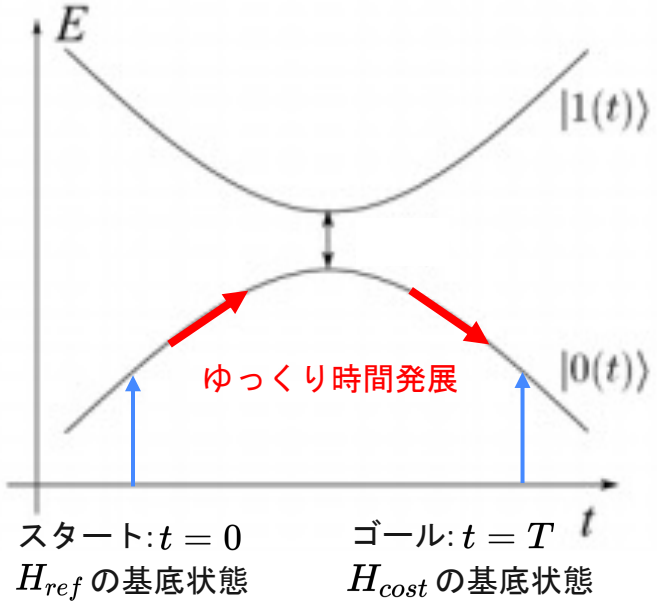
- NISQ でも実行できるよう修正されたのが QAOA
  - 上記の時間発展を Trotter 公式で近似的に展開する

$$U(t) = e^{-iH(t)t} = e^{-i((1-\frac{t}{T})H_{ref}+\frac{t}{T}H_{cost})t}$$

Trotter公式

$$e^{A+B} = \lim_{p\to\infty}\left(e^{A/p}e^{B/p}\right)^p$$

$$\approx \left(e^{-i\frac{1}{p}(1-\frac{t}{T})tH_{ref}}e^{-i\frac{1}{p}\frac{t}{T}tH_{cost}}\right)^p$$



大関真之, 西森秀稔. 本物理学会誌 66.4 (2011).
より引用し一部改変

断熱量子計算からQAOAへ

- Trotter 公式により得られた結果を整理する

$$U(t) \approx \left(e^{-i\frac{1}{p}\left(1-\frac{t}{T}\right)tH_{ref}} e^{-i\frac{1}{p}\frac{t}{T}tH_{cost}} \right)^p$$

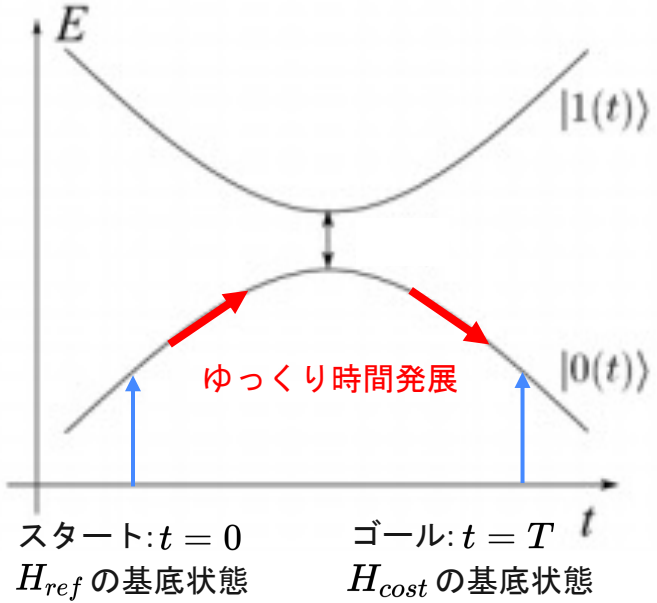
βと置く　　γと置く

$$= e^{-i\beta_p H_{ref}} e^{-i\gamma_p H_{cost}} \ldots e^{-i\beta_0 H_{ref}} e^{-i\gamma_0 H_{cost}}$$

- ここで以下のようにユニタリ演算子を置く
  - cost unitary : $U_C(\gamma_i) = e^{-i\gamma_i H_{cost}}$
  - mixer unitary : $U_M(\beta_i) = e^{-i\beta_i H_{ref}}$

- βとγを変分パラメータとする以下の形に帰着

$$U(\beta,\gamma) = U_M(\beta_p)U_C(\gamma_p)\ldots U_M(\beta_0)U_C(\gamma_0)$$



大関真之, 西森秀稔. 本物理学会誌 66.4 (2011).
より引用し一部改変

断熱量子計算からQAOAへ

- 流れを再確認

断熱量子計算

$$H(t) = \left(1 - \frac{t}{T}\right) H_{ref} + \frac{t}{T} H_{cost}$$

$$U(t) = e^{-iH(t)t} = e^{-i((1-\frac{t}{T})H_{ref} + \frac{t}{T}H_{cost})t}$$

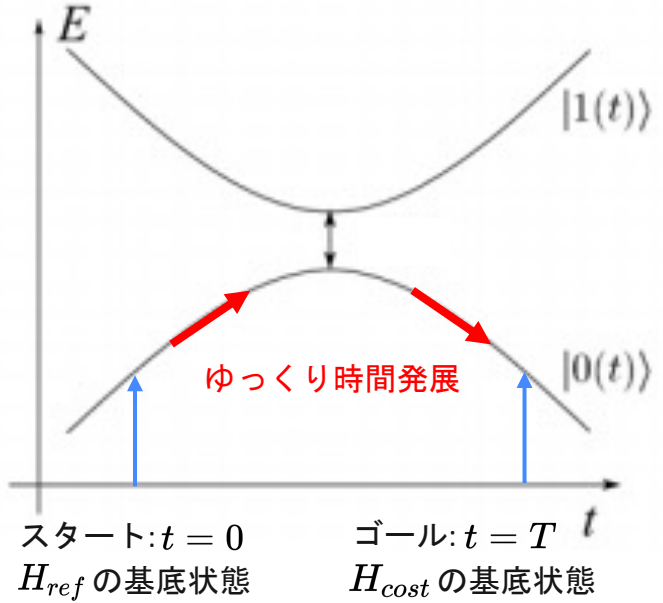
↓ Trotter 公式

$$U(t) \approx \left(e^{-i\frac{1}{p}(1-\frac{t}{T})tH_{ref}} e^{-i\frac{1}{p}\frac{t}{T}tH_{cost}}\right)^p$$

↓ 変分パラメータβ, γの導入

QAOA

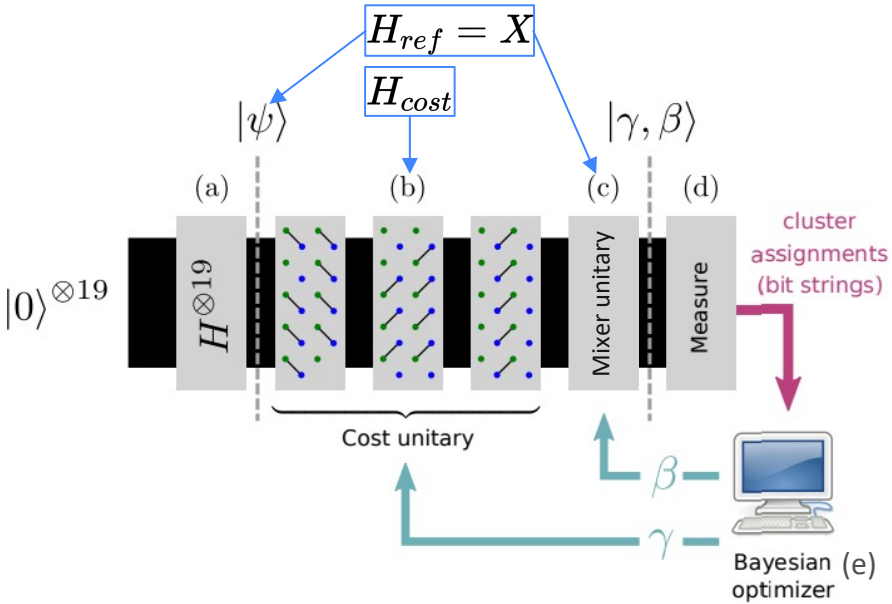
$$U(\beta, \gamma) = U_M(\beta_p) U_C(\gamma_p) \ldots U_M(\beta_0) U_C(\gamma_0)$$



大関真之, 西森秀稔. 本物理学会誌 66.4 (2011).
より引用し一部改変

QAOA再訪

- ここまでの議論と実際の手順を対応付ける
  - 発想
    - 断熱量子計算を近似的にシミュレート
  - ハミルトニアン
    - H_{ref} : X など自明な基底状態をもつもの
    - H_{cost} : コスト関数を落とし込んだもの
  - 初期状態
    - H_{ref} の自明な基底状態 (X ならば $|+\rangle^{\otimes n}$)
  - ansatz
    - cost unitary : $U_C(\gamma_i) = e^{-i\gamma_i H_{cost}}$
    - mixer unitary : $U_M(\beta_i) = e^{-i\beta_i H_{ref}}$
  - 変分パラメータ
    - $\{\beta_0, \beta_1, \ldots, \beta_p\}, \{\gamma_0, \gamma_1, \ldots, \gamma_p\}$



Otterbach, J. S., et al., *arXiv preprint arXiv:1712.05771* (2017). より引用し一部改変

QAOA再訪

問題設定：コスト関数 C を最小化する最適化問題

事前準備：コスト関数 C を以下の形に落とし込む

$$C \to H_{cost} = -\sum_i h_i Z_i - \sum_{i<j} J_{ij} Z_i Z_j$$

手順：

a. $|0\rangle^{\otimes n}$に $H^{\otimes n}$ をかけ，初期状態を用意する

$$|\psi\rangle = H^{\otimes n}|0\rangle^{\otimes n} = |+\rangle^{\otimes n}$$

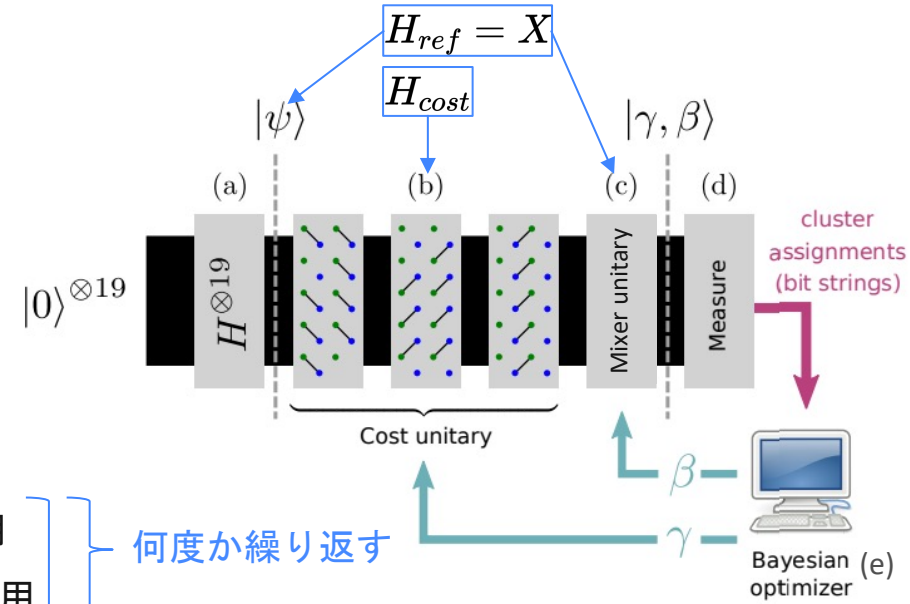
b. γ でパラメタ付けされた “cost unitary” を作用

c. β でパラメタ付けされた “mixer unitary” を作用

$$|\psi\rangle \to |\gamma, \beta\rangle$$

d. 測定を行って，H_{cost} の期待値を評価する

e. 期待値が小さくなるように γ, β を更新する



何度か繰り返す

収束するまで繰り返す

Otterbach, J. S., et al., *arXiv preprint arXiv:1712.05771* (2017). より引用し一部改変

QAOAに量子優位性はあるか

- 提案時には，特定の問題で既存の古典手法より高性能だと主張されていた
  - ここでの”性能”とは近似比のことで，厳密解に近似解がどれだけ近いかを表す
  - Farhi, Edward; Goldstone, Jeffrey; Gutmann, Sam (2014). "A Quantum Approximate Optimization Algorithm Applied to a Bounded Occurrence Constraint Problem". arXiv:1412.6062
- しかしその1年後には，より高い性能を示す古典アルゴリズムが提案された
  - Barak, Boaz; Moitra, Ankur; O'Donnell, Ryan; Raghavendra, Prasad; Regev, Oded; Steurer, David; Trevisan, Luca; Vijayaraghavan, Aravindan; Witmer, David; Wright, John (2015). "Beating the random assignment on constraint satisfaction problems of bounded degree". arXiv:1505.03424
- 現在は優位性の有無について，理論と実験の両面で研究が進められている

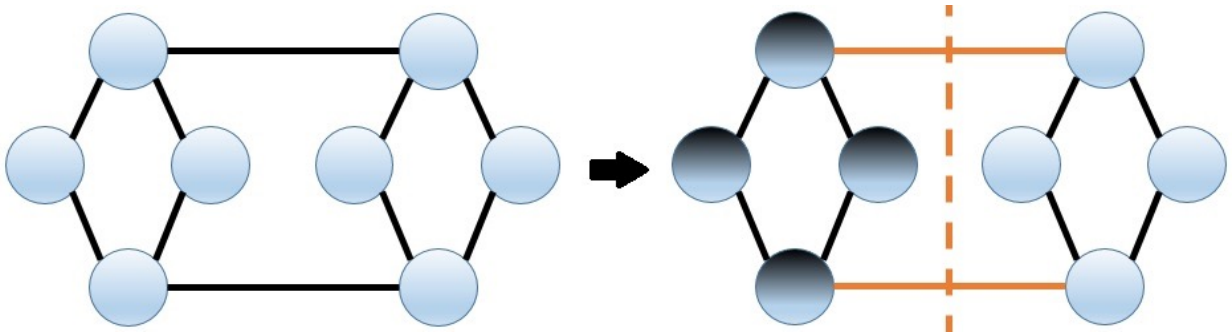
QAOA理論まとめ

QAOA とは

- VQE の観点から
  - VQE で組合せ最適化問題を近似的に解くアルゴリズム
  - 組合せ最適化問題に特化させた ansatz を採用している
- 断熱量子計算の観点から
  - 断熱量子計算を量子回路で近似的にシミュレートする
  - 近似の粗さを変分的なアプローチによってカバーする

グラフ分割問題

- Qiskit Tutorials ではグラフ分割問題を QAOA で解いている
- グラフ分割問題とは：
  - グラフを二分割する辺の数 (重み) を最小化する
  - 集積回路設計やプログラム分割などに応用可能



出典： https://amplify.fixstars.com/ja/techresources/research/ising-model-formulation/graph-partitioning/

グラフ分割問題

グラフの定義と可視化

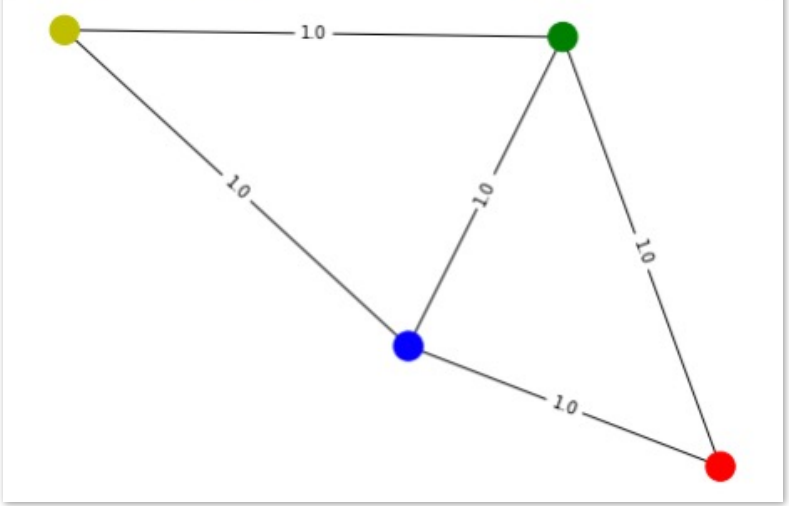
- 英語版と日本語版の違いに注意
  - 英語版 (右) : 辺重みが全て1
  - 日本語版 : 辺重みがランダム

[2]:

```
num_nodes = 4
w = np.array([[0., 1., 1., 0.],
              [1., 0., 1., 1.],
              [1., 1., 0., 1.],
              [0., 1., 1., 0.]])
G = nx.from_numpy_matrix(w)
```

[3]:

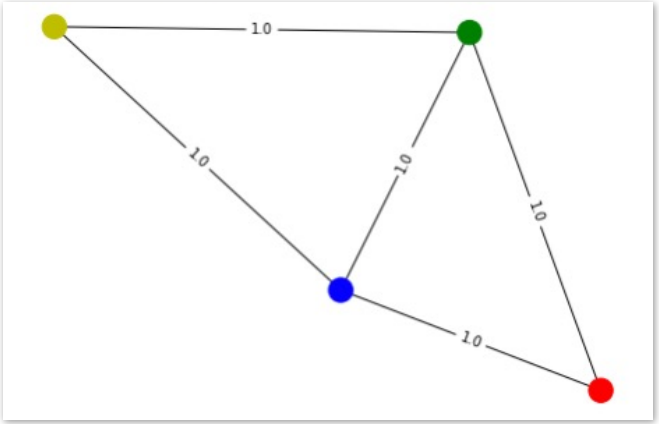
```
layout = nx.random_layout(G, seed=10)
colors = ['r', 'g', 'b', 'y']
nx.draw(G, layout, node_color=colors)
labels = nx.get_edge_attributes(G, 'weight')
nx.draw_networkx_edge_labels(G, pos=layout, edge_labels=labels);
```



グラフ分割問題

総当たり法による求解

- 総頂点数4より2頂点ずつに分割
- 可能な値は3, 4なので最適値は3



[4]:
```
def objective_value(x, w):
    X = np.outer(x, (1 - x))
    w_01 = np.where(w != 0, 1, 0)
    return np.sum(w_01 * X)

def brute_force():
    # use the brute-force way to generate the oracle
    def bitfield(n, L):
        result = np.binary_repr(n, L)
        return [int(digit) for digit in result]  # [2:] to chop off the "0b" part

    L = num_nodes
    max = 2**L
    minimal_v = np.inf
    for i in range(max):
        cur = bitfield(i, L)

        how_many_nonzero = np.count_nonzero(cur)
        if how_many_nonzero * 2 != L:  # not balanced
            continue

        cur_v = objective_value(np.array(cur), w)
        if cur_v < minimal_v:
            minimal_v = cur_v
    return minimal_v

sol = brute_force()
print(f'Objective value computed by the brute-force method is {sol}')
```

```
Objective value computed by the brute-force method is 3
```

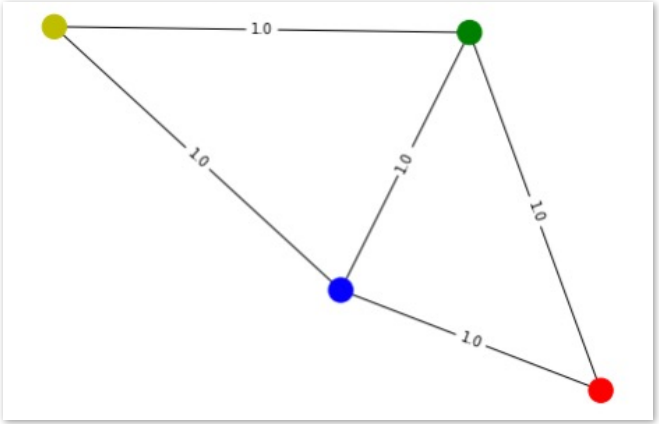
3. 実験

グラフ分割問題

コストハミルトニアンの実装

- コスト関数を以下の形式に定式化

$$H_{cost} = -\sum_i h_i Z_i - \sum_{i<j} J_{ij} Z_i Z_j$$




```
[5]: from qiskit.quantum_info import Pauli
     from qiskit.opflow import PauliSumOp

     def get_operator(weight_matrix):
         r"""Generate Hamiltonian for the graph partitioning
         Notes:
             Goals:
                 1 separate the vertices into two set of the same size
                 2 make sure the number of edges between the two set is minimized.
             Hamiltonian:
                 H = H_A + H_B
                 H_A = sum\_{(i,j)\in E}{(1-ZiZj)/2}
                 H_B = (sum_{i}{Zi})^2 = sum_{i}{Zi^2}+sum_{i!=j}{ZiZj}
                 H_A is for achieving goal 2 and H_B is for achieving goal 1.
         Args:
             weight_matrix (numpy.ndarray) : adjacency matrix.
         Returns:
             PauliSumOp: operator for the Hamiltonian
             float: a constant shift for the obj function.
         """
         num_nodes = len(weight_matrix)
         pauli_list = []
         shift = 0

         for i in range(num_nodes):
             for j in range(i):
                 if weight_matrix[i, j] != 0:
                     x_p = np.zeros(num_nodes, dtype=bool)
                     z_p = np.zeros(num_nodes, dtype=bool)
                     z_p[i] = True
                     z_p[j] = True
                     pauli_list.append([-0.5, Pauli((z_p, x_p))])
                     shift += 0.5

         for i in range(num_nodes):
             for j in range(num_nodes):
                 if i != j:
                     x_p = np.zeros(num_nodes, dtype=bool)
                     z_p = np.zeros(num_nodes, dtype=bool)
                     z_p[i] = True
                     z_p[j] = True
                     pauli_list.append([1, Pauli((z_p, x_p))])
                 else:
                     shift += 1

         pauli_list = [(pauli[1].to_label(), pauli[0]) for pauli in pauli_list]
         return PauliSumOp.from_list(pauli_list), shift

     qubit_op, offset = get_operator(w)
```


Qt

グラフ分割問題

QAOAの検証

- qiskit.algorithms の QAOA クラスを利用
- 総当たり法と同じ結果が得られている

```
optimizer = COBYLA()
qaoa = QAOA(optimizer, quantum_instance=Aer.get_backend('statevector_simulator'))

result = qaoa.compute_minimum_eigenvalue(qubit_op)

x = sample_most_likely(result.eigenstate)

print(x)
print(f'Objective value computed by QAOA is {objective_value(x, w)}')
```

```
[1. 0. 1. 0.]
Objective value computed by QAOA is 3.0
```

まとめ：
Qiskit Tutorials Algorithm編からQAOAを紹介

- QAOAとは，VQEの枠組みで組合せ最適化問題を近似的に解く手法
- ansatzの構成においては，その背景に断熱量子計算の考え方がある
- 量子優位性については，理論と実験の両面で研究が進められている

Shota Nakasuji